Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(2) R und RStudio Grundlagen

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

Was ist R?

Eine Programmiersprache und ein Softwarepaket.

Entwickelt von Ihaka and Gentleman (1996).

Freier Dialekt der propietären Software S (Becker, Chambers, and Wilks (1988)).

Weiterentwickelt und gepflegt durch R Core Team und R Foundation

Interpretierte imperativ-objektorientierte 4GL Sprache.

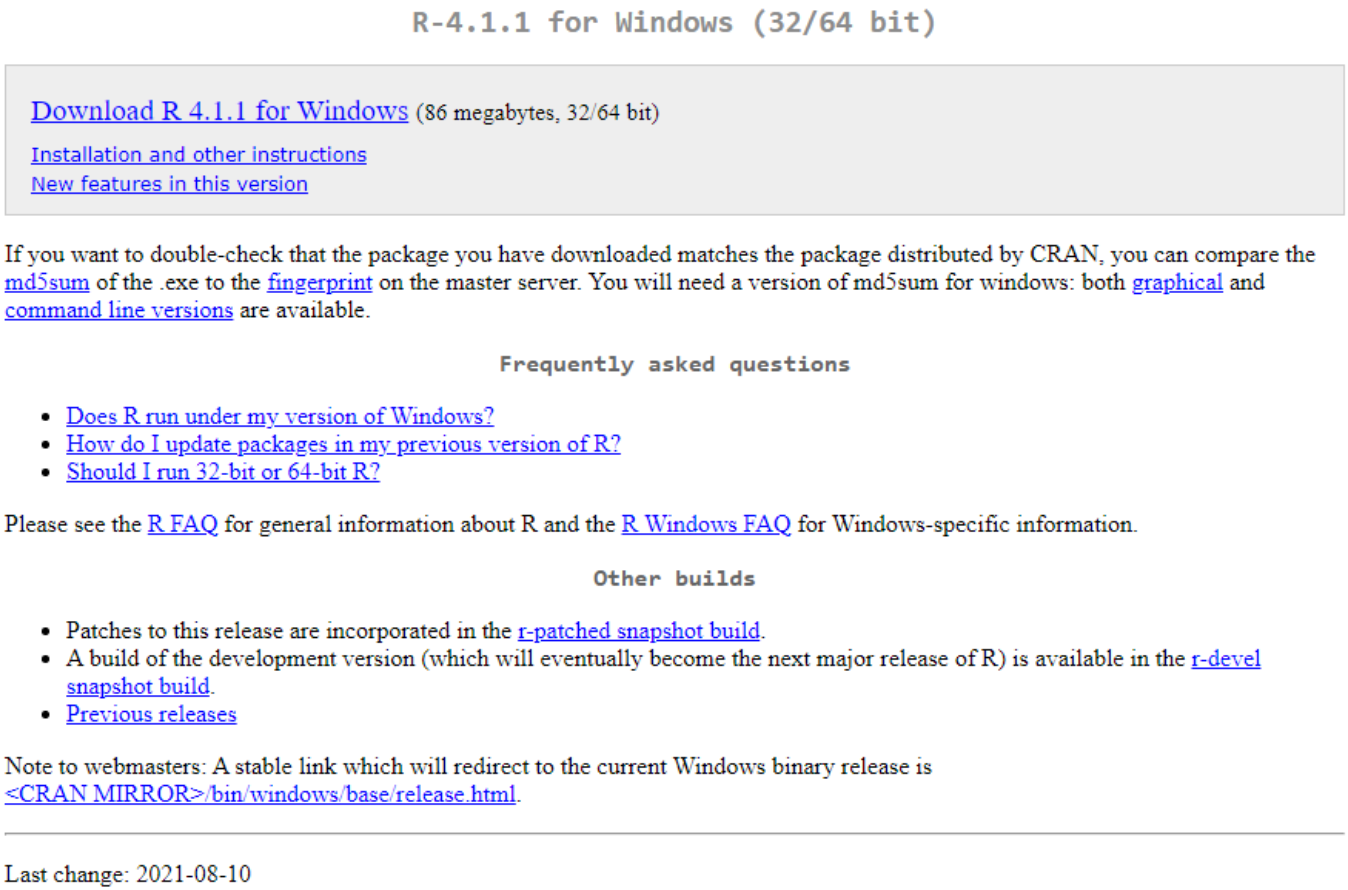
Optimiert und populär für statistische Datenanalysen.

Große Community mit etwa 20.000 beigetragenen R Paketen (Erweiterungen)

Evolviert und konservativ im Kern, konsistent und progressiv in R Paketen.

Wie bekommt man R?

Runterladen (z.B. https://cran.r-project.org/bin/windows/base/) und installieren.



R-4.1.1 for Windows (32/64 bit)

Download R 4.1.1 for Windows (86 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.html.

Last change: 2021-08-10

Was kann man mit R machen?

Datensätze laden, manipulieren, und speichern.

Eine Vielzahl von Berechnungen an verschiedenen Datenstrukturen durchführen.

Eine Vielzahl statistischer Analysemethoden auf Daten anwenden.

Datenanalyseskripte schreiben und Abbildungen generieren.

Präsentationen RMarkdown und Bücher RBookdown erstellen.

Was kann man mit R (bisher) nicht so gut machen?

In einer ansprechenden Umgebung programmieren ($\Rightarrow$ RStudio).

Scientific Computing ($\Rightarrow$ Python, Matlab, Julia).

Psychologische Experimente programmieren ($\Rightarrow$ Python, Matlab)

Wie bekommt man Hilfe zu R?

Googlen

https://stackoverflow.com/

Während der Programmierung und bei bekanntem Funktionsnamen

```
?mean
help(mean)
```

Für längere Tutorials

```
browseVignettes()
```

https://rseek.org/

https://www.rstudio.com/resources/cheatsheets/

https://www.r-bloggers.com/

Was ist RStudio?

Eine Softwareentwicklungsumgebung für R

Softwareentwicklungsumgebung = Integrated Development Environment

IDEs sind Programme zum Programmieren mit einer Programmiersprache

Kommandozeile, Skripteditor, Vielzahl weiterer Tools

Freemium Produkt von RStudio, Inc. (IDE frei, Server kostenpflichtig)

Initial Release 2011, Affero General Public License

Keine Verbindung zu R Core Team oder R Foundation

Wie bekommt man RStudio?

Runterladen (https://www.rstudio.com/products/rstudio/) und installieren

RStudio Desktop 1.4.1717 - Release Notes

1. Install R. RStudio requires R 3.0.1+.

2. Download RStudio Desktop. Recommended for your system:

DOWNLOAD RSTUDIO FOR WINDOWS
1.4.1717 | 156.18MB

Requires Windows 10 (64-bit)

All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

Was kann man mit RStudio machen?

R Skripte erzeugen, bearbeiten, und laufen lassen

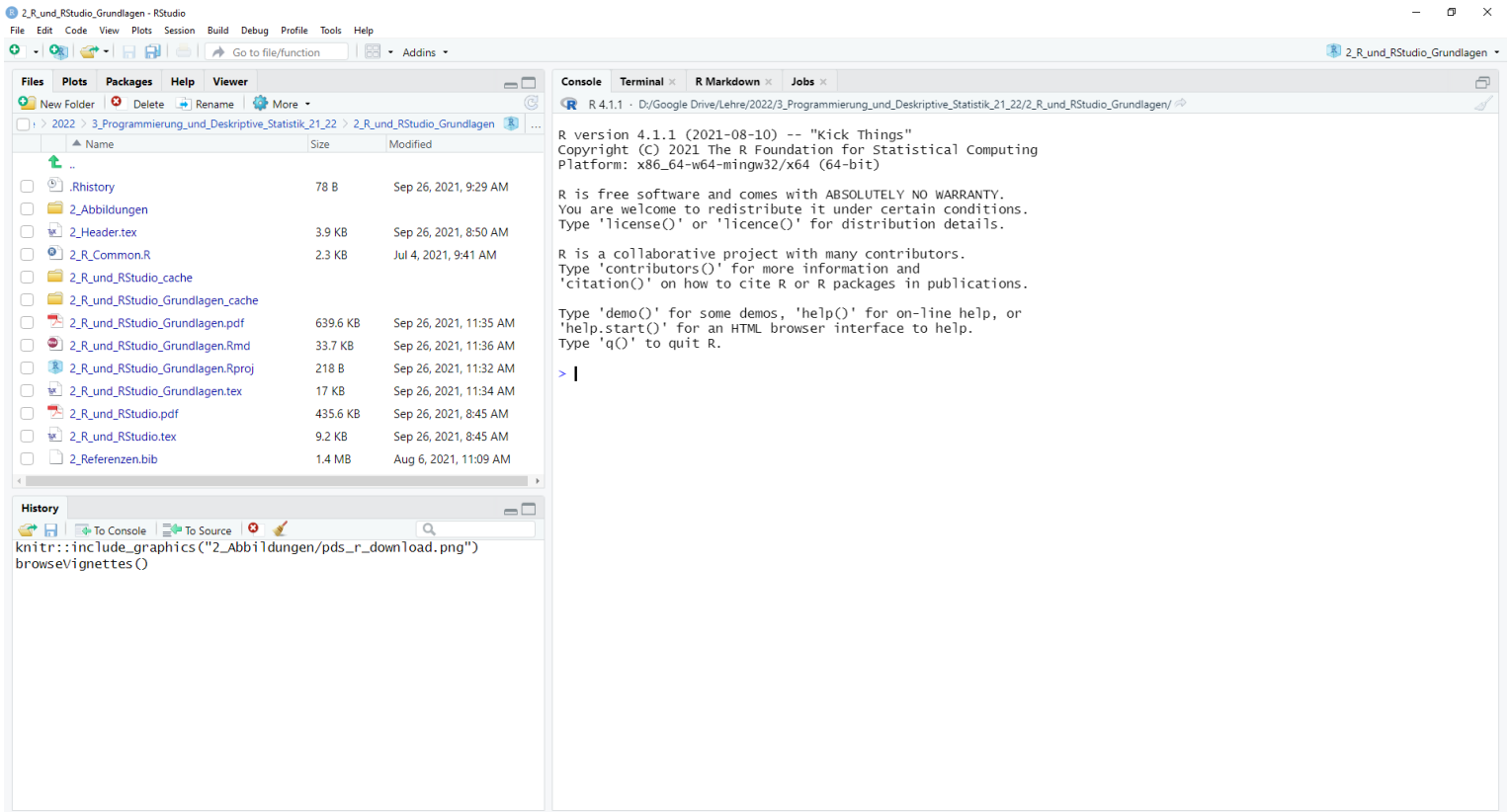
R Skripte in R Projekten organisieren

Laut Eigenwerbung

- Access RStudio locally
- Syntax highlighting, code completion, and smart indentation
- Execute R code directly from the source editor
- Quickly jump to function definitions
- View content changes in real-time with the Visual Markdown Editor
- Easily manage multiple working directories using projects
- Integrated R help and documentation
- Interactive debugger to diagnose and fix errors
- Extensive package development tools

Was kann man mit RStudio machen?

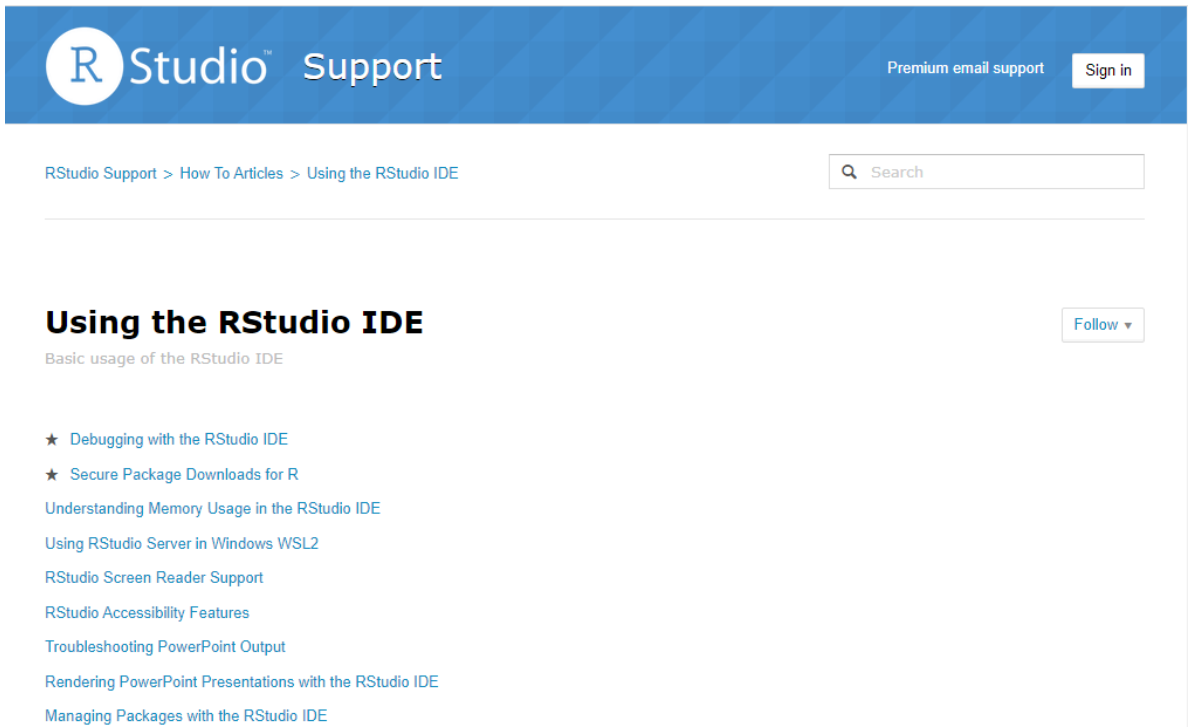
Custom Layout via Tools → Global Options . . .



Wie bekommt man Hilfe zu RStudio?

Googlen

Zur Einführung ⇒ Using the RStudio IDE



R Kommandozeile | Working in the Console

Eingabe von R Befehlen bei >

Autocomplete mit Tab

Vorherige Befehle mit Cursor ↑

Bereinigen des Konsolenoutputs mit Ctrl + L

Code Ausführungsstopp mit Esc

```
print("Hallo Welt!")
```

```
> [1] "Hallo Welt!"
```

Code-Snippets in diesen Folien immer aktiv in der Konsole nachvollziehen!

R Skripte | Executing and Editing Code

File → New File → R Script oder Ctrl + Shift + N für neue .R Datei

Open File oder Ctrl + O zum Öffnen bestehender .R Datei

Eintippen von

```
print("Hallo Welt!")  # Hinter Hashtags stehen dokumentierende Kommentare
print("Hallo R!")     # Kommentare werden nicht ausgefuehrt
```

Ausführen der einzelnen Zeile, auf welcher der Cursor ruht

⇒ Run oder Ctrl + Enter

Ausführen aller Zeilen

⇒ Source oder Ctrl + Shift + Enter oder

⇒ Tickmark bei Source on Save setzen und Ctrl + S

Code-Snippets in diesen Folien immer aktiv in einem R Skript dokumentieren!

Das R und RStudio Data Science Universum

Analyse & Explore

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying philosophy and common APIs.

Project Site Link ›

ggplot 2 is an enhanced data visualization package for R. Create stunning multi-layered graphics with ease.

Project Site Link ›

dplyr is the next iteration of plyr, focussing on only data frames. dplyr is faster and has a more consistent API.

Project GitHub Link ›

tidyr makes it easy to "tidy" your data. Tidy data is data that's easy to work with: it's easy to munge (with dplyr), visualise (with ggplot2 or ggvis) and model (with R's hundreds of modelling packages).

Project Paper Link ›

Model & Predict

TensorFlow™ is an open-source software library for Machine Intelligence. The R interface to TensorFlow lets you work productively using the high-level Keras and Estimator APIs and the core TensorFlow API.

Project Site Link ›

The tidymodels framework is a collection of packages for modeling and machine learning using tidyverse principles.

Project Site Link ›

Sparklyr provides bindings to Spark's distributed machine learning library. Together with sparklyr's dplyr interface, you can easily create and tune machine learning workflows on Spark, orchestrated entirely within R.

Project Site Link ›

Connect & Integrate

Sparklyr is an R interface to Apache Spark, a fast and general engine for big data processing. This package connects to local and remote Apache Spark clusters, a 'dplyr' compatible back-end, and an interface to Spark's ML algorithms.

Project Site Link ›

Plumber enables you to convert your existing R code into web APIs by merely adding a couple of special comments.

Project Site Link ›

The reticulate package provides a comprehensive set of tools for interoperability between Python and R.

Project Site Link ›

Communicate & Interact

Shiny makes it incredibly easy to build interactive web applications with R. Shiny has automatic "reactive" binding between inputs and outputs and extensive pre-built widgets.

Project Site Link ›

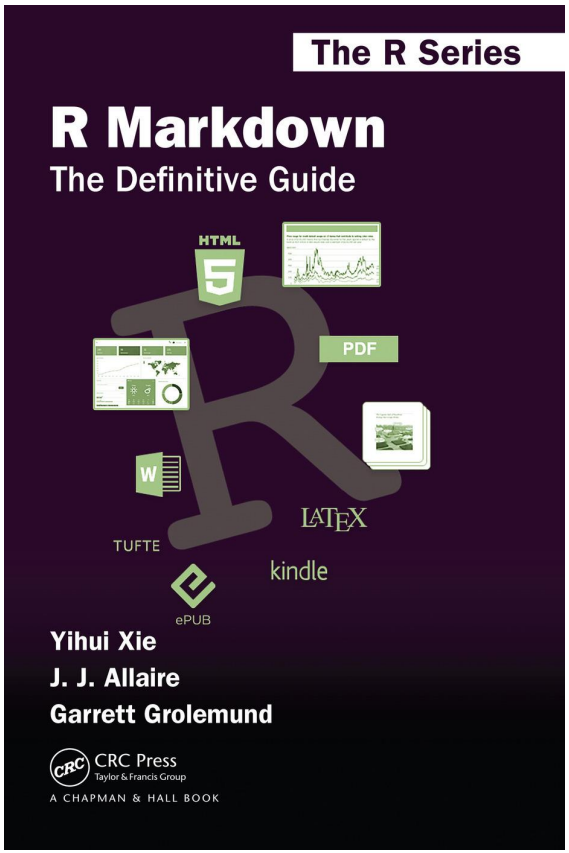
Use R Markdown to develop your code and ideas in a reproducible document. Knit plots, tables, and results together with narrative text, and create analyses ready to be shared.

Project Site Link ›

Use flexdashboard to publish groups of related data visualizations as a dashboard.

Project Site Link ›

Lehrmaterialien mit R und RStudio



R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

R Konsole als Taschenrechner

```
1+1
```

```
> [1] 2
```

```
2*3
```

```
> [1] 6
```

```
sqrt(2)
```

```
> [1] 1.41
```

```
exp(0)
```

```
> [1] 1
```

```
log(1)
```

```
> [1] 0
```

- [1] zeigt das erste und einzige Element des Ausgabevektors an
- Vektoren werden noch im Detail behandelt.

Arithmetische Operatoren

Operator	Bedeutung
$+$	Addition
$-$	Subtraktion
$*$	Multiplikation
$/$	Division
^ oder $**$	Potenz
%*%	Matrixmultiplikation
%/%	Ganzzahlige Teilung (5%/%2 = 2)
%%	Modulo (5%%2 = 1)

- Matrixmultiplikation, Modulo, ganzzahlige Teilung benötigen wir zunächst nicht.
- Ganzzahlige Teilung gibt das Resultat der ganzzahligen Teilung an.
- Modulo gibt den ganzzahligen Rest bei ganzzahliger Teilung an.

Logische Operatoren

Die Boolesche Algebra und R kennen zwei logische Werte: TRUE und FALSE

Bei Auswertung von Relationsoperatoren ergeben sich logische Werte

Relationsoperator	Bedeutung
==	Gleich
!=	Ungleich
<, >	Kleiner, Größer
<=, >=	Kleiner gleich, Größer gleich
|	ODER
&	UND

- <, <=, >, >= werden zumeist auf numerische Werte angewendet.
- ==, != werden zumeist auf beliebige Datenstrukturen angewendet.
- | und & werden zumeist auf logische Werte angewendet.
- Die Funktion xor() implementiert das exklusive ODER.

Mathematische Funktionen

Aufruf	Bedeutung
abs(x)	Betrag
sqrt(x)	Wurzel
ceiling(x)	Aufrunden (ceiling(2.7) = 3)
floor(x)	Abrunden (floor(2.7) = 2)
round(x)	Mathematisches Runden (round(2.5) = 2)
exp(x)	Exponentialfunktion
log(x)	Logarithmus Funktion

Es handelt sich um eine Auswahl, einen vollständigen Überblick gibt

```
names(methods:::.BasicFunsList)
```

R unterscheidet formal nicht zwischen Operatoren und Funktionen

Operatoren können mit der Infix Notation als Funktionen genutzt werden

```
`+`(2,3)            # Infixnotation für 2 + 3
```

```
> [1] 5
```

Operatorpräzedenz

Operatorrangfolge

Regeln der Form "Punktrechnung geht vor Strichrechnung"

Vordefinierte Operatorpräzedenz kann durch Klammern überschrieben werden

```
2 * 3 + 4
```

```
> [1] 10
```

```
2 * (3 + 4)
```

```
> [1] 14
```

Generell gilt

- Operatorrangfolge nicht raten oder folgern, sondern nachschauen!
- Lieber Klammern setzen, als keine Klammern setzen!
- Immer nachschauen, ob Berechnungen die erwarteten Ergebnisse liefern!

```
?Syntax
```

Operatorpräzedenz

Präzedenz und Ausführungsreihenfolge arithmetischer Operatoren

Operator	Reihenfolge
^	Rechts nach links
−x,+x	Unitäres Vorzeichen, links nach rechts
∗, /	Links nach Rechts
+, −	Links nach Rechts

Beispiele

```
2^2^3          # 2^(2^3)    = 2^8 = 256
(2^2)^3        # (2^2)^3    = 4^3 = 64
-1^2           # -(1^2)     = -1
(-1)^2         # (-1)^2     =  1
2+3/4*5        # 2+(3/4)*5 = 2+(0.75*5) = 2+3.75 = 5.75
2+3/(4*5)      # 2+3/(4*5) = 2+3/20 = 2+0.15 = 2.15
```

Operatorpräzedenz

Operator Syntax and Precedence

Description

Outlines R syntax and gives the precedence of operators.

Details

The following unary and binary operators are defined. They are listed in precedence groups, from highest to lowest.

`:: :::`	access variables in a namespace
`$ @`	component / slot extraction
`[ [[`	indexing
`^`	exponentiation (right to left)
`- +`	unary minus and plus
`:`	sequence operator
`%any% |>`	special operators (including `%%` and `%/%`)
`* /`	multiply, divide
`+ -`	(binary) add, subtract
`< > <= >= == !=`	ordering and comparison
`!`	negation
`& &&`	and
`| ||`	or
`~`	as in formulae
`-> ->>`	rightwards assignment
`<- <<-`	assignment (right to left)
`=`	assignment (right to left)
`?`	help (unary and binary)

Within an expression operators of equal precedence are evaluated from left to right except where indicated. (Note that `=` is not necessarily an operator.)

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

Definition

In der Programmierung ist eine Variable ein abstrakter Behälter für eine Größe, welche im Verlauf eines Rechenprozesses auftritt. Im Normalfall wird eine Variable im Quelltext durch einen Namen bezeichnet und hat eine Adresse im Speicher einer Maschine. Der durch eine Variable repräsentierte Wert kann – im Unterschied zu einer Konstante – zur Laufzeit des Rechenprozesses verändert werden.

Wikipedia

Grundlagen

Variablen sind vom Programmierenden benannte Platzhalter für Werte

In 3GL Sprachen wird der Variablentyp durch eine Initialisierungsanweisung festgelegt:

```
VAR A : INTEGER      # A ist eine Variable vom Typ Integer (ganze Zahl)
```

In 3GL Sprachen wird Variablen durch eine Zuweisungsanweisung ein Wert zugeschrieben:

```
A := 1               # Der Variable A wird der numerische Wert 1 zugewiesen
```

In 4GL Sprachen wie Matlab, Python, R werden Variablen durch Zuweisung initialisiert:

```
a = 1                # a ist eine Variable vom Typ double, ihr Wert ist 1
```

Der Zuweisungsbefehl in Matlab und Python ist =, der Zuweisungsbefehl in R ist <- oder =.

Offiziell empfohlen für R ist <-, aus Kohärenzgründen benutzen wir hier =.

Beispiel

Greta geht ins Schreibwarengeschäft und kauft vier Hefte, zwei Stifte und einen Füller. Wie viele analoge Gegenstände kauft Greta insgesamt?

```
hefte  = 4      # Definition der Variable 'hefte'  und Wertzuweisung 4
stifte = 2      # Definition der Variable 'stifte' und Wertzuweisung 2
fuller = 1      # Definition der Variable 'fuller' und Wertzuweisung 1
```

Nach Zuweisung existieren die Variablen im Arbeitsspeicher, dem sogenannten *Workspace*

Die Variablen können jetzt wie Zahlen in Berechnungen genutzt werden

```
gesamt  = hefte + stifte + fuller                    # Berechnung der Gegenstandsanzahl
print(gesamt)
```

```
> [1] 7
```

Ein Heft kostet einen Euro, ein Stift kostet zwei Euro, und ein Füller kostet 10 Euro. Wie viel Euro muss Greta insgesamt bezahlen?

```
gesamtpreis = hefte*1 + stifte*2 + fuller*10         # Berechung des Preises
print(gesamtpreis)
```

```
> [1] 18
```

`print()` gibt Variablenwerte in der R Konsole aus.

Workspace

`ls()` zeigt die existierenden benutzbaren Variablen im Arbeitsspeicher an

```
ls()                          # Anzeigen aller Variablennamen im Workspace
```

```
> [1] "error_wrap"  "fuller"      "gesamt"      "gesamtpreis"
> [5] "hefte"       "inline_hook" "stifte"
```

`rm()` erlaubt das Löschen von Variablen

```
rm(gesamtpreis)               # Löschen der Variable Gesamtpreis
ls()
```

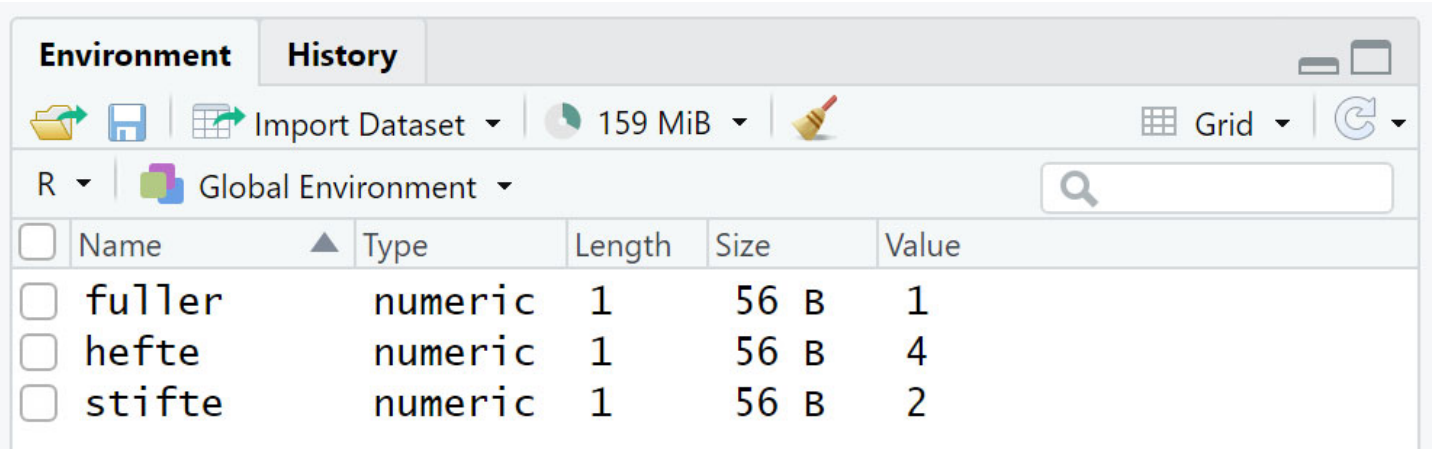
```
> [1] "error_wrap"  "fuller"      "gesamt"      "hefte"
> [5] "inline_hook" "stifte"
```

`rm(list=ls())` löscht alle Variablen

```
rm(list = ls())               # Löschen aller Variablen
ls()
```

```
> character(0)
```

Workspace


Environment
History
Import Dataset
159 MiB
Grid
R
Global Environment
Name
Type
Length
Size
Value
fuller numeric 1 56 B 1
hefte numeric 1 56 B 4
stifte numeric 1 56 B 2

Variablennamen

Zulässige Variablennamen

... bestehen aus Buchstaben, Zahlen, Punkten (.) und Unterstrichen (_)

... beginnen mit einem Buchstaben oder . nicht gefolgt von einer Zahl

... dürfen keine reserverd words wie for, if, NaN, usw. sein (>?reserved)

... werden unter ?make.names() beschrieben

Sinnvolle Variablennamen

... sind kurz ($\approx$ 1 bis 7 Zeichen) und aussagekräftig

... bestehen nur aus Kleinbuchstaben und Unterstrichen

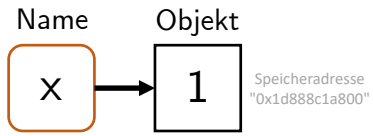
Variablenrepräsentation | Binding

```
x = 1
```

Intuitiv wird eine Variable genannt x mit dem Wert 1 erzeugt.

De-facto geschehen zwei Dinge:

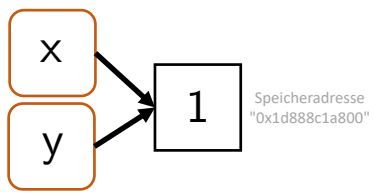
1. R erzeugt ein Objekt (Vektor mit Wert 1) mit Speicheradresse ‘lobstr::obj_addr(x).
2. R verbindet dieses Objekt mit dem Namen x, der das Objekt im Speicher referenziert.




```
y = x
```

Intuitiv wird eine Variable genannt y mit Wert gleich dem Wert von x erzeugt.

De-facto wird ein neuer Name y erzeugt, der dasselbe Objekt referenziert wie x:



Man überzeuge sich mit `lobstr::obj_addr(x)` und ‘lobstr::obj_addr(y).

Das Objekt (Vektor mit Wert 1) wird nicht kopiert, R spart Arbeitsspeicher.

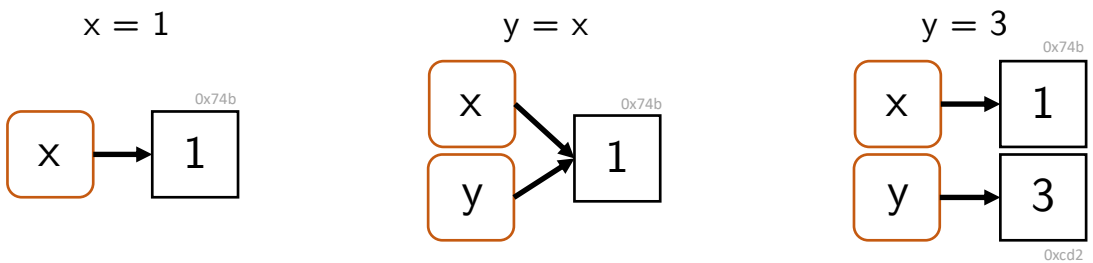
Variablenrepräsentation | Copy-on-modify

```
x = 1         # Objekt (0x74b) erzeugt, x referenziert Speicheradresse  des Objektes
y = x         # y referenziert dieselbe Speicheradresse wie x (0x74b)
y = 3         # y modifiziert, modifizierte Kopie (0xcd2) wird gespeichert
y             # y referenziert jetzt (0xcd2)
```

```
> [1] 3
```

```
x             # x referenziert weiterhin (0x74b)
```

```
> [1] 1
```



R Objekte sind *immutable*, können also nicht verändert werden.

Variablenrepräsentation | Copy-on-modify

Zur Immutability gibt allerdings zwei Ausnahmen, genannt *Modifications-in-place*

1. Objekte mit nur einem gebundenem Namen werden in-place modifiziert
   - Dieses Verhalten ist allerdings nur in R, nicht innerhalb RStudios reproduzierbar.

```
x = 1          # Objekt (0x74b) erzeugt, x referenziert Speicheradresse des Objektes

x[1] = 2  # Objekt (0x74b) veraendert
```

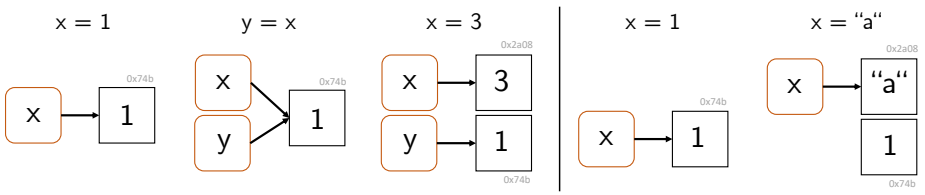
2. Environments werden in-place modifiziert ($\rightarrow$ Environments und Funktionen).

Variablenrepräsentation | Unbinding und Carbage Collection

Copy-on-modify gilt auch in umgekehrter Reihenfolge

```
x = 1       # Objekt (0x74b) erzeugt, x referenziert Speicheradresse  des Objektes
y = x       # y referenziert dieselbe Speicheradresse wie x (0x74b)
x = 3       # Ein neues Objekt (0x2a08) wird erzeugt, x referenziert (0x2a08)
y           # y referenziert weiterhin  Objekt (0x74b)
```

```
> [1] 1
```



Unbinding

```
x = 1       #  x referenziert Objekt (0x74b)
x = "a"     #  x referenziert Objekt (0x2a08), Objekt (0x74b) jetzt ohne Referenz
```

Carbage collection

- Nicht referenzierte Objekte im Arbeitsspeicher werden automatisch gelöscht.
- Das Löschen geschieht meist erst dann, wenn es wirklich nötig ist.
- Es ist nicht nötig, aktiv die Garbage Collection Funktion `gc()` zu benutzen.

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

Klassische Datenstrukturen einer 3GL Programmiersprache

Fundamentale Datenstrukturen

- Vordefiniert innerhalb der Programmiersprache
- Logische Werte (logical): TRUE, FALSE
- Ganze Zahlen (integer): int8 (-128,. . . ,127), int16 (–32768,. . . , 32767)
- Gleitkommazahlen (single, double): 1.23456, 12.3456, 123.456, . . .
- Zeichen (character): “a,” “b,” “c,” “!”
- Datentyp-spezifische assoziierte Operationen
  - AND, OR (logical) +, - (integer) +,-,*, / (single), Zeichenkonkatenation (character)

Zusammengesetzte Datenstrukturen

- Vordefinierte Container zur Zusammenfassung mehrerer Variablen gleichen Datentyps
- Zum Beispiel Vektoren, Listen, Arrays, Matrizen, . . .
- Container-spezifische Operationen (Z.B. Vektorindizierung, Matrixmultiplikation, . . .)

Selbstdefinierte Datenstrukturen

- Definition eigener Datenstrukturen aus vordefinierten Datenstrukturen und Containern
- Definition eigener Operationen

Datenstrukturenkennenlernen beim Erlernen einer Programmiersprache

Fundamentale Datenstrukturen

- Welche fundamentalen Datenstrukturen bietet die Sprache an?
- Welche Operationen darauf sind bereits definiert?
- Wie lautet die Syntax zur Definition einer Variable eines fundamentalen Datentyps?
- Wie lautet die Syntax, um vordefinierte Operationen aufzurufen?

Zusammengesetzte Datenstrukturen

- Welche Container und zugehörige Operationen bietet die Programmiersprache?
- Wie lautet die Syntax zum Umgang mit einem Containers?

Selbstdefinierte Datenstrukturen

- Wie erzeugt man selbstdefinierte Datenstrukturen und zugehörige Operationen?
- Wie lautet die Syntax zum Umgang mit einer selbstdefinierten Datenstruktur?

Organisation von Daten in R

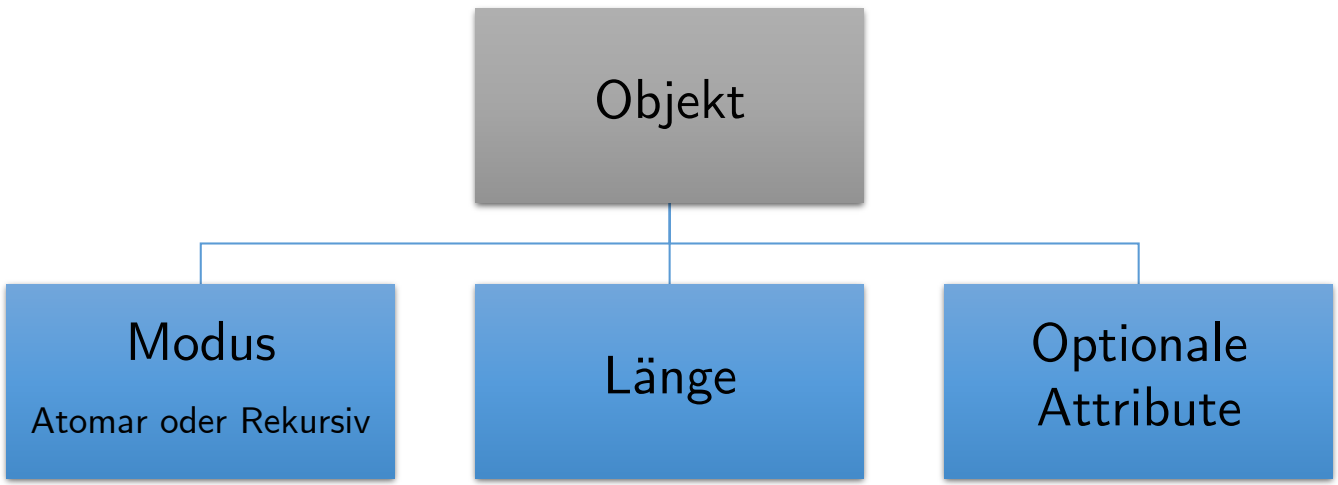
Alles, was in R vorkommt, ist ein **Objekt**

Jedem Objekt kann eindeutig zugeordnet werden

- ein **Modus**
  - Atomar | Komponenten sind vom gleichen Datentyp.
  - Rekursiv | Komponenten können von unterschiedlichem Datentyp sein.
- eine **Länge**
- optional weitere **Attribute**

Organisation von Daten in R

Alles, was in R vorkommt, ist ein **Objekt**



Übersicht der R Datentypen

Datentyp	Erläuterung
logical	Die beiden logischen Werte TRUE und FALSE
double	Gleitkommazahlen
integer	Ganze Zahlen
complex	Komplexe Zahlen, hier nicht weiter besprochen
character	Zeichen und Zeichenketten (strings), 'x' oder "Hallo Welt!"
raw	Bytes, hier nicht weiter besprochen

Double und integer werden zusammen auch als numeric bezeichnet.

Viele weitere Typen, hier relevant sind **logical**, **double**, **integer**, **character**.

Übersicht der R Datentypen

Automatische Festlegung von Datentypen durch Zuweisung

```
b = TRUE                    # logical
x = 2.5                     # double
y = 1L                      # (long) integer
c = 'a'                     # character
```

Testen von Datentypen durch `typeof()`

```
typeof(b)
```

```
> [1] "logical"
```

```
typeof(x)
```

```
> [1] "double"
```

```
typeof(y)
```

```
> [1] "integer"
```

```
typeof(c)
```

```
> [1] "character"
```

Testen von Datetypen durch `is.*()`

```
is.logical(x)
```

```
> [1] FALSE
```

```
is.double(x)
```

```
> [1] TRUE
```

Übersicht atomare Datenstrukturen in R

Datenstruktur	Erläuterung
Vektor	Container von indizierte Komponenten identischen Typs
Matrix	Interpretation eines Vektors als zweidimensionaler Container
Array	Interpretation eines Vektors als mehrdimensionaler Container

$\Rightarrow$ (3) Vektoren, Matrizen, Arrays

Übersicht rekursive Datenstrukturen in R

Datenstruktur	Erläuterung
Liste	Container von indizierten Komponenten beliebigen Datentyps
	Insbesondere auch rekursive Struktur, z.B. Liste von Listen
Dataframe	Symbiose aus Liste und Matrix
	Jede Komponente ist Vektor beliebigen Datentyps identischer Länge

$\Rightarrow$ (4) Listen und Dataframes

R und RStudio

Arithmetik, Logik und Präzedenz

Variablen

Datenstrukturen

Übungen und Selbstkontrollfragen

Übungen und Selbstkontrollfragen

1. Installieren Sie R und RStudio auf Ihrem Rechner.
2. Führen Sie die Befehlssequenz auf Folie R Skripte | Executing and Editing Code aus.
3. Dokumentieren Sie die in dieser Einheit eingeführten R Befehle in einem kommentierten R Skript.
4. Erläutern Sie den Begriff der Operatorpräzedenz.
5. Definieren Sie den Begriff der Variable im Kontext der Programmierung.
6. Erläutern Sie die Begriffe Initialisierungsanweisung und Zuweisungsanweisung für Variablen.
7. Erläutern Sie den Begriff Workspace.
8. Geben Sie jeweils ein Beispiel für einen zulässigen und einen unzulässigen Variablennamen in R.
9. Erläutern Sie die Prozesse, die R im Rahmen einer Zuweisungsanweisung der Form $x = 1$ durchführt.
10. Erläutern Sie den Begriffe Copy-on-modify und Modify-in-place.
11. Diskutieren Sie die klassischen Datenstrukturen einer 3GL Programmiersprache.
12. Diskutieren Sie die Organisation von Datenstrukturen in R.
13. Wodurch unterscheiden sich eine atomare und ein rekursive Datenstruktur in R?
14. Nennen und erläutern Sie vier zentrale Datentypen in R.
15. Nennen und erläutern Sie vier zentrale atomare Datenstrukturen in R.
16. Nennen und erläutern Sie zwei zentrale rekursive Datenstrukturen in R.

References

Becker, Richard A., John M. Chambers, and Allen Reeve Wilks. 1988. *The New S Language: A Programming Environment for Data Analysis and Graphics*. Reprint. London: Chapman & Hall.

Ihaka, Ross, and Robert Gentleman. 1996. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics* 5 (3): 2999–2314.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(3) Vektoren

Erzeugung

Charakterisierung

Indizierung

Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Erzeugung

Charakterisierung

Indizierung

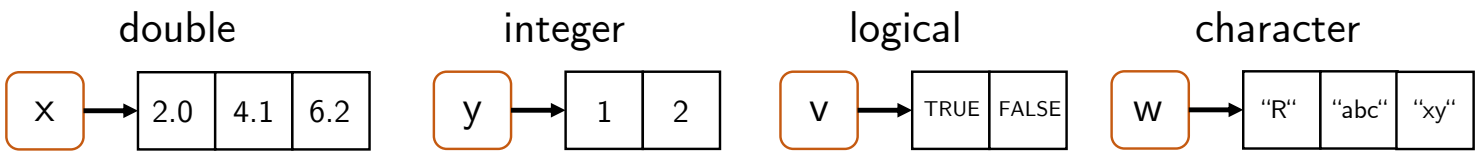
Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Erzeugung

- Vektoren sind geordnete Folgen von Datenwerten.
- Die einzelnen Datenwerte eines Vektors heißen Elemente des Vektors.
- Vektoren, deren Elemente alle vom gleichen Datentyp sind, heißen **atomar**.
- Die zentralen Datentypen sind **numeric**, (**double**, **integer**), **logical**, **character**



- Mit dem Begriff **Vektor** ist hier immer ein **atomarer Vektor** gemeint.

Erzeugung

Elementarwerte

Numeric

Double wird in Dezimalnotation oder wissenschaftlicher Notation spezifiziert

Weitere mögliche Werte sind Inf, -Inf, und NaN (Not-a-Number)

```
x = 1                          # Einelementiger Vektor vom Typ double
y = 2.1e2                      # Einelementiger Vektor vom Typ double
z = Inf                        # Einelementiger Vektor vom Typ double
```

Integer wird wie double ohne Dezimalstellen spezifiziert, gefolgt von L (long integer)

```
x = 1L                         # Einelementiger Vektor vom Typ integer
y = 200L                       # Einelementiger Vektor vom Typ integer
```

Logical

TRUE oder FALSE, abgekürzt T oder F

```
x = TRUE                       # Einelementiger Vektor vom Typ logical
y = F                          # Einelementiger Vektor vom Typ logical
```

Character

Anführungszeichen (“a”) oder Hochkommata (‘a’)

```
x = "a"                        # Einelementiger Vektor vom Typ character
y = 'test'                     # Einelementiger Vektor vom Typ character
```

Erzeugung

Direkte Konkatenation von Elementarwerten mit `c()`

```
x = c(1,2,3)                    # numeric vector [1,2,3]
y = c(0,x,4)                    # numeric vector [0,1,2,3,4]
s = c("a", "b", "c")            # character vector ["a", "b", "c"]
l = c(TRUE, FALSE)              # logical  vector [TRUE, FALSE]
```

`c()` konkateniert die Eingabeargumente und erzwingt einen einheitlichen Datentyp

```
x = c(1, "a", TRUE)             # character vector ["1", "a", "TRUE"]
```

Erzeugen "leerer" Vektoren mit `vector()`

```
v = vector("double",3)          # double vector [0,0,0]
w = vector("integer",3)         # integer vector [0,0,0]
l = vector("logical",2)         # logical vector [FALSE, FALSE]
s = vector("character",4)       # character vector ["", "", "", ""]
```

Erzeugen "leerer" Vektoren mit double(), integer(), logical(), character()

```
v = double(3)                   # double vector [0,0,0]
w = integer(3)                  # integer vector [0,0,0]
l = logical(2)                  # logical vector [FALSE, FALSE]
s = character(4)                # character vector ["", "", "", ""]
```

Erzeugung

Erzeugen von ganzzahligen Sequenzen mithilfe des **Colonoperators** :

`a:b` erzeugt ganzzahlige Sequenzen von a (inklusive) bis b (maximal)

```
x = 0:5                         # [0,1,2,3,4,5]
y = 1.5:6.1                     # [1.5, 2.5, 3.5, 4.5, 5.5]
```

Erzeugen von Sequenzen mit `seq()`

`seq(from, to, by = ((to - from)/(len - 1), len = NULL, ...)`

```
x_1 = seq(0,5)                  # wie 0:5, [0,1,2,3,4,5]
x_2 = seq(0,1,len = 5)          # 5 Zahlen zwischen 0 (inkl.) und 1 (inkl.)
                                # [0.00, 0.25, 0.50, 0.75, 1.00]
x_3 = seq(0,2,by = .15)         # 0.15 Schritte zwischen 0 (inkl.) und 2 (max.)
                                # [0.00, 0.15, 0.30, ..., 1.50 1.65 1.80 1.95]
x_4 = seq(1,0,by = -.1)         # -0.1 Schritte zwischen 1 (inkl.) und 0 (min.)
```

`seq.int()`, `seq_len()`, `seq_along()` als weitere Varianten

```
x_1 = seq.int(0,5)              # wie 0:5, [0,1,2,3,4,5]
x_2 = seq_len(5)                # Natuerliche Zahlen bis 5, [1,2,3,4,5]
x_3 = seq_along(c("a","b"))     # wie seq_len(length(c("a", "b")))
```

Erzeugung

Charakterisierung

Indizierung

Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Charakterisierung

`length()` gibt die Anzahl der Elemente eines Vektors aus

```
x = 0:10                    # Vektor
length(x)                   # Anzahl der Elemente des Vektors
```

```
> [1] 11
```

`typeof()` gibt den elementaren Datentyp eines Vektors aus

```
x = 1:3L                    # Vektor
typeof(x)                   # Datentyp des atomic vectors
```

```
> [1] "integer"
```

```
y = c(T,F,T)                # Vektor
typeof(y)                   # Der Datentyp des atomic vectors
```

```
> [1] "logical"
```

`mode()` und `storage.mode()` werden nicht empfohlen, sie existieren für S Kompatibilität

`is.logical()`, `is.double()`, `is.integer()`, `is.character()` testen den Datentyp

```
is.double(x)                # Testen obigen Vektors
```

```
> [1] FALSE
```

```
is.logical(y)               # Testen obigen Vektors
```

```
> [1] TRUE
```

Datentypangleichung (Coercion)

Bei Konkatenation verschiedener Datentypen wird ein einheitlicher Datentyp erzwungen. Es gilt

$$\text{character} > \text{double} > \text{integer} > \text{logical}$$

```
x = c(1.2,"a")          # Kombination gemischter Datentypen (character schlaegt double)
x
```

```
> [1] "1.2" "a"
```

```
typeof(x)               # Erzeugter Vektor ist vom Datentyp character
```

```
> [1] "character"
```

```
y = c(1L, TRUE)         # Kombination  gemischter Datentypen (integer schlaegt logical)
y
```

```
> [1] 1 1
```

```
typeof(y)               # Erzeugter Vektor ist vom Typ integer
```

```
> [1] "integer"
```

Datentypangleichung (Coercion)

Bei Konkatenation verschiedener Datentypen wird ein einheitlicher Datentyp erzwungen. Es gilt

$$\text{character} > \text{double} > \text{integer} > \text{logical}$$

`as.logical()`, `as.integer()`, `as.double()`, `as.character()` erlauben explizite Coercion:

```
x = c(0,1,1,0)          # double Vektor
y = as.logical(x)       # ... umgewandelt in logical
y
```

Coercion geschieht aber auch oft implizit:

```
x = c(T, F, T, T)       # logical Vektor
s = sum(x)              # Summation in integer gewandelter logical Elemente
s
```

```
> [1] 3
```

Erzeugung

Charakterisierung

Indizierung

Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Indizierung

Einzelne oder mehrere Vektorkomponenten werden durch Indizierung adressiert.

Indizierung wird auch Indexing, Subsetting, oder Slicing genannt.

Zur Indizierung werden eckige Klammern [ ] benutzt.

Indizierung kann zur Kopie oder Manipulation von Komponenten benutzt werden.

Der Index des ersten Elements ist 1 (nicht 0, wie in anderen Sprachen).

```
x       = c("a", "b", "c")      # character vector ["a", "b", "c"]
y       = x[2]                  # Kopie von "b" in y
x[3]    = "d"                   # Aenderung von x zu x = ["a", "b", "d"]
```

Prinzipien der Indizierung in R

Indizierung mit . . .

. . . einem Vektor positiver Zahlen adressiert entsprechende Komponenten.

. . . mit einem Vektor negativer Zahlen adressiert komplementäre Komponenten.

. . . einem logischen Vektor adressiert die Komponenten mit TRUE.

. . . einem character Vektor adressiert benannte Komponenten.

Indizierung

Beispiele

Indizierung mit einem Vektor positiver Zahlen

```
x = c(1,4,9,16,25)       # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[1:3]               # 1:3 erzeugt Vektor [1,2,3], x[1:3] = [1,4,9]
z = x[c(1,3,5)]          # c(1,3,5) erzeugt Vektor [1,3,5], x[c(1,3,5)] = [1,9,25]
```

Indizierung mit einem Vektor negativer Zahlen

```
x = c(1,4,9,16,25)       # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[c(-2,-4)]          # Alle Komponenten ausseer 2 und 4, x[c(-2,-4)] = [1,9,25]
z = x[c(-1,2)]           # Gemischte Indizierung nicht erlaubt (Fehlermeldung)
```

Indizierung mit einem logischen Vektor

```
x = c(1,4,9,16,25)       # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[c(T,T,F,F,T)]      # TRUE Komponenten,  x[c(T,T,F,F,T)] = [1,4,25]
z = x[x > 5]             # x > 5 = [F,F,T,T,T], x[x > 5] = [9,16,25]
```

Indizierung mit einem character Vektor

```
x = c(1,4,9,16,25)       # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
names(x) = c("a","b")    # Benennung der Komponenten als [a  b <NA> <NA> <NA>]
y = x["a"]               # x["a"] = 1
```

Indizierung

R hat eine (zu) hohe Flexibilität bei Indizierung

Out-of-range Indizes verursachen keine Fehler, sondern geben NA aus

```
x = c(1,4,9,16,25)        # [1,4,9,16,25] = [1^2, 2^2, 3^2, 4^2, 5^2]
y = x[10]                 # x[10] = NA (Not Applicable)
```

Nichtganzzahlige Indizes verursachen keine Fehler, sondern werden abgerundet

```
y = x[4.9]                # x[4.9]    = x[4]  = 16
z = x[-4.9]               # x[-4.9] = x[-4] = [1,4,9,25]
```

Leere Indizes indizieren den gesamten Vektor

```
y = x[]                   # y = x
```

Erzeugung

Charakterisierung

Indizierung

Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Arithmetik

Unitäre arithmetische Operatoren und Funktionen werden elementweise ausgewertet

```
a   = seq(0,1,len=11)   # a = [ 0.0   ,  0.1   , ...,  0.9    , 1.0    ]
b   = -a                # b = [-0.0   , -0.1   , ..., -0.9    ,-1.0    ]
v   = a^2               # v = [ 0.0^2 ,  0.1^2 , ...,  0.9^2  , 1.0^2  ]
w   = log(a)            # w = [ln(0.0), ln(0.1), ...,  ln(0.9), ln(1.0)]
```

Binäre arithmetische Operatoren werden elementweise ausgewertet

Vektoren gleicher Länge

```
a = c(1,2,3)            # a = [1,2,3]
b = c(2,1,4)            # b = [2,1,4]
v = a + b               # v = [1,2,3] + [2,1,4] = [1+2,2+1,3+4] = [   3,  3,    7]
w = a - b               # w = [1,2,3] - [2,1,4] = [1-2,2-1,3-4] = [  -1,  1,   -1]
x = a * b               # x = [1,2,3] * [2,1,4] = [1*2,2*1,3*4] = [   2,  2,   12]
y = a / b               # y = [1,2,3] / [2,1,4] = [1/2,2/1,3/4] = [0.50,  2, 0.75]
```

Vektoren und Skalare (= Vektoren der Länge 1)

```
a = c(1,2,3)            # a = [1,2,3]
b = 2                   # b = [2]
v = a + b               # v = [1,2,3] + [2,2,2] = [1+2,2+2,3+2] = [  3,  4,   5]
w = a - b               # w = [1,2,3] - [2,2,2] = [1-2,2-2,3-2] = [ -1,  2,   1]
x = a * b               # x = [1,2,3] * [2,2,2] = [1*2,2*2,3*2] = [  2,  4,   6]
y = a / b               # y = [1,2,3] / [2,2,2] = [1/2,2/2,3/2] = [0.5,  1, 1.5]
```

Recycling

R erlaubt (leider) auch Arithmetik mit Vektoren unterschiedlicher Länge

Bei ganzzahligen Vielfachen der Länge wird der kürzere Vektor wiederholt

```
x = 1:2                 # x = [1,2], length(x) = 2
y = 3:6                 # y = [3,4,5,6], length(y) = 4
v = x + y               # v = [1,2,1,2] + [3,4,5,6] = [4,6,6,8]
```

Arithmetik von Vektoren und Skalaren ist ein Spezialfall dieses Prinzips

Andernfalls werden die ersten Komponenten des kürzeren Vektors wiederholt

```
x = c(1,3,5)            # x = [1,3,5], length(x) = 3
y = c(2,4,6,8,10)       # y = [2,4,6,8,10], length(y) = 5
v = x + y               # v = [1,3,5,1,3] + [2,4,6,8,10] = [3,7,11,9,13]
```

Generell sollten nur Vektoren gleicher Länge arithmetisch verknüpft werden!

Fehlende Werte (NA)

Fehlende Werte werden in R mit NA (not applicable) repräsentiert

Das Rechnen mit NAs ergibt (meist) wieder NA

```
3 * NA                  # Multiplikation eines NA Wertes ergibt NA
```

```
> [1] NA
```

```
NA < 2                  # Relationaler Vergleich eines NA Wertes ergibt NA
```

```
> [1] NA
```

```
NA^0                    # NA hoch 0 ergibt 1, weil jeder Wert hoch 0 eins ergibt (?)
```

```
> [1] 1
```

```
NA & FALSE              # NA UND FALSE  ergibt FALSE, weil jeder Wert UND FALSE FALSE ergibt
```

```
> [1] FALSE
```

Auf NA testet man mit `is.na()`

```
x = c(NA, 5, NA, 10)    # Vektor mit NAs
x == NA                 # Kein Testen auf NAs : 5 == NA ist NA, nicht FALSE
```

```
is.na(x)                # Logisches Testen auf NA
```

```
> [1] FALSE FALSE FALSE FALSE
```

Erzeugung

Charakterisierung

Indizierung

Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Attribute

Attribute sind Metadaten von R Objekten in Form von Schlüssel-Wert-Paaren

‘attributes() ruft alle Attribute eines Objektes auf

Perse haben atomic vectors keine Attribute

```
a = 1:3              # ein numerischer Vektor
attributes(a)        # Aufrufen aller Attribute
```

```
> NULL
```

`attr()` kann zum Aufrufen und Definieren von Attributen genutzt werden

```
attr(a, "S") = "W"   # a bekommt Attribut mit Schluessel S und Wert W
attr(a, "S")         # Das Attribut mit Schluessel S hat den Wert W
```

```
> [1] "W"
```

Attribute werden bei Operationen oft entfernt (Ausnahmen sind `names` und `dim`)

```
b = a[1]             # Kopie des ersten Elements von a in Vektor b
attributes(b)        # Aufrufen aller Attribute von b
```

Attribute

Spezifikation des Attributs `names` gibt den Elementen eines Vektors Namen

```
v = c(x=1,y=2,z=3)          # Elementnamengeneration bei Vektorerzeugung
v                           # Vektorausgabe
```

```
> x y z
> 1 2 3
```

Die Namen können zur Indizierung benutzt werden

```
v["y"]                      # Indizierung per Namen
```

```
> y
> 2
```

`names()` kann zum Definieren und Aufrufen von Namen benutzt werden

```
y = 4:6                     # Erzeugung eines Vektors
names(y) = c("a","b","c")   # Definition von Namen
names(y)                    # Elementnamenaufruf
```

```
> [1] "a" "b" "c"
```

Benannte Namen können hilfreich sein, wenn der Vektor eine Sinneinheit bildet

```
p = c(age    = 31,          # Alter (Jahre), Groesse (cm), Gewicht (kg) einer Person
      height = 198,
      weight = 75)
p                           # Vektorausgabe
```

Erzeugung

Charakterisierung

Indizierung

Arithmetik

Attribute

Übungen und Selbstkontrollfragen

Übungen und Selbstkontrollfragen

1. Dokumentieren Sie die in dieser Einheit eingeführten R Befehle in einem R Skript.
2. Beschreiben Sie in einer Übersicht die R Datenstruktur "Atomarer Vektor."
3. Erläutern Sie die Funktion des Colonoperators in R.
4. Nennen Sie vier Prinzipien der Indizierung in R.
5. Erzeugen Sie einen Vektor der Dezimalzahlen 0.0, 0.05, 0.10 , 0.15, . . . , 0.90, 0.95, 1.0.
6. Wählen Sie mithilfe positiver Indices die Elemente 0.0, 0.1,. . . , 0.9, 1.0 dieses Vektors aus.
7. Wählen Sie mithilfe negativer Indices die Elemente 0.0, 0.1,. . . , 0.9, 1.0 dieses Vektors aus.
8. Wählen Sie die letzten 10 Elemente dieses Vektors aus.
9. Erläutern Sie den Begriff der Datentypangleichung (Coercion).
10. Erläutern Sie den Begriff des (Vektor)Recylings.
11. Erläutern Sie die Bedeutung des R Datentyps `NA`.
12. Erläutern Sie das Konzept der Attribute in R.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(4) Matrizen

Programmierung und Deskriptive Statistik

Datum	Einheit	Thema
14.10.2021	Einführung	(1) Einführung
21.10.2021	R Grundlagen	(2) R und RStudio
28.10.2021	R Grundlagen	(3) R und RStudio
04.11.2021	R Grundlagen	(4) Vektoren
11.11.2021	**R Grundlagen**	**(5) Matrizen**
18.11.2021	R Grundlagen	(6) Listen und Dataframes
25.11.2021	R Grundlagen	(7) Kontrollstruktur und Schleifen
02.12.2021	R Grundlagen	(8) R Base Graphics
09.12.2021	Deskriptive Statistik	(9) Datenmanagement
16.12.2021	Deskriptive Statistik	(10) Häufigkeitsverteilungen
	Weihnachtspause	
06.01.2022	Deskriptive Statistik	(11) Verteilungsfunktionen und Quantile
13.01.2022	Deskriptive Statistik	(12) Maße der zentralen Tendenz
20.01.2022	Deskriptive Statistik	(13) Maße der Datenvariabilität
27.01.2022	Deskriptive Statistik	(14) Bivariate Zusammenhangsmaße
25.02.2021	**Klausurtermin**	**9 - 10 Uhr, G26 - H11**
Jul 2022	Klausurwiederholungstermin	

Matrizen

Übungen und Selbstkontrollfragen

Matrizen

Übungen und Selbstkontrollfragen

Übersicht

Matrizen sind zweidimensionale, rechteckige Datenstrukturen der Form

$$M = \begin{pmatrix} m_{11} & m_{12} & \cdots & m_{1n_c} \\ m_{21} & m_{22} & \cdots & m_{2n_c} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n_r1} & m_{n_r2} & \cdots & m_{n_rn_c} \end{pmatrix} \tag{1}$$

- Die Elemente $m_{ij}, i = 1, ..., n_r, j = 1, ..., n_c$ sind vom gleichen Typ.
- n_r ist die Anzahl der Zeilen (rows), n_c ist die Anzahl der Spalten (columns).
- Jedes Element einer Matrix hat einen Zeilenindex i und einen Spaltenindex j.
- Intuitiv sind Matrizen numerisch indizierte Tabellen.
- Formal sind Matrizen in R zweidimensional interpretierte atomare Vektoren.
- Matrizen in R sind nicht identisch mit dem mathematischen Matrixbegriff.
- Matrizen in R können allerdings für Lineare Algebra verwendet werden.
- Lineare Algebra ist die Sprache (linearer) statistischer Modelle.

Erzeugung

Die `matrix()` Funktion befüllt Matrizen mit Vektorelementen

```
matrix(data, nrow, ncol, byrow)
```

```
matrix(c(1:12), nrow = 3)             # 3 x 4 Matrix der Zahlen 1,...,12, byrow = F
```

```
>      [,1] [,2] [,3] [,4]
> [1,]    1    4    7   10
> [2,]    2    5    8   11
> [3,]    3    6    9   12
```

```
matrix(c(1:12), ncol = 4)             # 3 x 4 Matrix der Zahlen 1,...,12, byrow = F
```

```
>      [,1] [,2] [,3] [,4]
> [1,]    1    4    7   10
> [2,]    2    5    8   11
> [3,]    3    6    9   12
```

```
matrix(c(1:12), nrow = 3, byrow = T)  # 3 x 4 Matrix der Zahlen 1,...,12, byrow = T
```

```
>      [,1] [,2] [,3] [,4]
> [1,]    1    2    3    4
> [2,]    5    6    7    8
> [3,]    9   10   11   12
```

Matrizen

Erzeugung Die Funktion `cbind()` konkateniert passende Matrizen spaltenweise

```
A = matrix(c(1:4) , nrow = 2)        # 2 x 2 Matrix der Zahlen 1,...,4
print(A)
```

```
>      [,1] [,2]
> [1,]    1    3
> [2,]    2    4
```

```
B = matrix(c(5:10), nrow = 2)        # 2 x 3 Matrix der Zahlen 5,...,10
print(B)
```

```
>      [,1] [,2] [,3]
> [1,]    5    7    9
> [2,]    6    8   10
```

```
C = cbind(A,B)                       # spaltenweise Konkatenierung von A und B
print(C)
```

```
>      [,1] [,2] [,3] [,4] [,5]
> [1,]    1    3    5    7    9
> [2,]    2    4    6    8   10
```

Erzeugung Die Funktion `rbind()` konkateniert passende Matrizen reihenweise

```
A = matrix(c(1:6) , nrow = 2, byrow = T)  # 2 x 3 Matrix der Zahlen 1,...,6
print(A)
```

```
>      [,1] [,2] [,3]
> [1,]    1    2    3
> [2,]    4    5    6
```

```
B = matrix(c(7:9), nrow = 1)              # 1 x 3 Matrix der Zahlen 5,...,10
print(B)
```

```
>      [,1] [,2] [,3]
> [1,]    7    8    9
```

```
C = rbind(A,B)                            # reihenweise Konkatenierung von A und B
print(C)
```

```
>      [,1] [,2] [,3]
> [1,]    1    2    3
> [2,]    4    5    6
> [3,]    7    8    9
```

Charakterisierung

`typeof()` gibt den elementaren Datentyp einer Matrix aus

```
A = matrix(c(T,T,F,F), nrow = 2)          # 2 x 2 Matrix von Elementen vom Typ logical
typeof(A)
```

```
> [1] "logical"
```

```
B = matrix(c("a","b","c"), nrow = 1)      # 1 x 3 Matrix von Elementen vom Typ character
typeof(B)
```

```
> [1] "character"
```

`nrow()` und `ncol()` geben die Zeilen- bzw. Spaltenanzahl aus

```
C = matrix(1:12, nrow = 3)                # 3 x 4 Matrix
nrow(C)                                   # Anzahl Zeilen
```

```
> [1] 3
```

```
ncol(C)                                   # Anzahl Spalten
```

```
> [1] 4
```

Indizierung

Generell gilt

- Matrixelemente werden mit einen Zeilenindex und einem Spaltenindex indiziert.
- Die Indexreihenfolge ist immer 1. Zeile, 2. Spalte.
- Die Prinzipien der Indizierung entsprechen der Vektorindizierung.
- Indizes verschiedener Dimensionen können unterschiedlich indiziert werden.
- Eindimensionale Resultate liegen als Vektor, nicht als Matrix vor.

```
A = matrix(c(2:7)^2, nrow = 2)      # 2 x 3 Matrix der Zahlen 2^2,...,7^2
print(A)
a_13  = A[1,3]                      # Element in 1. Zeile, 3. Spalte von A [36]
a_22  = A[2,2]                      # Element in 2. Zeile, 2. Spalte von A [35]
a_2.  = A[2,]                       # Alle Elemente der 2. Zeile [9,25,49]
a_.3  = A[,3]                       # Alle Elemente der 3. Spalte [36,49]
A_12  = A[1:2,1:2]                  # Submatrix der ersten zwei Zeilen und Spalten
A10   = A[A>10]                     # Elemente von A groesser 10 [16,25,36,49]
A_13  = A[1,c(F,F,T)]               # Element in 1. Zeile, 3. Spalte von A [36]
```

Arithmetik

Unitäre arithmetische Operatoren und Funktionen werden elementweise ausgewertet

```
A = matrix(c(1:4), nrow = 2)     # 2 x 2 Matrix der Zahlen 1,2,3,4
     [,1] [,2]
[1,]    1    3
[2,]    2    4

B = A^2                          # B[i,j]  = A[i,j]^2,  1 <= i,j <= 2
     [,1] [,2]
[1,]    1    9                   # 1^2, 3^2
[2,]    4   16                   # 2^2, 4^2

C = sqrt(B)                      # C[i,j]  = sqrt(A[i,j]^2),  1 <= i,j <= 2
     [,1] [,2]
[1,]    1    3                   # sqrt(1^2), sqrt(3^2)
[2,]    2    4                   # sqrt(2^2), sqrt(4^2)

D = exp(A)                       # D[i,j] = exp(A[i,j]),  1 <= i,j <= 2
     [,1] [,2]
[1,]  2.7  20.0                  # exp(1), exp(3)
[2,]  7.4  54.6                  # exp(2), exp(4)
```

Matrizen

Arithmetik

Matrizen passender Größe können mit binären arithmetischen Operatoren verknüpft werden

Binäre arithmetische Operatoren +,-,*,\werden bei gleicher Größe elementweise ausgewertet

```
A = matrix(c(1:4), nrow = 2)     # 2 x 2 Matrix der Zahlen 1,2,3,4
     [,1] [,2]
[1,]    1    3
[2,]    2    4

B = matrix(c(5:8), nrow = 2)     # 2 x 2 Matrix der Zahlen 5,6,7,8
     [,1] [,2]
[1,]    5    7
[2,]    6    8

C = A + B                        # C[i,j] = A[i,j] + B[i,j], 1 <= i,j <= 2
     [,1] [,2]
[1,]    6   10                   # 1 + 5, 3 + 7
[2,]    8   12                   # 2 + 6, 4 + 8

D = A * B
     [,1] [,2]
[1,]    5   21                   # 1 * 5, 3 * 7
[2,]   12   32                   # 2 * 6, 4 * 8
```

Matrizen

Arithmetik

Mit R Matrizen kann Lineare Algebra betrieben werden

- Addition, Subtraktion, Hadamardprodukt elementweise definiert wie oben
- Matrixmultiplikation, Transposition, Inversion, Determinante

```
C = A  %*% B                    # 2 x 2 Matrixprodukt
      [,1] [,2]
[1,]   23   31                  # 1*5 + 3*6, 1*7+3*8
[2,]   34   46                  # 2*5 + 4*6, 2*7+4*8

A_T = t(A)                      # Transposition von A
      [,1] [,2]
[1,]   1   2                    # A[1,1], A[2,1]
[2,]   3   4                    # A[1,2], A[2,2]

A_inv = solve(A)                # Inverse von A
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5

A_det = det(A)                  # Determinante von A
[1] -2                          # 1*4 - 2*3
```

Matrizen

Attribute

Formal sind Matrizen atomare Vektoren mit einem `dim` Attribut

```
A = matrix(1:12, nrow = 4 )                 #  4 x 3 Matrix
attributes(A)                               # Aufrufen der Attribute von A
```

```
> $dim
> [1] 4 3
```

`rownames()` und `colnames()` spezifizieren das Attribut `dimnames`

```
rownames(A) = c("P1","P2","P3","P4")        # Benennung der Zeilen von A
colnames(A) = c("Age", "Hgt", "Wgt")        # Benennung der Spalten von A
A                                           # A mit Attribut dimnames
```

```
>    Age Hgt Wgt
> P1   1   5   9
> P2   2   6  10
> P3   3   7  11
> P4   4   8  12
```

```
attr(A, "dimnames")                         # Aufrufen des Attributs dimnames
```

```
> [[1]]
> [1] "P1" "P2" "P3" "P4"
>
> [[2]]
> [1] "Age" "Hgt" "Wgt"
```

Bei Matrizen ist die Benennung von Zeilen und Spalten eher ungewöhnlich.

Matrizen

Übungen und Selbstkontrollfragen

Übungen und Selbstkontrollfragen

1. Dokumentieren Sie alle in dieser Einheit eingeführten Befehle in einem R Skript.
2. Erzeugen Sie in R die Matrizen

$$A = \begin{pmatrix} 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 4 \\ 2 & 1 & 4 & 3 \end{pmatrix} \text{ und } B = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

3. Kopieren Sie die zweite Zeile von A in einen Vektor.
4. Kopieren Sie die erste und dritte Spalte von B in eine 3 $\times$ 2 Matrix
5. Setzen Sie alle Nullen in B auf -1.
6. Setzen Sie die zweite Zeile von A auf $(1\,2\,3\,4)$.
7. Addieren Sie die Matrizen A und B.
8. Multiplizieren Matrix A mit 3.
9. Konkatenieren Sie die Matrizen A und B zeilenweise.
10. Konkatenieren Sie die Matrizen A und B spaltenweise.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(5) Listen und Dataframes

Listen

Dataframes

Übungen und Selbstkontrollfragen

Listen

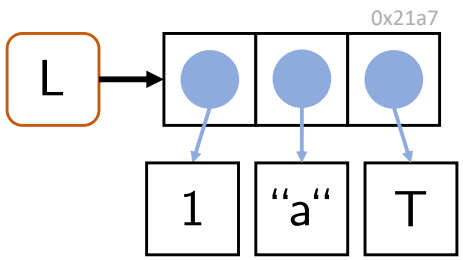
Dataframes

Übungen und Selbstkontrollfragen

Übersicht

- Listen sind geordnete Folgen von R Objekten.
- Listen sind rekursiv, können also Objekte verschiedenen Datentyps enthalten.
- Defacto enthalten Listen keine Objekte, sondern Referenzen zu Objekten.

$$L = \text{list}(1,\text{“a“},T)$$



- Listen sind ein wesentlicher Baustein von Dataframes.

Listen

Erzeugung

Direkte Konkatenation von Listenelementen mit `list()`

```
L = list(c(1,4,5),                          # Liste mit einem Vektor,
                matrix(1:8, nrow = 2),      # einer Matrix und
                exp)                        # einer Funktion
[[1]]                                       # 1. Listenelement
[1] 1 4 5
[[2]]                                       # 2. Listenelement
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
[[3]]
function (x)  .Primitive("exp")             # 3. Listenelement
```

Listen können Elemente von Listen sein

```
L = list(list(1))                           # Liste mit Element 1 in einer Liste
[[1]]
[[1]][[1]]
[1] 1
```

c() kann zum Verbinden von Listen genutzt werden

```
L = c(list(pi), list("a"))                  # Konkatenation zweier Listen
[[1]]                                       # 1. Listenelement
[1] 3.141593
[[2]]                                       # 2. Listenelement
[1] "a"
```

Listen

Charakterisierung

Der Datentyp von Listen ist `list`

```
L = list(1:2, "a", log)             # Erzeugung einer Liste
typeof(L)                           # Typenbestimmung
```

```
> [1] "list"
```

`length()` gibt die Anzahl der Toplevel Listenelemente aus

```
L = list(1:2, list("a", pi), exp)   # Liste mit drei Toplevelelementen
length(L)                           # length() ignoriert Elementinhalte, length() von L ist also 3
```

```
> [1] 3
```

Die Dimension, Zeilen-, und Spaltenanzahl von Listen ist `NULL`

```
L = list(1:2, "a", sin)             # eine Liste
dim(L)                              # Die Dimension von Listen ist NULL
```

```
> NULL
```

```
nrow(L)                             # Die Zeilenanzahl von Listen ist NULL
```

```
> NULL
```

```
ncol(L)                             # Die Spaltenanzahl von Listen ist NULL
```

```
> NULL
```

Indizierung

Einfache eckige Klammern [ ] indizieren Listenelemente als Listen

```
L  = list(1:3, "a", exp)          # eine Liste
l1 = L[1]                         # Indizierung eines Listenelements
[[1]]                             # das Listenelement l1
[1] 1 2 3                         # der Inhalt von Listenelement l1
typeof(l1)                        # Typbestimmung von l1
[1] "list"                        # L[] gibt eine Liste aus
```

Doppelte eckige Klammern [[ ]] indizieren den Inhalt von Listenelementen

```
L  = list(1:3, "a", exp)          # eine Liste
i2 = L[[2]]                       # Indizierung des Listenelementinhalts
[1] "a"                           # der Inhalt von Listenelement L[2]
typeof(i2)                        # Typbestimmung von i2
[1] "character"                   # L[[]] gibt den Listenelementinhalt aus
```

Ersetzen von Listenelement(inhalt)en

```
L       = list(1:3, "a", exp)     # eine Liste
L[1]    = 4:6                     # keine Typkonversion, Fehlermeldung
L[1]    = list(4:6)               # Ersetzung des 1. Listenelementes
L[[3]]  = "c"                     # Ersetzung des 3. Listenelementinhaltes
```

Indizierung

Die Prinzipien der Listenindizierung sind analog zur Vektorindizierung

Vektoren positiver Zahlen adressieren entsprechende Elemente

```
L = list(1:3, "a", pi)              # eine Liste
l = L[c(1,3)]                       # 1. und 3. Listenelement
[[1]]                               # 1. Listenelement
[1] 1 2 3
[[2]]                               # 2. Listenelement
[1] 3.141593
```

Vektoren negativer Zahlen adressieren komplementäre Elemente

```
L = list(1:3, "a", pi)              # eine Liste
l = L[-c(1,3)]                      # 2. Listenelement
[1] "a"
```

Logische Vektoren adressieren Elemente mit TRUE.

```
L = list(1:3, "a", pi)              # eine Liste
l = L[c(T,T,F)]                     # 1. und 2. Listenelement
[[1]]                               # 1. Listenelement
[1] 1 2 3
[[2]]                               # 2. Listenelement
[1] "a"
```

Indizierung

Listenelementen können bei Erzeugung Namen gegeben werden

```
L = list(greta  = 1:3,              # eine Liste mit benannten Elementen
         luisa  = "a",
         carla  = exp)
$greta                              # 1. Listenelement
[1] 1 2 3
$luisa                              # 2. Listenelement
[1] "a"
$carla                              # 3. Listenelement
function (x)  .Primitive("exp")
```

Listenelementen können mit `names()` Namen gegeben werden

```
K        = list(1:2, TRUE)          # eine unbenannte Liste
names(K) = c("Frodo", "Sam")        # Namensgebung mit names()
$Frodo                              # 1.Listenelement
[1] 1 2
$Sam                                # 2. Listelement
[1] TRUE
```

Indizierung

Listenelemente und Listenelementinhalte können mit Namen indiziert werden

```
L = list(greta  = 1:3,      # eine Liste mit benannten Elementen
         luisa  = "a",
         carla = exp)
L["carla"]                  # Listenelementindizierung
L[["carla"]]                # Listenelementinhaltsinindizierung
```

Listenelementinhalte können mit dem $ Operator indiziert werden

```
L = list(greta  = 1:3,       # eine Liste mit benannten Elementen
         luisa  = "a",
         carla = exp)
L$greta                      # Listenelementinhalt
```

```
> [1] 1 2 3
```

```
L$luisa                      # Listenelementinhalt
```

```
> [1] "a"
```

```
L$carla                      # Listenelementinhalt
```

```
> function (x)  .Primitive("exp")
```

Arithmetik

Listenarithmetik ist nicht definiert, da Listenelemente unterschiedlichen Typs sein können

```
L1 = list(1:3, "a" )          # eine Liste
L2 = list(T, exp )            # eine Liste
L1+L2                         # Versuch der Listenaddition
```

```
> Error in L1 + L2: nicht-numerisches Argument für binären Operator
```

Listenelementinhalte können bei Passung jedoch arithmetisch verknüpft werden

```
L1 = list(1:3, pi )           # eine Liste
L2 = list(4:6, exp )          # eine Liste
L1[[1]] + L2[[1]]             # Addition  der 1. Listenelementinhalte,  [1+4, 2+5,3+6]
```

```
> [1] 5 7 9
```

```
L2[[2]](1)                    # Anwendung des 2. Listenelementinhalts, exp(1)
```

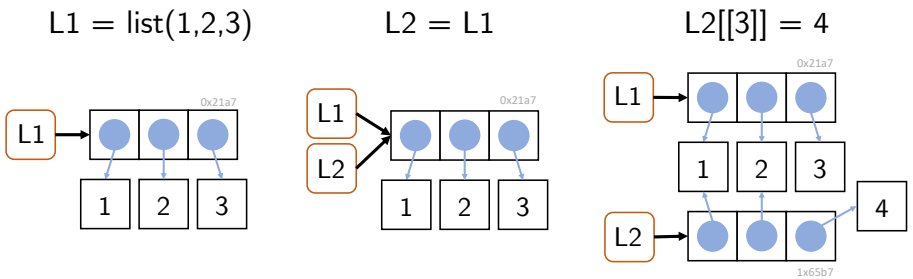
```
> [1] 2.72
```

Copy-on-modify

Wie bei Vektoren gilt bei Listen das Copy-on-Modify Prinzip.

"Shallow copy": Listenobjekt wird kopiert, aber nicht die gebundenen Objekte.

`lobstr::ref()` erlaubt es, dieses Verhalten zu studieren.




```
L1 = list(1,2,3)            # Erzeugen einer Liste als Objekt (0x1a3)
L2 = L1                     # L1 und L2 referenzieren das Objekt (0x1a3)
L2[[3]] = 4                 # Copy-on-Modify mit shallow Objekt Kopie
lobstr::ref(L1,L2)          # Ausgabe der Referenzen
```

```
> o [1:0x1d632f28] <list>
> +-[2:0x1d553248] <dbl>
> +-[3:0x1d553210] <dbl>
> \-[4:0x1d5531d8] <dbl>
>
> o [5:0x1d6e3ba8] <list>
> +-[2:0x1d553248]
> +-[3:0x1d553210]
> \-[6:0x1d5530f8] <dbl>
```

Listen

Dataframes

Übungen und Selbstkontrollfragen

Übersicht

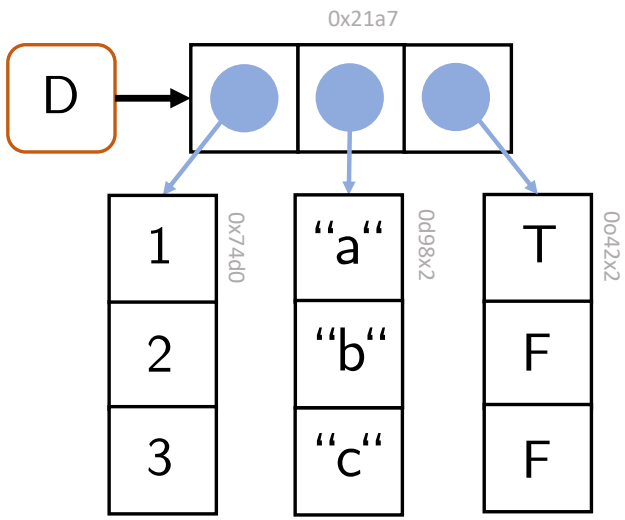
- Dataframes sind die zentrale Datenstruktur in R.
- Dataframes stellt man sich am besten als Tabelle vor.
- Die Zeilen und Spalten der Tabelle haben Namen.

	Control	drug1	drug2L	drug2R	delta1	delta2L	delta2R
1	0.6	1.3	2.5	2.1	0.7	1.9	1.5
2	3.0	1.4	3.8	4.4	-1.6	0.8	1.4
3	4.7	4.5	5.8	4.7	-0.2	1.1	0.0
4	5.5	4.3	5.6	4.8	-1.2	0.1	-0.7
5	6.2	6.1	6.1	6.7	-0.1	-0.1	0.5
6	3.2	6.6	7.6	8.3	3.4	4.4	5.1
7	2.5	6.2	8.0	8.2	3.7	5.5	5.7
8	2.8	3.6	4.4	4.3	0.8	1.6	1.5
9	1.1	1.1	5.7	5.8	0.0	4.6	4.7
10	2.9	4.9	6.3	6.4	2.0	3.4	3.5

Übersicht

- Formal ist ein Dataframe eine Liste, deren Elemente Vektoren gleicher Länge sind.
- Die Listenelemente entsprechen den Spalten einer Tabelle.
- Die Vektorelement gleicher Position entsprechen den Zeilen einer Tabelle.

```
D = data.frame(c(1,2,3),
               c("a", "b", "c"),
               c(T,F,F))
```



Erzeugung

`data.frame()` erzeugt einen Dataframe

```
D = data.frame(x = letters[1:4],     # 1. Spalte mit Name x
               y = 1:4,              # 2. Spalte mit Name y
               z = c(T,T,F,T))       # 3. Spalte mit Name z
print(D)
```

```
>   x y     z
> 1 a 1  TRUE
> 2 b 2  TRUE
> 3 c 3 FALSE
> 4 d 4  TRUE
```

Die Spalten des Dataframes müssen gleiche Länge haben

```
D = data.frame(x = letters[1:4],     # 1. Spalte mit Name x
               y = 1:4,              # 2. Spalte mit Name y
               z = c(T,T,F))         # 3. Spalte mit Name z
```

```
> Error in data.frame(x = letters[1:4], y = 1:4, z = c(T, T, F)): Argumente implizieren unterschiedliche A
```

Die Spalten eines Dataframes können offenbar unterschiedlichen Typs sein.

Charakterisierung

Ein Dataframe hat `names()`, `rownames()`, `colnames()`

```
D = data.frame(age    = c(30,35,40,45),      # 1. Spalte
               height = c(178,189,165,171), # 2. Spalte
               weight = c(67, 76, 81, 92))  # 3. Spalte
names(D)                                    # names gibt die Spaltennamen aus
```

```
> [1] "age"    "height" "weight"
```

```
colnames(D)                                 # colnames entspricht names
```

```
> [1] "age"    "height" "weight"
```

```
rownames(D)                                 # default rownames sind 1,2,...
```

```
> [1] "1" "2" "3" "4"
```

Ein Dataframe `nrow()` Zeilen und `length()` bzw. `ncol()` Spalten

```
nrow(D)                                     # Zeilenanzahl
```

```
> [1] 4
```

```
ncol(D)                                     # Spaltenanzahl
```

```
> [1] 3
```

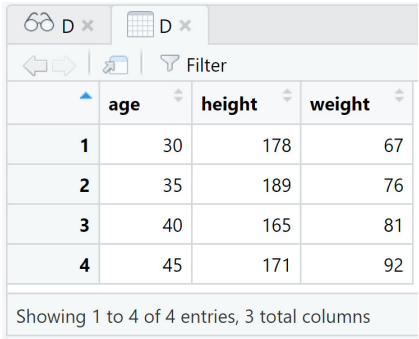
```
length(D)                                   # Länge ist die Spaltenanzahl
```

```
> [1] 3
```

Charakterisierung

`View()` öffnet den RStudio Data Viewer.

```
View(D)
```



`str()` zeigt in kompakter Form wesentliche Aspekte eines Dataframes an.

```
str(D)
```

```
> 'data.frame': 4 obs. of  3 variables:
>  $ age    : num  30 35 40 45
>  $ height: num  178 189 165 171
>  $ weight: num  67 76 81 92
```

Allgemein zeigt `str()` in kompakter Form wesentliche Aspekte eines R Objektes an.

Attribute

- Dataframes sind Listen mit Attributen für (column) `names` und `row.names`
- Dataframes haben `class` "data.frame"

```
typeof(D)
```

```
> [1] "list"
```

```
attributes(D)
```

```
> $names
> [1] "age"    "height" "weight"
>
> $class
> [1] "data.frame"
>
> $row.names
> [1] 1 2 3 4
```

Indizierung

Die Prinzipien der Indizierung für Vektoren und Matrizen gelten auch für Dataframes

⇒ Bei einem Index verhalten sich Dataframes wie Listen

```
D = data.frame(x = letters[1:4],    # 1. Spalte mit Name x
               y = 1:4,             # 2. Spalte mit Name y
               z = c(T,T,F,T))      # 3. Spalte mit Name z
class(D)                            # D ist ein Dataframe
```

```
> [1] "data.frame"
```

```
v = D[1]                            # 1. Listenelement als Dataframe
v
```

```
>   x
> 1 a
> 2 b
> 3 c
> 4 d
```

```
class(v)                            # v ist ein Dataframe
```

```
> [1] "data.frame"
```

Indizierung

Die Prinzipien der Indizierung für Vektoren und Matrizen gelten auch für Dataframes

$\Rightarrow$ Bei einem Index verhalten sich Dataframes wie Listen

```
D = data.frame(x = letters[1:4],     # 1. Spalte mit Name x
               y = 1:4,              # 2. Spalte mit Name y
               z = c(T,T,F,T))       # 3. Spalte mit Name z
w = D[[1]]                           # Inhalt des 1. Listenelements
w
```

```
> [1] "a" "b" "c" "d"
```

```
class(w)                             # w ist ein character vector
```

```
> [1] "character"
```

```
y = D$y                              # $ zur Indizierung der y Spalte
y
```

```
> [1] 1 2 3 4
```

```
class(y)                             # v ist ein Vektor vom Typ "integer" (!)
```

```
> [1] "integer"
```

Indizierung

Die Prinzipien der Indizierung für Vektoren und Matrizen gelten auch für Dataframes

$\Rightarrow$ Bei zwei Indices verhalten sich Dataframes wie Matrizen

```
D = data.frame(x = letters[1:4],     # 1. Spalte mit Name x
               y = 1:4,              # 2. Spalte mit Name y
               z = c(T,T,F,T))       # 3. Spalte mit Name z
```

```
D[2:3,-2]                            # 1. Index fuer Zeilen, 2. Index fuer Spalten
```

```
>   x     z
> 2 b  TRUE
> 3 c FALSE
```

```
D[c(T,F,T,F),]                       # 1. Index fuer Zeilen, 2. Index fuer Spalten
```

```
>   x y     z
> 1 a 1  TRUE
> 3 c 3 FALSE
```

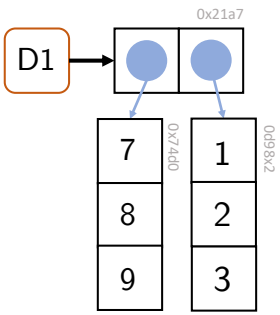
```
D[,c("x", "z")]                      # 1. Index fuer Zeilen, 2. Index fuer Spalten
```

```
>   x     z
> 1 a  TRUE
> 2 b  TRUE
> 3 c FALSE
> 4 d  TRUE
```

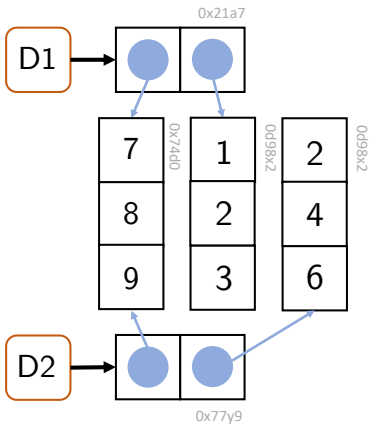
Copy-on-modify

- Die Copy-on-Modify Prinzipien für Listen gelten auch für Dataframes
- Modifikation einer Spalte führt zur Kopie der entsprechenden Spalte
- Modifikation einer Zeile führt zur Kopie des gesamten Dataframes

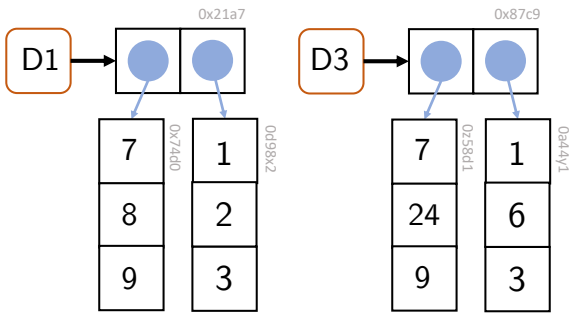
D1 = data.frame(
c(7,8,9),c(1,2,3))



D2 = D1
D2[,1] = D2[,1] * 2



D3 = D1
D3[2,] = D3[2,] * 3



Listen

Dataframes

Übungen und Selbstkontrollfragen

Übungen und Selbstkontrollfragen

1. Dokumentieren Sie die in dieser Einheit eingeführten R Befehle.
2. Beschreiben Sie in einer Übersicht die R Datenstruktur “List”.
3. Erzeugen Sie eine Liste mit vier Elementen.
4. L sei eine Liste. Was ist der Unterschied zwischen L[1] und L[[1]]?
5. Erzeugen Sie zwei Listen und fügen Sie diese zusammen.
6. L sei eine Liste. Was gibt length(L) an?
7. L sei eine Liste. Was bedeutet dann L$Student?
8. Erläutern Sie den Begriff “Shallow Copy” einer Liste.
9. Beschreiben Sie in einer Übersicht die R Datenstruktur "Dataframe".
10. Erzeugen Sie einen Dataframe mit vier Spalten.
11. D sei ein Dataframe. Was geben rownames(D) und colnames(D) an?
12. D sei ein Dataframe. Was ist der Unterschied zwischen D[1] und D[1,1]?
13. D sei ein Dataframe. Was bedeutet dann D$Student?
14. Erläutern Sie das Copy-on-modify Prinzip für Dataframes.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(6) Datenmanagement

Datum	Einheit	Thema
14.10.2021	Einführung	(1) Einführung
21.10.2021	R Grundlagen	(2) R und RStudio I
28.10.2021	R Grundlagen	(2) R und RStudio II
04.11.2021	R Grundlagen	(3) Vektoren
11.11.2021	R Grundlagen	(4) Matrizen
18.11.2021	R Grundlagen	(5) Listen und Dataframes
25.11.2021	**Deskriptive Statistik**	**(6) Datenmanagement**
02.12.2021	Deskriptive Statistik	(7) Häufigkeitsverteilungen
09.12.2021	Deskriptive Statistik	(9) Verteilungsfunktionen und Quantile
16.12.2021	Deskriptive Statistik	(10) Maße der zentralen Tendenz
	Weihnachtspause	
06.01.2022	Deskriptive Statistik	(11) Maße der Datenvariabilität
13.01.2022	Deskriptive Statistik	(12) Bivariate Zusammenhangsmaße
20.01.2022	Deskriptive Statistik	(13) R Base Graphics
27.01.2022	Deskriptive Statistik	(14) Q & A
25.02.2022	Klausurtermin	G26 – H1, 9:00 - 10:00 Uhr
Jul 2022	Klausurwiederholungstermin	

FAIR Prinzipien

Datenformate

Verzeichnismanagement

Datenimport und Datenexport

Übungen und Selbstkontrollfragen

FAIR Prinzipien

Datenformate

Verzeichnismanagement

Datenimport und Datenexport

Übungen und Selbstkontrollfragen

Daten

- Zahlenarrays
- Characterarrays
- Software
- Digitale Werkzeuge
- Workflows
- Analysispipelines
- u.v.a.m.

Forschungsdaten

"Grundsätzlich handelt es sich bei Forschungsdaten um elektronisch repräsentierte analoge oder digitale Daten, die im Zuge wissenschaftlicher Vorhaben entstehen oder genutzt werden, z.B. durch Beobachtungen, Experimente, Simulationsrechnungen, Erhebungen, Befragungen, Quellenforschungen, Aufzeichnungen von Audio- und Videosequenzen, Digitalisierung von Objekten, und Auswertungen."' ' "

Rat für Informationsinfrastrukturen

Empfehlungen zur Nutzung und Verwertung von Daten im wissenschaftlichen Raum (09/2021)

Herausforderung Datenqualität (11/2019)

Digitale Kompetenzen – dringend gesucht! (07/2019)

Aktuelle Empfehlungen zu Datenschutz und Forschungsdaten (03/2017)

Metadaten

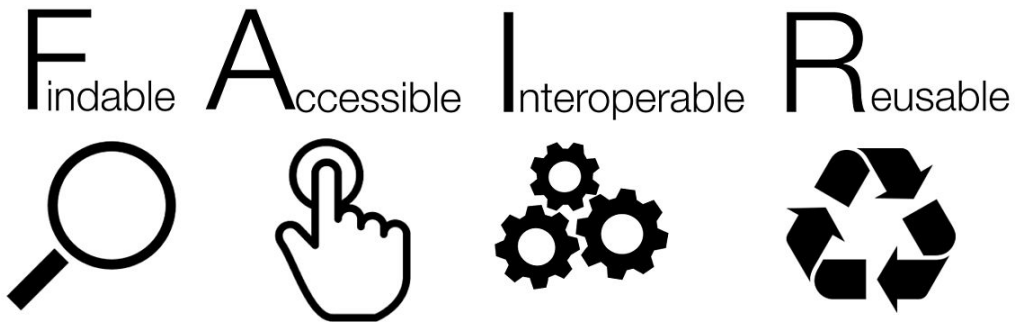
Metadaten repräsentieren Information über Daten

Deskriptive Metadaten dienen dem Auffinden und der Identifikation einer Datenquelle. Beispiele für deskriptive Metadaten sind Titel, Abstrakt, Autor:in, oder Keywords einer wissenschaftlichen Publikationen.

Strukturelle Metadaten sind Metadaten über Datencontainer und repräsentieren den strukturellen Aufbau einer Datenquelle. Beispiele sind die Ordnung der Seiten eines Buches, oder die Schleifenenkodierung dreidimensionaler Datenobjekte.

Administrative Metadaten sind Daten, die das Management einer Datenquelle erleichtern. Beispiele sind die Provenienz, das Dateiformat, die Zugangsrechte, oder weitere technische Informationen zu einer Datenquelle.

Das FAIR Datenideal



für Menschen und Maschinen

Ursprünge und Dokumentation

„Jointly designing a data fairport“ workshop in Leiden 2014

FORCE11

Wilkinson et al.(2016) The FAIR Guiding Principles for scientific data management and stewardship Scientific Data 160018

go-fair.org/FAIR Principles

Findability (Auffindbarkeit)

F1. (Meta)Daten haben einen persistenten global einzigartigen Identifikator.

F2. Daten werden mit Metadaten angereichert.

F3. Metadaten sind zweifelsfrei einem Datensatz zuzuordnen.

F4. (Meta)Daten sind in einer durchsuchbaren Ressource indexiert.

Accessibility (Zugänglichkeit)

A1. (Meta)Daten sind mit standardisierten Protokollen abrufbar.

A1.1. Das genutzte Protokoll ist offen, kostenlos und nutzbar.

A1.2. Das Protokoll ermöglicht Authentifizierung und Rechtevergabe.

A2. Metadaten bleiben zugänglich, auch wenn Daten nicht mehr vorliegen.

Interoperability (Interoperabilität)

I1. (Meta)Daten nutzen eine formale, zugängliche, gemeinsam genutzte und breit anwendbare Sprache zur Wissensrepräsentation.

I2. (Meta)Daten nutzen Vokabularien, die den FAIR-Prinzipien folgen.

I3. (Meta)Daten enthalten qualifizierte Referenzen auf andere (Meta)Daten.

Reusability (Wiederverwendbarkeit)

R1. (Meta)Daten haben eine Vielzahl genauer und relevanter Attribute.

R1.1. (Meta)Daten enthalten eine eindeutige Nutzungslizenz.

R1.2. (Meta)Daten enthalten detaillierte Provenienz-Informationen.

R1.3. (Meta)Daten genügen den Standards der jeweiligen Fachcommunity.

Fazit

- Die FAIR Prinzipien sind ein anzustrebendes Datenmanagementideal.
- Der Umgang mit digitalen Forschungsdaten ist oft noch sehr unstrukturiert.
- Die Universitäten begreifen das digitale Datenmanagement nur sehr langsam.
- Die Digitalisierung bleibt eine gesellschaftliche Hauptaufgabe.
- Die NFDI Initiative versucht, deutsches Wissenschaftsdatenmanagement zu verbessern.
- Beteiligung von OVGU // CBBS an NFDI Neurowissenschaft.
- NFDI ist dezentral, community, und Drittmittelprojekt-basiert $\Rightarrow$ Nicht nachhaltig.
- Nicht alle Wissenschaftler:innen wollen ihre Daten organisieren und teilen.
- Open Science bleibt eine wichtige Initiative verantwortungsvoller Wissenschaftler:innen.

FAIR Prinzipien

Datenformate

Verzeichnismanagement

Datenimport und Datenexport

Übungen und Selbstkontrollfragen

Dateiformate

- Ein Dateiformat definiert Syntax und Semantik von Daten innerhalb einer Datei.
- Dateiformate sind bijektive Abbildungen von Information auf binären Speicher.
- Allgemein unterscheidet man
  - Daten- gegenüber Softwareformaten,
  - textuelle gegenüber binären Dateiformaten, und
  - offene gegenüber propietären (urheberrechtlich geschützten) Dateiformaten.

Binäre Dateiformate

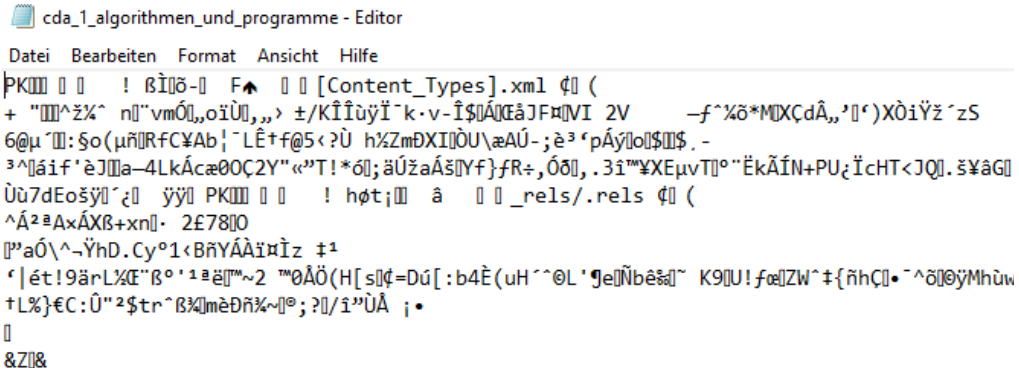
- Einlesen, Inspektion, und Manipulation ist nur mit spezieller Software möglich.
- .pdf, .xlsx, .jpg, .mp4 sind binäre Dateiformate.
- Binäre Dateiformate sind oft propietär.
- Binäre Dateiformate wurden früher aufgrund ihrer kleineren Größe bevorzugt eingesetzt.

Textuelle Dateiformate

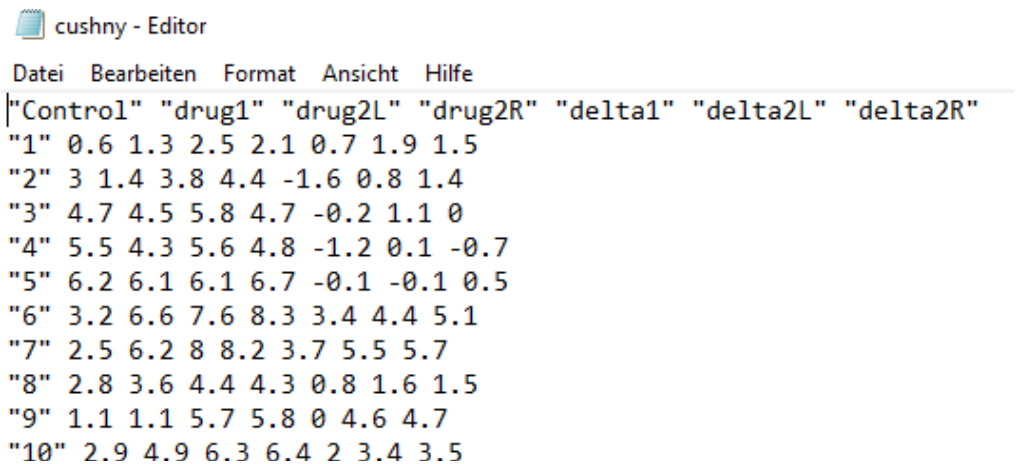
- Einlesen, Inspektion, und Manipulation ist mit einfachen allgemeinen Editoren möglich.
- .txt, .csv., .tsv, .json sind textuelle Dateiformate.
- Textuelle Dateiformate sind generell offene Dateiformate.

Datenformate

Binäres Dateiformat



Textuelles Dateiformat



Textuelle Dateiformate | CSV

- CSV = Comma- (oder auch character)-separated values, Dateiendung .csv
- Zentrales Format zur Speicherung einfach strukturierter Daten
- Repräsentation zeilenweise miteinander verknüpfter Datensätze
  - Trennung von Datenfeldern (Spalten) durch Komma oder Tab (TSV, .tsv)
  - Trennung von Datensätzen (Zeilen) durch Zeilenumbruch
- Erster Datensatz typischerweise Kopfdatensatz (Header) mit Spaltennamendefinition

Beispiel

- Einheit (experimental unit) repräsentiert z.B. eine Versuchsperson

.csv Dateiinhalt

```
Einheit, Variable 1, Variable 2
1, 10.1, 67.5
2, 12.9, 51.2
3, 20.4, 70.8
```

Tabellenrepräsentation

Einheit	Variable 1	Variable 2
1	10.1	67.5
2	12.9	51.2
3	20.4	70.8

Textuelle Dateiformate | CSV

Wide Format: Alle Variablen einer Einheit in einer Zeile

Einheit	Variable 1	Variable 2
1	10.1	67.5
2	12.9	51.2
3	20.4	70.8

Long Format: Variablen einer Einheit über Zeilen verteilt

Einheit	Variable	Messwert
1	Variable 1	10.1
1	Variable 2	67.5
2	Variable 1	12.9
2	Variable 2	51.2
3	Variable 1	20.4
3	Variable 2	70.8

Das Wide Format ist generell übersichtlicher als das Long Format

Textuelle Dateiformate | JSON

Übersicht

- JSON = JavaScript Object Notation
- Textuelles Datenformat zum Speichern strukturierter Daten in Key-Value Form.
- Ähnlichkeit mit R Listen mit benannten Listenelementen.
- Sinnvolles Format für das Speichern von Metadaten.

Elemente von JSON Dateien

- *Objekte* enthalten durch Kommata geteilte Liste von *Eigenschaften* in { }
- *Eigenschaften* bestehen aus Key-Value Paaren
- *Key* ist immer ein String mit Hochkommata “ ’ ’
- *Value* ist ein Objekt, ein Array, ein String, ein Boolean, oder eine Zahl

Textuelle Dateiformate | JSON

Beispiel

```
{
    “Vorname” : “Maxi”,
    “Nachname” : “Musterfrau”,
    “Matrikelnummer” : 12345,
    “Fachsemester” : 2,
    “Studiengang” : “BSc Psychologie”,
    “Module” :
    {
        “Deskriptive Statistik” : {“Abgeschlossen“’ : TRUE, “Note” : 1.0 },
        “Inferenzstatistik” : {“Abgeschlossen” : FALSE, “Note”: NA }
    }
}
```

FAIR Prinzipien

Datenformate

Verzeichnismanagement

Datenimport und Datenexport

Übungen und Selbstkontrollfragen

Arbeiten mit Strings

Die Grundeinheit für Text in R sind atomic vectors vom Typ character.

Die Elemente von character vectors sind strings, nicht einzelne characters.

Der Begriff "String" in R ist also nur informeller Natur.

Strings werden mit Anführungszeichen oder Hochkommata erzeugt

```
c("Dies ist ein character vector")  # Anfuehrungszeichen sind  der String Standard
```

```
> [1] "Dies ist ein character vector"
```

```
c('Dies ist ein "string"')          # Hochkommata koennen bei  Anfuehrungszeichen im String helfen
```

```
> [1] "Dies ist ein \"string\""
```

`paste()` konvertiert Vektoren in character und fügt sie elementweise zusammen.

```
paste(1,2)                          # Konvertierung und Konkatenation .einelementiger double vectors
```

```
> [1] "1 2"
```

```
paste("Dies ist", "ein String")     # Konkatenation einelementiger character strings
```

```
> [1] "Dies ist ein String"
```

Arbeiten mit Strings

`paste()` hat eine Reihe von weiteren Funktionalitäten

```
paste(c("Rote", "Gelbe"), "Blume")                    # Vector recylcing, elementweise Veknuepfungen
```

```
> [1] "Rote Blume"  "Gelbe Blume"
```

```
paste(c("Rote", "Gelbe"), "Blume",  sep = "-")        # Separatorspezifkation
```

```
> [1] "Rote-Blume"  "Gelbe-Blume"
```

```
paste(c("Rote", "Gelbe"), "Blume",  collapse = ", ") # Zusammenfuegen mit spezifiziertem  Separator
```

```
> [1] "Rote Blume, Gelbe Blume"
```

‘`toString()` ist eine `paste()` Variation für numerische Vektoren

```
toString(1:10)                          # Konversion eines double Vektors in formatierten String
```

```
> [1] "1, 2, 3, 4, 5, 6, 7, 8, 9, 10"
```

```
toString(1:10, width = 10)              # mit  Moeglichkeit der Beschraenkung auf width Zeichen
```

```
> [1] "1, 2, ...."
```

Dateipfade

- Daten sind üblicherweise in Dateien im permanenten Speicher (SSD, HD) abgelegt
- Zum Dateneinlesen benötig man ihre Addresse in der Verzeichnisstruktur des Rechners.
- Die Addressen von Dateien in der Verzeichnisstruktur heißen *Dateipfade*.
- Ein Pfad besteht aus einer durch Schrägstriche getrennten Liste von Verzeichnisnamen.

```
D:\Lehre\Daten              Pfad der auf einem Verzeichnisnamen endet

D:\Lehre\Daten\cushny.csv   Pfad der auf einem Dateinamen endet
```

- *Relative Dateipfade* bezieht sich auf einen Speicherort in Relation zum aktuellen Verzeichnis.
- Bei relativen Dateipfaden bezeichnen . und .. aktuelles und übergeordnetes Verzeichnis.
- *Absolute Dateipfade* gegeben die Addresse in der Gesamtverzeichnisstruktur der Festplatte an.
- Absolute Dateipfade sind weniger anfällig für Dateiverwechselungen.
- Die Verwendung adaptiv generierter absoluter Pfade wird stark empfohlen.

```
fname =  "D:\Lehre\Daten\cushny.csv"       # Dateiname in absoluter Pfadform
```

Working directory

R hat ein working directory aus dem per default Dateien gelesen werden.

In RStudio wird das working directory unter Tools → Global Options ... spezifiziert.

`getwd()` gibt das working directory an.

```
getwd()
```

```
> [1] "D:/Forschung und Lehre/Lehre/2022/3_Programmierung_und_Deskriptive_Statistik_21_22/6_Datenmanagemer
```

`setwd()` ändert das working directory

- Windowspfade haben backward slashes \, R arbeitet mit forward slashes /.
- Manuelle Spezifikation von Windowspfaden benötigt doppelte backward slashes \\.

```
setwd("D:\\Google Drive\\Lehre\\2022")
getwd()
```

Dateipfadspezifikation

`file.path()` konstruiert Verzeichnis- und Dateipfade.

```
file.path("D:", "Google Drive", "Lehre", "2022")
```

```
> [1] "D:/Google Drive/Lehre/2022"
```

`dirname()`gibt das Verzeichnis an, das ein Verzeichnis oder eine Datei enthält.

```
getwd()
```

```
> [1] "D:/Forschung und Lehre/Lehre/2022/3_Programmierung_und_Deskriptive_Statistik_21_22/6_Datenmanagemer
```

```
dirname(getwd())
```

```
> [1] "D:/Forschung und Lehre/Lehre/2022/3_Programmierung_und_Deskriptive_Statistik_21_22"
```

`basename()` gibt die unterste Ebene eines Datei- oder Verzeichnispfades an.

```
getwd()
```

```
> [1] "D:/Forschung und Lehre/Lehre/2022/3_Programmierung_und_Deskriptive_Statistik_21_22/6_Datenmanagemer
```

```
basename(getwd())
```

```
> [1] "6_Datenmanagement"
```

RStudio Projekte

- Um die Arbeit mit R's Working Directory zu erleichtern bietet RStudio "Projekte" an
- Die RStudio Dokumentation gibt eine Einführung
- RStudio Projekte können unter File → New Project ... erzeugt werden
- RStudio Projekte bieten unter anderem folgende Funktionalitäten
  - Im Projektverzeichnis wird eine Projektdatei (.Rproj) erstellt.
  - Die Projektdatei enthält Projekt-spezifische Metadaten.
  - Die Projektdatei kann zum Öffen von RStudio und des Projektes benutzt werden.
  - Bei Öffnung eines Projektes über die Projektdatei werden
    - das Projektverzeichnis zum Working Directory,
    - zuvor geöffnete Programmdateien im Editor geöffnet und
    - weitere RStudio Settings auf ihren Projekt-aktuellen Zustand gesetzt.

⇒ Die Arbeit mit RStudio Projekten für Sinneinheiten wird sehr empfohlen!

FAIR Prinzipien

Datenformate

Verzeichnismanagement

Datenimport und Datenexport

Übungen und Selbstkontrollfragen

Datenimport mit read.table()

`read.table()` ist die zentrale Funktion zum Einlesen von CSV Dateien.

`read.table()` liest eine Datei ein und speichert ihre Inhalte in einem Dataframe.

`read.table()` bietet eine Vielzahl weiterer Spezifikationsmöglichkeiten

```
wdir    = getwd()                    # Working directory
ddir    = file.path(wdir, "6_Data")  # Datenverzeichnispfad
fname   = "cushny.csv"               # (base) filename
fpath   = file.path(ddir, fname)     # filepath
D       = read.table(fpath)          # Einlesen der Datei
print(D)
```

```
>    Control drug1 drug2L drug2R delta1 delta2L delta2R
> 1      0.6   1.3    2.5    2.1    0.7     1.9     1.5
> 2      3.0   1.4    3.8    4.4   -1.6     0.8     1.4
> 3      4.7   4.5    5.8    4.7   -0.2     1.1     0.0
> 4      5.5   4.3    5.6    4.8   -1.2     0.1    -0.7
> 5      6.2   6.1    6.1    6.7   -0.1    -0.1     0.5
> 6      3.2   6.6    7.6    8.3    3.4     4.4     5.1
> 7      2.5   6.2    8.0    8.2    3.7     5.5     5.7
> 8      2.8   3.6    4.4    4.3    0.8     1.6     1.5
> 9      1.1   1.1    5.7    5.8    0.0     4.6     4.7
> 10     2.9   4.9    6.3    6.4    2.0     3.4     3.5
```

Datenimport mit read.table()

Einige weitere Spezifikationen bei Anwendung von `read.table()` sind

- `sep` für die Auswahl des Separators
- `dec` für die Auswahl des Dezimalpunktes
- `nrow` für die Anzahl der einzulesenden Zeilen
- `skip` für die Anzahl der am Anfang der Datei zu überspringenden Zeilen

```
D = read.table(file.path(fpath), nrow = 2)
print(D)
```

```
>   Control drug1 drug2L drug2R delta1 delta2L delta2R
> 1     0.6   1.3    2.5    2.1    0.7     1.9     1.5
> 2     3.0   1.4    3.8    4.4   -1.6     0.8     1.4
```

```
D = read.table(fpath, skip = 7)
print(D)
```

```
>   V1  V2  V3  V4  V5  V6  V7  V8
> 1  7 2.5 6.2 8.0 8.2 3.7 5.5 5.7
> 2  8 2.8 3.6 4.4 4.3 0.8 1.6 1.5
> 3  9 1.1 1.1 5.7 5.8 0.0 4.6 4.7
> 4 10 2.9 4.9 6.3 6.4 2.0 3.4 3.5
```

Import interner R Datensätze

R und R packages beinhalten eine Vielzahl von Beispieldatensätzen

Die Core R Datensätze werden aus der R Konsole mit `data()` angezeigt.

Die Datensätze in Paket P werden mit `data(package = "P")` angezeigt.

```
install.packages("psychTools")  # Installation des Pakets psychTools
data(package = "psychTools")    # Anzeige der psychTools Datensaetze
```

```
Data sets in package 'psychTools':

Damian                                  Project Talent data set from Marion Spengler and Rodica Damian
Pollack                                 Pollack et al (2012) correlation matrix for mediation example
Schutz                                  The Schutz correlation matrix example from Shapiro and ten Berge
Spengler (Damian)                       Project Talent data set from Marion Spengler and Rodica Damian
Spengler.stat (Damian)                  Project Talent data set from Marion Spengler and Rodica Damian
USAF                                    17 anthropometric measures from the USAF showing a general factor
ability                                 16 ability items scored as correct or incorrect.
ability.keys (ability)                  16 ability items scored as correct or incorrect.
affect                                  Two data sets of affect and arousal scores as a function of personality and movie
                                        conditions
all.income (income)                     US family income from US census 2008
bfi                                     25 Personality items representing 5 factors
```

Alle Datensätze werden mit `data(package = .packages(TRUE))` angezeigt.

Nach Installation und Laden eines Pakets werden Datensätze mit `data()` geladen.

```
library(psychTools)             # Laden des Paktes psychTools
data(cushny)                    # Laden des cushny Datensatzes aus psychTools
```

Beispiele für weitere Möglichkeiten des Datenimports

CSV und Text Dateien

- `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()` als `read.table()` Varianten.
- `readlines` für low-level Textdateiimport.
- `fromJSON()` aus dem Paket `rjson` für .json Dateien.

Binäre Dateien

- `read.xlsx()` und `read.xlsx2()` aus dem Paket `xlsx` für Excel .xlsx Dateien.
- `read.spss()` aus dem Paket `foreign` für SPSS .sav Dateien.
- `readMat` aus dem Paket `R.matlab` für Matlab .mat Dateien.

Webdaten und Datenbanken

- Twitterdaten können mithilfe der Pakete `rtweet` oder `twitteR` eingelesen werden.
- SQL Datenbanken können mithilfe der Pakete `DBI` und `RSQLite` abgefragt werden.

Datenexport mit write.table()

`write.table()` ist die zentrale Funktion zum Speichern von Daten in CSV Dateien.

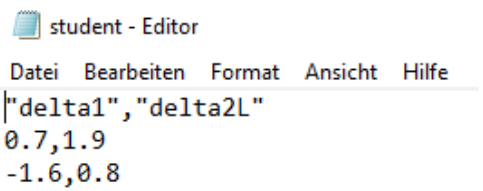
`write.table()` erzeugt eine Datei und schreibt Daten eines Dataframes hinein.

Der Dateiname wird mit dem Argument `file` angegeben, der Werteseparator mit `sep`

Das Argument `row.names = FALSE` unterdrückt das Schreiben von Zeilennahmen

```
fname       = "cushny.csv"                          # Dateiname (input)
rname         = "student.csv"                         # Dateiname (output)
D           = read.table(file.path(ddir, fname))    # Dateneinlesen
D           = D[,5:6]                               # Reduktion des Dataframes
R           = write.table(                          # .csv Schreibfunktion
                D,                                  # Zu speichernder Dataframe
                file = file.path(ddir, rname),      # Dateinname
                sep = ",",                          # Werteseparator fuer .csv
                row.names = F)                      # keine Zeilennamen
```

Ergebnisdatei student.csv



FAIR Prinzipien

Datenformate

Verzeichnismanagement

Datenimport und Datenexport

Übungen und Selbstkontrollfragen

Übungen und Selbstkontrollfragen

1. Dokumentieren Sie die in dieser Einheit eingeführten R Befehle in einem R Skript.
2. Erläutern Sie den Begriff “Forschungsdaten”.
3. Erläutern Sie den Begriff “Metadaten”.
4. Erläutern Sie das FAIR Datenideal.
5. Diskutieren Sie Unterschiede und Gemeinsamkeiten von binären und textuellen Dateien.
6. Nennen und erläutern Sie zwei textuelle Dateiformate.
7. Erläutern Sie den Unterschied zwischen dem Wide und Long Format von Tabellen.
8. Erläutern Sie den Unterschied zwischen absoluten und relativen Dateipfaden.
9. Erläutern Sie den Begriff des “Working Directories” in R.
10. Beschreiben Sie die Funktion von RStudio Projekten.
11. Nennen Sie eine R Funktion zum Einlesen von .csv Dateien.
12. Nennen Sie eine R Funktion zum Schreiben von .csv Dateien.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(7) Häufigkeitsverteilungen

Definition und Ziele der Deskriptive Statistik

- Die Deskriptive Statistik ist die *beschreibende* Statistik.
- Ziel der Deskriptiven Statistik ist es, Daten übersichtlich darzustellen.
- Deskriptive Statistik ist inbesondere bei großen Datensätzen sinnvoll.
- Die Deskriptive Statistik berechnet zusammenfassende Maße aus Daten.

Typische Methoden der Deskriptiven Statistik

- Häufigkeitsverteilungen und Histogramme
- Verteilungsfunktionen und Quantile
- Maße der zentralen Tendenz und der Datenvariabilität
- Zusammenhangsmaße

Die Deskriptive Statistik benutzt keine probabilistischen Modelle, aber die Methoden der Deskriptiven Statistik ergeben nur vor dem Hintergrund probabilistischer Modelle Sinn.

Beispieldatensatz

Häufigkeitsverteilungen

Histogramme

Übungen und Selbstkontrollfragen

Beispieldatensatz

Häufigkeitsverteilungen

Histogramme

Übungen und Selbstkontrollfragen

Evidenzbasierte Evaluation von Psychotherapieformen bei Depression

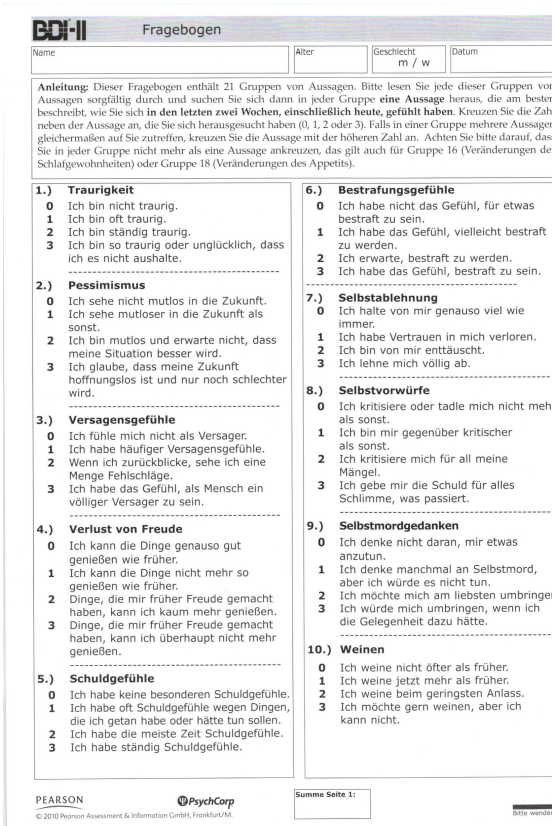
Welche Therapieform ist bei Depression wirksamer?

Online Psychotherapie

Klassische Psychotherapie

Evidenzbasierte Evaluation von Psychotherapieformen bei Depression

Becks Depressions-Inventar (BDI) zur Depressionsdiagnostik



BDI-II Fragebogen

Name | Alter | Geschlecht m / w | Datum

Anleitung: Dieser Fragebogen enthält 21 Gruppen von Aussagen. Bitte lesen Sie jede dieser Gruppen vor Aussagen sorgfältig durch und suchen Sie sich dann in jeder Gruppe **eine Aussage** heraus, die am bester beschreibt, wie Sie sich **in den letzten zwei Wochen, einschließlich heute, gefühlt haben**. Kreuzen Sie die Zah neben der Aussage an, die Sie sich herausgesucht haben (0, 1, 2 oder 3). Falls in einer Gruppe mehrere Aussagen gleichermaßen auf Sie zutreffen, kreuzen Sie die Aussage mit der höheren Zahl an. Achten Sie bitte darauf, das Sie in jeder Gruppe nicht mehr als eine Aussage ankreuzen, das gilt auch für Gruppe 16 (Veränderungen de Schlafgewohnheiten) oder Gruppe 18 (Veränderungen des Appetits).

1.) Traurigkeit
- 0 Ich bin nicht traurig.
- 1 Ich bin oft traurig.
- 2 Ich bin ständig traurig.
- 3 Ich bin so traurig oder unglücklich, dass ich es nicht aushalte.

2.) Pessimismus
- 0 Ich sehe nicht mutlos in die Zukunft.
- 1 Ich sehe mutloser in die Zukunft als sonst.
- 2 Ich bin mutlos und erwarte nicht, dass meine Situation besser wird.
- 3 Ich glaube, dass meine Zukunft hoffnungslos ist und nur noch schlechter wird.

3.) Versagensgefühle
- 0 Ich fühle mich nicht als Versager.
- 1 Ich habe häufiger Versagensgefühle.
- 2 Wenn ich zurückblicke, sehe ich eine Menge Fehlschläge.
- 3 Ich habe das Gefühl, als Mensch ein völliger Versager zu sein.

4.) Verlust von Freude
- 0 Ich kann die Dinge genauso gut genießen wie früher.
- 1 Ich kann die Dinge nicht mehr so genießen wie früher.
- 2 Dinge, die mir früher Freude gemacht haben, kann ich kaum mehr genießen.
- 3 Dinge, die mir früher Freude gemacht haben, kann ich überhaupt nicht mehr genießen.

5.) Schuldgefühle
- 0 Ich habe keine besonderen Schuldgefühle.
- 1 Ich habe oft Schuldgefühle wegen Dingen, die ich getan habe oder hätte tun sollen.
- 2 Ich habe die meiste Zeit Schuldgefühle.
- 3 Ich habe ständig Schuldgefühle.

6.) Bestrafungsgefühle
- 0 Ich habe nicht das Gefühl, für etwas bestraft zu sein.
- 1 Ich habe das Gefühl, vielleicht bestraft zu werden.
- 2 Ich erwarte, bestraft zu werden.
- 3 Ich habe das Gefühl, bestraft zu sein.

7.) Selbstablehnung
- 0 Ich halte von mir genauso viel wie immer.
- 1 Ich habe Vertrauen in mich verloren.
- 2 Ich bin von mir enttäuscht.
- 3 Ich lehne mich völlig ab.

8.) Selbstvorwürfe
- 0 Ich kritisiere oder tadle mich nicht meh als sonst.
- 1 Ich bin mir gegenüber kritischer als sonst.
- 2 Ich kritisiere mich für all meine Mängel.
- 3 Ich gebe mir die Schuld für alles Schlimme, was passiert.

9.) Selbstmordgedanken
- 0 Ich denke nicht daran, mir etwas anzutun.
- 1 Ich denke manchmal an Selbstmord, aber ich würde es nicht tun.
- 2 Ich möchte mich am liebsten umbringe
- 3 Ich würde mich umbringen, wenn ich die Gelegenheit dazu hätte.

10.) Weinen
- 0 Ich weine nicht öfter als früher.
- 1 Ich weine jetzt mehr als früher.
- 2 Ich weine beim geringsten Anlass.
- 3 Ich möchte gern weinen, aber ich kann nicht.

PEARSON PsychCorp © 2010 Pearson Assessment & Information GmbH, Frankfurt/M.

Summe Seite 1: | Bitte wende

11.) Unruhe
- 0 Ich bin nicht unruhiger als sonst.
- 1 Ich bin unruhiger als sonst.
- 2 Ich bin so unruhig, dass es mir schwerfällt, still zu sitzen.
- 3 Ich bin so unruhig, dass ich mich ständig bewegen oder etwas tun muss.

12.) Interessenverlust
- 0 Ich habe das Interesse an anderen Menschen oder an Tätigkeiten nicht verloren.
- 1 Ich habe weniger Interesse an anderen Menschen oder an Dingen als sonst.
- 2 Ich habe das Interesse an anderen Menschen oder Dingen zum größten Teil verloren.
- 3 Es fällt mir schwer, mich überhaupt für irgend etwas zu interessieren.

13.) Entschlussunfähigkeit
- 0 Ich bin so entschlussfreudig wie immer.
- 1 Es fällt mir schwerer als sonst, Entscheidungen zu treffen.
- 2 Es fällt mir sehr viel schwerer als sonst, Entscheidungen zu treffen.
- 3 Ich habe Mühe, überhaupt Entscheidungen zu treffen.

14.) Wertlosigkeit
- 0 Ich fühle mich nicht wertlos.
- 1 Ich halte mich für weniger wertvoll und nützlich als sonst.
- 2 Verglichen mit anderen Menschen fühle ich mich viel weniger wert.
- 3 Ich fühle mich völlig wertlos.

15.) Energieverlust
- 0 Ich habe so viel Energie wie immer.
- 1 Ich habe weniger Energie als sonst.
- 2 Ich habe so wenig Energie, dass ich kaum noch etwas schaffe.
- 3 Ich habe keine Energie mehr, um überhaupt noch etwas zu tun.

16.) Veränderungen der Schlafgewohnheiten
- 0 Meine Schlafgewohnheiten haben sich nicht verändert.
- 1a Ich schlafe etwas mehr als sonst
- 1b Ich schlafe etwas weniger als sonst.
- 2a Ich schlafe viel mehr als sonst.
- 2b Ich schlafe viel weniger als sonst.
- 3a Ich schlafe fast den ganzen Tag.
- 3b Ich wache 1-2 Stunden früher auf als gewöhnlich und kann dann nicht mehr einschlafen.

17.) Reizbarkeit
- 0 Ich bin nicht reizbarer als sonst.
- 1 Ich bin reizbarer als sonst.
- 2 Ich bin viel reizbarer als sonst.
- 3 Ich fühle mich dauernd gereizt.

18.) Veränderungen des Appetits
- 0 Mein Appetit hat sich nicht verändert.
- 1a Mein Appetit ist etwas schlechter als sonst.
- 1b Mein Appetit ist etwas größer als sonst.
- 2a Mein Appetit ist viel schlechter als sonst.
- 2b Mein Appetit ist viel größer als sonst.
- 3a Ich habe überhaupt keinen Appetit.
- 3b Ich habe ständig Heißhunger.

19.) Konzentrationsschwierigkeiten
- 0 Ich kann mich so gut konzentrieren wie immer.
- 1 Ich kann mich nicht mehr so gut konzentrieren wie sonst.
- 2 Es fällt mir schwer, mich längere Zeit auf irgend etwas zu konzentrieren.
- 3 Ich kann mich überhaupt nicht mehr konzentrieren.

20.) Ermüdung oder Erschöpfung
- 0 Ich fühle mich nicht müder oder erschöpfter als sonst.
- 1 Ich werde schneller müde oder erschöpft als sonst.
- 2 Für viele Dinge, die ich üblicherweise tue, bin ich zu müde oder erschöpft.
- 3 Ich bin so müde oder erschöpft, dass ich fast nichts mehr tun kann.

21.) Verlust an sexuellem Interesse
- 0 Mein Interesse an Sexualität hat sich in letzter Zeit nicht verändert.
- 1 Ich interessiere mich weniger für Sexualität als früher.
- 2 Ich interessiere mich jetzt viel weniger für Sexualität.
- 3 Ich habe das Interesse an Sexualität völlig verloren.

Summe Seite 2:

Übertrag Seite 1: | Gesamt Seite 1+2:

0 - 8 keine Depression

9 - 13 minimale Depression

14 - 19 leichte Depression

20 - 28 mittelschwere Depression

29 - 63 schwere Depression

Beispiel: Evaluation von Psychotherapieformen bei Depression

Experimentelle Bedingung
(Gruppen von n = 50)

Psychotherapie

Klassisch

Pre-BDI

Post-BDI

Online

Pre-BDI

Post-BDI

Beispieldatensatz

Einlesen des Datensatzes mit `read.table()`

```
fname       = file.path(getwd(), "7_Daten", "psychotherapie_datensatz.csv")
D           = read.table(fname, sep = ",")
print(D)
```

```
>      V1        V2      V3       V4
> 1    NA Bedingung Pre BDI Post BDI
> 2     1 Klassisch      17        9
> 3     2 Klassisch      20       14
> 4     3 Klassisch      16       13
> 5     4 Klassisch      18       12
> 6     5 Klassisch      21       12
> 7     6 Klassisch      17       14
> 8     7 Klassisch      17       12
> 9     8 Klassisch      17        9
> 10    9 Klassisch      18       11
> 11   10 Klassisch      18       14
> 12   11 Klassisch      20       10
> 13   12 Klassisch      17       15
> 14   13 Klassisch      16       17
> 15   14 Klassisch      18       12
> 16   15 Klassisch      16       10
> 17   16 Klassisch      18       13
> 18   17 Klassisch      17        9
> 19   18 Klassisch      14       13
> 20   19 Klassisch      18       12
```

Beispieldatensatz

Datensatzübersicht mit `View()`

D

Filter

	Bedingung	Pre BDI	Post BDI
1	Klassisch	17	9
2	Klassisch	20	14
3	Klassisch	16	13
4	Klassisch	18	12
5	Klassisch	21	12
6	Klassisch	17	14
7	Klassisch	17	12
8	Klassisch	17	9
9	Klassisch	18	11
10	Klassisch	18	14
11	Klassisch	20	10
12	Klassisch	17	15
13	Klassisch	16	17
14	Klassisch	18	12
15	Klassisch	16	10
16	Klassisch	18	13

Beispieldatensatz

Häufigkeitsverteilungen

Histogramme

Übungen und Selbstkontrollfragen

Definition (Absolute und relative Häufigkeitsverteilungen)

$x := (x_1, ..., x_n)$ mit $x_i \in \mathbb{R}$ sei ein *Datensatz* (manchmal auch "Urliste" genannt) und $A := \{a_1, ..., a_k\}$ mit $k \leq n$ seien die im Datensatz vorkommenden verschiedenen Zahlenwerte (manchmal auch "Merkmalsausprägungen" genannt). Dann heißt die Funktion

$$h : A \to \mathbb{N}, a \mapsto h(a) := \text{Anzahl der } x_i \text{ aus } x \text{ mit } x_i = a \tag{1}$$

die *absolute Häufigkeitsverteilung* der Zahlwerte von x und die Funktion

$$r : A \to [0, 1], a \mapsto r(a) := \frac{h(a)}{n} \tag{2}$$

die *relative Häufigkeitsverteilung* der Zahlwerte von x.

Bemerkungen

- Absolute und relative Häufigkeitsverteilungen fassen Datensätze zusammen
- Absolute und relative Häufigkeitsverteilungen können einen ersten Datenüberblick geben

Häufigkeitsverteilungen

Erzeugen der absoluten Häufigkeitsverteilung mit `table()`

Erzeugen der relativen Häufigkeitsverteilung durch Division mit n

```
x           = D$Pre.BDI                    # Double vector der Pre BDI Werte
n           = length(x)                    # Anzahl der Datenwerte (100)
H           = as.data.frame(table(x))      # absolute Haeufigkeitsverteilung (dataframe)
names(H)    = c("a", "h")                  # Spaltenbenennung
H$r         = H$h/n                        # relative Haeufigkeitsverteilung
print(H, digits = 1)                       # Ausgabe
```

Häufigkeitsverteilungen

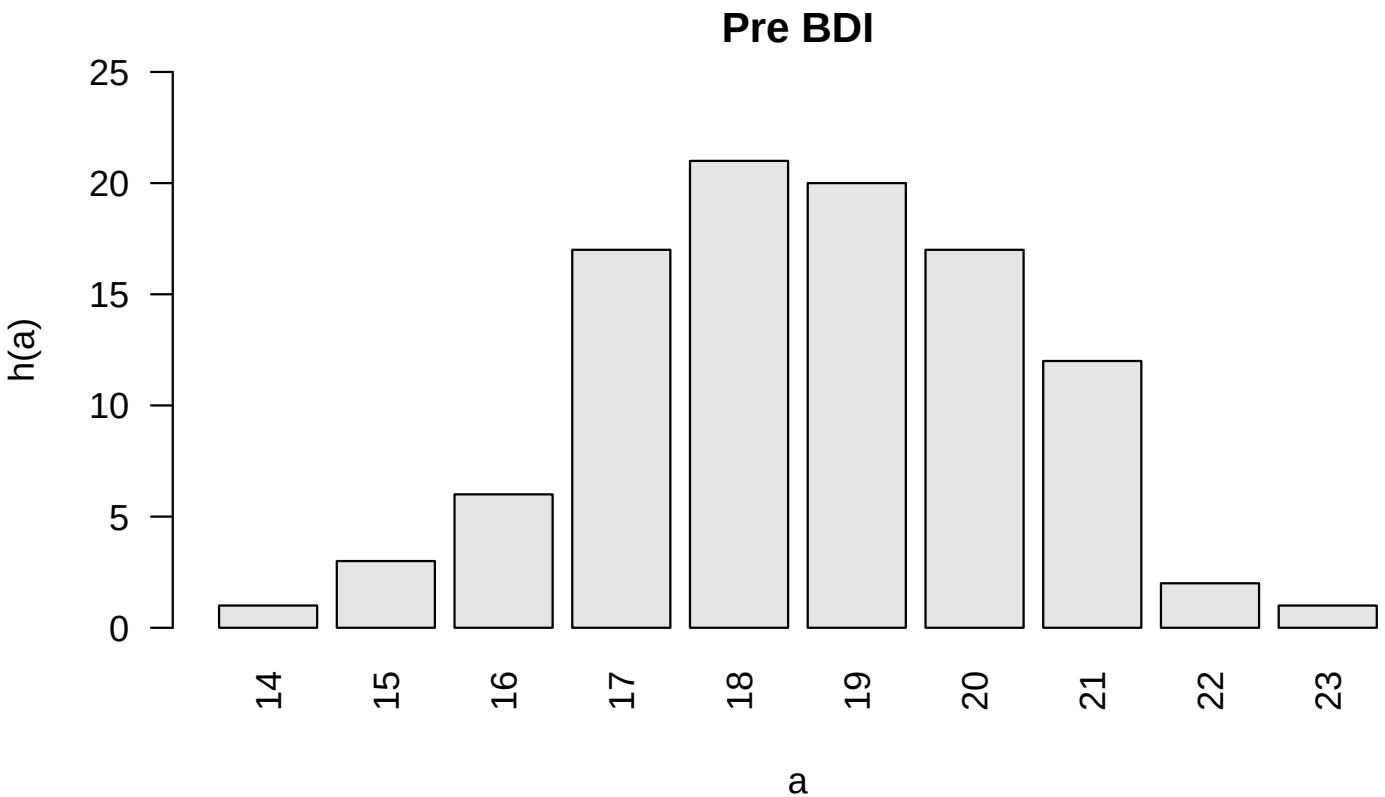
Visualisierung der absoluten Häufigkeitsverteilung mit `barplot()`

```
h           = H$h                                   # h(a) Werte
names(h)    = H$a                                   # barplot braucht a Werte als names
dev.new()                                           # Abbildungsinitialisierung
barplot(                                            # Balkendiagramm
h,                                                  # absolute Haeufigkeiten
col         = "gray90",                             # Balkenfarbe
xlab        = "a",                                  # x Achsenbeschriftung
ylab        = "h(a)",                               # y Achsenbeschriftung
ylim        = c(0,25),                              # y Achsengrenzen
las         = 2,                                    # x Tick Orientierung
main        = "Pre BDI")                            # Titel
```

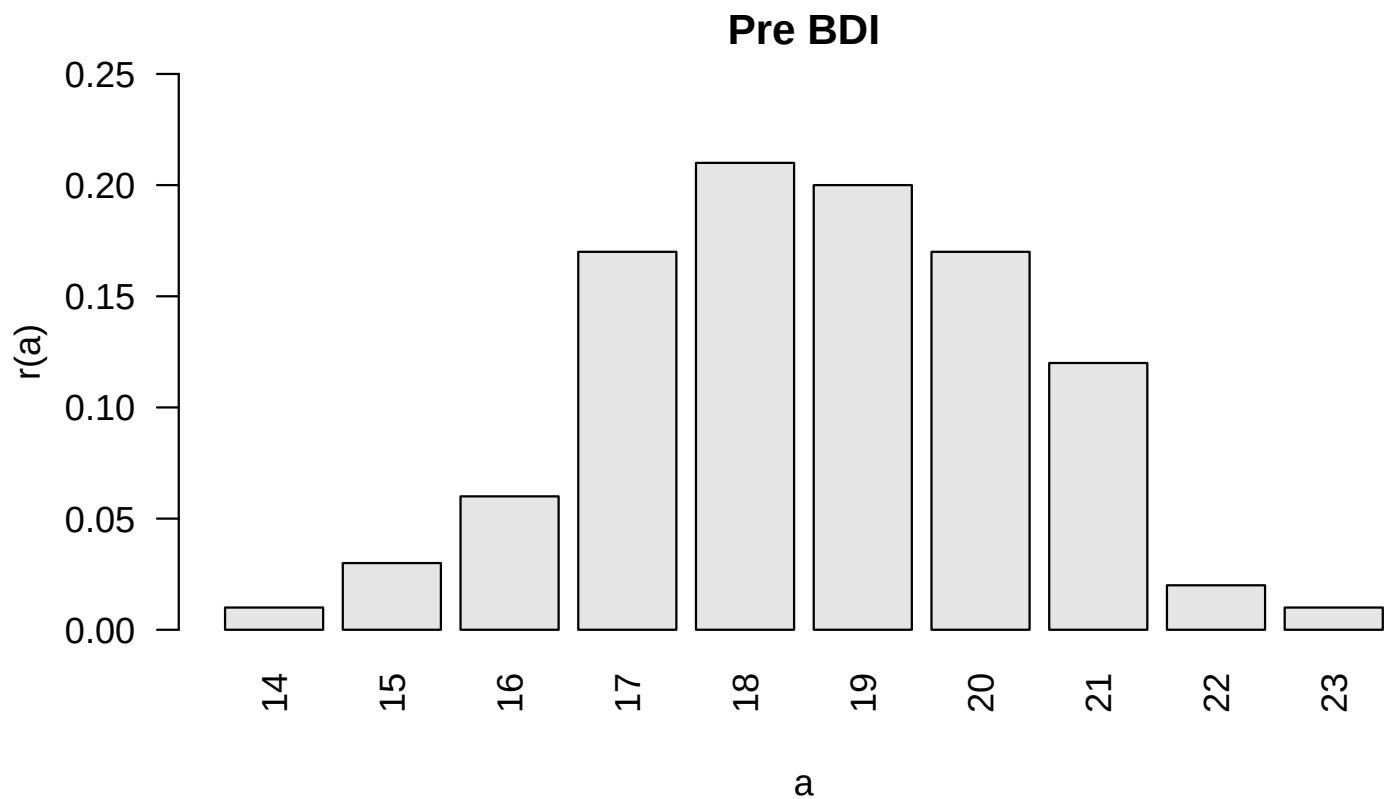
Speichern von Abbildungen mit `dev.copy2pdf()`

```
fdir        = file.path(getwd(), "7_Abbildungen")   # Abbildungsverzeichnis
dev.copy2pdf(                                       # PDF Kopiefunktion
file        = file.path(fdir, "pds_7_ha_prebdi.pdf"), # Dateiname
width       = 7,                                    # Breite (inch)
height      = 4)                                    # Höhe (inch)
```

Absolute Häufigkeitsverteilung aller Pre-BDI Werte



Relative Häufigkeitsverteilung aller Pre-BDI Werte



Beispieldatensatz

Häufigkeitsverteilungen

Histogramme

Übungen und Selbstkontrollfragen

Definition (Histogramm)

Ein *Histogramm* ist ein Diagramm, in dem zu einem Datensatz $x = (x_1, ..., x_n)$ mit verschiedenen Zahlwerten $A := \{a_1, ..., a_m\}, m \leq n$ über benachbarten Intervallen $[b_{j-1}, b_j[$, welche *Klassen* oder *Bins* genannt werden, für $j = 1, ..., k$ Rechtecke mit

Breite	$d_j = b_j - b_{j-1}$
Höhe	$h(a)$ oder $r(a)$ mit $a \in [b_{j-1}, b_j[$

abgebildet sind, wobei $b_0 := \min A$ und $b_k := \max A$ angenommen werden soll.

Bemerkungen

- Das Aussehen eines Histogramms ist stark von der Anzahl k der Klassen abhängig.
- Mit der Aufrundungsfunktion $\lceil \cdot \rceil$ sind konventionelle Werte für k

$k := \lceil (b_k - b_0)h \rceil$	h ist die gewünschte Klassenbreite
$k := \lceil \sqrt{n} \rceil$	Excelstandard
$k := \lceil \log_2 n + 1 \rceil$	Implizite Normalverteilungsannahme (Sturges, 1926)
$k := 3.49 S_n / \sqrt[3]{n}$	Min. MSE Dichteschätzung bei Normalverteilung (Scott, 1979)

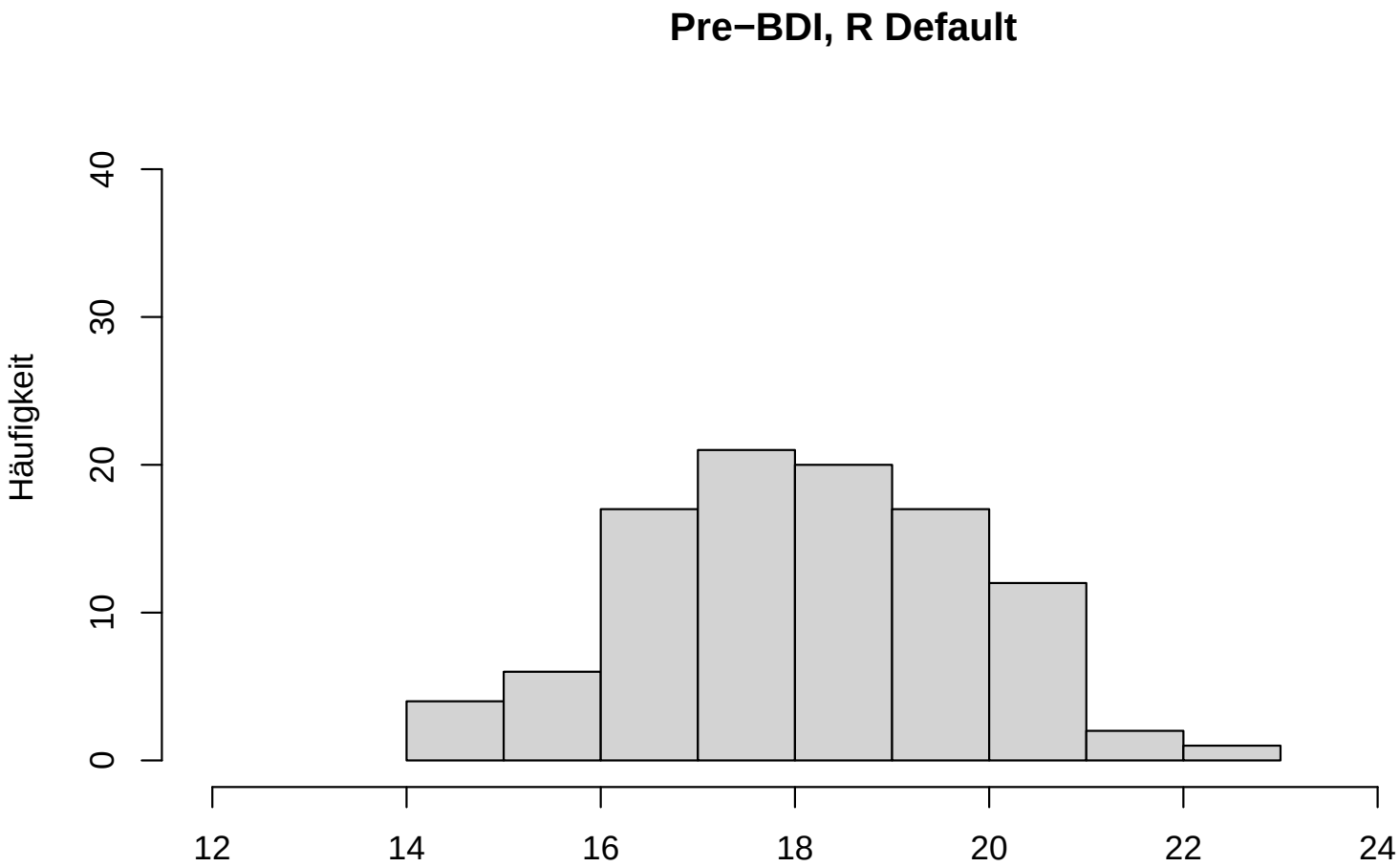
Berechnung und Visualisierung von Histogrammen mit `hist()`

- Die Klassen $[b_{j-1}, b_j[, j = 1, \ldots, k$ werden als Argument `breaks` festgelegt
- `breaks` ist der atomic vector c$(b_0, b_1, \ldots, b_k)$ mit Länge $k + 1$
- Per default benutzt `hist()` eine Modifikation der Sturges Empfehlung $k = \lceil \log_2 n + 1 \rceil$
- `hist()` bietet eine Vielzahl weiterer Spezifikationsmöglichkeiten

```
# Default Histogramm
x       = D$Pre.BDI                          # Datensatz
x_min   = 12                                 # x Achsengrenze (unten)
x_max   = 25                                 # x Achsengrenze (oben)
y_min   = 0                                  # y Achsengrenze (oben)
y_max   = 30                                 # y Achsengrenze (unten)
hist(                                        # Histogramm
x,                                           # Datensatz
xlim    = c(x_min, x_max),                   # x Achsengrenzen
ylim    = c(y_min, y_max),                   # y Achsengrenzen
ylab    = "Häufigkeit",                      # y-Achsenbezeichnung
xlab    = "",                                # x-Achsenbezeichnung
main    = "Pre-BDI, R Default")              # Titel
```

Histogramme

Berechnung und Visualisierung von Histogrammen mit `hist()`

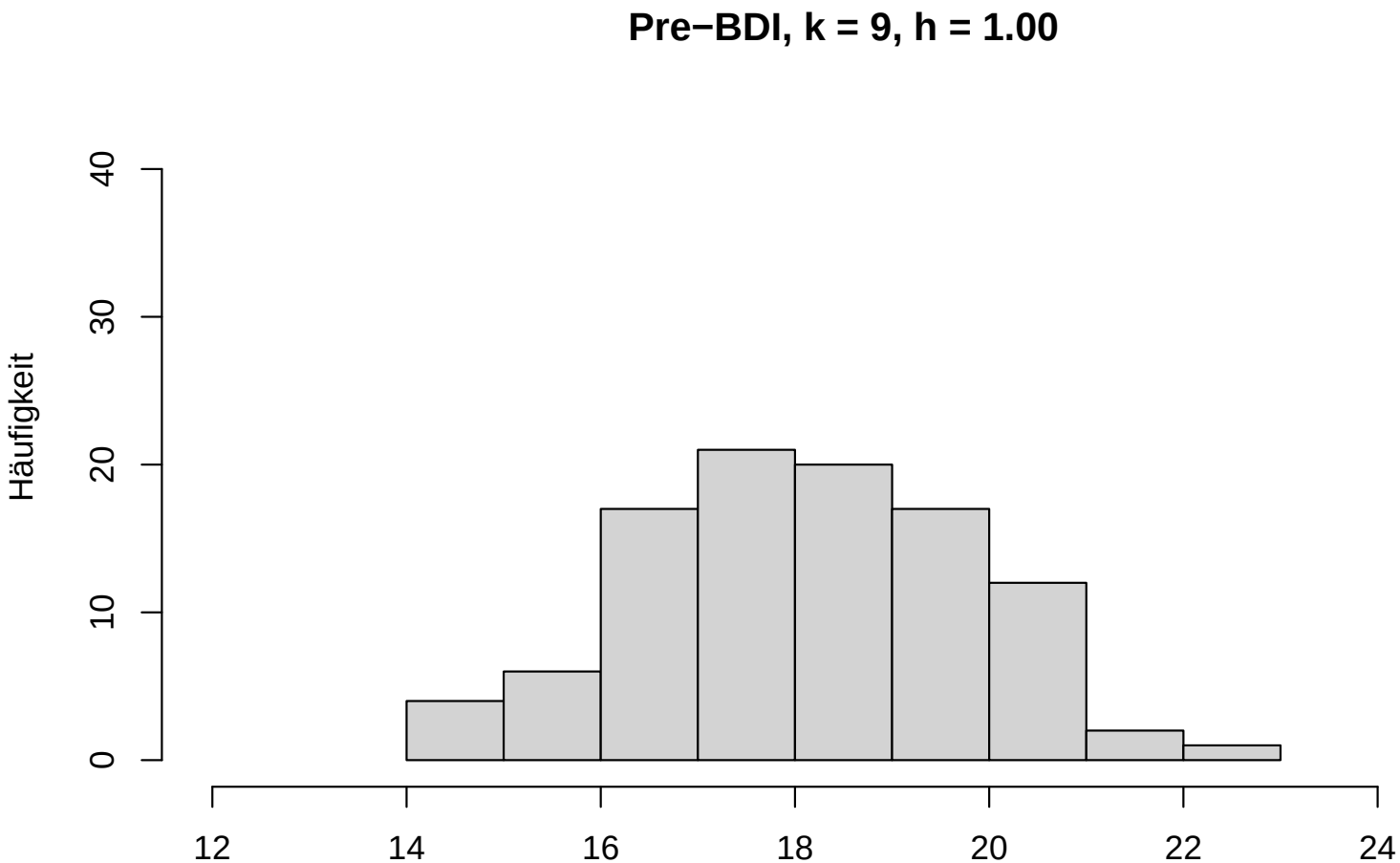


Berechnung von Klassenanzahlen und `breaks` Argument

```
# Histogramm mit gewuenschter Klassenbreite
h   = 1                                  # gewuenschte Klassenbreite
b_0 = min(x)                             # b_0
b_k = max(x)                             # b_k
k   = ceiling((b_k - b_0)/h)             # Anzahl der Klassen
b   = seq(b_0, b_k, by = h)              # Klassen [b_{j-1}, b_j[

# Excelstandard
n   = length(x)                          # Anzahl Datenwerte
k   = ceiling(sqrt(n))                   # Anzahl der Klassen
b   = seq(b_0, b_k, len = k)             # Klassen [b_{j-1}, b_j[
h   = b[2] - b[1]                        # Klassenbreite

# Sturges
n   = length(x)                          # Anzahl Datenwerte
k   = ceiling(log2(n)+1)                 # Anzahl der Klassen
b   = seq(b_0, b_k, len = k)             # Klassen [b_{j-1}, b_j[
h   = b[2] - b[1]                        # Klassenbreite

# Scott
n   = length(x)                          # Anzahl Datenwerte
S   = sd(x)                              # Stichprobenstandardabweichung
h   = ceiling(3.49*S/(n^(1/3)))          # Klassenbreite
k   = ceiling((b_k - b_0)/h)             # Anzahl der Klassen
b   = seq(b_0, b_k, len = k)             # Klassen [b_{j-1}, b_j[
```

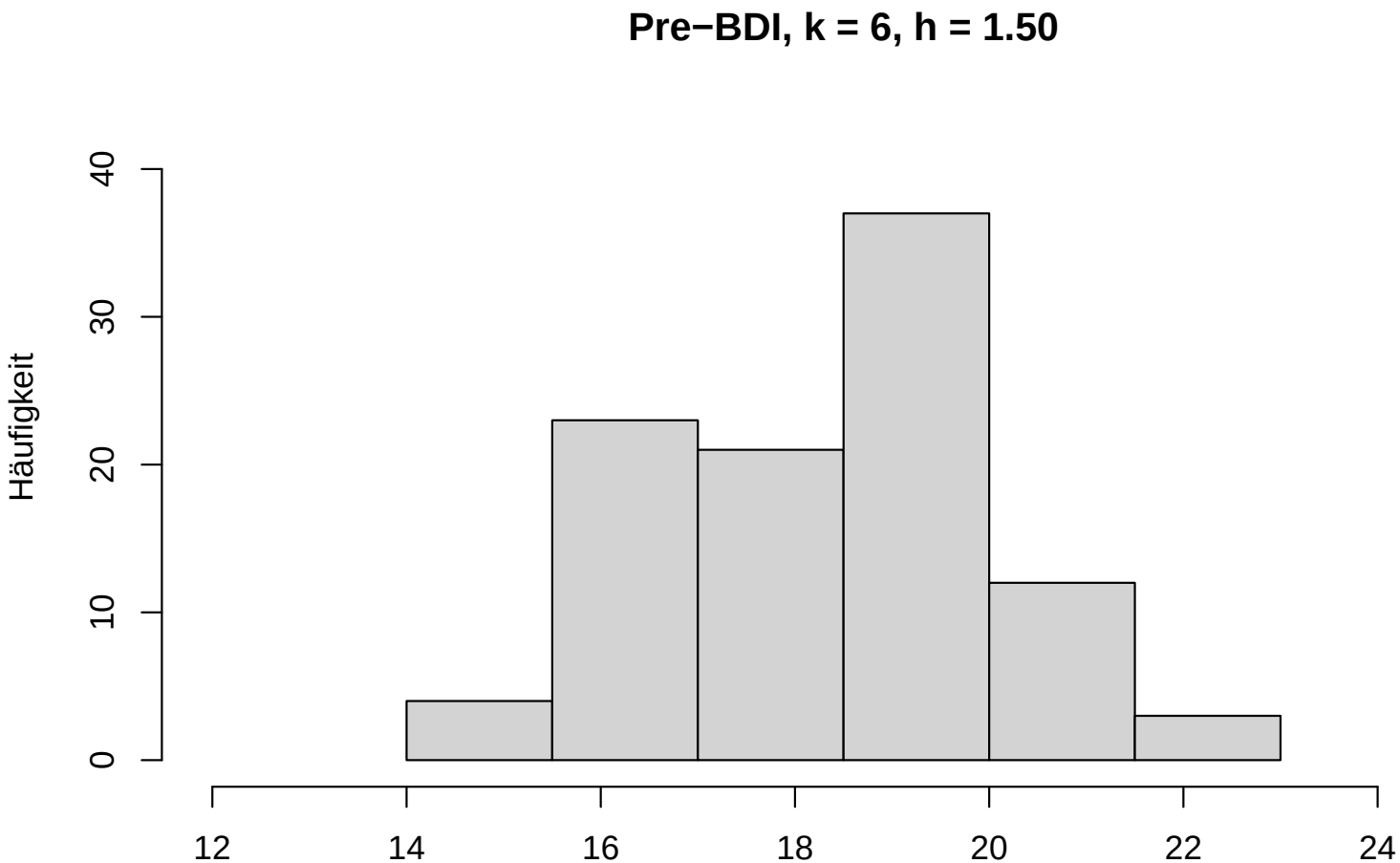
Histogramme

Gewünschte Klassenbreite $h := 1$



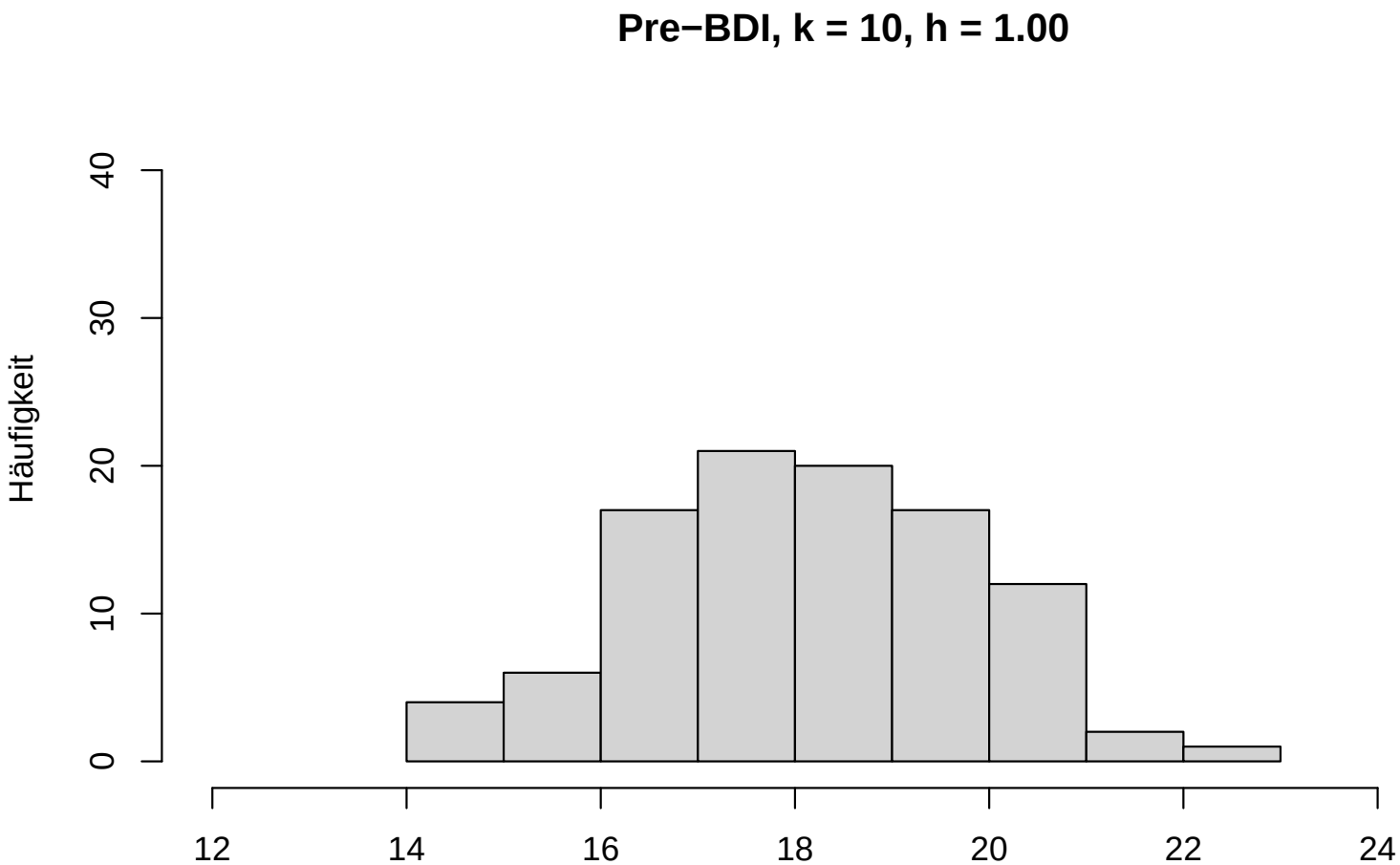
Histogramme

Gewünschte Klassenbreite $h := 1.5$



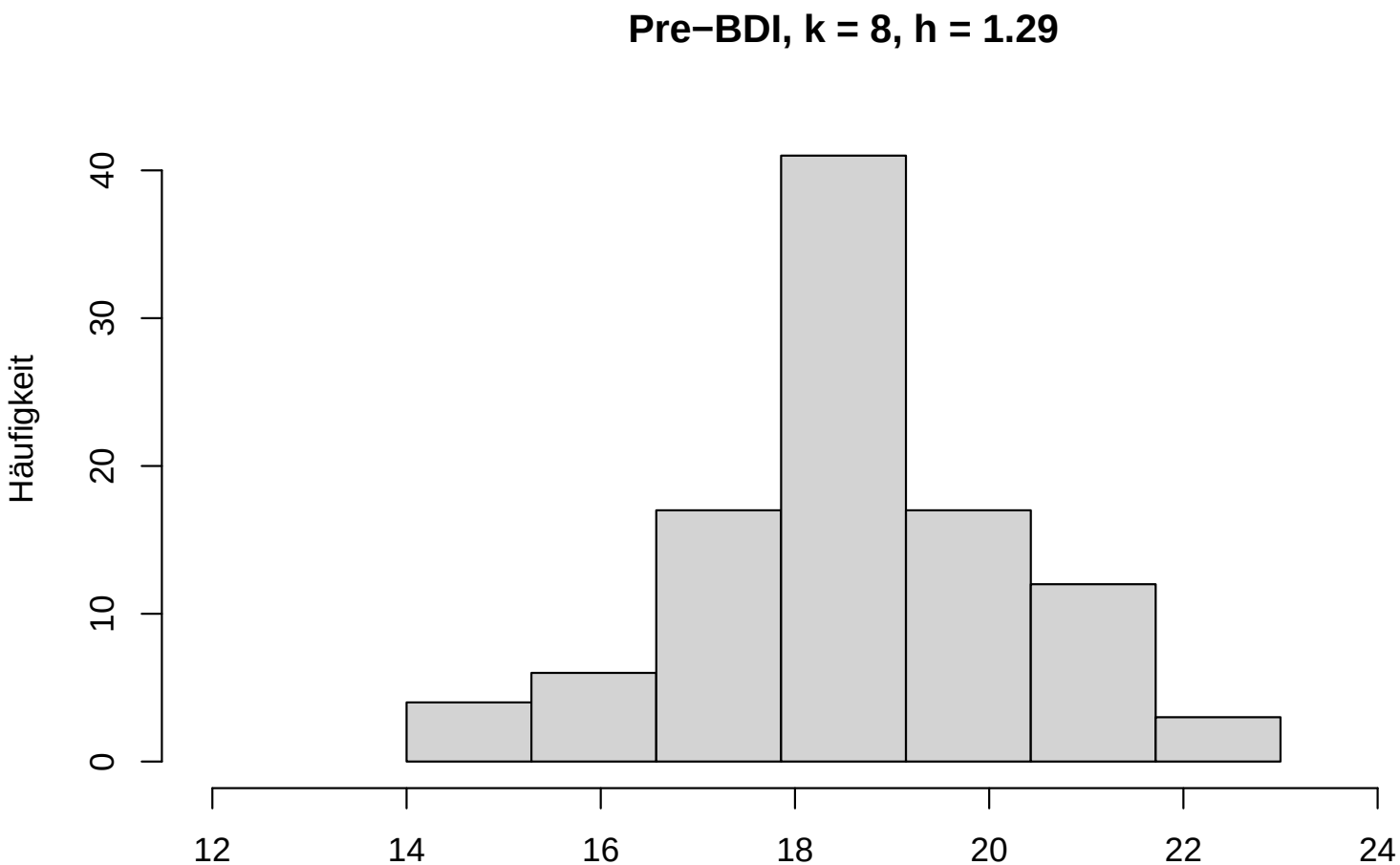
Histogramme

Excelstandard $k := \lceil \sqrt{n} \rceil$



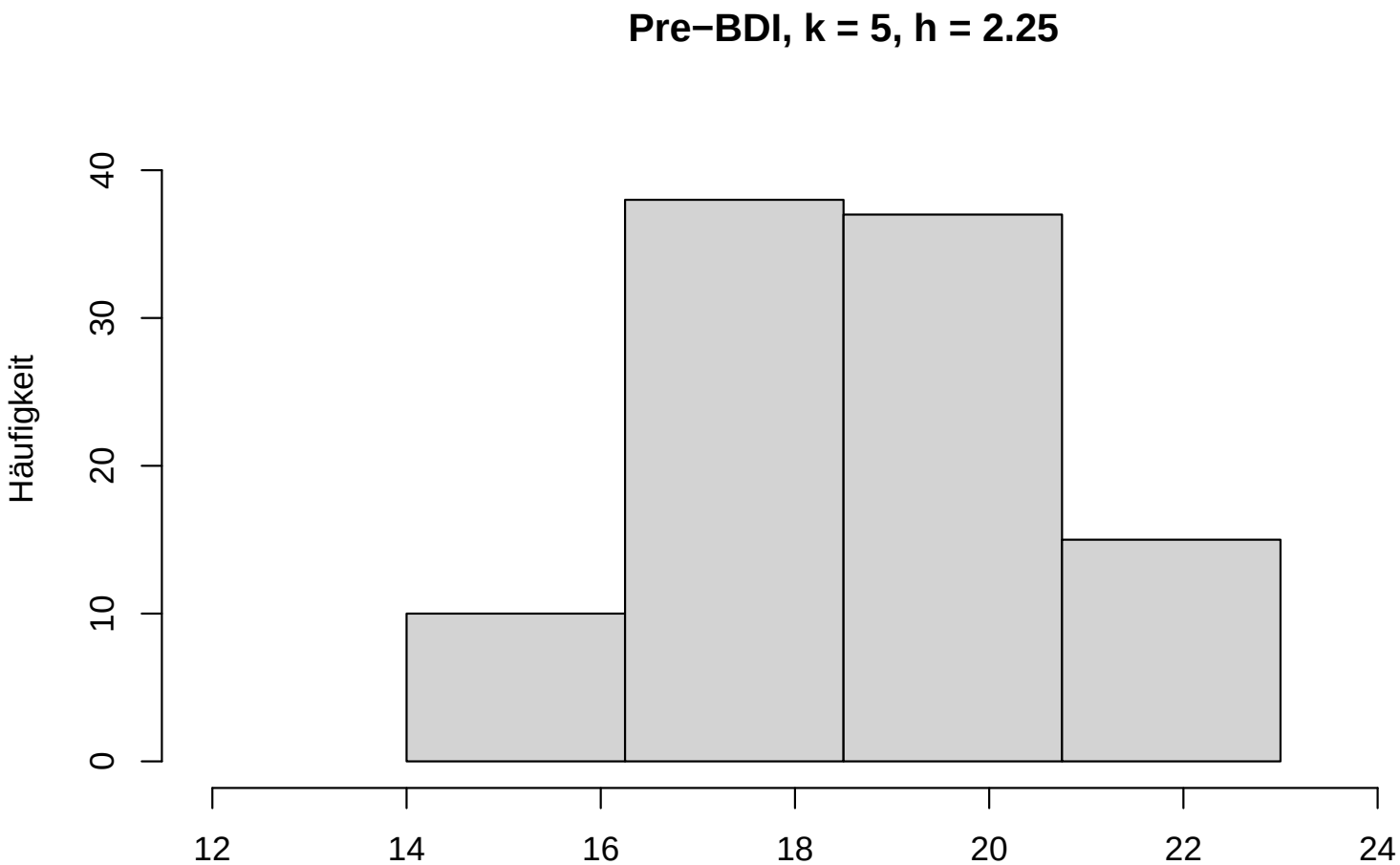
Histogramme

Sturges (1926) $k := \lceil \log_2 n + 1 \rceil$



Histogramme

Scott (1979) $h := 3.49 S_n / \sqrt[3]{n}$



Beispieldatensatz

Häufigkeitsverteilungen

Histogramme

Übungen und Selbstkontrollfragen

Übungen und Selbskontrollfragen

1. Definieren Sie die Begriffe der absoluten und relativen Häufigkeitsverteilungen.
2. Visualisieren Sie die Häufigkeitsverteilungen der Post-BDI Daten.
3. Visualisieren Sie die Häufigkeitsverteilungen der Differenzen von Post- und Pre-BDI Daten.
4. Visualisieren Sie die Häufigkeitsverteilungen der Differenzen von Post- und Pre-BDI Daten getrennt nach den experimentellen Bedingungen “Klassisch” und “Online”. Nutzen Sie dazu Ihr Wissen zu den Prinzipien der Indizierung in R.
5. Beschreiben Sie die in der vorherigen Aufgabe erstellten Häufigkeitsverteilungen.
6. Definieren Sie den Begriff des Histogramms.
7. Erläutern Sie die Bedeutung der Klassenanzahl für das Erscheinungsbild eines Histogramms.
8. Visualisieren Sie Histogramme der Daten wie in Aufgabe 4. mit einer Klassenbreite von 3, dem Excelstandard, der Sturges Klassenanzahl und der Scott Klassenanzahl.
9. Beschreiben Sie die in der vorherigen Aufgabe erstellten Histogramme.

References

Scott, David W. 1979. “On Optimal and Data-Based Histograms,” 6.

Sturges, Herbert A. 1926. “The Choice of a Class Interval.” *Journal of the American Statistical Association* 21 (153): 65–66. https://doi.org/10.1080/01621459.1926.10502161.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(8) Verteilungsfunktionen und Quantile

Empirische Verteilungsfunktionen

Quantile und Boxplots

Übungen und Selbstkontrollfragen

Empirische Verteilungsfunktionen

Quantile und Boxplots

Übungen und Selbstkontrollfragen

Definition (Kumulative absolute und relative Häufigkeitsverteilungen)

$x = (x_1, \ldots, x_n)$ sei ein Datensatz, $A := \{a_1, \ldots, a_k\}$ mit $k \leq n$ die im Datensatz vorkommenden verschiedenen Zahlenwerte und h und r die absoluten und relativen Häufigkeitsverteilungen von x, respektive. Dann heißt die Funktion

$$H : A \to \mathbb{N}, a \mapsto H(a) := \sum_{a' \leq a} h(a') \tag{1}$$

die *kumulative absolute Häufigkeitsverteilung* von x und die Funktion

$$R : A \to [0, 1], a \mapsto R(a) := \sum_{a' \leq a} r(a') \tag{2}$$

die *kumulative relative Häufigkeitsverteilung* der Zahlwerte von x.

Bemerkung

- Mit den Definitionen der absoluten und relativen Häufigkeitsverteilungen gilt also

$$H(a) = \text{Anzahl der } x_i \text{ aus } x \text{ mit } x_i \leq a \tag{3}$$

und

$$R(a) = \text{Anzahl der } x_i \text{ aus } x \text{ mit } x_i \leq a \text{ geteilt durch } n. \tag{4}$$

Empirische Verteilungsfunktionen

cumsum() erlaubt die Berechnung kumulativer Summen

```
# Einlesen des Beispieldatensatzes und Abbildungsverzeichnisdefinition
fname    = file.path(getwd(), "8_Daten", "psychotherapie_datensatz.csv")
D        = read.table(fname, sep = ",")
fdir     = file.path(getwd(), "8_Abbildungen")

# Evaluation der absoluten und relativen Häufigkeitsverteilugen von Pre.BDI
x        = D$Pre.BDI                  # Double vector der Pre.BDI Werte
n        = length(x)                  # Anzahl der Datenwerte
H        = as.data.frame(table(x))    # absolute Häufigkeitsverteilung als Dataframe
names(H) = c("a", "h")                # Spaltenbenennung
H$H      = cumsum(H$h)                # kumulative absolute Häufigkeitsverteilung
H$r      = H$h/n                      # relative Häufigkeitsverteilung
H$R      = cumsum(H$r)                # kumulative relative Häufigkeitsverteilung
print(H)
```

```
>     a  h   H    r    R
> 1   14  1   1 0.01 0.01
> 2   15  3   4 0.03 0.04
> 3   16  6  10 0.06 0.10
> 4   17 17  27 0.17 0.27
> 5   18 21  48 0.21 0.48
> 6   19 20  68 0.20 0.68
> 7   20 17  85 0.17 0.85
> 8   21 12  97 0.12 0.97
> 9   22  2  99 0.02 0.99
> 10 23  1 100 0.01 1.00
```

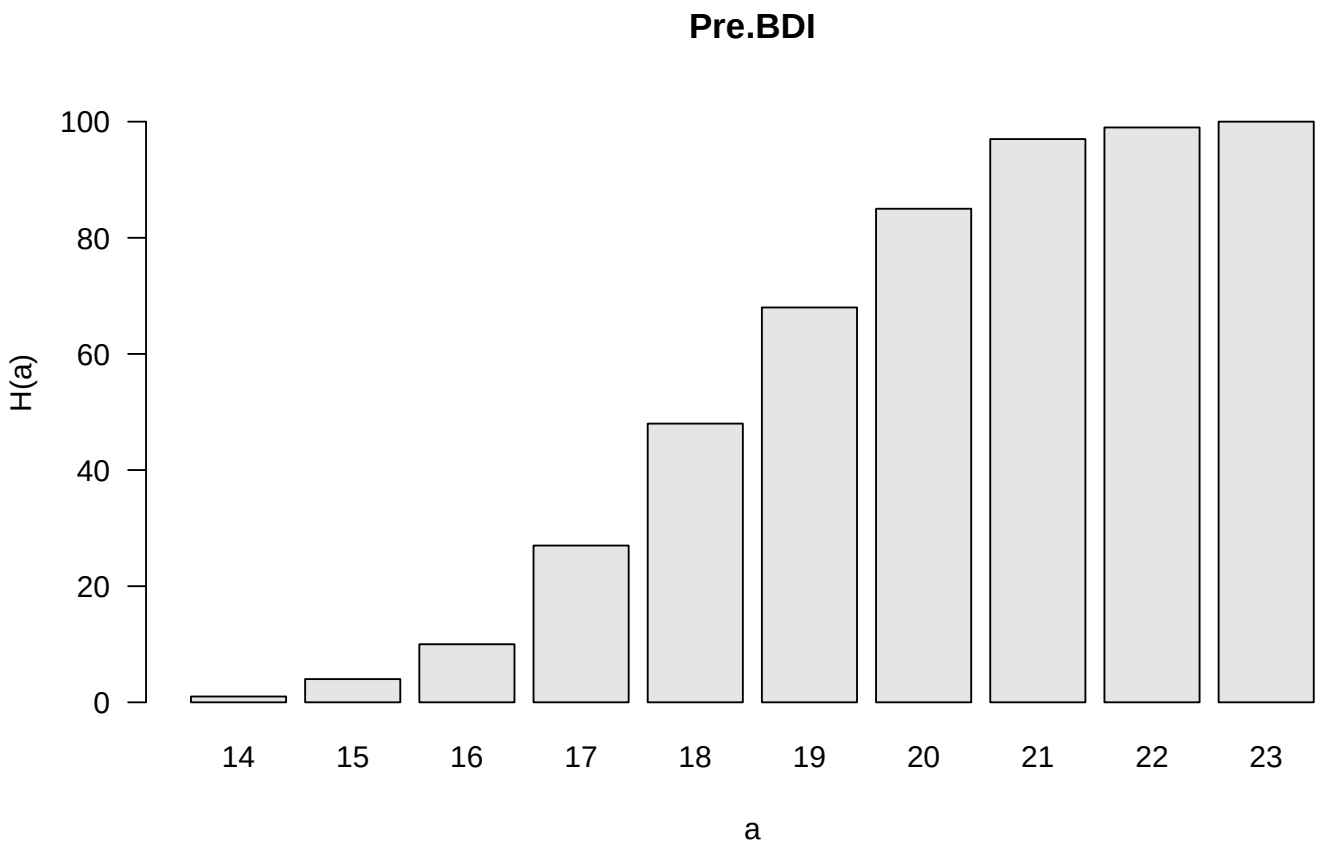
Empirische Verteilungsfunktionen

Kumulative absolute Häufigkeitsverteilung der Pre.BDI Werte

```
# Visualisierung der kumulativen absoluten Häufigkeitsverteilung
graphics.off()                      # Abbildungsinitialisierung
dev.new()                           # Abbildungsinitialisierung
Ha          = H$H                   # H(a) Werte
names(Ha)   = H$a                   # barplot braucht a Werte als names
barplot(                            # Balkendiagramm
Ha,                                 # H(a) Werte
col         = "gray90",             # Balkenfarbe
xlab        = "a",                  # x Achsenbeschriftung
ylab        = "H(a)",               # y Achsenbeschriftung
ylim        = c(0,110),             # y Achsenlimits
las         = 1,                    # Achsenticklabelorientierung
main        = "Pre.BDI")            # Titel

# PDF Speicherung
dev.copy2pdf(
file        = file.path(fdir, "pds_8_kh.pdf"),
width       = 8,
height      = 5)
```

Empirische Verteilungsfunktionen

Kumulative absolute Häufigkeitsverteilung der Pre.BDI Werte



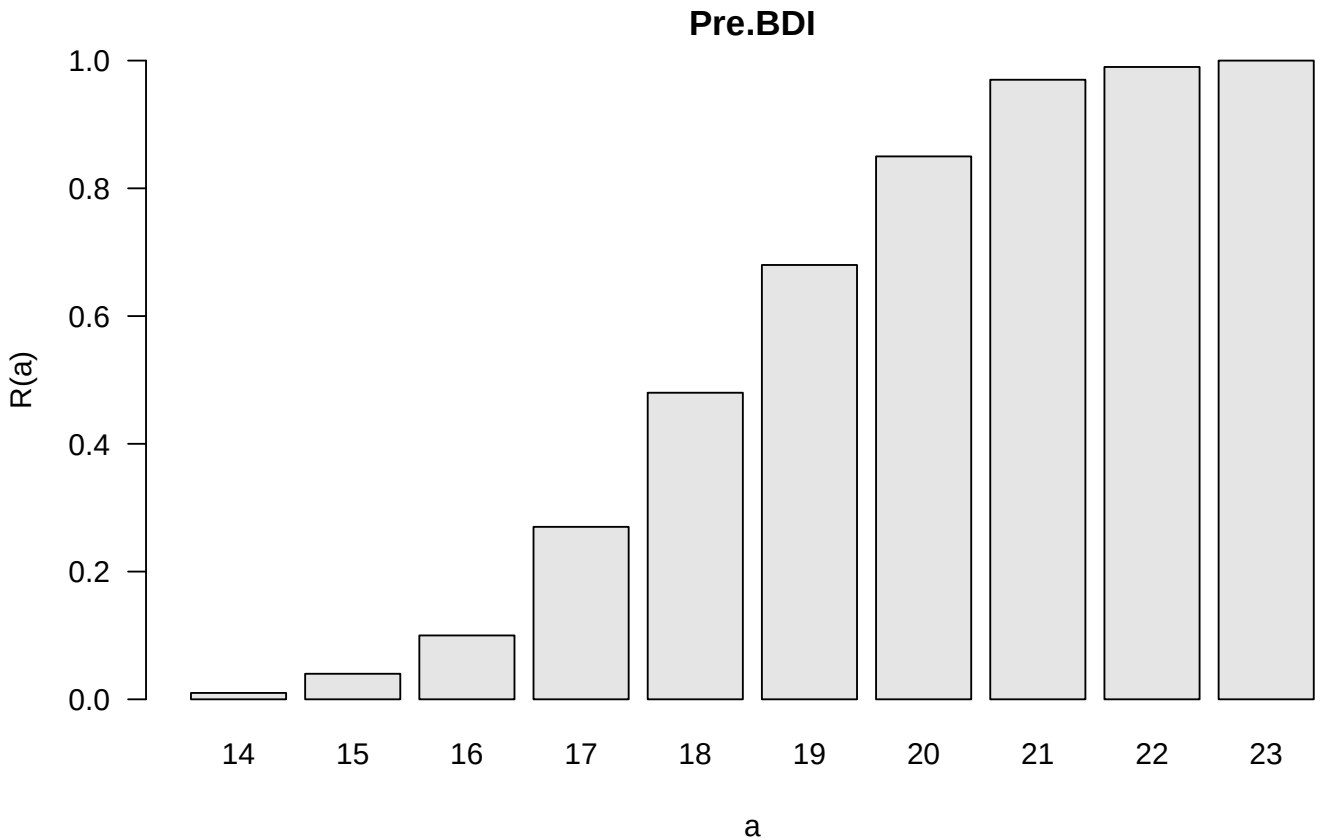
Empirische Verteilungsfunktionen

Kumulative relative Häufigkeitsverteilung der Pre.BDI Werte

```
# Visualisierung der kumulativen relativen Häufigkeitsverteilung
graphics.off()                  # Abbildungsinitialisierung
dev.new()                       # Abbildungsinitialisierung
R           = H$R               # R(a) Werte
names(R)    = H$a               # barplot braucht a Werte als names
dev.new()                       # Abbildungsinitialisierung
barplot(                        # Balkendiagramm
R,                              # R(a) Werte
col         = "gray90",         # Balkenfarbe
xlab        = "a",              # x Achsenbeschriftung
ylab        = "R(a)",           # y Achsenbeschriftung
ylim        = c(0,1),           # y Achsenlimits
las         = 1,                # Achsenticklabelorientierung
main        = "Pre.BDI")        # Titel

# PDF Speicherung
dev.copy2pdf(
file        = file.path(fdir, "pds_8_kr.pdf"),
width       = 8,
height      = 5)
```

Empirische Verteilungsfunktionen

Kumulative relative Häufigkeitsverteilung der Pre.BDI Werte



Definition (Empirische Verteilungsfunktion)

$x = (x_1, \ldots, x_n)$ sei ein Datensatz. Dann heißt die Funktion

$$F : \mathbb{R} \to [0, 1], \xi \mapsto F(\xi) := \frac{\text{Anzahl der } x_i \text{ aus } x \text{ mit } x_i \leq \xi}{n} \tag{5}$$

die empirische Verteilungsfunktion (EVF) von x.

Bemerkungen

- Die empirische Verteilungsfunktion wird auch *empirische kumulative Verteilungsfunktion* genannt.
- Die Definitionsmenge der EVF ist im Gegensatz zu Häufigkeitsverteilungen $\mathbb{R}$ und nicht A
- Die EVF verhält sich zu kumulativen Häufigkeitsverteilungen wie Histogramme zu Häufigkeitsverteilungen.
- Typischerweise sind empirische Verteilungsfunktionen Treppenfunktionen.
- Die (visuelle) Umkehrfunktion der EVF kann zur Bestimmung von Quantilen genutzt werden.

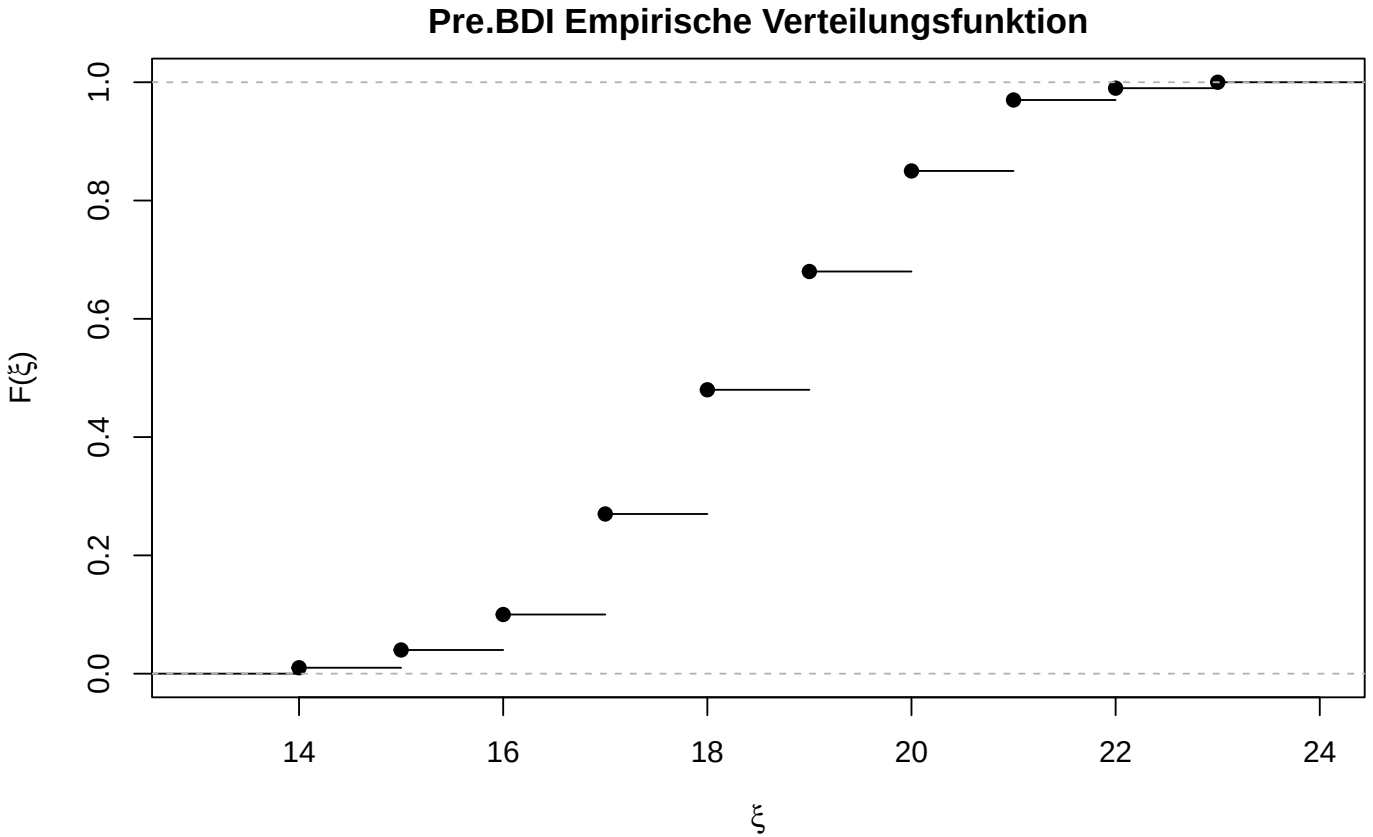
Empirische Verteilungsfunktionen

Empirische Verteilungsfunktion der Pre.BDI Werte

`ecdf()` evaluiert die empirischen Verteilungsfunktion eines Datensatzes

```
graphics.off()                                          # Abbildungsinitialisierung
dev.new()                                               # Abbildungsinitialisierung
x    = D$Pre.BDI                                        # double vector der Pre.BDI Werte
evf  = ecdf(x)                                          # Evaluation der EVF
plot(                                                   # plot() weiß mit ecdf object umzugehen
evf,                                                    # ecdf Objekt
xlab = TeX("$\\xi$"),                                   # x Achsenbeschriftung
ylab = TeX("$F(\\xi)$"),                                # y Achsenbeschriftung
main = "Pre.BDI Empirische Verteilungsfunktion")        # Titel

# PDF Speicherung
dev.copy2pdf(
file        = file.path(fdir, "pds_8_ecdf.pdf"),
width       = 8,
height      = 5)
```

Empirische Verteilungsfunktion der Pre.BDI Werte



Empirische Verteilungsfunktionen

Quantile und Boxplots

Übungen und Selbstkontrollfragen

Definition (p-Quantil)

$x = (x_1, ..., x_n)$ sei ein Datensatz und

$$x_s = \left(x_{(1)}, x_{(2)}, ..., x_{(n)}\right) \text{ mit } \min_{1 \leq i \leq n} x_i = x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)} = \max_{1 \leq i \leq n} x_i \tag{6}$$

der zugehörige aufsteigend sortierte Datensatz. Weiterhin bezeichne $\lfloor \cdot \rfloor$ die Abrundungsfunktion.Dann heißt für ein $p \in [0, 1]$ die Zahl

$$x_p := \begin{cases} x_{(\lfloor np+1 \rfloor)} & \text{falls } np \neq \mathbb{N} \\ \frac{1}{2}\left(x_{(np)} + x_{(np+1)}\right) & \text{falls } np \in \mathbb{N} \end{cases} \tag{7}$$

das p-*Quantil* von x.

Bemerkungen

- Mindestens $p \cdot 100\%$ aller Werte in x sind kleiner oder gleich x_p.
- Mindestens $(1 - p) \cdot 100\%$ aller Werte in x sind größer als x_p .
- Das p-Quantil teilt den geordneten Datensatz im Verhältnis p zu $(1 - p)$ auf.
- $x_{0.25}, x_{0.50}, x_{0.75}$ heißen *unteres Quartil*, *Median*, und *oberes Quartil*, respektive.
- $x_{j \cdot 0.10}$ für $j = 1, ..., 9$ heißen *Dezile*,
- $x_{j \cdot 0.01}$ für $j = 1, ..., 99$ heißen *Percentile*.

Beispiel p-Quantil (Henze (2018), Kapitel 5)

Datensatz und sortierter Datensatz

i	1	2	3	4	5	6	7	8	9	10
x_i	8.5	1.5	75	4.5	6.0	3.0	3.0	2.5	6.0	9.0
$x_{(i)}$	1.5	2.5	3.0	3.0	4.5	6.0	6.0	8.5	9.0	75

0.25-Quantil

Es ist $n = 10$ und es sei $p := 0.25$. Dann gilt $np = 10 \cdot 0.25 = 2.5 \notin \mathbb{N}$. Also folgt

$$x_{0.25} = x_{(\lfloor 2.5+1 \rfloor)} = x_{(3)} = 3.0 \tag{8}$$

0.80-Quantil

Es ist $n = 10$ und es sei $p := 0.80$. Dann gilt $np = 10 \cdot 0.80 = 8 \in \mathbb{N}$. Also folgt

$$x_{0.80} = \frac{1}{2}\left(x_{(8)} + x_{(8+1)}\right) = \frac{1}{2}\left(x_{(8)} + x_{(9)}\right) = \frac{8.5 + 9.0}{2} = 8.75. \tag{9}$$

Beispiel p-Quantil (Henze (2018), Kapitel 5) (fortgeführt)

"Manuelle" Quantilbestimmung anhand obiger Definition

```
x   = c(8.5,1.5,12,4.5,6.0,3.0,3.0,2.5,6.0,9.0)   # Beispieldaten
n   = length(x)                                    # Anzahl Datenwerte
x_s = sort(x)                                      # sortierter Datensatz
p   = 0.25                                         # np \notin \mathbb{N}
x_p = x_s[floor(n*p + 1)]                          # 0.25 Quantil
print(x_p)                                         # Ausgabe
```

```
> [1] 3
```

```
p   = 0.80                                         # np \in \mathbb{N}
x_p = (1/2)*(x_s[n*p] + x_s[n*p + 1])              # 0.80 Quantil
print(x_p)                                         # Ausgabe
```

```
> [1] 8.75
```

`quantile()` wertet Quantile anhand der Quantildefinition `type` aus.

Es gibt mindestens neun verschiedene Quantildefinitionen (cf. Hyndman and Fan (1996))

```
x_p = quantile(x, 0.80, type = 1)                  # 0.80 Quantil, Definition 1
print(x_p)
```

```
> 80%
> 8.5
```

```
x_p = quantile(x, 0.80, type = 2)                  # 0.80 Quantil, Definition 2
print(x_p)
```

```
>  80%
> 8.75
```

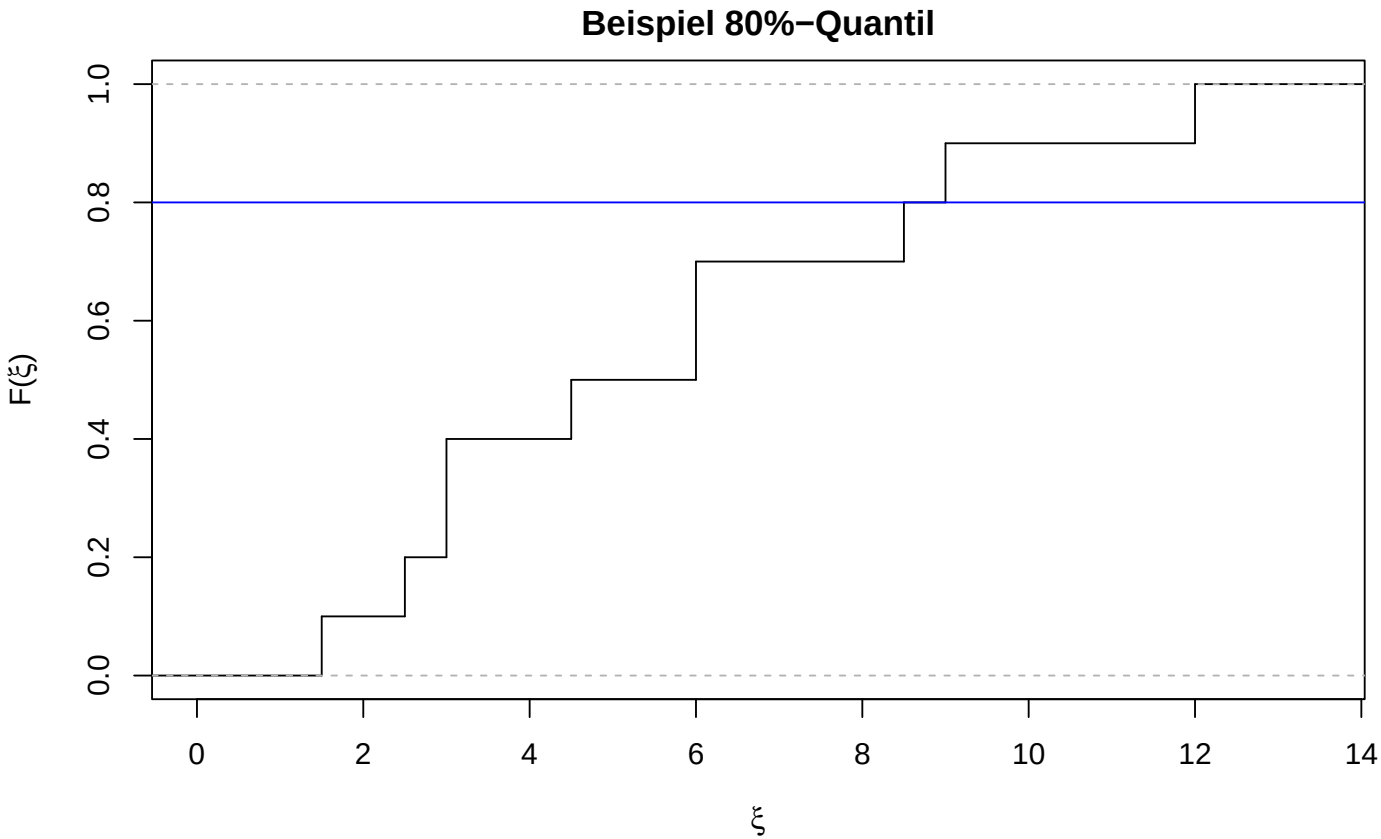
Beispiel p-Quantil (Henze (2018), Kapitel 5) (fortgeführt)

Kombination von `ecdf()`und `abline()` erlaubt prinzipiell die visuelle Bestimmung von Quantilen

```
graphics.off()                                  # Abbildungsinitialisierung
dev.new()                                       # Abbildungsinitialisierung
evf          = ecdf(x)                          # Evaluation der EVF
plot(                                           # plot() weiß mit ecdf object umzugehen
evf,                                            # ecdf Objekt
xlab         = TeX("$\\xi$"),                   # x Achsenbeschriftung
ylab         = TeX("$F(\\xi)$"),                # y Achsenbeschriftung
verticals    = TRUE,                            # vertikale Linien
do.points    = FALSE,                           # keine Punkte
main         = "Beispiel 80%-Quantil")          # Titel
abline(                                         # horizontale Linie
h            = 0.80,                            # y Ordinate der Linie
col          = "blue")                          # blau

# PDF Speicherung
dev.copy2pdf(
file         = file.path(fdir, "pds_8_ecdf_abline_x.pdf"),
width        = 8,
height       = 5)
```

Beispiel p-Quantil (Henze (2018), Kapitel 5) (fortgeführt)

Kombination von `ecdf()`und `abline()` erlaubt prinzipiell die visuelle Bestimmung von Quantilen

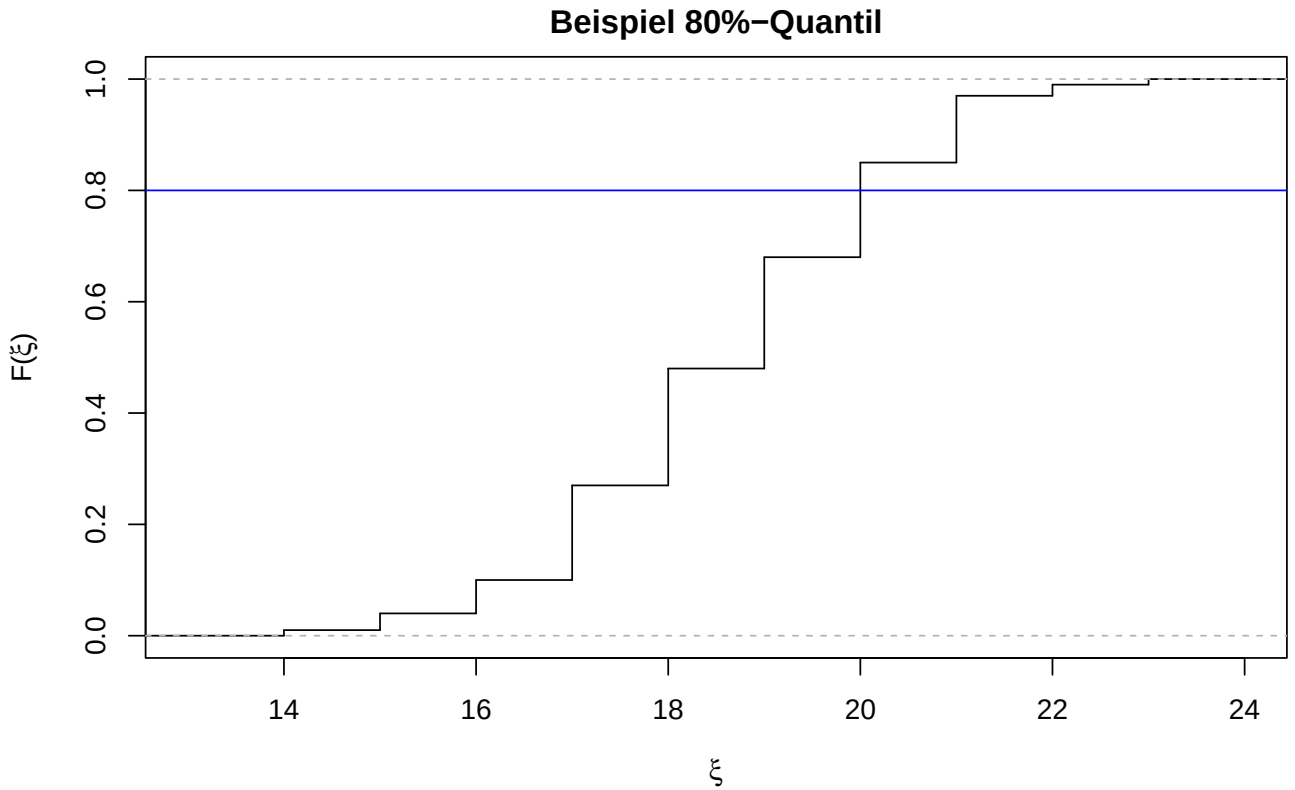


Empirische Verteilungsfunktionen

80% Quantil der Pre-BDI Daten

```
graphics.off()                                  # Abbildungsinitialisierung
dev.new()                                       # Abbildungsinitialisierung
x           = D$Pre.BDI                         # Double vector der Pre.BDI Werte
evf         = ecdf(x)                           # Evaluation der EVF
plot(                                           # plot() weiß mit ecdf object umzugehen
evf,                                            # ecdf Objekt
xlab        = TeX("$\\xi$"),                    # x Achsenbeschriftung
ylab        = TeX("$F(\\xi)$"),                 # y Achsenbeschriftung
verticals   = TRUE,                             # vertikale Linien
do.points   = FALSE,                            # keine Punkte
main        = "Beispiel 80%-Quantil")           # Titel
abline(                                         # horizontale Linie
h           = 0.80,                             # y Ordinate der Linie
col         = "blue")                           # blau

# PDF Speicherung
dev.copy2pdf(
file        = file.path(fdir, "pds_8_ecdf_abline_prebdi.pdf"),
width       = 8,
height      = 5)
```

80% Quantil der Pre-BDI Daten




```
x_p = quantile(D$Pre.BDI, 0.80, type = 2) # 0.80 Quantil, Definition 2
print(x_p)
```

```
> 80%
>  20
```

Boxplots

Ein Boxplot visualisiert eine Quantil-basierte Zusammenfassung eines Datensatzes.

Typischerweise werden $\min x, x_{0.25}, x_{0.50}, x_{0.75}, \max x$ visualisiert.

- $\min x$ und $\max x$ werden oft als "Whiskerendpunkte" dargestellt.
- $x_{0.25}$ und $x_{0.75}$ sind untere und obere Grenze der zentralen grauen Box.
- $x_{0.50}$ wird als Strich in der zentralen grauen Box abgebildet.

$d_Q := x_{0.75} - x_{0.25}$ heißt *Interquartilsabstand* und dient als Verteilungsbreitenmaß

`summary()` liefert wesentliche Kennzahlen

```
# Sechswertezusammenfassung
summary(D$Pre.BDI)
```

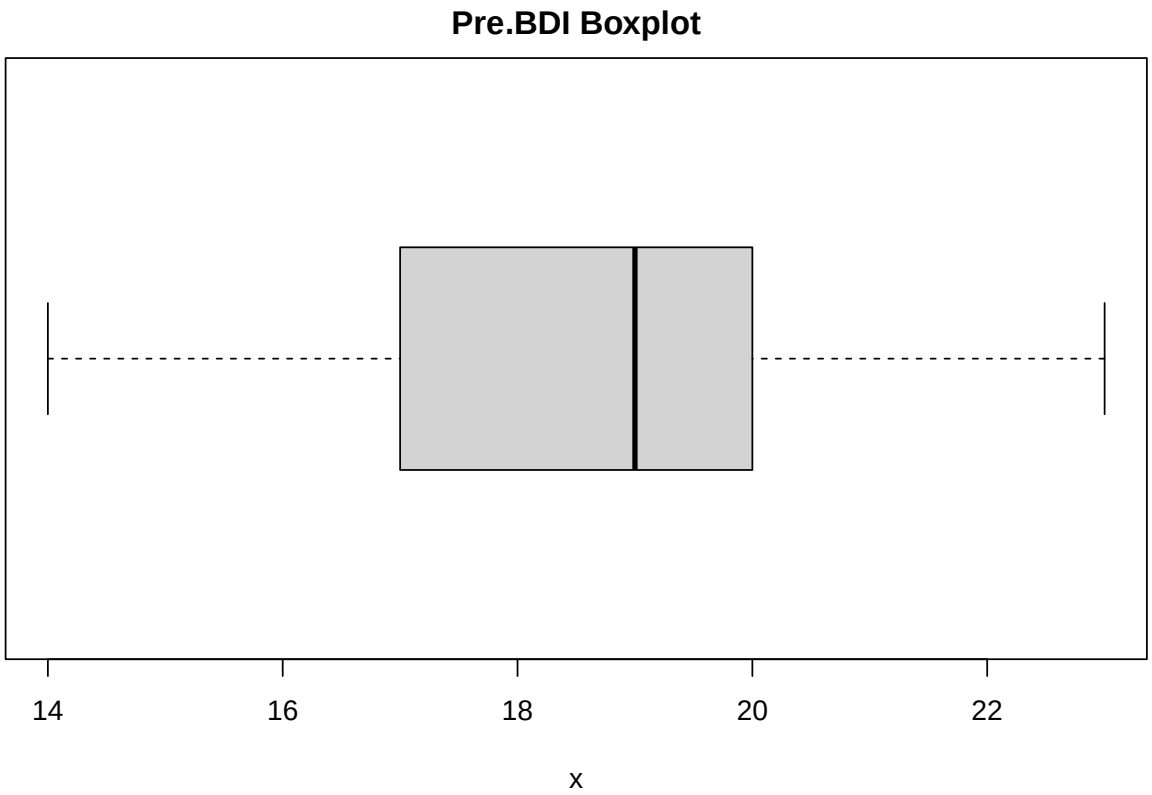
```
>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
>    14.0    17.0    19.0    18.6    20.0    23.0
```

Boxplots

`boxplot()` erstellt einen Boxplot,

```
# Boxplot
graphics.off()                          # Abbildungsinitialisierung
dev.new()                               # Abbildungsinitialisierung
boxplot(                                # Boxplot
D$Pre.BDI,                              # Datensatz
horizontal  = T,                        # horizontale Darstellung
range       = 0,                        # Whiskers bis zu min x und max x
xlab        = "x",                      # x Achsenbeschriftung
main        = "Pre.BDI Boxplot")        # Titel

# PDF Speicherung
dev.copy2pdf(
file        = file.path(fdir, "pds_8_boxplot_prebdi.pdf"),
width       = 8,
height      = 5)
```

Boxplots



Es gibt viele Boxplotvariationen (cf. McGill, Tukey, and Larsen (1978)), eine genaue Erläuterung ist immer nötig!

Empirische Verteilungsfunktionen

Quantile und Boxplots

Übungen und Selbstkontrollfragen

Übungen und Selbstkontrollfragen

1. Definieren Sie die Begriffe der kumulativen absoluten und relativen Häufigkeitsverteilungen.
2. Erzeugen und visualisieren Sie die kumulativen absoluten und relativen Häufigkeitsverteilungen der Post.BDI Daten des Beispieldatensatzes *psychotherapie_datensatz.csv*.
3. Definieren Sie den Begriff der empirischen Verteilungsfunktion.
4. Erzeugen und visualisieren Sie die empirische Verteilungsfunktion der Post.BDI Daten.
5. Erläutern Sie den Begriff des sortierten Datensatzes.
6. Definieren Sie den Begriff des p-Quantils.
7. Berechnen Sie das obere Quartil des Beispieldatensatzes auf Folie 16.
8. Definieren Sie die Begriffe unteres Quartil, Median und oberes Quartil mithilfe des p-Quantils.
9. Berechnen Sie das untere Quartil, den Median und das obere Quartil der Post.BDI Daten. Vergleiche Sie ihre Ergebnisse mit der Ausgabe der summary() Funktion.
10. Erstellen Sie einen Boxplot der Post.BDI Daten.

References

Henze, Norbert. 2018. *Stochastik für Einsteiger*. Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-22044-0.

Hyndman, Rob J., and Yanan Fan. 1996. "Sample Quantiles in Statistical Packages." *The American Statistician* 50 (4): 361. https://doi.org/10.2307/2684934.

McGill, Robert, John W. Tukey, and Wayne A. Larsen. 1978. "Variations of Box Plots." *The American Statistician* 32 (1): 12. https://doi.org/10.2307/2683468.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(9) Maße der Zentralen Tendenz

Mittelwert

Median

Modalwert

Visuelle Intuitionen

Übungen und Selbstkontrollfragen

Definition (Mittelwert)

$x = (x_1, ..., x_n)$ sei ein Datensatz. Dann heißt

$$\bar{x} := \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1}$$

der *Mittelwert* von x.

Bemerkung

- Im Kontext der Inferenzstatistik heißt der Mittelwert *Stichprobenmittel*.
- Die Inferenzstatistik gibt der Mittelwertsbildung ihren Sinn.

Berechnung des Mittelwerts

```
# Einlesen des Beispieldatensatzes und Abbildungsverzeichnisdefinition
fname    = file.path(getwd(), "9_Daten", "psychotherapie_datensatz.csv")
D        = read.table(fname, sep = ",")
fdir     = file.path(getwd(), "9_Abbildungen")

# Mittelwertberechnung
x        = D$Pre.BDI             # double Vektor der Pre-BDI Werte Werte
n        = length(x)             # Anzahl der Werte
x_bar    = (1/n)*sum(x)          # Mittelwertsberechnung
print(x_bar)                     # Ausgabe
```

```
> [1] 18.6
```

mean() zur Berechnung des Mittelwerts

```
x_bar    = mean(x)               # Mittelwertsberechnung
print(x_bar)                     # Ausgabe
```

```
> [1] 18.6
```

Theorem (Eigenschaften des Mittelwerts)

$x = (x_1, \ldots, x_n)$ und sei ein Datensatz und $\bar{x}$ sei der Mittelwert von x. Dann gelten

(1) Die Summe der Abweichungen vom Mittelwert ist Null,

$$\sum_{i=1}^{n} (x_i - \bar{x}) = 0. \tag{2}$$

(2) Die absoluten Summen negativer und positiver Abweichungen vom Mittelwert sind gleich, d.h. wenn $j = 1, \ldots, n_j$ die Datenpunktindizes mit $(x_j - \bar{x}) < 0$ und $k = 1, \ldots, n_k$ die Datenpunktindizes mit $(x_k - \bar{x}) \geq$ bezeichnen, dann gilt mit $n_j + n_k$

$$|\sum_{j=1}^{n_j} (x_j - \bar{x})| = |\sum_{k=1}^{n_k} (x_k - \bar{x})|. \tag{3}$$

(3) Der Mittelwert der Summe zweier gleich großer Datensätze entspricht der Summe ihrer Mittelwerte, d.h. für einen weiteren Datensatz $y = (y_1, \ldots, y_n)$ mit Mittelwert $\bar{y}$ gilt

$$\overline{x + y} = \bar{x} + \bar{y} \tag{4}$$

(4) Eine linear-affine Transformation eines Datensatz transformiert den Mittelwert des Datensatzes linear-affin, d.h für $a, b \in \mathbb{R}$ gilt

$$\overline{ax + b} = a\bar{x} + b$$

Beweis

(1) Es gilt

$$\sum_{i=1}^{n}(x_i - \bar{x}) = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \bar{x} = \sum_{i=1}^{n} x_i - n\bar{x} = \sum_{i=1}^{n} x_i - \frac{n}{n}\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} x_i = 0.$$

(2) Seien $j = 1, \ldots, n_j$ die Indizes mit $(x_j - \bar{x}) < 0$ und $k = 1, \ldots, n_k$ die Indizes mit $(x_k - \bar{x}) \geq 0$, so dass $n = n_j + n_k$. Dann gilt

$$\sum_{i=1}^{n}(x_i - \bar{x}) = 0 \Leftrightarrow \sum_{j=1}^{n_j}(x_j - \bar{x}) + \sum_{k=1}^{n_k}(x_k - \bar{x}) = 0 \Leftrightarrow \sum_{k=1}^{n_k}(x_k - \bar{x}) = -\sum_{j=1}^{n_j}(x_j - \bar{x})$$

$$\Leftrightarrow \left|\sum_{j=1}^{n_j}(x_j - \bar{x})\right| = \left|\sum_{k=1}^{n_k}(x_k - \bar{x})\right|.$$

Mittelwert

Beweis

(3) Es gilt

$$\overline{x+y} := \frac{1}{n}\sum_{i=1}^{n}(x_i + y_i) = \frac{1}{n}\sum_{i=1}^{n} x_i + \frac{1}{n}\sum_{i=1}^{n} y_i =: \bar{x} + \bar{y}$$

(4) Es gilt

$$\begin{aligned}
\overline{ax+b} &:= \frac{1}{n}\sum_{i=1}^{n}(ax_i + b) \\
&= \sum_{i=1}^{n}\left(\frac{1}{n}ax_i + \frac{1}{n}b\right) \\
&= \sum_{i=1}^{n}\left(\frac{1}{n}ax_i\right) + \sum_{i=1}^{n}\left(\frac{1}{n}b\right) \\
&= a\frac{1}{n}\sum_{i=1}^{n} x_i + \frac{1}{n}\sum_{i=1}^{n} b \\
&= a\bar{x} + b
\end{aligned}$$

Eigenschaften des Mittelwerts

Summe der Abweichungen

```
x       = D$Pre.BDI                     # double Vektor der  Werte
s       = sum(x - mean(x))              # Summe der Abweichungen vom Mittelwert
print(s)                                # Rundungsfehler
```

```
> [1] 5.68e-14
```

Beträge der positiven und negativen Abweichungen

```
x   = D$Pre.BDI                         # double Vektor der  Werte
s_1 = sum(x[x <= mean(x)] - mean(x))    # Summe aller negativer Abweichungen
s_2 = sum(x[x > mean(x)]  - mean(x))    # Summe aller positiver Abweichungen
print(s_1)                              # Ausgabe
```

```
> [1] -71.3
```

```
print(s_2)                              # Ausgabe
```

```
> [1] 71.3
```

Mittelwert

Summation von Datensätzen

```
x       = D$Pre.BDI                    # double Vektor der  Werte
x_bar   = mean(x)                      # Mittelwert der  Werte
y       = D$Post.BDI                   # double Vektor der Post.BDI Werte
y_bar   = mean(y)                      # Mittelwert der Post.BDI Werte
z       = x + y                        # double Vektor der  und Post.BDI Werte
z_bar   = mean(z)                      # Mittelwert der Summe der  und Post.BDI Werte
print(z_bar)                           # Ausgabe
```

```
> [1] 31.7
```

```
xy_bar  = x_bar + y_bar                # Summe der Mittelwerte der  und Post.BDI Werte
print(xy_bar)                          # Ausgabe
```

```
> [1] 31.7
```

Linear-affine Transformation

```
x        = D$Pre.BDI                   # double Vektor der Pre.BDI Werte
x_bar    = mean(x)                     # Mittelwert der Pre.BDI Werte
a        = 2                           # Multiplikationskonstante
b        = 5                           # Additionskonstante
y        = a*x + b                     # linear-affine Transformation der Pre.BDI Werte
y_bar    = mean(y)                     # Mittelwert der transfomierten Pre.BDI Werte
print(y_bar)                           # Ausgabe
```

```
> [1] 42.2
```

```
ax_bar_b = a*x_bar + b                 # Transformation des TA1 Mittelwerts
print(ax_bar_b)                        # Ausgabe
```

```
> [1] 42.2
```

Mittelwert

Median

Modalwert

Visuelle Intuitionen

Übungen und Selbstkontrollfragen

Definition (Median)

$x = (x_1, ..., x_n)$ sei ein Datensatz und $x_s = (x_{(1)}, ..., x_{(n)})$ der zugehörige aufsteigend sortierte Datensatz. Dann ist der Median von x definiert als

$$\tilde{x} := \begin{cases} x_{((n+1)/2)} & \text{falls } n \text{ ungerade} \\ \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right) & \text{falls } n \text{ gerade} \end{cases} \tag{5}$$

Bemerkungen

- Der Median ist identisch mit dem 0.5-Quantil.
- Mindestens 50% aller x_i sind kleiner oder gleich $\tilde{x}$
- Mindestens 50% aller x_i sind größer oder gleich $\tilde{x}$.
- Anstelle eines Beweises verweisen wir auf untenstehende Abbildungen.

Beispiele

Beispiel für n ungerade

$$n \coloneqq 5 \Rightarrow \left(\frac{5+1}{2}\right) = (3) \Rightarrow \tilde{x} \coloneqq x_{(3)}$$

$$x_{(1)}, x_{(2)}, x_{(3)} \leq \tilde{x} \leq x_{(3)}, x_{(4)}, x_{(5)}$$

Beispiel für n gerade

$$n \coloneqq 6 \Rightarrow \left(\frac{6}{2}\right) = (3), \left(\frac{6}{2}+1\right) = (4) \Rightarrow \tilde{x} \coloneqq \frac{1}{2}\left(x_{(3)} + x_{(4)}\right)$$

$$x_{(1)}, x_{(2)}, x_{(3)} < \tilde{x} < x_{(4)}, x_{(5)}, x_{(6)}$$

Median

Berechnung des Medians

```
x           = D$Pre.BDI                    # double Vektor der Pre.BDI Werte
n           = length(x)                    # Anzahl der Werte
x_s         = sort(x)                      # aufsteigend sortierter Vektor
if(n %% 2 == 1){                           # n ungerade, n mod 2 == 1
   x_tilde = x_s[(n+1)/2]
} else {                                   # n gerade, n mod 2 == 0
   x_tilde = (x_s[n/2] + x_s[n/2 + 1])/2
}
print(x_tilde)
```

```
> [1] 19
```

Berechnung des Medians mit `median()`

```
x_tilde     = median(x)                  # Berechnung des Medians
print(x_tilde)                           # Ausgabe
```

```
> [1] 19
```

Median

Der Median ist weniger anfällig für Ausreißer als der Mittelwert

```
x       = D$Pre.BDI                 # double Vektor der Pre.BDI Werte
x_bar   = mean(x)                   # Mittelwert der Pre.BDI Werte
x_tilde = median(x)                 # Median der Pre.BDI Werte
print(x_bar)                        # Ausgabe
```

```
> [1] 18.6
```

```
print(x_tilde)                      #  Ausgabe
```

```
> [1] 19
```

```
y       = x                         # neuer Datensatz mit
y[1]    = 10000                     # ... einem Extremwert
y_bar   = mean(y)                   # Mittelwert des neuen Datensatzes
print(y_bar)                        # Ausgabe
```

```
> [1] 118
```

```
y_tilde = median(y)                 # Mittelwert des neuen Datensatzes
print(y_tilde)                      # Ausgabe
```

```
> [1] 19
```

Mittelwert

Median

Modalwert

Visuelle Intuitionen

Übungen und Selbstkontrollfragen

Definition (Modalwert)

$x := (x_1, \ldots, x_n)$ mit $x_i \in \mathbb{R}$ sei ein Datensatz, $A := \{a_1, \ldots, a_k\}$ mit $k \leq n$ seien die im Datensatz vorkommenden verschiedenen Zahlenwerte und $h : A \to \mathbb{N}$ sei die absolute Häufigkeitsverteilung der Zahlwerte von x. Dann ist der *Modalwert (oder Modus)* von x definiert als

$$\text{argmax}_{a \in A} h(a), \tag{6}$$

also der am häufigsten im Datensatz vorkommende Wert.

Bemerkungen

- Modalwerte sind nur bei Datensätzen mit Datenpunktwiederholungen sinnvoll.

Bestimmung des Modalwertes

```
x        = D$Pre.BDI                # double Vektor der Pre.BDI Werte
n        = length(x)                # Anzahl der Datenwerte (100)
H        = as.data.frame(table(x))  # absolute Haeufigkeitsverteilung (dataframe)
names(H) = c("a", "h")              # Konsistente Benennung
mod      = H$a[which.max(H$h)]      # Modalwert
print(as.numeric(as.vector(mod)))   # Ausgabe als numeric vector, nicht factor
```

```
> [1] 18
```

Mittelwert

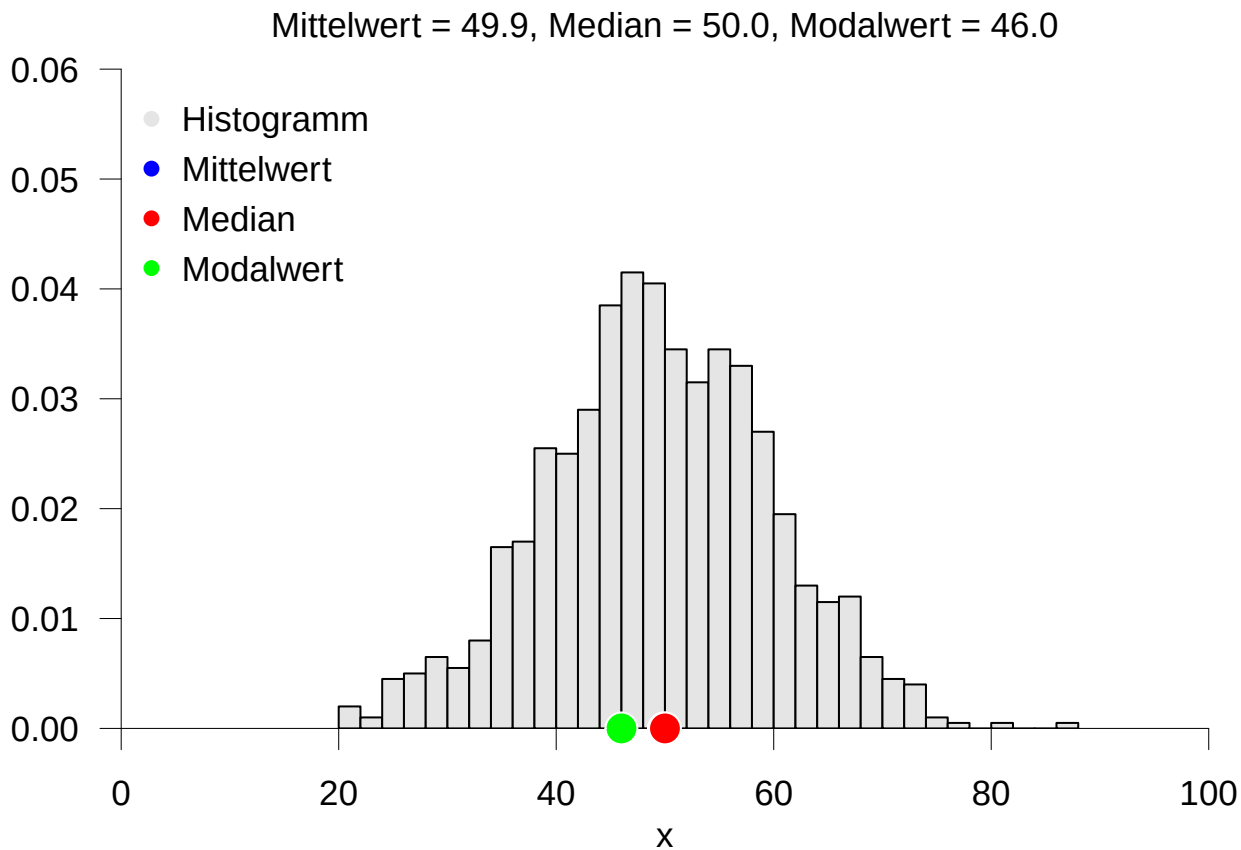
Median

Modalwert

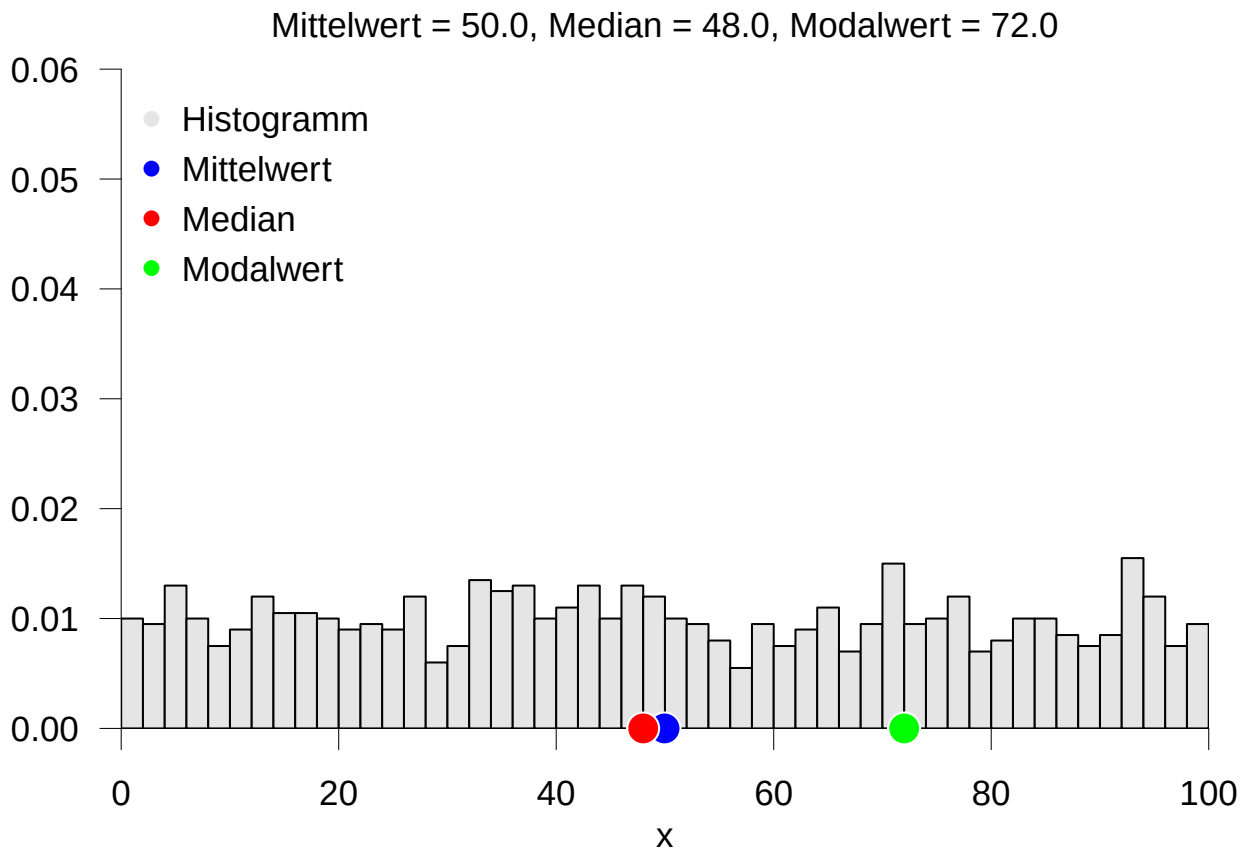
Visuelle Intuitionen

Übungen und Selbstkontrollfragen

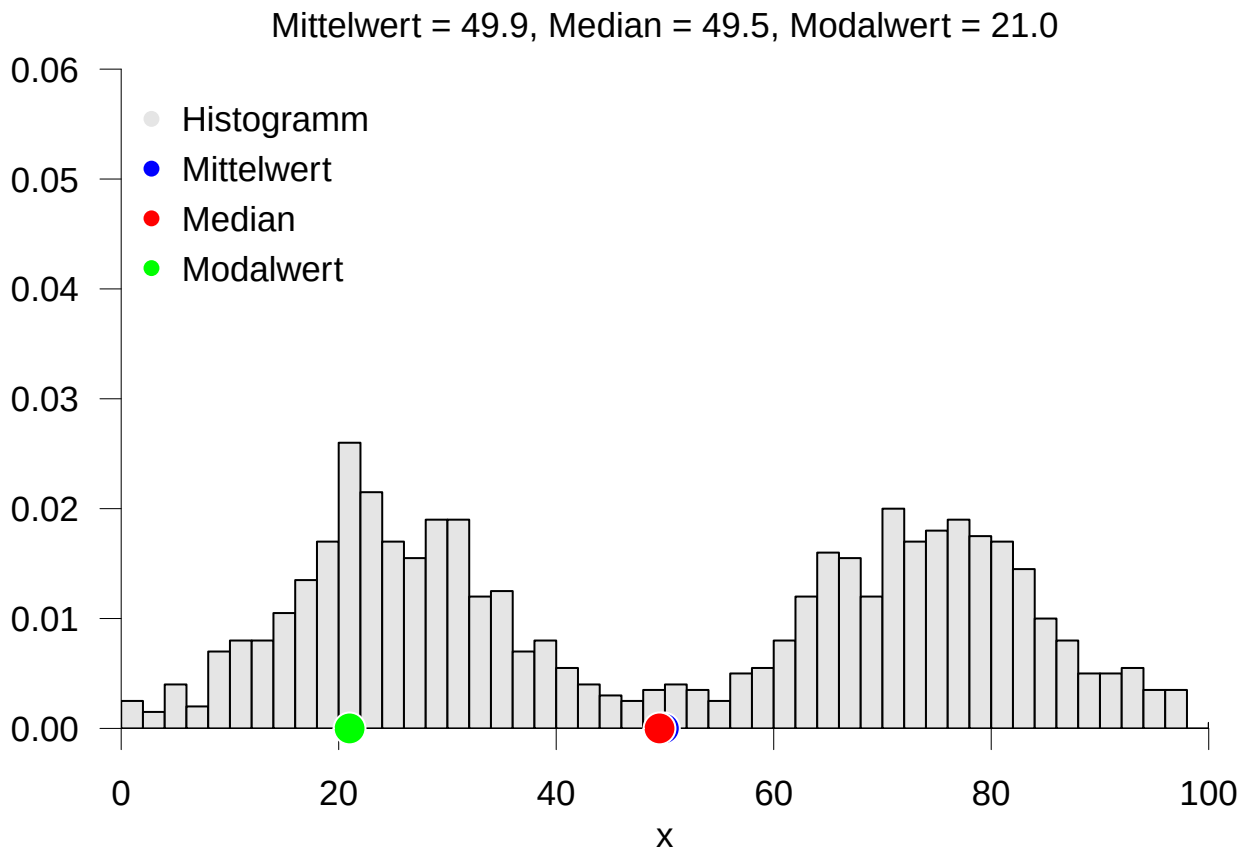
Visuelle Intuition zu Maßen zentraler Tendenz bei Normalverteilung



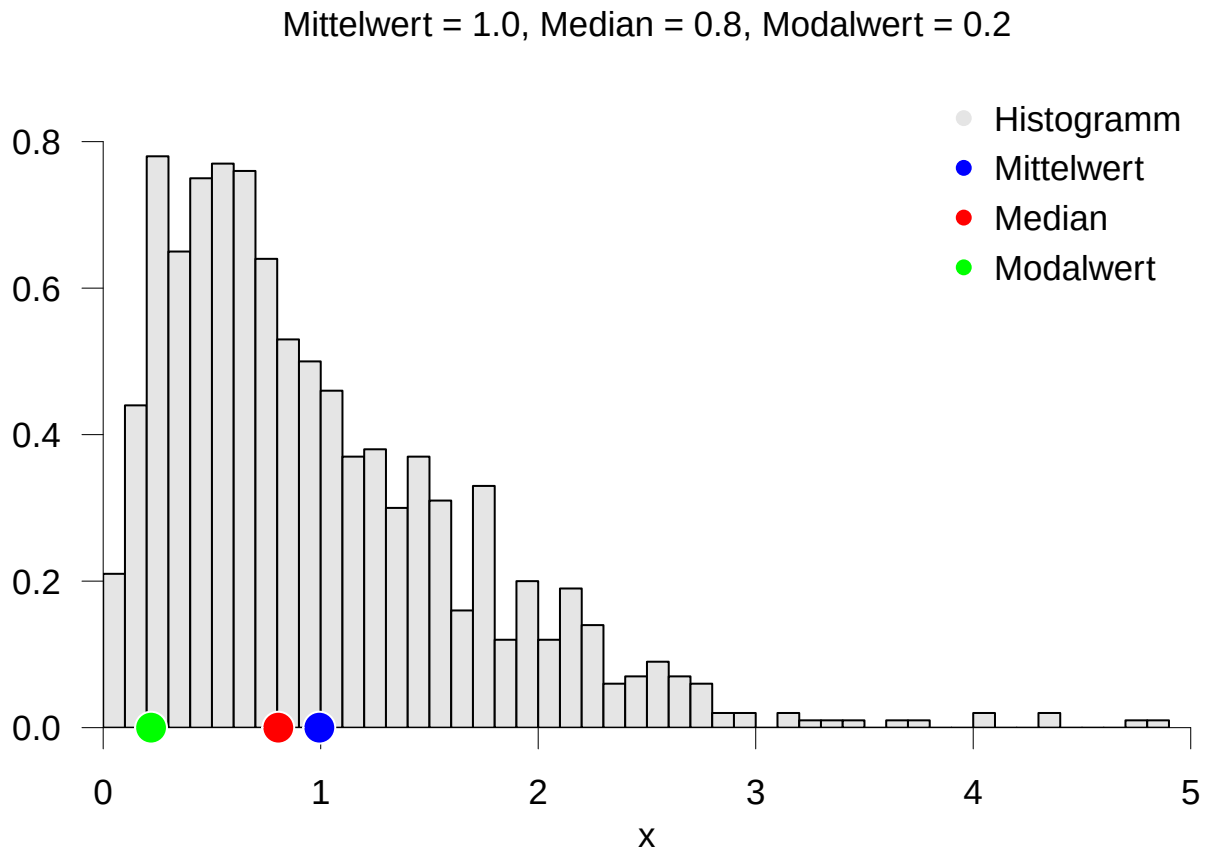
Visuelle Intuition zu Maßen zentraler Tendenz bei Gleichverteilung



Visuelle Intuition zu Maßen zentraler Tendenz bei bimodalen Verteilungen



Visuelle Intuition zu Maßen zentraler Tendenz bei nicht-symmetrischen Verteilungen



Mittelwert

Median

Modalwert

Visuelle Intuitionen

Übungen und Selbstkontrollfragen

Übungen und Selbskontrollfragen

1. Geben Sie die Definition des Mittelwertes eines Datensatzes wieder.
2. Berechnen Sie den Mittelwert der Post.BDI Daten.
3. Geben Sie das Theorem zu den Eigenschaften des Mittelwerts wieder.
4. Geben Sie die Definition des Median eines Datensatzes wieder.
5. Berechnen Sie den Median der Post.BDI Daten.
6. Wie verhalten sich Mittelwert und Median in Bezug auf Datenausreißer?
7. Geben Sie die Definition des Modalwertes eines Datensatzes wieder.
8. Berechnen Sie den Modalwert des Post.BDI Datensatzes.
9. Visualisieren Sie die Häufigkeitsverteilung des Post.BDI Datensatzes und diskutieren Sie die berechneten Werte von Mittelwert, Median und Modalwert vor dem Hintergrund dieser Häufigkeitsverteilung.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(10) Maße der Variabilität

Datum	Einheit	Thema
14.10.2021	Einführung	(1) Einführung
21.10.2021	R Grundlagen	(2) R und RStudio I
28.10.2021	R Grundlagen	(2) R und RStudio II
04.11.2021	R Grundlagen	(3) Vektoren
11.11.2021	R Grundlagen	(4) Matrizen
18.11.2021	R Grundlagen	(5) Listen und Dataframes
25.11.2021	Deskriptive Statistik	(6) Datenmanagement
02.12.2021	Deskriptive Statistik	(7) Häufigkeitsverteilungen
09.12.2021	Deskriptive Statistik	(8) Verteilungsfunktionen und Quantile
16.12.2021	Deskriptive Statistik	(9) Maße der zentralen Tendenz
	Weihnachtspause	
13.01.2022	**Deskriptive Statistik**	**(10) Maße der Datenvariabilität**
20.01.2022	Inferenzstatistik	(11) Anwendungsbeispiel
27.01.2022	Inferenzstatistik	(11) Anwendungsbeispiel
25.02.2022	Klausurtermin	G26 – H1, 9:00 - 10:00 Uhr
Jul 2022	Klausurwiederholungstermin	

Spannbreite

Stichprobenvarianz

Stichprobenstandardabweichung

Selbstkontrollfragen

Spannbreite

Stichprobenvarianz

Stichprobenstandardabweichung

Selbstkontrollfragen

Definition (Spannbreite)

$x = (x_1, ..., x_n)$ sei ein Datensatz. Dann ist die *Spannbreite* von $x_1, ..., x_n$ definiert als

$$S := \max(x_1, ..., x_n) - \min(x_1, ..., x_n). \tag{1}$$

Berechnen der Spannbreite mit `range()`

```
# Einlesen des Beispieldatensatzes
fname   = file.path(getwd(), "10_Daten", "psychotherapie_datensatz.csv")
D       = read.table(fname, sep = ",")
fdir    = file.path(getwd(), "10_Abbildungen")

# Manuelle Spanntbreitenberechnung
x       = D$Pre.BDI                         # double Vektor der Pre-BDI Werte Werte
x_max   = max(x)                            # Maximum der TA1 Werte
x_min   = min(x)                            # Mininum der TA1 Werte
S       = x_max - x_min                     # Spannbreite
print(S)
```

```
> [1] 9
```

```
# automatische Spannbreitenberechnung
MinMax  = range(x)                          # "automatische" Berechnung von min(x), max(x)
S       = MinMax[2] - MinMax[1]             # Spannbreite
print(S)
```

```
> [1] 9
```

Spannbreite

Stichprobenvarianz

Stichprobenstandardabweichung

Selbstkontrollfragen

Definition (Stichprobenvarianz, empirische Stichprobenvarianz)

$x = (x_1, \ldots, x_n)$ sei ein Datensatz. Die *Stichprobenvarianz* von x ist definiert als

$$S^2 := \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{2}$$

und die *empirische Stichprobenvarianz* von x ist definiert als

$$\tilde{S}^2 := \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2. \tag{3}$$

Bemerkungen

- S^2 ist ein unverzerrter Schätzer von $\mathbb{V}(X)$, $\tilde{S}^2$ ist ein verzerrter Schätzer $\mathbb{V}(X)$.
- Für $n \to \infty$ gilt $\frac{1}{n} \approx \frac{1}{n-1}$, $\tilde{S}^2$ ist ein asymptotisch unverzerrter Schätzer von $\mathbb{V}(X)$.
- $\tilde{S}^2$ ist der ML Schätzer, S^2 ist der ReML Schätzer von σ^2 bei $X_1, \ldots, X_n \sim N(\mu, \sigma^2)$.
- Es gelten

$$\tilde{S}^2 = \frac{n-1}{n} S^2, S^2 = \frac{n}{n-1} \tilde{S}^2 \text{ und } 0 \leq \tilde{S}^2 \leq S^2. \tag{4}$$

Berechnen der Stichprobenvarianz mit `var()`

```
x           = D$Pre.BDI                         # double Vektor der Pre-BDI Werte Werte
n           = length(x)                         # Anzahl der Werte
S2          = (1/(n-1))*sum((x - mean(x))^2)    # Stichprobenvarianz
print(S2)
```

```
> [1] 3.03
```

```
S2          = var(x)                            # "automatische" Stichprobenvarianz
print(S2)
```

```
> [1] 3.03
```

```
S2_tilde    = (1/n)*sum((x - mean(x))^2)        # Empirische Stichprobenvarianz
print(S2_tilde)
```

```
> [1] 3
```

```
S2_tilde    = ((n-1)/n)*var(x)                  # "automatische" empirische Stichprobenvarianz
print(S2_tilde)
```

```
> [1] 3
```

Theorem (Stichprobenvarianz bei linear-affinen Transformationen)

$x = (x_1, \ldots, x_n)$ sei ein Datensatz mit Stichprobenvarianz S_x^2 und $y = (ax_1 + b, \ldots, ax_n + b)$ sei der mit $a, b \in \mathbb{R}$ linear-affin transformierte Datensatz mit Stichprobenvarianz S_y^2. Dann gilt

$$S_y^2 = a^2 S_x^2. \tag{5}$$

Beweis

$$\begin{aligned} S_y^2 := \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(ax_i + b - (a\bar{x} + b))^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(ax_i + b - a\bar{x} - b)^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n}(a(x_i - \bar{x}))^2 \\ &= \frac{1}{n-1}\sum_{i=1}^{n}a^2(x_i - \bar{x})^2 = a^2\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = a^2 S_x^2 \end{aligned} \tag{6}$$

Stichprobenvarianz bei linear-affinen Transformationen

```
# Stichprobenvarianz nach Transformation
x       = D$Pre.BDI                          # double Vektor der Pre-BDI Werte Werte
S2x     = var(x)                             # Stichprobenvarianz von x_1,...,x_n
a       = 2                                  # Multiplikationskonstante
b       = 5                                  # Additionskonstante
y       = a*x + b                            # y_i = ax_i + b
S2y     = var(y)                             # Stichprobenvarianz y_1,...,y_n
print(S2y)
```

```
> [1] 12.1
```

```
# Stichprobenvarianz nach Theorem
S2y     = a^2*S2x                            # Stichprobenvarianz y_1,...,y_n
print(S2y)
```

```
> [1] 12.1
```

Theorem (Verschiebungssatz zur empirischen Stichprobenvarianz)

$x = (x_1, \ldots, x_n)$ sei ein Datensatz, $x^2 := (x_1^2, \ldots, x_n^2)$ sei sein elementweises Quadrat und $\bar{x}$ und $\overline{x^2}$ seien die respektiven Mittelwerte. Dann gilt

$$\tilde{S}^2 = \overline{x^2} - \bar{x}^2 \tag{7}$$

Beweis

$$\begin{aligned}
\tilde{S}^2 &:= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(x_i^2 - 2x_i\bar{x} + \bar{x}^2\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}x_i^2 - 2\bar{x}\frac{1}{n}\sum_{i=1}^{n}x_i + \frac{1}{n}\sum_{i=1}^{n}\bar{x}^2 \\
&= \overline{x^2} - 2\bar{x}\bar{x} + \frac{1}{n}n\bar{x}^2 \\
&= \overline{x^2} - 2\bar{x}^2 + \bar{x}^2 \\
&= \overline{x^2} - \bar{x}^2
\end{aligned} \tag{8}$$

Verschiebungssatz zur empirischen Stichprobenvarianz

```
# Direkte Berechnung der empirischen Stichprobenvarianz
x           = D$Pre.BDI                          # double Vektor der Pre-BDI Werte Werte
n           = length(x)                          # Anzahl Datenpunkte
x_bar       = mean(x)                            # Stichprobenmittel
S2_tilde    = ((n-1)/n)*var(x)                   # empirische Stichprobenvarianz
print(S2_tilde)
```

```
> [1] 3
```

```
# Berechnung der empirischen Stichprobenvarianz mit Theorem
S2_tilde    = mean(x^2) - (mean(x))^2            # \bar{x^2} - \bar{x}^2
print(S2_tilde)
```

```
> [1] 3
```

```
# Das Theorem gilt nicht für die Stichprobenvarianz
S2          = var(x)                             # S^2 \neq \bar{x^2} - \bar{x}^2
print(S2)
```

```
> [1] 3.03
```

Spannbreite

Stichprobenvarianz

Stichprobenstandardabweichung

Selbstkontrollfragen

Definition (Stichprobenstandardabweichung, empirische)

$x = (x_1, ..., x_n)$ sei ein Datensatz. Die *Stichprobenstandardabweichung* von x ist definiert als

$$S := \sqrt{S^2} \tag{9}$$

und die *empirische Stichprobenstandardabweichung* von x ist definiert als

$$\tilde{S} := \sqrt{\tilde{S}^2}. \tag{10}$$

Bemerkungen

- S ist ein verzerrter Schätzer von $\mathbb{S}(X)$.
- S^2 misst Variabilität in quadrierten Einheiten, zum Beispiel Quadratmeter (m^2).
- S misst Variabilität in unquadrierten Einheiten, zum Beispiel Meter (m).
- Es gilt

$$\tilde{S} = \sqrt{(n-1)/n}S. \tag{11}$$

Berechnung der Stichprobenstandardabweichung mit `sd()`

```
# Manuelle Berechnung der Stichprobenstandardabweichung
x       = D$Pre.BDI                                 # double Vektor der Pre-BDI Werte Werte
n       = length(x)                                 # Anzahl der Werte
S       = sqrt((1/(n-1))*sum((x - mean(x))^2))      # Standardabweichung
print(S)
```

```
> [1] 1.74
```

```
# Automatische Berechnung der Stichprobenstandardabweichung
S       = sd(x)                                     # "automatische" Berechnung
print(S)
```

```
> [1] 1.74
```

```
# Empirische Standardabweichung
S_tilde = sqrt((1/(n))*sum((x - mean(x))^2))    # empirische Standardabweichung
print(S_tilde)
```

```
> [1] 1.73
```

```
S_tilde = sqrt((n-1)/n)*sd(x)                   # empirische Standardabweichung
print(S_tilde)
```

```
> [1] 1.73
```

Theorem (Stichprobenvarianz bei linear-affinen Transformationen)

$x = (x_1, \ldots, x_n)$ sei ein Datensatz mit Stichprobenstandardabweichung S_x und $y = (ax_1 + b, \ldots, ax_n + b)$ sei der mit $a, b \in \mathbb{R}$ linear-affin transformierte Datensatz mit Stichprobenstandardabweichung S_y. Dann gilt

$$S_y = |a| S_x . \tag{12}$$

Beweis

$$\begin{aligned}
S_y := \left(\frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \right)^{1/2} &= \left(\frac{1}{n-1} \sum_{i=1}^{n} \Big(ax_i + b - (a\bar{x} + b) \Big)^2 \right)^{1/2} \\
&= \left(\frac{1}{n-1} \sum_{i=1}^{n} \Big(a(x_i - \bar{x}) \Big)^2 \right)^{1/2} \\
&= \left(\frac{1}{n-1} \sum_{i=1}^{n} a^2 (x_i - \bar{x})^2 \right)^{1/2} \\
&= \Big(a^2 \Big)^{1/2} \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1/2}
\end{aligned}$$

Also gilt $S_y = aS_x$, wenn $a \geq 0$ und $S_y = -aS_x$, wenn $a < 0$. Dies aber entspricht $S_y = |a| S_x$.

Stichprobenstandardabweichung bei linear-affinen Transformationen

```
# a >= 0
x   = D$Pre.BDI          # double Vektor der Pre-BDI Werte Werte
Sx  = sd(x)              # Stichprobenvarianz von x
a   = 2                  # Multiplikationskonstante
b   = 5                  # Additionskonstante
y   = a*x + b            # y_i = ax_i + b
Sy  = sd(y)              # Stichprobenvarianz von y
print(Sy)
```

```
> [1] 3.48
```

```
Sy  = a*Sx               # Stichprobenvarianz von y
print(Sy)
```

```
> [1] 3.48
```

```
# a < 0
x   = D$Pre.BDI          # double Vektor der Pre-BDI Werte Werte
Sx  = sd(x)              # Stichprobenvarianz von x
a   = -3                 # Multiplikationskonstante
b   = 10                 # Additionskonstante
y   = a*x + b            # y_i = ax_i + b
Sy  = sd(y)              # Stichprobenvarianz von y
print(Sy)
```

```
> [1] 5.22
```

```
Sy  = (-a)*Sx            # Stichprobenvarianz von y
print(Sy)
```

```
> [1] 5.22
```

Spannbreite

Stichprobenvarianz

Stichprobenstandardabweichung

Selbstkontrollfragen

Selbstkontrollfragen

1. Geben Sie die Definition der Spannbreite eines Datensatzes wieder.
2. Berechnen Sie die Spannbreite der Post.BDI Daten.
3. Geben Sie die Definition der Stichprobenvarianz und der empirischen Stichprobenvarianz wieder.
4. Berechnen Sie die Stichprobenvarianz und die empirische Stichprobenvarianz der Post.BDI Daten.
5. Geben Sie das Theorem zur Stichprobenvarianz bei linear-affinen Transformationen wieder.
6. Geben Sie den Verschiebungssatz zur empirischen Stichprobenvarianz wieder.
7. Geben Sie die Definition der Stichprobenstandardabweichung und der empirischen Stichprobenstandardabweichung wieder.
8. Berechnen Sie die Stichprobenstandardabweichung und die empirische Stichprobenstandardabweichung der Post.BDI Daten.
9. Geben Sie das Theorem zur Stichprobenstandardabweichung bei linear-affinen Transformationen wieder.

Programmierung und Deskriptive Statistik

BSc Psychologie WiSe 2021/22

Prof. Dr. Dirk Ostwald

(11) Anwendungsbeispiel

Beispieldatensatz

Visualisierung

Deskriptive Statistiken

Parameterschätzung

Konfidenzintervalle

Hypothesentests

Evidenzbasierte Evaluation von Psychotherapieformen bei Depression

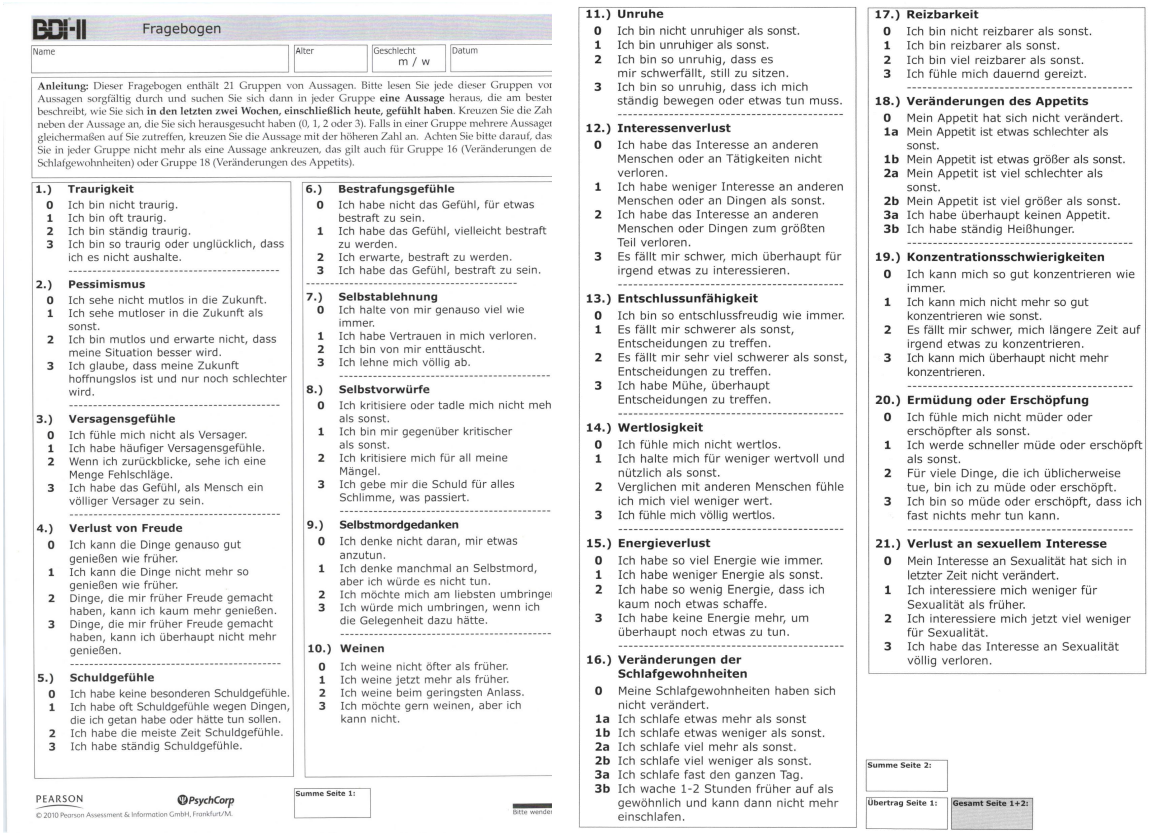
Welche Therapieform ist bei Depression wirksamer?

Online Psychotherapie

Klassische Psychotherapie

Evidenzbasierte Evaluation von Psychotherapieformen bei Depression

Becks Depressions-Inventar (BDI) zur Depressionsdiagnostik



BDI-II Fragebogen

Name | Alter | Geschlecht m / w | Datum

Anleitung: Dieser Fragebogen enthält 21 Gruppen von Aussagen. Bitte lesen Sie jede dieser Gruppen von Aussagen sorgfältig durch und suchen Sie sich dann in jeder Gruppe **eine Aussage** heraus, die am bester beschreibt, wie Sie sich **in den letzten zwei Wochen, einschließlich heute, gefühlt haben**. Kreuzen Sie die Zah neben der Aussage an, die Sie sich herausgesucht haben (0, 1, 2 oder 3). Falls in einer Gruppe mehrere Aussage gleichermaßen auf Sie zutreffen, kreuzen Sie die Aussage mit der höheren Zahl an. Achten Sie bitte darauf, das Sie in jeder Gruppe nicht mehr als eine Aussage ankreuzen, das gilt auch für Gruppe 16 (Veränderungen de Schlafgewohnheiten) oder Gruppe 18 (Veränderungen des Appetits).

1.) Traurigkeit
- 0 Ich bin nicht traurig.
- 1 Ich bin oft traurig.
- 2 Ich bin ständig traurig.
- 3 Ich bin so traurig oder unglücklich, dass ich es nicht aushalte.

2.) Pessimismus
- 0 Ich sehe nicht mutlos in die Zukunft.
- 1 Ich sehe mutloser in die Zukunft als sonst.
- 2 Ich bin mutlos und erwarte nicht, dass meine Situation besser wird.
- 3 Ich glaube, dass meine Zukunft hoffnungslos ist und nur noch schlechter wird.

3.) Versagensgefühle
- 0 Ich fühle mich nicht als Versager.
- 1 Ich habe häufiger Versagensgefühle.
- 2 Wenn ich zurückblicke, sehe ich eine Menge Fehlschläge.
- 3 Ich habe das Gefühl, als Mensch ein völliger Versager zu sein.

4.) Verlust von Freude
- 0 Ich kann die Dinge genauso gut genießen wie früher.
- 1 Ich kann die Dinge nicht mehr so genießen wie früher.
- 2 Dinge, die mir früher Freude gemacht haben, kann ich kaum mehr genießen.
- 3 Dinge, die mir früher Freude gemacht haben, kann ich überhaupt nicht mehr genießen.

5.) Schuldgefühle
- 0 Ich habe keine besonderen Schuldgefühle.
- 1 Ich habe oft Schuldgefühle wegen Dingen, die ich getan habe oder hätte tun sollen.
- 2 Ich habe die meiste Zeit Schuldgefühle.
- 3 Ich habe ständig Schuldgefühle.

6.) Bestrafungsgefühle
- 0 Ich habe nicht das Gefühl, für etwas bestraft zu sein.
- 1 Ich habe das Gefühl, vielleicht bestraft zu werden.
- 2 Ich erwarte, bestraft zu werden.
- 3 Ich habe das Gefühl, bestraft zu sein.

7.) Selbstablehnung
- 0 Ich halte von mir genauso viel wie immer.
- 1 Ich habe Vertrauen in mich verloren.
- 2 Ich bin von mir enttäuscht.
- 3 Ich lehne mich völlig ab.

8.) Selbstvorwürfe
- 0 Ich kritisiere oder tadle mich nicht meh als sonst.
- 1 Ich bin mir gegenüber kritischer als sonst.
- 2 Ich kritisiere mich für all meine Mängel.
- 3 Ich gebe mir die Schuld für alles Schlimme, was passiert.

9.) Selbstmordgedanken
- 0 Ich denke nicht daran, mir etwas anzutun.
- 1 Ich denke manchmal an Selbstmord, aber ich würde es nicht tun.
- 2 Ich möchte mich am liebsten umbringe
- 3 Ich würde mich umbringen, wenn ich die Gelegenheit dazu hätte.

10.) Weinen
- 0 Ich weine nicht öfter als früher.
- 1 Ich weine jetzt mehr als früher.
- 2 Ich weine beim geringsten Anlass.
- 3 Ich möchte gern weinen, aber ich kann nicht.

PEARSON PsychCorp © 2010 Pearson Assessment & Information GmbH, Frankfurt/M.

Summe Seite 1: | Bitte wende

11.) Unruhe
- 0 Ich bin nicht unruhiger als sonst.
- 1 Ich bin unruhiger als sonst.
- 2 Ich bin so unruhig, dass es mir schwerfällt, still zu sitzen.
- 3 Ich bin so unruhig, dass ich mich ständig bewegen oder etwas tun muss.

12.) Interessenverlust
- 0 Ich habe das Interesse an anderen Menschen oder an Tätigkeiten nicht verloren.
- 1 Ich habe weniger Interesse an anderen Menschen oder an Dingen als sonst.
- 2 Ich habe das Interesse an anderen Menschen oder Dingen zum größten Teil verloren.
- 3 Es fällt mir schwer, mich überhaupt für irgend etwas zu interessieren.

13.) Entschlussunfähigkeit
- 0 Ich bin so entschlussfreudig wie immer.
- 1 Es fällt mir schwerer als sonst, Entscheidungen zu treffen.
- 2 Es fällt mir sehr viel schwerer als sonst, Entscheidungen zu treffen.
- 3 Ich habe Mühe, überhaupt Entscheidungen zu treffen.

14.) Wertlosigkeit
- 0 Ich fühle mich nicht wertlos.
- 1 Ich halte mich für weniger wertvoll und nützlich als sonst.
- 2 Verglichen mit anderen Menschen fühle ich mich viel weniger wert.
- 3 Ich fühle mich völlig wertlos.

15.) Energieverlust
- 0 Ich habe so viel Energie wie immer.
- 1 Ich habe weniger Energie als sonst.
- 2 Ich habe so wenig Energie, dass ich kaum noch etwas schaffe.
- 3 Ich habe keine Energie mehr, um überhaupt noch etwas zu tun.

16.) Veränderungen der Schlafgewohnheiten
- 0 Meine Schlafgewohnheiten haben sich nicht verändert.
- 1a Ich schlafe etwas mehr als sonst
- 1b Ich schlafe etwas weniger als sonst.
- 2a Ich schlafe viel mehr als sonst.
- 2b Ich schlafe viel weniger als sonst.
- 3a Ich schlafe fast den ganzen Tag.
- 3b Ich wache 1-2 Stunden früher auf als gewöhnlich und kann dann nicht mehr einschlafen.

17.) Reizbarkeit
- 0 Ich bin nicht reizbarer als sonst.
- 1 Ich bin reizbarer als sonst.
- 2 Ich bin viel reizbarer als sonst.
- 3 Ich fühle mich dauernd gereizt.

18.) Veränderungen des Appetits
- 0 Mein Appetit hat sich nicht verändert.
- 1a Mein Appetit ist etwas schlechter als sonst.
- 1b Mein Appetit ist etwas größer als sonst.
- 2a Mein Appetit ist viel schlechter als sonst.
- 2b Mein Appetit ist viel größer als sonst.
- 3a Ich habe überhaupt keinen Appetit.
- 3b Ich habe ständig Heißhunger.

19.) Konzentrationsschwierigkeiten
- 0 Ich kann mich so gut konzentrieren wie immer.
- 1 Ich kann mich nicht mehr so gut konzentrieren wie sonst.
- 2 Es fällt mir schwer, mich längere Zeit auf irgend etwas zu konzentrieren.
- 3 Ich kann mich überhaupt nicht mehr konzentrieren.

20.) Ermüdung oder Erschöpfung
- 0 Ich fühle mich nicht müder oder erschöpfter als sonst.
- 1 Ich werde schneller müde oder erschöpft als sonst.
- 2 Für viele Dinge, die ich üblicherweise tue, bin ich zu müde oder erschöpft.
- 3 Ich bin so müde oder erschöpft, dass ich fast nichts mehr tun kann.

21.) Verlust an sexuellem Interesse
- 0 Mein Interesse an Sexualität hat sich in letzter Zeit nicht verändert.
- 1 Ich interessiere mich weniger für Sexualität als früher.
- 2 Ich interessiere mich jetzt viel weniger für Sexualität.
- 3 Ich habe das Interesse an Sexualität völlig verloren.

Summe Seite 2:

Übertrag Seite 1: | Gesamt Seite 1+2:

0 - 8 keine Depression

9 - 13 minimale Depression

14 - 19 leichte Depression

20 - 28 mittelschwere Depression

29 - 63 schwere Depression

Beispiel: Evaluation von Psychotherapieformen bei Depression

Experimentelle Bedingung
(Gruppen von n = 50)

Psychotherapie

Klassisch

Pre-BDI

Post-BDI

Online

Pre-BDI

Post-BDI

Beispieldatensatz

Einlesen des Datensatzes mit `read.table()`

```
fname       = file.path(getwd(), "11_Daten", "psychotherapie_datensatz.csv")
D           = read.table(fname, sep = ",", header = TRUE)
```

Daten der ersten acht Proband:innen jeder Gruppe

	Bedingung	Pre.BDI	Post.BDI
1	Klassisch	17	9
2	Klassisch	20	14
3	Klassisch	16	13
4	Klassisch	18	12
5	Klassisch	21	12
6	Klassisch	17	14
7	Klassisch	17	12
8	Klassisch	17	9
51	Online	22	16
52	Online	19	15
53	Online	21	13
54	Online	18	15
55	Online	19	13
56	Online	17	16
57	Online	20	13
58	Online	19	16

Beispieldatensatz

Datensatzübersicht mit `View()`

D

Filter

	Bedingung	Pre BDI	Post BDI
1	Klassisch	17	9
2	Klassisch	20	14
3	Klassisch	16	13
4	Klassisch	18	12
5	Klassisch	21	12
6	Klassisch	17	14
7	Klassisch	17	12
8	Klassisch	17	9
9	Klassisch	18	11
10	Klassisch	18	14
11	Klassisch	20	10
12	Klassisch	17	15
13	Klassisch	16	17
14	Klassisch	18	12
15	Klassisch	16	10
16	Klassisch	18	13

Datenvorverarbeitung

- Studienfokus ist die **Veränderung** der Depressionsymptomatik durch Therapieformen.
- Für jede Proband:in der ergibt sich diese Veränderung als **Differenz** von Post.BDI - Pre.BDI.
- Eine Reduktion der Depressionssymptomatik ergibt dabei eine **negative Zahl**.
- Es ist sinnvoller, Verbesserungen mit **positiven Zahlen** zu repräsentieren.
- Als Maß des Therapieeffekts bei Proband:in i bietet sich also an

$$\Delta\text{BDI[i]} := -(\text{Post.BDI[i]} - \text{Pre.BDI[i]}) \tag{1}$$

- Wir betrachten in der Folge also das ΔBDI Maß mit folgender Interpretation

$\Delta\text{BDI} > 0$	Verminderung der Depressionsymptomatik	Wirksame Therapie
$\Delta\text{BDI} = 0$	Keine Veränderung der Depressionsymptomatik	Wirkungslose Therapie
$\Delta\text{BDI} < 0$	Verstärkung der Depressionsymptomatik	Schädigende Therapie

Beispieldatensatz

Datenvorverarbeitung

Hinzufügen einer ΔBDI Spalte zum Dataframe

```
fname       = file.path(getwd(), "11_Daten", "psychotherapie_datensatz.csv") # Einlesen
D           = read.table(fname, sep = ",", header = TRUE)                    # Rohdaten
D$Delta.BDI = -(D$Post.BDI - D$Pre.BDI)                                      # \Delta BDI Maß
```

Daten der ersten acht Proband:innen jeder Gruppe

	Bedingung	Pre.BDI	Post.BDI	Delta.BDI
1	Klassisch	17	9	8
2	Klassisch	20	14	6
3	Klassisch	16	13	3
4	Klassisch	18	12	6
5	Klassisch	21	12	9
6	Klassisch	17	14	3
7	Klassisch	17	12	5
8	Klassisch	17	9	8
51	Online	22	16	6
52	Online	19	15	4
53	Online	21	13	8
54	Online	18	15	3
55	Online	19	13	6
56	Online	17	16	1
57	Online	20	13	7
58	Online	19	16	3

Beispieldatensatz

Visualisierung

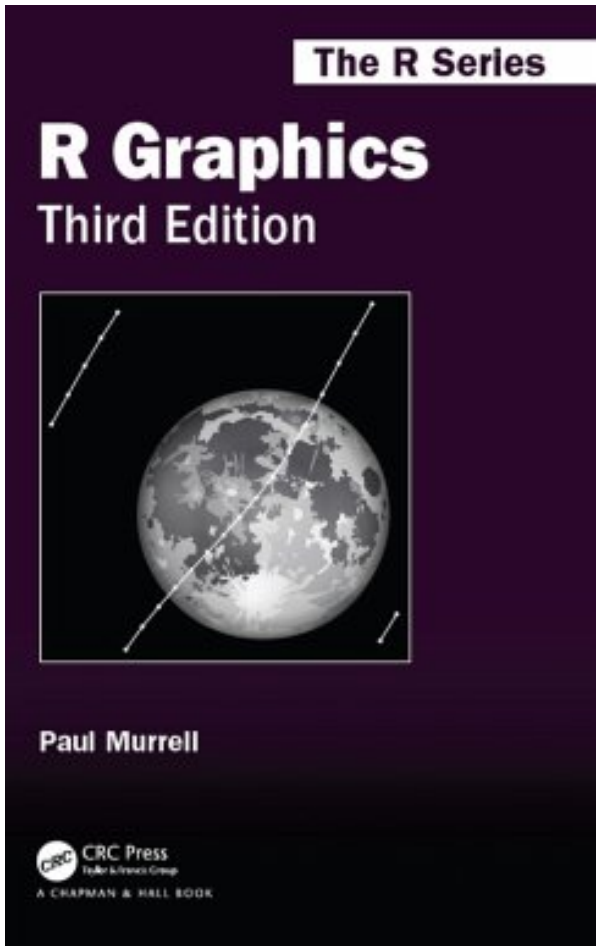
Deskriptive Statistiken

Parameterschätzung

Konfidenzintervalle

Hypothesentest

Visualisierung



R Funktionalitäten für Abbildungen

Base Graphics

- Erstellung und bedarfsgerechte Anpassung von Abbildungen
- Eher low-level, fine tuning orientiert

Lattice und ggplot2

- Erstellung und bedarfsgerechte Anpassung von Abbildungen
- Eher high level, an der eigenen Philosophie orientiert

Base Graphics, lattice und ggplot2 können ähnliche Abbildungen generieren

LaTeX Typesetting ist in allen Paketen unterentwickelt

R Funktionalitäten für Abbildungen

Base Graphics

- **Erstellung und bedarfsgerechte Anpassung von Abbildungen**
- **Eher low-level, fine tuning orientiert**

Lattice und ggplot2

- Erstellung und bedarfsgerechte Anpassung von Abbildungen
- Eher high level, an der eigenen Philosophie orientiert

Base Graphics, lattice und ggplot2 können ähnliche Abbildungen generieren

LaTeX Typesetting ist in allen Paketen unterentwickelt

Visualisierung

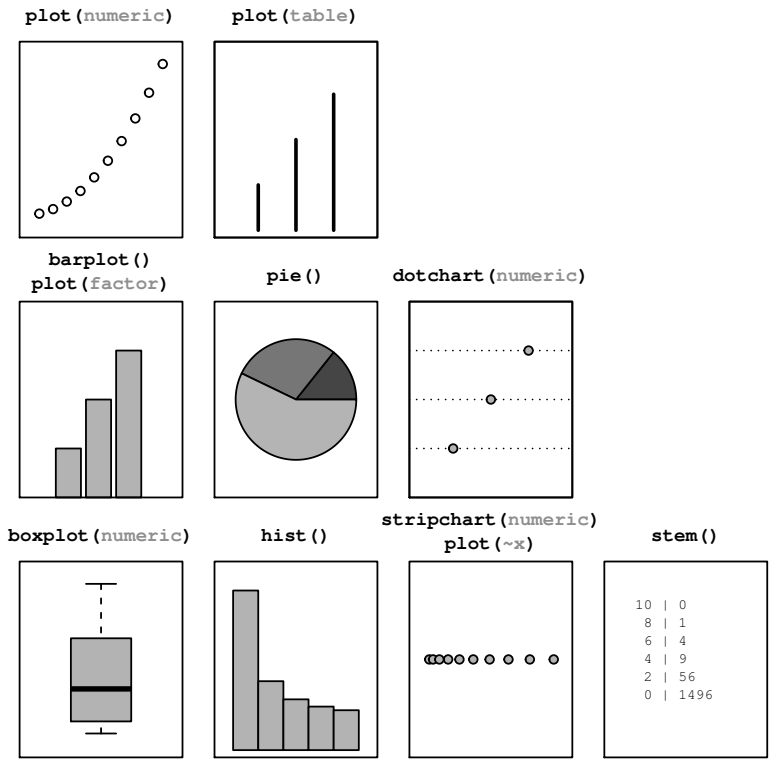


Figure 2.5
High-level base graphics plotting functions for producing plots of a single variable. Where the function can be used to produce more than one type of plot, the relevant data type is shown (in gray). For example, `plot(numeric)` means that this is what the `plot()` produces when it is given a single numeric argument.

Murrell (2019)

Visualisierung

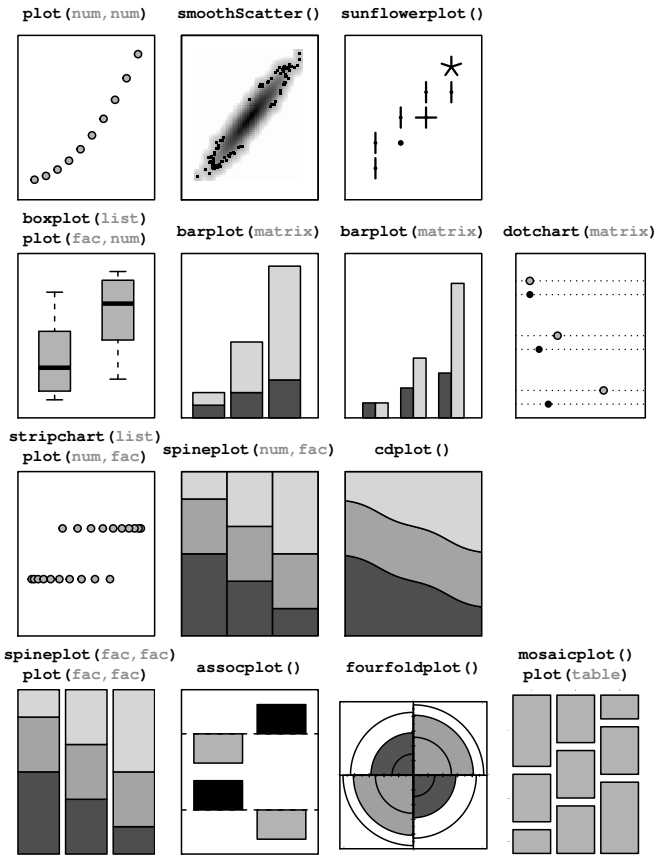


Figure 2.6
High-level base graphics plotting functions for producing plots of two variables. Where the function can be used to produce more than one type of plot, the relevant data type is shown (in gray). For example `plot(num,fac)` represents calling the `plot()` function with a numeric vector as the first argument and a factor as the second argument.

Murrell (2019)

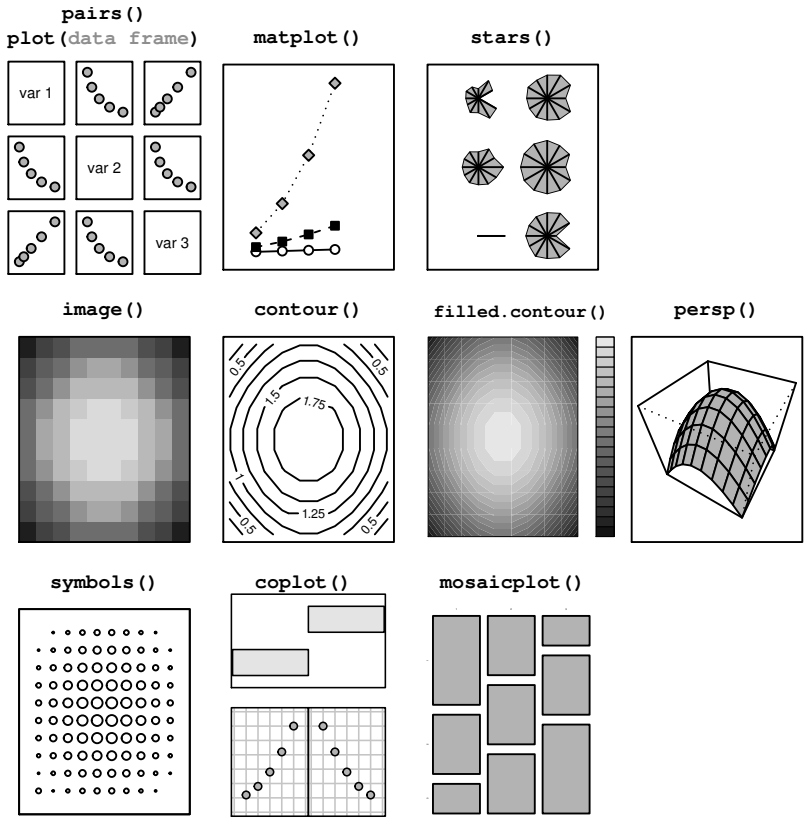


Figure 2.7
High-level base graphics plotting functions for producing plots of many variables. Where the function can be used to produce more than one type of plot, the relevant data type is shown (in gray).

Murrell (2019)

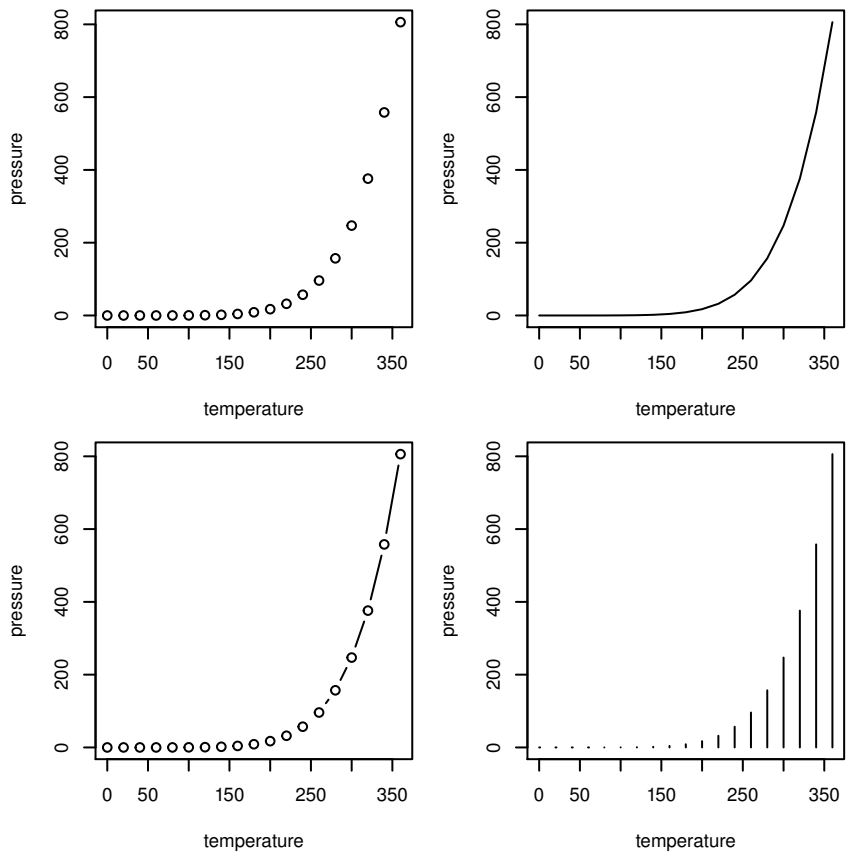


Figure 2.2
Four variations on a scatterplot. In each case, the plot is produced by a call to the `plot()` function with the same data; all that changes is the value of the `type` argument. At top-left, `type="p"` to give **p**oints (data symbols), at top-right, `type="l"` to give **l**ines, at bottom-left, `type="b"` to give **b**oth, and at bottom-right, `type="h"` to give **h**istogram-like vertical lines.

Murrell (2019)

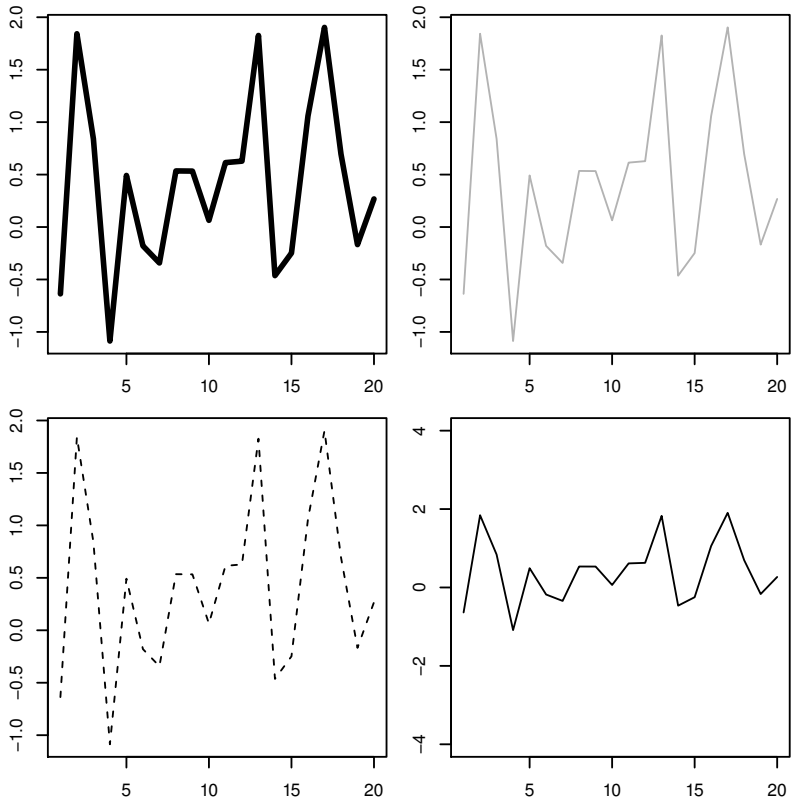


Figure 2.9
Standard arguments for high-level functions. All four plots are produced by calls to the `plot()` function with the same data, but with different standard plot function arguments specified: the top-left plot makes use of the `lwd` argument to control line thickness; the top-right plot uses the `col` argument to control line color; the bottom-left plot makes use of the `lty` argument to control line type; and the bottom-right plot uses the `ylim` argument to control the scale on the y-axis.

Murrell (2019)

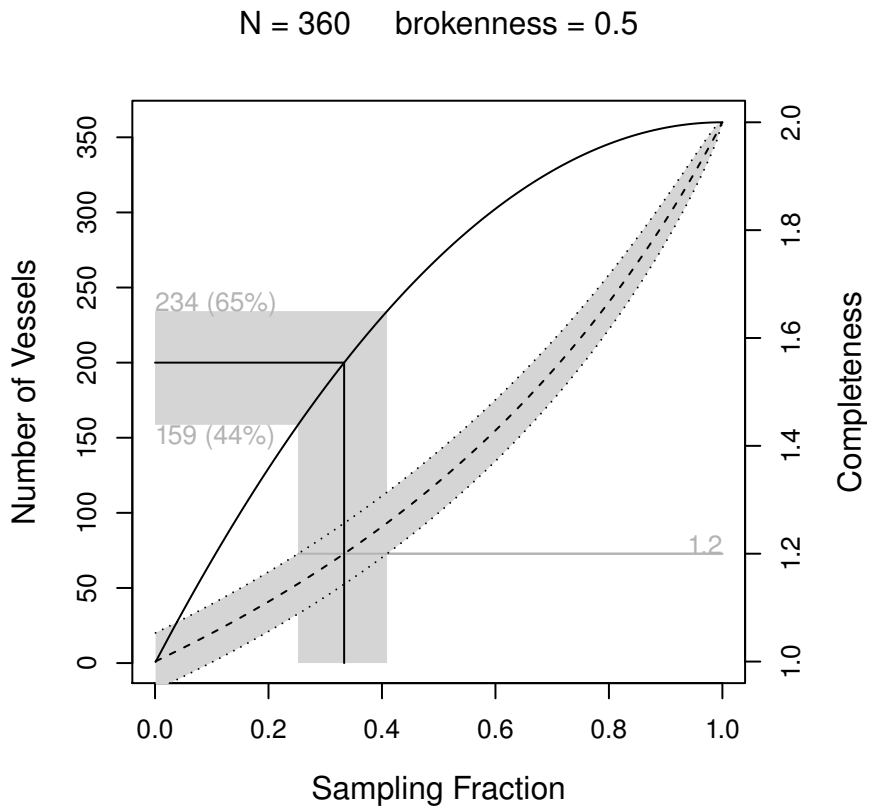


Figure 1.3
A customized scatterplot produced using R. This is created by starting with a simple scatterplot and augmenting it by adding an additional y-axis and several additional sets of lines, polygons, and text labels.

Murrell (2019)

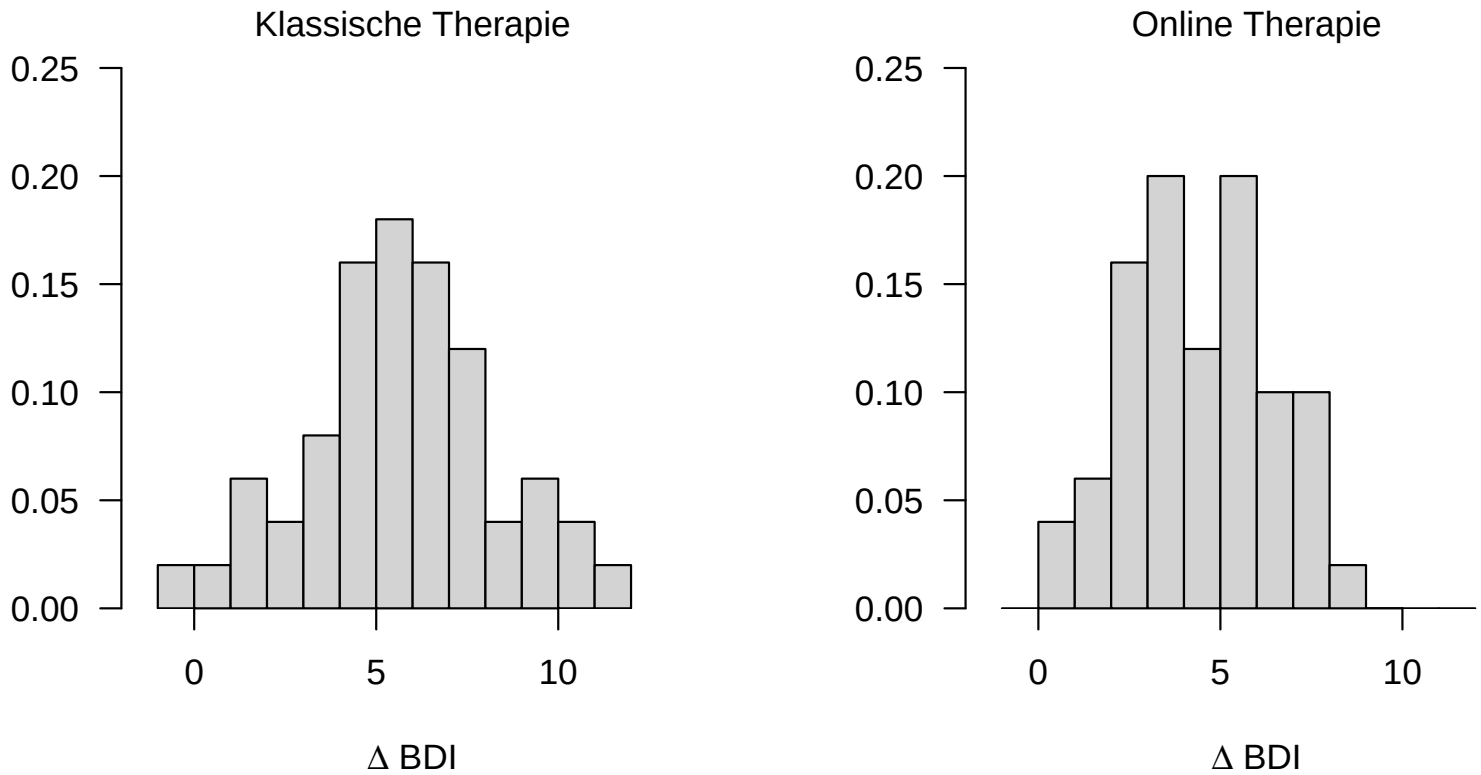
Code Outline

```
# Initialisierung einer neuen Abbildung
dev.new()

# Abbildungsparameter
par(
z.B. Arrangement von Panels, Begrenzungsstile, Schriftfonts, etc
)

# Higher-level Abbildungsfunktion wie plot(), hist(), barplot(), ...
plot(
z.B. x- und y-Daten, Achsenlimits, Achsenbeschriftungen, Titel, Farben, etc.
Jeder Aufruf einer higher-level Graphikfunktion belegt ein neues Subpanel!
)

# Hinzufügen weiterer Daten mit lower-level Abbildungsfunktionen zum aktuellen Panel
z.B. points(), lines(), abline()

# Weitere Graphikannotation zu aktuellem Panel
z.B. legend(), text()

# Speichern der Abbildung (Größenverhältnisse erst hier final festgelegt)
z.B. dev.copy2pdf()
```

Histogramme

```
# Histogrammparameter
h           = 1                                 # gewünschte Klassenbreite
b_0         = min(D$Delta.BDI)                  # b_0
b_k         = max(D$Delta.BDI)                  # b_0
k           = ceiling((b_k - b_0)/h)            # Anzahl der Klassen
b           = seq(b_0, b_k, by = h)             # Klassen [b_{j-1}, b_j[
ylimits     = c(0,.25)                          # y-Achsenlimits
xlimits     = c(-2,14)                          # x-Achsenlimits
therapie    = c("Klassisch" , "Online")         # Therapiebedingungen
labs        = c("Klassische Therapie",          # Abbildungslabel
                "Online Therapie")
# Abbildungsparameter
par(                                            # für Details siehe ?par
mfcol       = c(1,2),                           # 1 x 2 Panelstruktur
family      = "sans",                           # Serif-freier Fonttyp
pty         = "m",                              # Maximale Abbildungsregion
bty         = "l",                              # L förmige Box
las         = 1,                                # Horizontale Achsenbeschriftung
xaxs        = "i",                              # x-Achse bei y = 0
yaxs        = "i",                              # y-Achse bei x = 0
font.main   = 1,                                # Non-Bold Titel
cex         = 1,                                # Textvergrößerungsfaktor
cex.main    = 1)                                # Titeltextvergrößerungsfaktor

# Iteration über Therapiebedingungen
for(i in 1:2){
  hist(
  D$Delta.BDI[D$Bedingung == therapie[i]],      # Delta.BDI Werte von Therapiebedingung i
  breaks    = b,                                # Histogrammklassen
  freq      = F,                                # normierte relative Häufigkeit
  xlim      = xlimits,                          # x-Achsenlimits
  ylim      = ylimits,                          # y-Achsenlimits
  xlab      = TeX("$\\Delta$ BDI"),             # x-Achsenbeschriftung
  ylab      = "",                               # y-Achsenbeschriftung
  main      = labs[i])                          # Titelbeschriftung
}

# PDF Speicherung
dev.copy2pdf(
file        = file.path(getwd(), "11_Abbildungen", "pds_11_histogramm.pdf"),
width       = 8,
height      = 4)
```

Histogramme



Beispieldatensatz

Visualisierung

Deskriptive Statistiken

Parameterschätzung

Konfidenzintervalle

Hypothesentest

Bedingungsabhängige Auswertung deskriptiver Statistiken

```
# Initialisierung eines Dataframes
tp            = c("Klassisch", "Online")          # Therapiebedingungen
ntp           = length(tp)                        # Anzahl Therapiebedingungen
S             = data.frame(                       # Dataframeerzeugung
                n         = rep(NaN,ntp),         # Stichprobengrößen
                Max       = rep(NaN,ntp),         # Maxima
                Min       = rep(NaN,ntp),         # Minima
                Median    = rep(NaN,ntp),         # Mediane
                Mean      = rep(NaN,ntp),         # Mittelwerte
                Var       = rep(NaN,ntp),         # Varianzen
                Std       = rep(NaN,ntp),         # Standardabweichungen
                row.names = tp)                   # Therapiebedingungen

# Iterationen über Therapiebedingungen
for(i in 1:ntp){
  data        = D$Delta.BDI[D$Bedingung == tp[i]]   # Daten
  S$n[i]      = length(data)                        # Stichprobengröße
  S$Max[i]    = max(data)                           # Maxima
  S$Min[i]    = min(data)                           # Minima
  S$Median[i] = median(data)                        # Mediane
  S$Mean[i]   = mean(data)                          # Mittelwerte
  S$Var[i]    = var(data)                           # Varianzen
  S$Std[i]    = sd(data)                            # Standardabweichungen
}
```

Bedingungsabhängige Auswertung deskriptiver Statistiken

```
# Ausgabe
print.AsIs(S)
```

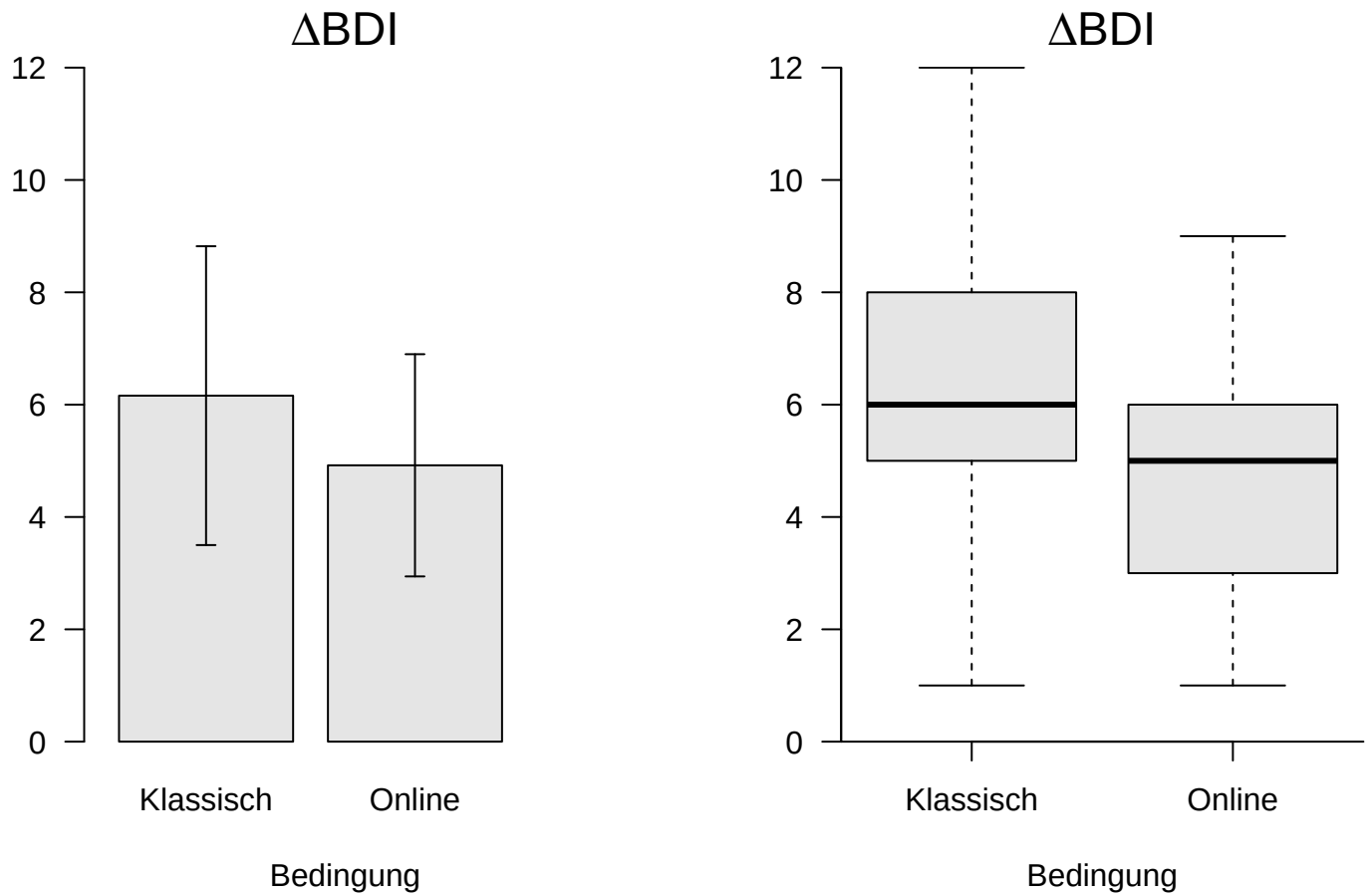
```
>            n Max Min Median Mean  Var  Std
> Klassisch 50  12  -1      6 6.16 7.08 2.66
> Online    50   9   1      5 4.92 3.91 1.98
```

- Die Anzahl der Proband:innen in beiden Therapiegruppen ist gleich.
- Die Spannbreite der ΔBDI Daten ist in der klassischen Therapieform leicht erhöht.
- Median und Mittelwert nehmen für die klassische Therapieform leicht höhere Werte an.
- Ein ΔBDI Mittelwertsunterschied von 1 ist klinisch wohl eher vernachlässigbar.
- Median und Mittelwert sind in beiden Therapieformen ähnlich (unimodale Verteilung).
- Die Variabilitätsmaße zeigen eine etwas erhöhte Varabilität in der klassischen Therapieform.

Bedingungsabhängige Visualisierung deskriptiver Statistiken

```
# Abbildungsparameter
par(                                    # für Details siehe ?par
mfcol       = c(1,2),                   # 1 x 2 Panelstruktur
family      = "sans",                   # Serif-freier Fonttyp
pty         = "m",                      # Maximale Abbildungsregion
bty         = "l",                      # L förmige Box
las         = 1,                        # Horizontale Achsenbeschriftung
xaxs        = "i",                      # x-Achse bei y = 0
yaxs        = "i",                      # y-Achse bei x = 0
font.main   = 1,                        # Non-Bold Titel
cex         = 1,                        # Textvergrößerungsfaktor
cex.main    = 1.5)                      # Titeltextvergrößerungsfaktor

# Linkes Panel: Balkendiagramm mit Fehlerbalken
mw          = S$Mean                    # Gruppenmittelwert
sd          = S$Std                     # Gruppenstandardabweichung
names(mw)   = tp                        # barplot braucht x-Werte als names
x = barplot(                            # Ausgabe der x-Ordinaten (?barplot für Details)
mw,                                     # Mittelwerte = Balkenhöhe
col         = "gray90",                 # Balkenfarbe
ylim        = c(0,12),                  # y-Achsenbegrenzung
xlim        = c(0,3),                   # x-Achsenbegrenzung
xlab        = "Bedingung",              # x-Achsenbeschriftung
main        = TeX("$\\Delta BDI$"))     # Titel
arrows(                                 # arrows() für Fehlerbalken (siehe ?arrows)
x0          = x,                        # arrow start x-ordinate
y0          = mw - sd,                  # arrow start y-ordinate
x1          = x,                        # arrow end   x-ordinate
y1          = mw + sd,                  # arrow end   y-ordinate
code        = 3,                        # Pfeilspitzen beiderseits
angle       = 90,                       # Pfeilspitzenwinkel -> Linie
length      = 0.05)                     # Linielänge

# Rechtes Panel: Boxplot
boxplot(
D$Delta.BDI ~ D$Bedingung,              # Gruppierung der Delta.BDI Daten nach D$Bedingung
ylim        = c(0,12),                  # y-Achsenbegrenzung
col         = "gray90",                 # Boxfarbe
ylab        = "",                       # y-Achsenbeschriftung
xlab        = "Bedingung",              # x-Achsenbeschriftung
main        = TeX("$\\Delta BDI$"))     # Titel
```

Bedingungsabhängige Visualisierung deskriptiver Statistiken



Beispieldatensatz

Visualisierung

Deskriptive Statistiken

Parameterschätzung

Konfidenzintervalle

Hypothesentests

Modellannahmen für Parameterschätzung und Konfidenzintervalle

Motiviert durch die therapieabhängige Visualisierung der ΔBDI Daten und unseren wissenschaftssoziologischen Kontext legen wir nun das Normalverteilungsmodell zugrunde.

Wir nehmen also an, dass die Δ BDI Werte Realisierungen von unabhängig verteilten Zufallsvariablen

$$X_{ij} \sim N\left(\mu_i, \sigma_i^2\right), i = 1, 2, j = 1, ..., 50 \tag{2}$$

sind, wobei i die Therapiebedingung (1 = Klassisch, 2 = Online) und j den Proband:innen Index in der $i$ten experimentellen Bedingung bezeichnen. Innerhalb einer Bedingung sind diese Zufallsvariablen also unabhängig und identisch verteilt.

Dies entspricht der Annahme, dass sich der ΔBDI Wert einer Proband:in durch Addition einer normalverteilten Fehlervariable mit Erwartungswertparameter 0 und Varianzparameter σ_i^2 zu den innerhalb einer Therapiebedingung identischen Wert μ_i ergibt.

Parameterschätzung

Zur Parameterschätzung im vorliegenden Modell nutzen wir

- den Maximum Likelihood Schätzer für μ_i
- den Varianzschätzer für σ^2

```
# Initialisierung eines Dataframes
tp                = c("Klassisch", "Online")                # Therapiebedingungen
ntp               = length(tp)                              # Anzahl Therapiebedingungen
S                 = data.frame(                             # Datafrmeerzeugung
  mu_ML      = rep(NaN,ntp),                                # ML Schätzer für \mu_i
  sigsqr_VAR = rep(NaN,ntp))                                # Varianzschätzer für \sigma^2_i

# Iterationen über Therapiebedingungen
for(i in 1:ntp){
  data            = D$Delta.BDI[D$Bedingung == tp[i]]       # Daten
  S$mu_ML[i]      = mean(data)                              # ML Schätzer für \mu_i
  S$sigsqr_VAR[i] = var(data)                               # Varianzschätzer für \sigma_2^_i
}

# Ausgabe
print.AsIs(S)
```

```
>   mu_ML sigsqr_VAR
> 1  6.16       7.08
> 2  4.92       3.91
```

Tipps für μ_i und σ_i^2 auf Grundlage dieser unverzerrten Schätzer sind also

$$\hat{\mu}_1 = 6.16, \quad \hat{\mu}_2 = 4.92, \quad \hat{\sigma}_1^2 = 7.08, \quad \hat{\sigma}_2^2 = 3.91. \tag{3}$$

Die mit diesen Tipps assoziierte Unsicherheit ist hier nicht angegeben.

Beispieldatensatz

Visualisierung

Deskriptive Statistiken

Parameterschätzung

Konfidenzintervalle

Hypothesentest

Konfidenzintervalle für die Erwartungswertparameterschätzer

```
# Analyseparameter
t       = c("Klassisch", "Online")                  # Therapiebedingungen
ntp     = length(tp)                                # Anzahl an Therapiebedingungen
n       = 50                                        # Anzahl von Beobachtungen pro Therapiebedingung
C       = data.frame(                               # Datarameerzeugung
          G_u       = rep(NaN,ntp),                 # untere KI Grenze
          mu_hat    = rep(NaN,ntp),                 # Erwartungswertparameterschätzer
          G_o       = rep(NaN,ntp),                 # obere KI Grenze
          row.names = tp)                           # Therapiebedingungen


# Konfidenzintervallparameter
delta   = 0.95                                      # Konfidenzlevel
psi_inv = qt((1+delta)/2,n-1)                       # \psi^-1((\delta + 1)/2, n-1)

# Konfidenzintervallevaluation
for(i in 1:ntp){
   data        = D$Delta.BDI[D$Bedingung == t[i]]   # Stichprobenrealisierung
   X_bar       = mean(data)                         # Stichprobenmittel
   S           = sd(data)                           # Stichprobenstandardabweichung
   C$G_u[i]    = X_bar - (S/sqrt(n))*psi_inv        # untere KI Grenze
   C$mu_hat[i] = X_bar                              # Erwartungswertparameterschätzer
   C$G_o[i]    = X_bar + (S/sqrt(n))*psi_inv        # obere KI Grenze
}

# Ausgabe
print.AsIs(C)
```

```
>            G_u mu_hat  G_o
> Klassisch 5.40   6.16 6.92
> Online    4.36   4.92 5.48
```

Konfidenzintervalle für die Varianzparameterschätzer

```
# Analyseparameter
t       = c("Klassisch", "Online")                    # Therapiebedingungen
ntp     = length(tp)                                  # Anzahl an Therapiebedingungen
n       = 50                                          # Anzahl von Beobachtungen pro Therapiebedingung
C       = data.frame(                                 # Dataframeerzeugung
          G_u        = rep(NaN,ntp),                  # untere KI Grenze
          sigsqr_hat = rep(NaN,ntp),                  # Varianzparameterschätzer
          G_o        = rep(NaN,ntp),                  # obere KI Grenze
          row.names  = tp)                            # Therapiebedingungen


# Konfidenzintervallparameter
delta   = 0.95                                        # Konfidenzlevel
xi_1    = qchisq((1-delta)/2, n - 1)                  # \Xi^2((1-\delta)/2; n - 1)
xi_2    = qchisq((1+delta)/2, n - 1)                  # \Xi^2((1+\delta)/2; n - 1)

# Konfidenzintervallevaluation
for(i in 1:ntp){
   data            = D$Delta.BDI[D$Bedingung == t[i]] # Stichprobenrealisierung
   S2              = var(data)                        # Stichprobenvarianz
   C$G_u[i]        = (n-1)*S2/xi_2                    # untere KI Grenze
   C$sigsqr_hat[i] = S2                               # Varianzparameterschätzer
   C$G_o[i]        = (n-1)*S2/xi_1                    # obere KI Grenze
}

# Ausgabe
print.AsIs(C)
```

```
>            G_u sigsqr_hat   G_o
> Klassisch 4.94       7.08 10.99
> Online    2.73       3.91  6.07
```

Beispieldatensatz

Visualisierung

Deskriptive Statistiken

Parameterschätzung

Konfidenzintervalle

Hypothesentest

Anwendungsszenario, Modellannahmen und Hypothese

Es liegen zwei Stichproben experimenteller Einheiten (Klassische Bedingung, Online Bedingung) vor. Wir nehmen unabhängige identische Normalverteilungen $N(\mu_1, \sigma^2)$ und $N(\mu_2, \sigma^2)$ an, wobei die Parameter μ_1, μ_2, σ^2 unbekannt sind. Wir beabsichtigen das Quantifizieren der Unsicherheit beim inferentiellen Vergleich von μ_1 mit μ_2.

Wir führen also einen Zweistichproben-T-Test bei unabhängigen Stichproben unter Annahme identischer Varianz durch und wollen die Hypothesen $H_0 : \mu_1 = \mu_2$ und $H_1 : \mu_1 \neq \mu_2$ mit einem Signifikanzniveau von $\alpha_0 = 0.05$ testen.

Manueller Zweistichproben-T-Test

```
# Datenauswahl
x_1       = D$Delta.BDI[D$Bedingung == "Klassisch"]            # \Delta.BDI Daten Klassische Therapie
x_2       = D$Delta.BDI[D$Bedingung == "Online"]               # \Delta.BDI Daten Klassische Therapie
n_1       = length(x_1)                                        # Stichprobengröße n_1
n_2       = length(x_2)                                        # Stichprobengröße n_2
alpha_0   = 0.05                                               # Signifikanzniveau
k_alpha_0 = qt(1 - (alpha_0/2), n_1+n_2-2)                     # kritischer Wert
x_bar_1   = mean(x_1)                                          # x_bar_1
x_bar_2   = mean(x_2)                                          # x_bar_2
s_12      = sqrt((sum((x_1-x_bar_1)^2)+sum((x_2-x_bar_2)^2))/  # gepoolte Standardabweichung s_12
                 (n_1+n_2-2))
t       = sqrt((n_1*n_2)/(n_1+n_2))*((x_bar_1-x_bar_2)/s_12)   # Zweistichproben-T-Teststatistik
if(abs(t) >= k_alpha_0){                                       # Test 1_{|T(X) >= k_alpha_0|}
    phi = 1                                                    # Ablehnen von H_0
} else {
    phi = 0                                                    # Nicht Ablehnen von H_0
}
pval      = 2*(1 - pt(abs(t), n_1+n_2-2))                      # p-Wert
```

Manueller Zweistichproben-T-Test

```
# Ausgabe
cat("\nx_bar_1   = ", x_bar_1,
    "\nx_bar_2   = ", x_bar_2,
    "\nfg        = ", n_1 + n_2 - 2,
    "\nalpha_0   = ", alpha_0,
    "\nk_alpha_0 = ", k_alpha_0,
    "\nt         = ", t,
    "\nphi       = ", phi,
    "\np-Wert    = ", pval)
```

```
>
> x_bar_1   =  6.16
> x_bar_2   =  4.92
> fg        =  98
> alpha_0   =  0.05
> k_alpha_0 =  1.98
> t         =  2.65
> phi       =  1
> p-Wert    =  0.00951
```

$\Rightarrow$ Wir lehnen die Nullhypothese $H_0 : \mu_1 = \mu_2$ ab.

R Implementation des Zweistichproben-T-Tests

```
# Automatischer Zweistichproben-T-Test
varphi    = t.test(                          # ?t.test für Details
            x_1,                             # Datensatz x_1
            x_2,                             # Datensatz x_2
            var.equal   = TRUE,              # \sigma_1^2 = \sigma_2^2
            alternative = c("two.sided"),    # H_1: \mu_1 \neq \mu_2
            conf.level  = 1-alpha_0)         # \delta = 1 - \alpha_0 (sic!)

# Ausgabe
print(varphi)
```

```
>
>   Two Sample t-test
>
> data:  x_1 and x_2
> t = 3, df = 98, p-value = 0.01
> alternative hypothesis: true difference in means is not equal to 0
> 95 percent confidence interval:
>  0.31 2.17
> sample estimates:
> mean of x mean of y
>      6.16      4.92
```

```
# Genauere Ausgabe t
paste(varphi[1])
```

```
> [1] "c(t = 2.64516155336263)"
```

```
# Genauere Ausgabe p
paste(varphi[3])
```

```
> [1] "0.00951137026459394"
```

References

Murrell, Paul. 2019. *R Graphics*. Third edition. The R Series. Boca Raton: CRC Press, Taylor & Francis Group.