Computergestützte Datenanalyse

BSc Psychologie SoSe 2021

Prof. Dr. Dirk Ostwald

(5) Datenmanagement

Datenmanagement

- FAIR Prinzipien
- Datenformate
- Directory Management
- Datenimport und Datenexport
- Übungen und Selbstkontrollfragen

Datenmanagement

- **FAIR Prinzipien**
- Datenformate
- Directory Management
- 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
Herausforderung Datenqualität (11/2019)
Digitale Kompetenzen – dringend gesucht! (07/2019)
Aktuelle Empfehlungen zu Datenschutz und Forschungsdaten (03/2017)

Metadaten

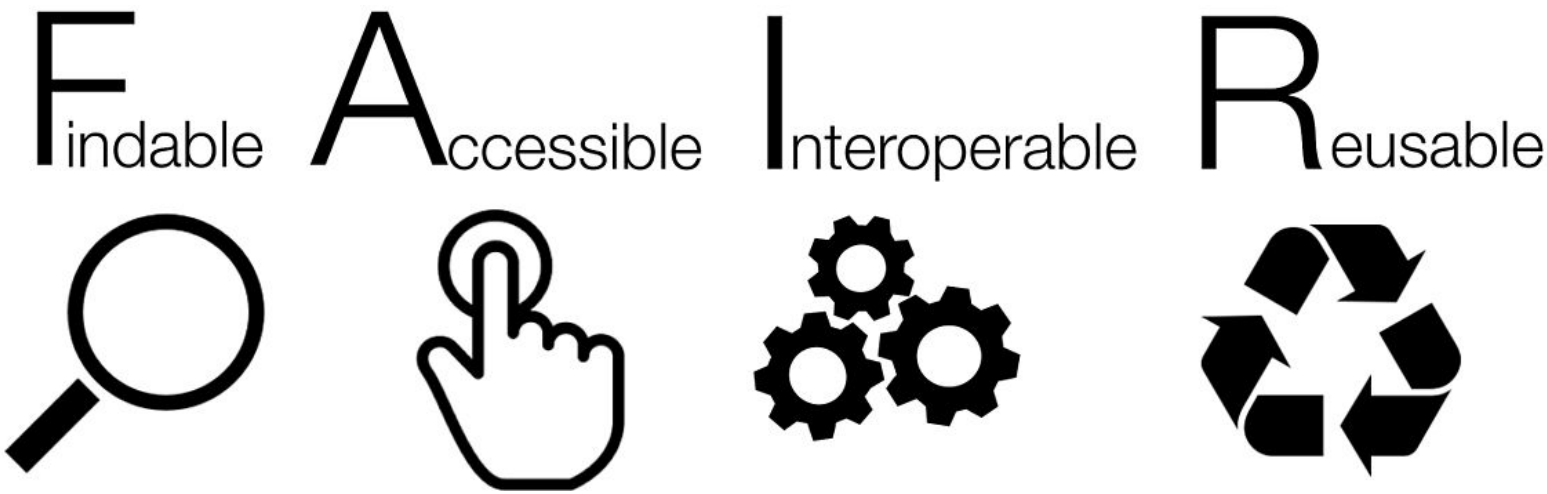
Metadaten repräsentieren Information über Daten

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 Vol 3, 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 Hauptaufgabe nach dem Ende Merkel Ära.
- Nicht alle Wissenschaftler:innen wollen ihre Daten organisieren und teilen.
- Open Science bleibt eine wichtige Initiative verantwortungsvoller Forscher:innen.

Datenmanagement

- FAIR Prinzipien
- **Datenformate**
- Directory Management
- 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.

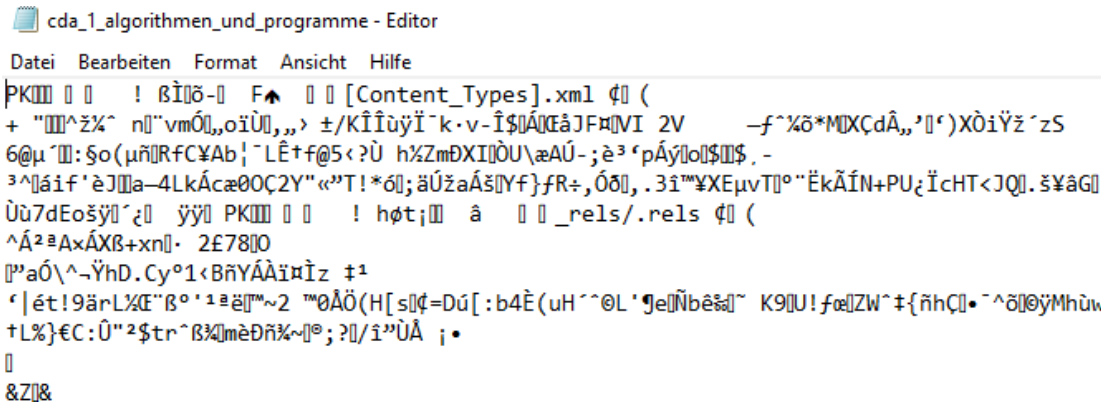
Binäre Dateiformate

- 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.

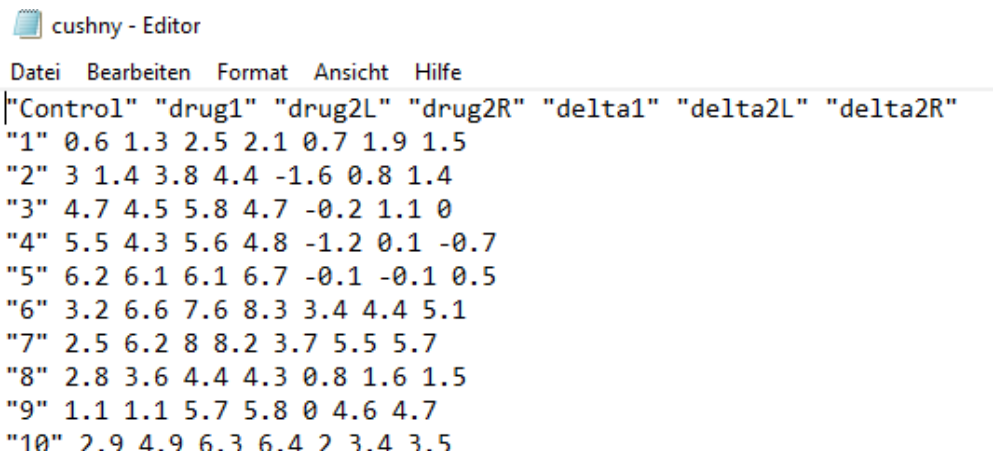
Textuelle Dateiformate

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

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 }
    }
}
```

Datenmanagement

- FAIR Prinzipien
- Datenformate
- **Directory Management**
- 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

```
s = c("Dies ist ein character vector")        # Anfuehrungszeichen sind ...
[1] "Dies ist ein character vector"           # der String Standard
s = c('Dies ist ein "string"')                # Hochkommata koennen bei ...
[1] "Dies ist ein \"string\""                 # Anfuehrungszeichen im String helfen
```

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

```
s = paste(1,2)                                # Konvertierung und Konkatenation
[1] "1 2"                                     # ...einelementiger double vectors
s = paste("Dies ist", "ein String").          # Konkatenation einelementiger
[1] "Dies ist ein String"                     # ...  character strings
```

Arbeiten mit Strings

- **paste()** hat eine Reihe von weiteren Funktionalitäten

```
s = paste(c("Rote", "Gelbe"), "Blume")          # vector recylcing und
[1] "Rote Blume" "Gelbe Blume"                  # elementweise Veknuepfungen
s = paste(c("Rote", "Gelbe"), "Blume",          # Separatorspezifkation
          sep = "-")
[1] "Rote-Blume" "Gelbe-Blume"
s = paste(c("Rote", "Gelbe"), "Blume",          # Zusammenfuegen mit spezifiziertem
          collapse = ", ")                      # Separator
[1] "Rote Blume, Gelbe Blume"
```

- **toString()** ist eine **paste()** Variation für numerische Vektoren

```
s = toString(1:10)                              # Konversion eines double Vektors
[1] "1, 2, 3, 4, 5, 6, 7, 8, 9, 10"             # in formatierten String
s = toString(1:10, width = 10)                  # mit Moeglichkeit der
[1] "1, 2, ..."                                 # Beschraenkung auf width Zeichen.
```

- **cat()** ist eine low-level paste Alternative mit wenig Funktionalität

```
s = cat(c("Gelbe", "Rote"), "Blume")            # Kein Recyling und
Gelbe Rote Blume                                # insbesondere kein
typeof(s)                                       # resultierender
NULL                                            # character vector
```

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.

```
wd = getwd()
[1] "D:/Google Drive/Lehre/2021/1_Inferenzstatistik_21/Code"
```

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\\2021")
getwd()
[1] "D:/Google Drive/Lehre/2021"
```

Dateipfadspezifikation

file.path() konstruiert Verzeichnis- und Dateipfade.

```
p = file.path("D:", "Google Drive", "Lehre", "2021")
[1] "D:/Google Drive/Lehre/2021"
```

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

```
d = getwd()
[1] "D:/Google Drive/Lehre/2021"
p = dirname(getwd())
[1] "D:/Google Drive/Lehre"
```

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

```
d = getwd()
[1] "D:/Google Drive/Lehre/2021"
p = basename(getwd())
[1] "2021"
```

Datenmanagement

- FAIR Prinzipien
- Datenformate
- Directory Management
- **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.
- Im einfachsten Fall reicht die Angabe des Dateinamens zum Einlesen.
- **read.table()** bietet eine Vielzahl weiterer Spezifikationsmöglichkeiten

```
rootdir = "D:\\...\\...\\"                          # root directory name
codedir = file.path(rootdir, "\\...\\Data_Analysis") # Analysecodeverzeichnis
wdir    = setwd(codedir)                            # working directory
ddir    = file.path(dirname(wdir), "Data")          # data directory
fname   = "cushny.csv"                              # (base) filename
D       = read.table(file.path(ddir, fname))        # Einlesen der Datei
print(D)                                            # Konsolenanzeige
```

	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(ddir, fname), nrow = 4)
  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

D = read.table(file.path(ddir, fname), skip = 6)
  V1  V2  V3  V4  V5  V6  V7  V8
1  6 3.2 6.6 7.6 8.3 3.4 4.4 5.1
2  7 2.5 6.2 8.0 8.2 3.7 5.5 5.7
3  8 2.8 3.6 4.4 4.3 0.8 1.6 1.5
4  9 1.1 1.1 5.7 5.8 0.0 4.6 4.7
5 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

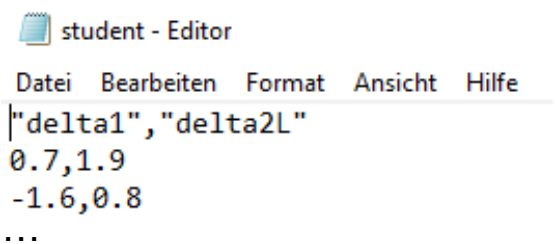
- Twitterdaten können mithilfe der Pakete **rtweet** oder **twitteR** eingelesen werden.
- SQL Datenbanken können mithilfe der Pakete **DBI** und **RSQLite** abgefragt werden.

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(rdir, rname),      # Dateinname
                sep = ",",                          # Werteseparator fuer .csv
                row.names = F)                      # keine Zeilennamen
```

- Ergebnisdatei student.csv



Datenmanagement

- FAIR Prinzipien
- Datenformate
- Directory Management
- 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 Begriff des "Working Directories" in R.
9. Nennen Sie eine R Funktion zum Einlesen von .csv Dateien.
10. Nennen Sie eine R Funktion zum Schreiben von .csv Dateien.

Literatur