

Daten

Datascience mit R lernen

FS 2022

Tidy Daten

- 1 Jede Spalte ist eine Variable.
- 2 Jede Zeile ist eine Beobachtung.
- 3 Jede Zelle beinhaltet nur einen Wert.

...nach [Hadley Wickham](#)

Data frame

M Spalten (Variablen)

	id	geschl	alter	groess
1	1	W	44	174
2	2	M	65	180
3	3	M	31	168
...

N Zeilen (Fälle)

3 Klassen von Datenobjekten

list - R's Mehrzweck-Container

- Kann alle Daten beinhalten, inkl. lists
- Nützlich für komplexe Funktionsoutputs

data_frame - R's Tabelle

- Spezialfall einer list
- R's Tidy-Format für Daten

vector - R's Daten-Container

- Primärer Daten-Container
- Beinhaltet Daten genau einer Klasse

List

The diagram illustrates the three classes of R data objects. On the left, a curly brace groups 'list', 'data frame', and 'vector' under the heading 'List'. An arrow points from this group to a central diagram. The central diagram shows a data frame with columns id, geschl, alter, and groess, and rows 1, 2, 3, and an ellipsis. To its right is a vector with elements 44, 65, 31, and an ellipsis. Labels 'N Zeilen (Fälle)' and 'N Elemente (Fälle)' point to the respective dimensions of the data frame and vector.

Data frame			
M Spalten (Variablen)			
id	geschl	alter	groess
1	W	44	174
2	M	65	180
3	M	31	168
...

0 Spalten

44

65

31

...

data_frame

- 1 Eine liste bestehend aus **vectoren gleicher Länge**.
- 2 Die vectoren haben verschiedene **Klassen**: numeric, character, etc.
- 3 Verschiedene Varianten: `data.frame`, `data.table`, **tibble**.

Data frame

M Spalten (Variablen)

	id	geschl	alter	groess
1	1	W	44	174
2	2	M	65	180
3	3	M	31	168
...

N Zeilen (Fälle)

Inhalt ansehen

```
# printe basel  
basel
```

```
## # A tibble: 10,000 × 20  
##       id geschlecht alter groesse gewicht  
##   <dbl> <chr>     <dbl>    <dbl>    <dbl>  
## 1     1 F           87      165      NA  
## 2     2 M           54      175.     85.6  
## 3     3 F           34      147.     53.9  
## 4     4 M           31      166.     105  
## 5     5 M           24      180.     102.  
## # ... with 9,995 more rows
```

Data frame

M Spalten (Variablen)

	id	geschl	alter	groess
1	1	W	44	174
2	2	M	65	180
3	3	M	31	168
...

N Zeilen (Fälle)

Inhalt ansehen

```
# Zeige den data frame in neuem tab  
View(basel)
```

	id	geschlecht	alter	groesse	gewicht	einkommen
1	1	f	87	165.0	NA	NA
2	2	m	54	174.8	85.6	7500
3	3	f	34	147.4	53.9	5500
4	4	m	31	165.5	105.0	NA
5	5	m	24	180.5	101.6	3800
6	6	m	59	170.5	86.7	NA
7	7	f	48	144.6	74.8	5000
8	8	f	53	176.2	72.4	8100
9	9	m	50	168.3	80.7	7600
10	10	f	62	170.4	52.8	10100
11	11	m	73	167.3	93.8	12200

Data frame

M Spalten (Variablen)

	id	geschl	alter	groess
1	1	w	44	174
2	2	m	65	180
3	3	m	31	168

Zugriff mit \$

```
# Extrahiere die Variable Alter  
basel$alter
```

```
## [1] 87 54 34 31 24 59 48 53 50 62 73 53  
## [13] 38 26  
## [ reached getOption("max.print") -- omitted 9986 €
```

```
# Extrahiere die Variable Bildung  
basel$bildung
```

```
## [1] "obligatorisch" "sek III"  
## [3] "sek III"      "lehre"  
## [5] "obligatorisch" "sek III"  
## [7] "obligatorisch" "lehre"  
## [9] "sek III"      "lehre"  
## [11] "sek III"      "lehre"  
## [13] "lehre"        "obligatorisch"  
## [ reached getOption("max.print") -- omitted 9986 €
```

Data frame

M Spalten (Variablen)

	id	geschl	alter	groess
1	W	44	174	
2	M	65	180	
3	M	31	168	
...

N Zeilen (Fälle)

Verändern mit \$

```
# Teile die Variable einkommen durch 1000  
basel$einkommen <- basel$einkommen / 1000  
  
# zeige data frame  
basel
```

```
## # A tibble: 10,000 × 20  
##       id geschlecht alter groesse gewicht  
##   <dbl> <chr>     <dbl>    <dbl>    <dbl>  
## 1     1 F           87      165      NA  
## 2     2 M           54      175.     85.6  
## 3     3 F           34      147.     53.9  
## 4     4 M           31      166.     105  
## 5     5 M           24      180.     102.  
## # ... with 9,995 more rows
```

Data frame

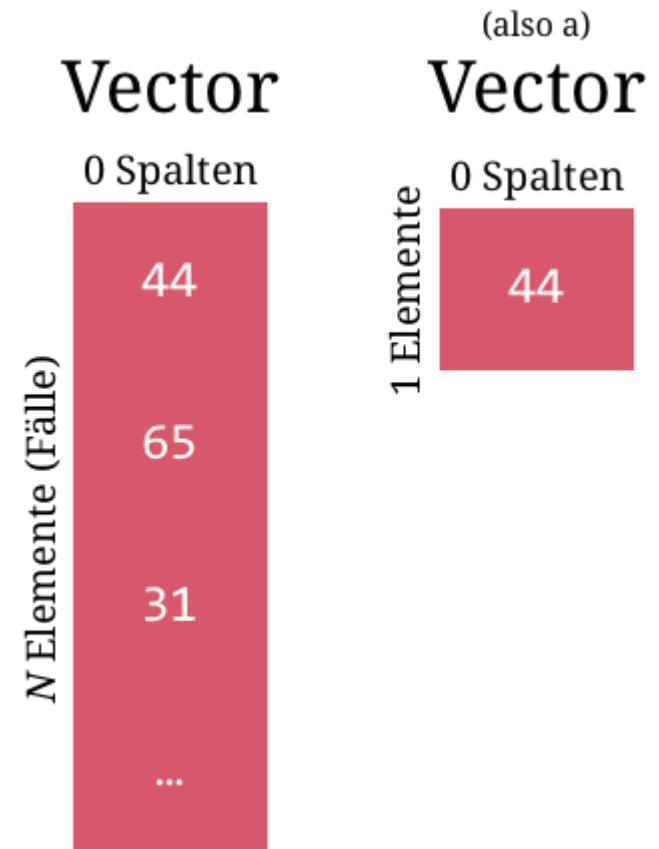
M Spalten (Variablen)

	id	geschl	alter	groess
1	1	W	44	174
2	2	M	65	180
3	3	M	31	168
...

N Zeilen (Fälle)

vector

- 1 R's **primärer Daten-Container**
- 2 Kann nur eine einzelne **Daten-Klasse** beinhalten (und fehlende Werte).
- 3 Daten-Klassen
 - o **numeric** - Alle Zahlen
 - o **character** - Alle Zeichen (e.g., Namen)
 - o **logical** - TRUE oder FALSE
 - o ...
 - o **NA** - fehlende Werte



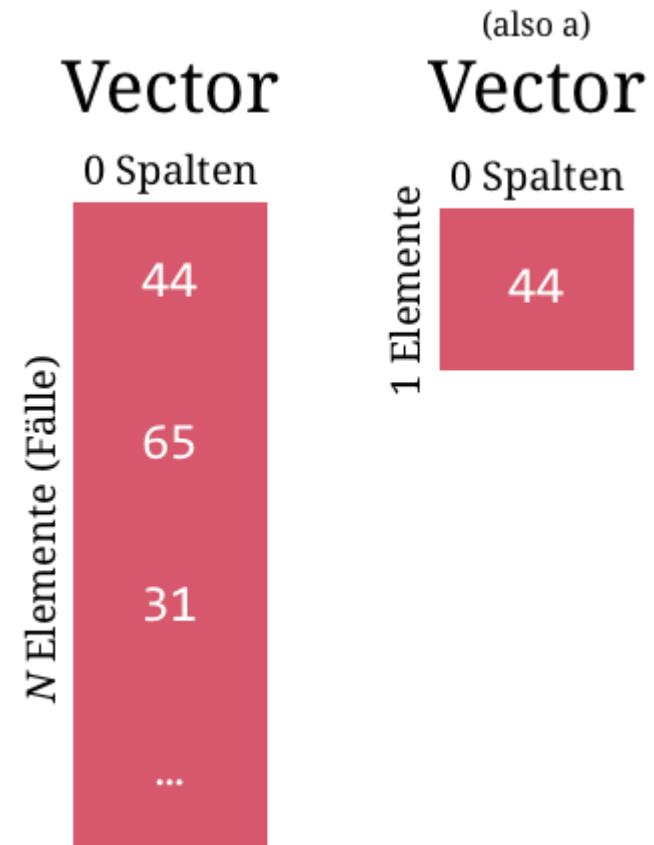
Zugriff / Ändern via []

```
# Extrahiere die Variable Alter  
alter <- basel$alter  
alter  
  
## [1] 87 54 34 31 24 59 48 53 50 62
```

```
# Extrahiere zweiten Wert  
alter[2]
```

```
## [1] 54
```

```
# Ändere zweiten Wert  
alter[2] <- 100  
alter
```



Datentyp: numeric

numeric Vektoren **beinhalten Zahlen** und nur Zahlen.

```
# Extrahiere die Variable Alter  
basel$alter
```



```
## [1] 87 54 34 31 24 59 48 53 50 62
```



```
# Zeige die Klasse von Alter  
class(basel$alter)
```



```
## [1] "numeric"
```



```
# Ist Alter numeric  
is.numeric(basel$alter)
```



```
## [1] TRUE
```

numeric Vector	character Vector	logical Vector
.\$alter	.\$geschl	.\$geschl=="M"
44	"W"	FALSE
65	"M"	TRUE
31	"M"	TRUE
...

Datentyp: character

character Vektoren beinhalten alle Zeichen um die herum **Anführungszeichen** stehen.

```
# Extrahiere die Variable Geschlecht  
basel$geschlecht
```

```
## [1] "F" "M" "F" "M" "M" "M" "F" "F"
```

```
# Extrahiere die Variable Bildung  
basel$bildung
```

```
## [1] "obligatorisch" "sek III"  
## [3] "sek III" "lehre"  
## [5] "obligatorisch" "sek III"  
## [7] "obligatorisch" "lehre"
```

numeric	character	logical
Vector	Vector	Vector
\$.alter	.\$geschl	.\$geschl=="M"
44	"W"	FALSE
65	"M"	TRUE
31	"M"	TRUE
...

Datentyp: character

character Vektoren beinhalten alle Zeichen um die herum **Anführungszeichen** stehen.

```
# Extrahiere die Variable Alter  
basel$alter
```

```
## [1] 87 54 34 31 24 59 48 53 50 62
```

```
# Wandle Alter in character um  
as.character(basel$alter)
```

```
## [1] "87" "54" "34" "31" "24" "59" "48"  
## [8] "53" "50" "62"
```

numeric Vector	character Vector	logical Vector
.\$alter	.\$geschl	.\$geschl=="M"
44	"W"	FALSE
65	"M"	TRUE
31	"M"	TRUE
...

Datentyp: logical

logical Vektoren werden typischerweise durch **logische Vergleiche** erstellt und benutzt um in Data Frames oder Vektoren bestimmte **Fälle auszuwählen**.

```
# Welche Werte in Geschlecht sind M  
basel$geschlecht == "M"
```

```
## [1] FALSE TRUE FALSE TRUE TRUE TRUE  
## [7] FALSE FALSE TRUE FALSE
```

```
# Welche Werte in Alter sind kleiner 30  
basel$alter < 30
```

```
## [1] FALSE FALSE FALSE FALSE TRUE FALSE  
## [7] FALSE FALSE FALSE FALSE
```

numeric	character	logical
Vector	Vector	Vector
.\$alter	.\$geschl	.\$geschl=="M"
44	"W"	FALSE
65	"M"	TRUE
31	"M"	TRUE
...

Datentyp: logical

logical Vektoren werden typischerweise durch **logische Vergleiche** erstellt und benutzt um in Data Frames oder Vektoren bestimmte **Fälle auszuwählen**.

Logische Operatoren

== - ist gleich

<, > - kleiner/grösser als

<=, >= - kleiner/grösser als oder gleich

&, && - logisches UND

|, || - logisches ODER

numeric Vector	character Vector	logical Vector
.\$alter	“W”	FALSE
44	“M”	TRUE
65	“M”	TRUE
31
...

Input/Output

Datentypen ausserhalb von R

1 Strukturierte Daten

- Delimiter getrennt:
.csv, .txt, etc.
- Relationale
Datenbanken: SQL

```
id,geschlecht,alter,groesse,gewicht,einkommen,bildung,  
1,f,87,165,NA,NA,obligatorisch,katholisch,1,7,7,NA,16,  
2,m,54,174.8,85.6,7500,sek III,konfessionslos,2,9,3,95  
3,f,34,147.4,53.9,5500,sek III,konfessionslos,1,3,6,41  
4,m,31,165.5,105,NA,lehre,katholisch,1,8,8,NA,31,0,6,6  
5,m,24,180.5,101.6,3800,obligatorisch,katholisch,2,9,4  
6,m,59,170.5,86.7,NA,sek III,NA,3,8,4,NA,31,0,2,61,3,9  
7,f,48,144.6,74.8,5000,obligatorisch,konfessionslos,2,  
8,f,53,176.2,72.4,8100,lehre,katholisch,1,3,2,1000,29,  
9,m,50,168.3,80.7,7600,sek III,konfessionslos,3,6,5,79  
10,f,62,170.4,52.8,10100,lehre,anderen,1,3,7,1380,0,0,4  
11,m,73,167.3,93.8,12200,sek III,konfessionslos,3,6,6,  
12,f,53,163.1,67.6,6800,lehre,konfessionslos,1,5,4,740  
13,m,38,182.8,70.4,7600,lehre,NA,3,8,5,910,43,0,5,58,2  
14,f,26,160.2,NA,4300,obligatorisch,katholisch,1,4,4,3  
15,f,53,162.7,76.3,7800,obligatorisch,konfessionslos,2  
16,m,19,177,77.4,3800,obligatorisch,evangelisch-reform  
17,m,59,195.5,65.7,10000,sek III,konfessionslos,1,10,3  
18,m,39,178.9,76.9,5500,lehre,katholisch,2,9,4,1140,0,  
19,f,44,173.2,61.5,8100,lehre,konfessionslos,2,7,5,850  
20,f,34,160,66.3,5400,lehre,katholisch,0,9,10,890,28,0  
21,m,73,194.4,68,14200,sek III,konfessionslos,3,3,3,12  
22,f,57,178,66.4,12700,sek III,katholisch,1,7,5,1440,2  
23,f,57,154.9,42.2,9400,sek II,katholisch,3,7,7,1050,1  
24,m,39,182.7,98.9,NA,lehre,konfessionslos,2,3,2,NA,65
```

2 Semi-strukturierte Daten

- Markup: .xml, .xls,
.html etc.
- Non markup: JSON,
MongoDB

```
<!doctype html>  
<html lang="en" class="gr__therbootcamp_github_io">  
  <head>...</head>  
  <body data-gr-c-s-loaded="true">  
    <script async src="https://www.google-analytics.com/analytics">  
    <script type="text/javascript" async src="https://snap.licdn.com/insight.min.js"></script>  
    <script type="text/javascript">  
      linkedin_data_partner_id = "111419";  
    </script>  
    <script type="text/javascript">...</script>  
  <div id="particles-js">  
    <div class="content">  
      <h1>  
        <span class="site-title">TheRBootcamp</span>  
        <span class="site-description">Learn Data Science in R</span>  
      <a class="link" href="#upcoming" data-scroll>  
        <font size="6" color="#FF3A2A">Basel July 21 22 28 29</font>  
      </a>  
    </h1>
```

3 Unstrukturierte Daten

- z.B. Text

R ist eine Interpretersprache, die nicht kompiliert werden muss und Benutzereingaben in der Kommandozeile ausführen. Im Folgenden wird auf die Programmierparadigmen, Syntax und Datentypen eingegangen sowie

Programmierparadigmen

R ist eine Multiparadigmensprache der vierten Generation. Der kanadische Statistiker John M. Chambers.

"To understand computations in R, two slogans are helpful: Everything that exists is an object. Every computation produces an object." „Um Berechnungen in R zu verstehen sind zwei Sätze hilfreich: Alles, was existiert, ist ein Objekt. A – JOHN M. CHAMBERS^[26]

Das funktionale Herz von Scheme und Haskell inspiriert. Funktionen können als First-Class-Objekte ne an andere Funktionen übergeben werden (Closures). Es ist möglich Funktionen zu benennen oder anonyme Rekursion wurde nicht optimiert. Viele Funktionen arbeiten unterschiedlich in Abhängigkeit vom Input (Recursion). Argumente können in Abhängigkeit anderer Argumente definiert werden. Argumente werden per deep copy übergeben und nicht verändert. Wenn die Reihenfolge der Argumente mit denjenigen der Funktion Standardwerte zu setzen. Auch Currying ist möglich. Sofern nicht anders spezifiziert ist das zuletzt zugewiesene Argument die Standardwerte. Für die Funktionsweise von Funktionen ist die Umgebung entscheidend, in der sie übernommen wurde und in der anderen S-Implementierungen nicht existiert. Neu erstellte Objekte befinden sich in einer eigenen Spezialsymbolik. Die Spezialsymbole sind durch geschweifte Klammern gekennzeichnet. R verwendet Lazy Evaluation, das heißt Code wird erst ausgewertet, wenn er benötigt wird. Das bedeutet, dass R nur die benötigten Teile einer Funktion auswertet, sobald sie benötigt werden. Dies ist eine Form der Prozeduralen Programmierung.

R hat zudem Eigenschaften, die für dynamische Programmierung typisch sind. Variablen können flexibel ausgewertet werden. Des Weiteren können die Futures mehrfach ausgewertet werden.

R implementiert die in der vierten Version von S hinzugefügten Klassen und Multimethoden für Ad-hoc-Polytypie. Diese Klassen sind später hinzugefügt (siehe Unterabschnitt Klassen).

Eingelesene Daten speichert R im Hauptspeicher. Die Datenspeicherung erfolgt spaltenorientiert. R nutzt

Delimiter getrennte Daten

- 1 **Delimiter** separieren die Spalten.
- 2 Meist als **lokale Textdatei** vorliegend.
- 3 **Datenklassen** werden inferiert.



```
id,geschlecht,alter,groesse,gewicht,einkommen,bildung,  
1,f,87,165,NA,NA,obligatorisch,katholisch,1,7,7,NA,16,  
2,m,54,174.8,85.6,7500,sek III,konfessionslos,2,9,3,95  
3,f,34,147.4,53.9,5500,sek III,konfessionslos,1,3,6,41  
4,m,31,165.5,105,NA,lehre,katholisch,1,8,8,NA,31,0,6,6  
5,m,24,180.5,101.6,3800,obligatorisch,katholisch,2,9,4  
6,m,59,170.5,86.7,NA,sek III,NA,3,8,4,NA,31,0,2,61,3,9  
7,f,48,144.6,74.8,5000,obligatorisch,konfessionslos,2,  
8,f,53,176.2,72.4,8100,lehre,katholisch,1,3,2,1000,29,  
9,m,50,168.3,80.7,7600,sek III,konfessionslos,3,6,5,79  
10,f,62,170.4,52.8,10100,lehre,anderer,1,3,7,1380,0,0,4  
11,m,73,167.3,93.8,12200,sek III,konfessionslos,3,6,6,  
12,f,53,163.1,67.6,6800,lehre,konfessionslos,1,5,4,740  
13,m,38,182.8,70.4,7600,lehre,NA,3,8,5,910,43,0,5,58,2  
14,f,26,160.2,NA,4300,obligatorisch,katholisch,1,4,4,3  
15,f,53,162.7,76.3,7800,obligatorisch,konfessionslos,2  
16,m,19,177,77.4,3800,obligatorisch,evangelisch-reform  
17,m,59,195.5,65.7,10000,sek III,konfessionslos,1,10,3  
18,m,39,178.9,76.9,5500,lehre,katholisch,2,9,4,1140,0,  
19,f,44,173.2,61.5,8100,lehre,konfessionslos,2,7,5,850  
20,f,34,160,66.3,5400,lehre,katholisch,0,9,10,890,28,0  
21,m,73,194.4,68,14200,sek III,konfessionslos,3,3,3,12  
22,f,57,178,66.4,12700,sek III,katholisch,1,7,5,1440,2  
23,f,57,154.9,42.2,9400,sek II,katholisch,3,7,7,1050,1  
24,m,39,182.7,98.9,NA,lehre,konfessionslos,2,3,2,NA,65
```

Delimiter getrennte Daten

- 1 **Delimiter** separieren die Spalten.
- 2 Meist als **lokale Textdatei** vorliegend.
- 3 **Datenklassen** werden inferiert.

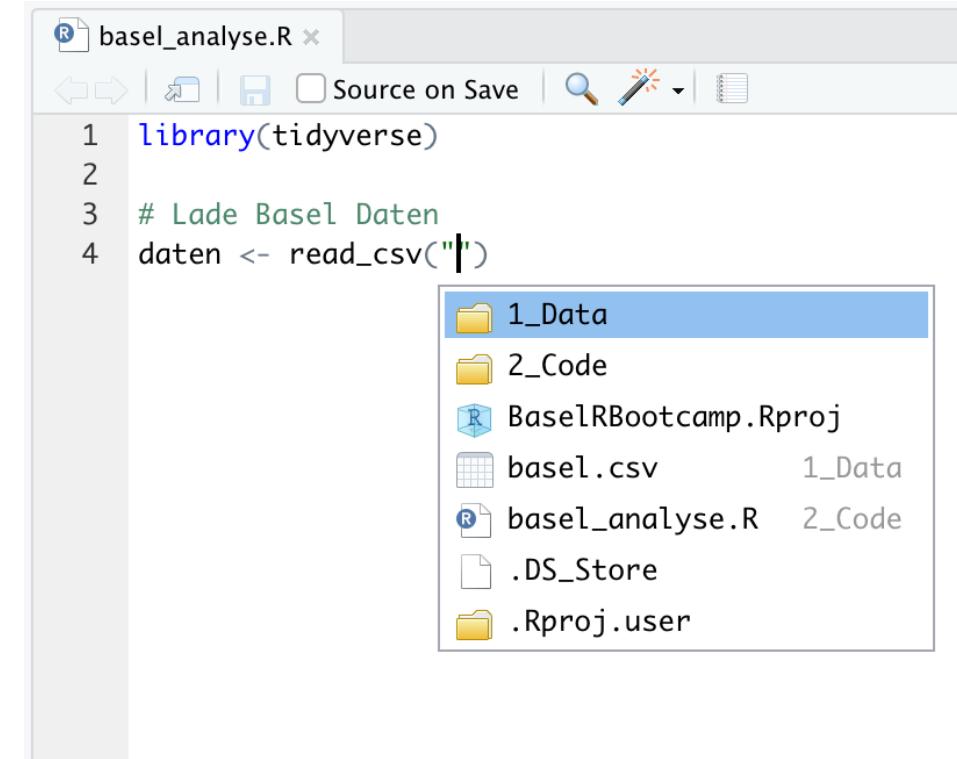


```
# Lese Basel Datensatz ein  
basel <- read_csv("1_Data/basel.csv")  
  
# Benutze expliziten Delimiter  
basel <- read_delim("1_Data/basel.csv",  
                      delim = ",")  
basel
```

```
## # A tibble: 10,000 × 20  
##       id geschlecht alter groesse  
##     <dbl> <chr>      <dbl>   <dbl>  
## 1     1 f             87    165.  
## 2     2 m             54    175.  
## 3     3 f             34    147.  
## 4     4 m             31    166.  
## 5     5 m             24    180.  
## # ... with 9,995 more rows
```

Den Filepath finden

- 1 Finde den Filepath mittels RStudio's **Auto-Complete**.
- 2 Setze den Cursor zwischen Anführungszeichen und drücke **Tab**.



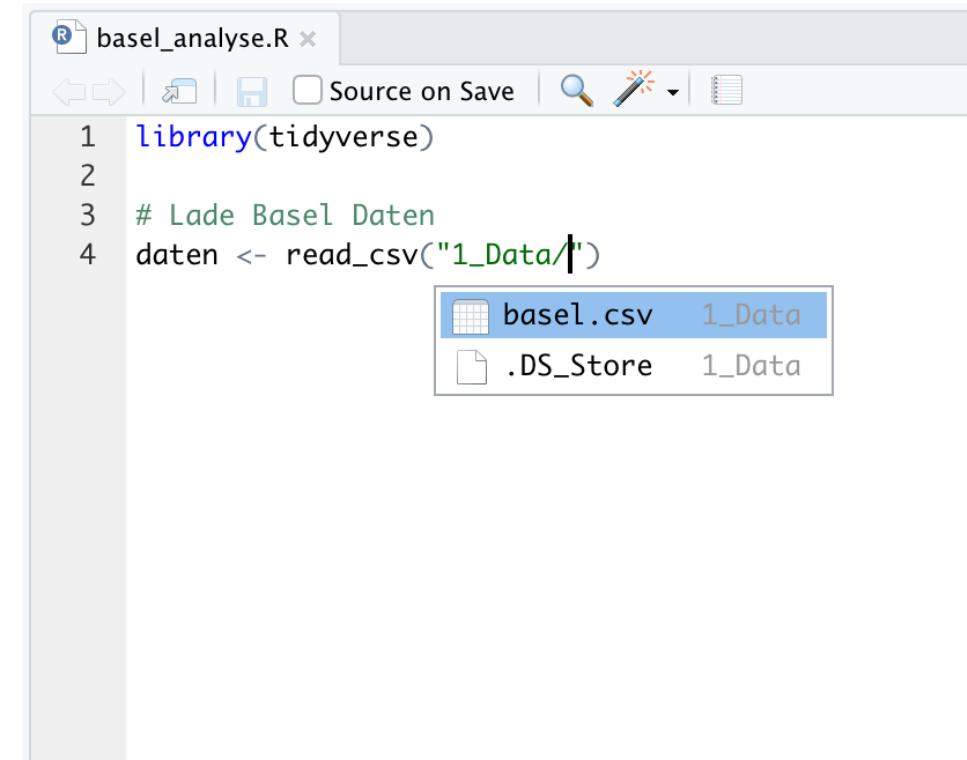
```
library(tidyverse)
# Lade Basel Daten
daten <- read_csv("")


```

1	1_Data
2	2_Code
3	BaselRBootcamp.Rproj
4	basel.csv 1_Data
	basel_analyse.R 2_Code
	.DS_Store
	.Rproj.user

Den Filepath finden

- 1 Finde den Filepath mittels RStudio's **Auto-Complete**.
- 2 Setze den Cursor zwischen Anführungszeichen und drücke **Tab**.



```
library(tidyverse)
# Lade Basel Daten
daten <- read_csv("1_Data/")
```

Inferierte Datentypen

```
# Lese Basel Datensatz ein  
basel <- read_csv("1_Data/basel.csv")  
  
## Rows: 10000 Columns: 20  
## — Column specification ——————  
## Delimiter: ","  
## chr (5): geschlecht, bildung, konfe...  
## dbl (15): id, alter, groesse, gewich...  
##  
## i Use `spec()` to retrieve the full column specification  
## i Specify the column types or set `show_col_types` = TRUE
```

```
id,geschlecht,alter,groesse,gewicht,einkommen,bildung,  
1,f,87,165,NA,NA,obligatorisch,katholisch,1,7,7,NA,16,  
2,m,54,174.8,85.6,7500,sek III,konfessionslos,2,9,3,95  
3,f,34,147.4,53.9,5500,sek III,konfessionslos,1,3,6,41  
4,m,31,165.5,105,NA,lehre,katholisch,1,8,8,NA,31,0,6,6  
5,m,24,180.5,101.6,3800,obligatorisch,katholisch,2,9,4  
6,m,59,170.5,86.7,NA,sek III,NA,3,8,4,NA,31,0,2,61,3,9  
7,f,48,144.6,74.8,5000,obligatorisch,konfessionslos,2,  
8,f,53,176.2,72.4,8100,lehre,katholisch,1,3,2,1000,29,  
9,m,50,168.3,80.7,7600,sek III,konfessionslos,3,6,5,79  
10,f,62,170.4,52.8,10100,lehre,anderer,1,3,7,1380,0,0,4  
11,m,73,167.3,93.8,12200,sek III,konfessionslos,3,6,6,  
12,f,53,163.1,67.6,6800,lehre,konfessionslos,1,5,4,740  
13,m,38,182.8,70.4,7600,lehre,NA,3,8,5,910,43,0,5,58,2  
14,f,26,160.2,NA,4300,obligatorisch,katholisch,1,4,4,3  
15,f,53,162.7,76.3,7800,obligatorisch,konfessionslos,2  
16,m,19,177,77.4,3800,obligatorisch,evangelisch-reform  
17,m,59,195.5,65.7,10000,sek III,konfessionslos,1,10,3  
18,m,39,178.9,76.9,5500,lehre,katholisch,2,9,4,1140,0,  
19,f,44,173.2,61.5,8100,lehre,konfessionslos,2,7,5,850  
20,f,34,160,66.3,5400,lehre,katholisch,0,9,10,890,28,0  
21,m,73,194.4,68,14200,sek III,konfessionslos,3,3,3,12  
22,f,57,178,66.4,12700,sek III,katholisch,1,7,5,1440,2  
23,f,57,154.9,42.2,9400,sek II,katholisch,3,7,7,1050,1  
24,m,39,182.7,98.9,NA,lehre,konfessionslos,2,3,2,NA,65
```

Inferierte Datentypen

Manchmal ist es nötig `readr` etwas auf die Sprünge zu helfen, damit **Datentypen korrekt inferiert** werden.

```
# Setze Symbol für fehlende Werte
basel <- read_csv("1_Data/basel.csv",
                   na = c('NA'))

# Re-inferiere Datentypen
basel <- type_convert(basel)
```

```
id,geschlecht,alter,groesse,gewicht,einkommen,bildung,
1,f,87,165,NA,NA,obligatorisch,katholisch,1,7,7,NA,16,
2,m,54,174.8,85.6,7500,sek III,konfessionslos,2,9,3,95
3,f,34,147.4,53.9,5500,sek III,konfessionslos,1,3,6,41
4,m,31,165.5,105,NA,lehre,katholisch,1,8,8,NA,31,0,6,6
5,m,24,180.5,101.6,3800,obligatorisch,katholisch,2,9,4
6,m,59,170.5,86.7,NA,sek III,NA,3,8,4,NA,31,0,2,61,3,9
7,f,48,144.6,74.8,5000,obligatorisch,konfessionslos,2,
8,f,53,176.2,72.4,8100,lehre,katholisch,1,3,2,1000,29,
9,m,50,168.3,80.7,7600,sek III,konfessionslos,3,6,5,79
10,f,62,170.4,52.8,10100,lehre,andere,1,3,7,1380,0,0,4
11,m,73,167.3,93.8,12200,sek III,konfessionslos,3,6,6,
12,f,53,163.1,67.6,6800,lehre,konfessionslos,1,5,4,740
13,m,38,182.8,70.4,7600,lehre,NA,3,8,5,910,43,0,5,58,2
14,f,26,160.2,NA,4300,obligatorisch,katholisch,1,4,4,3
15,f,53,162.7,76.3,7800,obligatorisch,konfessionslos,2
16,m,19,177,77.4,3800,obligatorisch,evangelisch-reform
17,m,59,195.5,65.7,10000,sek III,konfessionslos,1,10,3
18,m,39,178.9,76.9,5500,lehre,katholisch,2,9,4,1140,0,
19,f,44,173.2,61.5,8100,lehre,konfessionslos,2,7,5,850
20,f,34,160,66.3,5400,lehre,katholisch,0,9,10,890,28,0
21,m,73,194.4,68,14200,sek III,konfessionslos,3,3,3,12
22,f,57,178,66.4,12700,sek III,katholisch,1,7,5,1440,2
23,f,57,154.9,42.2,9400,sek II,katholisch,3,7,7,1050,1
24,m,39,182.7,98.9,NA,lehre,konfessionslos,2,3,2,NA,65
```

Relationale Datenbanken

siehe db.rstudio.com

- 1 R kann direkt **auf Datenbanken arbeiten**:
MySQL, MariaDB, BigQuery, Redshift, etc.
Siehe [hier](#).
- 2 dplyr wird z.B. **automatisch übersetzt**.

```
# Verbinde mit MySQL Datenbank
con <- dbConnect(MySQL(),
  user='studiech_rbootca',
  password='Du*5hA+7NU:8T',
  dbname='studiech_rbootcamp',
  host='studie.ch',
  port = 3306)

# Zeige Tabellen
dbListTables(con)

## [1] "basel"
```

Relationale Datenbanken

siehe db.rstudio.com

- 1 R kann direkt **auf Datenbanken arbeiten**:
MySQL, MariaDB, BigQuery, Redshift, etc.
Siehe [hier](#).
- 2 dplyr wird z.B. **automatisch übersetzt**.

```
# Extrahiere Tabelle Customers  
basel <- tbl(con, "basel")  
basel
```

```
## # Source:   table<basel> [?? x 20]  
## # Database: mysql 5.7.26-log-cll-lve  
## #   [at studie.ch:/studiech_rbootcamp]  
## #       id geschlecht alter_jahre groesse  
## #   <dbl> <chr>           <dbl>    <dbl>  
## # 1     1 f                 87     165  
## # 2     2 m                 54     175.  
## # 3     3 f                 34     147.  
## # 4     4 m                 31     166.  
## # 5     5 m                 24     180.  
## # ... with more rows
```

Relationale Datenbanken

siehe db.rstudio.com

- 1 R kann direkt **auf Datenbanken arbeiten**:
MySQL, MariaDB, BigQuery, Redshift, etc.
Siehe [hier](#).
- 2 dplyr wird z.B. **automatisch übersetzt**.

```
# Extrahiere Tabelle Customers  
basel <- tbl(con, "basel")  
  
# Extrahiere CompanyName Variable  
basel %>% pull(konfession)
```

```
## [1] "katholisch"      "konfessionslos"  
## [3] "konfessionslos" "katholisch"  
## [5] "katholisch"      NA  
## [7] "konfessionslos" "katholisch"  
## [9] "konfessionslos" "andere"  
## [ reached getOption("max.print") -- omitted 999
```

Semi-strukturierte Daten

mit `xml2` und `rvest`

```
# Tabelle laden von Wikipedia (mit xml2 und rvest)
read_html("https://en.wikipedia.org/wiki/R_(programming_language)") %>%
  html_node(xpath = '//*[@id="mw-content-text"]/div/table[2]') %>%
  html_table() %>% as_tibble()

## # A tibble: 22 × 3
##   Release Date      Description
##   <chr>   <chr>      <chr>
## 1 0.16    ""          "This is the last alpha version developed primarily by Ihaka and ...
## 2 0.49    "1997-04-23" "This is the oldest source release which is currently available o...
## 3 0.60    "1997-12-05" "R becomes an official part of the GNU Project. The code is hoste...
## 4 0.65.1   "1999-10-07" "First versions of update.packages and install.packages functions...
## 5 1.0     "2000-02-29" "Considered by its developers stable enough for production use.[5...
## 6 1.4     "2001-12-19" "S4 methods are introduced and the first version for Mac OS X is ...
## 7 1.8     "2003-10-08" "Introduced a flexible condition handling mechanism for signallin...
## 8 2.0     "2004-10-04" "Introduced lazy loading, which enables fast loading of data with...
## 9 2.1     "2005-04-18" "Support for UTF-8 encoding, and the beginnings of internationali...
## 10 2.6.2   "2008-02-08" "Last version to support Windows 95, 98, Me and NT 4.0[60]"
## # ... with 12 more rows
```

Andere Datentypen

siehe [rio](#)

readr



```
# read fixed width files (can be fast)
data <- read_fwf(file, ...)

# read Apache style log files
data <- read_log(file, ...)
```

haven



```
# read SAS's .sas7bat and sas7bcat files
data <- read_sas(file, ...)

# read SPSS's .sav files
data <- read_sav(file, ...)

# etc
```

readxl



```
# read Excel's .xls and xlsx files
data <- read_excel(file, ...)
```

Other

```
# Read Matlab .mat files
data <- R.matlab:::readMat(file, ...)

# Read and wrangle .xml and .html
data <- XML:::xmlParseParse(file, ...)

# from package jsonlite: read .json files
data <- jsonlite:::read_json(file, ...)
```

Agenda