Daten importieren, exportieren, zusammenfügen und pivotieren

Unit 4

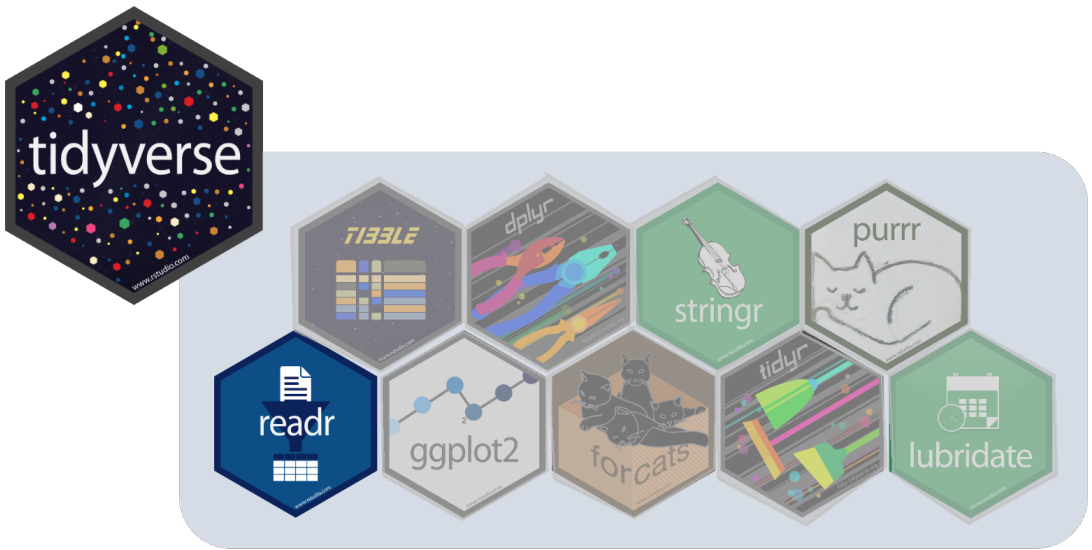
BASEL LANDSCHAFT
FINANZ- UND KIRCHENDIREKTION
AMT FÜR DATEN UND STATISTIK

Ziele für heute

1. grundlegenden Befehle für den Datenimport und -export benennen
2. Befehle zur Zusammenfügung von Daten identifizieren
3. Grundprinzipien des *tidy data*-Konzepts nennen
4. `{tidyr}`-Befehle um Daten zu pivotieren auflisten

Daten importieren und exportieren

Rechteckige Daten importieren




```
1 install.packages("readxl")
2 library(readxl)
```

Rechteckige Daten importieren

`readr`

Funktion	Trennung
`read_csv()`	,
`read_csv2()`	;
`read_tsv()`	⇥ (Tab)
`read_delim()`	Selbst-definiert

`readxl`

Funktion	Dateityp
`read_excel()`	xls oder xlsx

Rechteckige Daten exportieren


```r
install.packages("writexl")
library(writexl)
```

Rechteckige Daten exportieren

readr

- write_csv()
- write_csv2()
- write_tsv()
- write_delim()

writexl

- write_xlsx()

Daten lesen

```r
df <- read_delim("data/ogd_10130.csv", delim = ";")
glimpse(df)
```

```
Rows: 1,487
Columns: 13
$ date                <date> 2024-11-01, 2024-10-01, 2024-09-01, 2024-08-01, 20…
$ `station/location` <chr> "BAS", "BAS", "BAS", "BAS", "BAS", "BAS", "BAS", "B…
$ station_name        <chr> "Basel / Binningen", "Basel / Binningen", "Basel / …
$ gre000m0            <dbl> 54, 80, 139, 247, 247, 216, 198, 161, 120, 69, 44, …
$ hto000m0            <dbl> 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, …
$ nto000m0            <dbl> 72, 76, 69, 48, 59, 76, 76, 80, 81, 83, 79, 80, 82,…
$ prestam0            <dbl> 984.7, 980.0, 978.5, 979.3, 979.0, 977.6, 976.4, 97…
$ rre150m0            <dbl> 53.5, 81.4, 81.2, 25.8, 62.2, 76.4, 166.7, 39.3, 78…
$ sre000m0            <dbl> 5349, 4917, 7705, 17098, 13519, 9151, 9188, 7279, 6…
$ tre200m0            <dbl> 6.2, 12.9, 15.6, 22.2, 21.0, 18.7, 14.9, 11.1, 9.3,…
$ tre200mn            <dbl> -4.4, 4.8, 3.8, 10.3, 12.6, 8.3, 6.3, 0.4, 1.4, 0.3…
$ tre200mx            <dbl> 16.1, 21.8, 32.1, 35.4, 34.5, 32.2, 27.1, 28.8, 20.…
$ ure200m0            <dbl> 84.3, 85.8, 78.2, 68.7, 70.5, 73.4, 75.0, 68.6, 74.…
```

Daten schreiben

```
fussball_weltmeister
```

```
# A tibble: 10 × 3
    jahr weltmeisterschaft titeltraeger
   <int> <chr>             <chr>
 1  2023 Frauen            Spanien
 2  2022 Männer            Argentinien
 3  2019 Frauen            USA
 4  2018 Männer            Frankreich
 5  2015 Frauen            USA
 6  2014 Männer            Deutschland
 7  2011 Frauen            Japan
 8  2010 Männer            Spanien
 9  2007 Frauen            Deutschland
10  2006 Männer            Italien
```

```
write_csv(x = fussball_weltmeister, file = "data/fussball_weltmeister.csv")
```

Daten wieder einlesen

```
read_csv("data/fussball_weltmeister.csv")
```

```
# A tibble: 10 × 3
    jahr weltmeisterschaft titeltraeger
   <dbl> <chr>             <chr>
 1  2023 Frauen            Spanien
 2  2022 Männer            Argentinien
 3  2019 Frauen            USA
 4  2018 Männer            Frankreich
 5  2015 Frauen            USA
 6  2014 Männer            Deutschland
 7  2011 Frauen            Japan
 8  2010 Männer            Spanien
 9  2007 Frauen            Deutschland
10  2006 Männer            Italien
```

rio

import(), export(), convert()

```
1 install.packages("rio")
2 library(rio)
```

```
1 x <- import("mtcars.csv")
2 y <- import("mtcars.rds") # R data format
3 z <- import("mtcars.sav") # SPSS
4 u <- import("mtcars.xlsx")
5 w <- import("mtcars.json")
```

```
1 export("mtcars.csv")
2 export(list(mtcars, penguins), "mtcars-penguins.xlsx") # multiple sheets
3 import("mtcars-penguins.xlsx", which = "penguins") # select one sheet
4 import_list("mtcars-penguins.xlsx") # import multiple objects
```

Andere Formate

`readRDS()` und `writeRDS()`

- Zwischenergebnisse als `CSV` zu speichern unzuverlässig, wenn bestimmte Variablentypen beibehalten werden sollen
- `read_csv()` kann nicht wissen welche Levels eine Faktor-Variable hat
- Alternative: `RDS`-Dateien, ein R-internes Dateiformat

Variablen-Namen

```
1 wetter <- read_delim("data/ogd_12030.csv")
2 names(wetter)
```

```
 1  [1] "Datum"                          "Jahr"
 2  [3] "Globalstrahlung in W/m2"        "Gesamtschneemenge"
 3  [5] "Gesamtbewölkung"                "Luftdruck in hPa"
 4  [7] "Niederschlag"                   "Sonnenscheindauer"
 5  [9] "Tagesmittel Lufttemperatur"     "Tagesminimum Lufttemperatur"
 6 [11] "Tagesmaximum Lufttemperatur"    "Relative Luftfeuchtigkeit"
```

```
1 wetter |>
2   filter(Globalstrahlung in W/m2 > 111)
```

```
Error in parse(text = input): <text>:2:26: unexpected 'in'
1: wetter |>
2:   filter(Globalstrahlung in
                             ^
```

Variablen-Namen - Backticks `

```r
wetter |>
  filter(`Globalstrahlung in W/m2` > 111)
```

```
# A tibble: 8,013 × 12
   Datum       Jahr `Globalstrahlung in W/m2` Gesamtschneemenge Gesamtbewölkung
   <date>     <dbl>                     <dbl>               <dbl>            <dbl>
 1 2001-02-15  2001                       113                   0               17
 2 2001-02-19  2001                       117                   0               42
 3 2001-02-26  2001                       113                   0               13
 4 2001-02-27  2001                       118                   0               88
 5 2001-03-24  2001                       122                   0               88
 6 2001-03-25  2001                       112                   0               92
 7 2001-04-02  2001                       218                   0               83
 8 2001-04-05  2001                       188                   0               50
 9 2001-04-13  2001                       203                   0               63
10 2001-04-14  2001                       160                   0               79
# ℹ 8,003 more rows
# ℹ 7 more variables: `Luftdruck in hPa` <dbl>, Niederschlag <dbl>,
#   Sonnenscheindauer <dbl>, `Tagesmittel Lufttemperatur` <dbl>,
#   `Tagesminimum Lufttemperatur` <dbl>, `Tagesmaximum Lufttemperatur` <dbl>,
#   `Relative Luftfeuchtigkeit` <dbl>
```

Mühsam

Variablen-Namen - {readr}-Funktion

```r
wetter<- read_delim(
  "data/ogd_12030.csv",
  col_names = c(
    "datum",
    "jahr",
    "globalstrahlung_in_w_m2",
    "gesamtschneemenge",
    "gesamtbewolkung",
    "luftdruck_in_h_pa",
    "niederschlag",
    "sonnenscheindauer",
    "tagesmittel_lufttemperatur",
    "tagesminimum_lufttemperatur",
    "tagesmaximum_lufttemperatur",
    "relative_luftfeuchtigkeit"
  )
)
names(wetter)
```

```
 [1] "datum"                          "ja
 [3] "globalstrahlung_in_w_m2"        "ge
 [5] "gesamtbewolkung"                "lu
 [7] "niederschlag"                   "so
 [9] "tagesmittel_lufttemperatur"     "ta
[11] "tagesmaximum_lufttemperatur"    "re
```

Auch mühsam

Variablen-Namen - {janitor}


```r
# install.packages("janitor")
library(janitor)

wetter <- read_delim("data/ogd_12030.csv")

wetter |>
  clean_names() |>
  names()
```

```
 [1] "datum"                          "jahr"
 [3] "globalstrahlung_in_w_m2"        "gesamtschneemenge"
 [5] "gesamtbewolkung"                "luftdruck_in_h_pa"
 [7] "niederschlag"                   "sonnenscheindauer"
 [9] "tagesmittel_lufttemperatur"     "tagesminimum_lufttemperatur"
[11] "tagesmaximum_lufttemperatur"    "relative_luftfeuchtigkeit"
```

Praktikum 04a: Daten importieren und exportieren

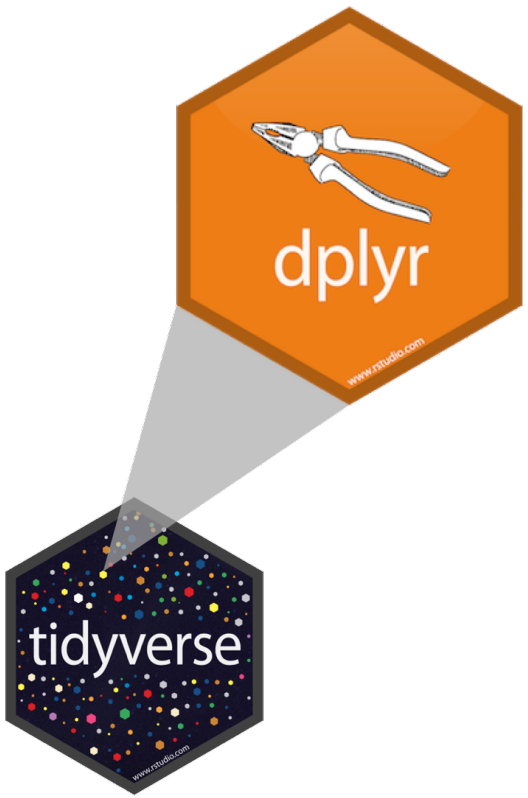
prak-04a-import-export.qmd

30:00

Break ☕ 🍵 🍜

10:00

Daten-Transformation mit `dplyr`



Zeilen: auswählen, anordnen

Spalten: aswählen, anordnen, umbenennen, erstellen

Gruppen: zusammenfassen, zählen

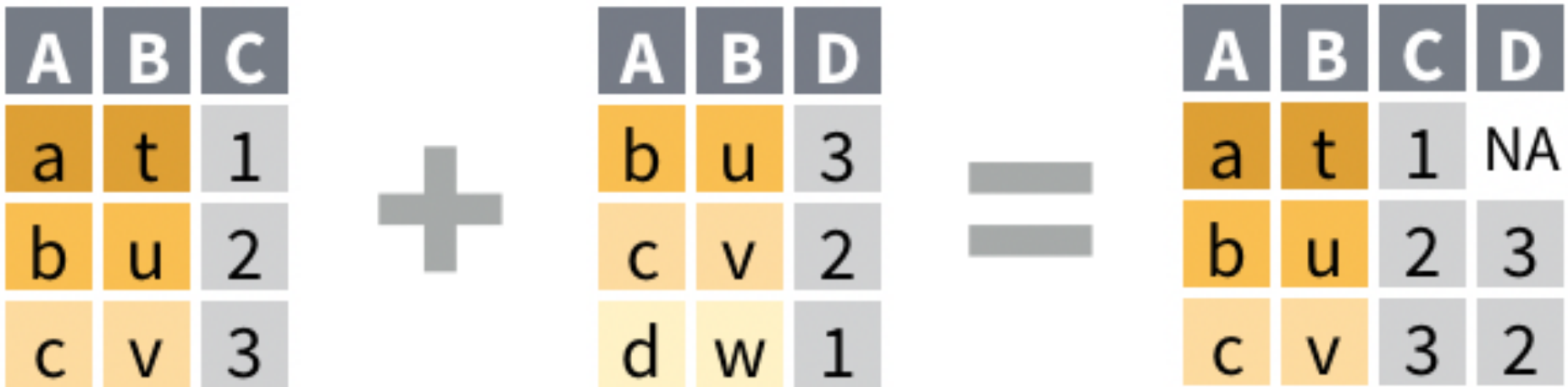
Tabellen: zusammenfügen

Daten zusammenfügen mit `dplyr`

Wir…

haben mehrere Dataframes

wollen diese zusammenbringen



A	B	C
a	t	1
b	u	2
c	v	3

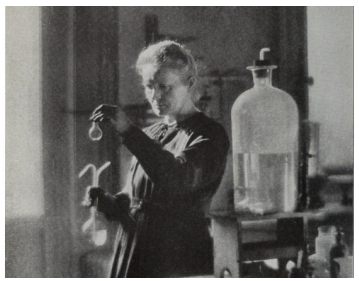
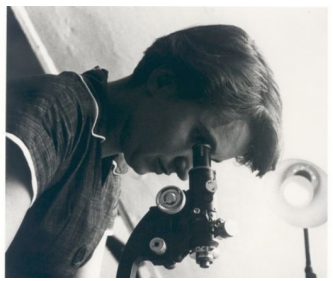
+

A	B	D
b	u	3
c	v	2
d	w	1

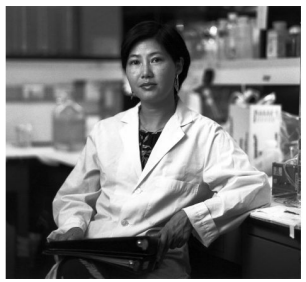
=

A	B	C	D
a	t	1	NA
b	u	2	3
c	v	3	2

Daten: Frauen in der Wissenschaft



Inputs: drei Dataframes

professions | dates | works

name	profession
Ada Lovelace	Mathematician
Marie Curie	Physicist and Chemist
Janaki Ammal	Botanist
Chien-Shiung Wu	Physicist
Katherine Johnson	Mathematician
Rosalind Franklin	Chemist
Vera Rubin	Astronomer
Gladys West	Mathematician
Flossie Wong-Staal	Virologist and Molecular Biologist
Jennifer Doudna	Biochemist

Gewünschter Output

name	profession	birth_year	death_year	known_for
Ada Lovelace	Mathematician	NA	NA	first computer algorithm
Marie Curie	Physicist and Chemist	NA	NA	theory of radioactivity, discovery of elements polonium and radium, first woman to win a Nobel Prize
Janaki Ammal	Botanist	1897	1984	hybrid species, biodiversity protection
Chien-Shiung Wu	Physicist	1912	1997	confim and refine theory of radioactive beta decy, Wu experiment overturning theory of parity
Katherine Johnson	Mathematician	1918	2020	calculations of orbital mechanics critical to sending the first Americans into space
Rosalind Franklin	Chemist	1920	1958	NA
Vera Rubin	Astronomer	1928	2016	existence of dark matter
Gladys West	Mathematician	1930	NA	mathematical modeling of the shape of the Earth which served as the foundation of GPS technology
Flossie Wong-Staal	Virologist and Molecular Biologist	1947	NA	first scientist to clone HIV and create a map of its genes which led to a test for the virus
Jennifer Doudna	Biochemist	1964	NA	one of the primary developers of CRISPR, a ground-breaking technology for editing genomes

Inputs: drei Dataframes

```
names(professions)
```

```
[1] "name"       "profession"
```

```
nrow(professions)
```

```
[1] 10
```

```
names(dates)
```

```
[1] "name"       "birth_year" "death_year"
```

```
nrow(dates)
```

```
[1] 8
```

```
names(works)
```

```
[1] "name"      "known_for"
```

```
nrow(works)
```

```
[1] 9
```

Dataframes zusammenfügen

`***_join(x, y)`

- `left_join(x, y)`: alle Reihen aus `x`
- `right_join(x, y)`: alle Reihen aus `y`
- `full_join(x, y)`: alle Reihen aus `x` und `y`
- `inner_join(x, y)`: gemeinsame Reihen aus `x` und `y`
- `semi_join(x, y)`: wie `inner_join(x, y)`, nur Spalten aus `x`
- `anti_join(x, y)`: Reihen aus `x` ohne Übereinstimmung in `y`

Beispiel

Für die nächsten Folien

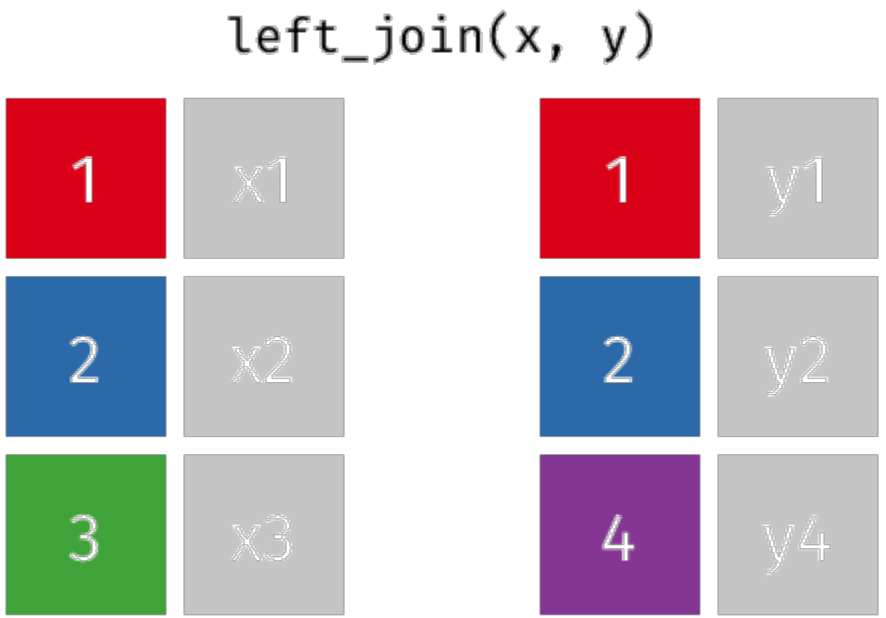
```
1 x
```

```
# A tibble: 3 × 2
     id var_x
  <dbl> <chr>
1     1 x1
2     2 x2
3     3 x3
```

```
1 y
```

```
# A tibble: 3 × 2
     id var_y
  <dbl> <chr>
1     1 y1
2     2 y2
3     4 y4
```

left_join()




```
1 left_join(x, y)
```

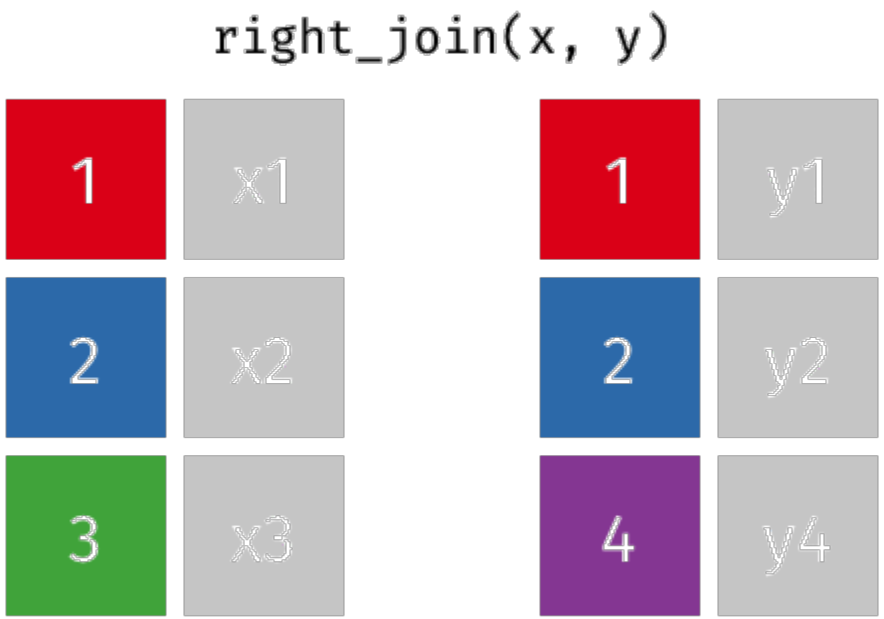
```
1 # A tibble: 3 × 3
2      id var_x var_y
3   <dbl> <chr> <chr>
4 1     1 x1    y1
5 2     2 x2    y2
6 3     3 x3    <NA>
```

left_join()

```
professions |>
  left_join(dates)
```

```
# A tibble: 10 × 4
   name              profession                      birth_year death_year
   <chr>             <chr>                                <dbl>      <dbl>
 1 Ada Lovelace      Mathematician                           NA         NA
 2 Marie Curie       Physicist and Chemist                   NA         NA
 3 Janaki Ammal      Botanist                              1897       1984
 4 Chien-Shiung Wu   Physicist                             1912       1997
 5 Katherine Johnson Mathematician                         1918       2020
 6 Rosalind Franklin Chemist                               1920       1958
 7 Vera Rubin        Astronomer                            1928       2016
 8 Gladys West       Mathematician                         1930         NA
 9 Flossie Wong-Staal Virologist and Molecular Biologist   1947         NA
10 Jennifer Doudna   Biochemist                            1964         NA
```

right_join()




```
1  right_join(x, y)
```

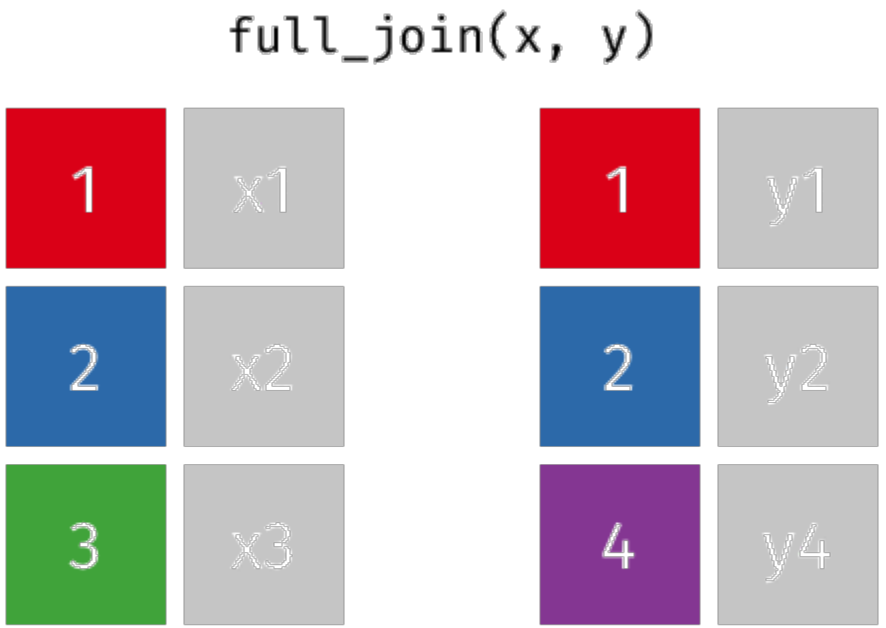
```
1  # A tibble: 3 × 3
2       id var_x var_y
3    <dbl> <chr> <chr>
4  1     1 x1    y1
5  2     2 x2    y2
6  3     4 <NA>  y4
```

right_join

```
professions |>
  right_join(dates)
```

```
# A tibble: 8 × 4
  name              profession                      birth_year death_year
  <chr>             <chr>                                <dbl>      <dbl>
1 Janaki Ammal      Botanist                              1897       1984
2 Chien-Shiung Wu   Physicist                             1912       1997
3 Katherine Johnson Mathematician                         1918       2020
4 Rosalind Franklin Chemist                               1920       1958
5 Vera Rubin        Astronomer                            1928       2016
6 Gladys West       Mathematician                         1930         NA
7 Flossie Wong-Staal Virologist and Molecular Biologist   1947         NA
8 Jennifer Doudna   Biochemist                            1964         NA
```

full_join()




```
1 full_join(x, y)
```

```
1 # A tibble: 4 × 3
2      id var_x var_y
3   <dbl> <chr> <chr>
4 1     1 x1    y1
5 2     2 x2    y2
6 3     3 x3    <NA>
7 4     4 <NA>  y4
```

full_join()

```
dates |>
  full_join(works)
```

```
# A tibble: 10 × 4
   name               birth_year death_year known_for
   <chr>                   <dbl>      <dbl> <chr>
 1 Janaki Ammal             1897       1984 hybrid species, biodiversity protec…
 2 Chien-Shiung Wu          1912       1997 confim and refine theory of radioac…
 3 Katherine Johnson        1918       2020 calculations of orbital mechanics c…
 4 Rosalind Franklin        1920       1958 <NA>
 5 Vera Rubin               1928       2016 existence of dark matter
 6 Gladys West              1930         NA mathematical modeling of the shape …
 7 Flossie Wong-Staal       1947         NA first scientist to clone HIV and cr…
 8 Jennifer Doudna          1964         NA one of the primary developers of CR…
 9 Ada Lovelace               NA         NA first computer algorithm
10 Marie Curie                NA         NA theory of radioactivity,  discovery…
```

Alles in einer Code-Sequenz

```
professions |>
  left_join(dates) |>
  left_join(works)
```

```
# A tibble: 10 × 5
   name              profession                birth_year death_year known_for
   <chr>             <chr>                          <dbl>      <dbl> <chr>
 1 Ada Lovelace      Mathematician                     NA         NA first co…
 2 Marie Curie       Physicist and Chemist             NA         NA theory o…
 3 Janaki Ammal      Botanist                        1897       1984 hybrid s…
 4 Chien-Shiung Wu   Physicist                       1912       1997 confim a…
 5 Katherine Johnson Mathematician                   1918       2020 calculat…
 6 Rosalind Franklin Chemist                         1920       1958 <NA>
 7 Vera Rubin        Astronomer                      1928       2016 existenc…
 8 Gladys West       Mathematician                   1930         NA mathemat…
 9 Flossie Wong-Staal Virologist and Molecular …     1947         NA first sc…
10 Jennifer Doudna   Biochemist                      1964         NA one of t…
```

join_by()

```r
mitarbeiter <- tibble(
  id = c(1, 2, 3),
  name = c("Alice", "Bob", "Charlie")
)
mitarbeiter
```

```
# A tibble: 3 × 2
     id name
  <dbl> <chr>
1     1 Alice
2     2 Bob
3     3 Charlie
```

```r
gehälter <- tibble(
  persid = c(1, 2, 4),
  gehalt = c(50000, 60000, 70000)
)
gehälter
```

```
# A tibble: 3 × 2
  persid gehalt
   <dbl>  <dbl>
1      1  50000
2      2  60000
3      4  70000
```

join_by()

```
ergebnis <- mitarbeiter |>
  left_join(gehälter, join_by(id == persid))
ergebnis
```

```
# A tibble: 3 × 3
     id name    gehalt
  <dbl> <chr>    <dbl>
1     1 Alice    50000
2     2 Bob      60000
3     3 Charlie     NA
```

Praktikum 04b: Daten zusammenfügen

prak-04b-join-firmen.qmd

20:00

Break ☕ 🍵 🍜

10:00

Tidy Data

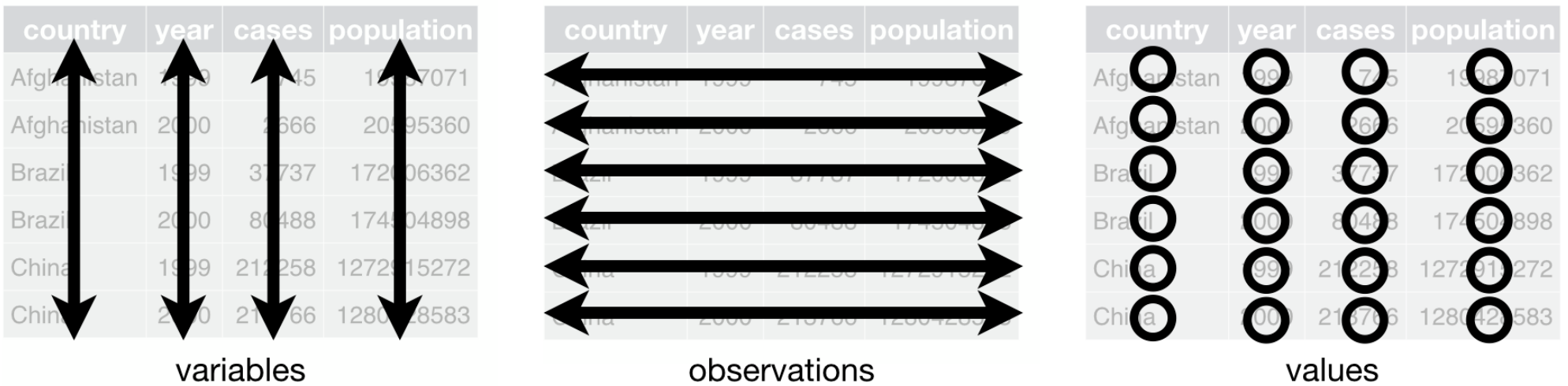
Tidy Data

> “Alle glücklichen Familien gleichen einander, jede unglückliche Familie ist auf ihre eigene Weise unglücklich.” – *Leo Tolstoy*

> “Tidy datasets are all alike, but every messy dataset is messy in its own way.” — *Hadley Wickham*

tidy = ordentlich, sauber, aufgeräumt.

Tidy Data



1. Jede Variable muss eine eigene Spalte haben
2. Jede Beobachtung muss eine eigene Zeile haben
3. Jeder Wert muss eine eigene Zelle haben

species	island	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	sex	year
Adelie	Torgersen	39.1	18.7	181	3750	male	2007
Adelie	Torgersen	39.5	17.4	186	3800	female	2007
Adelie	Torgersen	40.3	18.0	195	3250	female	2007
Adelie	Torgersen	NA	NA	NA	NA	NA	2007
Adelie	Torgersen	36.7	19.3	193	3450	female	2007
Adelie	Torgersen	39.3	20.6	190	3650	male	2007
Adelie	Torgersen	38.9	17.8	181	3625	female	2007
Adelie	Torgersen	39.2	19.6	195	4675	male	2007
Adelie	Torgersen	34.1	18.1	193	3475	NA	2007
Adelie	Torgersen	42.0	20.2	190	4250	NA	2007
Adelie	Torgersen	37.8	17.1	186	3300	NA	2007
Adelie	Torgersen	37.8	17.3	180	3700	NA	2007
Adelie	Torgersen	41.1	17.6	182	3200	female	2007
Adelie	Torgersen	38.6	21.2	191	3800	male	2007

	A	B	C	D	E	F	G	H	I
1	**cc-d-01.02.04.02**	**Ständige Wohnbevölkerung nach Altersklasse und Altersmasszahlen nach Kanton, am 31.12.2020**							
2		Provisorische Jahresergebnisse							
3	Grossregionen Kantone	Total	0-19 Jahre	20-39 Jahre	40-64 Jahre	65-79 Jahre	80 Jahre und mehr	Jugendquotient [1]	Altersquotient [2]
4	**Schweiz**	**8 667 088**	**1 723 565**	**2 280 707**	**3 032 787**	**1 171 506**	**458 523**	**32.4**	**30.7**
5	Genferseeregion	1 668 471	351 278	455 418	574 194	205 266	82 315	34.1	27.9
6	Waadt	814 075	177 447	225 069	276 942	96 173	38 444	35.3	26.8
7	Wallis	348 318	67 628	89 388	121 294	51 839	18 169	32.1	33.2
8	Genf	506 078	106 203	140 961	175 958	57 254	25 702	33.5	26.2
9	Espace Mittelland	1 894 879	373 753	481 384	659 098	275 838	104 806	32.8	33.4
10	Bern	1 042 516	197 760	260 221	362 072	160 402	62 061	31.8	35.7
11	Freiburg	325 419	71 505	88 526	112 785	39 646	12 957	35.5	26.1
12	Solothurn	277 396	52 800	69 973	98 981	40 445	15 197	31.3	32.9
13	Neuenburg	175 860	36 459	44 834	60 437	23 984	10 146	34.6	32.4
14	Jura	73 688	15 229	17 830	24 823	11 361	4 445	35.7	37.1
15	Nordwestschweiz	1 181 397	230 093	302 831	418 419	165 110	64 944	31.9	31.9

Quelle: Bundesamt für Statistik - Ständige Wohnbevölkerung nach Altersklasse und Altersmasszahlen nach Kanton, Provisorische Jahresergebnisse, 2020

	A	B	C	D	E	F	G	H	I
1	**cc-d-01.02.04.02**	**Ständige Wohnbevölkerung**							
2		Provisorische Jahresergebnisse							
3	Grossregionen Kantone	Total	0-19 Jahre	20-39 Jahre	40-64 Jahre	65-79 Jahre	80 Jahre und mehr	Jugendquotient [1]	Altersquotient [2]
4	**Schweiz**	**8 667 088**	**1 723 565**	**2 280 707**	**3 032 787**	**1 171 506**	**458 523**	**32.4**	**30.7**
5	Genferseeregion	1 668 471	351 278	455 418	574 194	205 266	82 315	34.1	27.9
6	Waadt	814 075	177 447	225 069	276 942	96 173	38 444	35.3	26.8
7	Wallis	348 318	67 628	89 388	121 294	51 839	18 169	32.1	33.2
8	Genf	506 078	106 203	140 961	175 958	57 254	25 702	33.5	26.2
9	Espace Mittelland	1 894 879	373 753	481 384	659 098	275 838	104 806	32.8	33.4
10	Bern	1 042 516	197 760	260 221	362 072	160 402	62 061	31.8	35.7
11	Freiburg	325 419	71 505	88 526	112 785	39 646	12 957	35.5	26.1
12	Solothurn	277 396	52 800	69 973	98 981	40 445	15 197	31.3	32.9
13	Neuenburg	175 860	36 459	44 834	60 437	23 984	10 146	34.6	32.4
14	Jura	73 688	15 229	17 830	24 823	11 361	4 445	35.7	37.1
15	Nordwestschweiz	1 181 397	230 093	302 831	418 419	165 110	64 944	31.9	31.9

Variable Altersgruppe als Reihe

Quelle: Bundesamt für Statistik - Ständige Wohnbevölkerung nach Altersklasse und Altersmasszahlen nach Kanton, Provisorische Jahresergebnisse, 2020

	A	B	C	D	E	F	G	H	I
1	**cc-d-01.02.04.02**	**Ständige Wohnbevölkerung**							
2		Provisorische Jahresergebnisse							
3	Grossregionen Kantone	Total	0-19 Jahre	20-39 Jahre	40-64 Jahre	65-79 Jahre	80 Jahre und mehr	Jugendquotient [1]	Altersquotient [2]
4	**Schweiz**	**8 667 088**	**1 723 565**	**2 280 707**	**3 032 787**	**1 171 506**	**458 523**	**32.4**	**30.7**
5	Genferseeregion	1 668 471	351 278	455 418	574 194	205 266	82 315	34.1	27.9
6	Waadt	814 075	177 447	225 069	276 942	96 173	38 444	35.3	26.8
7	Wallis	348 318	67 628	89 388	121 294	51 839	18 169	32.1	33.2
8	Genf	506 078	106 203	140 961	175 958	57 254	25 702	33.5	26.2
9	Espace Mittelland	1 894 879	373 753	481 384	659 098	275 838	104 806	32.8	33.4
10	Bern	1 042 516	197 760	260 221	362 072	160 402	62 061	31.8	35.7
11	Freiburg	325 419	71 505	88 526	112 785	39 646	12 957	35.5	26.1
12	Solothurn	277 396	52 800	69 973	98 981	40 445	15 197	31.3	32.9
13	Neuenburg	175 860	36 459	44 834	60 437	23 984	10 146	34.6	32.4
14	Jura	73 688	15 229	17 830	24 823	11 361	4 445	35.7	37.1
15	Nordwestschweiz	1 181 397	230 093	302 831	418 419	165 110	64 944	31.9	31.9

Variable Altersgruppe als Reihe

Reihen als Zusammenfassung (Summe)

Quelle: Bundesamt für Statistik - Ständige Wohnbevölkerung nach Altersklasse und Altersmasszahlen nach Kanton, Provisorische Jahresergebnisse, 2020

Data extracted on 05/09/2023 10:17:29	from [ESTAT]					
Dataset:	**Fertility indicators [DEMO_FIND__custom_684440]**					
Last updated:	06/07/2023 11:00					
Time frequency		Annual				
Demographic indicator		Total fertility rate				
TIME	**2021**		**2020**		**2019**	
GEO (Labels)						
European Union - 27 countries (fron	1,53	bep	1,50	ep	1,53	bep
Belgium	1,60		1,55		1,60	
Bulgaria	1,58		1,56		1,58	
Czechia	1,83	b	1,71		1,71	
Denmark	1,72		1,68		1,70	
Germany	1,58		1,53		1,54	
Estonia	1,61		1,58		1,66	
Ireland	1,78		1,63		1,71	e
Greece	1,43		1,39		1,34	
Spain	1,19		1,19		1,23	
France	1,84	p	1,83	p	1,86	p
Croatia	1,58	b	1,48		1,47	
Italy	1,25		1,24		1,27	b

Quelle: Eurostat Fertility Indicators

?

Data extracted on 05/09/2023 10:17:29	from [ESTAT]					
Dataset:	**Fertility indicators [DEMO_FIND__custom_684440]**					
Last updated:	06/07/2023 11:00					
Time frequency	Annual					
Demographic indicator	Total fertility rate					
TIME	**2021**		**2020**		**2019**	
GEO (Labels)						
European Union - 27 countries (fron	1,53	bep	1,50	ep	1,53	bep
Belgium	1,60		1,55		1,60	
Bulgaria	1,58		1,56		1,58	
Czechia	1,83	b	1,71		1,71	
Denmark	1,72		1,68		1,70	
Germany	1,58		1,53		1,54	
Estonia	1,61		1,58		1,66	
Ireland	1,78		1,63		1,71	e
Greece	1,43		1,39		1,34	
Spain	1,19		1,19		1,23	
France	1,84	p	1,83	p	1,86	p
Croatia	1,58	b	1,48		1,47	
Italy	1,25		1,24		1,27	b

Variable Jahr als Zeile

Quelle: Eurostat Fertility Indicators

?

Data extracted on 05/09/2023 10:17:29	from [ESTAT]					
Dataset:	**Fertility indicators [DEMO_FIND__custom_684440]**					
Last updated:	06/07/2023 11:00					
Time frequency	Annual					
Demographic indicator	Total fertility rate					
TIME	2021		2020		2019	
GEO (Labels)						
European Union - 27 countries (fron	1,53	bep	1,50	ep	1,53	bep
Belgium	1,60		1,55		1,60	
Bulgaria	1,58		1,56		1,58	
Czechia	1,83	b	1,71		1,71	
Denmark	1,72		1,68		1,70	
Germany	1,58		1,53		1,54	
Estonia	1,61		1,58		1,66	
Ireland	1,78		1,63		1,71	e
Greece	1,43		1,39		1,34	
Spain	1,19		1,19		1,23	
France	1,84	p	1,83	p	1,86	p
Croatia	1,58	b	1,48		1,47	
Italy	1,25		1,24		1,27	b

Variable Jahr als Zeile

Zeile als Zusammenfassung (Durchschnitt)

Quelle: Eurostat Fertility Indicators

?

Data extracted on 05/09/2023 10:17:29	from [ESTAT]					
Dataset:	**Fertility indicators [DEMO_FIND__custom_684440]**					
Last updated:	06/07/2023 11:00					
Time frequency	Annual					
Demographic indicator	Total fertility rate					
TIME	**2021**		**2020**		**2019**	
GEO (Labels)						
European Union - 27 countries (fron	1,53	bep	1,50	ep	1,53	bep
Belgium	1,60		1,55		1,60	
Bulgaria	1,58		1,56		1,58	
Czechia	1,83	b	1,71		1,71	
Denmark	1,72		1,68		1,70	
Germany	1,58		1,53		1,54	
Estonia	1,61		1,58		1,66	
Ireland	1,78		1,63		1,71	e
Greece	1,43		1,39		1,34	
Spain	1,19		1,19		1,23	
France	1,84	p	1,83	p	1,86	p
Croatia	1,58	b	1,48		1,47	
Italy	1,25		1,24		1,27	b

Variable Jahr als Zeile

Zeile als Zusammenfassung (Durchschnitt)

3 Spalten für eine Variable

Quelle: Eurostat Fertility Indicators

Daten mit `tidyr` aufräumen

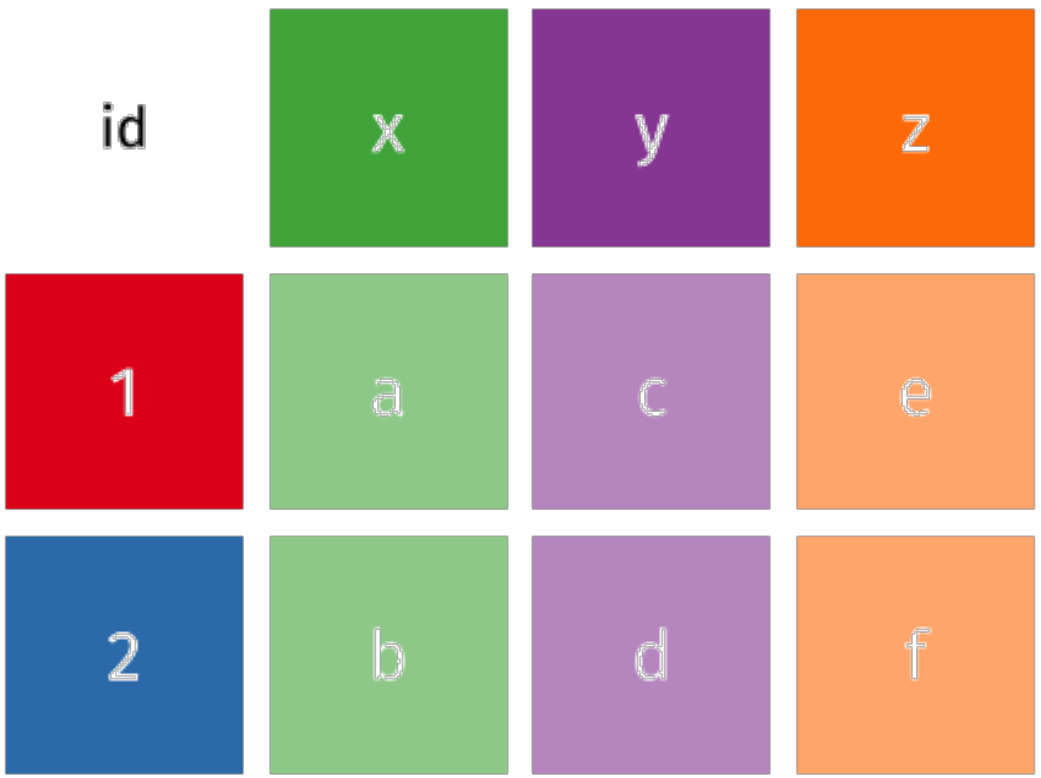
- Daten umformen/pivotieren (erweitern, verlängern)
- Zellen teilen
- Fehlende Werte (`NA`) behandeln

Daten pivotieren

Nicht das…

sondern das!

wide



Daten pivotieren

wide

id	x	y	z
1	a	c	e
2	b	d	f

long

id	key	val
1	x	a
2	x	b
1	y	c
2	y	d
1	z	e
2	z	f

```
wide |>
  pivot_longer(
    cols = x:z,
    names_to = "key",
    values_to = "val"
  )

long |>
  pivot_wider(
    names_from = key,
    values_from = val
  )
```

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

country	type	1999	2000
Afghanistan	cases	745	2666
Afghanistan	population	19987071	20595360
Brazil	cases	37737	80488
Brazil	population	172006362	174504898
China	cases	212258	213766
China	population	1272915272	1280428583

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

```
table1 |>
  mutate(rate = cases / population * 10000)
```

```
# A tibble: 6 × 5
  country      year  cases population  rate
  <chr>       <dbl>  <dbl>      <dbl> <dbl>
1 Afghanistan  1999    745   19987071 0.373
2 Afghanistan  2000   2666   20595360 1.29
3 Brazil       1999  37737  172006362 2.19
4 Brazil       2000  80488  174504898 4.61
5 China        1999 212258 1272915272 1.67
6 China        2000 213766 1280428583 1.67
```

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

```
table2 |>
  pivot_wider(
    names_from = type,
    values_from = count
  ) |>
  mutate(rate = cases / population * 10000)
```

```
# A tibble: 6 × 5
  country      year  cases population  rate
  <chr>       <dbl>  <dbl>      <dbl> <dbl>
1 Afghanistan  1999    745   19987071 0.373
2 Afghanistan  2000   2666   20595360 1.29
3 Brazil       1999  37737  172006362 2.19
4 Brazil       2000  80488  174504898 4.61
5 China        1999 212258 1272915272 1.67
6 China        2000 213766 1280428583 1.67
```

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

```r
table3 |>
  separate_wider_delim(
    cols = rate,
    delim = "/",
    names = c("cases", "population")
  ) |>
  mutate(
    # separate_wider_delim() outputs character
    cases = as.numeric(cases),
    population = as.numeric(population),
    rate = cases / population * 10000
  )
```

```
# A tibble: 6 × 5
  country      year  cases population  rate
  <chr>       <dbl>  <dbl>      <dbl> <dbl>
1 Afghanistan  1999    745   19987071 0.373
2 Afghanistan  2000   2666   20595360 1.29
3 Brazil       1999  37737  172006362 2.19
4 Brazil       2000  80488  174504898 4.61
5 China        1999 212258 1272915272 1.67
6 China        2000 213766 1280428583 1.67
```

country	type	1999	2000
Afghanistan	cases	745	2666
Afghanistan	population	19987071	20595360
Brazil	cases	37737	80488
Brazil	population	172006362	174504898
China	cases	212258	213766
China	population	1272915272	1280428583

```
table4 |>
  pivot_longer(cols = `1999`:`2000`, names_to = "year")
```

```
# A tibble: 12 × 4
   country     type       year       value
   <chr>       <chr>      <chr>      <dbl>
 1 Afghanistan cases      1999         745
 2 Afghanistan cases      2000        2666
 3 Afghanistan population 1999    19987071
 4 Afghanistan population 2000    20595360
 5 Brazil      cases      1999       37737
 6 Brazil      cases      2000       80488
 7 Brazil      population 1999   172006362
 8 Brazil      population 2000   174504898
 9 China       cases      1999      212258
10 China       cases      2000      213766
11 China       population 1999  1272915272
12 China       population 2000  1280428583
```

country	type	1999	2000
Afghanistan	cases	745	2666
Afghanistan	population	19987071	20595360
Brazil	cases	37737	80488
Brazil	population	172006362	174504898
China	cases	212258	213766
China	population	1272915272	1280428583

```
table4 |>
  pivot_longer(cols = `1999`:`2000`, names_to = "year")
  pivot_wider(names_from = type, values_from = value) |>
  mutate(rate = cases / population * 10000)
```

```
# A tibble: 6 × 5
  country     year   cases population  rate
  <chr>       <chr>  <dbl>      <dbl> <dbl>
1 Afghanistan 1999     745   19987071 0.373
2 Afghanistan 2000    2666   20595360 1.29
3 Brazil      1999   37737  172006362 2.19
4 Brazil      2000   80488  174504898 4.61
5 China       1999  212258 1272915272 1.67
6 China       2000  213766 1280428583 1.67
```

Praktikum 04c: Daten pivotieren

prak-04c-pivot.qmd

30:00

Workflow: Code-Style

Using = instead of <- for assignment

R (1993) ← S (1976) ← APL (1962)

The tidyverse Style Guide

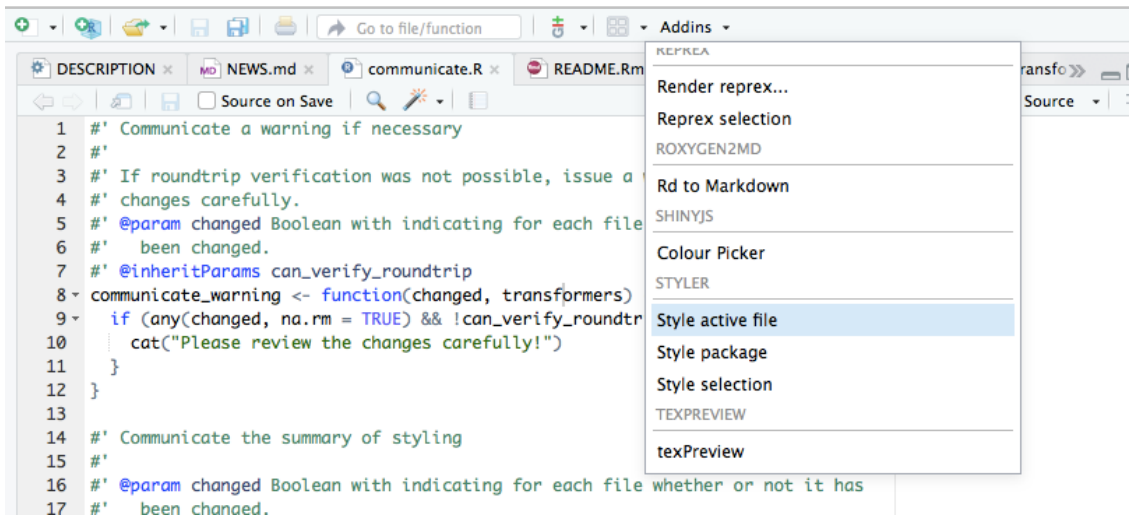
> "Ein guter Kodierungsstil ist wie eine korrekte Zeichensetzung: Man kann auch ohne sie auskommen, abersiemachtalleseinfacherzulesen." – *Hadley Wickham*

```r
# Good
df |>
  mutate(
    sum_xy = x + y,
    prod_xy = x * y
  ) |>
  arrange(sum_xy)

# Bad
df|>mutate(  sum_xy=x+y,prod_xy=x*y)|>arrange( sum_xy )
```

styler

```
1  install.packages("styler")
```



Praktikum: Code-Style

Den Code in eine neue Quarto-Datei formatieren:

```
1 library( palmerpenguins )
2 library(tidyverse   )
3
4 penguins|>filter( species=="Adelie" )|>group_by(island)|>summarize(n=n(),mean_bill
5 mean(bill_length_mm,na.rm=TRUE))|>filter(n>10)
6
7 penguins|>filter(   species=="Chinstrap",island%in%c("Dream","Biscoe"),flipper_len
8 body_mass_g<4000)|>group_by(sex)|>summarize(
9 mean_mass=mean(body_mass_g,na.rm=TRUE),count=n())|>filter(count>5)
```

10:00

Danke! 

Slides created via revealjs and Quarto.

Access slides as PDF.