

# rstatsZH - Data Science mit R

## Vektoren Teil 2 / Daten Importieren / Daten Aufräumen

Lars Schöbitz

2020-04-22

# Rückblick - Woche 5

- Git / GitHub
  - push / pull
  - fork
- Vektoren
  - logical
  - double
  - integer
  - character
- Funktionen zum Umgang mit einem Datum
  - `lubridate::ymd()`
  - `readr::parse_date()`
- Visualisierungen
  - Verbundene Streudiagramm
  - `geom_point()` + `geom_path()`

# Hausaufgabe 5

# Hausaufgabe 5

Gibt es Fragen?

# Ziele für diese Woche

Am Ende dieser Woche könnt ihr:

- Daten aus verschiedenen Formaten in R importieren
- Mittels SQL mit Daten in R Markdown arbeiten
- Erkennen wann es notwendig ist Datums- und Zeitwerte aus Rohdaten in R selbst zu codieren
- Daten aus einem weiten Format in ein langes Format bringen
- Erkennen ob Daten als Tidy data klassifiziert werden können

# Demonstration 2 - Vektoren

1. Schaut mir nochmals beim Programmieren zu
2. Macht euch Notizen und stellt Fragen

# Daten importieren

# Rechteckige Daten





## readr

- `read_csv()` - Dateien mit Kommatrennung der Spalten
- `read_csv2()` - Dateien mit Semicolon getrennten Spalten
- `read_tsv()` - Dateien mit Tab getrennten Spalten
- `read_delim()` - Dateien mit selbst definierter Trennung

## readxl

- `read_excel()` - read xls or xlsx files

# Daten lesen

```
treibhaus <- read_csv("data/ugz_treibhausgasbilanz.csv")
treibhaus
```

```
# A tibble: 27 x 9
  Jahr Strom Kerosin Diesel Benzin Holz_UW_BG_SK Fernwaerme
  <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>       <dbl>
1  1990 0.324   0.638   0.418   1.09     0.021     0.228
2  1991 0.292   0.614   0.415   1.08     0.02     0.227
3  1992 0.263   0.637   0.418   1.08     0.02     0.229
4  1993 0.243   0.651   0.419   1.09     0.02     0.232
5  1994 0.229   0.661   0.416   1.10     0.02     0.234
6  1995 0.242   0.689   0.415   1.10     0.02     0.237
# ... with 21 more rows, and 2 more variables: Erdgas <dbl>,
#   Heizoel_EL <dbl>
```

# Daten schreiben

## Eine Datei schreiben

```
fussball_weltmeister <- tibble(  
  jahr = as.integer(c(2018, 2014, 2010, 2006,  
                     2019, 2015, 2011, 2007)),  
  weltmeisterschaft = c(rep("Männer", 4), rep("Frauen", 4)),  
  titeltraeger = c("Frankreich", "Deutschland", "Spanien",  
                  "Italien", "USA", "USA",  
                  "Japan", "Deutschland"))  
  
write_csv(x = fussball_weltmeister, file = "data/fussball_weltmeister.csv")
```

# Die Datei wieder einlesen

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

```
# A tibble: 8 x 3
  jahr weltmeisterschaft titeltraeger
  <dbl> <chr>              <chr>
1  2018 Männer          Frankreich
2  2014 Männer          Deutschland
3  2010 Männer          Spanien
4  2006 Männer          Italien
5  2019 Frauen          USA
6  2015 Frauen          USA
# ... with 2 more rows
```

# Variablen Namen

# Variablen Namen

```
schlechte_namen <- read_csv("data/bsp_
names(schlechte_namen)
```

```
[1] "Nachname Frau" "Nachname Mann"
```

In R sind Leerzeichen in Variablen nicht erlaubt

```
schlechte_namen %>%
  filter(Nachname Frau == "Meier")
```

```
Error: <text>:2:20: unexpected symbol
1: schlechte_namen %>%
2:   filter(Nachname Frau
               ^
```

Möglich mit Backticks, aber mühsam

```
schlechte_namen %>%
  filter(`Nachname Frau` == "Meier")
```

```
# A tibble: 1 x 2
  `Nachname Frau` `Nachname Mann`
  <chr>           <chr>
1 Meier          Müller
```

# Möglichkeit 1 - Variablen namen in readr Funktion definieren

```
read_csv("data/bsp_namen.csv",  
         col_names = c("nachname_frau", "nachname_mann"),  
         skip = 1)
```

```
# A tibble: 3 x 2  
  nachname_frau nachname_mann  
  <chr>         <chr>  
1 Müller       Meier  
2 Meier        Müller  
3 Schmid       Schmid
```

# Möglichkeit 2 - Variablen namen mit janitor Package bereinigen

- Namen werden standardmässig im sogenannten snake\_case formatiert

```
library(janitor)

namen <- read_csv("data/bsp_namen.csv")

namen %>%
  clean_names()
```

```
# A tibble: 3 x 2
  nachname_frau nachname_mann
  <chr>         <chr>
1 Müller       Meier
2 Meier        Müller
3 Schmid       Schmid
```



# Variable Typen

Welcher Variablen Typ ist die Spalte **id**?

1. character
2. double
3. integer
4. logical

	A	B	C
1	<b>id</b>	<b>name</b>	<b>alter</b>
2	1	Not applicable	46
3	2	Ellen	14
4	NA	Tilde	zwanzig
5	4	Lorraine	39
6	5	Juan	91
7	6	Anush	63
8	.	Sandile	9999
9	8	Martha	38
10	9	Mason	43
11	10	Ege	36

```
read_csv("data/data-na.csv")
```

```
# A tibble: 10 x 4
```

	id	name	alter	bewertung
	<chr>	<chr>	<chr>	<chr>
1	1	Not applicable	46	trifft nicht zu
2	2	Ellen	14	trifft zu
3	<NA>	Tilde	zwanzig	trifft eher zu
4	4	Lorraine	39	teils-teils
5	5	Juan	91	trifft eher nicht zu
6	6	Anush	63	trifft eher nicht zu
7	.	Sandile	9999	trifft zu
8	8	Martha	38	trifft zu
9	9	Mason	43	teils-teils
10	10	Ege	36	teils-teils

# NAs beim einlesen definieren

```
read_csv("data/data-na.csv",  
         na = c("NA", ".", "9999", "Not applicable"))
```

	A	B	C
1	id	name	alter
2	1	Not applicable	46
3	2	Ellen	14
4	NA	Tilde	zwanzig
5	4	Lorraine	39
6	5	Juan	91
7	6	Anush	63
8	.	Sandile	9999
9	8	Martha	38
10	9	Mason	43
11	10	Ege	36

```
# A tibble: 10 x 4
```

	id	name	alter	bewertung
	<dbl>	<chr>	<chr>	<chr>
1	1	<NA>	46	trifft nicht
2	2	Ellen	14	trifft zu
3	NA	Tilde	zwanzig	trifft eher
4	4	Lorraine	39	teils-teils
5	5	Juan	91	trifft eher
6	6	Anush	63	trifft eher
7	NA	Sandile	<NA>	trifft zu
8	8	Martha	38	trifft zu
9	9	Mason	43	teils-teils
10	10	Ege	36	teils-teils

Welcher Variablen Typ ist die Spalte `alter`?

1. character
2. double
3. integer
4. logical

```
dat <- read_csv("data/data-na.csv",  
               na = c("NA", ".", "9999", "Not applicable"))  
dat
```

```
# A tibble: 10 x 4  
      id name      alter bewertung  
  <dbl> <chr>    <chr>    <chr>  
1     1 <NA>      46      trifft nicht zu  
2     2 Ellen    14      trifft zu  
3    NA Tilde    zwanzig trifft eher zu  
4     4 Lorraine 39      teils-teils  
5     5 Juan     91      trifft eher nicht zu  
6     6 Anush    63      trifft eher nicht zu  
7    NA Sandile <NA>    trifft zu  
8     8 Martha   38      trifft zu  
9     9 Mason    43      teils-teils  
10    10 Ege     36      teils-teils
```

# Variable `alter` umwandeln - numerisch

```
dat <- read_csv("data/data-na.csv",  
               na = c("NA", ".", "9999", "Not applicable"))
```

```
dat <- dat %>%  
  mutate(alter = case_when(  
    alter == "zwanzig" ~ "20",          # Wenn "alter" gleich zwanzig dann "20"  
    TRUE ~ alter)) %>%                 # Sonst "alter"  
  mutate(alter = as.numeric(alter))    # Umwandlung in den Typ numerisch
```

```
# A tibble: 10 x 4  
   id name      alter bewertung  
  <dbl> <chr>    <dbl> <chr>  
1     1 <NA>      46 trifft nicht zu  
2     2 Ellen     14 trifft zu  
3    NA Tilde     20 trifft eher zu  
4     4 Lorraine  39 teils-teils  
5     5 Juan      91 trifft eher nicht zu  
# ... with 5 more rows
```

# Variable bewertung - Häufigkeitstabelle

```
dat %>%  
  count(bewertung)
```

```
# A tibble: 5 x 2  
  bewertung      n  
  <chr>      <int>  
1 teils-teils      3  
2 trifft eher nicht zu 2  
3 trifft eher zu    1  
4 trifft nicht zu    1  
5 trifft zu         3
```

	A	B	C	D
1	id	name	alter	bewertung
2	1	Not applicable	46	trifft nicht zu
3	2	Ellen	14	trifft zu
4	NA	Tilde	zwanzig	trifft eher zu
5	4	Lorraine	39	teils-teils
6	5	Juan	91	trifft eher nicht zu
7	6	Anush	63	trifft eher nicht zu
8	.	Sandile	9999	trifft zu
9	8	Martha	38	trifft zu
10	9	Mason	43	teils-teils
11	10	Ege	36	teils-teils



# Variable bewertung - Visualisierung

```
ggplot(dat, aes(x = bewertung)) +  
  geom_bar() +  
  coord_flip()
```

# Variable bewertung umwandeln - faktor

```
vek_bewertung_lvl <- c("trifft nicht zu", "trifft eher nicht zu",  
                       "teils-teils", "trifft eher zu", "trifft zu")  
  
dat <- dat %>%  
  mutate(bewertung = fct_relevel(bewertung, vek_bewertung_lvl))
```

# Variable bewertung - Häufigkeitstabelle

```
dat %>%  
  count(bewertung)
```

```
# A tibble: 5 x 2  
  bewertung      n  
  <fct>      <int>  
1 trifft nicht zu      1  
2 trifft eher nicht zu    2  
3 teils-teils            3  
4 trifft eher zu          1  
5 trifft zu              3
```

# Variable bewertung - Visualisierung

```
ggplot(dat, aes(x = bewertung)) +  
  geom_bar() +  
  coord_flip()
```

# Als eine Code Sequenz

```
vek_bewertung_lvl <- c("trifft nicht zu", "trifft eher nicht zu",  
                      "teils-teils", "trifft eher zu", "trifft zu")  
  
dat_clean <- read_csv("data/data-na.csv",  
                     na = c("NA", ".", "9999", "Not applicable")) %>%  
  mutate(alter = case_when(  
    alter == "zwanzig" ~ "20",          # Wenn "alter" gleich zwanzig dann "20"  
    TRUE ~ alter)) %>%                 # Sonst "alter"  
  mutate(alter = as.numeric(alter)) %>% # Umwandlung in den Typ numerisch  
  mutate(bewertung = fct_relevel(bewertung, # Umwandlung in den Typ faktor  
                                vek_bewertung_lvl)) # Mit definierten Levels
```

# Daten schreiben und wieder lesen

Was ist denn nun wieder mit der Variable `bewertung` passiert?

```
write_csv(dat_clean, file = "data/data-bewertung-clean.csv")  
  
dat_clean_csv <- read_csv(file = "data/data-bewertung-clean.csv")  
  
dat_clean_csv
```

```
# A tibble: 10 x 4  
   id name      alter bewertung  
  <dbl> <chr>    <dbl> <chr>  
1     1 <NA>      46 trifft nicht zu  
2     2 Ellen     14 trifft zu  
3    NA Tilde     20 trifft eher zu  
4     4 Lorraine   39 teils-teils  
5     5 Juan      91 trifft eher nicht zu  
6     6 Anush     63 trifft eher nicht zu
```

# Funktionen: `read_rds()` und `write_rds()`

- Zwischenergebnisse als CSV zu speichern ist unzuverlässig, wenn bestimmte Variablen Typen beibehalten werden sollen
- `read_csv()` kann nicht wissen welche Level eine Faktor Variable hat
- Eine gute Alternative sind RDS-Dateien, ein R-internes Dateiformat

```
write_rds(dat_clean, file = "data/data-bewertung-clean.rds")  
dat_clean_rds <- read_rds(file = "data/data-bewertung-clean.rds")
```

```
dat_clean_rds
```

```
# A tibble: 10 x 4  
   id name      alter bewertung  
  <dbl> <chr>    <dbl> <fct>  
1     1 <NA>      46 trifft nicht zu  
2     2 Ellen     14 trifft zu  
3    NA Tilde     20 trifft eher zu  
4     4 Lorraine   39 teils-teils
```

# Datenbanken - SQL queries in R

- DBI Package: <https://dbi.r-dbi.org/>
- dbplyr Package: <https://dbplyr.tidyverse.org/index.html>



# DBI - Mit einer Datenbank verbinden

Argumente variieren je nach Datenbank, aber das erste Argument ist immer das Datenbank Backend.

```
library(DBI)

con <- dbConnect(

  # Hier wird das Backend definiert
  RMariaDB::MariaDB(),

  host = "relational.fit.cvut.cz",
  port = 3306,
  username = "guest",

  # Niemals Passwort in Skript speichern
  password = rstudioapi::askForPassword("Datenbank Passwort"),
  dbname = "sakila"
)
```

# DBI - Tabellen in Datenbank anzeigen

```
dbListTables(con)
```

```
[1] "actor"      "address"    "category"
[4] "city"       "country"    "customer"
[7] "film"       "film_actor" "film_category"
[10] "film_text"  "inventory"  "language"
[13] "payment"    "rental"     "staff"
[16] "store"
```

# DBI - Spaltennamen einer Tabelle anzeigen

```
dbListFields(con, "film")
```

```
[1] "film_id"      "title"  
[3] "description"  "release_year"  
[5] "language_id"  "original_language_id"  
[7] "rental_duration" "rental_rate"  
[9] "length"       "replacement_cost"  
[11] "rating"       "special_features"  
[13] "last_update"
```

# SQL Queries in R Markdown Dateien

1. Datenbank Verbindung: Code-chunk mit `sql connection=con` starten
2. Daten Output: Resultierende Daten mit `output.var = "NAME"` als Objekt im Environment speichern
3. SQL Code schreiben

```
```{sql connection=con, output.var="film_length_big180"}  
SELECT film_id, title, length  
FROM film  
WHERE length > 180  
```
```

# SQL Query - Mit Daten weiter arbeiten

1. Objekt mit Funktion `as_tibble()` in einen Tibble umwandeln

```
```{r, echo=TRUE}
film_length_big180 %>%
  as_tibble()
```
```

```
# A tibble: 39 x 3
  film_id title          length
  <int> <chr>          <int>
1     24 ANALYZE HOOSIERS    181
2     50 BAKED CLEOPATRA    182
3    128 CATCH AMISTAD    183
4    141 CHICAGO NORTH    185
5    180 CONSPIRACY SPIRIT 184
6    182 CONTROL ANTHEM    185
# ... with 33 more rows
```

# dbplyr - Mit Tabelle in Datenbank verbinden

```
film_tab <- tbl(con, "film")  
film_tab
```

```
# Source:   table<film> [?? x 13]  
# Database: mysql [guest@relational.fit.cvut.cz:NA/sakila]  
  film_id title      description      release_year language_id  
    <int> <chr>      <chr>          <int>      <int>  
1         1 ACADEMY... A Epic Drama of a Fe...      2006          1  
2         2 ACE GOL... A Astounding Epistle...      2006          1  
3         3 ADAPTAT... A Astounding Reflect...      2006          1  
4         4 AFFAIR ... A Fanciful Documenta...      2006          1  
5         5 AFRICAN... A Fast-Paced Documen...      2006          1  
6         6 AGENT T... A Intrepid Panorama ...      2006          1  
# ... with more rows, and 8 more variables:  
#   original_language_id <int>, rental_duration <int>,  
#   rental_rate <dbl>, length <int>, replacement_cost <dbl>,  
#   rating <chr>, special_features <chr>, last_update <dtm>
```

# dbplyr - Queries als dplyr code

```
film_tab %>%  
  select(film_id, title, length) %>%  
  filter(length > 180)
```

```
# Source:   lazy query [?? x 3]  
# Database: mysql [guest@relational.fit.cvut.cz:NA/sakila]  
  film_id title          length  
    <int> <chr>             <int>  
1      24 ANALYZE HOOSIERS      181  
2      50 BAKED CLEOPATRA      182  
3     128 CATCH AMISTAD        183  
4     141 CHICAGO NORTH        185  
5     180 CONSPIRACY SPIRIT    184  
6     182 CONTROL ANTHEM       185  
# ... with more rows
```

# dbplyr - Resultierende Daten aus Datenbank holen

```
film_tab %>%  
  select(film_id, title, length) %>%  
  filter(length > 180) %>%  
  collect()
```

```
# A tibble: 39 x 3  
  film_id title          length  
    <int> <chr>          <int>  
1      24 ANALYZE HOOSIERS      181  
2      50 BAKED CLEOPATRA      182  
3     128 CATCH AMISTAD      183  
4     141 CHICAGO NORTH      185  
5     180 CONSPIRACY SPIRIT  184  
6     182 CONTROL ANTHEM     185  
# ... with 33 more rows
```



# dbplyr - Queries als dplyr code

```
film_tab %>%  
  summarise(min_rate = min(rental_rate),  
            max_rate = max(rental_rate),  
            mean_rate = mean(rental_rate)) %>%  
  collect()
```

```
# A tibble: 1 x 3  
  min_rate max_rate mean_rate  
    <dbl>    <dbl>    <dbl>  
1    0.99    4.99    2.98
```

# R Packages für andere Datentypen

- **googlesheets4**: Google Sheets
- **haven**: SPSS, Stata, und SAS Dateien
- **jsonline**: JSON
- **xml2**: xml
- **rvest**: web scraping
- **httr**: web APIs
- **sparklyr**: data loaded into spark

# Pause

10 : 00

Photo by: Blake Wisz



# Praktikum 9 - Daten importieren - Treibhausgasbilanz

## 2er Teams

1. **E-Mail:** Öffne deine Email und klicke auf den Link zu deinem persönlichen GitHub repo
2. **GitHub:** Klicke auf den grünen Button "Code" und kopiere den Link für das Repo in deine Zwischenablage
3. **RStudio Cloud:** Öffne deinen Arbeitsbereich für den Kurs in der RStudio Cloud
4. **RStudio Cloud / Projects:** Klicke auf "New Project from GitHub Repository"

# Tidy data

# Tidy data

Eigenschaften von Tidy data:

- Eigenschaft 1: Jede Spalte ist eine Variable
- Eigenschaft 2: Jede Reihe ist eine Beobachtung
- Eigenschaft 3: Jede Zelle enthält eine Messung

# Penguins

Erfüllen die Daten die Eigenschaften für Tidy data?

| 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 |

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?



Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?



Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

# Relevanter Unterschied - Ziel der Daten Publikation

## Daten in Tabellen darstellen

- Layout
  - Gut leserlich
  - Kompakt
  - Erkenntnis bringend
- Metadaten

## Daten für weitere Nutzung bereitstellen




- Layout (Tidy data)
  - Eigenschaft 1: Jede Spalte ist eine Variable
  - Eigenschaft 2: Jede Reihe ist eine Beobachtung
  - Eigenschaft 3: Jede Zelle enthält eine Messung
- Keine Metadaten
- Keine Farben, Formatierungen, etc.
- Folgt Standards (Datum: ISO 8601)
- etc.

# Data tidying

# Treibhausgasemissionen

Welche Eigenschaften von Tidy data sind hier nicht erfüllt?

| Jahr | Strom | Kerosin | Diesel | Benzin | Holz_UW_BG_SK | Fernwaerme | Erdgas | Heizuel_EL |
|------|-------|---------|--------|--------|---------------|------------|--------|------------|
| 1990 | 0.324 | 0.638   | 0.418  | 1.087  | 0.021         | 0.228      | 1.015  | 2.445      |
| 1991 | 0.292 | 0.614   | 0.415  | 1.081  | 0.020         | 0.227      | 1.037  | 2.338      |
| 1992 | 0.263 | 0.637   | 0.418  | 1.083  | 0.020         | 0.229      | 1.071  | 2.258      |
| 1993 | 0.243 | 0.651   | 0.419  | 1.093  | 0.020         | 0.232      | 1.108  | 2.185      |
| 1994 | 0.229 | 0.661   | 0.416  | 1.099  | 0.020         | 0.234      | 1.143  | 2.109      |

-  Eigenschaft 1: Jede Spalte ist eine Variable
-  Eigenschaft 2: Jede Reihe ist eine Beobachtung
-  Eigenschaft 3: Jede Zelle enthält eine Messung

# Treibhausgasemissionen

Wie wären alle Eigenschaften erfüllt?

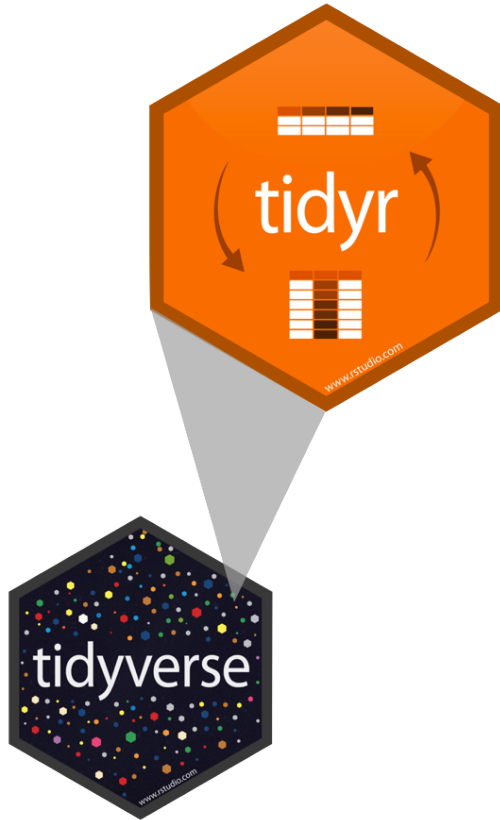
## Vorher

```
# A tibble: 27 x 9
  Jahr Strom Kerosin Diesel Benzin Ho
  <dbl> <dbl>   <dbl> <dbl> <dbl>
1  1990 0.324    0.638 0.418 1.09
2  1991 0.292    0.614 0.415 1.08
3  1992 0.263    0.637 0.418 1.08
4  1993 0.243    0.651 0.419 1.09
5  1994 0.229    0.661 0.416 1.10
6  1995 0.242    0.689 0.415 1.10
# ... with 21 more rows, and 2 more vari
# Heizoel_EL <dbl>
```

## Nachher

```
# A tibble: 216 x 3
  Jahr Energietraeger Emissionen
  <dbl> <chr>             <dbl>
1  1990 Strom          0.324
2  1990 Kerosin        0.638
3  1990 Diesel         0.418
4  1990 Benzin         1.09
5  1990 Holz_UW_BG_SK  0.021
6  1990 Fernwaerme     0.228
# ... with 210 more rows
```

# R Package `tidyr` - Grammatik zum Daten aufräumen



Das Ziel des `tidyr` Package ist des Daten aufzuräumen mittels:

- drehen (pivoting) von Daten um das Datenformat zwischen lang und weit zu wechseln
- teilen und kombinieren von Spalten
- klarstellen mit `NA`s umgegangen werden soll



# Pivoting

- `pivot_longer()` - Daten in ein langes Format bringen
- `pivot_wider()` - Daten in ein weites Format bringen

# pivot\_longer()

- **cols**: Spalten die in das lange Format konvertiert werden sollen
- **names\_to**: Name der neuen Spalte in welcher die gedrehten Variablen auftauchen sollen
- **values\_to**: Name der neuen Spalte in welcher die Werte der gedrehten Variablen auftauchen sollen

```
ghg_tidy <- ghg %>%  
  pivot_longer(  
    cols = Strom:Heizoel_EL,      # Variablen von Strom bis Heizoel_EL  
    names_to = "Energietraeger", # Variablen Namen -> Neue Spalte Energietraege  
    values_to = "Emissionen"     # Variablen Werte -> Neue Spalte Emissionen  
  ) %>%  
  mutate(Jahr = as_factor(Jahr)) # Die Variable Jahr als Faktor definiert
```

# Warum Tidy data? Warum pivoting?

Code

Plot

```
ggplot(data = ghg_tidy,                                # Daten im neuen Format
       mapping = aes(x = Jahr,                          # Bestehende Variable Jahr
                     y = Emissionen,                    # Neue Variable Emissionen
                     fill = Energietraeger)) +          # Energieträger als Farben
  geom_col() +

  # Plot Styling ab hier
  scale_fill_brewer(type = "qual", palette = 1) +
  scale_y_continuous(breaks = seq(0, 7, 1), expand = c(0, 0), limits = c(0, 7))
  labs(title = "Treibhausgasbilanz 1990 bis 2016",
       y = "Treibhausgasemissionen [t CO2eq/Person]",
       x = NULL,
       caption = "Daten: https://data.stadt-zuerich.ch/dataset/ugz\_treibhausgasb
       fill = "Energieträger") +
  theme_minimal(base_size = 14) +
  theme(panel.grid.major.x = element_blank(),
```

# Hausaufgabe

# Feedback

# Ziele erreicht?

Bitte ausfüllen: [kutt.it/rstatszh-eval](https://kutt.it/rstatszh-eval)

Photo by: Virgil Cayasa





Für die Aufmerksamkeit!

Für die R packages `{xaringan}` und `{xaringanthemer}` mit welchen die Folien geschrieben wurden.

Eine PDF Version der Folien kann hier heruntergeladen werden:

[https://github.com/rstatsZH/website/raw/master/slides/e1\\_d06-data-import-tidy/e1\\_d06-data-import-tidy.pdf](https://github.com/rstatsZH/website/raw/master/slides/e1_d06-data-import-tidy/e1_d06-data-import-tidy.pdf)

---

Für [Data Science in a Box](#) und [Remaster the Tidyverse](#), von welchen ich Materialien für diesen Kurs nutze und welche genau wie diese Folien mit [Creative Commons Attribution Share Alike 4.0 International](#) lizenziert sind.