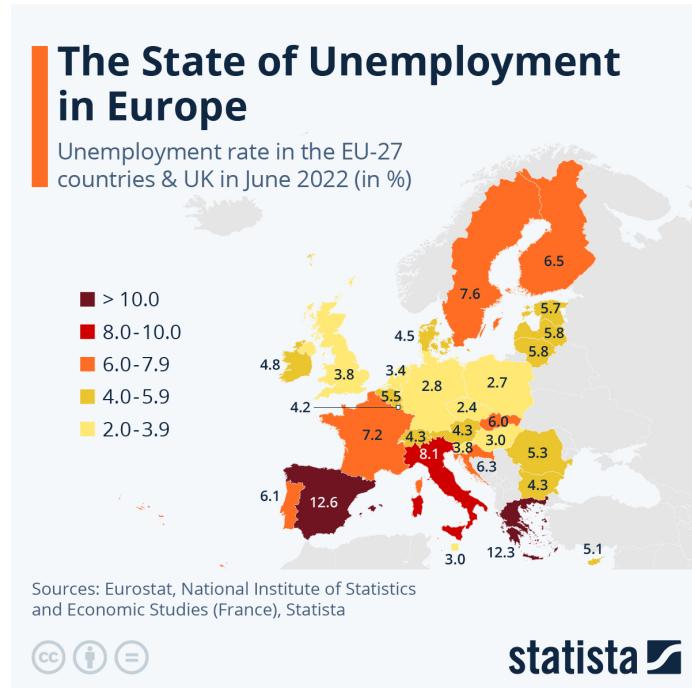


# Case-Study zur Arbeitslosigkeit in Deutschland



# Ziel der Case-Study

Deutschland hat europaweit eine der niedrigsten Arbeitslosenquoten:



Doch gilt dies für alle Regionen in Deutschland?

Warum ist die Arbeitslosenquote in manchen Regionen höher als in anderen?

Dem werden wir in dieser Case-Study auf den Grund gehen.

Quelle: [Fleck, A. \(August 11, 2022\). The State of Unemployment in Europe \(Digital image\).](#)

# Ziele der Case Study

Diese Case-Study besteht aus **mehreren Teilen** und wird Sie durch die komplette Vorlesung als **konkretes Anschauungsobjekt** begleiten.

Diese Case-Study dient als:

- + konkretes und umfangreiches Beispiel für ein Projekt
- + ökonomische und geographische Kenntnisse über Deutschland erhalten
- + Beispiel wie statistische und programmiertechnische Kenntnisse in der empirischen Arbeit eingesetzt werden können

In diesem Foliensatz nutze ich die magitr Pipe %>%<sup>1</sup>, da dies mit dem Paket flipbookr nicht anders möglich ist. Es ist jedoch empfehlenswert grundsätzlich die Base R Pipe zu nutzen | >

# Datensätze herunterladen

# Ersten Teil der Case Study

- + Daten einlesen
- + Daten bearbeiten und in eine geeignete Form bringen (`tidy`)

Anwenden auf

- + Daten zur Arbeitslosenstatistik
- + Daten zur Verschuldung einzelner Landkreise bzw. Gemeinden
- + Daten zum BIP

# Wichtig für die Datenbeschaffung

- ✚ Zuverlässige und qualitativ hochwertige Datenquellen ausfindig machen ([Hier ein Beispiel warum man BIP Daten von diktatorisch regierten Staaten nicht trauen kann](#))
- ✚ Automatisierten Download programmieren
- ✚ Einlesen, verarbeiten und zusammenführen verschiedener Datensätze in R

Verbindung zum 2. RTutor Problem Set:

- ✚ **Im Problem Set:** Kennzahlen zu verschiedenen Ländern der europäischen Union
- ✚ **Hier:** Kennzahlen innerhalb Deutschlands

Sowohl in der Case-Study als auch in den RTutor Problem Sets treffen Sie auf konkrete Probleme, die Sie mit ihren Kenntnissen aus der Vorlesung lösen sollen.

# Daten beschaffen

Woher beziehen wir unsere Informationen?

- + Die Informationen über die Verschuldung der **Gemeinden** finden wir auf den Seiten des Statistischen Bundesamts im Report: [Integrierte Schulden der Gemeinden und Gemeindeverbände](#).
- + Die Informationen zur Arbeitslosigkeit auf **Verwaltungsgemeinschaftsebene** finden wir auf den Seiten der [Bundesagentur für Arbeit](#).
- + Die Informationen zum BIP auf **Landkreisebene** finden wir auf den Seiten der [Statistischen Ämter des Bundes und der Länder](#).

Zuverlässige und qualitativ hochwertige Datenquellen ausfindig machen



# Daten herunterladen

- + Daten können von URLs mit Befehlen aus den Paketen `readxl` und `readr` direkt eingelesen werden
  - + Für Text und Excel-Dateien
- + Allerdings, wenn URL nicht mehr verfügbar, was dann?
  - + Daten immer mit `download.file()` herunterladen und in einem Unterordner `data` abspeichern!

Automatisierten Download programmieren (wird in der ausformulierten Case-Study gemacht) (✓)

Wir haben die Daten bereits im Github Repository `case-study-germany` heruntergeladen und abgespeichert. Klonen Sie dieses Repository von Github auf ihren PC!

# Klonen Sie unsere Github Seite

- + Gehen Sie auf die [Github Seite des Projektkurses](#)
- + Klicken Sie auf des grünen "Code" Button
- + Kopieren Sie sich die [angezeigte HTTPS](#)
- + Gehen Sie in Github Desktop und fügen dort die kopierte HTTPS in "Clone a repository" -> "URL"

[Hier eine Step-by-Step Anleitung](#)

Wenn Sie zu Beginn der Woche in Github Desktop auf "Pull" klicken werden alle Vorlesungsinhalte automatisch aktualisiert, d.h. alle Vorlesungsfolien, die Case-Study, Tutorials etc.! 

05 : 00

# Nötige Pakete laden

```
library(readxl)
library(skimr)
library(tidyverse)
```

```
## — Attaching core tidyverse packages ————— tidyverse 2.0.0 —
## ✓ dplyr     1.1.4    ✓ readr     2.1.5
## ✓forcats   1.0.0    ✓ stringr   1.5.2
## ✓ ggplot2   3.5.2    ✓ tibble    3.3.0
## ✓ lubridate 1.9.4    ✓ tidyverse  1.3.1
## ✓ purrr    1.1.0
## — Conflicts ————— tidyverse_conflicts() —
## ✘ dplyr::filter() masks stats::filter()
## ✘ dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

Was bedeuten die "Messages" nach dem Laden von library(tidyverse)?

# Daten einlesen

Unterschiedliche Dateien und unterschiedliche Tabellenblätter, was sollten wir verwenden?

```
# Öffnen des ZIP-Archivs
# Es sind zwei Tabellen in dem ZIP Archiv, wir interessieren uns für die Anzahl der Arbeitslosen und wählen c
alo_name <- as.character(unzip("../case-study/data/Arbeitslose_2023.xlsx.zip", list = TRUE)$Name)
alo_name <- alo_name[1]
unzip("../case-study/data/Arbeitslose_2023.xlsx.zip", alo_name)
```

**Vermutung:** Durch Tabellenblatt "Inhaltsverzeichnis" könnten wir schlauer werden

```
alo_inhalt <- read_xlsx(alo_name, sheet = "Inhaltsverzeichnis")
head(alo_inhalt, 15)
```

```
## # A tibble: 15 × 1
##   Inhaltsverzeichnis
##   <chr>
## 1 NA
## 2 NA
## 3 Arbeitslose - Zeitreihe
## 4 NA
## 5 NA
## 6 Tabelle
## 7 Bestand an Arbeitslosen
## 8 Kreiszusammenfassung
## 9 Übersicht nach Kreisen
## 10 NA
## 11 Insgesamt
## 12 Rechtskreis
## 13 SGB III
## 14 SGB II
## 15 Geschlecht
```



**Alternative:** Schauen Sie sich die Excel-Datei in Excel oder LibreOffice an und entscheiden Sie dann, welches Tabellenblatt Sie einlesen möchten.

# Spezifizieren welche Spalten eingelesen werden sollen

Welche Information benötigen wir aus der Tabelle

- + Die Anzahl aller Arbeitslosen pro Gemeinde (d.h. SGB II und III gemeinsam) **aus dem Jahr 2023**
- + Die Anzahl der Arbeitslosen pro Gemeinde für einen bestimmten Rechtskreis (z.B. nur SGB II)
- + Die Anzahl der Arbeitslosen pro Gemeinde für einen bestimmten Rechtskreis und ein bestimmtes Alter (z.B. SGB II alle unter 25 Jahre)

Was ist hier eine Beobachtung?

	A	B	C	D	E	F	G	H	I	J
1	 Bundesagentur für Arbeit Statistik									
2										
3	<b>Bestand an Arbeitslosen - Gesamt</b>									
4	Länder, Regierungsbezirke, Kreise und Gemeinden (Gebietsstand = Datenstand)									
5	Zeitreihe, Datenstand: Februar 2023									
6										
7										
8										
9										
10										
11	<a href="#">zurück zum Inhalt</a>									
12	Region	Jahresdurchschnitte		Rechtskreis Insgesamt - Bestand an Arbeitslosen - Gesamt						
13		Jahresdurchschnitt	Jahresdurchschnitt	Januar 2022	Februar 2022	März 2022	April 2022	Mai 2022	Juni 2022	Juli 2022
14	2022	2023	1	2	3	4	5	6	7	
Deutschland	2.418.133	2.608.672	2.462.162	2.427.956	2.362.162	2.309.207	2.259.648	2.362.888	2.470.243	
01 Schleswig-Holstein	81.564	87.757	83.746	83.077	80.512	78.034	75.480	78.646	84.048	
01001 Flensburg, Stadt	3.970	4.131	3.972	3.944	3.901	3.882	3.763	3.824	4.192	
01002 Kiel, Landeshauptstadt	10.315	10.581	10.154	10.049	9.924	9.906	9.615	10.439	11.073	
01003 Lübeck, Hansestadt	8.776	9.216	8.783	8.642	8.509	8.467	8.402	8.724	9.120	
01004 Neumünster, Stadt	3.359	3.604	3.390	3.402	3.364	3.394	3.300	3.332	3.382	
01051 Dithmarschen	3.858	4.126	4.095	4.082	3.922	3.737	3.506	3.750	3.898	
01053 Herzogtum Lauenburg	5.351	5.587	5.333	5.250	5.141	5.053	4.985	5.238	5.694	
01054 Nordfriesland	4.155	4.491	4.790	4.923	4.478	3.777	3.532	3.718	3.921	
01055 Ostholstein	4.824	5.177	5.544	5.556	5.056	4.370	4.136	4.180	4.635	
01056 Pinneberg	8.547	9.280	8.527	8.398	8.180	8.027	7.754	8.429	9.058	
01057 Plön	2.572	2.848	2.810	2.736	2.592	2.492	2.411	2.386	2.537	
01058 Rendsburg-Eckernförde	5.705	6.303	5.891	5.720	5.588	5.492	5.316	5.392	5.952	
01059 Schleswig-Flensburg	5.026	5.573	5.306	5.253	5.070	4.841	4.679	4.723	4.962	
01060 Segeberg	6.804	7.645	6.907	6.898	6.705	6.597	6.398	6.586	6.918	
01061 Steinburg	3.851	4.113	3.867	3.884	3.739	3.711	3.554	3.823	4.059	
01062 Stormarn	4.450	5.083	4.377	4.340	4.343	4.288	4.129	4.102	4.647	
02 Hamburg	73.800	80.806	73.665	73.137	71.989	70.799	69.803	72.685	76.909	
02000 Hamburg, Freie und Hansestadt	73.800	80.806	73.665	73.137	71.989	70.799	69.803	72.685	76.909	
03 Niedersachsen	230.553	251.873	231.323	229.244	222.957	218.422	213.356	223.444	236.115	
031 Statistische Region Braunschwe	46.871	51.773	46.054	45.695	44.639	44.142	43.418	45.694	49.042	
03101 Braunschweig, Stadt	7.014	7.633	6.986	6.930	6.791	6.683	6.498	6.878	7.450	
03102 Salzgitter, Stadt	4.944	5.120	4.869	4.828	4.753	4.734	4.840	4.847	5.076	

# Spezifizieren welche Spalten eingelesen werden sollen

Neben der Anzahl aller Arbeitslosen pro Gemeinde (d.h. SGB II und III gemeinsam) **aus dem Jahr 2023** benötigen wir noch die "Gemeinde-ID" und den Gemeindenamen.

Wie können wir die von uns benötigte Information möglichst einfach extrahieren?

- ✚ Der einfachste Weg: Die ersten acht Zeilen abzuschneiden und die Daten erst ab dort einzulesen.
- ✚ Anschließend behalten wir nur die ersten 3 Spalten

```

alo_skip <- read_xlsx(alo_name, sheet = "Gesamt", sk

alo_skip %>%
  select(c(`...1`, Jahresdurchschnitte, `...3`)) %>%
  mutate(Regionalschluessel = str_extract(`...1`, "[[A-Z].*"))
  Gemeinde = str_extract(`...1`, "[A-Z].*"))
  mutate(alo = as.numeric(`...3`)) %>%
  select(-c(`...1`, Jahresdurchschnitte, `...3`)) %>%
  filter(!is.na(alo))

## # A tibble: 11,177 × 3
##   Regionalschluessel Gemeinde      alo
##   <chr>           <chr>        <dbl>
## 1 <NA>            Region       2023
## 2 <NA>            Deutschland 2608672.
## 3 01              Schleswig-Holstein 87757.
## 4 01001           Flensburg, Stadt 4131.
## 5 01001000        Flensburg, Stadt 4131.
## 6 01002           Kiel, Landeshauptstadt 10581
## 7 01002000        Kiel, Landeshauptstadt 10581
## 8 01003           Lübeck, Hansestadt 9216.
## 9 01003000        Lübeck, Hansestadt 9216.
## 10 01004          Neumünster, Stadt 3604.
## # i 11,167 more rows

```

#Abspeichern als Datensatz data\_alo

```

data_alo <- alo_skip %>%
  select(c(`...1`, Jahresdurchschnitte, `...3`)) %>%
  mutate(Regionalschluessel = str_extract(`...1`, "[[:digit:]]+"),
  Gemeinde = str_extract(`...1`, "[A-Z].*")) %>%
  mutate(alo = as.numeric(`...3`)) %>%
  select(-c(`...1`, Jahresdurchschnitte, `...3`)) %>%
  filter(!is.na(alo))

data_alo <- data_alo[-c(1,2),]

```

# Konsistenzcheck

- + Machen die Angaben Sinn und sind die Daten in sich konsistent?
- + Externe Datenquelle suchen und intern auf Konsistenz prüfen.
- + Informationen aggregieren und mit anderen Quellen vergleichen
- + Zunächst: Anzahl an Arbeitslosen für jedes **Bundesland** in 2023.
  - + zweistelligen Regionalschlüssel
  - + "Buchstaben" für jeden Regionalschlüssel zählen (`nchar()` (number of characters))
- + Alternative Datenquelle: Die Anzahl der Arbeitslosen für das Jahr 2023 unterteilt nach Ländern der Arbeitsagentur
  - + Wichtig: Tabellenblatt 8

```
data_alo %>%
  filter(nchar(Regionalschluessel) == 2) %>%
  rename(bundesland = Regionalschluessel)
```

```
## # A tibble: 16 × 3
##   bundesland Gemeinde      alo
##   <chr>     <chr>       <dbl>
## 1 01        Schleswig-Holstein 87757.
## 2 02        Hamburg          80806.
## 3 03        Niedersachsen  251873.
## 4 04        Bremen            39050.
## 5 05        Nordrhein-Westfalen 710175.
## 6 06        Hessen            181344
## 7 07        Rheinland-Pfalz  110716
## 8 08        Baden-Württemberg 245466.
## 9 09        Bayern            257096.
## 10 10       Saarland          35598.
## 11 11       Berlin            187930.
## 12 12       Brandenburg      78996.
## 13 13       Mecklenburg-Vorpommern 63191.
## 14 14       Sachsen           131069.
## 15 15       Sachsen-Anhalt    82627
## 16 16       Thüringen         64978.
```

```
# Abspeichern als check_alo_bundesland
```

```
check_alo_bundesland <- data_alo %>%
  filter(nchar(Regionalschluessel) == 2) %>%
  rename(bundesland = Regionalschluessel)
```

check\_alo\_bundesland

```
## # A tibble: 16 × 3
##   bundesland Gemeinde      alo
##   <chr>     <chr>       <dbl>
## 1 01        Schleswig-Holstein 87757.
## 2 02        Hamburg          80806.
## 3 03        Niedersachsen  251873.
## 4 04        Bremen            39050.
## 5 05        Nordrhein-Westfalen 710175.
## 6 06        Hessen            181344
## 7 07        Rheinland-Pfalz    110716
## 8 08        Baden-Württemberg  245466.
## 9 09        Bayern            257096.
## 10 10       Saarland          35598.
## 11 11       Berlin             187930.
## 12 12       Brandenburg       78996.
## 13 13       Mecklenburg-Vorpommern 63191.
## 14 14       Sachsen           131069.
## 15 15       Sachsen-Anhalt    82627
## 16 16       Thüringen         64978.
```

Deutschland und Länder  
Berichtsjahr: 2023

Region	Insgesamt	
	absolut	Anteil in %
	1	2
Deutschland	2.608.672	100
Westdeutschland	1.999.881	76,7
Ostdeutschland	608.791	23,3
01 Schleswig-Holstein	87.757	3,4
02 Hamburg	80.806	3,1
03 Niedersachsen	251.873	9,7
04 Bremen	39.050	1,5
05 Nordrhein-Westfalen	710.175	27,2
06 Hessen	181.344	7,0
07 Rheinland-Pfalz	110.716	4,2
08 Baden-Württemberg	245.466	9,4
09 Bayern	257.096	9,9
10 Saarland	35.598	1,4
11 Berlin	187.930	7,2
12 Brandenburg	78.996	3,0
13 Mecklenburg-Vorpommern	63.191	2,4
14 Sachsen	131.069	5,0
15 Sachsen-Anhalt	82.627	3,2
16 Thüringen	64.978	2,5

Beide Datenreihen sind identisch

# INTERNE KONSISTENZ ÜBERPRÜFEN

Berechne: Anzahl an Arbeitslosen für jedes Bundesland als Summe der Arbeitslosen einer Gemeinde.

```
# Nur Gemeindedaten nutzen, dann auf Bundeslandebende die Summe aus den Gemeindedaten berechnen
alo_meta <- data_alo %>%
  filter(nchar(Regionalschluessel) == 8) %>%
  mutate(landkreis = str_extract(Regionalschluessel, "^.{5}"),
        bundesland = str_extract(Regionalschluessel, "^.{2}))
```

```
alo_bundesland <- alo_meta %>%
  group_by(bundesland) %>%
  summarise(total_alo = sum(alo))
```

```
alo_landkreis <- alo_meta %>%
  group_by(landkreis) %>%
  summarise(total_alo = sum(alo)) %>%
  rename(Regionalschluessel = landkreis)
```

```
data_alo %>%
  filter(nchar(Regionalschluessel) == 8) %>%
  mutate(landkreis = str_extract(Regionalschluessel,
  mutate(bundesland = str_extract(Regionalschluessel
```

```
## # A tibble: 10,736 × 5
##   Regionalschluessel Gemeinde      alo landkreis bundesland
##   <chr>           <chr>       <dbl> <chr>    <chr>
## 1 01001000 Flensburg, Stadt 4131. 01001    01
## 2 01002000 Kiel, Landeshauptstadt 10581 01002    01
## 3 01003000 Lübeck, Hansestadt 9216. 01003    01
## 4 01004000 Neumünster, Stadt 3604. 01004    01
## 5 01051001 Albersdorf 123. 01051    01
## 6 01051002 Arkebek 6.08 01051    01
## 7 01051003 Averlak 9.75 01051    01
## 8 01051004 Bargenstedt 14 01051    01
## 9 01051005 Barkenholm 5.17 01051    01
## 10 01051006 Barlt 16.6 01051    01
## # i 10,726 more rows
```

```
alo_meta %>%
  group_by(bundesland) %>%
  summarise(total_alo = sum(alo))
```

```
## # A tibble: 16 × 2
##   bundesland total_alo
##   <chr>        <dbl>
## 1 01          87757.
## 2 02          80806.
## 3 03         251873.
## 4 04          39050.
## 5 05         710175.
## 6 06         181344
## 7 07         110716
## 8 08         245466.
## 9 09         257096.
## 10 10        35598.
## 11 11        187930.
## 12 12        78996.
## 13 13        63191.
## 14 14       131069.
## 15 15        82627
## 16 16        64978.
```

```
alo_meta %>%
  group_by(landkreis) %>%
  summarise(total_alo = sum(alo)) %>%
  rename(Regionalschluessel = landkreis)

## # A tibble: 400 × 2
##       Regionalschluessel total_alo
##   <chr>                <dbl>
## 1 01001                 4131.
## 2 01002                 10581
## 3 01003                 9216.
## 4 01004                 3604.
## 5 01051                 4126.
## 6 01053                 5586.
## 7 01054                 4491.
## 8 01055                 5176.
## 9 01056                 9280.
## 10 01057                2848.
## # i 390 more rows
```

## INTERNE KONSISTENZ ÜBERPRÜFEN

Wir wollen nun die zwei Tabellen miteinander verbinden (besserer Überblick)

- + Datensatz `check_alo_bundeland`: Auf Bundesland aggregierte Zahlen der Arbeitslosigkeit aus den Gemeinden
- + Datessatz `alo_bundesland`: Die schon von der Arbeitsagentur aggregierte Zahlen in unserem Datensatz

```

left_join(check_alo_bundesland, alo_bundesland, by =
  check_consistency
check_consistency %>%
  mutate(diff = alo - total_alo)
## # A tibble: 16 × 5
##   bundesland Gemeinde      alo total_alo    diff
##   <chr>     <chr>      <dbl>    <dbl>    <dbl>
## 1 01        Schleswig-Holstein 87757.  87757.    0
## 2 02        Hamburg       80806.  80806.    0
## 3 03        Niedersachsen 251873. 251873.    0
## 4 04        Bremen        39050.  39050. -7.28e-12
## 5 05        Nordrhein-Westfalen 710175. 710175.    0
## 6 06        Hessen        181344  181344    0
## 7 07        Rheinland-Pfalz 110716  110716    0
## 8 08        Baden-Württemberg 245466. 245466.    0
## 9 09        Bayern        257096. 257096.    0
## 10 10       Saarland      35598.  35598.    0
## 11 11       Berlin         187930. 187930.    0
## 12 12       Brandenburg   78996.  78996.    0
## 13 13       Mecklenburg-Vorpommern 63191. 63191.    0
## 14 14       Sachsen        131069. 131069.    0
## 15 15       Sachsen-Anhalt 82627  82627    0
## 16 16       Thüringen     64978.  64978.    0

```

Es bestehen keine Unstimmigkeiten.

# Pro-Kopf Verschuldung

# Pro-Kopf Verschuldung auf Gemeindeebene

- ✚ Auf Gemeindeebene aus dem Jahr 2023
- ✚ Querschnittsdaten
- ✚ Vom Statistischen Bundesamt direkt als Excel-Tabelle heruntergeladen (✓)

## Welche Tabellenblätter sollten wir nutzen?

```
excel_sheets("../case-study/data/Schulden_2023.xlsx")
```

```
## [1] "Titel"                  "Impressum"      "Inhalt"  
## [4] "Abkürzungen"           "Erläuterungen"  "SH"  
## [7] "NI"                     "NW"            "HE"  
## [10] "RP"                    "BW"            "BY"  
## [13] "SL"                    "BB"            "MV"  
## [16] "SN"                    "ST"            "TH"  
## [19] "Statistische Ämter"
```

# Mehrere Tabellenblätter einlesen

- + Nicht alle Informationen in **einem Tabellenblatt** enthalten
  - + Viele separate Tabellenblätter
  - + Hier müssen wir potentiell eine Operation über mehrere Tabellenblättern anwenden, z.B. durch eine Schleife

Zuerst schauen wir jedoch welche Informationen wir benötigen anhand eines Beispiels:

# Mehrere Tabellenblätter einlesen

```
sh <- read_xlsx("../case-study/data/Schulden_2023.xlsx", sheet = "SH")
head(sh, 20)
```

```
## # A tibble: 20 × 21
##   `Zurück zum Inhalt...` ...1` ...2` ...3` ...4` ...5` ...6` ...7` ...8` ...9` ...10` 
##   <chr>                  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 <NA>                  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 2 <NA>                  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 3 "Tabelle 1: Schulde... <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 4 "nach Höhe der Beteili... <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 5 "Regional-\r\nschlüsse... Geme... Verw... "Bev... Schu... Verä... "Sch... Schu... <NA>  <NA>
## 6 <NA>                  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  zusa... Verä... Schu...
## 7 <NA>                  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 8 <NA>                  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>  <NA>
## 9 <NA>                  <NA>  <NA>  <NA>  EUR  %    "EUR" <NA>  %    EUR
## 10 <NA>                 <NA>  <NA>  <NA>  1    2    "3"   4    5    6
## 11 "010010000000"       Flen... krei... "923... 4370... -5.0... "473... 2289... 2.7   3212...
## 12 "010020000000"       Kiel... krei... "248... 1102... 0     "444... 5804... -6.7   5498...
## 13 "010030000000"       Lübe... krei... "218... 1123... 12.5  "515... 4493... 17.7  3792...
## 14 "010040000000"       Neum... krei... "797... 4607... -8.5  "577... 1183... -5.9   1077...
## 15 "01051"               Krei... Krei... "{13... 6323... 8.9   "466... 4436... 20    4433...
```

# Mehrere Tabellenblätter einlesen

Wir benötigen:

- + "Regionalschlüssel"
- + "Gemeindename"
- + "Einwohner"
- + "Schuldes des öffentlichen Bereichs insgesamt"
- + "Schulden je Einwohner"

Variablenbezeichnungen beginnen in Zeile 5, d.h. wir ignorieren die ersten 4 Zeilen beim Einlesen.

Was ist hier eine Beobachtung?

# Mehrere Tabellenblätter einlesen

Der Übersicht halber wollen wir noch eine Spalte hinzufügen, welche den Namen des Tabellenblattes enthält, welches wir gerade eingelesen haben.

```
# Einlesen des Tabellenblattes "SH" ohne die ersten 5 Zeilen und nur die Spalten 1-7
schulden_individuell <- read_xlsx("../case-study/data/Schulden_2023.xlsx", sheet = "SH", skip = 5) [1:7]
# Umbenennen der ersten 7 Spalten
colnames(schulden_individuell) <- c("Regionalschlüssel", "Gemeinde",
                                      "Verwaltungsform", "Einwohner", "Schulden_gesamt", "Veraenderung_Vorjahr")

# Zusätzliche Spalte hinzufügen mit dem Namen des Tabellenblattes
schulden_individuell$Bundesland <- "SH"
```

```

# Einlesen des Tabellenblattes "SH" ohne die ersten
read_xlsx("../case-study/data/Schulden_2023.xlsx", s
schulden_individuell

# Umbenennen der ersten 7 Spalten
colnames(schulden_individuell) <- c("Regionalschlues
"Verwaltungsform

# Zusätzliche Spalte hinzufügen mit dem Namen des Ta
schulden_individuell$Bundesland <- "SH"

schulden_individuell

```

```

## # A tibble: 1,307 × 8
##   Regionalschlüssel Gemeinde Verwaltungsform Einwohner Schulden_gesam
##   <chr>           <chr>   <chr>           <chr>       <chr>
## 1 <NA>            <NA>    <NA>            <NA>       <NA>
## 2 <NA>            <NA>    <NA>            <NA>       <NA>
## 3 <NA>            <NA>    <NA>            <NA>       <NA>
## 4 <NA>            <NA>    <NA>            <NA>       EUR
## 5 <NA>            <NA>    <NA>            <NA>       1
## 6 010010000000 Flensburg, Stadt kreisfreie Sta... 92323 437050633.509
## 7 010020000000 Kiel, Landeshau... kreisfreie Sta... 248034 1102397047.88
## 8 010030000000 Lübeck, Hansest... kreisfreie Sta... 218062 1123041995.98
## 9 010040000000 Neumünster, Sta... kreisfreie Sta... 79711 460718488.159
## 10 01051          Kreisverwaltung... Kreisverwaltung {135 462} 63235989.909
## # i 1,297 more rows
## # i 3 more variables: Veraenderung_Vorjahr <chr>, Schulden_pro_kopf <chr>,
## #     Bundesland <chr>

```

# Variablen umformen

```
head(schulden_individuell, 15)
```

```
## # A tibble: 15 × 8
##   Regionalschlüssel Gemeinde      Verwaltungsform Einwohner Schulden_gesamt
##   <chr>           <chr>           <chr>           <chr>           <chr>
## 1 <NA>            <NA>            <NA>            <NA>            <NA>
## 2 <NA>            <NA>            <NA>            <NA>            <NA>
## 3 <NA>            <NA>            <NA>            <NA>            <NA>
## 4 <NA>            <NA>            <NA>            <NA>            EUR
## 5 <NA>            <NA>            <NA>            <NA>            1
## 6 010010000000  Flensburg, Stadt kreisfreie Sta... 92323  437050633.5099...
## 7 010020000000  Kiel, Landeshau... kreisfreie Sta... 248034 1102397047.889...
## 8 010030000000  Lübeck, Hansest... kreisfreie Sta... 218062 1123041995.98
## 9 010040000000  Neumünster, Sta... kreisfreie Sta... 79711  460718488.1599...
## 10 01051 Kreisverwaltung... Kreisverwaltung {135 462} 63235989.90999...
## 11 010510011011 Brunsbüttel, St... amtsfreie Geme... 12594  63155054.50999...
## 12 010510044044 Heide, Stadt    amtsfreie Geme... 22252  53835429.41999...
## 13 010515163 Amtsverwaltung ... Amtsverwaltung {15 806} 786603.17
## 14 010515163003 Averlak        amtsangehörige... 587   2379636.319999...
## 15 010515163010 Brickeln      amtsangehörige... 195   1635054.099999...
## # i 3 more variables: Veraenderung_Vorjahr <chr>, Schulden_pro_kopf <chr>,
```

# Variablen umformen

Wir sehen, es gibt immer noch einige Probleme:

- + Die Werte unserer Variablen stehen nicht direkt unter dem Variablenamen
  - + Dies können wir am einfachsten bereinigen indem wir alle NAs im Regionalschlüssel entfernen
- + Die Variablen "Einwohner", "Schulden\_gesamt" und "Schulden\_pro\_Kopf" sind alle als character hinterlegt (<chr> unter dem Variablenamen in der vorherigen Tabelle)
  - + Beispiel warum Klasse character (Zeile 28): Es sind geschweifte Klammern enthalten

```
schulden_individuell[28, ]
```

```
## # A tibble: 1 × 8
##   Regionalschluessel    Gemeinde      Verwaltungsform Einwohner Schulden_gesamt
##   <chr>                  <chr>          <chr>           <chr>       <chr>
## 1 010515163_Summe(Amt) Amt Burg-St. M... Amtsgebiet     {15 806}  {51 577 930}
## # i 3 more variables: Veraenderung_Vorjahr <chr>, Schulden_pro_kopf <chr>,
## #   Bundesland <chr>
```

- + Definition einer Variablen landkreis: Ersten 5 Zeichen im Regionalschlüssel

```
# Die Daten wurden noch nicht schön eingelesen, in d  
# waren die Variablennamen über mehrere Reihen gezog  
schulden_individuell %>%  
  filter(!is.na(Regionalschluessel)) %>%  
  mutate(Schulden_gesamt = as.numeric(Schulden_gesam  
  mutate(Einwohner = as.numeric(Einwohner)) %>%  
  mutate(Schulden_pro_kopf = as.numeric(Schulden_pro  
  mutate(landkreis = str_extract(Regionalschluessel,  
  select(-Veraenderung_Vorjahr) %>%  
#manche Landkreise haben keine Infos zu den Einwohne  
  filter( !is.na( Einwohner ))
```

```
## # A tibble: 10,771 × 8  
##   Regionalschluessel Gemeinde Verwaltungsform Einwohner Schulden_gesamt  
##   <chr>           <chr>       <chr>          <dbl>        <dbl>  
## 1 010010000000 Flensburg, Stadt kreisfreie Sta... 92323 43705061  
## 2 010020000000 Kiel, Landeshau... kreisfreie Sta... 248034 110239704  
## 3 010030000000 Lübeck, Hansest... kreisfreie Sta... 218062 112304199  
## 4 010040000000 Neumünster, Sta... kreisfreie Sta... 79711 46071848  
## 5 010510011011 Brunsbüttel, St... amtsfreie Geme... 12594 6315505  
## 6 010510044044 Heide, Stadt amtsfreie Geme... 22252 5383542  
## 7 010515163003 Averlak      amtsangehörige... 587 237963  
## 8 010515163010 Brickeln     amtsangehörige... 195 163505  
## 9 010515163012 Buchholz     amtsangehörige... 1003 382767  
## 10 010515163016 Burg (Dithmarsc... amtsangehörige... 4204 1446218  
## # i 10,761 more rows  
## # i 3 more variables: Schulden_pro_kopf <dbl>, Bundesland <chr>,  
## #   landkreis <chr>
```

# Konsistenzcheck zum Schulden-Datensatz

# Interne Validität Schulden pro Kopf

✚ `Schulden_pro_Kopf_new` von Hand berechnen

✚ Beachte:

- ✚ Geschweifte Klammern entfernen bei `Schulden_gesamt` (mit `gsub( "[ { } ] )`), als auch die Leerzeichen innerhalb der Zahlen (z.B. 15 653), was wir mit `gsub( " [ [:space:] ] "`) erreichen.
- ✚ Tun wir das nicht, so würden wir wieder NAs im Datensatz erhalten

```
# Erstellen der Vergleichstabelle
schulden_consistency <- schulden_individuell %>%
  filter(!is.na(Einwohner) & !is.na(Regionalschluessel)) %>%
  mutate(
    Einwohner_num = as.numeric(gsub(" [[:space:]{}]", "", Einwohner)),
    Schulden_gesamt = as.numeric(gsub(" [[:space:]{}]", "", Schulden_gesamt)),
    Schulden_pro_kopf = as.numeric(gsub(" [[:space:]{}]", "", Schulden_pro_kopf)),
    Schulden_pro_kopf_new = round(Schulden_gesamt / Einwohner_num, 2),
    landkreis = str_extract(Regionalschluessel, "^.{5}"),
    differenz = Schulden_pro_kopf - Schulden_pro_kopf_new
  ) %>%
  relocate(Regionalschluessel, Einwohner, Einwohner_num, Schulden_pro_kopf, Schulden_pro_kopf_new)
```

Was macht `gsub()` und `str_extract()` hier genau? Und was sind reguläre Ausdrücke?

```

# Erstellen der Vergleichstabelle
schulden_individuell %>%
  filter(!is.na(Einwohner) & !is.na(Regionalschluess)
  mutate(
    Einwohner_num = as.numeric(gsub("[:space:]{ }", ,
    Schulden_gesamt = as.numeric(gsub("[:space:]{ }")
    Schulden_pro_kopf = as.numeric(gsub("[:space:]{ "}
    Schulden_pro_kopf_new = round(Schulden_gesamt /
    landkreis = str_extract(Regionalschluessel, "^.{"
    differenz = Schulden_pro_kopf - Schulden_pro_kopf
  ) %>%
  relocate(Regionalschluessel, Einwohner, Einwohner_

```

## # A tibble: 13,095 × 12

	Regionalschluessel	Einwohner	Einwohner_num	Schulden_pro_kopf
## 1	010010000000	92323	92323	4734.
## 2	010020000000	248034	248034	4445.
## 3	010030000000	218062	218062	5150.
## 4	010040000000	79711	79711	5780.
## 5	01051	{135 462}	135462	467.
## 6	010510011011	12594	12594	5015.
## 7	010510044044	22252	22252	2419.
## 8	010515163	{15 806}	15806	49.8
## 9	010515163003	587	587	4054.
## 10	010515163010	195	195	8385.
## # i 13,085 more rows				
## # i 8 more variables: Schulden_pro_kopf_new <dbl>, Gemeinde <chr>, Verwaltungsform <chr>, Schulden_gesamt <dbl>, Veraenderung_Vorjahr <chr>, Bundesland <chr>, landkreis <chr>, differenz <dbl>				

# Interne Validität Schulden pro Kopf

```
range(schulden_consistency$differenz, na.rm=TRUE)
```

```
## [1] -0.49  0.50
```

Die Differenzen liegen zwischen +/- 50 Cent

# Interne Validität Schulden pro Kopf

Es gibt keine nicht verfügbaren Werte, was gut ist bzgl. der internen Validität.

```
filter(schulden_consistency, is.na(differenz))
```

```
## # A tibble: 1 × 12
##   Regionalschlüssel      Einwohner Einwohner_num Schulden_pro_kopf
##   <chr>                  <chr>       <dbl>            <dbl>
## 1 091765116_Summe(Verwaltungsgemeinsc... {4 803}        4803        NA
## # ℹ 8 more variables: Schulden_pro_kopf_new <dbl>, Gemeinde <chr>,
## #   Verwaltungsform <chr>, Schulden_gesamt <dbl>, Veraenderung_Vorjahr <chr>,
## #   Bundesland <chr>, landkreis <chr>, differenz <dbl>
```

# Bruttoinlandsprodukt

# Informationen bzgl. des Bruttoinlandsprodukts

Nach dem Download bei den Statistischen Ämtern des Bundes und der Länder und einer ersten Betrachtung interessieren uns folgende Tabellenblätter:

- + Betrachten der Daten
  - + Tabellenblatt "1.1" ist für unsere Analyse ausschlaggebend (für das BIP)
  - + Tabellenblatt "3.1" ist für die Anzahl an Erwerbstätigen ausschlaggebend
  - + Tabellenblatt "5" ist für die Anzahl an Einwohnern ausschlaggebend
- + Die ersten vier Zeilen benötigen wir nicht
- + Die letzte Zeile enthält eine kurze Beschreibung die wir nicht benötigen
  - + **Lösung:** Behalte alle Zeilen, welche bei der Lfd. Nr. numerisch sind
- + Die folgenden Variablen benötigen wir nicht für unsere Analyse und können entfernt werden: Lfd. Nr., EU-Code, NUTS 1, NUTS 2, NUTS 3, Land, Gebietseinheit

# Informationen bzgl. des Bruttoinlandsprodukts

```
# Blatt 1.1 einlesen und die ersten 4 Zeilen skippen  
bip <- read_xlsx("../case-study/data/BIP_2023.xlsx", sheet="1.1", skip = 4)  
erwerb <- read_xlsx("../case-study/data/BIP_2023.xlsx", sheet="3.1", skip = 4)  
einwohner <- read_xlsx("../case-study/data/BIP_2023.xlsx", sheet = "5", skip = 4)
```

```
# Zeile löschen in der die `Lfd. Nr.` nicht nummeriert ist
# Zusätzliche Spalten löschen
bip %>%
  filter(is.na(as.numeric(`Lfd. Nr.`))==FALSE) %>%
  select(-c(`Lfd. Nr.`, `EU-Code`, `NUTS 1`, `NUTS 2`))
  rename(Regionalschluessel = `Regional-schlüssel`)
```

```
## # A tibble: 444 × 31
##   Regionalschluessel `1992`  `1994`  `1995`  `1996`  `1997`  `1998`  `1999`  `2000` ...
##   <chr>          <chr>    <chr>    <chr>    <chr>    <chr>    <chr>    <chr>    <dbl>
## 1 08             255866.4... 26264... 27174... 27677... 28219... 29109... 30072... 3.09
## 2 081            110977.0... 11160... 11528... 11678... 12086... 12384... 12779... 1.30
## 3 08111          32946.88... 31736... 32281... 32802... 34339... 33553... 35048... 3.53
## 4 08115          12090.93   11833... 11937... 12097... 13919... 13679... 14424... 1.39
## 5 08116          12275.605  12482... 12748... 13169... 13284... 13952... 14192... 1.44
## 6 08117          5062.037... 5180.... 5447.... 5643.... 5667.... 5838.... 5920.... 6.00
## 7 08118          11714.16   12163... 12756... 12895... 13143... 13516... 13866... 1.47
## 8 08119          8500.405... 8723.... 9320.... 8780.... 8928.... 9175.... 9707.... 1.04
## 9 08121          4219.259  4387.... 4522.... 4510.... 4581.... 5645.... 5282.... 5.27
## 10 08125         6073.524... 6126.... 6577.... 6811.... 7019.... 7645.... 7928.... 8.45
## # i 434 more rows
## # i 22 more variables: `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>,
## #   `2005` <dbl>, `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>,
## #   `2010` <dbl>, `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>,
## #   `2015` <dbl>, `2016` <dbl>, `2017` <dbl>, `2018` <dbl>, `2019` <dbl>,
## #   `2020` <dbl>, `2021` <dbl>, `2022` <dbl>
```

# Informationen bzgl. des Bruttoinlandsprodukts

## Was ist hier eine Beobachtung?

Entsprechend können wir bei den Erwerbstätigen und den Einwohnern vorgehen:

```
# Zeile löschen in der die `Lfd. Nr.` nicht nummerisch ist
# Zusätzliche Spalten löschen
erwerb_wide <- erwerb %>%
  filter(is.na(as.numeric(`Lfd. Nr.`))==FALSE) %>%
  select(-c(`Lfd. Nr.`, `EU-Code`, `NUTS 1`, `NUTS 2`, `NUTS 3`, Land, Gebietseinheit)) %>%
  rename(Regionalschluessel = `Regional-schlüssel`)

einwohner_wide <- einwohner %>%
  filter(is.na(as.numeric(`Lfd. Nr.`))==FALSE) %>%
  select(-c(`Lfd. Nr.`, `EU-Code`, `NUTS 1`, `NUTS 2`, `NUTS 3`, Land, Gebietseinheit)) %>%
  rename(Regionalschluessel = `Regional-schlüssel`)
```

# Informationen bzgl. des Bruttoinlandsprodukts

Datensatz,

- + ist ein Panel: Mehrere Jahre für mehrere Landkreise in Deutschland vorhanden
- + ist im wide Format -> d.h. die Daten sind nicht tidy

```
head(bip_wide, 3)
```

```
## # A tibble: 3 × 31
##   Regionalschlüssel `1992`    `1994` `1995` `1996` `1997` `1998` `1999` `2000` 
##   <chr>           <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 08             255866.41... 26264... 27174... 27677... 28219... 29109... 30072... 3.09e5
## 2 081            110977.071 11160... 11528... 11678... 12086... 12384... 12779... 1.30e5
## 3 08111          32946.883... 31736... 32281... 32802... 34339... 33553... 35048... 3.53e4
## # i 22 more variables: `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>,
## #   `2005` <dbl>, `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>,
## #   `2010` <dbl>, `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>,
## #   `2015` <dbl>, `2016` <dbl>, `2017` <dbl>, `2018` <dbl>, `2019` <dbl>,
## #   `2020` <dbl>, `2021` <dbl>, `2022` <dbl>
```

Was sind die Bedingungen für einen tidy Datensatz?

# Daten in das long-Format überführen

Datensatz ins long-Format überführen mit pivot\_longer:

```
bip_long <- pivot_longer(bip_wide, cols = c("1992":"2022") , names_to = "Jahr", values_to = "BIP")
```

```
Fehler: Can't combine `1992` <character> and `2000` <double>.
```

# Daten in das long-Format überführen

BIP sollte normalerweise nummerisch sein:

- + Klasse `double` sollte korrekt sein
- + umformatieren der Spalten 1992 - 1999
- + mit `across()` kann der `mutate()`-Befehl über mehrere Spalten angewendet werden

```
#BIP von 1992 - 1999 umformen (als numerische Variab  
bip_wide %>%  
  select(`1992`:`1999`) %>%  
  mutate(across(is.character, as.double))
```

```
## # A tibble: 444 × 7  
##   `1992`   `1994`   `1995`   `1996`   `1997`   `1998`   `1999`  
##   <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>  
## 1 255866. 262645. 271747. 276777. 282190. 291100. 300727.  
## 2 110977. 111603. 115281. 116787. 120867. 123842. 127799.  
## 3 32947. 31737. 32281. 32803. 34340. 33553. 35048.  
## 4 12091. 11834. 11938. 12097. 13919. 13679. 14424.  
## 5 12276. 12483. 12749. 13169. 13285. 13952. 14192.  
## 6 5062. 5180. 5447. 5643. 5668. 5839. 5920.  
## 7 11714. 12164. 12756. 12895. 13144. 13516. 13867.  
## 8 8500. 8723. 9320. 8781. 8928. 9176. 9708.  
## 9 4219. 4387. 4523. 4511. 4581. 5646. 5282.  
## 10 6074. 6126. 6577. 6812. 7020. 7646. 7929.  
## # i 434 more rows
```

Entsprechend dann bei den Einwohnern und Erwerbstätigen:

Es wird eine Warnmeldung ausgegeben das NAs bei der Umwandlung erzeugt wurden. Warum?

```
# Erwerbstätige von 1992 – 1999 umformen (als numerische Variable)
erwerb_double <- erwerb_wide %>%
  select(`1992`:`1999`) %>%
  mutate(across(is.character, as.double))
```

```
## Warning: There were 7 warnings in `mutate()` .
## The first warning was:
##  i In argument: `across(is.character, as.double)` .
## Caused by warning:
## ! NAs durch Umwandlung erzeugt
##  i Run `dplyr::last_dplyr_warnings()` to see the 6 remaining warnings.
```

```
# Einwohner von 1992 – 1999 umformen (als numerische Variable)
einwohner_double <- einwohner_wide %>%
  select(`1992`:`1999`) %>%
  mutate(across(is.character, as.double))
```

```
## Warning: There were 7 warnings in `mutate()` .
## The first warning was:
##  i In argument: `across(is.character, as.double)` .
## Caused by warning:
```

# Daten in das long-Format überführen

Wir überprüfen, welche Spalten die Warnung hervorgerufen haben und wo NAs erzeugt wurden

```
bip_wide_test <- bip_wide %>%
  bind_cols(bip_double)

head(filter(bip_wide_test, is.na(`1992...32`)))
```

```
## # A tibble: 6 × 38
##   Regionalschlüssel `1992...2` `1994...3` `1995...4` `1996...5` `1997...6`
##   <chr>          <chr>      <chr>      <chr>      <chr>      <chr>
## 1 13003          .          .          .          .          .
## 2 13004          .          .          .          .          .
## 3 13071          .          .          .          .          .
## 4 13072          .          .          .          .          .
## 5 13073          .          .          .          .          .
## 6 13074          .          .          .          .          .
## # i 32 more variables: `1998...7` <chr>, `1999...8` <chr>, `2000` <dbl>,
## #   `2001` <dbl>, `2002` <dbl>, `2003` <dbl>, `2004` <dbl>, `2005` <dbl>,
## #   `2006` <dbl>, `2007` <dbl>, `2008` <dbl>, `2009` <dbl>, `2010` <dbl>,
## #   `2011` <dbl>, `2012` <dbl>, `2013` <dbl>, `2014` <dbl>, `2015` <dbl>,
## #   `2016` <dbl>, `2017` <dbl>, `2018` <dbl>, `2019` <dbl>, `2020` <dbl>,
```

Eine Umwandlung zu NA geschieht bei den Werten bei denen – eingetragen wurde. D.h. für uns ist es ok hier ein NA einzutragen. Somit können wir die Umwandlung in die Klasse double durchführen:

```
bip_wide <- bip_wide %>%
  select(-(`1992`:`1999`)) %>%
  bind_cols(bip_double)

erwerb_wide <- erwerb_wide %>%
  select(-(`1992`:`1999`)) %>%
  bind_cols(erwerb_double)

einwohner_wide <- einwohner_wide %>%
  select(-(`1992`:`1999`)) %>%
  bind_cols(einwohner_double)
```

# Daten in das long-Format überführen

Nun können wir den Datensatz ins long-Format transferieren und nach dem Jahr sortieren.

- ✚ Einwohner und Erwerbstätige sind in 1000 Personen angegeben, daher Erwerbstätige und Einwohner mit 1000 multiplizieren.
- ✚ BIP ist in 1 Mio. Euro angegeben, daher die Multiplikation mit 1 Mio.

```
# BIP ins long-Format
pivot_longer(bip_wide, cols = c("2000":"1999") , nam
  mutate( Jahr = as.numeric(Jahr),
         bip = bip * 1000000) %>%
  arrange( Jahr )
```

```
## # A tibble: 13,320 × 3
##   Regionalschlüssel Jahr      bip
##   <chr>           <dbl>    <dbl>
## 1 08              1992 255866419000
## 2 081             1992 110977071000
## 3 08111           1992 32946884000
## 4 08115           1992 12090930000
## 5 08116           1992 12275605000
## 6 08117           1992 5062037000
## 7 08118           1992 11714160000
## 8 08119           1992 8500405000
## 9 08121           1992 4219259000
## 10 08125          1992 6073525000
## # i 13,310 more rows
```

Für die Erwerbstätigen und Einwohner entsprechend:

```
# Anzahl der Erwerbstätigen ins long-Format
erwerb_long <- pivot_longer(erwerb_wide, cols = c("2000":"1999") , names_to = "Jahr", values_to = "erw") %>%
  mutate( Jahr = as.numeric(Jahr),
         erw = erw * 1000) %>%
  arrange( Jahr )

# Anzahl der Einwohner ins long-Format
einwohner_long <- pivot_longer(einwohner_wide, cols = c("2000":"1999") , names_to = "Jahr", values_to = "einwohner")
  mutate( Jahr = as.numeric(Jahr),
         einwohner = einwohner * 1000) %>%
  arrange( Jahr )
```

# Konsistenzchecks

Hier sollten Sie selbst aktiv werden und die Daten auf Konsistenz prüfen:

Als Konsistenzcheck könnten Sie hier die Anzahl der Einwohner aus den verschiedenen Datensätzen vergleichen.

# Kartenmaterial hinzufügen

Wir benötigen hier eine Karte von Deutschland mit den einzelnen Verwaltungsgrenzen als SHAPE-File und können diese mittels des `sf`-Pakets einlesen.

Das [OpenData Portal des Bundesamts für Kartographie und Geodäsie](#) stellt die nötigen Informationen kostenlos zur Verfügung.

[Die Dokumentation der Daten](#) sollten wir uns immer zuerst anschauen, bevor wir die Datenquelle herunterladen.

Dies gilt nicht nur für die Geodaten, sondern allgemein für alle Datenreihen.

Bitte versuchen Sie selbst die Daten herunterzuladen und anhand des Regionalschlüssels (ARS) mit dem BIP, den Arbeitslosen und den Schulden zusammenzuführen.

# Datensätze zusammenführen

Nun möchten wir die unterschiedlichen Datensätze noch zu einem zusammenfügen!

Zuerst müssen wir folgende Schritte unternehmen:

- ✚ Informationen zur Verschuldung auf Landkreisebene aggregieren
- ✚ Daten zum BIP auf das Jahr 2022 einschränken (letzter verfügbarer Zeitpunkt, wenn 2023 verfügbar wäre es besser!)
- ✚ Datensätze anhand des Regionalschlüssels miteinander verbinden.

Weiterhin können wir die geografischen Daten separat abspeichern und bei Bedarf anhand des Regionalschlüssels zu unserem Datensatz hinzumergen.

```
# Schulden auf Landkreisebene
schulden_bereinigt %>%
  group_by(landkreis) %>%
  summarise( Schulden_pro_kopf_lk = sum(Schulden_gesamt),
             Einwohner = sum(Einwohner),
             Schulden_gesamt = sum(Schulden_gesamt))
ungroup() %>%
  rename(Regionalschlüssel = landkreis)
```

```
## # A tibble: 396 × 4
##   Regionalschlüssel Schulden_pro_kopf_lk Einwohner Schulden_gesamt
##   <chr>                  <dbl>        <dbl>          <dbl>
## 1 01001                  4734.       92323        437050634.
## 2 01002                  4445.       248034        1102397048.
## 3 01003                  5150.       218062        1123041996.
## 4 01004                  5780.       79711         460718488.
## 5 01051                  3748.       135462        507701224.
## 6 01053                  1814.       204293        370625747.
## 7 01054                  3570.       170722        609555752.
## 8 01055                  3333.       204522        681615233.
## 9 01056                  3290.       322829        1061981989.
## 10 01057                 2667.       131234        350063798.
## # i 386 more rows
```

```
# Anzahl an Erwerbst tigen f r das Jahr 2022  
erwerb_long %>%  
  filter(nchar(Regionalschluessel) == 5 & Jahr == 20  
  select(-Jahr)
```

```
## # A tibble: 398 x 2  
##   Regionalschluessel    erw  
##   <chr>                 <dbl>  
## 1 08111                537749  
## 2 08115                230680  
## 3 08116                286414  
## 4 08117                120195  
## 5 08118                270068  
## 6 08119                206802  
## 7 08121                100428  
## 8 08125                182581  
## 9 08126                75160  
## 10 08127               116646  
## # i 388 more rows
```

```
# Anzahl an Einwohner für das Jahr 2022
einwohner_long %>%
  filter(nchar(Regionalschluessel) == 5 & Jahr == 20
  select(-Jahr)
```

```
## # A tibble: 398 × 2
##   Regionalschluessel einwohner
##   <chr>                <dbl>
## 1 08111               629570
## 2 08115               395862
## 3 08116               536807
## 4 08117               260452
## 5 08118               547865
## 6 08119               429857
## 7 08121               126974
## 8 08125               350541
## 9 08126               114191
## 10 08127              201116
## # i 388 more rows
```

```
# Anzahl der Einwohner mit dem BIP verbinden um das  
left_join(bip_long, einwohner_long, by=c("Regionalsc  
    mutate(bip_pro_kopf = bip / einwohner) %>%  
# BIP auf Landkreisebene  
    filter(nchar(Regionalschluessel) == 5 & Jahr == 20  
    select(-c(Jahr, einwohner))
```

```
## # A tibble: 398 × 3  
##   Regionalschluessel      bip bip_pro_kopf  
##   <chr>          <dbl>        <dbl>  
## 1 08111     58703587000  93244.  
## 2 08115     29582502000  74729.  
## 3 08116     27010925000  50318.  
## 4 08117     9005138000  34575.  
## 5 08118     26245037000  47904.  
## 6 08119     16541107000  38480.  
## 7 08121     8132814000  64051.  
## 8 08125     23554259000  67194.  
## 9 08126     6489785000  56833.  
## 10 08127    10200716000  50721.  
## # i 388 more rows
```

```
# Datensätze zusammenführen

# Basisdatensatz -> Arbeitslosenzahlen pro Landkreis
# Namen der Bundesländer zumergen
alo_landkreis %>%
  mutate(bundesland = str_extract(Regionalschluessel
    left_join(., schulden_kombi, by = "Regionalschlues
    left_join(., bip_kombi, by = "Regionalschluessel"))
# Zahl der Erwerbstätigen zumergen
  left_join(., erwerb_kombi, by = "Regionalschluesse
```

```
## # A tibble: 400 × 9
##   Regionalschluessel total_alo bundesland Schulden_pro_kopf_lk Einwohner
##   <chr>          <dbl> <chr>           <dbl>      <dbl>
## 1 01001          4131. 01             4734.     92323
## 2 01002          10581  01             4445.     248034
## 3 01003          9216.  01             5150.     218062
## 4 01004          3604.  01             5780.     79711
## 5 01051          4126.  01             3748.     135462
## 6 01053          5586.  01             1814.     204293
## 7 01054          4491.  01             3570.     170722
## 8 01055          5176.  01             3333.     204522
## 9 01056          9280.  01             3290.     322829
## 10 01057         2848.  01             2667.     131234
## # i 390 more rows
## # i 4 more variables: Schulden_gesamt <dbl>, bip <dbl>, bip_pro_kopf <dbl>,
## #     erw <dbl>
```

# Übungsaufgaben

Laden Sie sich das durchschnittliche [Arbeitnehmerentgelt pro Arbeitnehmer und Landkreis](#) auf der Seite der Statistischen Ämter des Bundes und der Länder herunter und lesen Sie diesen in R ein.

- ✚ Bilden Sie ein Team von 4-5 Personen
- ✚ Nutzung der Case-Study und Chat AI um die Aufgaben zu erledigen

## Aufgabe:

- ✚ Finden Sie in dem heruntergeladenen Datensatz heraus, was der Unterschied zwischen *Arbeitnehmerentgelt* und *Bruttolöhne- und Gehälter* ist.
- ✚ Lesen Sie die für Sie relevante Tabelle *Bruttolöhne- und Gehälter* in R ein.
- ✚ Bereinigen Sie die Tabelle, d.h. der Datensatz sollte danach `tidy` sein.
- ✚ Berechnen Sie die Bruttolöhne pro Bundesland mit den Bruttolöhnen der einzelnen Landkreise als Konsistenzcheck.
- ✚ Vergleichen Sie ihren Datensatz mit dem auf Github bereitgestellten Datensatz ("einkommen.rds"). Stimmen diese überein?
- ✚ Verbinden Sie die Informationen zu den durchschnittlichen Einkommen mit dem gesamtdatensatz aus dem Abschnitt.

20 : 00

# Hinweise zum Prompting von Chat AI

## Wie sollte ein Prompt aufgebaut sein?

- + Aufgabe:** Was soll Chat AI tun?
  - + Möglichst genau und spezifisch sein
- + Instruktionen:** Wie soll Chat AI es tun?
  - + Soll Chat AI eine besondere Persona sein?
  - + Beispiele für den gewünschten Output mitgeben
  - + Chat AI fragen zur Lösung Schritt-für-Schritt zu kommen
- + Kontext:** Was sollte Chat AI zu seiner Aufgabe wissen?
  - + Wie passt die Aufgabe z.B. zur Vorlesung die Sie gerade besuchen?

## Wenn Sie nicht weiter wissen, dann Fragen Sie Chat AI welche Infos es noch benötigt

- + "Sag mir, welche Informationen du benötigst, um ..."
- + "Stelle mir 5 Fragen, die dir helfen werden, mir die bestmögliche Antwort zu geben."

## Wenn Sie glauben Chat AI hat einen Fehler gemacht, dann sagen Sie ihm dies

## Komplexe Aufgaben in kleine Unteraufgaben herunterbrechen