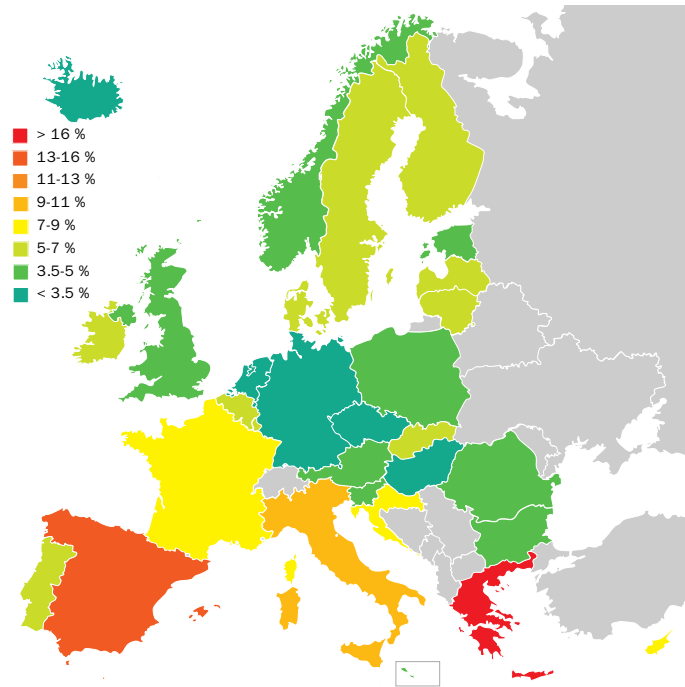


Case-Study zur Arbeitslosigkeit in Deutschland

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 wollen wir in dieser Case-Study auf den Grund gehen.

Quelle: Von Heycci - Daten von Eurostat, CC BY-SA 2.5

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

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
- + 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



Nötige Pakete laden


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

```
## — Attaching core tidyverse packages — tidyverse 2.0.0 —
## ✔ dplyr      1.1.2      ✔ readr      2.1.4
## ✔ forcats   1.0.0      ✔ stringr   1.5.0
## ✔ ggplot2   3.4.2      ✔ tibble    3.2.1
## ✔ lubridate 1.9.3      ✔ tidyr     1.3.0
## ✔ purrr     1.0.2
## — 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
```

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

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_2021.xlsx.zip", list = TRUE)$Name)
alo_name <- alo_name[1]
unzip("../case-study/data/Arbeitslose_2021.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 2021
- + 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?

Weiterhin benötigen wir noch die "Gemeinde-ID" und den Gemeindennamen.

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

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	 Bundesagentur für Arbeit Statistik															
2																
3	Bestand an Arbeitslosen - Gesamt															
4	Länder, Regierungsbezirke, Kreise und Gemeinden (Gebietsstand = Datenstand)															
5	Zeitreihe, Datenstand: Februar 2021															
6	Rechtskreis Insgesamt															
7																
8																
9		Aus Datenschutzgründen und Gründen der statistischen Geheimhaltung werden Zahlenwerte von 1 oder 2 und Daten, aus denen rechnerisch auf einen solchen Zahlenwert geschlossen werden kann, anonymisiert.														
10	zurück zum Inhalt	Jahresdurchschnitte	Rechtskreis Insgesamt - Bestand an Arbeitslosen - Gesamt													Rechtskre
		Jahresdurchschnitt	Jahresdurchschnitt	Januar 2020	Februar 2020	März 2020	April 2020	Mai 2020	Juni 2020	Juli 2020	August 2020	September 2020	Oktober 2020	November 2020	Dezember 2020	Januar 2021
11	Region	2020	2021	1	2	3	4	5	6	7	8	9	10	11	12	13
12	Deutschland	2.695.444	2.613.489	2.425.523	2.395.604	2.335.367	2.643.744	2.812.986	2.853.307	2.910.008	2.955.487	2.847.148	2.759.780	2.699.133	2.707.242	2.900.000
13	01 Schleswig-Holstein	92.140	88.865	85.827	85.007	81.837	92.188	97.207	96.704	98.778	97.552	93.961	92.174	91.617	92.824	98.000
14	01001 Flensburg, Stadt	4.722	4.369	4.421	4.393	4.286	4.897	5.132	5.075	5.198	4.958	4.721	4.522	4.514	4.544	4.000
16	01002 Kiel, Landeshauptstadt	11.776	11.097	10.791	10.733	10.522	11.589	12.170	12.431	12.808	12.720	12.400	12.022	11.622	11.506	12.000
18	01003 Lübeck, Hansestadt	9.404	9.347	8.605	8.425	8.305	9.585	10.078	9.959	10.093	9.907	9.592	9.418	9.396	9.485	10.000
20	01004 Neumünster, Stadt	3.846	3.771	3.522	3.470	3.397	3.782	3.980	4.010	4.114	4.065	4.003	3.956	3.940	3.910	4.000
22	01051 Dithmarschen	4.261	4.143	4.278	4.180	4.012	4.412	4.609	4.490	4.416	4.329	4.048	3.978	4.052	4.331	4.000
139	01053 Herzogtum Lauenburg	5.785	5.603	5.177	5.146	5.127	5.758	6.078	6.128	6.283	6.270	6.037	5.892	5.738	5.784	6.000
272	01054 Nordfriesland	4.970	4.699	5.454	5.539	4.845	5.249	5.497	4.956	4.846	4.652	4.441	4.294	4.708	5.160	5.000
406	01055 Ostholstein	5.788	5.371	6.305	6.262	5.599	5.955	6.178	5.685	5.745	5.637	5.321	5.186	5.592	5.994	6.000
443	01056 Pinneberg	9.486	9.371	8.456	8.316	8.034	9.152	9.845	10.050	10.384	10.314	9.931	9.905	9.752	9.694	10.000
493	01057 Plön	3.149	2.854	2.927	2.877	2.756	3.136	3.301	3.323	3.462	3.386	3.269	3.172	3.093	3.088	3.000
579	01058 Rendsburg-Eckernförde	6.378	6.170	5.785	5.707	5.462	6.273	6.671	6.704	6.977	6.896	6.712	6.558	6.397	6.391	6.000
745	01059 Schleswig-Flensburg	5.694	5.567	5.348	5.324	5.202	5.740	5.937	5.816	5.975	6.031	5.758	5.727	5.694	5.775	6.000
871	01060 Segeberg	7.694	7.456	6.708	6.732	6.574	7.585	7.965	8.199	8.457	8.330	8.122	8.042	7.824	7.790	8.000
967	01061 Steinburg	4.179	4.250	3.670	3.587	3.580	4.189	4.447	4.521	4.526	4.536	4.315	4.277	4.230	4.266	4.000
1079	01062 Stormarn	5.008	4.798	4.380	4.316	4.136	4.886	5.319	5.357	5.494	5.521	5.291	5.225	5.065	5.106	5.000
1135	02 Hamburg	80.677	80.395	68.161	67.710	66.533	77.518	84.426	87.775	91.140	89.807	85.591	84.131	82.969	82.359	86.000
1136	02000 Hamburg, Freie und Hansestadt	80.677	80.395	68.161	67.710	66.533	77.518	84.426	87.775	91.140	89.807	85.591	84.131	82.969	82.359	86.000
1138	03 Niedersachsen	251.377	243.021	230.000	227.926	220.490	246.761	261.588	264.855	269.582	276.410	263.257	255.407	249.156	251.096	267.000
1139	031 Statistische Region Braunschweig	50.066	48.055	45.076	44.726	43.787	49.419	52.468	53.135	54.077	55.517	52.689	51.148	49.408	49.336	52.000
1140	03101 Braunschweig, Stadt	7.563	7.340	6.731	6.567	6.394	7.420	7.801	7.983	8.131	8.499	7.990	7.924	7.666	7.654	8.000
1142	03102 Salzgitter, Stadt	5.399	5.002	4.835	4.886	4.795	5.360	5.699	5.745	5.800	5.913	5.758	5.559	5.266	5.176	5.000
1144	03103 Wolfsburg, Stadt	3.504	3.599	2.947	3.002	2.968	3.396	3.720	3.764	3.815	3.879	3.784	3.701	3.529	3.548	3.000
1146	03151 Gifhorn	4.279	4.150	3.937	3.877	3.732	4.132	4.432	4.515	4.629	4.829	4.531	4.372	4.210	4.154	4.000
1188	03153 Goslar	4.756	4.527	4.353	4.243	4.143	4.797	5.162	5.118	5.103	5.143	4.890	4.724	4.637	4.756	5.000
1196	03154 Helmstedt	3.096	3.018	2.931	2.812	2.808	3.008	3.160	3.192	3.299	3.323	3.273	3.174	3.111	3.058	3.000
1220	03155 Northeim	3.977	3.802	3.786	3.626	3.499	4.054	4.298	4.369	4.251	4.287	3.954	3.871	3.797	3.931	4.000
1232	03157 Peine	3.796	3.740	3.297	3.358	3.292	3.711	3.920	3.918	4.058	4.219	4.065	3.993	3.884	3.833	4.000
1240	03158 Wolfenbüttel	3.127	3.038	2.788	2.842	2.778	3.092	3.274	3.310	3.397	3.563	3.315	3.113	3.009	3.047	3.000
1273	03159 Göttingen	10.568	9.840	9.471	9.513	9.378	10.449	11.002	11.221	11.594	11.862	11.129	10.717	10.299	10.179	10.000
1313	032 Statistische Region Hannover	78.580	78.518	70.434	70.055	68.354	76.192	80.780	82.654	84.521	86.567	83.312	81.276	79.170	79.648	83.000
1314	03241 Region Hannover	47.396	48.229	41.548	41.324	40.671	45.457	48.549	49.986	51.277	52.504	50.862	49.562	48.270	48.736	51.000
1336	03251 Diepholz	5.607	5.362	5.093	4.944	4.770	5.449	5.822	5.873	6.037	6.216	5.926	5.855	5.622	5.675	5.000
1382	03252 Hameln-Pyrmont	5.261	5.169	5.010	5.007	4.750	5.169	5.389	5.452	5.509	5.693	5.473	5.314	5.231	5.131	5.000
1391	03254 Hildesheim	9.614	9.572	8.651	8.793	8.705	9.579	9.996	10.166	10.291	10.553	10.041	9.721	9.433	9.442	9.000
1412	03255 Holzminden	2.375	2.326	2.307	2.315	2.215	2.300	2.404	2.403	2.470	2.567	2.408	2.408	2.324	2.378	2.000
1445	03256 Nienburg (Weser)	3.605	3.448	3.398	3.342	3.120	3.593	3.763	3.809	3.883	3.914	3.699	3.580	3.559	3.602	3.000
1482	03257 Schaumburg	4.723	4.412	4.427	4.330	4.123	4.645	4.857	4.965	5.054	5.120	4.903	4.836	4.731	4.684	4.000

```
alo_skip <- read_xlsx(alo_name, sheet = "Gesamt", sk
alo_skip %>%
  select(c(`...1`, Jahresdurchschnitte, `...3`)) %>%
  mutate(Regionalschlüssel = str_extract(`...1`, "[
    Gemeinde = str_extract(`...1`, "[A-Z].*"))
  mutate(alo = as.numeric(`...3`)) %>%
  select(-c(`...1`, Jahresdurchschnitte, `...3`)) %>
  filter(!is.na(alo))
```

```
## # A tibble: 11,182 × 3
##   Regionalschlüssel Gemeinde      alo
##   <chr>              <chr>      <dbl>
## 1 <NA>              Region      2021
## 2 <NA>              Deutschland 2613489
## 3 01                Schleswig-Holstein 88865.
## 4 01001            Flensburg, Stadt  4369.
## 5 01001000         Flensburg, Stadt  4369.
## 6 01002            Kiel, Landeshauptstadt 11097.
## 7 01002000         Kiel, Landeshauptstadt 11097.
## 8 01003            Lübeck, Hansestadt  9347.
## 9 01003000         Lübeck, Hansestadt  9347.
## 10 01004           Neumünster, Stadt  3771.
## # i 11,172 more rows
```

```
#Abspeichern als Datensatz data_alo
```

```
data_alo <- alo_skip %>%
  select(c(`...1`, Jahresdurchschnitte, `...3`)) %>%
  mutate(Regionalschlüssel = 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 2021.
 - + zweistelligen `Regionalschlüssel`
 - + "Buchstaben" für jeden `Regionalschlüssel` zählen (`nchar()` (number of characters))
- + **Alternative Datenquelle:** Die Anzahl der Arbeitslosen für das Jahr 2021 unterteilt nach Ländern der Arbeitsagentur
 - + Wichtig: Tabellenblatt 8

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

```
## # A tibble: 16 × 3  
##   bundesland Gemeinde      alo  
##   <chr>      <chr>      <dbl>  
## 1 01      Schleswig-Holstein  88865.  
## 2 02      Hamburg      80395.  
## 3 03      Niedersachsen  243021.  
## 4 04      Bremen        39292.  
## 5 05      Nordrhein-Westfalen  718220.  
## 6 06      Hessen        178086.  
## 7 07      Rheinland-Pfalz   112137.  
## 8 08      Baden-Württemberg  247774.  
## 9 09      Bayern        262186.  
## 10 10     Saarland        36156.  
## 11 11     Berlin        198401.  
## 12 12     Brandenburg     78463.  
## 13 13     Mecklenburg-Vorpommern  62410.  
## 14 14     Sachsen        124743.  
## 15 15     Sachsen-Anhalt    81093.  
## 16 16     Thüringen       62249.
```

```
# Abspeichern als check_alo_bundesland
```

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

```
check_alo_bundesland
```

```
## # A tibble: 16 × 3
##   bundesland Gemeinde      alo
##   <chr>      <chr>      <dbl>
## 1 01 Schleswig-Holstein 88865.
## 2 02 Hamburg           80395.
## 3 03 Niedersachsen    243021.
## 4 04 Bremen           39292.
## 5 05 Nordrhein-Westfalen 718220.
## 6 06 Hessen           178086.
## 7 07 Rheinland-Pfalz  112137.
## 8 08 Baden-Württemberg 247774.
## 9 09 Bayern           262186.
## 10 10 Saarland         36156.
## 11 11 Berlin           198401.
## 12 12 Brandenburg      78463.
## 13 13 Mecklenburg-Vorpommern 62410.
## 14 14 Sachsen           124743.
## 15 15 Sachsen-Anhalt     81093.
## 16 16 Thüringen        62249.
```

```
include_graphics("../figs/Alo_Laender.png")
```

Deutschland und Länder
Berichtsjahr: 2021

Region	Be	
	Insgesamt	
	absolut	Anteil in %
	1	2
Deutschland	2.613.489	100
Westdeutschland	2.006.132	76,8
Ostdeutschland	607.357	23,2
01 Schleswig-Holstein	88.865	3,4
02 Hamburg	80.395	3,1
03 Niedersachsen	243.021	9,3
04 Bremen	39.292	1,5
05 Nordrhein-Westfalen	718.220	27,5
06 Hessen	178.086	6,8
07 Rheinland-Pfalz	112.137	4,3
08 Baden-Württemberg	247.774	9,5
09 Bayern	262.186	10,0
10 Saarland	36.156	1,4
11 Berlin	198.401	7,6
12 Brandenburg	78.463	3,0
13 Mecklenburg-Vorpommern	62.410	2,4
14 Sachsen	124.743	4,8
15 Sachsen-Anhalt	81.093	3,1
16 Thüringen	62.249	2,4

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 Bundeslandebene die Summe aus den Gemeindedaten berechnen
alo_meta <- data_alo %>%
  filter(nchar(Regionalschlüssel) == 8) %>%
  mutate(landkreis = str_extract(Regionalschlüssel, "^.{5}"),
         bundesland = str_extract(Regionalschlüssel, "^.{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(Regionalschlüssel = landkreis)
```

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

```

```
## # A tibble: 10,741 × 5
##   Regionalschlüssel Gemeinde      alo landkreis bundesland
##   <chr>              <chr>      <dbl> <chr>      <chr>
## 1 01001000          Flensburg, Stadt  4369.  01001      01
## 2 01002000          Kiel, Landeshauptstadt 11097.  01002      01
## 3 01003000          Lübeck, Hansestadt   9347.  01003      01
## 4 01004000          Neumünster, Stadt   3771.  01004      01
## 5 01051001          Albersdorf         122.   01051      01
## 6 01051002          Arkebek             5.17  01051      01
## 7 01051003          Averlak              8.08  01051      01
## 8 01051004          Bargenstedt         11.2   01051      01
## 9 01051005          Barkenholm          1.58  01051      01
## 10 01051006          Barlt               17.7   01051      01
## # i 10,731 more rows
```

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

```
## # A tibble: 16 × 2  
##   bundesland total_alo  
##   <chr>         <dbl>  
## 1 01             88865.  
## 2 02             80395.  
## 3 03            243021.  
## 4 04             39292.  
## 5 05            718220.  
## 6 06            178086.  
## 7 07            112137.  
## 8 08            247774.  
## 9 09            262186.  
## 10 10             36156.  
## 11 11            198401.  
## 12 12             78463.  
## 13 13             62410.  
## 14 14            124743.  
## 15 15             81093.  
## 16 16             62249.
```

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

```
## # A tibble: 400 × 2  
##   Regionalschluessel total_alo  
##   <chr>              <dbl>  
## 1 01001              4369.  
## 2 01002             11097.  
## 3 01003              9347.  
## 4 01004              3771.  
## 5 01051              4143.  
## 6 01053              5603.  
## 7 01054              4699  
## 8 01055              5371  
## 9 01056              9371.  
## 10 01057             2854.  
## # 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
- + Datensatz `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 88865.    88865.    0
## 2 02 Hamburg          80395.    80395.    0
## 3 03 Niedersachsen    243021.   243021.    0
## 4 04 Bremen           39292.    39292.    0
## 5 05 Nordrhein-Westfalen 718220.   718220.    0
## 6 06 Hessen           178086.   178086.    0
## 7 07 Rheinland-Pfalz   112137.   112137.    0
## 8 08 Baden-Württemberg 247774.   247774.    0
## 9 09 Bayern            262186.   262186.    0
## 10 10 Saarland          36156.    36156.    0
## 11 11 Berlin            198401.   198401.    0
## 12 12 Brandenburg        78463.    78463.    0
## 13 13 Mecklenburg-Vorpommern 62410.    62410.    0
## 14 14 Sachsen            124743.   124743.    0
## 15 15 Sachsen-Anhalt      81093.    81093.    0
## 16 16 Thüringen          62249.    62249.    0

```

Es bestehen keine Unstimmigkeiten.

Pro-Kopf Verschuldung

Pro-Kopf Verschuldung auf Gemeindeebene

- + Auf Gemeindeebene aus dem Jahr 2021
- + Querschnittsdaten
- + Vom Statistischen Bundesamt direkt als Excel-Tabelle heruntergeladen (✓)

Welche Tabellenblätter sollten wir nutzen?

```
excel_sheets("../case-study/data/Schulden_2021.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 kommt die `for`-Schleife zum Einsatz

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

Mehrere Tabellenblätter einlesen

```
sh <- read_xlsx("../case-study/data/Schulden_2021.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> <chr>
## 1 <NA>                  <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 2 <NA>                  <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 3 "Tabelle 1:   Schulde... <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 4 "nach Höhe der Beteili... <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
## 5 "Regional-\r\nschlüsse... Geme... Verw... "Ein... 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> <NA>
## 8 <NA>                  <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... "899... 4545... 0      "505... 2310... -9.5  4625...
## 12 "010020000000"      Kiel... krei... "245... 1039... 6.1    "422... 5534... 6.7   5531...
## 13 "010030000000"      Lübe... krei... "215... 1090... 1.2    "507... 4305... -5.9  4256...
## 14 "010040000000"      Neum... krei... "796... 4532... 5.2    "568... 1141... 10.6  1029...
## 15 "01051"             Krei... Krei... "{13... 4296... -14.8  "322... 1954... -20.7 1952...
```

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_2021.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_2021.xlsx", sheet = "SH",
           schulden_individuell)

# Umbenennen der ersten 7 Spalten
colnames(schulden_individuell) <- c("Regionalschlüssel", "Gemeinde",
                                   "Verwaltungsform", "Einwohner",
                                   "Schulden_gesamt", "Schulden_pro_kopf")

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

```
## # A tibble: 1,310 × 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... 89949      454539445.970
## 7 010020000000    Kiel, Landeshau... kreisfreie Sta... 245841     1039095143.43
## 8 010030000000    Lübeck, Hansest... kreisfreie Sta... 215051     1090890891.45
## 9 010040000000    Neumünster, Sta... kreisfreie Sta... 79683      453215674.295
## 10 01051          Kreisverwaltung... Kreisverwaltung {133 401} 42967438.6200
## # i 1,300 more rows
## # i 3 more variables: Veraenderung_Vorjahr <chr>, Schulden_pro_kopf <chr>,
## #   Bundesland <chr>
```


Mehrere Tabellenblätter einlesen

Nun können wir genauso bei allen anderen Tabellenblättern vorgehen:

```
# Daten mit for-Schleife einlesen (Struktur gleich wie im vorherigen Chunk)
sheet_names <- excel_sheets("../case-study/data/Schulden_2021.xlsx")
# Einlesen der Tabellenblätter 7-18 (alle Bundesländer)
sheet_read <- sheet_names[7:18]
for (i in 1:length(sheet_read)){
  tmp <- read_xlsx("../case-study/data/Schulden_2021.xlsx", sheet = sheet_read[i], skip = 5)[1:7]
  tmp$Bundesland <- sheet_read[i]
  colnames(tmp) <- c("Regionalschlüssel", "Gemeinde", "Verwaltungsform",
                    "Einwohner", "Schulden_gesamt", "Veraenderung_Vorjahr", "Schulden_pro_kopf", "Bundesland")
  # Daten aller weiteren Tabellenblätter unter den aktuellen Datensatz anheften
  schulden_individuell <- bind_rows(schulden_individuell, tmp)
}
```

Eine zusätzliche Spalte generieren, welche die Information pro Bundesland enthält

```
# Daten mit for-Schleife einlesen (Struktur gleich wie
excel_sheets("../case-study/data/Schulden_2021.xlsx")
sheet_names

# Einlesen der Tabellenblätter 7-18 (alle Bundesländer)
sheet_names[7:18] ->
sheet_read

length(sheet_read)

for (i in 1:length(sheet_read)){
  tmp <- read_xlsx("../case-study/data/Schulden_2021_
  tmp$Bundesland <- sheet_read[i]
  colnames(tmp) <- c("Regionalschlüssel", "Gemeinde",
                    "Einwohner", "Schulden_gesamt",
                    "Veraenderung_Vorjahr", "Schulden_pro_kopf")
# Daten aller weiteren Tabellenblätter unter den akt
schulden_individuell <- bind_rows(schulden_individu
}

schulden_individuell
```

```
## [1] 12
```

```
## # A tibble: 25,234 × 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... 89949      454539445.970
## 7 010020000000      Kiel, Landeshau... kreisfreie Sta... 245841     1039095143.43
## 8 010030000000      Lübeck, Hansest... kreisfreie Sta... 215051     1090890891.45
## 9 010040000000      Neumünster, Sta... kreisfreie Sta... 79683      453215674.299
## 10 01051            Kreisverwaltung... Kreisverwaltung {133 401} 42967438.6200
## # i 25,224 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
##   Regionalschluessel 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... 89949      454539445.9700...
## 7 010020000000      Kiel, Landeshau... kreisfreie Sta... 245841     1039095143.430...
## 8 010030000000      Lübeck, Hansest... kreisfreie Sta... 215051     1090890891.450...
## 9 010040000000      Neumünster, Sta... kreisfreie Sta... 79683      453215674.2999...
## 10 01051      Kreisverwaltung... Kreisverwaltung {133 401} 42967438.62000...
## 11 010510011011      Brunsbüttel, St... amtsfreie Geme... 12324      54641169.33000...
## 12 010510044044      Heide, Stadt      amtsfreie Geme... 21515      42854871.67999...
## 13 010515163      Amtsverwaltung ... Amtsverwaltung  {15 547} 850386.17
## 14 010515163003      Averlak           amtsangehörige... 554        1917334.710000...
## 15 010515163010      Brickeln          amtsangehörige... 200        1176540.52
## # 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 Variablennamen
 - + Dies können wir am einfachsten bereinigen indem wir alle `NA`s im Regionalschlüssel entfernen
- + Die Variablen "Einwohner", "Schulden_gesamt" und "Schulden_pro_Kopf" sind alle als `character` hinterlegt (`<chr>` unter dem Variablennamen in der vorherigen Tabelle)
 - + Beispiel warum Klasse `character` (Zeile 28): Es sind geschweifte Klammern enthalten

```
schulden_individuell[28,]
```

```
## # A tibble: 1 × 8
##   Regionalschlüssel  Gemeinde      Verwaltungsform Einwohner Schulden_gesamt
##   <chr>             <chr>          <chr>           <chr>      <chr>
## 1 010515163_Summe(Amt) Amt Burg-St. M... Amtsgebiet      {15 547} {35 628 306}
## # 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: 10,785 × 8
##   Regionalschluessel Gemeinde      Verwaltungsform Einwohner Schulden_gesam
##   <chr>                <chr>                <chr>          <dbl>          <dbl>
## 1 0100100000000      Flensburg, Stadt kreisfreie Sta...    89949    45453944
## 2 0100200000000      Kiel, Landeshau... kreisfreie Sta...   245841   103909514
## 3 0100300000000      Lübeck, Hansest... kreisfreie Sta...   215051   109089089
## 4 0100400000000      Neumünster, Sta... kreisfreie Sta...    79683    45321567
## 5 010510011011      Brunsbüttel, St... amtsfreie Geme...    12324    5464116
## 6 010510044044      Heide, Stadt      amtsfreie Geme...    21515    4285487
## 7 010515163003      Averlak           amtsangehörige...     554     191733
## 8 010515163010      Brickeln          amtsangehörige...     200     117654
## 9 010515163012      Buchholz          amtsangehörige...     997     197665
## 10 010515163016      Burg (Dithmarsc... amtsangehörige...    4166    1035990
## # i 10,775 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:**
 - + Geschweiften Klammern entfernen bei Schulden_gesamt (mit `str_remove_all`), 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
 - + Durch die ifelse Bedingung wird der Befehl `str_remove_all` nur angewendet, wenn tatsächlich geschweifte Klammern vorhanden sind

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

```
# Erstellen der Vergleichstabelle
schulden_individuell %>%
  filter( !is.na(Einwohner) & !is.na(Regionalschlues
  mutate(Schulden_gesamt = ifelse(is.na(as.numeric(S
    as.numeric(gsub("[
    as.numeric(Schulde
  mutate(Schulden_pro_kopf = ifelse(is.na(as.numeric
    as.numeric(gsub(
    as.numeric(Schul
  mutate(Einwohner_num = ifelse(is.na(as.numeric(Ein
    as.numeric(gsub("[[:
    as.numeric(Einwohner
  mutate(Schulden_pro_kopf_new = round(Schulden_gesa
  relocate(Regionalschluessel, Einwohner, Einwohner_
  mutate(landkreis = str_extract(Regionalschluessel,
    differenz = Schulden_pro_kopf - Schulden_pr
```

```
## # A tibble: 13,114 × 12
##   Regionalschluessel Einwohner Einwohner_num Schulden_pro_kopf
##   <chr>                <chr>          <dbl>          <dbl>
## 1 0100100000000      89949          89949          5053.
## 2 0100200000000      245841        245841          4227.
## 3 0100300000000      215051        215051          5073.
## 4 0100400000000      79683         79683          5688.
## 5 01051           {133 401}      133401           322.
## 6 010510011011      12324         12324          4434.
## 7 010510044044      21515         21515          1992.
## 8 010515163        {15 547}      15547            54.7
## 9 010515163003       554           554           3461.
## 10 010515163010      200           200           5883.
## # i 13,104 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: 0 × 12  
## #   12 variables: Regionalschlüssel <chr>, Einwohner <chr>,  
## #   Einwohner_num <dbl>, Schulden_pro_kopf <dbl>, 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_2022.xlsx", sheet="1.1", skip = 4)  
erwerb <- read_xlsx("../case-study/data/BIP_2022.xlsx", sheet="3.1", skip = 4)  
einwohner <- read_xlsx("../case-study/data/BIP_2022.xlsx", sheet = "5", skip = 4)
```

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

```
## # A tibble: 445 × 30
##   Regionalschlüssel `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 435 more rows
## # i 21 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>
```

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 numerisch 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(Regionalschlüssel = `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(Regionalschlüssel = `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 × 30
##   Regionalschlüssel `1992`      `1994` `1995` `1996` `1997` `1998` `1999` `2000`
##   <chr>             <chr>      <chr> <chr> <chr> <chr> <chr> <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 21 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>
```

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":"2021") , names_to = "Jahr", values_to = "BIP")
```

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

Daten in das `long`-Format überführen

BIP sollte normalerweise numerisch 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: 445 × 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 435 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...31`)))
```

```
## # A tibble: 6 × 37  
##   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 31 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 Erwerbstätigen in 1000 Personen angegeben, daher Erwerbstätigen 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: 12,905 × 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 12,895 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 = "einw") %>%
  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 2021 einschränken.
- + 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))
  rename(Regionalschlüssel = landkreis)
```

```
## # A tibble: 396 × 4
##   Regionalschlüssel Schulden_pro_kopf_lk Einwohner Schulden_gesamt
##   <chr>                <dbl>      <dbl>          <dbl>
## 1 01001                5053.    89949    454539446.
## 2 01002                4227.    245841   1039095143.
## 3 01003                5073.    215051   1090890891.
## 4 01004                5688.     79683    453215674.
## 5 01051                2826.    133401    376925967.
## 6 01053                1613.    199992    322536846.
## 7 01054                3206.    167710    537632139.
## 8 01055                2405.    202229    486299418.
## 9 01056                2917.    317385    925938817.
## 10 01057               2333.    129640    302408108.
## # i 386 more rows
```

```
# Anzahl an Erwerbstätigen für das Jahr 2021
erwerb_long %>%
  filter(nchar(Regionalschlüssel) == 5 & Jahr == 20
  select(-Jahr)
```

```
## # A tibble: 399 × 2
##   Regionalschlüssel   erw
##   <chr>             <dbl>
## 1 08111             532593
## 2 08115             226864
## 3 08116             280672
## 4 08117             119505
## 5 08118             268651
## 6 08119             205809
## 7 08121              98279
## 8 08125             180424
## 9 08126              73805
## 10 08127            115040
## # i 389 more rows
```

```
# Anzahl an Einwohner für das Jahr 2021
einwohner_long %>%
  filter(nchar(Regionalschlüssel) == 5 & Jahr == 20
  select(-Jahr)
```

```
## # A tibble: 399 × 2
##   Regionalschlüssel einwohner
##   <chr>             <dbl>
## 1 08111             628290
## 2 08115             393047
## 3 08116             533503
## 4 08117             258914
## 5 08118             544825
## 6 08119             427301
## 7 08121             126036
## 8 08125             347081
## 9 08126             113042
## 10 08127            198629
## # i 389 more rows
```

```
# Anzahl der Einwohner mit dem BIP verbinden um das
left_join(bip_long, einwohner_long, by=c("Regionalschlüssel"))
mutate(bip_pro_kopf = bip / einwohner) %>%
# BIP auf Landkreisebene
filter(nchar(Regionalschlüssel) == 5 & Jahr == 2010)
select(-c(Jahr, einwohner))
```

```
## # A tibble: 399 × 3
##   Regionalschlüssel      bip bip_pro_kopf
##   <chr>            <dbl>    <dbl>
## 1 08111            54983349000    87513.
## 2 08115            26471627000    67350.
## 3 08116            23940184000    44874.
## 4 08117             8665366000    33468.
## 5 08118            24894174000    45692.
## 6 08119            16119439000    37724.
## 7 08121             7598902000    60292.
## 8 08125            21144932000    60922.
## 9 08126            6241990000    55218.
## 10 08127           9549854000    48079.
## # i 389 more rows
```



```
# Datensätze zusammenführen

# Basisdatensatz -> Arbeitslosenzahlen pro Landkreis
# Namen der Bundesländer zumergen
alo_landkreis %>%
  mutate(bundesland = str_extract(Regionalschlüssel, "(\\w+\\.\\w+\\.\\w+)"))
  left_join(., schulden_kombi, by = "Regionalschlüssel")
  left_join(., bip_kombi, by = "Regionalschlüssel")
# Zahl der Erwerbstätigen zumergen
  left_join(., erwerb_kombi, by = "Regionalschlüssel")
gesamtdaten

#saveRDS(gesamtdaten, "data/gesamtdaten.rds") #save
#saveRDS(schulden_bereinigt, "data/schulden_bereinigt.rds") #
#saveRDS(bip_zeitreihe, "data/bip_zeitreihe.rds") #
```

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.

- ✚ 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.