Case Study für Deutschland

Alexander Rieber - 27 April 2020

Angenommen Sie haben einen Cousin in Spanien mit dem Sie regen Kontakt haben und den Sie auch immer wieder besuchen.

Sie kommen bei einem Besuch in Spanien auf die aktuelle Lage in seinem Heimatland zu sprechen. Ihr Cousin berichtet über die sehr hohe Arbeitslosigkeit, insbesondere Jugendarbeitslosigkeit in seinem Land. Er behauptet, dass Sie dies nicht nachvollziehen könnten, da es in Deutschland praktisch keine Arbeitslosigkeit gibt. Daheim angekommen schauen Sie sich die Daten zur Arbeitslosigkeit im Euro-Raum an (was Sie auch im 2. - 4. RTutor Problem Set machen) und sehen in der Tat, dass Deutschland eine der niedrigsten Arbeitslosenquoten aller Euro-Länder hat und Spanien deutlich höhere Arbeitslosenquoten aufweist. Doch hat ihr Cousin recht, wenn er davon spricht, dass Deutschland keine Arbeitslosigkeit kennt? Gilt dies für alle Regionen in Deutschland, oder gibt es auch in Deutschland Regionen mit hohen Arbeitslosenquoten? Wenn Sie regionale Unterschiede finden, welche Gründe könnte dies haben?

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

Ziele der Case Study

Diese Case-Study besteht aus mehreren Teilen und wird Sie durch die komplette Vorlesung als konkretes Anschauungsobjekt begleiten. Hierbei dient die Case-Study hauptsächlich dazu, ihnen an einem konkreten und umfangreichen Beispiel die Kenntnisse für eine erfolgreiche Projektarbeit zu vermitteln und diese Kenntnisse zu vertiefen. Natürlich können Sie die Case-Study auch als Referenz heranziehen, wenn Sie ihre eigene Projektarbeit anfertigen. Das Design des HTML-Outputs ist diesem Blogeintrag nachempfunden.

Ersten Teil der Case Study

Im ersten Teil der Case Study werden Sie ihre Kenntnisse aus der Vorlesungseinheit zum Datenhändling, d.h. alles bzgl. Daten einlesen, bearbeiten und in eine geeignete Form bringen, direkt auf Daten zur Arbeitslosenstatistik, dem BIP und der Verschuldung einzelner Landkreise bzw. Gemeinden anwenden. Ziele des ersten Teils der Case Study:

- Zuverlässig Datenquellen für eine Fragestellung ausfinding machen und diese automatisiert herunterladen
- Das technische Verständnis wie Daten in R eingelesen und bearbeitet werden können. Hierbei lesen Sie relativ feingranulare Informationen zum Arbeitsmarkt und zur gesamtwirtschaftlichen Lage in Deutschland ein und bearbeiten diese in R
- Wissen, wie verschiedene Datensätze miteinander verknüpft werden können

Ergänzend zu den unterschiedlichen Regionen innerhalb Deutschlands werden Sie im 2. RTutor Problem Sets Kennzahlen zu verschiedenen Ländern der europäischen Union untersuchen und die regionalen Unterschiede innerhalb der europäischen Union erleben. 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.

Zweiter Teil der Case Study

Im zweiten Teil der Case Study werden Sie die eingelesenen und aufgearbeiten Daten aus Teil 1 deskriptiv untersuchen. Hierbei erhalten Sie einen Eindruck von den Daten und können mögliche Zusammenhänge entdecken, indem Sie unterschiedliche Informationen visualisieren und auch in Tabellenform auswerten. Ziele des zweiten Teils der Case Study:

- Daten visualisieren und Zusämmenhänge grafisch veranschaulichen
- Deskriptive Analysen mittels Korrelationstabellen und deskriptiven Tabellen anfertigen
- Das Verständnis wie Sie ihre Informationen zu bestimmten Fragestellungen möglichst effektiv aufbereiten
- Interaktive Grafiken erstellen

Sie erhalten durch deskriptive Analysen einen sehr guten Eindruck von den regionalen Unterschieden innerhalb Deutschlands. Das begleitende 3. RTutor Problem Set gibt ihnen einen sehr guten Eindruck davon, wie die Unterschiede zwischen den einzelnen Ländern auf europäischer Ebene aussehen.

Dritter Teil der Case Study

Im dritten Teil der Case Study untersuchen Sie mögliche Gründe für die regionalen Unterschiede innerhalb Deutschlands. Mit den ihnen zur Verfügung stehenden Daten zum BIP und der Verschuldung der einzelnen Landkreise wollen Sie die Arbeitslosenquoten in den einzelnen Regionen Deutschlands erklären. Ziele des dritten Teils der Case Study:

- Regressionen in R durchführen
- Interpretation von Regressionskoeffizienten

Sie lernen, wie Sie eine lineare Regression dazu Nutzen können um mögliche Zusammenhängen zwischen der Arbeitslosigkeit und anderen Faktoren näher zu beleuchten. Jedoch lernen Sie auch die Grenzen der lineraren Regression kennen, insbesondere im Hinblick auf die Interpetation der Koeffizienten der Regression. Ergänzend hierzu erhalten Sie im 4. RTutor Problem Set Einblicke in die Zusammenhänge verschiedener gesamtwirtschaftlicher Faktoren und der Arbeitslosigkeit in den einzelnen Ländern der europäischen Union. Im 5. RTutor Problem Set werden Sie zusätzlich erfahren, welche Möglichkeiten wir in den Wirtschaftswissenschaften haben, um kausale Schlüsse ziehen zu können.

Daten beschaffen

Wir wollen uns in diesem Tutorial mit der Pro-Kopf Verschuldung, der Arbeitslosigkeit und dem BIP in einzelnen Regionen in Deutschland beschäftigen und hier mögliche regionale Unterschiede aufdecken.

Im Ersten Schritt ist es wichtig sich zu überlegen, woher Sie ihre Datensätze beziehen. Um makroökonomische Informationen zum BIP oder der Arbeitslosigkeit zu erhalten empfiehlt es sich immer auf die Seiten des Statistischen Bundesamtes oder der Bundesagentur für Arbeit zu schauen. Hier finden Sie z.B. Quartalsinformationen zu BIP und Arbeitslosigkeit für ganz Deutschland.

In dieser Case-Study wollen wir jedoch etwas feingranularere Informationenen sammeln, und zwar auf Landkreis-, Verwaltungs-, bzw. Gemeindeebene.

Uns interessieren die **Pro-Kopf Verschulung**, **Arbeitslosigkeit** und das **BIP**.

- 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.

Nötige Pakete laden

Bevor wir mit der Analyse starten sollten wir einige Pakete in R laden, welche wir später verwenden möchten, da sie uns bei der Analyse unterstützen können. Dies geschieht mit dem `library()` Befehl.

(**Alternative:** Vor jeden Befehl das dazugehörige Paket schreiben, d.h. statt `read_xlsx` könnten wir auch `readxl::read_xlsx`. Jedoch wollen wir dies in dem Projektkurs nicht tun.)

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

Daten herunterladen

Mit den Befehlen aus dem `readxl` und `readr` Paketen könnten Sie direkt URLs einlesen, wenn sich dahinter Text,- bzw. Excel Datei verbergen, was bei uns der Fall ist. Allerdings sollten wir davon nur selten Gebrauch machen, denn es könnte immer sein das die Daten im Internet modifiziert oder unter der vorherigen URL nicht mehr auffindbar sind. Daher wollen wir die gewünschten Daten, welche wir zur Analyse benötigen immer in einem Unterordner `data` abspeichern und dann aus diesem Ordner einlesen. So stellen wir sicher, dass wir immer auf die Daten zurückgreifen können, auch wenn diese aus dem Netz gelöscht oder modifiziert werden.

Daten können innerhalb von R mit dem Befehl `download.file()` heruntergeladen werden:

```
# Zuerst sollten Sie prüfen ob der Unterordner "data" bereits bei ihnen exisitert, und falls er nicht es
# Dies können Sie beispielsweise mit dem folgenden Befehl machen, wenn der Ordner schon existiert wird

dir.create(file.path(".", "data"))
```

```
## Warning in dir.create(file.path(".", "data")): './data' existiert bereits
```

```
# Durch die if-Bedingung prüfen Sie, ob die Datei bereits im "data"-Ordner vorhanden ist

# Verschuldung
if (!file.exists("./data/Schulden_2017.xlsx")){
  download.file("https://www.statistikportal.de/sites/default/files/2018-11/Schulden_2018.xlsx", "./data
}

# Arbeitslose
if (!file.exists("./data/Arbeitslose_2017.xlsx.zip")){
  download.file("https://statistik.arbeitsagentur.de/Statistikdaten/Detail/201712/iiia4/gem-jz/gem-jz-d
}

# BIP pro Gemeinde
if (!file.exists("./data/BIP_2017.xlsx.zip")){
  download.file("https://www.statistik-bw.de/VGRdL/tbls/RV2014/R2B1.zip", "./data/BIP_2017.xlsx.zip")
}
```

Die Daten zur Verschuldung wollen wir absichtlich nicht unter "Schulden_2018.xlsx" abspeichern, sondern unter "Schulden_2017.xlsx", da die Tabelle zwar in 2018 generiert wurde, sich aber auf das Jahr 2017 bezieht.

Die `if`-Bedingung prüft ob der angegebene Datensatz bereits in unserem "data" Ordner enthalten ist. Wenn dies der Fall ist, so werden sie nicht mehr erneut heruntergeladen.

Daten einlesen

Im nächsten Schritt sollten wir die Daten in R einlesen. Beim Download haben wir schon an den URLs und auch an den heruntergeladenen Dateien gesehen, dass es sich bei zwei Downloads um ZIP-Archive (BIP und

Anzahl an Arbeitslosen) handelt. Diese ZIP-Archive können wir mittels R extrahieren, diese anschließend einlesen und die extrahierte Datei wieder löschen. Dies hat den Vorteil, dass Sie ihre Dateien platzsparend auf ihrer Festplatte abspeichern und nur bei Bedarf entsprechend entpacken.

Anzahl an Arbeitslosen

Im ersten Schritt wollen wir uns mit den Daten zu den Arbeitslosen beschäftigen und die in dem ZIP-Archiv enthaltenen Dateien in R einlesen.

ZIP-Archiv entpacken

```
# Öffnen des ZIP-Archivs
alo_name <- as.character(unzip("./data/Arbeitslose_2017.xlsx.zip", list = TRUE)$Name)
unzip("./data/Arbeitslose_2017.xlsx.zip", alo_name)
```

Ok, nun haben wir die Daten entzipped und sehen, dass es sich um eine Excel-Datei handelt. Doch diese hat sehr viele unterschiedliche Tabellenblätter. Wie wissen wir, welches Tabellenblatt für uns von Interesse ist?

Dies können wir zum Einen mit `excel_sheets` herausfinden, wenn die Tabellenblätter gut benannt sind:

```
excel_sheets(alo_name)
```

```
##  [1] "Deckblatt"           "Impressum"           "Inhalt"
##  [4] "Hinweis"             "5"                   "6"
##  [7] "7"                   "8"                   "9"
## [10] "Statistik-Infoseite"
```

Da die Tabellenblätter jedoch nicht unbedingt vielsagend beschriftet sind sollten wir das Tabellenblatt "Inhalt" einlesen und uns dieses anschauen. Eventuell werden wir hier schlauer.

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

```
## # A tibble: 15 x 4
##    ...1                  ...2          ...3  `Arbeitslose nach Gemeinden     Ja~
##    <chr>                 <chr>         <lgl> <chr>
##  1 <NA>                  <NA>          NA    <NA>
##  2 Inhaltsverzeichnis    <NA>          NA    <NA>
##  3 <NA>                  <NA>          NA    <NA>
##  4 Berichtsmonat: Jahre~ <NA>          NA    <NA>
##  5 <NA>                  <NA>          NA    <NA>
##  6 Arbeitslose nach Gem~ <NA>          NA    <NA>
##  7 <NA>                  Impressum     NA    2
##  8 <NA>                  Inhaltsverze~ NA    3
##  9 <NA>                  Hinweise      NA    4
## 10 Zugang                <NA>          NA    <NA>
## 11 <NA>                  Insgesamt     NA    5
## 12 Bestand               <NA>          NA    <NA>
## 13 <NA>                  Insgesamt     NA    6
## 14 <NA>                  Männer        NA    7
## 15 <NA>                  Frauen        NA    8
```

Hier erhalten wir einen Überblick über die Tabellenblätter und wo welche Informationen abgespeichert sind. Da wir uns für den Bestand an Arbeitslosen interessieren und hier nicht nach Frauen und Männern unterscheiden möchten, ist das Tabellenblatt `6` für uns das richtige.

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

Entpackte Datei einlesen

Nun wissen wir, welches Tabellenblatt die für uns wichtige Information enthält:

```
alo <- read_xlsx(alo_name, sheet="6")

head(alo,10)
```

```
## # A tibble: 10 x 32
##    ...1  ...2  ...3  ...4  ...5  ...6  ...7  ...8  ...9  ...10 ...11 ...12 ...13
##    <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
##  1 Best~  <NA>  <NA>  <NA> <NA>   <NA>  <NA> <NA>   <NA> <NA>   <NA>  <NA>  <NA>
##  2 Länd~  <NA>  <NA>  <NA> <NA>   <NA>  <NA> <NA>   <NA> <NA>   <NA>  <NA>  <NA>
##  3 Beri~  <NA>  <NA>  <NA> <NA>   <NA>  <NA> <NA>   <NA> <NA>   <NA>  <NA>  <NA>
##  4 <NA>   <NA>  <NA>  <NA> <NA>   <NA>  <NA> <NA>   <NA> <NA>   <NA>  <NA>  <NA>
##  5 <NA>   <NA> "Rec~  <NA> <NA>   <NA>  <NA> <NA>   <NA> <NA>   <NA>  <NA> "Rec~
##  6 <NA>  "Bun~ "ins~ "aus~ <NA>   <NA>  <NA> daru~ "aus~ <NA>   <NA>  <NA> "ins~
##  7 <NA>  "Reg~  <NA> "unt~ unte~ "50 ~ "55 ~ <NA>  "unt~ unte~ "50 ~ "55 ~  <NA>
##  8 Schl~ "Gem~ "1"   "2"   3     "4"   "5"   6     "7"   8     "9"   "10"  "11"
##  9 <NA>  "Deu~ "253~ "478~ 2306~ "839~ "529~ 6552~ "140~ 71133 "136~ "732~ "855~
## 10 01    "Sch~ "924~ "207~ 9703  "301~ "187~ 18688 "437" 2297  "347~ "190~ "309~
## # ... with 19 more variables: ...14 <chr>, ...15 <chr>, ...16 <chr>,
## #   ...17 <chr>, ...18 <chr>, ...19 <chr>, ...20 <chr>, ...21 <chr>,
## #   ...22 <chr>, ...23 <chr>, ...24 <chr>, ...25 <chr>, ...26 <chr>,
## #   ...27 <chr>, ...28 <chr>, ...29 <chr>, ...30 <chr>, ...31 <chr>,
## #   `Arbeitslose nach Gemeinden Jahreszahlen 2017` <chr>
```

Ok. Hier ist es wohl nicht vorteilhaft von der ersten Zeile ab die Informationen aus dem Tabellenblatt einzulesen. So wie es aussieht sind in den ersten 3 Zeilen Informationen zum Tabellenblatt und dem Berichtsjahr enthalten, dann kommt eine leere Zeile und dann kommen die eigentlichen Spaltenbeschriftungen. Diese sind dann jedoch wiederum in 4 Zeilen unterteilt. Was sind denn hier nun die Spaltenüberschriften, d.h. die Variablennamen im Datensatz?

Spezifizieren welche Spalten eingelesen werden sollen

Hierzu überlegen wir uns folgendes:

- Welche Information benötigen wir aus der Tabelle
  - Die Anzahl aller Arbeitslosen pro Gemeinde (d.h. SGB II und III gemeinsam)
  - 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)
- Wie können wir die von uns benötigte Information möglichst einfach extrahieren

In unserem Fall benötigen wir die Information zur Anzahl aller Arbeitslosen pro Gemeinde, d.h. uns interessiert Spalte `...3`. Weiterhin benötigen wir den `Schlüssel` um diese Datenquelle später mit anderen Datenquellen verbinden zu können. Außerdem wäre der Name der Gemeinde noch eine ganz nett Information. Der einfachste Weg an diese Information zu gelangen ist die ersten acht Zeilen abzuschneiden und die Daten erst ab dort einzulesen. Anschließend behalten wir nur die ersten 3 Spalten, da uns nur diese interessieren.

Das wollen wir nun machen:

```
# Daten einlesen von Tabellenblatt 6, ohne die ersten 8 Zeilen
alo <- read_xlsx(alo_name, sheet = "6", skip = 8)
```

```r
# Die entzippte Datei wieder löschen
unlink(alo_name)

# Nun beschränken wir uns auf die ersten drei Spalten und benennen die Spalte `Schlüssel` noch in "Regi
# Weiterhin löschen wir alle Zeilen, für die "alo" auf NA gesetzt ist und die erste Zeile, die alle Arb
# Wir speichern den Datensatz als `data_alo` ab
data_alo <- alo %>%
  select(c(`Schlüssel`, Gemeinde, `1`)) %>%
  rename(Regionalschluessel = `Schlüssel`,
         alo = `1`) %>%
  filter(!is.na(alo) & Gemeinde!= "Deutschland")
```

Konsistenzcheck

Nun sollten wir noch die Daten auf Konsistenz prüfen. D.h. machen die Angaben Sinn und sind die Daten in sich konsistent? Hierfür sollten wir zum Einen externe Datenquellen untersuchen und zum Anderen die Daten intern auf konsistenz prüfen.

Wir haben hier sehr feingranulare Informationen über die Arbeitslosenzahl in 2017 vorliegen, jedoch können wir die Daten auch auf eine höhere Ebene aggregieren und damit leicht mit anderen Quellen vergleichen. Dies wollen wir hier tun:

- Zunächst lassen wir uns die Anzahl an Arbeitslosen für jedes Bundesland in 2017 ausgeben. In unserem Datensatz sind dies alle Datenpunkte mit einem zweistelligen `Regionalschluessel`. Wir müssen hier beachten, dass die `Regionalschluessel` in der Klasse `character` vorliegen, d.h. als Strings und nicht als Zahl. Deshalb können wir die Anzahl an "Buchstaben" für jeden `Regionalschluessel` zählen. Dies geschieht über den Befehl `nchar()` (number of characters)
- Nun sollten wir eine andere Datenquelle heranziehen und die Informationen gegenchecken. Bspw. könnten wir die Anzahl der Arbeitslosen für das Jahr 2017 unterteilt nach Ländern heranziehen. (Tabellenblatt 8)

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

check_alo_bundesland
```

```
## # A tibble: 16 x 3
##    bundesland Gemeinde                 alo
##    <chr>      <chr>                  <dbl>
##  1 01         Schleswig-Holstein      92434
##  2 02         Hamburg                 69248
##  3 03         Niedersachsen          244260
##  4 04         Bremen                  35687
##  5 05         Nordrhein-Westfalen    701219
##  6 06         Hessen                 166286
##  7 07         Rheinland-Pfalz        106299
##  8 08         Baden-Württemberg      212837
##  9 09         Bayern                 231353
## 10 10         Saarland                34672
## 11 11         Berlin                 168991
## 12 12         Brandenburg             92648
## 13 13         Mecklenburg-Vorpommern  70982
## 14 14         Sachsen                140348
## 15 15         Sachsen-Anhalt          96960
```

```
## 16 16         Thüringen               68614
```

Wenn wir die Überprüfung mit der anderen Tabelle der Bundesagentur für Arbeit machen, dann sind beide Datenreihen identisch (lediglich Hessen weicht um eine Person ab).

Nun wollen wir noch die interne Konsistenz überprüfen. Hierfür berechnen wir die Anzahl an Arbeitslosen für jedes Bundesland als Summe der Arbeitslosen einer jeden 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)
```

Um einen besseren Überblick zu erhalten können wir unsere berechneten und die von der Agentur für Arbeit angegebenen Werte miteinander verbinden und die Differenz zwischen den beiden Tabellen berechnen:

```
check_consitency <- left_join(check_alo_bundesland, alo_bundesland, by = "bundesland")

check_consitency <- check_consitency %>%
  mutate(diff = alo - total_alo)

check_consitency
```

```
## # A tibble: 16 x 5
##    bundesland Gemeinde                 alo total_alo  diff
##    <chr>      <chr>                  <dbl>     <dbl> <dbl>
##  1 01         Schleswig-Holstein     92434     92449   -15
##  2 02         Hamburg                69248     69248     0
##  3 03         Niedersachsen         244260    244277   -17
##  4 04         Bremen                 35687     35687     0
##  5 05         Nordrhein-Westfalen   701219    701212     7
##  6 06         Hessen                166286    166296   -10
##  7 07         Rheinland-Pfalz       106299    106287    12
##  8 08         Baden-Württemberg     212837    212835     2
##  9 09         Bayern                231353    231355    -2
## 10 10         Saarland               34672     34675    -3
## 11 11         Berlin                168991    168991     0
## 12 12         Brandenburg            92648     92644     4
## 13 13         Mecklenburg-Vorpommern 70982     70989    -7
## 14 14         Sachsen               140348    140348     0
## 15 15         Sachsen-Anhalt         96960     96960     0
## 16 16         Thüringen              68614     68609     5
```

Es scheint so, als ob es tatsächlich kleinere Differenzen zwischen den von der Agentur für Arbeit auf Bundeslandebene postulierten Zahlen zur Arbeitslosigkeit und den auf Basis der einzelnen Gemeinden berechneten Werten gibt. Diese Unterschiede sind jedoch marginal (Werte zwischen [-17,12]) und werden von uns im folgenden nicht näher untersucht.

Pro-Kopf Verschuldung

Der nächste Datensatz beinhaltet die Pro-Kopf-Verschuldung der deutschen Gemeinden. Hier handelt es sich wieder um Querschnittsdaten auf Gemeindeebene aus dem Jahr 2017.

Diesen Datensatz können wir von der Homepage des Statistischen Bundesamtes direkt als Excel-Tabelle herunterladen und müssen kein ZIP-Archiv entpacken. Allerdings sehen wir sehr schnell, das auch dieser Datensatz seine Tücken beim Einlesen bereithält, insbesondere wenn wir schauen, welche Tabellenblätter für unsere Analyse relevant sind:

```
excel_sheets("./data/Schulden_2017.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

Nun sind nicht mehr alle Informationen in **einem Tabellenblatt** enthalten, sondern jedes Bundesland hat sein eigenes Tabellenblatt bekommen. Sprich, wir müssen eine Möglichkeit finden alle Tabellenblätter nacheinander einzulesen und zu verarbeiten.

Dies wollen wir mit einer `for`-Schleife lösen, doch zuerst schauen wir uns an, welche Informationen wir aus den Tabellenblättern benötigen:

```
sh <- read_xlsx("./data/Schulden_2017.xlsx", sheet = "SH")
head(sh,20)
```

```
## # A tibble: 20 x 16
##    `Zurück zum Inh~ ...2  ...3  ...4  ...5  ...6  ...7  ...8  ...9  ...10 ...11
##    <chr>            <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
##  1 <NA>             <NA>  <NA>   <NA> <NA>   <NA> <NA>  <NA>  <NA>  <NA>  <NA>
##  2 <NA>             <NA>  <NA>   <NA> <NA>   <NA> <NA>  <NA>  <NA>  <NA>  <NA>
##  3 "Tabelle 1:    ~ <NA>  <NA>   <NA> <NA>   <NA> <NA>  <NA>  <NA>  <NA>  <NA>
##  4 "nach Höhe der ~ <NA>  <NA>   <NA> <NA>   <NA> <NA>  <NA>  <NA>  <NA>  <NA>
##  5 "Regional-\r\ns~ Geme~ Verw~ "Ein~ Schu~ "Sch~ Schu~ <NA>  <NA>  <NA>  <NA>
##  6 <NA>             <NA>  <NA>   <NA> <NA>   <NA> zusa~ Schu~ ante~ <NA>  <NA>
##  7 <NA>             <NA>  <NA>   <NA> <NA>   <NA> <NA>  <NA>  zusa~ davo~ <NA>
##  8 <NA>             <NA>  <NA>   <NA> <NA>   <NA> <NA>  <NA>  <NA>  100%  50% ~
##  9 <NA>             <NA>  <NA>   <NA> EUR    <NA> <NA>  <NA>  <NA>  <NA>  <NA>
## 10 <NA>             <NA>  <NA>   <NA> 1      "2"   3     4     5     6     7
## 11 "010010000000"   Flen~ krei~ "877~ 5082~ "579~ 2979~ 1126~ 1853~ 1822~ 2906~
## 12 "010020000000"   Kiel~ krei~ "247~ 9488~ "383~ 5632~ 5626~ 5701~ 4437~ 0
## 13 "010030000000"   Lübe~ krei~ "216~ 1206~ "556~ 6014~ 5938~ 7606~ 7027~ 0
## 14 "010040000000"   Neum~ krei~ "787~ 4260~ "540~ 1233~ 1152~ 8176~ 8103~ 0
## 15 "01051"          Krei~ Krei~ "{13~ 6517~ "487~ 4309~ 4302~ 72975 0     0
## 16 "010510011011"   Brun~ amts~ "127~ 3493~ "273~ 5707~ 5195~ 5125~ 0     0
## 17 "010510044044"   Heid~ amts~ "215~ 3705~ "172~ 2237~ 1999~ 2375~ 0     1863~
## 18 "010515163"      Amts~ Amts~ "{15~ 1047~ "66.~ 1047~ 9742~ 72975 0     0
## 19 "010515163003"   Aver~ amts~ "576" 1197~ "207~ 7796~ 6325~ 1471~ 0     0
## 20 "010515163010"   Bric~ amts~ "219" 4962~ "226~ 3336~ 1865~ 1471~ 0     0
## # ... with 5 more variables: ...12 <chr>, ...13 <chr>, ...14 <chr>, `Zurück zum
## #   Inhalt...15` <chr>, ...16 <chr>
```

Für uns wichtig sind die Infos bzgl. des "Regionalschlüssels", der "Gemeindename", die "Einwohner" und die "Schuldes des öffentlichen Bereichs insgesamt". Zur Überprüfung unserer Ergebnisse nehmen wir noch die "Schulden je Einwohner" mit in unseren Datensatz auf, d.h. die ersten sechs Spalten. Weiterhin stehen unsere Variablenbezeichnungen in Zeile 5, d.h. wir ignorieren die ersten 4 Zeilen beim 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 1-6
schulden_individuell <- read_xlsx("./data/Schulden_2017.xlsx", sheet = "SH", skip = 5)[1:6]
# Umbenennen der ersten 6 Spalten
colnames(schulden_individuell) <- c("Regionalschluessel", "Gemeinde",
                                    "Verwaltungsform", "Einwohner", "Schulden_gesamt", "Schulden_pro_kop

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

Ok, nun haben wir die Daten für Schleswig-Holstein eingelesen und können mit einer `for`-Schleife alle weiteren Bundesländer (Tabellenblätter) in der gleichen Form durchgehen:

```
# Daten mit for-Schleife einlesen (Struktur gleich wie im vorherigen Chunk)
sheet_names <- excel_sheets("./data/Schulden_2017.xlsx")
# Einlesen der Tabellenblätter 7-18 (alle Bundesländer)
sheet_read <- sheet_names[7:18]
for (i in 1:length(sheet_read)){
  tmp <- read_xlsx("./data/Schulden_2017.xlsx", sheet = sheet_read[i], skip = 5)[1:6]
  tmp$Bundesland <- sheet_read[i]
  colnames(tmp) <- c("Regionalschluessel", "Gemeinde", "Verwaltungsform",
                     "Einwohner", "Schulden_gesamt", "Schulden_pro_kopf", "Bundesland")
  # Daten aller weiteren Tabellenblätter unter den aktuellen Datensatz anheften
  schulden_individuell <- bind_rows(schulden_individuell, tmp)
}
```

Variablen umformen

```
head(schulden_individuell,30)
```

```
## # A tibble: 30 x 7
##    Regionalschlues~ 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     Flensbu~ kreisfreie Sta~ 87770     508281539
##  7 010020000000     Kiel, L~ kreisfreie Sta~ 247135    948848421
##  8 010030000000     Lübeck,~ kreisfreie Sta~ 216739    1206620094
##  9 010040000000     Neumüns~ kreisfreie Sta~ 78759     426019276
## 10 01051            Kreisve~ Kreisverwaltung {133 684} 65179097
## # ... with 20 more rows, and 2 more variables: Schulden_pro_kopf <chr>,
## #   Bundesland <chr>
```

Wir sehen, es gibt immer noch einige Probleme:

- Die Werte unserer Variablen stehen nicht direkt unter dem Variablennamen, das ist für uns nicht optimal und ist der Anordnung in der Excel Datei geschuldet.

  - Dies können wir am einfachsten bereinigen indem wir alle `NA`s im Regionalschlüssel entfernen (kein Regionalschlüssel bedeutet keine Zuordnung zu einer Region und damit für uns nicht nachvollziehbar).
- Die Variablen "Einwohner", "Schulden_gesamt" und "Schulden_pro_Kopf" sind alle als `character` hinterlegt (`<chr>` unter dem Variablennamen in der vorherigen Tabelle), wir wollen diese jedoch in numerischer Form um Berechnungen durchführen zu können
  - Der Grund dür die Klasse `character` kann z.B. in Zeile 28 beobachtet werden. Hier wurden geschweifte Klammern verwendet um die Summe aller Variablen eines Amtsgebiets, Landkreis, Region etc. zu kennzeichnen.
  - Im ersten Schritt wollen wir diese Summen einfach ignorieren da wir die jeweiligen Summen auch selbst berechnen können.

Anschließend wollen wir noch den `landkreis` als die ersten 5 Zeichen im Regionalschlüssel definieren.

```
# Die Daten wurden noch nicht schön eingelesen, in der Excel Tabelle
# waren die Variablennamen über mehrere Reihen gezogen, dies müssen wir noch ausgleichen
schulden_bereinigt <- schulden_individuell %>%
  filter(!is.na(Regionalschluessel)) %>%
  mutate(Schulden_gesamt = as.numeric(Schulden_gesamt),
         Einwohner = as.numeric(Einwohner),
         Schulden_pro_kopf = as.numeric(Schulden_pro_kopf)) %>%
  mutate(landkreis = str_extract(Regionalschluessel, "^.{5}"))
```

Es wurden immer noch einige `NA`s erzeugt. Diese wollen wir uns noch näher anschauen:

```
filter(schulden_bereinigt, is.na(Einwohner))
```

```
## # A tibble: 2,400 x 8
##    Regionalschlues~ Gemeinde Verwaltungsform Einwohner Schulden_gesamt
##    <chr>            <chr>    <chr>               <dbl>           <dbl>
##  1 01051            Kreisve~ Kreisverwaltung        NA        65179097
##  2 010515163        Amtsver~ Amtsverwaltung         NA         1047175
##  3 010515163_Summe~ Burg-Sa~ Amtsgebiet             NA              NA
##  4 010515166        Amtsver~ Amtsverwaltung         NA         3654294
##  5 010515166_Summe~ Marne-N~ Amtsgebiet             NA              NA
##  6 010515169        Amtsver~ Amtsverwaltung         NA         8632641
##  7 010515169_Summe~ Eider    Amtsgebiet             NA              NA
##  8 010515172        Amtsver~ Amtsverwaltung         NA         1867573
##  9 010515172_Summe~ Heider ~ Amtsgebiet             NA              NA
## 10 010515175        Amtsver~ Amtsverwaltung         NA          896103
## # ... with 2,390 more rows, and 3 more variables: Schulden_pro_kopf <dbl>,
## #   Bundesland <chr>, landkreis <chr>
```

Wir müssen wohl noch mehr ausschließen als nur `NA`s beim Regionalschlüssel, insgesamt 2400 Einträge bei denen die Variable "Einwohner" nicht vorhanden ist. Wir hatten bereits gesehen, dass die Summe aller Einwohner eines Landkreisen mit `{ Zahl }` in der Excel-Datei hervorgehoben wird. Wenn wir hier in R eine Typumwandlung erzwingen, dann kann R mit den `{}` nichts anfangen und gibt uns deshalb ein `NA` aus. Wir hier alle Einträge bei denen die Einwohner ein `NA` stehen haben löschen, da wir die Daten selbst auf Basis der Informationen zu den Gemeinden des Landkreises berechnen können.

```
schulden_bereinigt <- schulden_bereinigt %>%
  filter( !is.na( Einwohner ) )
```

Konsistenzcheck

Berechnung der Schulden pro Kopf von Hand

Um die interne Validität unserer Daten beurteilen zu können wollen wir im ersten Schritt eine Variable `Schulden_pro_Kopf_new` generieren, welche die `Schulden_pro_Kopf` von Hand berechnet. Wie schon im Abschnitt Variablen umformen erwähnt, müssen wir hierfür jedoch erst folgendes beachten, bevor wir Berechnungen durchführen können: - Wir müssen die geschweiften Klammern entfernen (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 `NA`s 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)) %>%
  mutate(landkreis = str_extract(Regionalschluessel, "^.{5}"),
         differenz = Schulden_pro_kopf - Schulden_pro_kopf_new)
```

Nun können wir uns anschauen, ob die von uns berechneten und die vom Statistischen Bundesamt angegebenen Werte zu den "Schulden_pro_Kopf" signifikant voneinander abweichen:

```
# range(schulden_consistency$differenz)
# oder schöne skim
skim_without_charts(schulden_consistency$differenz)
```

Table 1: Data summary

Name	schulden_consistency$diff...
Number of rows	13421
Number of columns	1
Column type frequency:	
numeric	1
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
data	12	1	0	0.09	-0.49	0	0	0	0.5

Die Differenzen liegen zwischen +/- 50 Cent und können vermutlich auf Rundungsfehler zurückgeführt werden. D.h. hier können wir die vom statistischen Bundesamt herausgegebenen Berechnungen auf Gemeindeebene verifizieren. Die angegebenen 12 nicht verfügbaren Werte sollten wir noch untersuchen, da hier keine Differenz

berechnet werden konnte:

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

```
## # A tibble: 12 x 11
##    Regionalschlues~ Gemeinde Verwaltungsform Einwohner Schulden_gesamt
##    <chr>            <chr>    <chr>           <chr>               <dbl>
##  1 033519501_Summe~ Lohheid~ Samtgemeindege~ {768}                  NA
##  2 033589501_Summe~ Osterhe~ Samtgemeindege~ {2 884}                NA
##  3 052              Landsch~ Landschaftsver~ X                736700545
##  4 056              Landsch~ Landschaftsver~ X                671832895
##  5 058              Regiona~ Kommunalverband X                132203729
##  6 067              Verwalt~ Landeswohlfahr~ X                 30001419
##  7 074              Bezirks~ Bezirksverband  X                140470575
##  8 081a             Landesw~ Landeswohlfahr~ X                  5323363
##  9 081b             Kommuna~ Landeswohlfahr~ X                        0
## 10 091785127_Summe~ Allersh~ Verwaltungsgem~ {7 190}                NA
## 11 095725512_Summe~ Auracht~ Verwaltungsgem~ {4 368}                NA
## 12 144              Kommuna~ Landeswohlfahr~ X                 10043701
## # ... with 6 more variables: Schulden_pro_kopf <dbl>, Bundesland <chr>,
## #   Einwohner_num <dbl>, Schulden_pro_kopf_new <dbl>, landkreis <chr>,
## #   differenz <dbl>
```

Hier gibt es meist keine Angaben zu den Einwohnern, oder keine Angaben zu den Schulden. Der Großteil der angegebenen Verwaltungsformen sind Verbände und interessieren uns nicht. Daher müssen wir den fehlenden Werten keine gesonderte Beachtung schenken.

Vergleich der Schulden pro Kopf auf Landkreisebene

In einem weiteren Konsistenzcheck wollen wir die durchschnittliche Verschuldung pro Kopf auf Landkreisebene selbst berechnen und diese mit den vom Statistischen Bundesamt angegebenen Werten in der Tabelle abgleichen.

Hierfür entnehmen wir der Tabelle zuerst alle Informationen bzgl. Anzahl der "Einwohner", "Schulden_gesamt" und "Schulden_pro_Kopf" für die Landkreise. Im Datensatz sehen wir, dass die Regionalschluessel für Landkreise die Worte "Summe" und "Kreis" enthalten, wenn das Statistische Bundesamt die Daten auf Landkreisebene aggregiert. Das wollen wir im folgenden nutzen:

```
# Wir filtern alle Reihen heraus, welche "_Summe" oder "Kreis" im Regionalschlüssel aufweisen
# Anschließend berechnen wir die durchschnittliche Verschuldung auf Landkreisebene
avg_versch_kreis <- schulden_consistency %>%
  filter(str_detect(Regionalschluessel, "_Summe") & str_detect(Regionalschluessel, "Kreis")) %>%
  group_by(landkreis, Gemeinde) %>%
  summarise(avg_verschuldung = Schulden_pro_kopf, einwohner = Einwohner_num,
            Gesamtschuld = Schulden_gesamt) %>%
  arrange(desc(avg_verschuldung))

# Hier berechnen wir die Daten selbst
avg_versch_kreis_calc <- schulden_consistency %>%
  # Ersetze Einwohner_num mit 0 für alle Regionalschlüssel kleiner als 12
  mutate(Einwohner_num = ifelse(nchar(Regionalschluessel)<12,0, Einwohner_num)) %>%
  # Nur Gemeinden betrachten
  filter(nchar(Regionalschluessel)>=5 & str_detect(Regionalschluessel, "_Summe")==FALSE) %>%
  # Auf Landkreisebene gruppieren
  group_by(landkreis) %>%
  summarise(einwohner_calc = sum(Einwohner_num), Gesamtschuld_calc = sum(Schulden_gesamt),
```

```
            avg_verschuldung_calc = round(Gesamtschuld_calc/einwohner_calc,2)) %>%
  arrange(desc(avg_verschuldung_calc))

# Verbinde beide Datensätze und berechne ob es siginfikante Abweichungen zwischen
# den ausgegebenen und berechneten Werten gibt
new <- left_join(avg_versch_kreis, avg_versch_kreis_calc, by="landkreis") %>%
  mutate(differenz = avg_verschuldung - avg_verschuldung_calc) %>%
  arrange( desc(differenz) )
```

```
# Ergebnis anschauen
#range(new$differenz)
# oder mit skim
skim_without_charts(new$differenz)
```

Table 3: Data summary

Name	new$differenz
Number of rows	294
Number of columns	1
_____	
Column type frequency:	
numeric	1
_____	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
data	0	1	0.1	1.81	-0.49	-0.22	-0.01	0.23	30.58

Die Differenzen liegen auch hier zwischen +/- 50 Cent und können vermutlich auf Rundungsfehler zurückgeführt werden. D.h. hier können wir die vom statistischen Bundesamt herausgegebenen Berechnungen auf Landkreisebene verifizieren. Der Landkreis Plön bildet hier eine Ausnahme, dies liegt aber vermutlich an der Zuordnung einzelner Teilgemeinden zum Landkreis, welche sich in 2007 geändert hat (siehe Fußnote 2 im Excel-Tabellenblatt “SH”).

Bruttoinlandsprodukt

Im nächsten Schritt wollen wir uns die Daten zum Bruttoinlandsprodukt einzelner Landkreise anschauen und diese in R einlesen. Da diese wieder als ZIP-Archiv heruntergeladen wurden können wir unser vorheriges Schema auch auf diese Daten anwenden.

ZIP-Archiv entpacken und nur einzelne Spalten einlesen

Folgende Schritte wollen wir in einem Chunk erledigen:

- Betrachten der entzippten Daten
  - Tabellenblatt “1.1” ist für unsere Analyse ausschlaggebend (für das BIP)
  - Tabellenblatt “3.1” ist für die Anzahl an Erwerbstätigen ausschlaggebend
- Die ersten vier Zeilen benötigen wir nicht
- Die letzte Zeile enthält eine kurze Beschreibung die wir nicht benötigen -> 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`

```
#Anzahl der Erwerbstätigen
```

```
# Einlesen der Daten mit anschließendem aufräumen
bip_name <- as.character(unzip("./data/BIP_2017.xlsx.zip", list = TRUE)$Name)
unzip("./data/BIP_2017.xlsx.zip", bip_name)

# Blatt 1.1 einlesen und die ersten 4 Zeilen skippen
bip <- read_xlsx(bip_name, sheet="1.1", skip = 4)
erwerb <- read_xlsx(bip_name, sheet="3.1", skip = 4)
unlink(bip_name)

# Zeile löschen in der die `Lfd. Nr.` nicht nummerisch ist
# Zusätzliche Spalten löschen
bip_wide <- bip %>%
  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`)

# 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`)

head(bip_wide)
```

```
## # A tibble: 6 x 26
##   Regionalschlues~ `1992` `1994` `1995` `1996` `1997` `1998` `1999` `2000`
##   <chr>            <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>   <dbl>
## 1 08               25493~ 26274~ 27240~ 27745~ 28310~ 29167~ 30138~ 3.10e5
## 2 081              10985~ 11086~ 11474~ 11624~ 12040~ 12321~ 12714~ 1.30e5
## 3 08111            32014~ 30887~ 31479~ 31988~ 33589~ 32711~ 34128~ 3.43e4
## 4 08115            11977~ 11750~ 11867~ 12025~ 13820~ 13545~ 14308~ 1.38e4
## 5 08116            12212~ 12462~ 12748~ 13166~ 13291~ 13937~ 14189~ 1.44e4
## 6 08117            5119.~ 5264.~ 5546.~ 5746.7 5775.~ 5944.~ 6032.~ 6.12e3
## # ... with 17 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>
```

Dieser Datensatz ist ein sogenanntes Panel. In den vorherigen Datensätzen zur Anzahl der Arbeitslosen und der Pro-Kopf-Verschuldung hatten wir nur das Jahr 2017 gegeben. Nun haben wir die Entwicklung des BIP seit 1992 bis 2017 für alle Landkreise in Deutschland.

Daten in das `long`-Format überführen

Allerdings ist der Datensatz im `wide`-Format, d.h. nicht `tidy` und damit nicht so, wie wir ihn gerne hätten. Erinnern wir uns noch an die Bedingungen damit ein Datensätzen `tidy` ist?

Im nächsten Schritt wollen wir den Datensatz nun ins `long`-Format überführen und nutzen hierfür die Funktion `pivot_longer`:

```
bip_long <- pivot_longer(bip_wide, cols = c("1992":"2017") , names_to = "Jahr", values_to = "BIP")

# Produziert den folgenden Fehler:
# Fehler: No common type for `1992` <character> and `2000` <double>.
```

Leider ist es hier nicht möglich den Datensatz direkt in das `long`-Format zu überführen, insbesondere da die Klassen der Variablen 1992 bis 1999 `character` sind und ab 2000 dann `double`. Dies sagt uns die erscheinende Fehlermeldung:

Da wir wissen, dass das BIP normalerweise numerisch wiedergegeben wird, ist wohl die Klasse `double` korrekt und wir sollten die Spalten von 1992 bis 1999 entsprechend umformatieren.

```
#BIP von 1992 - 1999 umformen (als numerische Variable)
bip_double <- bip_wide %>%
  select(`1992`:`1999`) %>%
  mutate_if(is.character, as.double)

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

Wir bekommen hier eine Warnmeldung das `NA`s bei der Umwandlung erzeugt wurden. Derartige Warnungen sollten wir beachten und auf den Grund gehen. Nur so wissen wir, ob die Warnung für uns später unbeabsichtigte Auswirkungen hat.

Hierfür verbinden wir den neuen Datensatz `bip_double` mit unserem bisher bestehenden `bip_wide` und betrachten die Spalten in denen `bip_double` `NA`s enthält:

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

head(filter(bip_wide_test, is.na(`19921`)))
```

```
## # A tibble: 6 x 33
##   Regionalschlues~ `1992` `1994` `1995` `1996` `1997` `1998` `1999` `2000`
##   <chr>            <chr>  <chr>  <chr>  <chr>  <chr>  <chr>  <chr>   <dbl>
## 1 13003            .      .      .      .      .      .      .       4909.
## 2 13004            .      .      .      .      .      .      .       2542.
## 3 13071            .      .      .      .      .      .      .       5154.
## 4 13072            .      .      .      .      .      .      .       3529.
## 5 13073            .      .      .      .      .      .      .       3611.
## 6 13074            .      .      .      .      .      .      .       2363.
## # ... with 24 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>, `19921` <dbl>,
## #   `19941` <dbl>, `19951` <dbl>, `19961` <dbl>, `19971` <dbl>, `19981` <dbl>,
## #   `19991` <dbl>
```

Hier sehen wir bereits warum die Klasse der Variablen `1992` bis `1999` `character` war und nicht `double`. Für diese Jahre gab es für einige Regionen keine Angaben zum BIP und daher wurden in der Excel Tabelle `-` eingefügt. Daher ist für uns die Umwandlung zu `NA` folgerichtig und wir können die Warnmeldung ignorieren und nur die transformierten Variablen mit der Klasse `double` verwenden:

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

```
  bind_cols(bip_double)

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

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

```
bip_long <- pivot_longer(bip_wide, cols = c("2000":"1999") , names_to = "Jahr", values_to = "bip") %>%
  mutate( Jahr = as.numeric(Jahr),
          bip = bip * 1000000) %>%
  arrange( Jahr )

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 )
```

Kartenmaterial hinzufügen (optional)

Für eine spätere Visualisierung der Daten mittels einer Deutschlandkarte sollten wir uns noch Informationen zu den einzelnen Verwaltungsgrenzen als SHAPE-File herunterladen. Diese Informationen sind über das OpenData Portal des Bundesamts für Kartographie und Geodäsie verfügbar.

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.

Wir extrahieren uns hier die Informationen zu den Grenzen der Gemeinden, Verwaltungseinheiten, Landkreise und Bundesländer und speichern diese jeweils entsprechend ab. Um Geometriedaten einzulesen und diese später schön als Karte darstellen zu können müssen wir hier die Funktion `st_read` aus dem `sf`-Paket verwenden:

Da wir nur die Informationen zur Geometrie, z.B. der Landkreisgrenzen möchten, können wir die anderen Variablen auch aus dem Datensatz löschen. Wir behalten den Regionalschlüssel (RS), den Namen des Kreises/der Gemeinde (GEN) und die Geometrie (geometry).

Wir müssen uns zusätzlich noch etwas mit der Dokumentation des Kartenmaterials beschäftigen. Da wir nur die Verwaltungseinheiten ohne die Nord- und Ostsee und ohne den Bodensee darstellen möchten, so müssen wir noch auf GF $= 4$ filtern, wie auf Seite 9 der Dokumentation beschrieben wird (tun wir dies nicht, so hätten wir die Nordseegebiete doppelt drin):

Grundsätzlich gilt: Jede Verwaltungseinheit besitzt genau einen Attributsatz mit dem GF-Wert 4. Zusätzlich kann eine Verwaltungseinheit einen Attributsatz mit dem GF-Wert 2 besitzen.

```
# Auf GF == 4 filtern und RS als String speichern (ist aktuell als factor abgespeichert)
landkreise <- landkreise %>%
  filter( GF==4 ) %>%
  select(RS, GEN, geometry) %>%
  mutate(Regionalschluessel = as.character(RS))

gemeinden <- gemeinden %>%
  filter( GF==4 ) %>%
  select(RS, GEN, geometry) %>%
```

```
  mutate(Regionalschluessel = as.character(RS))

bundesland <- bundesland %>%
  filter( GF==4 ) %>%
  select(RS, GEN, geometry) %>%
  mutate(Regionalschluessel = as.character(RS))
```

Datensätze zusammenführen

In diesem letzten Abschnitt möchten wir alles für die nächsten Schritte der Case Study vorbereiten. Genauer: Neben den einzelnen Datensätzen wollen wir zusätzlich die Informationen aus den verschiedenen Datensätzen kombinieren. Hierfür müssen wir zuerst die Informationen zur Verschuldung auf Landkreisebene aggregieren und die Daten zum BIP auf das Jahr 2017 einschränken. Anschließend können wir die Datensätze anhand des Regionalschlüssels miteinander verbinden.

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

```
# Schulden auf Landkreisebene
schulden_kombi <- schulden_bereinigt %>%
  group_by(landkreis) %>%
  summarise( Schulden_pro_kopf_lk = sum(Schulden_gesamt)/sum(Einwohner), Einwohner = sum(Einwohner), Sch
  rename(Regionalschluessel = landkreis)

# BIP auf Landkreisebene im Jahr 2017
bip_kombi <- bip_long %>%
  filter(nchar(Regionalschluessel) == 5 & Jahr == 2017) %>%
  select(-Jahr)

# Anzahl an Erwerbstätigen für das Jahr 2017
erwerb_kombi <- erwerb_long %>%
  filter(nchar(Regionalschluessel) == 5 & Jahr == 2017) %>%
  select(-Jahr)

# Namen der Landkreise
landkreis_name <- landkreise %>%
  st_drop_geometry() %>%
  select(-RS) %>%
  mutate(bundesland = str_extract(Regionalschluessel, "^.{2}")) %>%
  rename(landkreis_name = GEN)

# Namen der Bundesländer
bundesland_name <- bundesland %>%
  st_drop_geometry() %>%
  select(-RS) %>%
  rename(bundesland = Regionalschluessel,
         bundesland_name = GEN)

# Datensätze zusammenführen

# Basisdatensatz -> Arbeitslosenzahlen pro Landkreis
# Name der Landkreise zumergen
daten1 <- left_join(alo_landkreis, landkreis_name, by = "Regionalschluessel")
# Namen der Bundesländer zumergen
```

```r
daten1 <- daten1 %>% mutate(bundesland = str_extract(Regionalschluessel, "^.{2}"))
daten1 <- left_join(daten1, bundesland_name, by = "bundesland")
# Schulden zumergen
daten2 <- left_join(daten1, schulden_kombi, by = "Regionalschluessel")
# BIP zumergen
daten3 <- left_join(daten2, bip_kombi, by = "Regionalschluessel")
# Zahl der Erwerbstätigen zumergen
gesamtdaten <- left_join(daten3, erwerb_kombi, by = "Regionalschluessel")

saveRDS(gesamtdaten, "data/gesamtdaten.rds")
saveRDS(schulden_bereinigt, "data/schulden_bereinigt.rds")
saveRDS(bip_long, "data/bip_long.rds")
saveRDS(bundesland, "data/bundesland.rds")
saveRDS(gemeinden, "data/gemeinden.rds")
saveRDS(landkreise, "data/landkreise.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.

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