Computational Social Science Workshop

A Gentle Introduction to R: Basics

R User Group

05.11.2018

```
pacman::p_load(tidverse, janitor, purrr, texreg)
```

```
## Installing package into '/home/rstudio-user/R/x86_64-pc-linux-gnu-library/3.5'
## (as 'lib' is unspecified)
```

```
## Warning: package 'tidverse' is not available (for R version 3.5.0)
```

```
## Warning: 'BiocManager' not available.  Could not check Bioconductor.
##
## Please use `install.packages('BiocManager')` and then retry.
```

```
## Warning in p_install(package, character.only = TRUE, ...):
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'tidverse'
```

```
## Warning in pacman::p_load(tidverse, janitor, purrr, texreg): Failed to install/load:
## tidverse
```

Dataframes

Die meiste Zeit werden wir in R nicht mit Vektoren arbeiten, sondern mit *dataframes*. Dataframes sind letztendlich einfach nur angeordnete Vektoren und essentiell wichtig für die Datenanalyse.

Lasst uns mal einen Beispieldatensatz einladen.

DOWNLOAD European Social Survey Data

Der European Social Survey ist ein transnationales Survey-Projekt und enthält über 34.000 Befragte und hunderte von Variablen zu politischen und sozio-ökonomischen Fragestellungen.

Zunächst laden wir den Datensatz ein

```
start_pirus <- get(load("data/start_pirus.Rdata")) %>% tibble::as_tibble()
```

Hier ein kleiner Ausschnitt aus dem Datensatz, welches nur das Subset *Deutschland* zeigt:

```
start_pirus
```

Möglicherweise habt ihr bemerkt, dass R Buchstabenabk?rzungen unter den Spaltennamen des Datensatzes “ anzeigt. Diese Abkürzungen beschreiben den Typ der Variablen, die in jeder Spalte gespeichert sind:

- `int` steht für ganze Zahlen (integers).
- `dbl` steht für Doubles oder reelle Zahlen.
- `chr` steht für Zeichenvektoren oder Zeichenfolgen.
- `fctr` steht für Faktoren, die R verwendet, um kategoriale Variablen darzustellen.

Eine Sache fällt dir vielleicht auch auf: das Kürzel *NA* steht für *Not Available* und denotiert missing values oder fehlende Werte.

Codebook

- id - ID des Befragten
- gender - Geschlecht des Befragten
- age - Alter des Befragten
- left_right - Links-Rechts Einstufung des Befragten
- party_ger - Parteiidentifikation
- fake_refugee - "Die meisten Flüchtlinge sind nicht echt"

Datenzugriff

Sehr angenehm kann man auf einzelne Variablen mit Hilfe des Dollarzeichens in der Form `data$variable` zugegriffen werden.

```
start_pirus

head(start_pirus$terror_group)   #zeige die ersten 5 Stellen von terror_group
tail(start_pirus$terror_group)   #zeige die letzten 5 Stellen von terror_group

start_pirus$terror_group[1]      #zeige die erste Stelle von terror_group
start_pirus$terror_group[250]    #zeige die 250. Stelle von terror_group
```

Summary und Mean

```
mean(start_pirus$age, na.rm = T)
```

```
## [1] 33.61223
```

```
summary(start_pirus$gender)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   1.902   2.000   2.000
```

Indexieren mit eckigen Klammern

Wollen wir auf verschiedene Reihen oder Spalten des dataframes zugreifen, können wir das in folgenderweise tun

```
start_pirus[1, ]        #1. Reihe
start_pirus[, 2]        #2. Spalte

start_pirus[1:10, 4]    #die ersten 10 Reihen und die vierte Spalte
```

Aufgabe

5. Gib die Reihe 1000 von `start_pirus` aus
6. Gib die Reihe 1000 bis 2000 und die erste Spalte von `start_pirus` aus

Mit der Funktion `table()` können wir uns die Häufigkeiten der Variable ausgeben lassen.

```
tabyl(start_pirus, gender)    #zeige Häufigkeiten von Geschlecht
```

```
##  gender    n    percent
##       1  182 0.09758713
##       2 1683 0.90241287
```

```
tabyl(start_pirus, type) #zeige Häufigkeiten von Parteiidentifikation
```

```
##          type   n   percent
##      Far Left 324 0.1737265
##     Far Right 746 0.4000000
##      Islamism 455 0.2439678
##  Single Issue 340 0.1823056
```

Das Ganze macht so natürlich noch nicht so viel Sinn, da die Werte nicht gelabelled sind. Daher müssen wir die Variablen **rekodieren**. Das lernen wir sogleich mit dem **Tidyverse**.

Tidyverse

Neben `Base R` gibt es das **Tidyverse**, dass viele mächtige Packages enthält!

Hier eine kleine Übersicht von relevanten Funktionen, die wir brauchen werden:

Bedeutung	tidyverse - Funktionen
Neue Variable erstellen	mutate()
Rekodieren (binär)	ifelse()
Rekodieren	case_when()
Variablen auswählen	select()
Subset erstellen	filter()
Variablennamen ändern	rename()
pipe operator	%>%
Datensatz gruppieren	group_by()
Zusammenfassen	summarize()
Zähle die Ausprägungen	count()

Zunächst einmal müssen wir das Package installieren und laden. Das geht mit in Base R mit den folgenden zwei Befehlen: `install.packages` und `library`.

Eine viel enstpannteres Funktion, welche Packages gleichzeitig installiert und einlädt nennt sich `pacman`. Zunächsten installieren wir es:

```
# install.packages("pacman")
```

Wir laden dann alle packages die wir brauchen auf folgende Weise:

```
pacman::p_load(tidyverse)
```

Wenn man ein Package nicht laden will sondern nur eine Funktion daraus, dann kann man auch zwei Doppelpunkte `::` hinter den Packagenamen schreiben und die nötige Funktion danach denotieren.

Let's get it started!

mutate

Neue Variablen erstellen mit mutate()

Mit `mutate()` wird/werden eine oder mehrere neue Variable(n) erzeugt und an den Datensatz hinten angefügt.

Beispiel

Statt dem Alter wollen wir nun das Geburstjahr haben. Einfacherweise ziehen wir das jetzige Jahr (2018) vom Alter ab um das Geburtsjahr zu erhalten. Die neue Variable nennen wir `birth`.

```
mutate(start_pirus, birth = 2018 - age)
```

```
## # A tibble: 1,865 x 11
##    subject_id   age gender student abuse_child crime_history extent
##         <dbl> <dbl>  <dbl>   <dbl>       <dbl>         <dbl>  <dbl>
##  1       1000    32      2       1           0             3      2
##  2       1001    20      2       1           0             0      0
```

```
##  3       1002    28      2       0           0             3     NA
##  4       1005    25      2       0           0             0      5
##  5       1006    25      2       0           0            NA      5
##  6       1010    27      2       0           0            NA      2
##  7       1013    23      2       0           0             1      0
##  8       1014    28      2       0           0             1      0
##  9       1015    26      2       0           0             1      0
## 10       1016    25      2       0           0            NA      0
## # ... with 1,855 more rows, and 4 more variables: terror_group <chr>,
## #   radicalization <dbl>, type <chr>, birth <dbl>
```

Weisen wir den Datensatz wieder `ess_ger` zu, so wird unsere Veränderung auch im Objekt festgehalten

```
start_pirus <- mutate(start_pirus, birth = 2018 - age)

start_pirus
```

```
## # A tibble: 1,865 x 11
##    subject_id   age gender student abuse_child crime_history extent
##         <dbl> <dbl>  <dbl>   <dbl>       <dbl>         <dbl>  <dbl>
##  1       1000    32      2       1           0             3      2
##  2       1001    20      2       1           0             0      0
##  3       1002    28      2       0           0             3     NA
##  4       1005    25      2       0           0             0      5
##  5       1006    25      2       0           0            NA      5
##  6       1010    27      2       0           0            NA      2
##  7       1013    23      2       0           0             1      0
##  8       1014    28      2       0           0             1      0
##  9       1015    26      2       0           0             1      0
## 10       1016    25      2       0           0            NA      0
## # ... with 1,855 more rows, and 4 more variables: terror_group <chr>,
## #   radicalization <dbl>, type <chr>, birth <dbl>
```

Aufgabe

Die Links-Rechts Skala geht von 0 - 10. Teile die Variable durch 10 um einen Range von 0 - 1 zu erhalten. Nenne die Variable `lr01`. Mit `mutate()` wird die Variable hinten angehangen.

```
mutate(start_pirus, radicalization01 = radicalization  / 5)
```

```
## # A tibble: 1,865 x 12
##    subject_id   age gender student abuse_child crime_history extent
##         <dbl> <dbl>  <dbl>   <dbl>       <dbl>         <dbl>  <dbl>
##  1       1000    32      2       1           0             3      2
##  2       1001    20      2       1           0             0      0
##  3       1002    28      2       0           0             3     NA
##  4       1005    25      2       0           0             0      5
##  5       1006    25      2       0           0            NA      5
##  6       1010    27      2       0           0            NA      2
##  7       1013    23      2       0           0             1      0
##  8       1014    28      2       0           0             1      0
##  9       1015    26      2       0           0             1      0
## 10       1016    25      2       0           0            NA      0
## # ... with 1,855 more rows, and 5 more variables: terror_group <chr>,
## #   radicalization <dbl>, type <chr>, birth <dbl>, radicalization01 <dbl>
```

ifelse

Die Funktion `ifelse()` testet eine logische Bedingung in ihrem ersten Argument. Wenn der Test `TRUE` ist, gibt `ifelse()` das zweite Argument zur?ck. Wenn der Test `FALSE` ist, gibt `ifelse()` das dritte Argument zur?ck.

Also in folgender Form:

`ifelse(logischer Test, was passiert wenn zutrifft, was passiert wenn nicht zutrifft)`

Das können wir in Kombination mit `mutate` benutzen.

Beispiel:

Nehmen wir an, dass wir die Altervariable als eine dummy Variable (zwei Ausprägungen) benutzen wollen. Am besten geht das mit `ifelse()`. Kodieren wir doch einmal die 18-40 Jährigen als eine Gruppe und über 40 Jährige als eine andere Gruppe.

Eine Variable `u40` erstellen welche die 18 bis 40 J?hrigen und die ?ber 40 J?hrigen in eine Gruppe teilt.

```
mutate(start_pirus, u40 = ifelse(age <= 40, "18 - 40" ,"> 40"))
```

```
## # A tibble: 1,865 x 12
##    subject_id   age gender student abuse_child crime_history extent
##         <dbl> <dbl>  <dbl>   <dbl>       <dbl>         <dbl>  <dbl>
##  1       1000    32      2       1           0             3      2
##  2       1001    20      2       1           0             0      0
##  3       1002    28      2       0           0             3     NA
##  4       1005    25      2       0           0             0      5
##  5       1006    25      2       0           0            NA      5
##  6       1010    27      2       0           0            NA      2
##  7       1013    23      2       0           0             1      0
##  8       1014    28      2       0           0             1      0
##  9       1015    26      2       0           0             1      0
## 10       1016    25      2       0           0            NA      0
## # ... with 1,855 more rows, and 5 more variables: terror_group <chr>,
## #   radicalization <dbl>, type <chr>, birth <dbl>, u40 <chr>
```

```
#darauf achten, dass bei strings ('Wörtern') Anführungsstriche gemacht werden müssen!
```

Aufgabe

7. Kodiere `gender_rec` in der folgenden Art und Weise:

1 = “männlich” 2 = “weiblich”

Überschreibe `ess_ger` zeige den Datensatz!

```
start_pirus <- mutate(start_pirus, gender = ifelse(gender == 2, "male" ,"female"))
#darauf achten, dass bei strings ('Wörtern') Anführungsstriche gemacht werden müssen!

start_pirus
```

```
## # A tibble: 1,865 x 11
##    subject_id   age gender student abuse_child crime_history extent
##         <dbl> <dbl> <chr>    <dbl>       <dbl>         <dbl>  <dbl>
##  1       1000    32 male         1           0             3      2
##  2       1001    20 male         1           0             0      0
##  3       1002    28 male         0           0             3     NA
##  4       1005    25 male         0           0             0      5
##  5       1006    25 male         0           0            NA      5
```

```
##  6       1010    27 male         0           0            NA      2
##  7       1013    23 male         0           0             1      0
##  8       1014    28 male         0           0             1      0
##  9       1015    26 male         0           0             1      0
## 10       1016    25 male         0           0            NA      0
## # ... with 1,855 more rows, and 4 more variables: terror_group <chr>,
## #   radicalization <dbl>, type <chr>, birth <dbl>
```

case_when

Was aber wenn wir mehrere Werte rekodieren wollen? Enter the world of `case_when`.

Mit `case_when` können wir einzelne Variablen rekodieren und sogar Bedingungen nach Lust und Laune mixen. Hier ein Beispiel:

Variable Description: Prior to their radicalization, does the individual have a history of involvement in non-ideologically motivated criminal activities?

- 0 = No previous criminal activity
- 1 = Previous (non-violent) minor criminal activity (e.g., convicted of a misdemeanor crime)
- 2 = Previous (non-violent) serious criminal activity (e.g., convicted of a felony crime)
- 3 = Previous violent crime

```
start_pirus <- mutate(start_pirus, crime_history = case_when(
    crime_history == 0 ~ "None",
    crime_history == 1 ~ "Non-Violent",
    crime_history == 2 ~ "Non-Violent",
    crime_history == 3 ~ "Violent"
  ))

start_pirus
```

```
## # A tibble: 1,865 x 11
##    subject_id   age gender student abuse_child crime_history extent
##         <dbl> <dbl> <chr>    <dbl>       <dbl> <chr>          <dbl>
##  1       1000    32 male         1           0 Violent            2
##  2       1001    20 male         1           0 None               0
##  3       1002    28 male         0           0 Violent           NA
##  4       1005    25 male         0           0 None               5
##  5       1006    25 male         0           0 <NA>               5
##  6       1010    27 male         0           0 <NA>               2
##  7       1013    23 male         0           0 Non-Violent        0
##  8       1014    28 male         0           0 Non-Violent        0
##  9       1015    26 male         0           0 Non-Violent        0
## 10       1016    25 male         0           0 <NA>               0
## # ... with 1,855 more rows, and 4 more variables: terror_group <chr>,
## #   radicalization <dbl>, type <chr>, birth <dbl>
```

Jede Bedingung untereinander gereiht und mit einem Komma getrennt. Wenn eine Bedingung ein `TRUE` Statement ist, dann wird mit dem ~ Operator der entsprechende Wert zugewiesen. Gibt man `TRUE` selbst an, so kann man "alle anderen" Ausprägungen, die nicht vorher abgefragt worden sind einen Wert zuweisen. Gibt man die Originalvariable an, so bleiben die restlichen Werte wie sie sind.

select

Mit `select` werden Spalten (=Vektoren/Variablen) mittels dem Variablennamen oder einer Hilfsfunktion ausgewählt.

Wählen wir wieder nur die zwei folgenden Variables aus:

- `age`
- `fake_refugee`

```
select(start_pirus , age, gender)
```

```
## # A tibble: 1,865 x 2
##      age gender
##    <dbl> <chr>
##  1    32 male
##  2    20 male
##  3    28 male
##  4    25 male
##  5    25 male
##  6    27 male
##  7    23 male
##  8    28 male
##  9    26 male
## 10    25 male
## # ... with 1,855 more rows
```

`Select` eignet sich auch dafür Variablen aus einem bestehenden dataframe zu entfernen. Dies ist ganz einfach zu lösen mit einem `-` (Minus).

Aufgaben

10. Wähle die folgende Variablen aus:

- `gender_rec`
- `left_right`
- `party_ger_cat`

11. Wähle alle Variablen **außer** `fake_refugee` und `age`.

```
select(start_pirus , terror_group, type, crime_history)
```

```
## # A tibble: 1,865 x 3
##    terror_group  type     crime_history
##    <chr>         <chr>    <chr>
##  1 al-Qaeda core Islamism Violent
##  2 Taliban       Islamism None
##  3 al-Qaeda core Islamism Violent
##  4 <NA>          Islamism None
##  5 <NA>          Islamism <NA>
##  6 al-Qaeda core Islamism <NA>
##  7 al-Qaeda core Islamism Non-Violent
##  8 al-Qaeda core Islamism Non-Violent
##  9 al-Qaeda core Islamism Non-Violent
## 10 al-Qaeda core Islamism <NA>
## # ... with 1,855 more rows
```

```
select(start_pirus , -student)
```

```
## # A tibble: 1,865 x 10
```

```
##    subject_id   age gender abuse_child crime_history extent terror_group
##         <dbl> <dbl> <chr>        <dbl> <chr>          <dbl> <chr>
##  1       1000    32 male             0 Violent            2 al-Qaeda co~
##  2       1001    20 male             0 None               0 Taliban
##  3       1002    28 male             0 Violent           NA al-Qaeda co~
##  4       1005    25 male             0 None               5 <NA>
##  5       1006    25 male             0 <NA>               5 <NA>
##  6       1010    27 male             0 <NA>               2 al-Qaeda co~
##  7       1013    23 male             0 Non-Violent        0 al-Qaeda co~
##  8       1014    28 male             0 Non-Violent        0 al-Qaeda co~
##  9       1015    26 male             0 Non-Violent        0 al-Qaeda co~
## 10       1016    25 male             0 <NA>               0 al-Qaeda co~
## # ... with 1,855 more rows, and 3 more variables: radicalization <dbl>,
## #   type <chr>, birth <dbl>
```

filter

Zeilen auswählen mit filter()

Mit `filter()` behält man oder selektiert man Zeilen eines Datensatzes, welche bestimmte logische Kriterien oder Konditionen erfüllen. Damit wird ein Subset (Untergruppe) gebildet. Wir wählen jetzt nur die Fälle aus, welche unser Kriterium erfüllen. Jetzt zeigen wir uns Personen an, welchse sich mit der *FDP* identifizieren.

Beispiel:

```
filter(start_pirus, type == "Far Right")
```

Aufgaben

12. Filtere den Datensatz und zeige nur die Personen, welche sich mit der AfD identifizieren.
13. Filtere den Datensatz und zeige nur die Personen, welche unter 30 Jahre alt sind und sich als eher ganz rechts (gröer gleich 8) einstufen.
14. Filtere den Datensatz und zeige nur die Personen, welche sich mit der FDP identifizieren und die Mehrheit der Fl?chtlinge als nicht wirklich politisch verfolgt ansehen ("Agree" und "Agree strongly").

Hier nochmal die Operatoren zum spicken ;)

- `==` (logisch) ist gleich
- `!=` (logisch) ist ungleich
- `>` größer als
- `<` kleiner als
- `>=` größer gleich
- `<=` kleiner gleich

rename

Variablen umbennen mit `rename()`

Mit `rename()` lassen sich die Variablen umbennen. Beispielhaft ?bersetzen wir `age`, `gender` und `links_rechts` Variablennamen ins Deutsche.

Beispiel:

```
rename(start_pirus,
    group = terror_group,
```

```
    rad = radicalization,
    ideology = type)
```

```
## # A tibble: 1,865 x 11
##    subject_id   age gender student abuse_child crime_history extent group
##         <dbl> <dbl> <chr>    <dbl>       <dbl> <chr>          <dbl> <chr>
##  1       1000    32 male         1           0 Violent            2 al-Q~
##  2       1001    20 male         1           0 None               0 Tali~
##  3       1002    28 male         0           0 Violent           NA al-Q~
##  4       1005    25 male         0           0 None               5 <NA>
##  5       1006    25 male         0           0 <NA>               5 <NA>
##  6       1010    27 male         0           0 <NA>               2 al-Q~
##  7       1013    23 male         0           0 Non-Violent        0 al-Q~
##  8       1014    28 male         0           0 Non-Violent        0 al-Q~
##  9       1015    26 male         0           0 Non-Violent        0 al-Q~
## 10       1016    25 male         0           0 <NA>               0 al-Q~
## # ... with 1,855 more rows, and 3 more variables: rad <dbl>,
## #   ideology <chr>, birth <dbl>
```

Ist eine Abkürzung des folgenden Code:

```
start_pirus2 <- rename(start_pirus, group = terror_group)
start_pirus2 <- rename(start_pirus2, rad = radicalization)
start_pirus2 <- rename(start_pirus2, ideology = type)

start_pirus2
```

```
## # A tibble: 1,865 x 11
##    subject_id   age gender student abuse_child crime_history extent group
##         <dbl> <dbl> <chr>    <dbl>       <dbl> <chr>          <dbl> <chr>
##  1       1000    32 male         1           0 Violent            2 al-Q~
##  2       1001    20 male         1           0 None               0 Tali~
##  3       1002    28 male         0           0 Violent           NA al-Q~
##  4       1005    25 male         0           0 None               5 <NA>
##  5       1006    25 male         0           0 <NA>               5 <NA>
##  6       1010    27 male         0           0 <NA>               2 al-Q~
##  7       1013    23 male         0           0 Non-Violent        0 al-Q~
##  8       1014    28 male         0           0 Non-Violent        0 al-Q~
##  9       1015    26 male         0           0 Non-Violent        0 al-Q~
## 10       1016    25 male         0           0 <NA>               0 al-Q~
## # ... with 1,855 more rows, and 3 more variables: rad <dbl>,
## #   ideology <chr>, birth <dbl>
```

So sparen wir uns da mehrmals abspeichern! Noch besser geht das aber mit der...

%>%

Mit der Hilfe von `%>%` können alle diese Operationen auf einmal ausgeführt werden!

[Prozent größer Prozent] wird pipe operator genannt. Diese Pipe ermöglicht es Daten einfacher, verständlicher und lesbarer und ohne Verlust von Flexibilität zu transformieren.

Den pipe operator kann man sich als "danach" vorstellen.

Hier ein Beispiel:

```
ess_ger %>%
  select(age, gender, left_right, party_ger) %>%
  rename(alter = age, geschl = gender, links_rechts = left_right) %>%
  select(alter, geschl)
```

Jetzt seid ihr dran!

Benutzt die pipe für folgende Aufgaben:

15. Selektiere die folgenden Variablen
    - `gender_rec`
    - `age`
    - `left_right`
    - `party_ger_cat`
    - `fake_refugee`
16. Filtere die Variable und zeige nur Fälle mit der Ausprägung `männlich`
17. Erstelle eine neue Variable namens `links` mit der Ausprägung `1` für die Werte 0, 1, 2, 3 und `0` für alle anderen.
18. Benenne die Variable `fake_refugee` in `anti_refugee` um.
19. Selektiere die folgenden Variablen
    - `gender_rec`
    - `age`
    - `links`
    - `anti_refugee`

count

```
start_pirus %>%
  count(terror_group, sort = T)
```

```
## # A tibble: 80 x 2
##    terror_group                                 n
##    <chr>                                    <int>
##  1 <NA>                                       730
##  2 Ku Klux Klan                               148
##  3 Islamic State of Iraq and the Levant (ISIL)  146
##  4 Jewish Defense League (JDL)                 60
##  5 al-Shabaab                                  57
##  6 Weather Underground                         54
##  7 al-Qaeda core                               51
##  8 Earth Liberation Front (ELF)                44
##  9 Animal Liberation Front (ALF)               36
## 10 Aryan Nations                               36
## # ... with 70 more rows
```

```
## ist das gleiche wie:

start_pirus %>%
  tabyl(type)
```

```
##          type   n   percent
```

```
##      Far Left 324 0.1737265
##     Far Right 746 0.4000000
##      Islamism 455 0.2439678
##  Single Issue 340 0.1823056
```

```
start_pirus %>%
  count(type, gender, sort = T)
```

```
## # A tibble: 8 x 3
##   type         gender     n
##   <chr>        <chr>  <int>
## 1 Far Right    male     706
## 2 Islamism     male     427
## 3 Single Issue male     305
## 4 Far Left     male     245
## 5 Far Left     female    79
## 6 Far Right    female    40
## 7 Single Issue female    35
## 8 Islamism     female    28
```

```
## ist das gleiche wie:

start_pirus %>%
  group_by(type, gender) %>%
  tally()
```

```
## # A tibble: 8 x 3
## # Groups:   type [?]
##   type         gender     n
##   <chr>        <chr>  <int>
## 1 Far Left     female    79
## 2 Far Left     male     245
## 3 Far Right    female    40
## 4 Far Right    male     706
## 5 Islamism     female    28
## 6 Islamism     male     427
## 7 Single Issue female    35
## 8 Single Issue male     305
```

```
## ist das gleiche wie:

start_pirus %>%
  group_by(type, gender) %>%
  summarize(n = n())
```

```
## # A tibble: 8 x 3
## # Groups:   type [?]
##   type         gender     n
##   <chr>        <chr>  <int>
## 1 Far Left     female    79
## 2 Far Left     male     245
## 3 Far Right    female    40
## 4 Far Right    male     706
## 5 Islamism     female    28
## 6 Islamism     male     427
## 7 Single Issue female    35
```

```
## 8 Single Issue male     305
```

group_by + summary

Gruppieren und Summaries

Mit `group_by()` werden die nachfolgden Operationen gruppenweise ausgeführt. Mit `summarise()` werden die gruppierten Variablen aggregiert

```
start_pirus %>%
  group_by(type) %>% # Gruppenvariable
  summarise(mean_age = mean(age, na.rm = T))
```

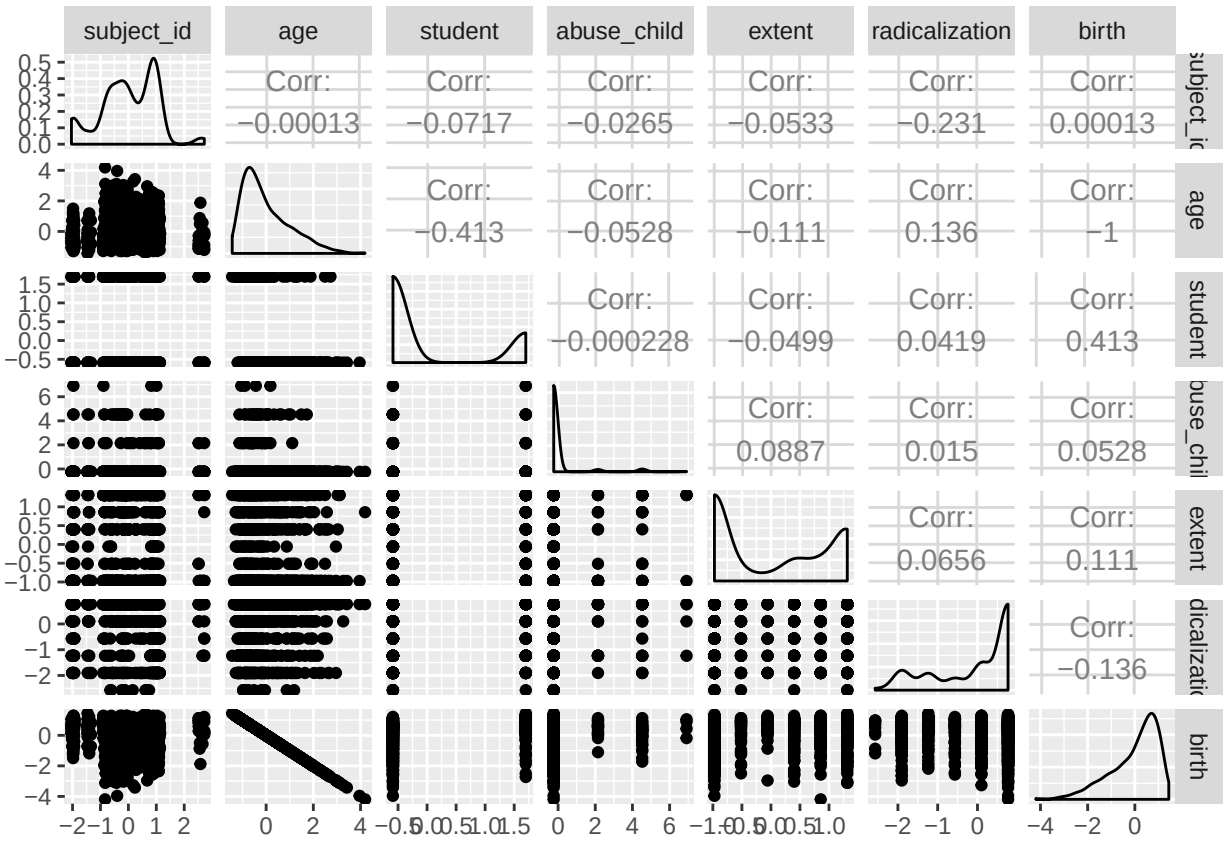
```
start_pirus %>%
  group_by(type) %>% # Gruppenvariable
  summarise(mean_extent = mean(extent, na.rm = T),
            n = n()) %>%
  arrange(desc(mean_extent))
```

```
## # A tibble: 4 x 3
##   type         mean_extent     n
##   <chr>              <dbl> <int>
## 1 Far Left            2.59   324
## 2 Single Issue        2.38   340
## 3 Far Right           2.27   746
## 4 Islamism            1.16   455
```

Aufgabe

20. Gruppiere `ess_ger` nach `gender_rec` und rechne den Mittelwerte der Links-Rechts Skala (`left_right`) per Geschlecht aus. Vergiss nicht `na.rm = T` zu spezifizieren!

```
z <- function(x) scale(x)[,1]
start_lm <- start_pirus %>%
  drop_na(radicalization, age, gender, abuse_child, crime_history) %>%
  mutate_if(is.numeric, z)  %>%
#  glimpse()
  mutate(crime_history = forcats::fct_relevel(crime_history, "None"))


start_lm %>%
  select_if(is.numeric) %>%
  #select(extent, radicalization, abuse_child, crime_history, age, gender) %>%
  GGally::ggpairs(progress = F)
```



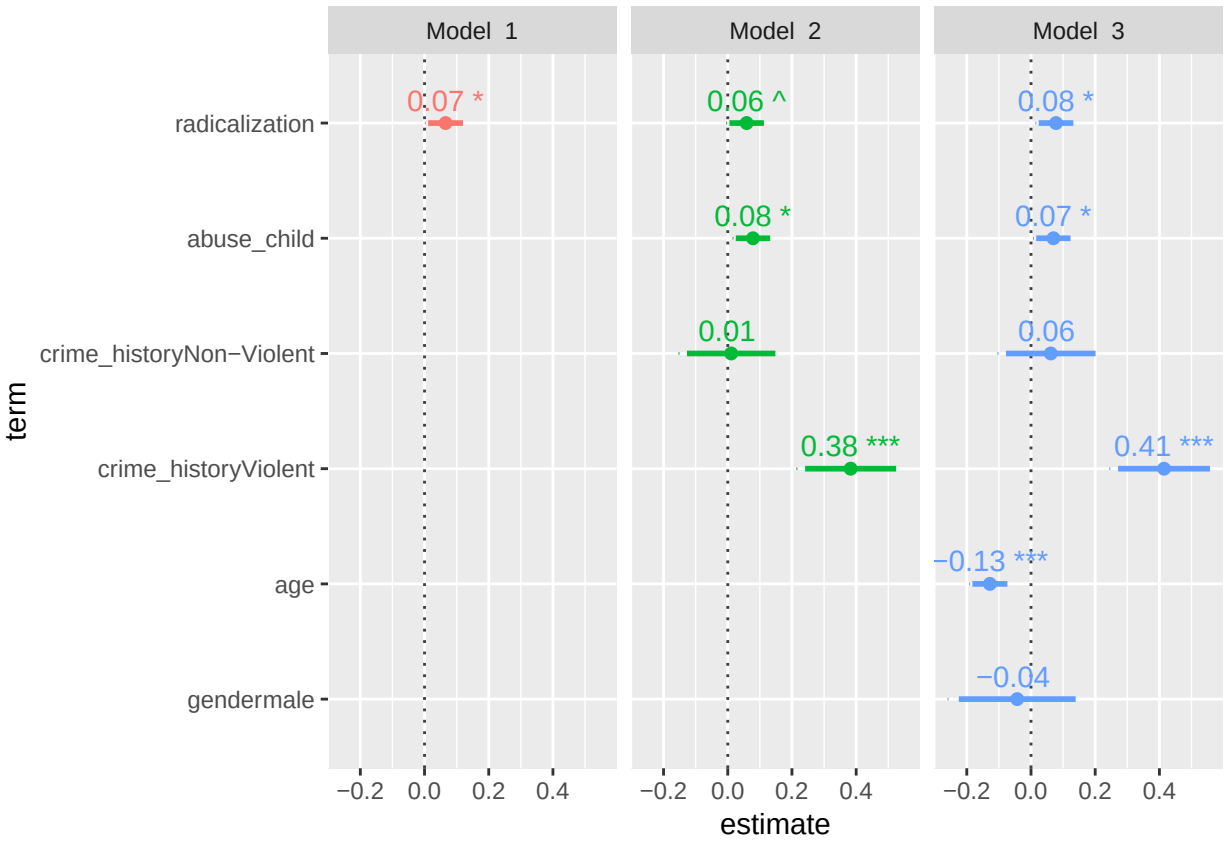

```
model1 <- start_lm %>%
  lm(extent ~ radicalization, data = .)

model2 <- start_lm %>%
  lm(extent ~  radicalization + abuse_child + crime_history, data = .)

model3 <- start_lm %>%
  lm(extent ~  radicalization + abuse_child + crime_history + age + gender, data = .)

screenreg(list(model1, model2, model3))
```

```
##
## ============================================================
##                             Model 1   Model 2     Model 3
## ------------------------------------------------------------
## (Intercept)                  0.00     -0.07       -0.05
##                             (0.03)    (0.04)      (0.10)
## radicalization               0.07 *    0.06        0.08 *
##                             (0.03)    (0.03)      (0.03)
## abuse_child                            0.08 *      0.07 *
##                                       (0.03)      (0.03)
## crime_historyNon-Violent               0.01        0.06
##                                       (0.08)      (0.08)
## crime_historyViolent                   0.38 ***    0.41 ***
##                                       (0.09)      (0.09)
## age                                               -0.13 ***
##                                                   (0.03)
```

```
## gendermale                                            -0.04
##                                                       (0.11)
## ------------------------------------------------------------
## R^2                            0.00      0.03        0.05
## Adj. R^2                       0.00      0.03        0.04
## Num. obs.                    913       913         913
## RMSE                           1.00      0.99        0.98
## ============================================================
## *** p < 0.001, ** p < 0.01, * p < 0.05
```

```r
interval90 <- -qnorm((1-0.9)/2)  # 90% multiplier
interval95 <- -qnorm((1-0.95)/2) # 95% multiplier
model_dat <- list(
  model1,
  model2,
  model3
) %>%
  map2_df(.y = paste("Model ", 1:3), ~{broom::tidy(.x) %>% mutate(model = .y)}) %>%
  mutate(low90 = estimate - std.error *interval90) %>%
  mutate(high90 = estimate + std.error * interval90)  %>%
  mutate(low95 = estimate - std.error * interval95)  %>%
  mutate(high95 = estimate - std.error * interval95) %>%
  mutate(stars = tidytemplate::get_stars(p.value)) %>%
  filter(term != "(Intercept)") %>%
  mutate(term = factor(term, levels = rev(c("radicalization", "abuse_child", "crime_historyNon-Violent"
```

```r
model_dat %>%
 ggplot() +
  aes(x = term, y = estimate, color = model) +
  geom_hline(yintercept = 0,
             color = "gray25",
             linetype = "dotted") +
  geom_linerange(aes(x = term,
                     ymin = low90,
                     ymax = high90),
                     lwd = 1, position = position_dodge(width = 0.7), show.legend = F) +
  geom_pointrange(aes(x = term,
                      y = estimate, ymin = low95,
                      ymax = high95),
                      lwd = 1/2, position = position_dodge(width = 0.7),
                  shape = 20, fill = "white") +
  coord_flip() +
  geom_text(aes(x = term,
                y = estimate,
                label = paste(sprintf('%.2f', estimate, 2), stars)),
          nudge_x = .2,
            show.legend = F) +
  facet_wrap(~model) +
  theme(legend.position = "none")
```


Model 1
Model 2
Model 3
radicalization
abuse_child
crime_historyNon−Violent
crime_historyViolent
age
gendermale
term
0.07 *
0.06 ^
0.08 *
0.08 *
0.07 *
0.01
0.06
0.38 ***
0.41 ***
−0.13 ***
−0.04
−0.2
0.0
0.2
0.4
estimate