

Datenanalyse mit R

9 Visualisierung Einzelvariablen

Tobias Wiß, Carmen Walenta und Felix Wohlgemuth

02.05.2020



Institut für
Gesellschafts-
und Sozialpolitik

Wiederholung

Visualisierungen mit ggplot2

ggplot2 - Syntax

Um einen besser Überblick über die unterschiedlichen Bausteine von ggplot2 zu bekommen, öffnen Sie das ggplot2 cheatsheet

<https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf>

Complete the template below to build a graph.

ggplot (data = **<DATA>**) +

<GEOM_FUNCTION> (mapping = aes(**<MAPPINGS>**),

stat = **<STAT>**, position = **<POSITION>**) +

<COORDINATE_FUNCTION> +

<FACET_FUNCTION> +

<SCALE_FUNCTION> +

<THEME_FUNCTION>

required

Not
required,
sensible
defaults
supplied

ggplot2 - Syntax

1. In `ggplot()` Daten definieren (optional auch `aes()` falls Variablen für alle `geom_functions` gleich bleiben):
`ggplot(data = <DATAFRAMENAMEN>)`
2. Die gewünschte `geom_function` als zweite Ebene per `+` definieren (zB `geom_line()`):
`geom_line(mapping = aes(<MAPPINGS>))`
3. In den aesthetics `aes()` die Variablen definieren:
`geom_line(mapping = aes(x = <X-ACHSE>, y = <Y-ACHSE>, colour = <GRUPPIERUNGSVARIABLE>))`
4. Aussehen des Plots außerhalb von `aes()` definieren: zB Linienstärke mit `size = 2`
5. optional per `+` eine weitere `geom_function` hinzufügen oder das Aussehen der Grafik bearbeiten (zB mit `labs()` die Achsen beschriften und Titel festlegen)
6. optional ein fertiges theme anwenden oder selbst erstellen
(Überblick: <https://ggplot2.tidyverse.org/reference/ggtheme.html>)

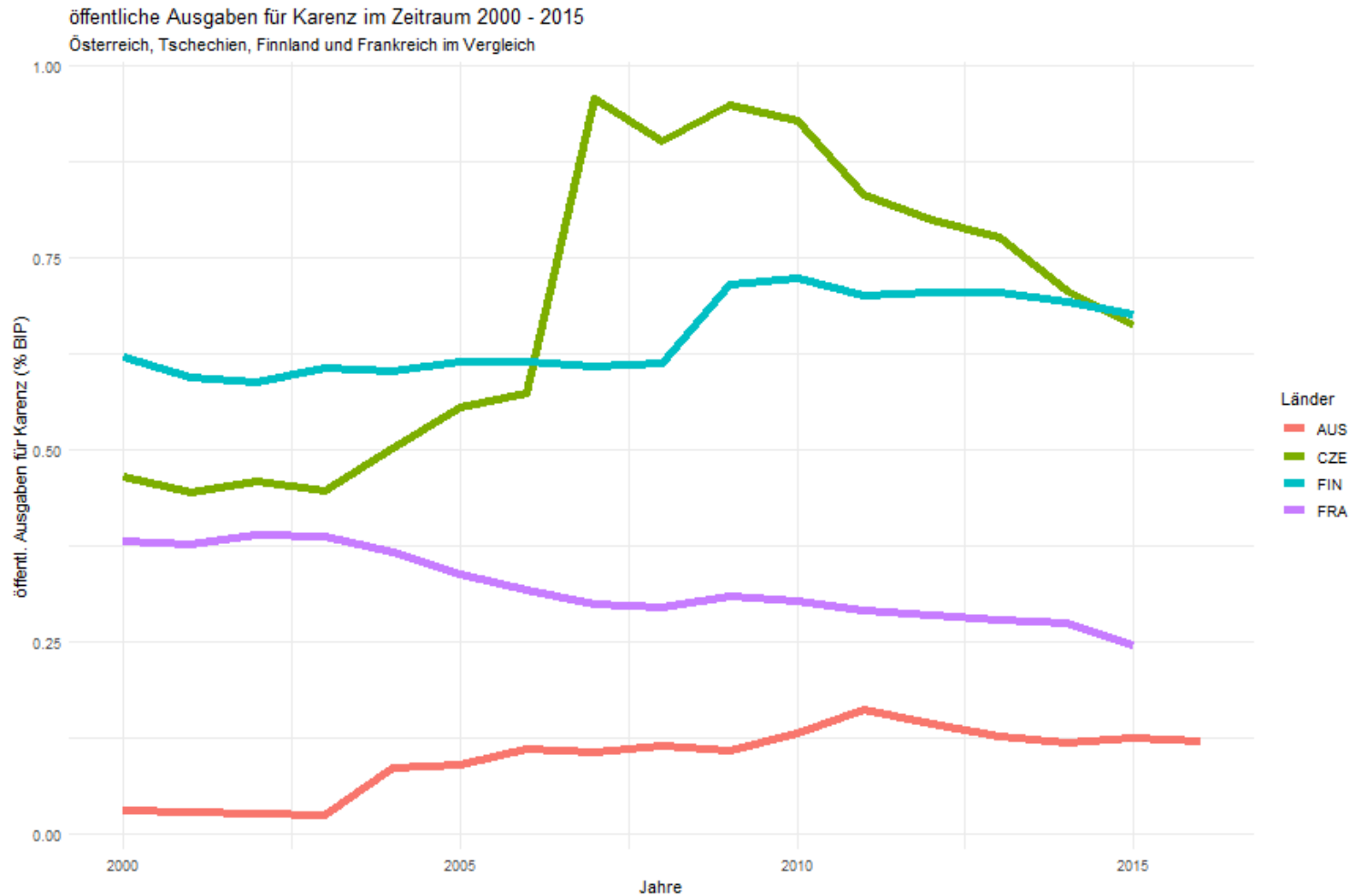
```

# load packages & import data
library(tidyverse)
socx_data <- read_csv("_raw/SOCX_AGG_20042020191205895.csv")

# visualisations with ggplot2
socx_data %>%
  filter(COUNTRY %in% c("AUS", "FIN", "FRA", "CZE")) %>%
# define data with %>%
  ggplot() +
# define geom_function of choice
  geom_line(aes(x = YEAR, # define variable on x-axis
                y = family_cash_leave_pct_gdp, # define variable on y-axis
                colour = COUNTRY), # define grouping variable
            size = 2) + # set strength of lines
# set title
  labs(title = "Öffentliche Ausgaben für Karenz im Zeitraum 2000 - 2020",
# set subtitle
       subtitle = "Österreich, Tschechien, Finnland und Frankreich im Vergleich",
# set name of x-axis
       x = "Jahre",
# set name of y-axis
       y = "Öffentl. Ausgaben für Karenz (% BIP)",
# set name of legend
       colour = "Länder") +
# apply predefined theme
  theme_minimal()

```

ggplot2 - Syntax



Falls Sie noch Fragen haben, nutzen Sie das **Forum** auf moodle und unterstützen Sie Ihre Kolleg*innen mit Ihrem Wissen!



Forum für R & RStudio Fragen

Hier können Sie alle Fragen, die Sie zu R und RStudio haben, stellen und auch Probleme diskutieren. Wir werden auf Ihre Fragen antworten. Bitte unterstützen Sie auch Ihre Kolleg*innen mit Ihrem Wissen. Falls Sie die Lösung für ein Problem haben, dann antworten Sie einfach unter der Frage ihrer Kolleg*in.

Nutzen Sie auch unsere **R Sprechstunde**.
Jeden Donnerstag von 11:00 bis 11:45 auf zoom oder im Anschluss an die zoom-Sitzung zu Familienpolitik (Link finden Sie auf moodle).

Einschub

Daten unformen: langes vs. breites Format

Falls Sie es interessiert, wenn nicht springen Sie direkt zu Folie 24. Ab dort geht es um die Visualisierung von Einzelvariablen.

langes vs. breites Format

Wenn Sie Daten der OECD oder Eurostat herunterladen, haben diese meistens kein tidy-data-Format.

Für unsere Analysen benötigen wir aber meistens tidy data, wo jede Spalte eine eigene Variable ist und jede Zeile eine Beobachtung.

country	year	cases	population
Afghanistan	1999	211745	19997071
Afghanistan	2000	21666	20095360
Brazil	1999	31737	17206362
Brazil	2000	80488	17404898
China	1999	211258	1272015272
China	2000	210766	128042583

variables

country	year	cases	population
Afghanistan	1999	211745	19997071
Afghanistan	2000	21666	20095360
Brazil	1999	31737	17206362
Brazil	2000	80488	17404898
China	1999	211258	1272015272
China	2000	210766	128042583

observations

country	year	cases	population
Afghanistan	1999	211745	19997071
Afghanistan	2000	21666	20095360
Brazil	1999	31737	17206362
Brazil	2000	80488	17404898
China	1999	211258	1272015272
China	2000	210766	128042583

values

`gather()` und `spread()` aus dem `tidyr`-Paket des `tidyverse` wandeln die Daten vom langen zum breiten Format und andersherum um.

Ausführliches Tutorial: <http://ohi-science.org/data-science-training/tidyr.html>

Grafik: <https://r4ds.had.co.nz/tidy-data.html>

breites Format

Zuerst laden wir Daten und laden tidyverse. Die eurostat-Daten können zB im Excel-Format heruntergeladen werden, deshalb brauch wir readxl und müssen einige zusätzlich Optionen definieren.

```
library(tidyverse)
library(readxl)
data_care_Y3_M30 <- read_excel("_raw/ilc_caindformal-1_excelData.xls",
  sheet = "Data3", range = "A10:O38", na = ":")
```

Die Daten des Excel-Sheets "Data3" zeigen den Anteil der Kinder unter 3 Jahren die 30 Stunden oder mehr in formaler Kinderbetreuung sind.

Der Datensatz besteht aus 15 Variablen:

- Ländernamen
- Jahre 2005 bis 2018

Andere Altersgruppen und Betreuungszeiten sind auf den anderen Sheets. Jedes Sheet beinhaltet aber nur eine Variable (zusätzlich zu Länder und Jahre).

breites Format

Show entries

Search:

	GEO/TIME ▾	2005 ▾	2006 ▾	2007 ▾	2008 ▾	2009 ▾	2010 ▾	2011 ▾	2012 ▾
1	BE	19	23	23	23	16	19	20	27
2	BG		16	6	9	7	6	7	8
3	CZ	0	1	0	0	0	0	1	1
4	DK	60	66	63	65	63	68	69	59
5	DE	8	7	9	9	12	13	15	15
6	EE	9	12	14	16	21	19	15	14
7	IE	6	5	11	8	5	8	11	10.2

Showing 1 to 7 of 28 entries

Previous

1

2

3

4

Next

breites Format umformen - gather()

Das Problem mit den Daten ist, dass wir die Jahre als eigene Variable benötigen und dass die Werte unserer Variable über 14 Jahresvariablen verteilt sind.

Wir müssen den Datensatz so umwandeln, dass wir drei Variablen haben: COUNTRY, YEAR und care_Y3_M30.

Wir machen aus dem breiten Format ein langes Format, dafür nutzen wir `gather()` (macht den Datensatz schmaler).

```
data_care_Y3_M30 <- gather(data_care_Y3_M30, # define data
                           "2005":"2018",    # set columns for trans
                           key = "YEAR",       # new variable name for
                           value = "care_Y3_M30") %>% # new variable
  rename(COUNTRY = "GEO/TIME") # not necessary: rename country variable
```

Das passende Cheatsheet zu `gather()` und `spread()` Seite 2

<https://github.com/rstudio/cheatsheets/blob/master/data-import.pdf>

breites Format umformen - gather()

Syntax von `gather()`:

1. Die Daten definieren `data_care_Y3_M30` (`%>%` funktioniert auch).
2. Die Spalten benennen in denen sich die Werte befinden. Entweder jede Spalte einzeln oder `"2005": "2018"`, d.h. von der Spalte "2005" bis zur Spalte "2018".
3. Den Variablennamen definieren in denen die Namen der vorherigen Spalten gespeichert werden (also 2005 bis 2018) mit `key = "YEAR"`.
4. Den Variablennamen definieren in denen die Werte gespeichert werden mit `value = "care_Y3_M30"`

Unabhängig von `gather()` haben wir im letzten Schritt `GEOM/TIME` in `COUNTRY` umbenannt. Da wir die Ländervariable in `gather()` nicht erwähnt haben, wird sie einfach mitkopiert und nicht verändert.

breites Format umformen - gather()

Show entries

Search:

COUNTRY	YEAR	care_Y3_M30
BE	2005	19
BG	2005	
CZ	2005	0
DK	2005	60
DE	2005	8
EE	2005	9
IE	2005	6

Showing 1 to 7 of 392 entries

Previous

1

2

3

4

5

...

56

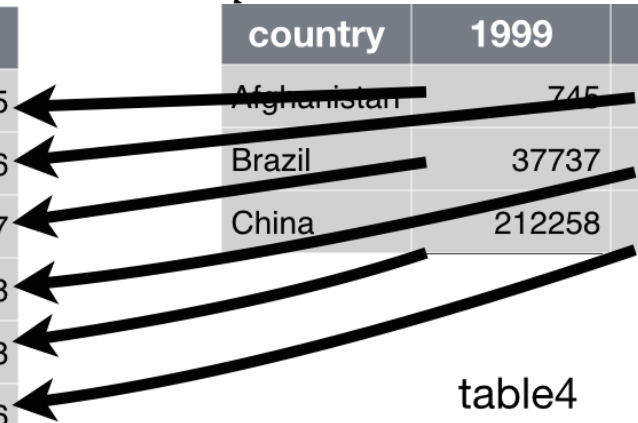
Next

langes Format

- Der resultierende Dataframe ist im tidy-Format und im langen (long) Format.
- Der ursprüngliche dataframe hatte mehr Spalten.
- Der umgewandelte dataframe hat jetzt mehr Zeilen.

Jetzt sieht der dataframe so aus wie wir es gewohnt sind und kann für alle bisher gelernten Funktionen verwendet werden.

Übersicht (Werte aus einem anderen Beispiel)



country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Grafik: <https://r4ds.had.co.nz/tidy-data.html>

langes Format

OECD oder eurostat Daten können aber auch im langen Format heruntergeladen werden. Das passiert meistens wenn man mehrere Variablen auswählt und diese im .csv Format bezieht.

Die gleichen Daten zum Anteil der Kinder in formaler Kinbetreuung mit allen Untergliederungen nach Altersgruppen und nach Betreuungszeiten können in einer Datei ohne mehrere Excel-Sheets heruntergeladen werden.

```
data_care_age_hours <- read_csv("_raw/ilc_caindformal_1_csvData.csv",  
  select(-"Flag and Footnotes")
```

Für die Spalte Flag and Footnotes kommen einige Fehler beim Einlesen. Da sie nicht essentiell ist, löschen wir sie vom dataframe. Flags und Footnotes sollten nicht ignoriert werden, aber den besten Überblick bekommt man, wenn man sich die Einträge auf der eurostat Homepage anschaut. Die neuesten Werte waren zB nur geschätzt, deshalb habe ich sie für den Download nicht ausgewählt.

langes Format

Die Daten sind zwar in einem langen Format, aber die Untergliederungen sind auf mehrere Variablen aufgeteilt (AGE und DURATION) und alle Werte sind in der Spalte `Value`.

Altersgruppen AGE:

- Y_LTE3 Jünger als 3 Jahre
- Y3-CSA 3 bis Schulalter
- CSA-Y12 Schulalter bis 12 Jahre

Betreuungszeiten DURATION:

- H0 Null Stunden
- H1-29 1 bis 29 Stunden
- H_GE30 30 Stunden und mehr

Das ist kein tidy Dataset, den wir so verwenden können.

langes Format

Show entries

Search:

TIME	GEO	AGE	DURATION	Value
2005	BE	Y_LT3	H0	59
2005	BE	Y_LT3	H1-29	23
2005	BE	Y_LT3	H_GE30	19
2005	BE	Y3-CSA	H0	2
2005	BE	Y3-CSA	H1-29	50
2005	BE	Y3-CSA	H_GE30	48
2005	BE	CSA-Y12	H0	

Showing 1 to 7 of 3,528 entries

Previous

1

2

3

4

5

...

504

Next

langes Format umformen - spread()

Wir müssen eine Spalte schaffen die alle Variablennamen beinhaltet. Im nächsten Schritt müssen die Werte aus `Value` auf die Variablen verteilt werden. Das geht mit `spread()`, damit machen wir den dataframe breiter.

Nach der Umwandlung haben wir mehrere Variablen: `GEO`, `TIME` und alle Kombination von `AGE` & `DURATION` in eigenen Variablen.

Zuerst erstellen wir mit `mutate()` und `paste()` eine neue Variable, die einfach die Werte von `AGE` und `DURATION` mit `_` zusammenklebt. Dann löschen wir `AGE` und `DURATION` weil wir sie nicht mehr benötigen.

```
data_care_age_hours <- data_care_age_hours %>%  
  # data defined per %>%  
  mutate(var_names = # new variable "var_names"  
    paste(AGE, DURATION, # set variables for composition  
          sep = "_")) %>% # spaceholder between values  
  select(-AGE, -DURATION) # delete AGE and DURATION variables
```

langes Format umformen - spread()

Show entries

Search:

TIME	GEO	Value	var_names
2005	BE	59	Y_LT3_H0
2005	BE	23	Y_LT3_H1-29
2005	BE	19	Y_LT3_H_GE30
2005	BE	2	Y3-CSA_H0
2005	BE	50	Y3-CSA_H1-29
2005	BE	48	Y3-CSA_H_GE30
2005	BE		CSA-Y12_H0

Showing 1 to 7 of 3,528 entries

Previous

1

2

3

4

5

...

504

Next

langes Format umformen - spread()

Jetzt haben wir das Format, dass wir für `spread()` benötigen.

```
data_care_age_hours <- data_care_age_hours %>% # data defined per %>%  
  spread(key = "var_names", # set variable containing names of new va  
         value = "Value")   # set variable containing values of new v
```

Syntax von `spread()`:

1. Das dataframe definieren, hier per `%>%` (kann auch direkt in `spread()` an erster Stelle gemacht werden).
2. Per `key = "var_names"` die Variable definieren, welche die neuen Variablennamen beinhaltet.
3. Per `value = "Value"` die Variable definieren, welche die Werte der neuen Variablen beinhaltet.

Wir haben AGE und DURATION schon gelöscht, aber per `-AGE` und `-DURATION` in `spread()` hätten wir auch die Spalten löschen können. Alle nicht genannten Spalten also GEO und TIME werden kopiert.

languages tidy Format

Show entries

Search:

TIME	GEO	CSA- Y12_H_GE30	CSA- Y12_H0	CSA- Y12_H1- 29	Y_LT3_H_GE30	Y_LT3_H
2005	AT	32	2	66	0	
2005	BE	56		44	19	
2005	BG					
2005	CY	47	1	53	12	
2005	CZ	45	2	53	0	
2005	DE	26	16	58	8	
2005	DK	65	1	34	60	

Showing 1 to 7 of 392 entries

Previous

1

2

3

4

5

...

56

Next

langes tidy Format

Jetzt haben wir ein dataframe zum Anteil der Kinder unterschiedlicher Altersgruppen (3 Gruppen) in formaler Kinderbetreuung nach Betreuungszeiten (3 Gruppen).

- Daher haben wir 9 neue Variablen mit allen Kombination von Altersgruppen und Betreuungszeiten.
- Zusätzlich haben wir die GEO Variable mit den Ländernamen und TIME mit den Jahren.

Am besten wir benennen die Variablen noch um, so dass die Namen besser zum Inhalt passen:

- Sie können mit `rename(neuer Name = alter Name)` Variablennamen wählen die für Sie am verständlichsten sind.
- Wir hätten die Umbenennung auch vor `spread()` machen können als die Betreuungszeiten und Altersgruppen noch Werte waren. Das funktioniert mit `recode(variablennamen, alter Wert = neuer Wert)`. Nach `paste()` und `spread()` sind die Werte dann Variablennamen.

Visualisierung Einzelvariablen

World Value Survey Daten aufbereiten

Die WVS-Zeitreihedaten von 1981 bis 2016 finden Sie auf
<http://www.worldvaluessurvey.org/WVSDocumentationWVL.jsp>

WVS stellt die Daten im .RDS-Format zu Verfügung - Einlesen ist daher keine Problem.

Der Datensatz beinhaltet die Antworten von Einzelpersonen aus 113 Länder für 6 Wellen, daher besteht er aus 50.779 Beobachtungen (Zeilen) und 1.261 Variablen - **der Datensatz ist riesig**.

Für den Kurs werden wir den Datensatz auf ein Befragungswelle (wave 6), zwei Länder (DEU, SWE) und wenige Variablen reduzieren (den reduzierten Datensatz finde Sie auf moodle).

Leider ist Österreich nicht im WVS-Datensatz.

Im Codebook finden Sie die Beschreibung für jede Variablen.

World Value Survey Daten aufbereiten

Für unsere Einheit hab ich folgende Variablen ausgewählt:

- S003: country/region
- S007: unified respondent number ID
- S017: weight
- S019: equilibrated weight-1500
- X001: sex/gender
- X003: age
- X007: marital status
- X011: How many children do you have?
- X025A: Highest educational level attained
- X047: Scale of incomes
- C001: Jobs scarce: Men should have more right to a job than women
- D057: Being a housewife just as fulfilling
- D063_B: Job best way for women to be independent(b)

WVS Datensatz reduzieren

- Für die Datenmanipulation und die Visualisierung benötigen wir Pakete aus dem tidyverse.
- Alle Werte zwischen [-99, -1] im WVS sind fehlende Werte (Im Codebook werden die genauen Inhalte der unterschiedlichen fehlende Werte erläutert).
- Deshalb brauchen wir `replace_with_na_all()` aus dem naniar Paket (Mehr Infos: <https://cran.r-project.org/web/packages/naniar/vignettes/replace-with-na.html>).

WVS Datensatz reduzieren

```
# load packages & import data
wvs_data <- readRDS("_raw/F00008390-WVS_Longitudinal_1981_2016_r_v20:
library(tidyverse)
library(nanianr)
# filter countries & wave
wvs_data <- wvs_data %>%
  filter(S003 %in% c("276", "752") & S002 == 6) %>%
  select(S003, S007, S017, S019,
         X001, X003, X007, X011, X025A, X047,
         C001, D057, D063_B) %>%
  mutate(S003 = recode(S003, "276" = "DEU", "752" = "SWE")) %>%
  replace_with_na_all(condition = ~.x <= -1)
# all values equal and smaller than -1 are defined as NA for all var
saveRDS(wvs_data, file = "data/wvs_short.RDS")
# optional save new data set
```

Verteilung von Einzelvariablen visualisieren

Für die Visualisierung von Einzelvariablen verwenden wir die Variable C001
Jobs scarce: Men should have more right to a job than women:

- 1:Agree
- 2:Disagree
- 3:Neither
- -5:Missing; Unknown
- -4:Not asked in survey
- -3:Not applicable
- -2:No answer
- -1:Don´t know

Balkendiagramm geom_bar

Die geom_function für ein Balkendiagramm ist geom_bar(). Da wir nur eine Variable darstellen, können wir entweder aes(x = C001) oder aes(y = C001) verwenden (auf welcher Achse die Balken angezeigt werden sollen).

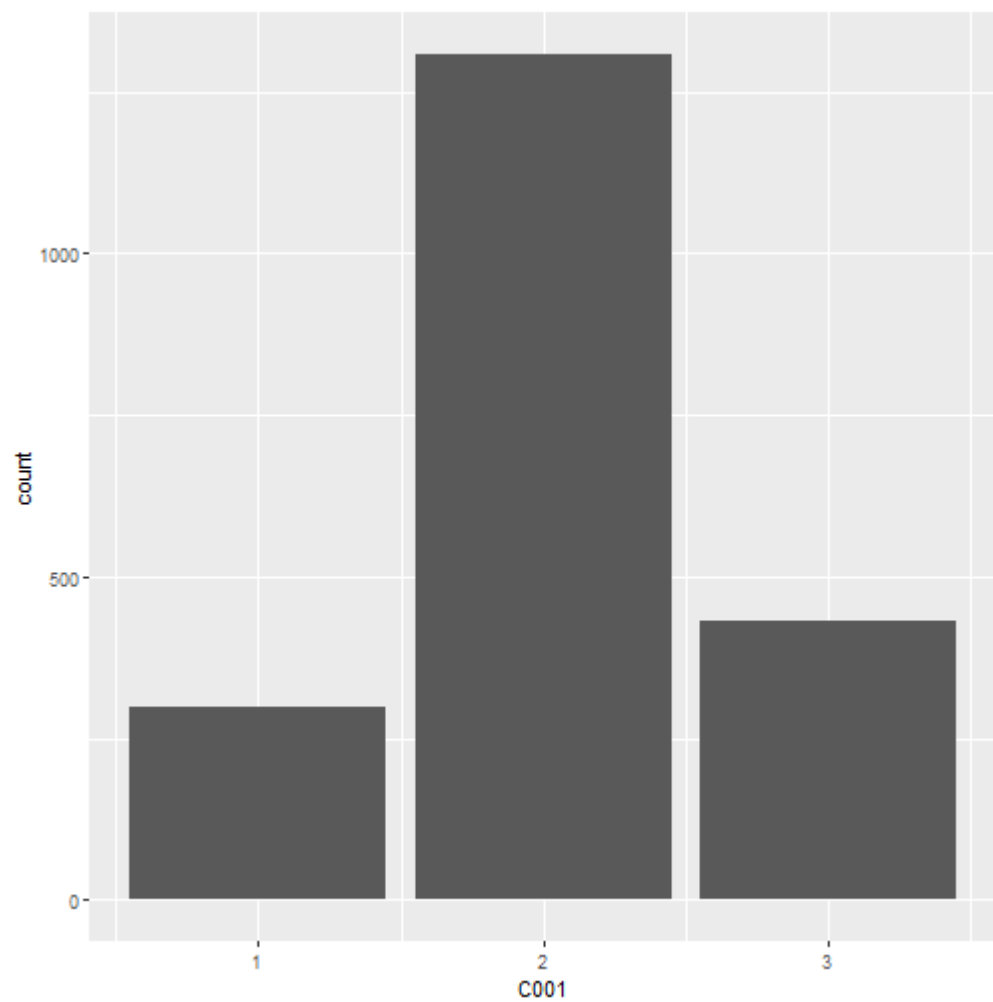
```
wvs_data %>%  
  filter(S003 == "DEU") %>%  
  ggplot(aes(x = C001)) +  
  geom_bar()
```

Syntax:

- filter() um Daten für Deutschland auszuwählen.
- ggplot() Daten werden per %>% definiert, somit haben wir die gefilterten Beobachtungen.
- ggplot(aes(x = C001)) Variable für die x-Achse kann direkt in ggplot() aber auch in geom_bar() festgelegt werden. aes() nicht vergessen!

Fehlermeldung bedeutet, dass 11 Beobachtungen entfernt wurden, weil sie fehlende Werte haben.

```
## Warning: Removed 11 rows containing non-finite values (stat_count).
```



Index erstellen

Für die Verwendung eines Histogramms benötigen wir eine kontinuierliche numerische Variable. Leider gibt es im WVS nur wenige numerische Variablen, da die meisten Surveyfragen mit einer Likert-Skala oder einer Nominal-Skala beantwortet werden.

Daher bilden wir einen Index aus den unterschiedlichen Variablen zur Einstellung bezüglich der Erwerbsarbeit von Frauen (Erklärung & Code auf den nächsten zwei Seiten).

Die Erstellung eines Indexes mit `scale()` und die dazugehörigen Plots finden Sie am Ende der Präsentation

Vor der Bildung des Indexes passen wir die Werte so an, dass die Spannweite jeder Variable 0-1 ist. Dafür müssen wir die Variablen manipulieren:

1. Bei C001 ist die Reihenfolge der Antwortitems: agree, disagree und neither, wohingegen ist bei D063_B die Reihenfolge agree, neither und disagree. Haben beide neither die gleiche Bedeutung? Man könnte entweder die Reihenfolge bei C001 ändern, so dass neither Platz 2 hat. Oder wir definieren neither als NA, da es nicht die gleiche Bedeutung wie bei D063_B hat. Per `replace_with_na(replace = list(C001 = 3))` definieren wir neither als NA.
2. Subtrahieren wir 1 von jeder Variable, so dass die Spannweite 0 bis höchstes Antwortitem minus 1 ist.
3. Dividieren wir jede Variable durch die Zahl des höchsten theoretischen Antwortitems minus 1 (1 bei C001, 3 bei D507 und 2 bei D063_B). Somit haben wir die Spannweite 0 bis 1 bei jeder Variable.
4. Die Aussagerichtung von C001 und D507 ist je höher der Wert, desto negativer ist die Einstellung bezüglich der Erwerbsarbeit von Frauen. Deshalb müssen wir die Antworten umdrehen, so dass eine höhere Antwort eine positivere Einstellung zur Erwerbsarbeit von Frauen darstellt. Das machen wir per Subtraktion von 1 und Multiplikation mit -1.

Um den Index zu erstellen, addieren wir alle Werte und teilen sie durch die Anzahl der Indexteilvariablen (hier 3).

Index erstellen

Umso positiver der Wert des Indexes, umso stärker ist die Zustimmung, dass Hausfrauen-sein *nicht* erfüllend ist, dass Männer auf dem Arbeitsmarkt *nicht* bevorzugt werden sollen und dass ein Beruf die Unabhängigkeit von Frauen fördert.

```
wvs_data_scale <- wvs_data %>%  
  # answer "neither" transformed to NA  
  replace_with_na(replace = list(C001 = 3)) %>%  
  # standardise each variable  
  mutate(C001 = (((C001 - 1) / 1) - 1) * -1 ) %>%  
  mutate(D057 = (((D057 - 1) / 3) - 1) * -1 ) %>%  
  mutate(D063_B = (D063_B - 1) / 2) %>%  
  # create women_index variable  
  mutate(women_index = (C001 + D057 + D063_B) / 3)
```

Histogramm geom_histogram

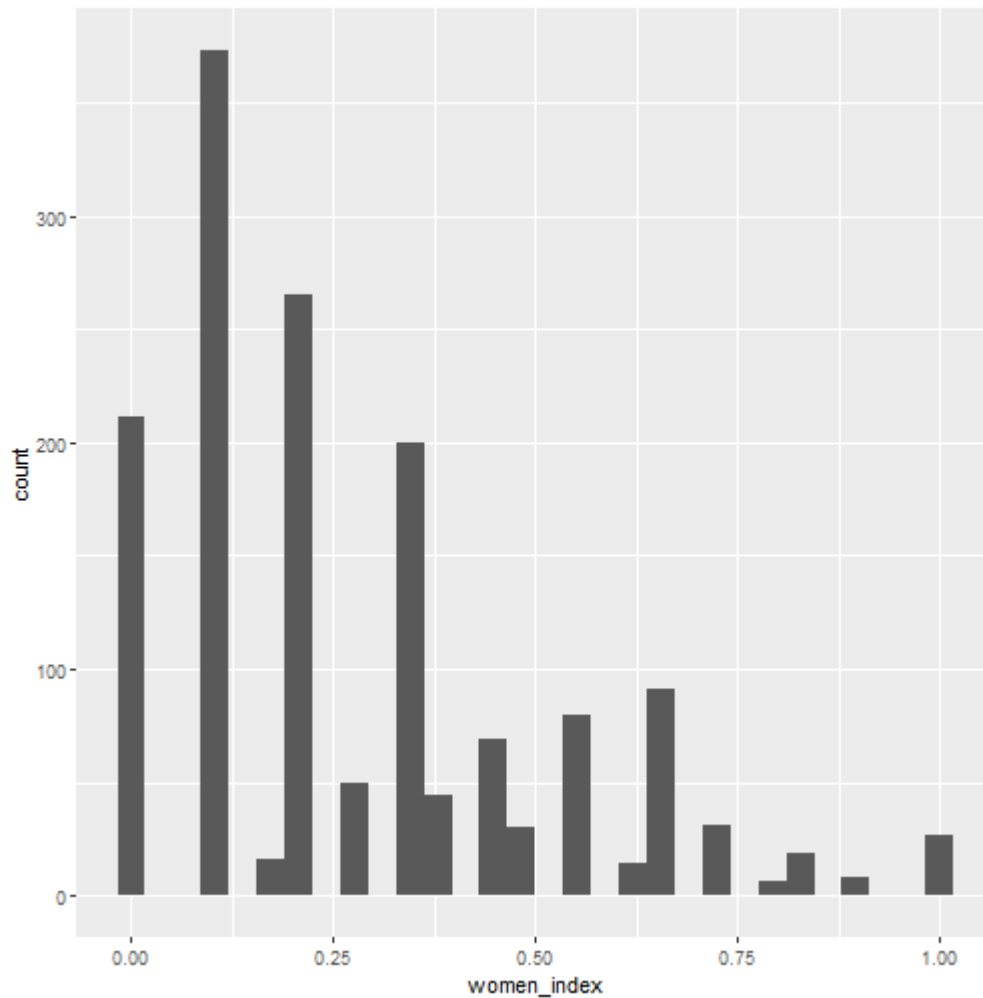
Histogramm für unsere neue Indexvariable zur Einstellung bezüglich der Erwerbsarbeit von Frauen. Wir zeigen nur die Antworten aus Deutschland.

```
wvs_data_scale %>%  
  filter(S003 == "DEU") %>%  
  ggplot(aes(x = women_index)) +  
  geom_histogram()
```

Die Syntax ist fast gleich wie beim Balkendiagramm. Anstelle von `geom_bar()` nehmen wir `geom_histogram()`.

```
## stat_bin() using bins = 30. Pick better value with binwidth.
```

```
## Warning: Removed 512 rows containing non-finite values (stat_bin).
```



Histogramm geom_histogram

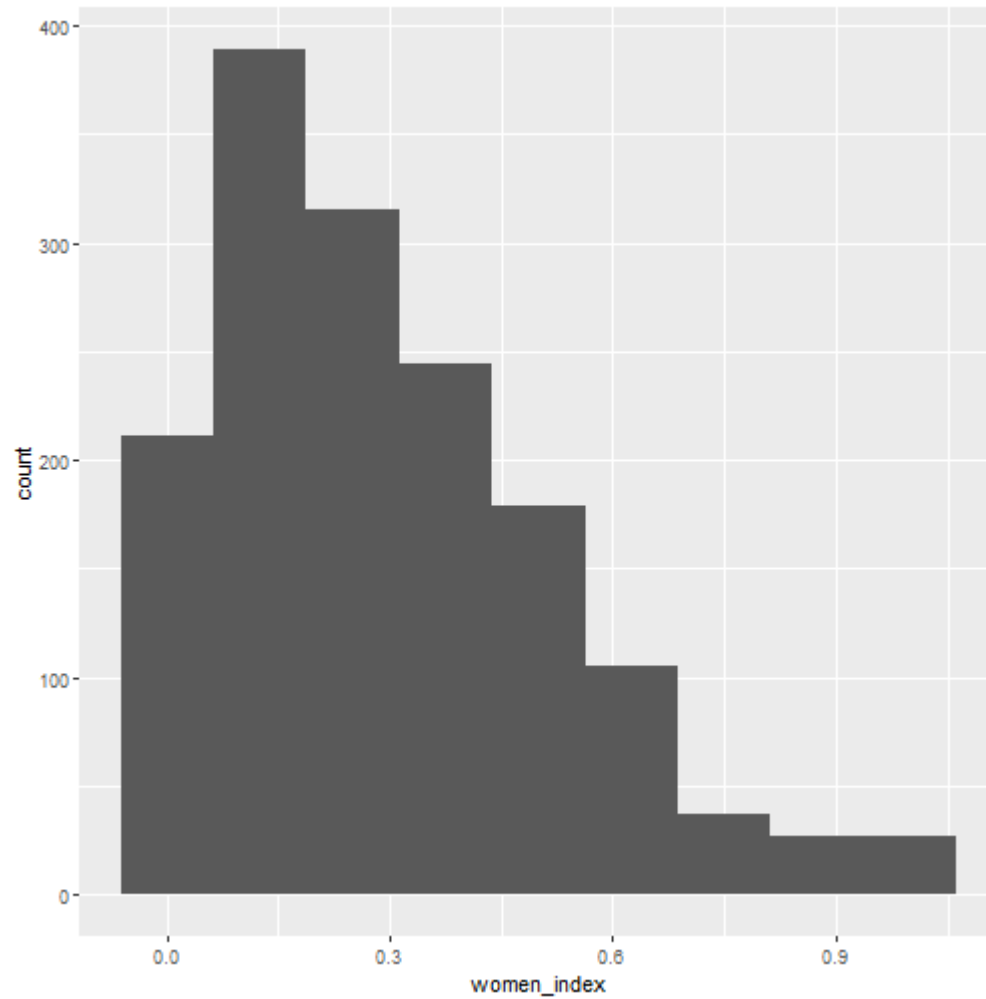
Das Ziel eines Histogrammes ist die Verteilung einer Variable darzustellen aber auch die Komplexität zu reduzieren.

R wählt die Säulenbreite und damit die Anzahl der Säulen automatisch. Wir können diese aber auch manuell festlegen mit `binwidth = Säulenbreite` in Einheit der Variable und somit die Komplexität reduzieren.

Wir wählen `binwidth = 0.125`, um 8 Säulen zu haben.

```
wvs_data_scale %>%  
  filter(S003 == "DEU") %>%  
  ggplot(aes(x = women_index)) +  
  geom_histogram(binwidth = 0.125) # bin width 0.125 = 8 bars
```

Warning: Removed 512 rows containing non-finite values (stat_bin).



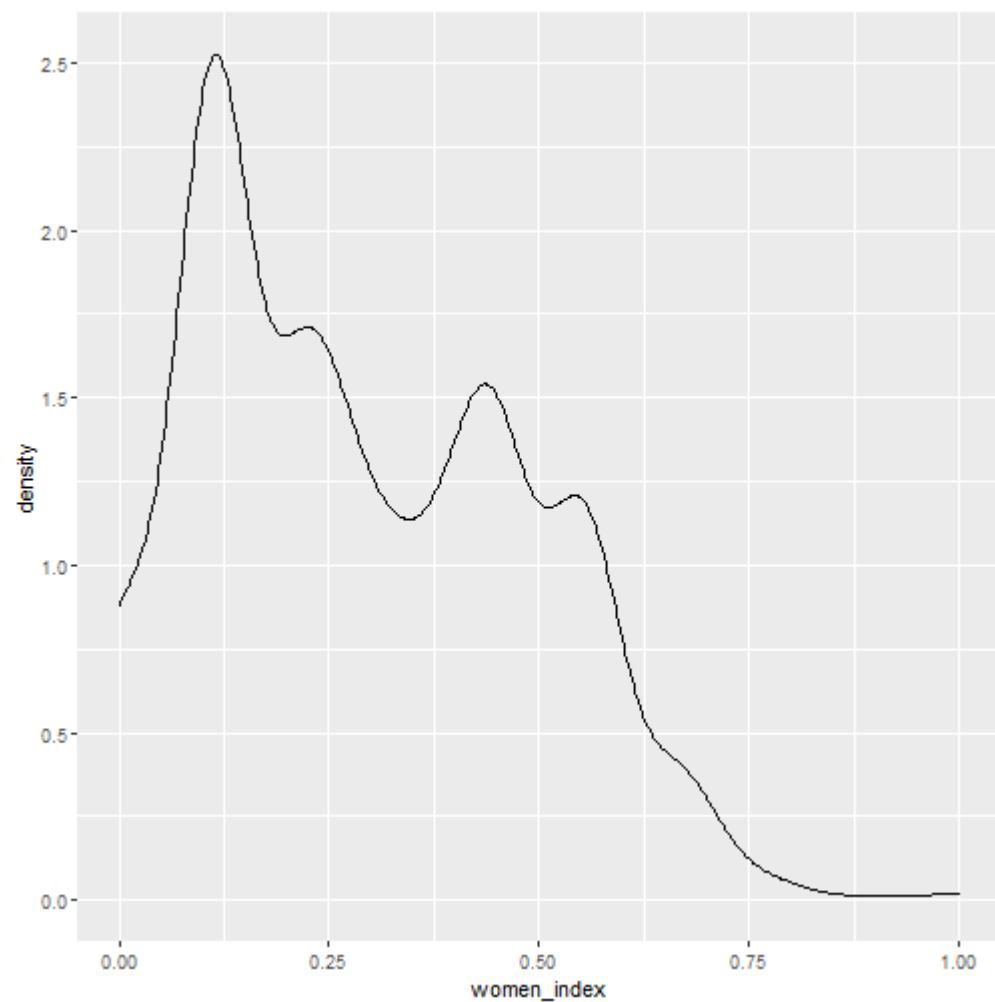
Visualisierung DichtepLOTS geom_density

Für DichtepLOTS gilt das gleiche wie für Histogramme: Sie sind am besten für kontinuierliche numerische Variablen geeignet. Daher nehmen wir unsere neue Indexvariable.

```
wvs_data_scale %>%  
  filter(S003 == "SWE") %>% # data for Sweden  
  ggplot(aes(x = women_index)) +  
  geom_density()
```

Syntax bleibt gleich, nur das wir `geom_density()` verwenden.

```
## Warning: Removed 256 rows containing non-finite values (stat_density).
```



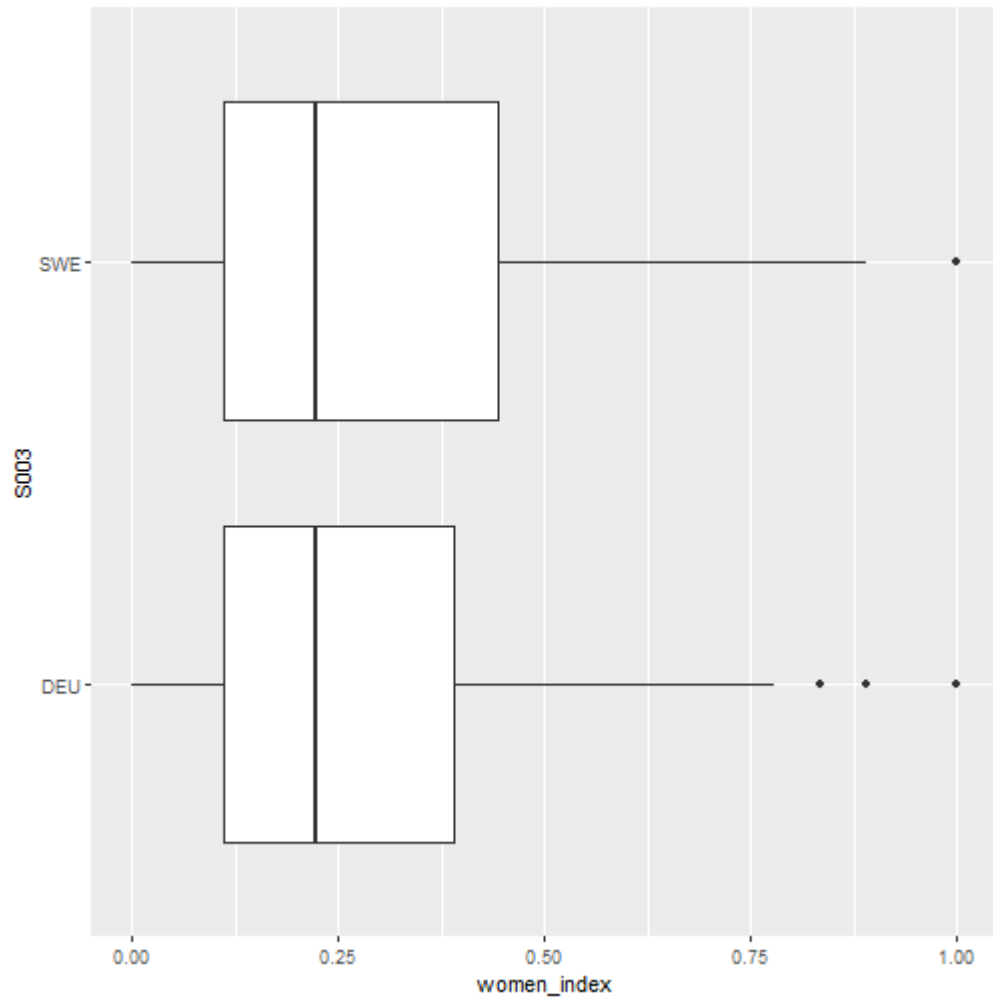
Boxplots

Die Verteilung der Indexvariable je Land können wir mit einem Boxplot vergleichen.

```
wvs_data_scale %>%  
  ggplot(aes(x = women_index,  
             y = S003)) + # y-axis: "DEU" & "SWE"  
  geom_boxplot()
```

Da wir für die y-Achse die Ländervariable S003 festlegen, bekommen wir je ein Boxplot pro Land.

```
## Warning: Removed 768 rows containing non-finite values (stat_boxplot).
```



Plots je Gruppe (Länder)

Letzte Woche haben wir gelernt, dass wir per `colour` = oder auch `shape` = Daten für unterschiedliche Gruppen anzeigen können.

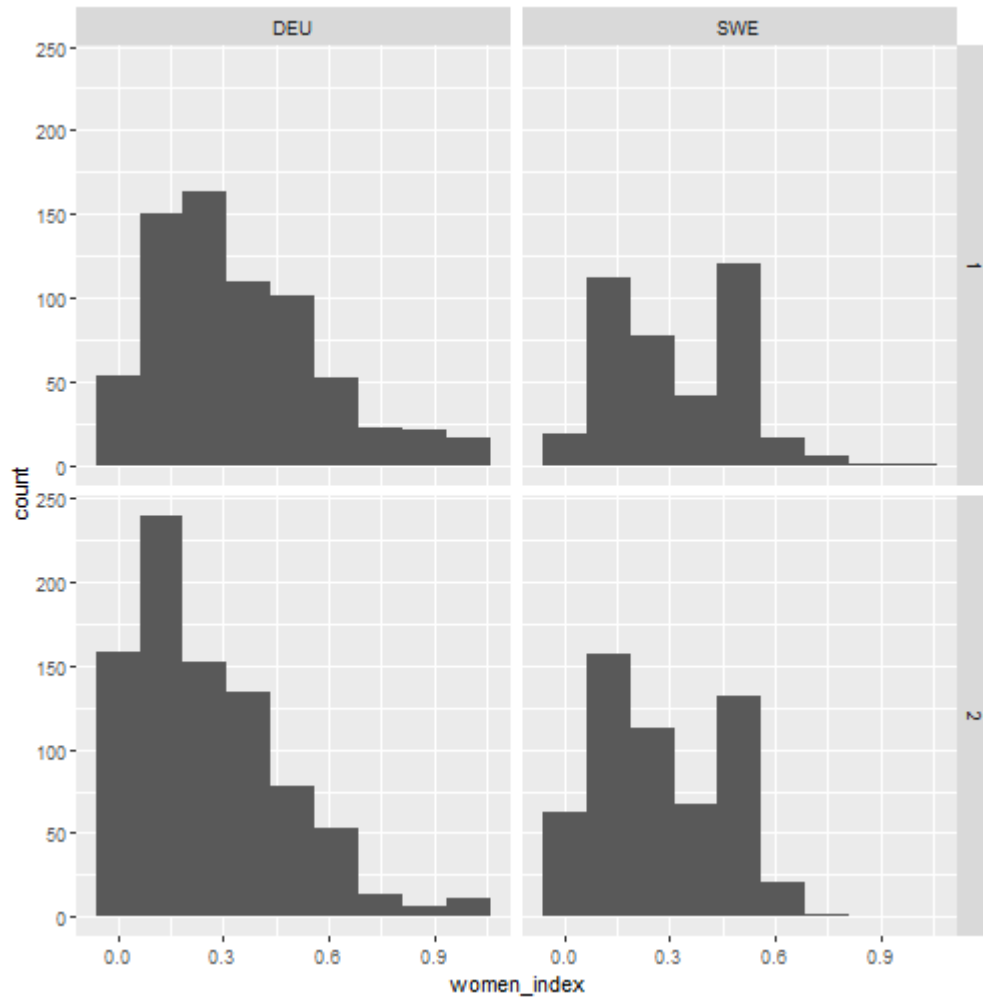
Mit `facet_grid()` können wir in einer Visualisierung je Gruppe einen Plot getrennt erstellen. ZB für Frauen und Männer getrennt (Reihe) sowie für Deutschland und Schweden (Spalten).

```
wvs_data_scale %>%  
  ggplot(aes(x = women_index)) +  
  geom_histogram(binwidth = 0.125) + # bin width 0.125 = 8 bars  
  facet_grid(cols = vars(S003), # "SWE" & "DEU" in separate columns  
             rows = vars(X001)) # men and women in separate rows
```

`facet_grid()` ist mit fast allen `geom_functions` kompatibel.

Es muss nicht beides `cols` = und `rows` = angegeben werden. Es reicht auch, wenn nur eins von beiden verwendet wird. Probieren Sie es einfach aus.

```
## Warning: Removed 768 rows containing non-finite values (stat_bin).
```



Schauen Sie im Codebook nach wie 1 und 2 für X001 definiert sind

Plots speichern ggsave()

Plots könne direkt im Vorschaufenster gespeichert werden. Einfach auf *Export > Save as image* klicken. Im neuen Fenster können Sie die Größe der Grafik, Dateinamen und Speicherort angeben.

Das Gleiche können Sie auch mit dem Befehl `ggsave()` machen:

- Falls kein Objektnamen angegeben wird, speichert `ggsave()` immer den zuletzt erstellten Plot.
- Den Dateinamen mit Dateiendung angeben. `filename` steht an erster Position in `ggsave()`.
- R speichert wie immer in der Working Directory. `Path` = kann der genau Dateipfad angegeben werden, wenn nicht dann wird direkt in der Working directory gespeichert.
- Falls keine Größe angegeben wird, wird automatisch die passende Größe ausgewählt.
- Per `width` = und `height` = kann das überschrieben werden (R wählt die Größeneinheit des Systems, diese kann per `units = c("in", "cm", "mm")` und `dpi` = überschrieben werden).

```
ggsave("women_index.png", path = "figures", width = 5, height = 5)
```

Übung 9

- Laden Sie sich den reduzierten WVS Datensatz `wvs_short.RDS` von moodle herunter.
- *Optional: Verwenden Sie den ursprünglichen Datensatz direkt von WVS und reduzieren Sie den Datensatz nach Ihren Wünschen (Länder, Welle, Variablen und fehlende Werte definieren).*
- erstellen Sie ein Boxplot für eine Variable Ihrer Wahl, entweder für Deutschland oder Schweden (oder selbstgewählte Länder).
- Erstellen Sie Boxplots für Schweden und Deutschland und Männer und Frauen (andere Variable nach Wahl) in einer Grafik mit `facet_grid()`.
- Laden Sie Ihr Skript und die Grafiken als .png bis zum 8. Mai 12:00 auf moodle hoch.

Optional: Falls Sie `gather()` und `spread()` testen wollen, dann schicken Sie mir Ihr Skript per Mail und ich gebe Ihnen gerne Feedback.

Falls Sie noch Fragen haben, nutzen Sie das **Forum** auf moodle und unterstützen Sie Ihre Kolleg*innen mit Ihrem Wissen!



Forum für R & RStudio Fragen

Hier können Sie alle Fragen, die Sie zu R und RStudio haben, stellen und auch Probleme diskutieren. Wir werden auf Ihre Fragen antworten. Bitte unterstützen Sie auch Ihre Kolleg*innen mit Ihrem Wissen. Falls Sie die Lösung für ein Problem haben, dann antworten Sie einfach unter der Frage ihrer Kolleg*in.

Nutzen Sie auch unsere **R Sprechstunde**.
Jeden Donnerstag von 11:00 bis 11:45 auf zoom oder im Anschluss an die zoom-Sitzung zu Familienpolitik (Link finden Sie auf moodle).

Archiv: Index erstellen mit sclale()

Bei dem folgenden Ansatz der Erstellung eines Indexes werden Variablen mit mehr Ausprägungen bevorzugt.

Index erstellen - mit scale()

Für die Verwendung eines Histogramms benötigen wir eine kontinuierliche numerische Variable. Leider gibt es im WVS nur wenige numerische Variablen, da die meisten Surveyfragen mit einer Likert-Skala oder einer Nominal-Skala beantwortet werden.

Daher bilden wir einen Index aus den unterschiedlichen Variablen zur Einstellung bezüglich der Erwerbsarbeit von Frauen.

- Vor der Bildung des Indexes standardisieren wir die Teilvariablen mit `scale()`.
- Die neuen Werte sind die Abweichung vom Mittelwert relativ zur Standardabweichung. Der Mittelwert der transformierten Variablen ist nun 0 und die Standardabweichung 1.
- Die Bildung des Index ist simple: Wir addieren die einzelnen Teilvariablen und ziehen die Werte von D063_B ab, da die Antworten nicht in die gleiche Aussagerichtung gehen.

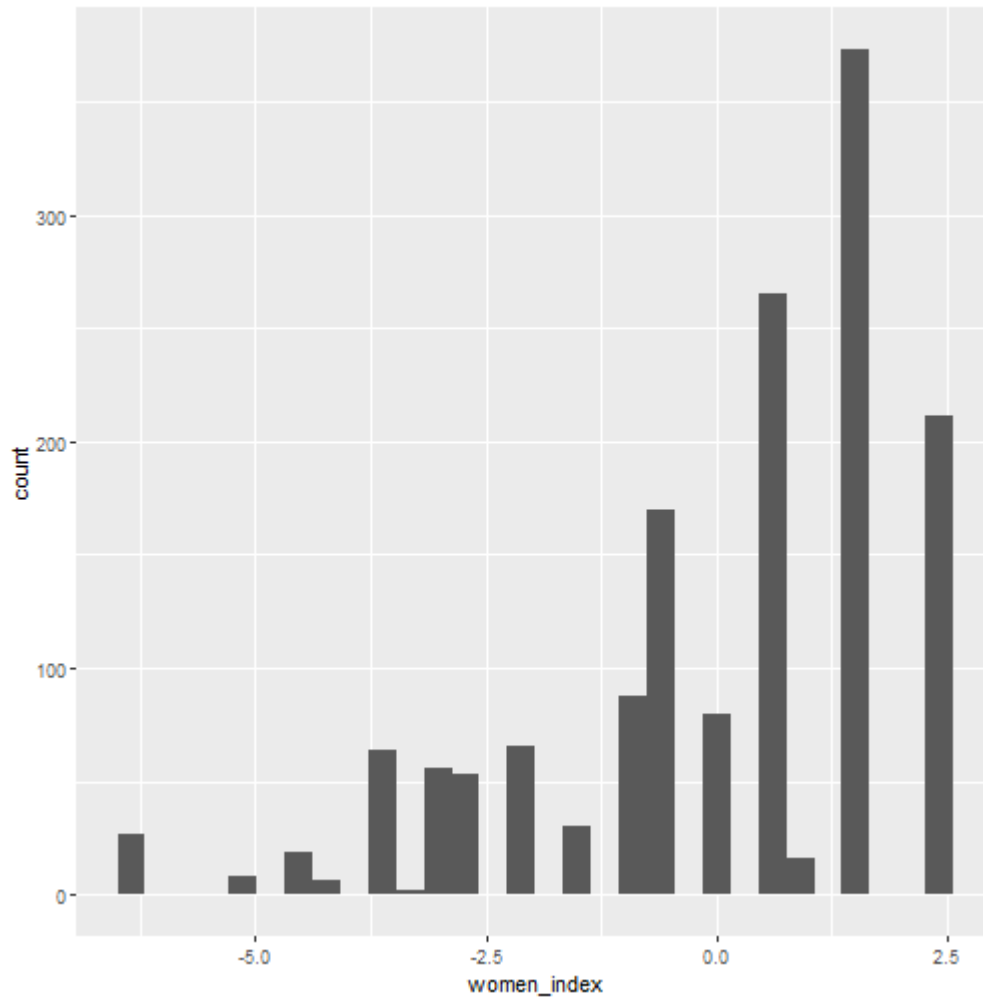
Umso negativer der Wert des Indexes umso mehr stimmen die Leute zu, dass Frauen Hausfrauen sein sollen, Männer auf dem Arbeitsmarkt einen Vorzug haben sollen und dass ein Beruf die Unabhängigkeit von Frauen **nicht** fördert (Variablenwerte umgedreht für D063_B, weil wir sie abziehen von den anderen).

Index erstellen - mit scale()

```
wvs_data_scale <- wvs_data %>%  
  # index creation for each country separately  
  group_by(S003) %>%  
  # answer "neither" transformed to NA  
  replace_with_na(replace = list(C001 = 3)) %>%  
  # standardise each variable  
  mutate(C001 = scale(C001)) %>%  
  mutate(D057 = scale(D057)) %>%  
  mutate(D063_B = scale(D063_B)) %>%  
  # ungrouping probably not necessary  
  ungroup() %>%  
  # create women_index variable  
  mutate(women_index = C001 + D057 - D063_B)
```

```
## stat_bin() using bins = 30. Pick better value with binwidth.
```

```
## Warning: Removed 512 rows containing non-finite values (stat_bin).
```



Histogramm geom_histogram

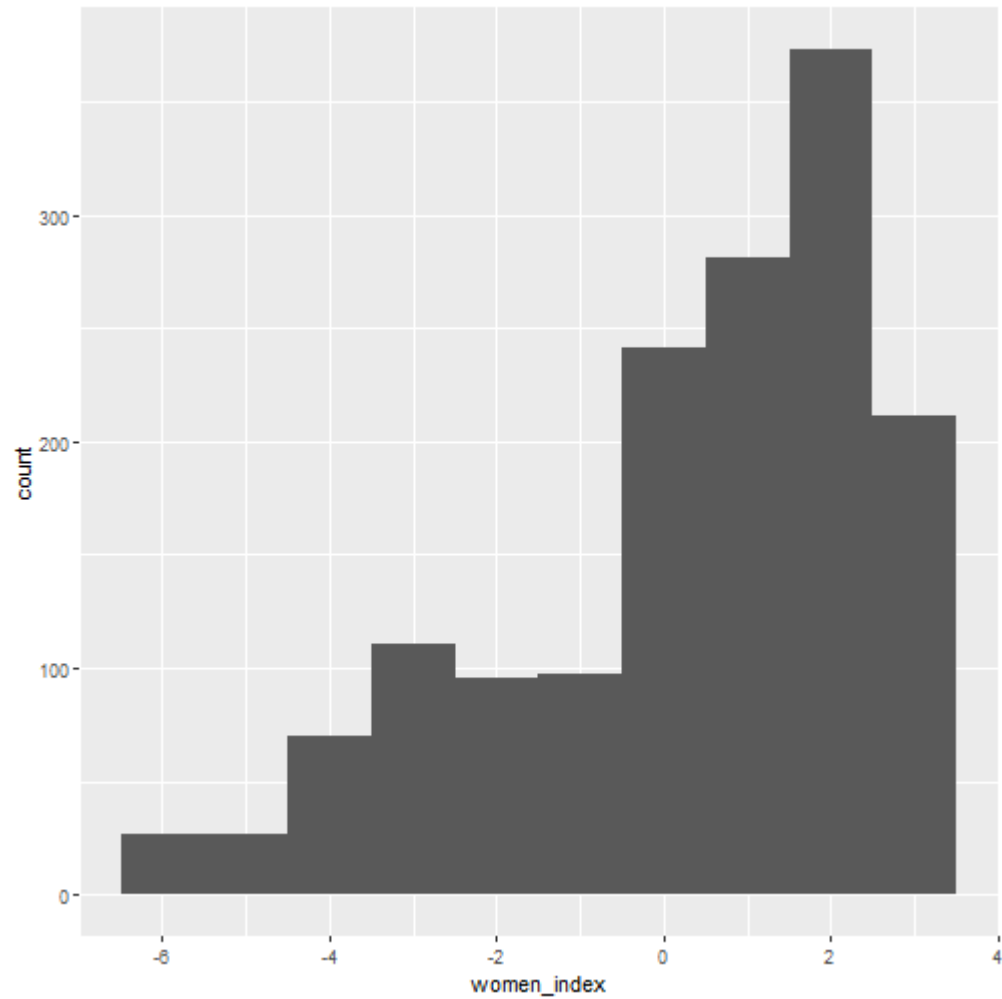
Das Ziel eines Histogrammes ist die Verteilung einer Variable darzustellen aber auch die Komplexität zu reduzieren.

R wählt die Säulenbreite und damit die Anzahl der Säulen automatisch. Wir können diese aber auch manuell festlegen mit `binwidth = Säulenbreite` in Einheit der Variable und somit die Komplexität reduzieren.

Wir wählen `binwidth = 1`, somit eine Standardabweichung.

```
wvs_data_scale %>%  
  filter(S003 == "DEU") %>%  
  ggplot(aes(x = women_index)) +  
  geom_histogram(binwidth = 1) # bin width 1 standard deviation
```

Warning: Removed 512 rows containing non-finite values (stat_bin).



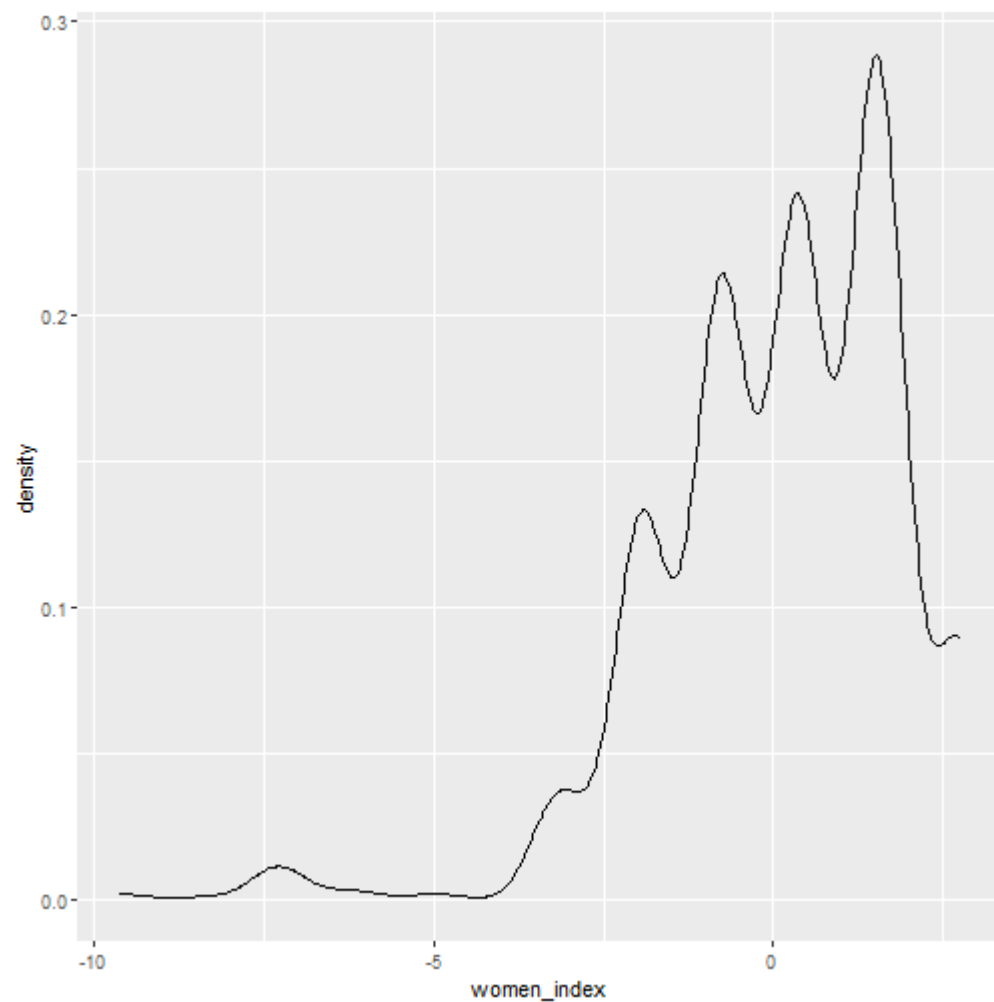
Visualisierung DichtepLOTS geom_density

Für DichtepLOTS gilt das gleiche wie für Histogramme: Sie sind am besten für kontinuierliche numerische Variablen geeignet. Daher nehmen wir unsere neue Indexvariable.

```
wvs_data_scale %>%  
  filter(S003 == "SWE") %>% # data for Sweden  
  ggplot(aes(x = women_index)) +  
  geom_density()
```

Syntax bleibt gleich, nur das wir `geom_density()` verwenden.

Warning: Removed 256 rows containing non-finite values (stat_density).



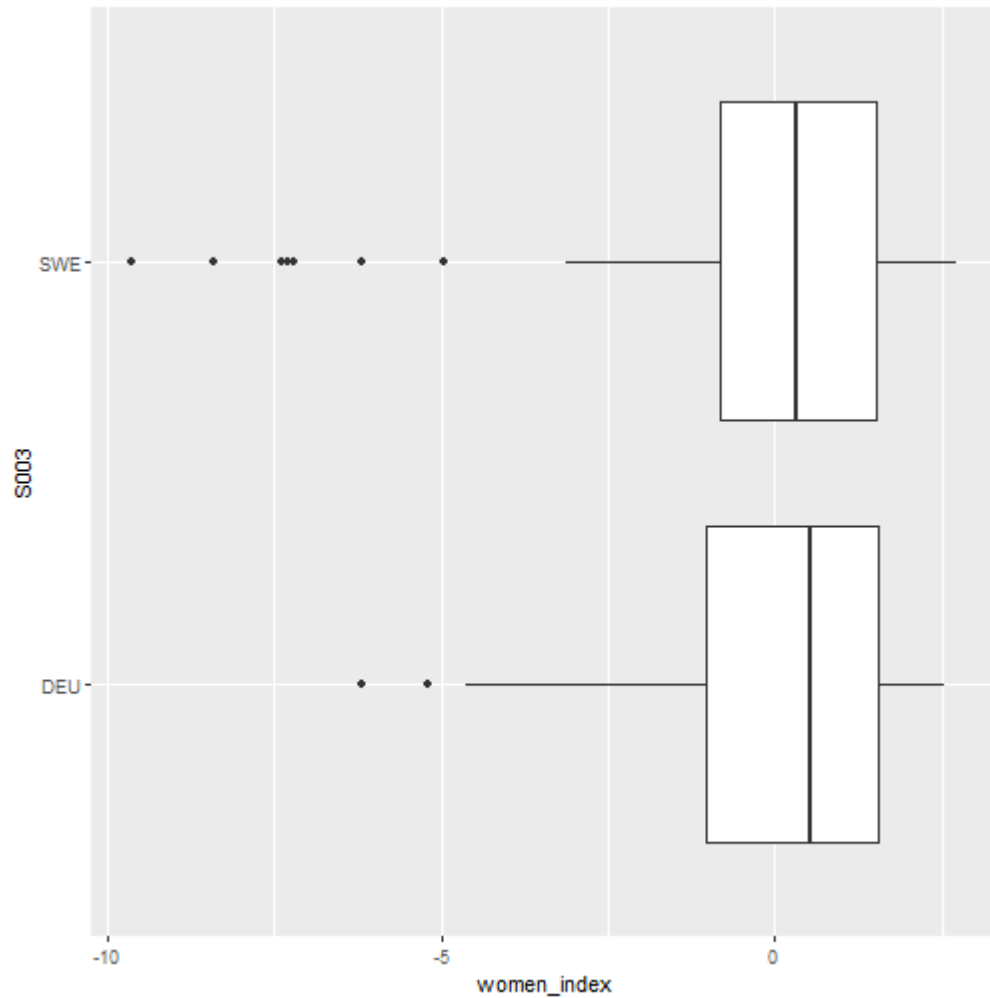
Boxplots

Die Verteilung der Indexvariable je Land können wir mit einem Boxplot vergleichen.

```
wvs_data_scale %>%  
  ggplot(aes(x = women_index,  
             y = S003)) + # y-axis: "DEU" & "SWE"  
  geom_boxplot()
```

Da wir für die y-Achse die Ländervariable S003 festlegen, bekommen wir je ein Boxplot pro Land.


```
## Warning: Removed 768 rows containing non-finite values (stat_boxplot).
```



Plots je Gruppe (Länder)

Letzte Woche haben wir gelernt, dass wir per `colour` = oder auch `shape` = Daten für unterschiedliche Gruppen anzeigen können.

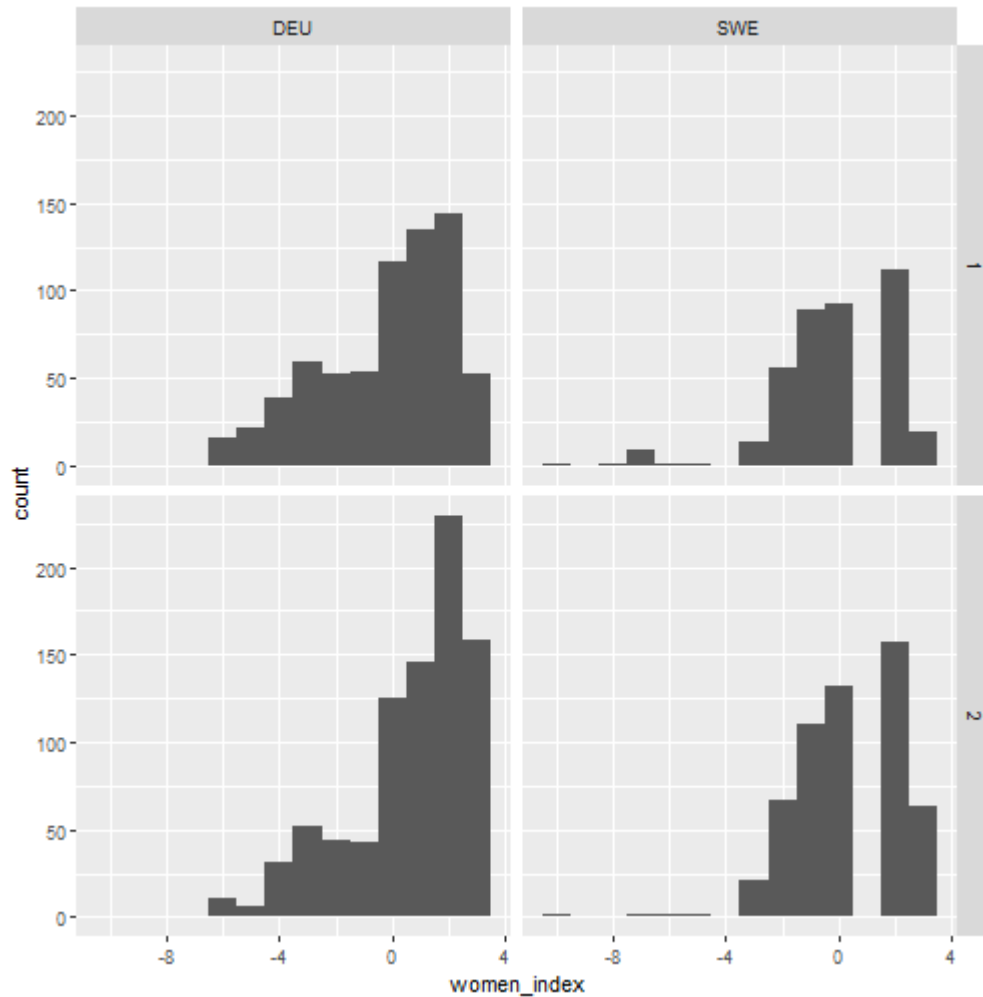
Mit `facet_grid()` können wir in einer Visualisierung je Gruppe einen Plot getrennt erstellen. ZB für Frauen und Männer getrennt (Reihe) sowie für Deutschland und Schweden (Spalten).

```
wvs_data_scale %>%  
  ggplot(aes(x = women_index)) +  
  geom_histogram(binwidth = 1) +  
  facet_grid(cols = vars(S003), # "SWE" & "DEU" in separate columns  
             rows = vars(X001)) # men and women in separate rows
```

`facet_grid()` ist mit fast allen `geom_functions` kompatibel.

Es muss nicht beides `cols` = und `rows` = angegeben werden. Es reicht auch, wenn nur eins von beiden verwendet wird. Probieren Sie es einfach aus.

Warning: Removed 768 rows containing non-finite values (stat_bin).



Schauen Sie im Codebook nach wie 1 und 2 für X001 definiert sind