Skript zur Vorlesung 6: Datenaufbereitung und -visualisierung

Claudius Gräbner

KW 49 2020

Contents

1 Einleitung **1**

2 Datenaufbereitung **1**
2.1 Einlesen von Daten 1
2.2 Data wrangling 3
2.3 Visualisierung 8

1 Einleitung

In diesem Dokument werden alle Abbildungen und Tabellen aus der sechsten Vorlesung repliziert. Dabei gebe ich der Info wegen *allen* R Code. Entsprechend sind bestimmt auch einige Befehle dabei, die Sie jetzt noch nicht kennen.

Zudem nehme ich an, dass im Arbeitsverzeichnis der Ordner `data/T6/` existiert und in diesem folgende Datensätze enthalten sind (diese sind über die Repository zur Vorlesung verfügbar): `wb_example.csv`, `wb_unemployment.csv`, `data/tidy/export_data.csv`, `export_daten_dt.csv`, `export_data.csv`, `bip-lebenserwartung.csv`, `government_openness.csv` und `data/tidy/export_daten.csv`.

Folgende Pakete werden zudem in diesem Skript verwendet:

```
library(tidyverse)
library(data.table)
library(ggpubr)
library(latex2exp)
library(icaeDesign)
library(here)
library(countrycode)
library(viridis)
```

Beachten Sie, dass das Paket icaeDesign nicht über die zentrale Paketverwaltung verfübar ist. Es muss folgendermaßen installiert werden:

```
devtools::install_github("graebnerc/icaeDesign")
```

2 Datenaufbereitung

2.1 Einlesen von Daten

```
datei_pfad <- here("data/T6/export_data.csv")
export_daten <- fread(datei_pfad)
head(export_daten)
```

```
##                 cgroup commoditycode         pci    exp_share
## 1:      Core countries           101  0.06424262 0.0001312370
## 2: Periphery countries           101  0.06424262 0.0004639794
## 3:      Core countries           102 -0.49254290 0.0005162508
## 4: Periphery countries           102 -0.49254290 0.0003700469
## 5:      Core countries           103  0.51082386 0.0005324995
## 6: Periphery countries           103  0.51082386 0.0004082251
```

Wie Sie sehen ist es immer besser die Spalten-Typen manuell festzulegen, denn der `commoditycode` wird sonst als Zahl interpretiert und führende Nullen entsprechend eliminiert:

```
typeof(export_daten[["commoditycode"]])
```

```
## [1] "integer"
```

```
datei_pfad <- here("data/T6/export_data.csv")
export_daten <- fread(datei_pfad,
                      colClasses = c("character", "character",
                                     "double", "double")
                      )
head(export_daten)
```

```
##                 cgroup commoditycode         pci    exp_share
## 1:      Core countries          0101  0.06424262 0.0001312370
## 2: Periphery countries          0101  0.06424262 0.0004639794
## 3:      Core countries          0102 -0.49254290 0.0005162508
## 4: Periphery countries          0102 -0.49254290 0.0003700469
## 5:      Core countries          0103  0.51082386 0.0005324995
## 6: Periphery countries          0103  0.51082386 0.0004082251
```

```
typeof(export_daten[["commoditycode"]])
```

```
## [1] "character"
```

Wenn Sie nicht-standardmäßige csv-Dateien einlesen wollen ist es sinnvoll die Spalten- und Kommatrenn-Zeichen explizit zu setzen.

Datei:

```
iso2c,year,Exporte
AT,2012,53.97
AT,2013,53.44
AT,2014,53.38
```

Einlesen mit Standardwerten kein Problem:

```
daten_pfad <- here("data/T6/export_daten.csv")
daten <- fread(daten_pfad)
daten
```

```
##    iso2c year Exporte
## 1:    AT 2012   53.97
## 2:    AT 2013   53.44
## 3:    AT 2014   53.38
```

Wenn 'deutsche Variante' mit `;` als Spalten- und `,` als Komma-Trennzeichen:

```
iso2c;year;Exporte
AT;2012;53,97
AT;2013;53,44
```

```
AT;2014;53,38
```

```
daten_pfad <- here("data/T6/export_daten_dt.csv")
daten <- fread(daten_pfad,
               colClasses = c("character", "double", "double"),
               sep = ";",
               dec = ","
               )
daten
```

```
##    iso2c year Exporte
## 1:    AT 2012   53.97
## 2:    AT 2013   53.44
## 3:    AT 2014   53.38
```

Um Dateien zu speichern verwenden Sie `fwrite()`:

```
test_data <- data.frame(
  Land = c("DEU", "DEU"),
  Jahr = c(2011:2012),
  BIP = c(1,2)
)

datei_name <- here("data/T6/test_data.csv")
fwrite(test_data, file = datei_name)
```

2.2 Data wrangling

Beispieldaten:

2.2.1 Breite und weite Datensätze

```
data_wide
```

```
##   Land  2013  2014
## 1   AT 5.335 5.620
## 2   DE 5.231 4.981
```

Daten können folgendermaßen 'länger' gemacht werden:

```
data_long <- pivot_longer(data = data_wide,
                          cols = one_of("2013", "2014"),
                          names_to = "Jahr",
                          values_to = "Arbeitslosenquote")
data_long
```

```
## # A tibble: 4 x 3
##   Land  Jahr  Arbeitslosenquote
##   <chr> <chr>             <dbl>
## 1 AT    2013               5.34
## 2 AT    2014               5.62
## 3 DE    2013               5.23
## 4 DE    2014               4.98
```

Um Daten 'breiter' zu machen gehen wir folgendermaßen vor:

```
data_wide_neu <- pivot_wider(data = data_long,
                             id_cols = one_of("Land"),
```

```
                              names_from = "Jahr",
                              values_from = "Arbeitslosenquote")
data_wide_neu
```

```
## # A tibble: 2 x 3
##   Land  `2013` `2014`
##   <chr>  <dbl>  <dbl>
## 1 AT      5.34   5.62
## 2 DE      5.23   4.98
```

Hier eine kombinierte Anwendung:

```
## # A tibble: 4 x 5
##   Land  Variable         `2012` `2013` `2014`
##   <chr> <chr>             <dbl>  <dbl>  <dbl>
## 1 AT    Exporte           54.0   53.4   53.4
## 2 AT    Arbeitslosigkeit   4.86   5.34   5.62
## 3 DE    Exporte           46.0   45.4   45.6
## 4 DE    Arbeitslosigkeit   5.38   5.23   4.98
```

Sie können mehrere Operationen übersichtlicher und kompakter schreiben wenn Sie die Pipe `%>%` verwenden. Sie wird über das Paket `magrittr` bereitgestellt, das auch Teil des `tidyverse` ist. Ohne die Pipe sähe der Code so aus:

```
data_al_exp_longer <- pivot_longer(data = data_al_exp,
                                   cols = -one_of("Land", "Variable"),
                                   names_to = "Jahr",
                                   values_to = "Wert")

data_al_exp_tidy <- pivot_wider(data = data_al_exp_longer,
                                id_cols = one_of("Land", "Jahr"),
                                values_from = "Wert",
                                names_from = "Variable")
```

Mit `%>%` lässt sich dies kompakter und lesbarer darstellen:

```
data_al_exp_tidy <- data_al_exp %>%
  pivot_longer(
    cols = -one_of("Land", "Variable"),
    names_to = "Jahr",
    values_to = "Wert") %>%
  pivot_wider(
    id_cols = one_of("Land", "Jahr"),
    values_from = "Wert",
    names_from = "Variable")
```

2.2.2 Datensätze kombinieren

Die Daten wurden manuell erstellt:

Anwendung von 'left_join()

```
data_bip_gini_left_join <- left_join(data_BIP, data_gini,
                                     by=c("Jahr"="year", "Land"="country"))
data_bip_gini_left_join
```

```
##   Jahr Land BIP Gini
## 1 2010  DEU   1    1
```

```
## 2 2011  DEU   2    2
## 3 2012  DEU   3   NA
## 4 2010  AUT   4   NA
## 5 2011  AUT   5   NA
## 6 2012  AUT   6    3
```

Anwendung von 'right_join()

```
data_gini_bip_right_join <- right_join(data_gini, data_BIP,
                                       by=c("year"="Jahr", "country"="Land"))
data_gini_bip_right_join
```

```
##   year country Gini BIP
## 1 2010     DEU    1   1
## 2 2011     DEU    2   2
## 3 2012     AUT    3   6
## 4 2012     DEU   NA   3
## 5 2010     AUT   NA   4
## 6 2011     AUT   NA   5
```

Anwendung von 'inner_join()

```
data_bip_gini_inner_join <- inner_join(data_BIP, data_gini,
                                       by=c("Jahr"="year", "Land"="country"))
data_bip_gini_inner_join
```

```
##   Jahr Land BIP Gini
## 1 2010  DEU   1    1
## 2 2011  DEU   2    2
## 3 2012  AUT   6    3
```

Anwendung von 'full_join()

```
data_bip_gini_full_join <- full_join(data_BIP, data_gini,
                                       by=c("Jahr"="year", "Land"="country"))
data_bip_gini_full_join
```

```
##   Jahr Land BIP Gini
## 1 2010  DEU   1    1
## 2 2011  DEU   2    2
## 3 2012  DEU   3   NA
## 4 2010  AUT   4   NA
## 5 2011  AUT   5   NA
## 6 2012  AUT   6    3
## 7 2013  AUT  NA    4
```

2.2.3 Filtern und Selektieren

Um Spalten zu selektieren verwenden wir `select`:

```
head(
  select(data_al_exp_tidy, Land, Exporte),
  2)
```

```
## # A tibble: 2 x 2
##   Land  Exporte
##   <chr>   <dbl>
## 1 AT       54.0
## 2 AT       53.4
```

Spalten können auch ausgeschlossen werden:

```
head(
  select(data_al_exp_tidy, -Exporte),
  2)
```

```
## # A tibble: 2 x 3
##   Land  Jahr  Arbeitslosigkeit
##   <chr> <chr>            <dbl>
## 1 AT    2012              4.86
## 2 AT    2013              5.34
```

Mit Hilfe von `one_of()` können wir Spalten über `character` ansprechen (zudem gibt R hier keinen Fehler aus wenn eine Spalte nicht existiert, daher auch der Name):

```
head(
  select(data_al_exp_tidy, one_of("Land", "Jahr")),
  2)
```

```
## # A tibble: 2 x 2
##   Land  Jahr
##   <chr> <chr>
## 1 AT    2012
## 2 AT    2013
```

Auch der Ausschluss von Spalten funktioniert mit `one_of()`:

```
head(
  select(data_al_exp_tidy, -one_of("Land", "Jahr")),
  2)
```

```
## # A tibble: 2 x 2
##   Exporte Arbeitslosigkeit
##     <dbl>            <dbl>
## 1    54.0             4.86
## 2    53.4             5.34
```

Bessere Formulierung:

```
data_al_exp_selected <- data_al_exp_tidy %>%
  select(one_of("Land", "Jahr", "Exporte"))
head(data_al_exp_selected, 2)
```

```
## # A tibble: 2 x 3
##   Land  Jahr  Exporte
##   <chr> <chr>   <dbl>
## 1 AT    2012     54.0
## 2 AT    2013     53.4
```

Um Zeilen zu filtern:

```
data_al_exp_filtered <- data_al_exp_tidy %>%
  filter(Land == "AT",
         Jahr > 2012)
data_al_exp_filtered
```

```
## # A tibble: 2 x 4
##   Land  Jahr  Exporte Arbeitslosigkeit
##   <chr> <chr>   <dbl>            <dbl>
## 1 AT    2013     53.4             5.34
```

```
## 2 AT    2014     53.4             5.62
```

Um Spalten umzubenennen:

```
data_al_exp_tidy %>%
  rename(country=Land,
         year_observation=Jahr,
         exports=Exporte,
         unemployment=Arbeitslosigkeit)
```

```
## # A tibble: 6 x 4
##   country year_observation exports unemployment
##   <chr>   <chr>              <dbl>        <dbl>
## 1 AT      2012                54.0         4.86
## 2 AT      2013                53.4         5.34
## 3 AT      2014                53.4         5.62
## 4 DE      2012                46.0         5.38
## 5 DE      2013                45.4         5.23
## 6 DE      2014                45.6         4.98
```

2.2.4 Daten ändern und zusammenfassen

Eine Spalte ändern:

```
unemp_data_wb <- unemp_data_wb %>%
  mutate(
    country = countrycode(country, "iso2c", "iso3c")
    )
head(unemp_data_wb, 2)
```

```
##    country year laborforce_female workforce_total population_total
## 1:     AUT 2010          46.13933         4276558          8363404
## 2:     AUT 2011          46.33455         4305310          8391643
```

Eine neue Spalte hinzufügen:

```
unemp_data_wb <- unemp_data_wb %>%
  mutate(
    workers_female_total = laborforce_female*workforce_total/100
    )
head(unemp_data_wb, 2)
```

```
##    country year laborforce_female workforce_total population_total
## 1:     AUT 2010          46.13933         4276558          8363404
## 2:     AUT 2011          46.33455         4305310          8391643
##    workers_female_total
## 1:              1973175
## 2:              1994846
```

Zusammenfassen:

```
unemp_data_wb_summarized <- unemp_data_wb %>%
  summarise(
    fem_workers_avg = mean(workers_female_total)
    )
unemp_data_wb_summarized
```

```
##   fem_workers_avg
## 1        10761223
```

Gruppiertes Zusammenfassen:

```
unemp_data_wb %>%
  group_by(country) %>%
  summarise(
    fem_workers_avg = mean(workers_female_total)
    ) %>%
  ungroup()
```

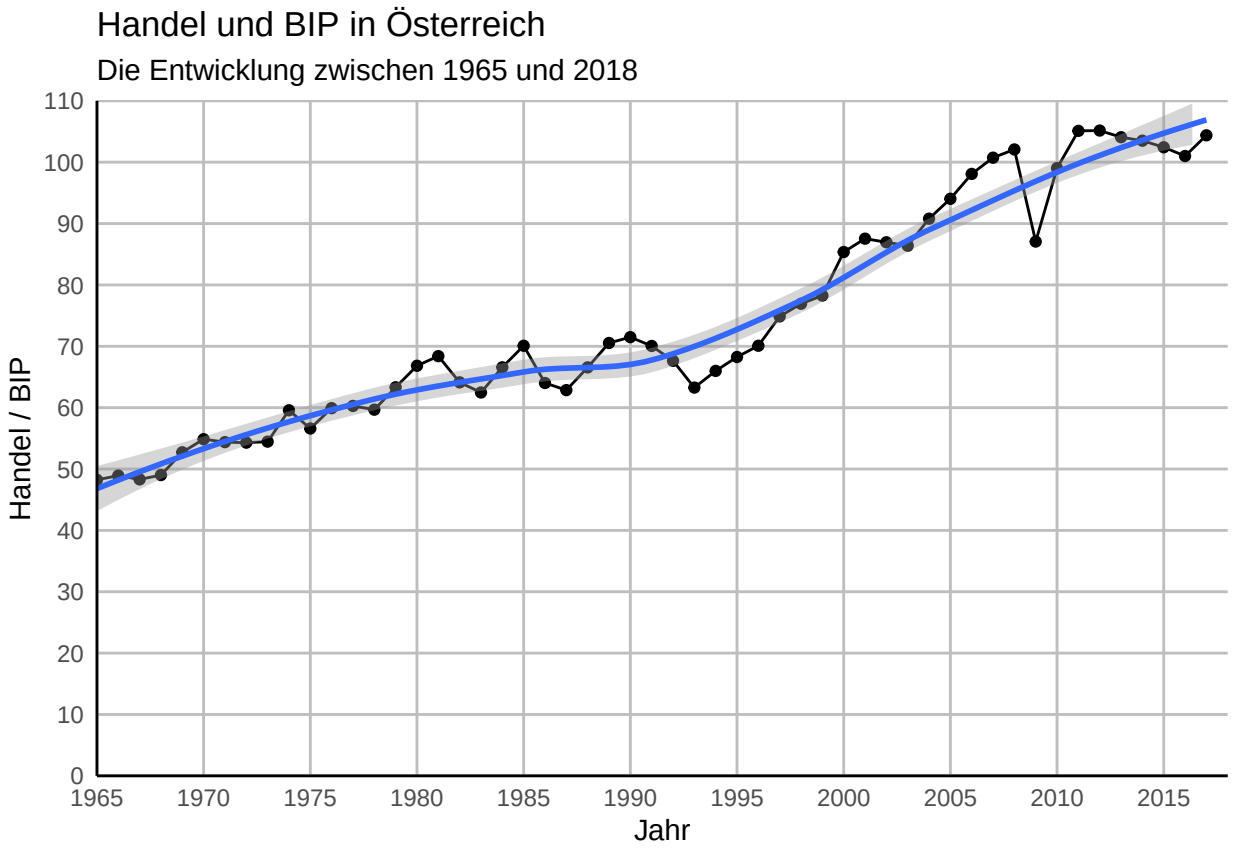
```
## # A tibble: 2 x 2
##   country fem_workers_avg
##   <chr>             <dbl>
## 1 AUT             2042685.
## 2 DEU            19479761.
```

2.3 Visualisierung

2.3.1 Laufendes Beispiel

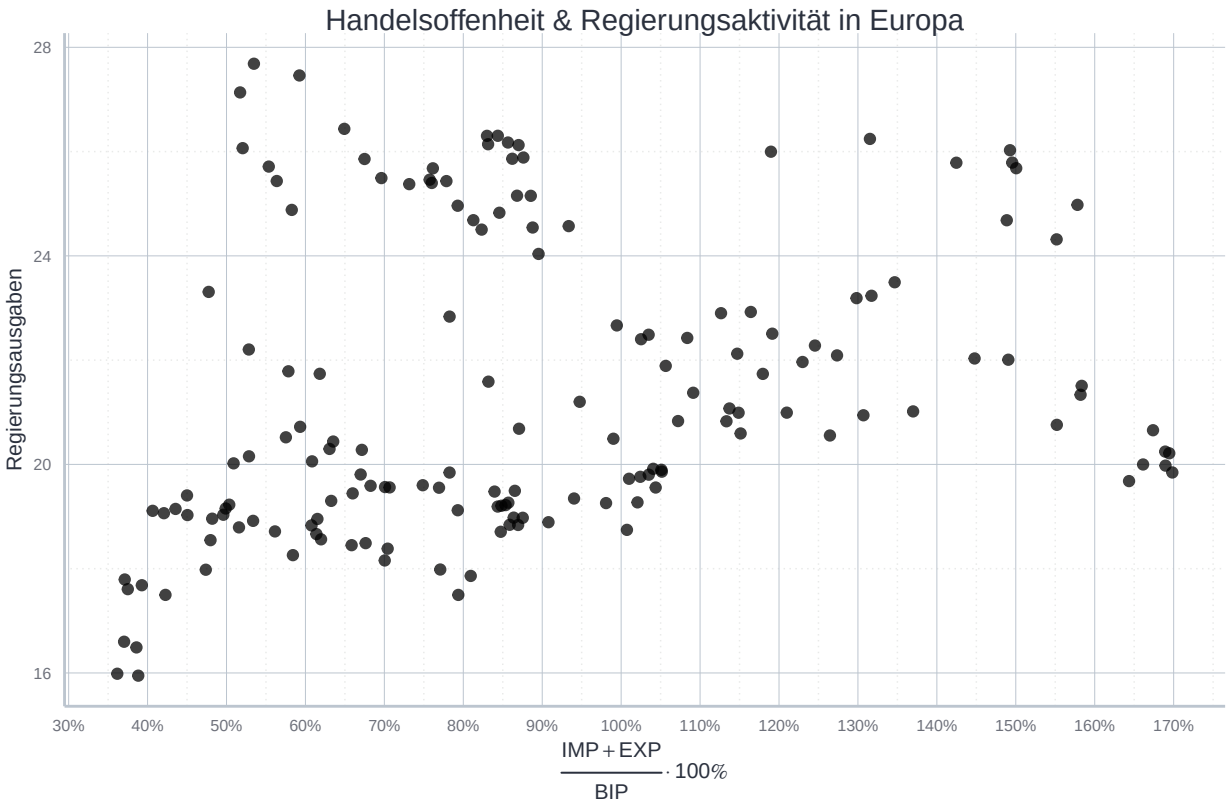
```
aut_trade <- fread(here("data/T6/government_openness.csv")) %>%
  select(iso3c, year, trade_total_GDP, gvnt_cons) %>%
  rename(Land=iso3c, Jahr=year,
         HandelGDP=trade_total_GDP,
         StaatsausgabenGDP=gvnt_cons) %>%
  select(Land, Jahr, HandelGDP) %>%
  filter(Land=="AUT")

aut_trade_plot <- ggplot(
  data = aut_trade,
  mapping = aes(x = Jahr,
                y = HandelGDP)
) +
  geom_line() +
  geom_point() +
  geom_smooth() +
  scale_x_continuous(
    limits = c(1965, 2018),
    breaks = seq(1960, 2017, 5),
    expand = c(0, 0)
  ) +
  scale_y_continuous(
    name = "Handel / BIP",
    limits = c(0, 110),
    breaks = seq(0, 110, 10),
    expand = c(0, 0)
  ) +
  ggtitle(
    label = "Handel und BIP in Österreich",
    subtitle = "Die Entwicklung zwischen 1965 und 2018"
    ) +
  theme(
    panel.background = element_rect(fill = "white"),
    panel.grid.major = element_line(colour = "grey"),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "black"),
```

```
    axis.ticks = element_blank()
  )
aut_trade_plot
```



2.3.2 Streudiagramm

```
streudiagramm <- ggplot(
  data = offenheits_daten,
  mapping = aes(x=trade_total_GDP,
                y=gvnt_cons)
  ) +
  geom_point(alpha=0.75) +
  scale_y_continuous(name = "Regierungsausgaben") +
  scale_x_continuous(name = TeX("$\\frac{IMP + EXP}{BIP}\\cdot 100\\%$"),
                     breaks = seq(30, 180, 10),
                     labels = scales::percent_format(accuracy = 1, scale = 1)
                     )+
  labs(
    title = "Handelsoffenheit & Regierungsaktivität in Europa",
    caption = "Quelle: Weltbank; Daten von 1990-2017."
    ) +
  theme_icae()
streudiagramm
```

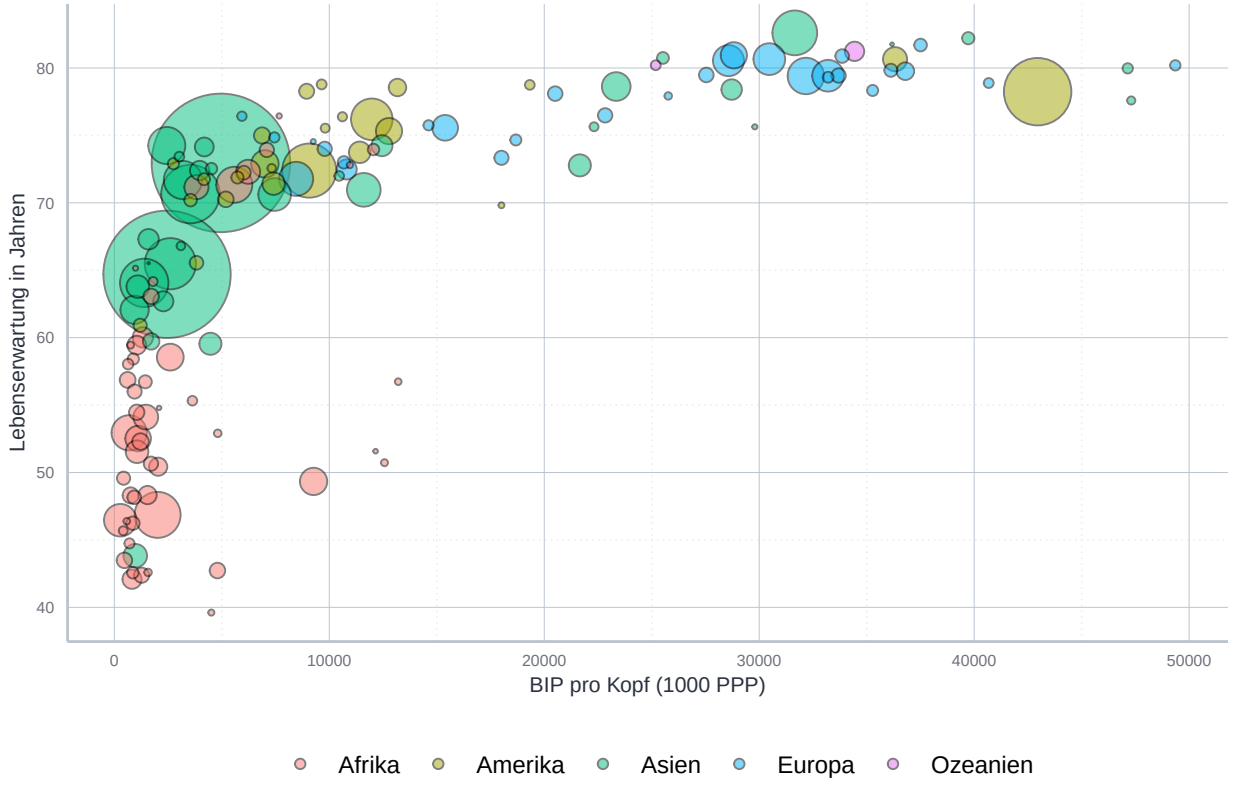


Quelle: Weltbank; Daten von 1990−2017.

2.3.3 Blasendiagramm

```
bubble_plot <- ggplot(
  data = bip_lebenserwartung_data,
  mapping = aes(x = gdpPercap,
                y = lifeExp,
                size = pop,
                fill = continent)
    ) +
  geom_point(
    alpha=0.5, shape=21, color="black"
    ) +
  scale_size(
    range = c(0.1, 24), name="Bevölkerung", guide = FALSE
    ) +
  scale_y_continuous(
    name = "Lebenserwartung in Jahren"
    ) +
  scale_x_continuous(
    name = "BIP pro Kopf (1000 PPP)"
    ) +
  labs(
    caption = "Hinweis: Größe der Blasen repräsentiert Bevölkerungsanzahl. Quelle: Gapminder."
    ) +
  theme_icae() +
  theme(
```

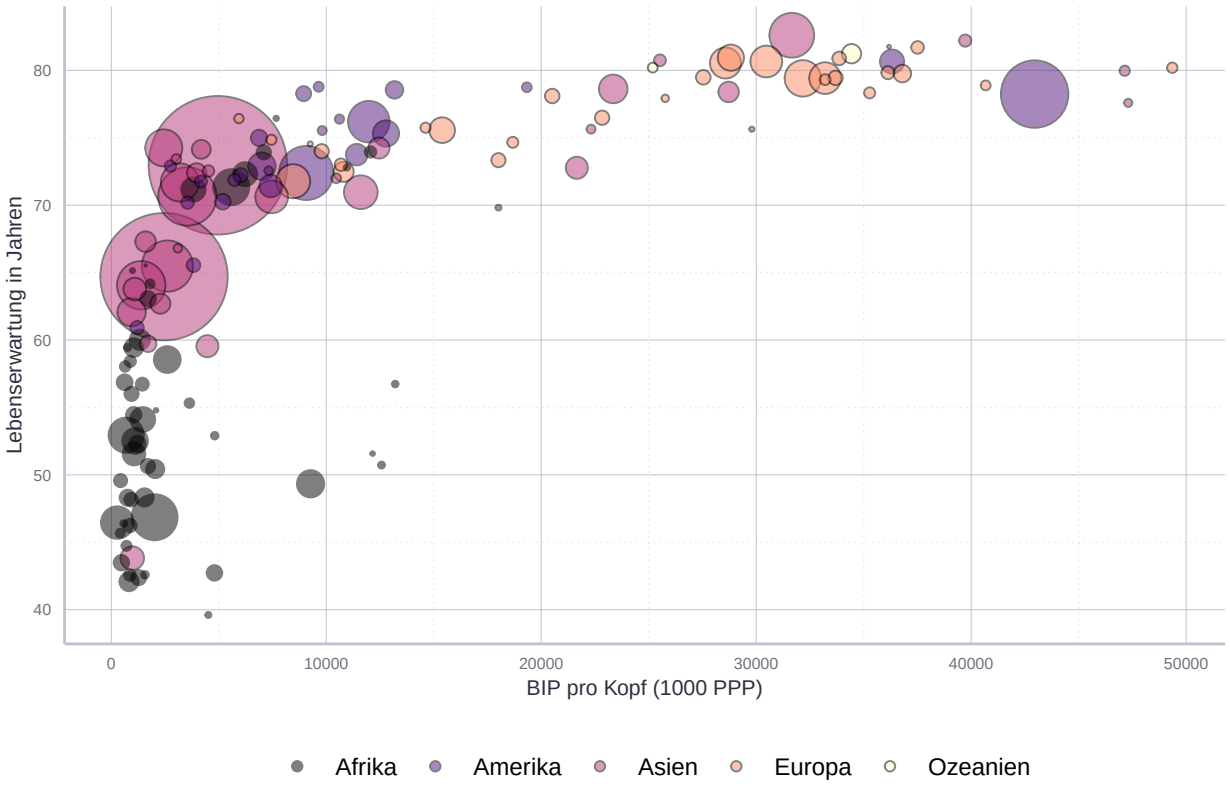
```
    legend.position="bottom",
    plot.caption = element_text(hjust = 0)
    )
bubble_plot
```



Hinweis: Größe der Blasen repräsentiert Bevölkerungsanzahl. Quelle: Gapminder.

Um das schöne Farbschema aus den Slides zu bekommen verwenden wir die Funktion `scale_fill_viridis()`, welche das schöne und gut lesbare Viridis-Farbschema implementiert:

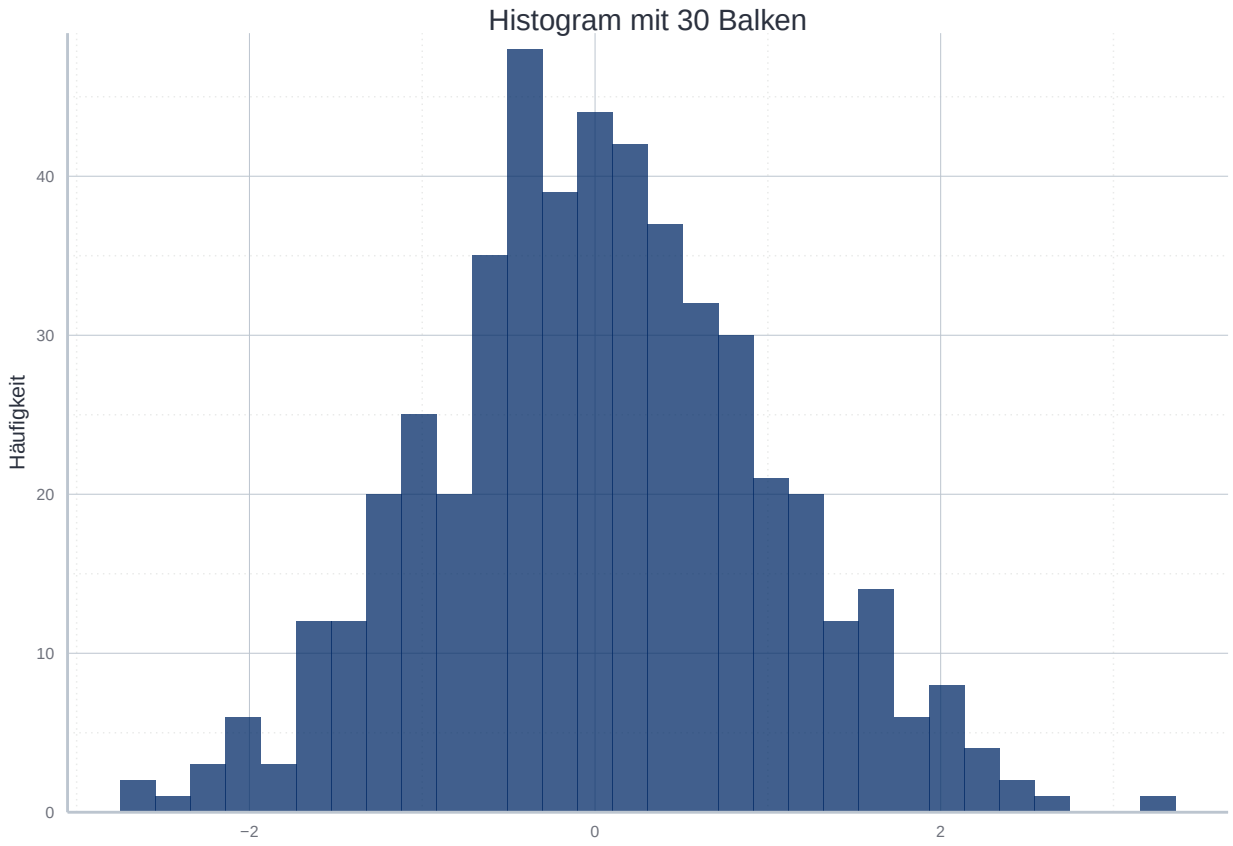
```
bubble_plot +
  scale_fill_viridis(
    discrete=TRUE,
    option="A"
    )
```



Hinweis: Größe der Blasen repräsentiert Bevölkerungsanzahl. Quelle: Gapminder.

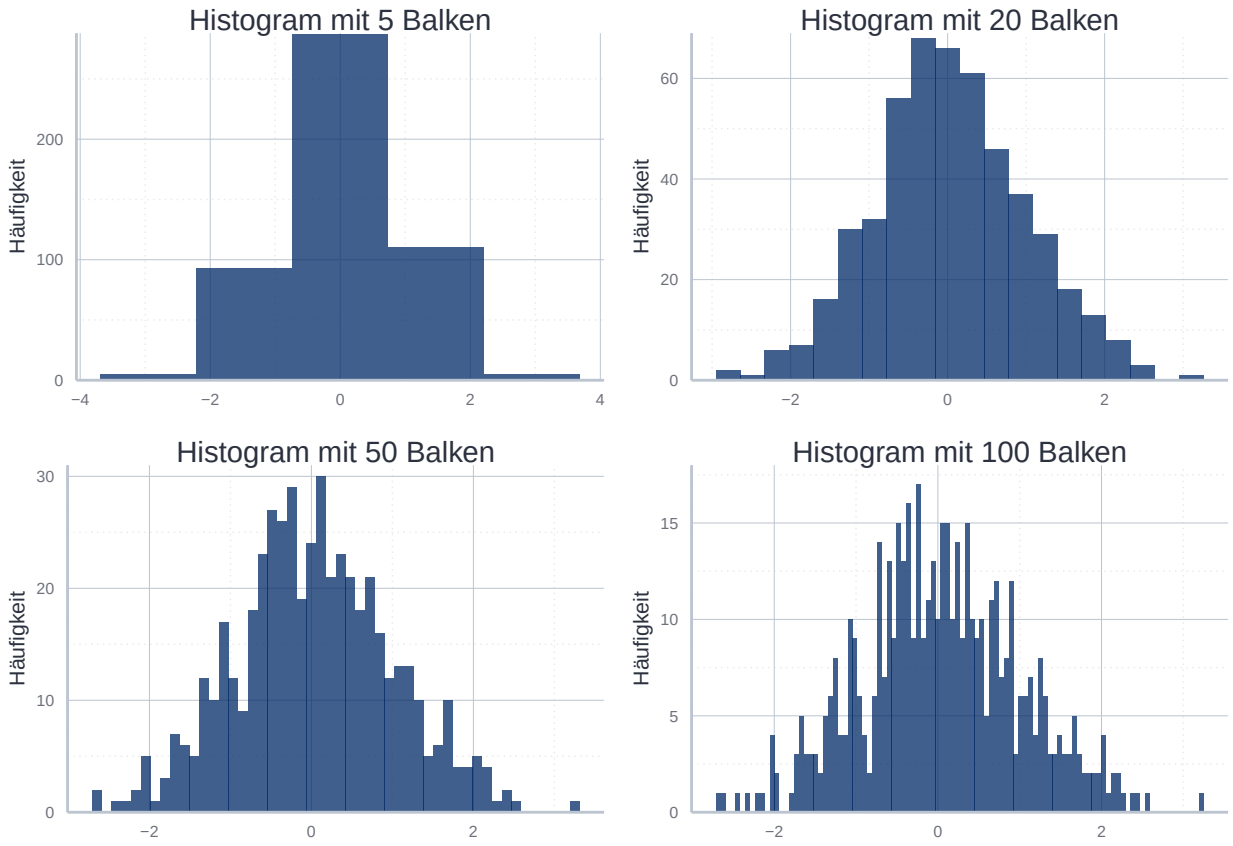
2.3.4 Histogramm

```
ggplot(data = histogram_daten,
       mapping = aes(x=x)) +
  geom_histogram(alpha=0.75, color=NA, fill="#002966") +
  scale_y_continuous(name = "Häufigkeit",
                     expand = expand_scale(c(0, 0), c(0, 1))) +
  ggtitle("Histogram mit 30 Balken") +
  theme_icae() +
  theme(axis.title.x = element_blank())
```



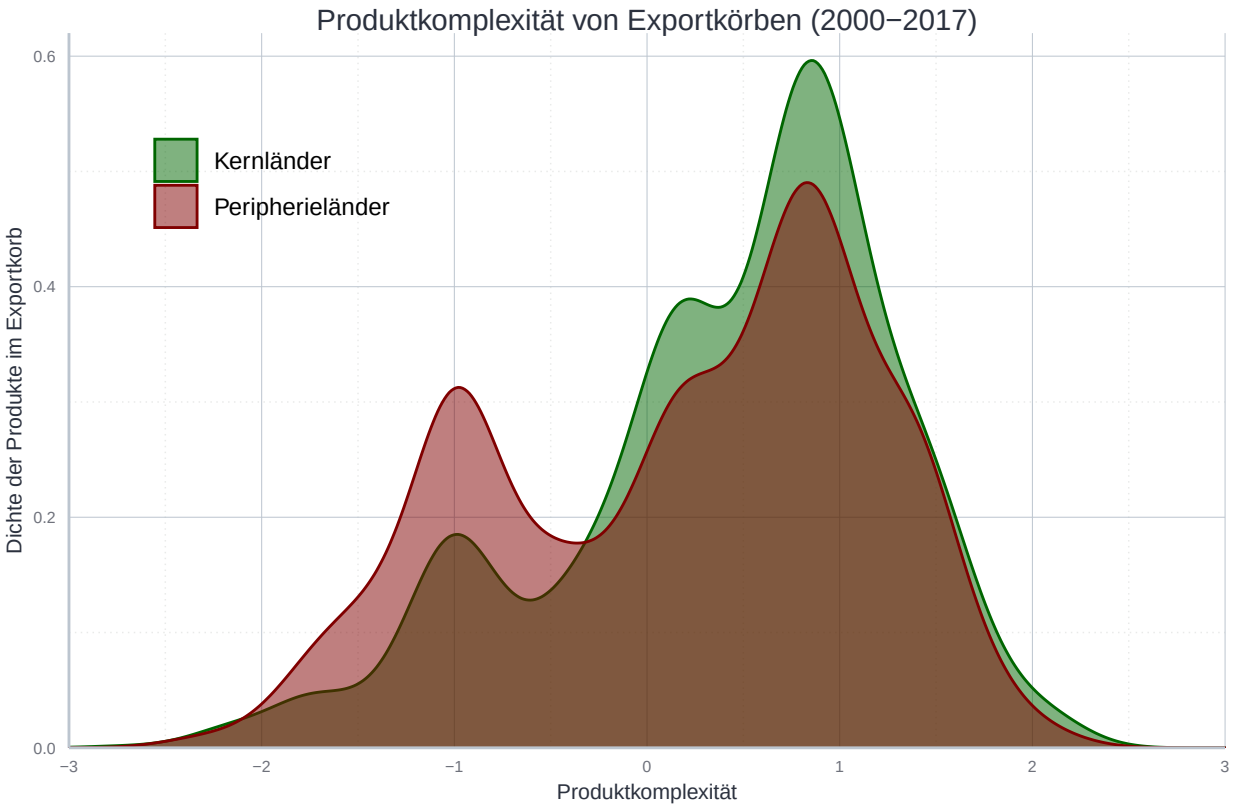
Im folgenden sehen Sie auch den großen Effekt unterschiedlicher Balkendicken:

```
bin_size <- c(5, 20, 50, 100)
hist_list <- list()
for (i in 1:length(bin_size)){
  hist_list[[i]] <- ggplot(data = histogram_daten,
       mapping = aes(x=x)) +
  geom_histogram(alpha=0.75, color=NA, fill="#002966", bins = bin_size[i]) +
  scale_y_continuous(name = "Häufigkeit",
                     expand = expansion(c(0, 0), c(0, 1))) +
  ggtitle(paste0("Histogram mit ", bin_size[i], " Balken")) +
  theme_icae() +
  theme(axis.title.x = element_blank())
}
ggarrange(plotlist = hist_list, ncol = 2, nrow = 2)
```



2.3.5 Dichtefunktion

```
ggplot(data = exportzusammensetzung,
       mapping = aes(
         x=pci,
         color=cgroup,
         fill=cgroup)
       ) +
  geom_density(
    mapping = aes(weight=exp_share),
    alpha=0.5
    ) +
  labs(
    title = "Produktkomplexität von Exportkörben (2000-2017)",
    caption = "Quelle: Gräbner et al. (2020, CJE)"
    ) +
  ylab("Dichte der Produkte im Exportkorb") +
  xlab("Produktkomplexität") +
  scale_y_continuous(limits = c(0, 0.62), expand = c(0, 0)) +
  scale_x_continuous(limits = c(-3, 3), expand = c(0, 0)) +
  scale_color_icae(palette = "mixed", aesthetics = c("color", "fill")) +
  theme_icae() +
  theme(legend.position = c(0.175, 0.8))
```



Quelle: Gräbner et al. (2020, CJE)