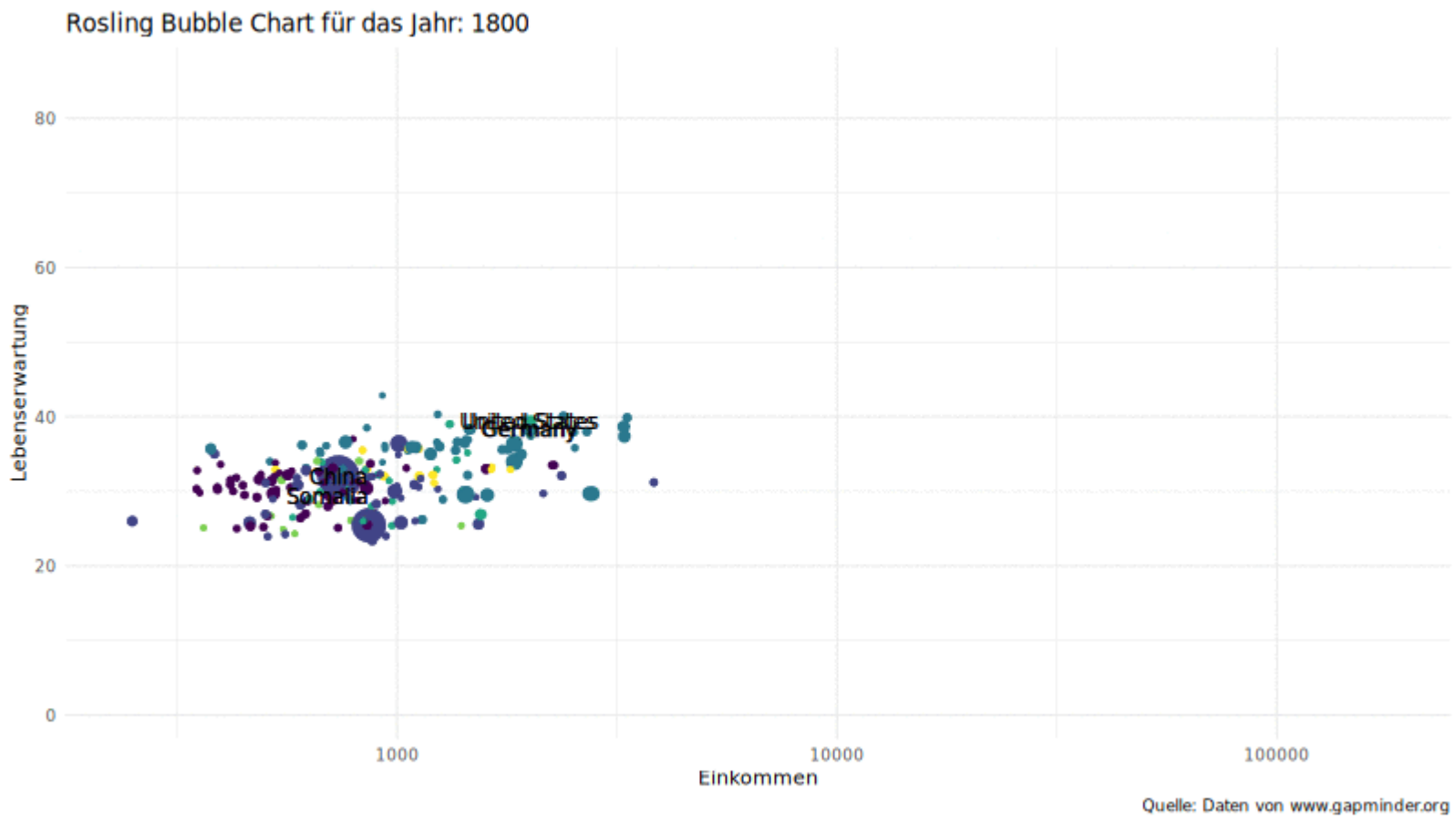

Datenvisualisierung

Roslings Bubble Chart

Quelle: Hans Rosling auf Youtube: https://youtu.be/Z8t4k0Q8e8Y

Roslings Bubble Chart



Einführung in ggplot2

- Grafiken nutzen zur Veranschaulichung von Sachverhalten
- Explorative Datenanalyse um die Daten kennen zu lernen
  - wird häufig vernachlässigt
  - Fehler oder unerwartete Probleme können häufig durch eingehende visuelle Datenanalyse vermieden werden
- Nutzen des Pakets `ggplot2` und `dplyr`
- Das Cheat Sheet zu ggplot2
  - Ein deutsches Cheat Sheet

Komponenten in ggplot2

- **Daten**: In unserem Beispiel bisher immer der `Datensatz` Datensatz
- **Geometry**: Verschiedene Arten von Grafiken, z.B. Streudiagramm, Boxplot, Histogramm, Kerndichteschätzung
- **Aesthetic**: Definieren was die x-Achse und was die y-Achse zeigen soll. Weiterhin können wir Farben und Formen unserer Grafik bestimmen, alles abhängig von der **Geometry**, welche wir wählen
- **scale**: Definieren wie die Skalierung unserer x-Achse und y-Achse sein soll (eventuell logarithmisch?)

Mittels der nächsten Folien wollen wir eine Grafik Schritt für Schritt erzeugen

Generierung eines leeren `ggplot` Objekts

Zuerst generieren wir uns ein leeres `ggplot` Objekt, in welchem wir definieren, welche Daten wir zeigen wollen.

- ✚ Dadurch erhalten wir eine graue Box, welche von ggplot gerendet wurde

```
Datensatz <- readRDS("data/Gapminder_1800-2020.rds")
ggplot(data = Datensatz)
```

Layer

In diese graue Box können wir nun eine Grafik einbetten

Es gibt verschiedene Layer (Grafikarten) die wir verwenden können (hier einige Beispiele):

- geom_bar, geom_point, geom_line, geom_smooth, geom_histogram, geom_boxpolot, geom_density

Um einen Layer hinzuzufügen nutzen wir das + Symbol. Code könnte entsprechend so aussehen:

> DATENSATZ |> `ggplot()` + LAYER 1 + LAYER 2 + ...

Layer

Angenommen Sie wollen ein Streudiagramm erzeugen, dann können Sie schauen, welcher Layer dafür geeignet ist. Das Cheat Sheet sagt uns, dass `geom_point` dafür geeignet ist.

Anschließend sollten Sie wissen, wie viele Argumente ihre Grafik benötigt um dargestellt werden zu können:

- Gehen Sie zu der Hilfeseite von `?geom_point` und scrollen Sie zu der Überschrift `Aesthetics` um mehr darüber zu erfahren
- Ein Streudiagramm benötigt mindestens zwei Argumente `x` und `y`

Die Aesthetics Funktion `aes`

Mittels der Funktion `aes` kann anschließend die Grafik auf Grundlage der Daten erstellt werden

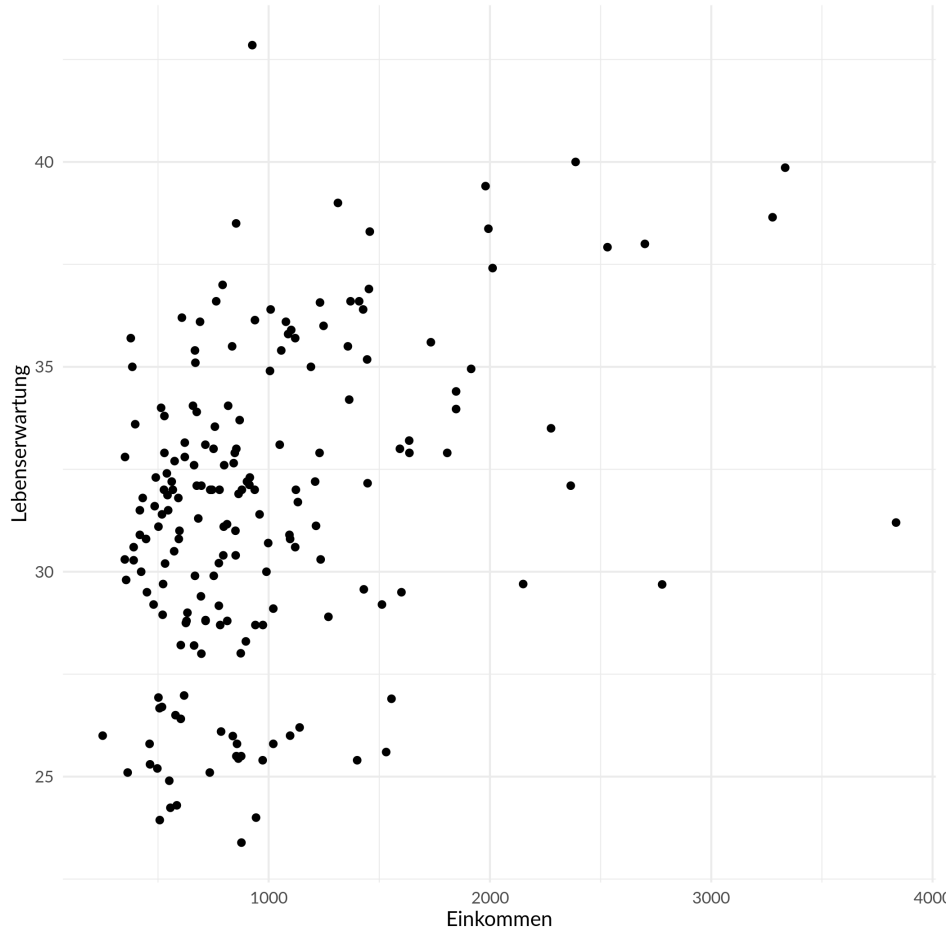
- `aes` wird oft als Argument einer Geometry Funktion verwendet

Beispiel für ein Streudiagramm mit Einkommen und Lebenserwartung für das Jahr 1800:

```
Datensatz |>
  filter(Jahr == 1800) |>
  ggplot() +
  geom_point(aes(x = Einkommen,
                 y = Lebenserwartung))
```

- `x =` und `y =` können wir hier auch weg lassen
  - Die ersten zwei Argumente werden von ggplot automatisch als `x` und `y` aufgefasst
- Skalierung und Benennung erfolgen automatisch, wenn nicht anders spezifiziert

Die Aesthetics Funktion `aes`



Zusätzliche Layer

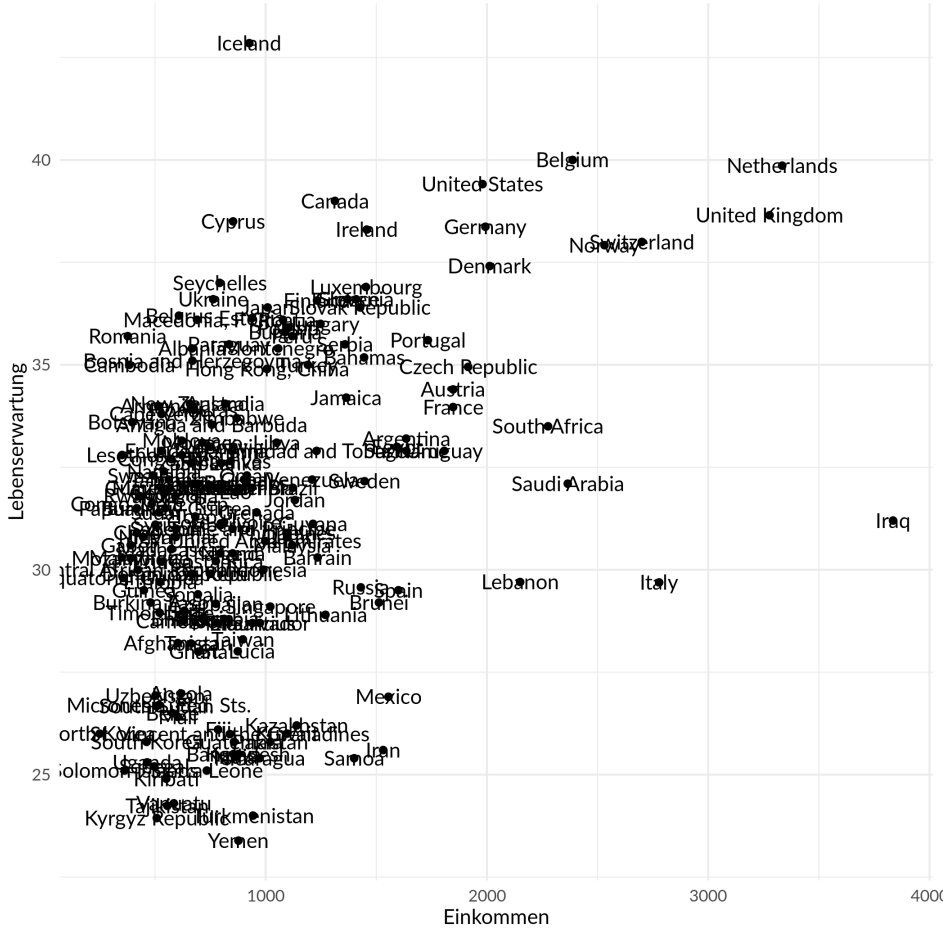
In manchen Fällen möchten wir gerne einen zusätzlichen Layer in unsere Grafik einfügen. Beispielsweise wollen wir jedem Datenpunkt ein Label geben, welches genau beziffert um welchen Wert es sich handelt.

- Möglich mit `geom_label` and `geom_text`
- Durch den `label` Befehl in `aes` können wir dies umsetzen
- **Beachten:** Das `label` Argument muss innerhalb von `aes` aufgerufen werden, ansonsten bekommen Sie einen Fehler

Ein Beispiel auf der nächsten Folie

```
library(ggthemes)
library(ggrepel)
Datensatz |>
  filter(Jahr == 1800) |>
  ggplot() +
  geom_point(aes(Einkommen, Lebenserwartung)) +
  geom_text(aes(Einkommen, Lebenserwartung, label=country))
```

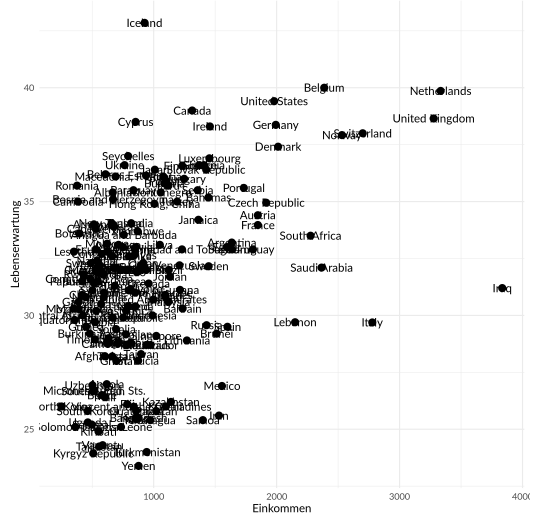
Zusätzliche Layer



Weitere Argumente

- Jede Geometry Funktion hat ihre individuellen Argumente neben `aes` und `data`
- Beispielsweise kann über `size` die Größe der Punkte im Streudiagramm verändert werden

```
Datensatz |>
  filter(Jahr == 1800) |>
  ggplot() +
  geom_point(aes(Einkommen, Lebenserwartung), size = 3) +
  geom_text(aes(Einkommen, Lebenserwartung, label=country))
```

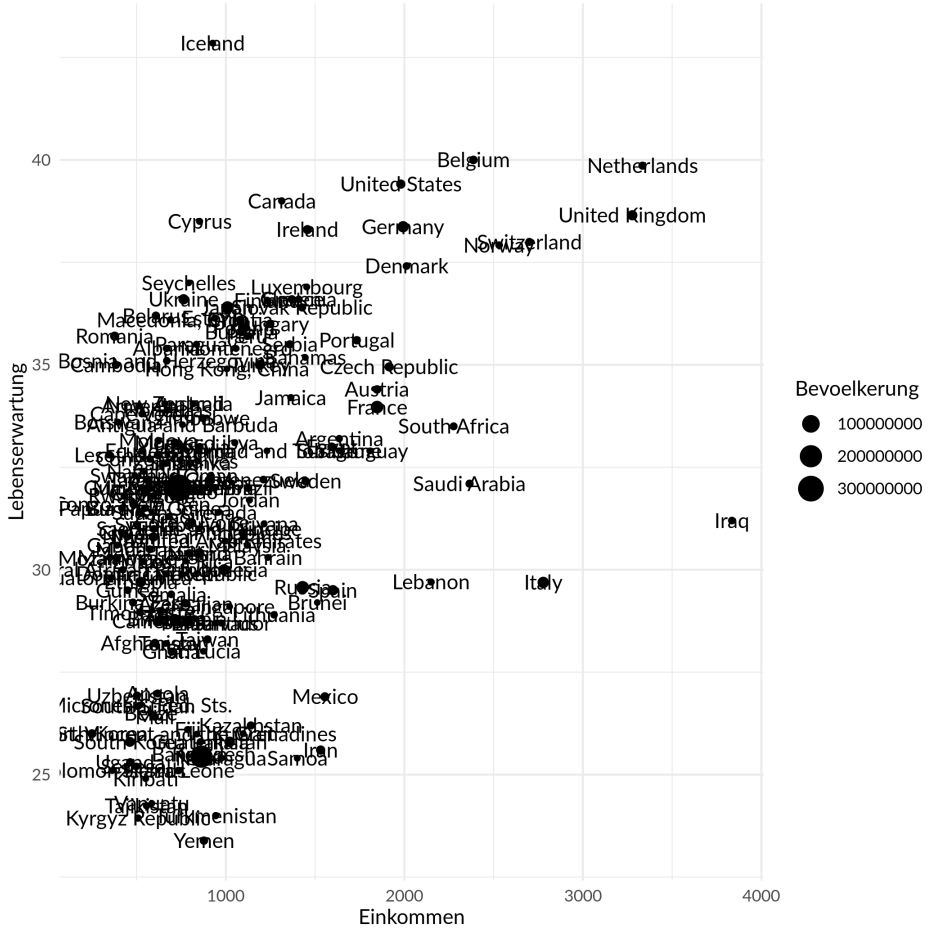


Weitere Argumente

- `size` betrifft hier alle Punkte im Streudiagramm und ist nicht in der `aes` enthalten
- Sie können `size` jedoch auch auf der Basis einer anderen Variable definieren, z.B. der Bevölkerungszahl
- Nun muss die Definition von `size` jedoch innerhalb der `aes` erfolgen, da auf den Datensatz zurückgegriffen werden soll

```
Datensatz |>
  filter(Jahr == 1800) |>
  ggplot() +
  geom_point(aes(Einkommen, Lebenserwartung, size = Bevoelkerung)) +
  geom_text(aes(Einkommen, Lebenserwartung, label=country))
```

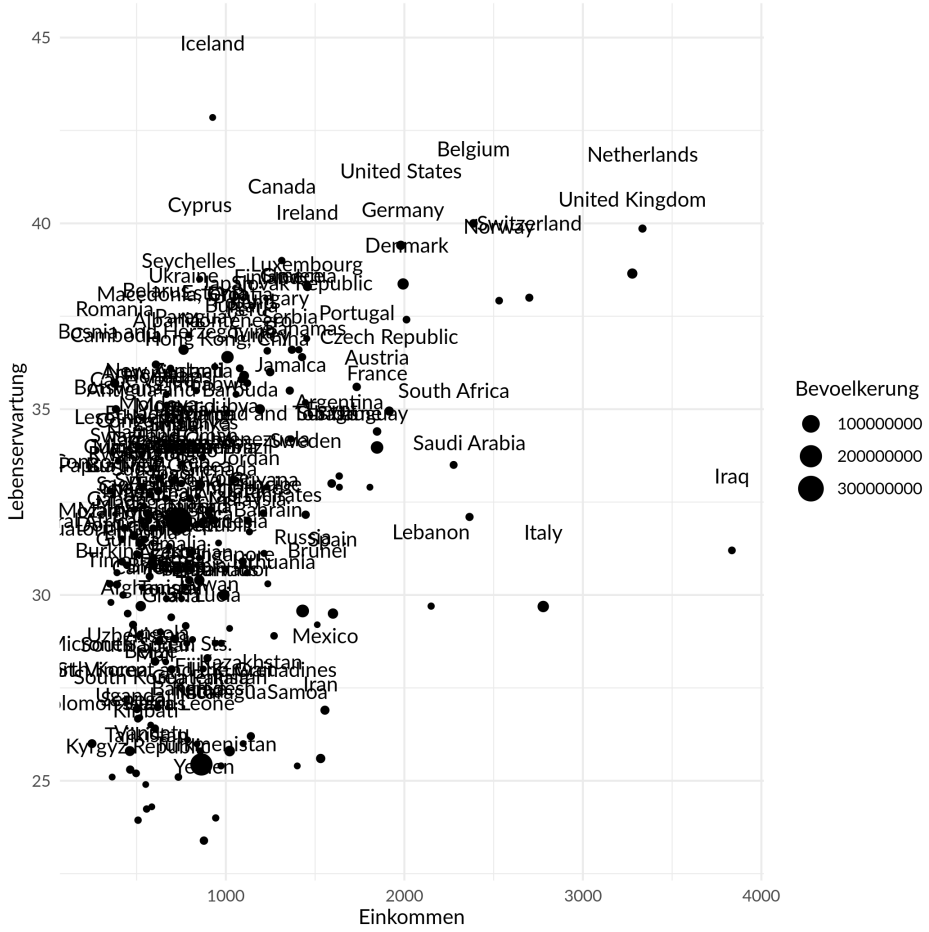
Weitere Argumente



Weitere Argumente

- Leider können wir durch größere Punkte die Labels nicht mehr lesen
- Aus der Hilfeseite von `?geom_text` erfahren wir, wie die Labels verändert werden können:
  - Durch `nudge_x` können die Labels etwas nach rechts abgesetzt werden
  - Durch `nudge_y` können die Labels etwas nach oben abgesetzt werden

Weitere Argumente



Globale `aes` Abbildungen

- Im vorherigen Beispiel hatten wir `aes(Einkommen, Lebenserwartung)` doppelt verwendet
- Wir können statt dessen auch eine *globale* aesthetic Abbildung erstellen
- Dies wird erreicht, indem wir als erstes eine *Grafikvorlage* erstellen und uns eine *globale* `aes` definieren

Wir speichern unsere *Grafikvorlage* in `p`:

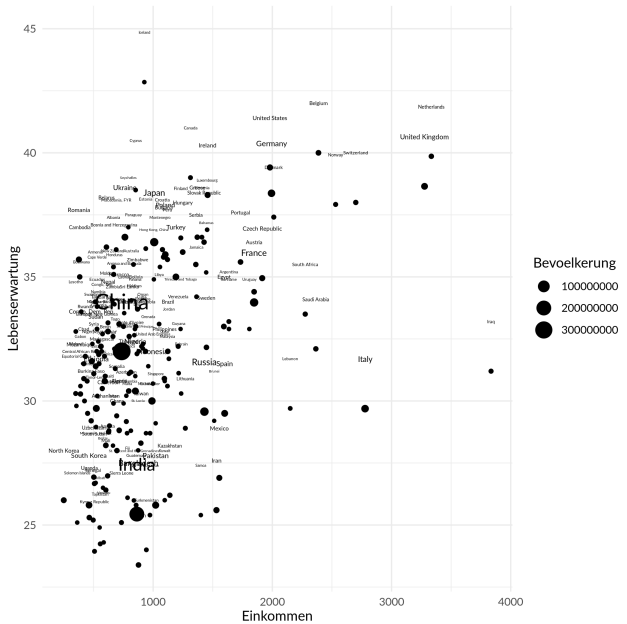
```
p <- Datensatz |>
  filter(Jahr == 1800) |>
  ggplot(aes(x = Einkommen, y = Lebenserwartung, label = country, size = Bevoelkerung))
```

Globale `aes` Abbildungen

Anschließend verändern wir unser Vorlage `p` wie wir es wünschen

- Hier verändern wir nur die Größe der Punkte und Position der Labels

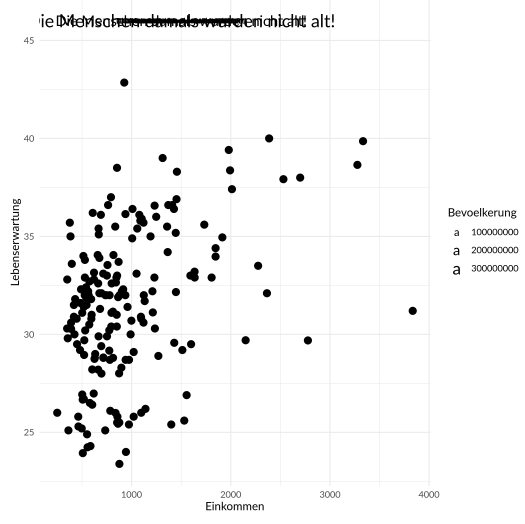
```
p + geom_point() +
  geom_text(nudge_y = 2)
```



Lokale `aes` überschreiben globale `aes`

- Wir können diese *globalen* Abbildungen durch *lokale* überschreiben
- Beispielsweise können wir den Text *Die Menschen damals wurden nicht alt!* auf Höhe der x-Achse bei 1500 und der y-Achse bei 50 platzieren:

```
p + geom_point(size = 3) +
  geom_text(aes(x = 1500, y = 46,
  label = "Die Menschen damals wurden nicht alt!"))
```

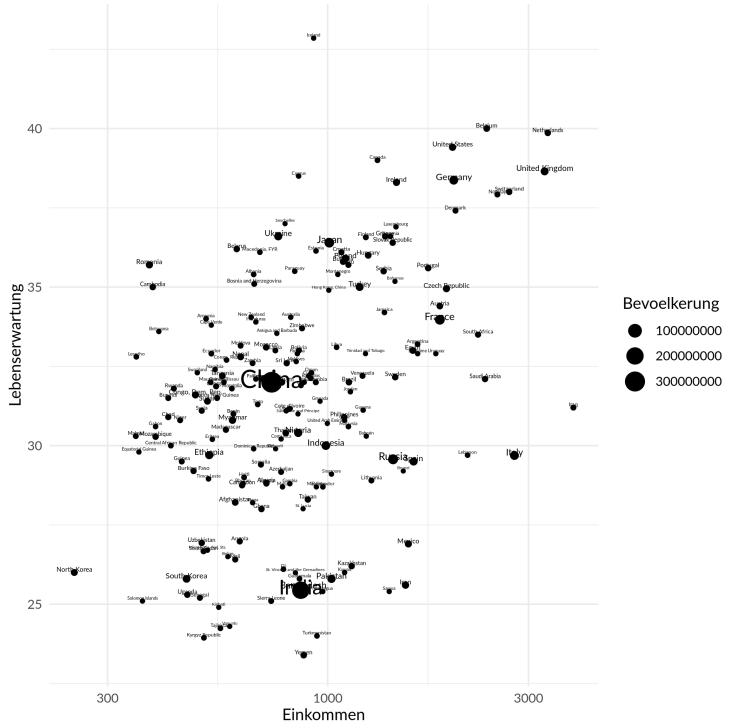


Skalierung

- Falls eine andere Skalierung gewünscht wird
  - durch `scale_x_continous` verändern wir die x-Achse
  - durch `scale_y_continous` verändern wir die y-Achse
  - durch `scale_x_log10()` verändern wir die x-Achse zu logarithmierter Skala
  - durch `scale_y_log10()` verändern wir die y-Achse zu logarithmierter Skala

Skalierung

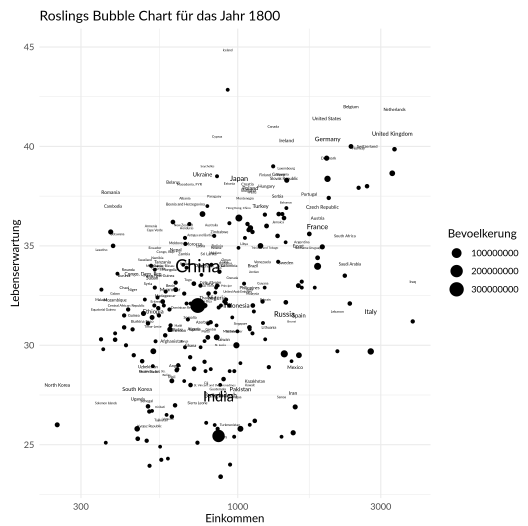
```
p + geom_point() +
  geom_text(nudge_y = 0.1) +
  scale_x_log10()
```



Titel und Label

- Titel oder Label können wir durch `ggtitle` bzw. `xlab`, `ylab` verändern:

```
p + geom_point() +
  geom_text(nudge_y = 2) +
  scale_x_log10() +
  xlab("Einkommen") +
  ylab("Lebenserwartung") +
  ggtitle("Roslings Bubble Chart für das Jahr 1800")
```



Farben

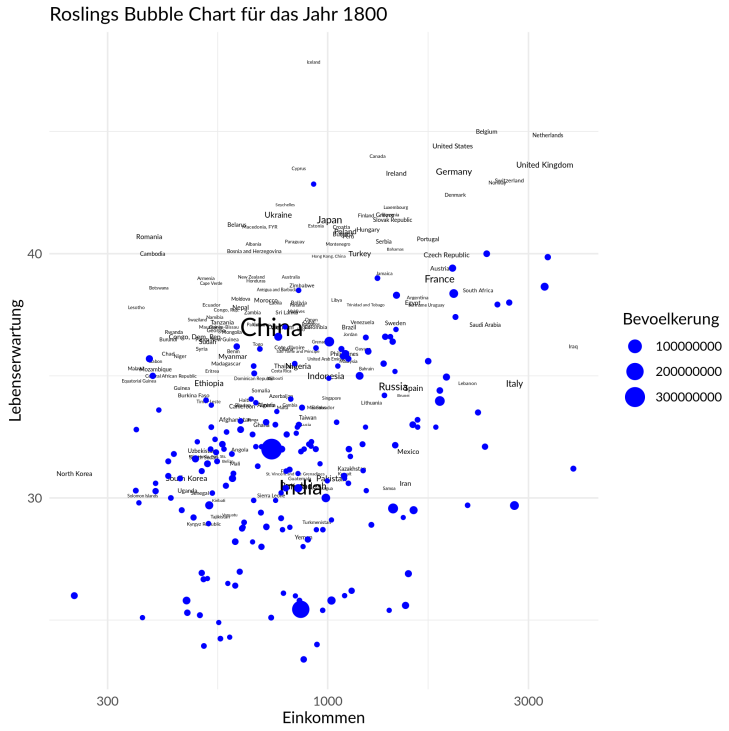
- Durch das Argument `col` innerhalb von `geom_point` kann die Farbe der Punkte verändert werden
- Erstellen wir uns der einfachheit halber eine neue Grafikvorlage:

```
p <- Datensatz |>
  filter(Jahr == 1800) |>
  ggplot(aes(x = Einkommen, y = Lebenserwartung,
             label = country, size = Bevoelkerung)) +
  geom_text(nudge_y = 5) +
  scale_x_log10() +
  xlab("Einkommen") +
  ylab("Lebenserwartung") +
  ggtitle("Roslings Bubble Chart für das Jahr 1800")
```

Farben

- Nun sollen alle Punkte blau sein

```
p + geom_point(color = "blue")
```

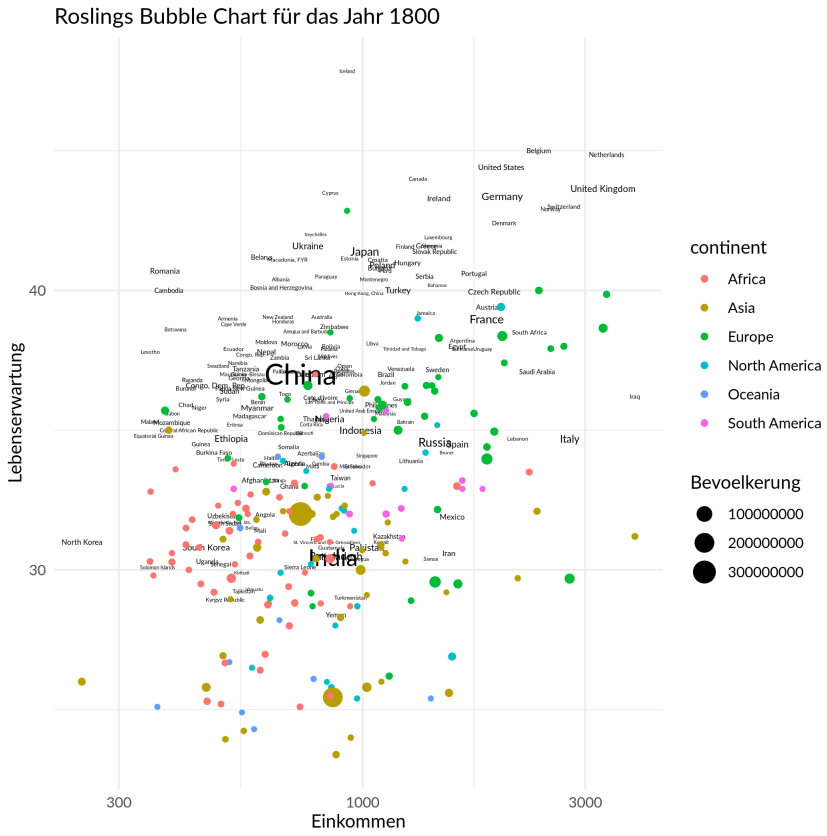


Farben

- Nicht unbedingt unser Ziel
- *Besser:* Bestimmte Farben für bestimmte Gruppen
- **Schön bei `ggplot2`:** Wir können `ggplot2` eine katogoriale Variable angeben:
  - nun werden automatisch unterschiedliche Farben für alle Gruppen dieser Variablen zugeteilt
  - es wird automatisch eine Legende erzeugt

Farben

```
p + geom_point(aes(color=continent))
```

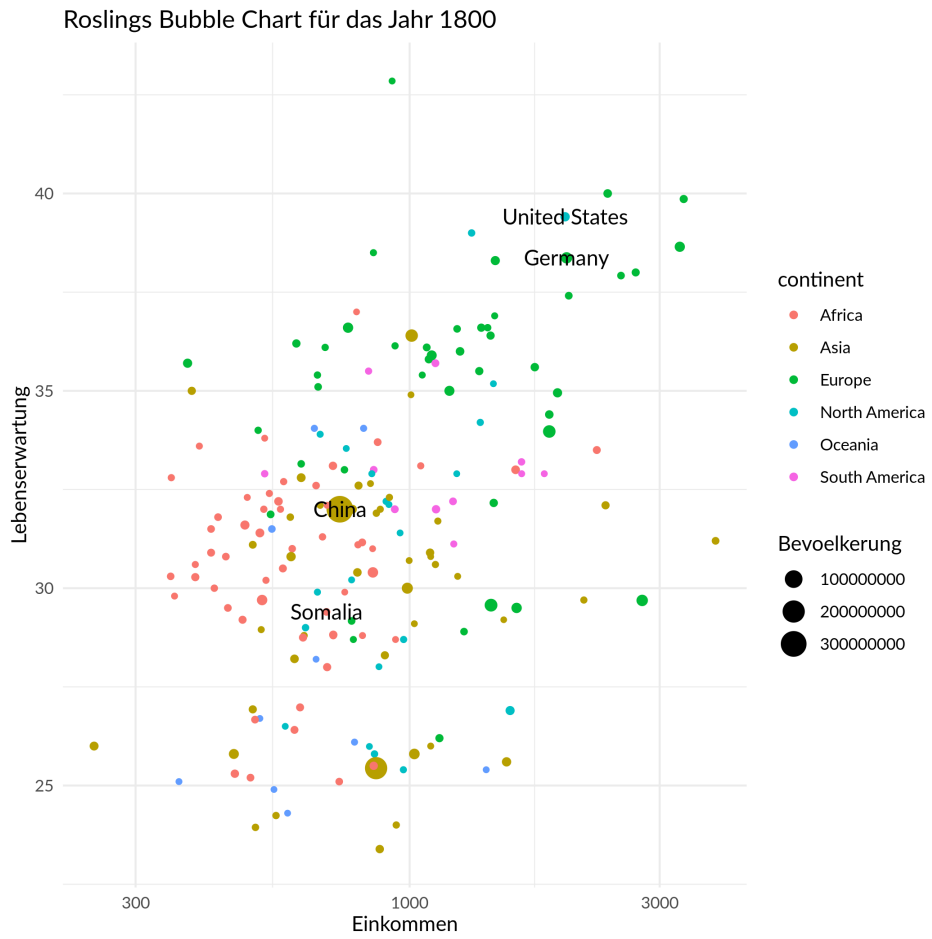


Nur bestimmte Länder beschriften

- Es sollen nun nicht alle Ländern, sondern nur China, Somalia, Deutschland und die USA beschriftet werden.

```
p_pre <- Datensatz |>
  filter(Jahr == 1800) |>
  mutate(annotation = case_when(
           country %in% c("China", "Somalia", "Germany", "United States") ~ "yes"))

p <- p_pre |>
  ggplot(aes(x = Einkommen, y = Lebenserwartung,
             label = country, size = Bevoelkerung)) +
  geom_point(aes(color=continent)) +
  geom_text(data = p_pre |> filter(annotation=="yes"), aes(label=country), size=4) +
  scale_x_log10() +
  xlab("Einkommen") +
  ylab("Lebenserwartung") +
  ggtitle("Roslings Bubble Chart für das Jahr 1800")

p
```

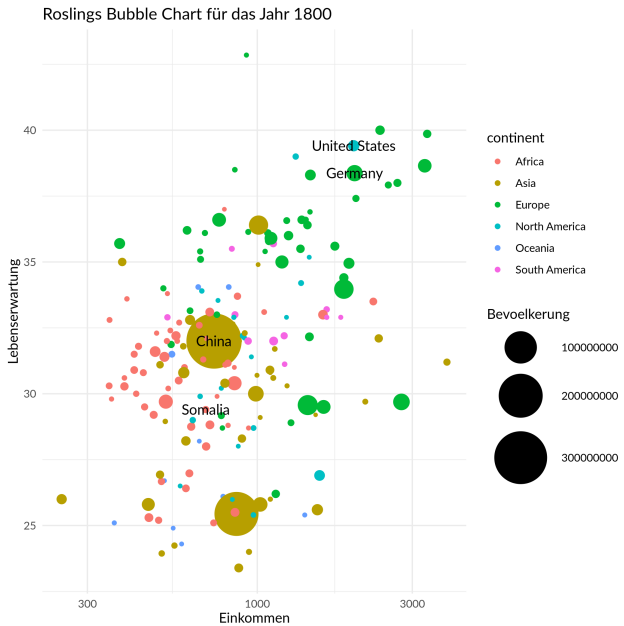
Nur bestimmte Länder beschriften



Die Größe der Länderpunkte

- Nun können Sie noch die Größe der Kreise etwas besser skalieren.
- Hier soll die Bevölkerung des größten Landes 20 mal so groß dargestellt werden wie das kleineste Land

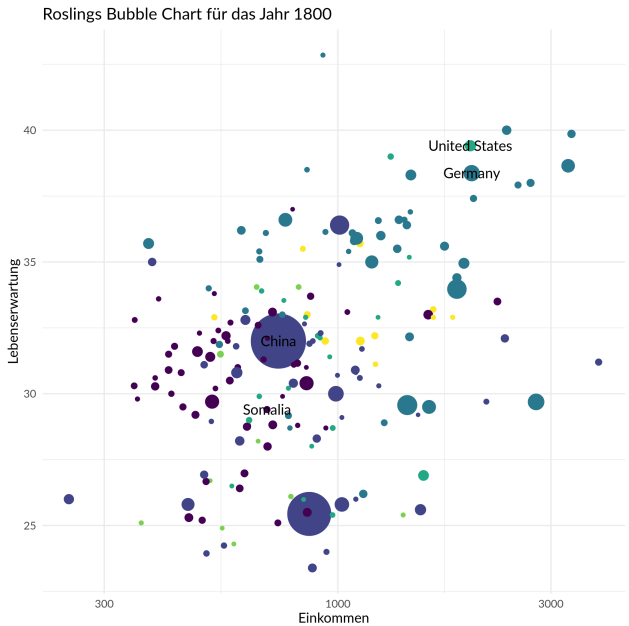
```
p <- p + scale_size(range = c(1, 20))

p
```



Farbpalette und Legende

- Es gibt in ggplot verschiedenste Farbpaletten, hier soll die Farbpalette von viridis verwendet werden

```
library(viridis)
p + scale_color_viridis(discrete=TRUE) + theme(legend.position="none")
```

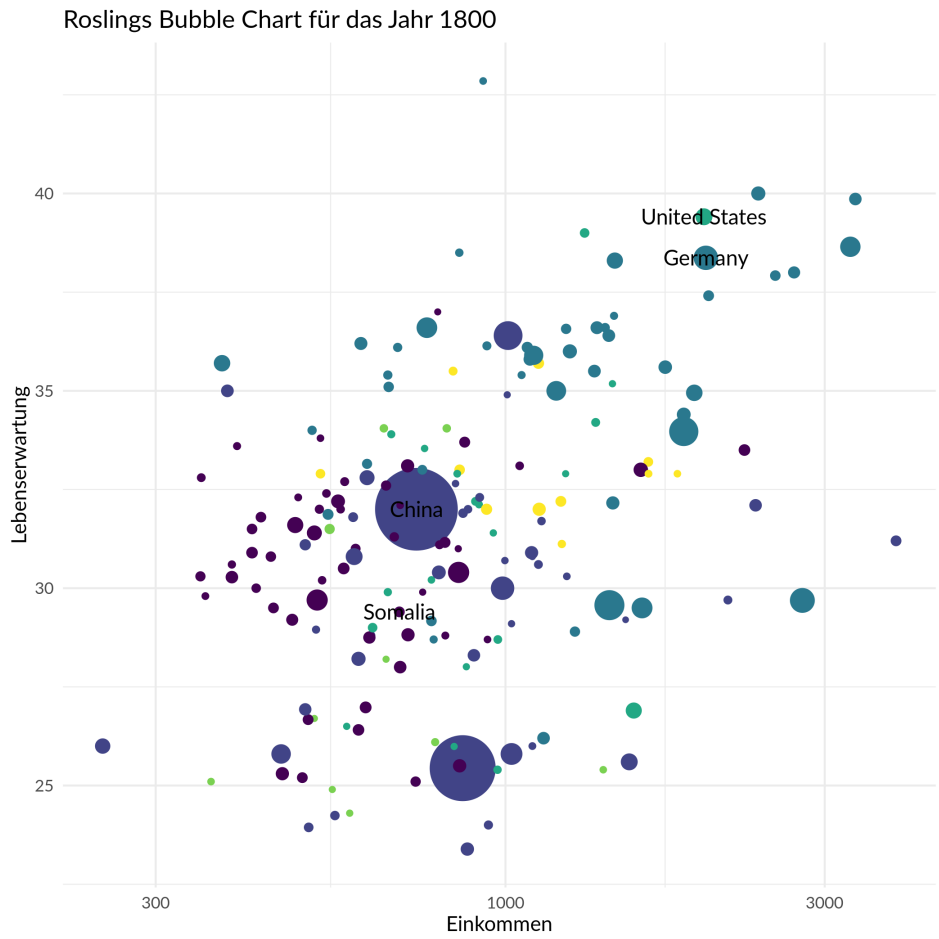


Code für die Grafik aus dem Jahr 1800

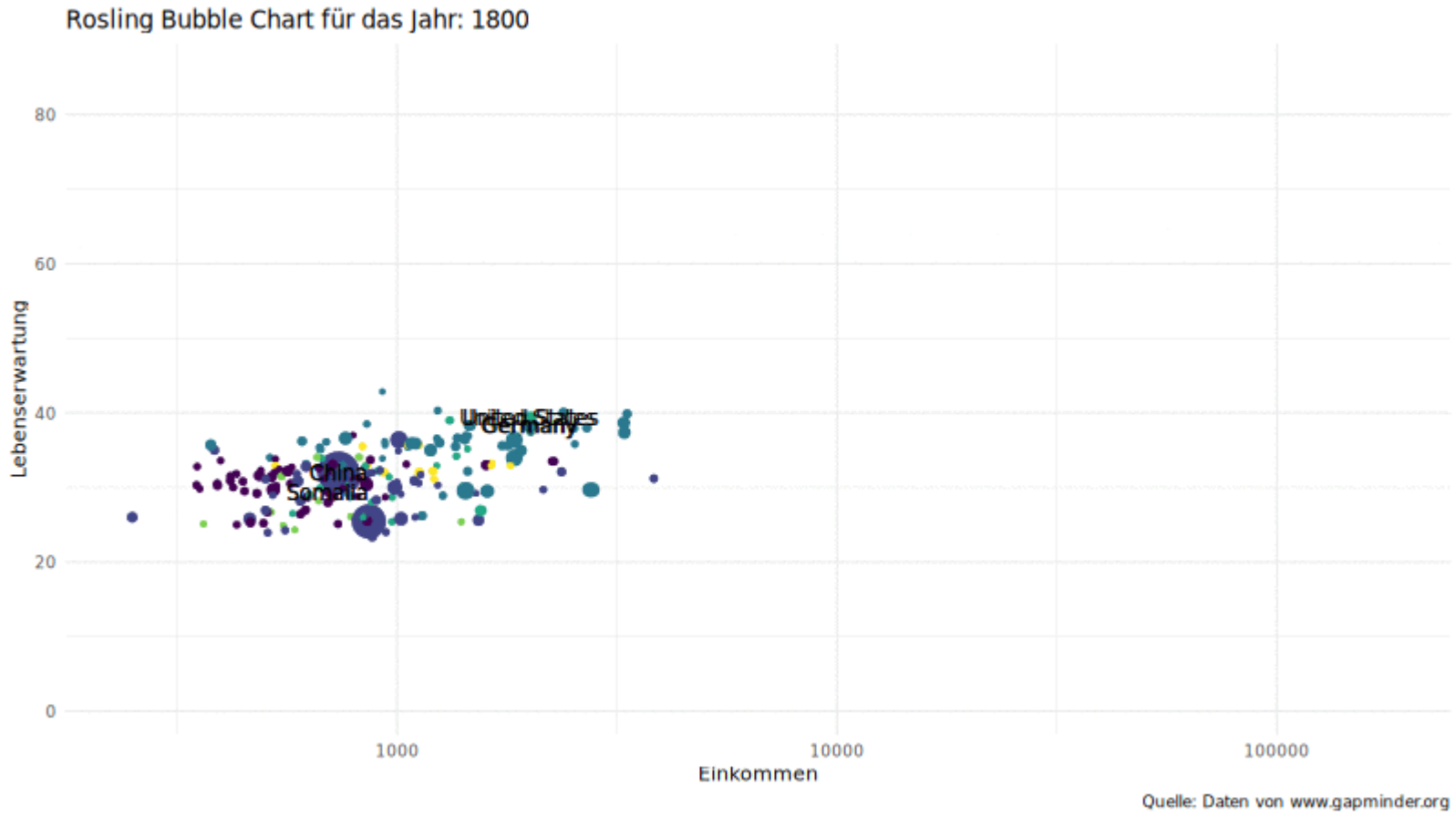
- Gesamter Code für die Grafik

```
process <- Datensatz |>
  filter(Jahr == 1800) |>
  mutate(annotation = case_when(
    country %in% c("China", "Somalia", "Germany", "United States") ~ "yes")
  )

process |>
  ggplot(aes(x = Einkommen, y = Lebenserwartung,
             label = country, size = Bevoelkerung)) +
  geom_point(aes(color=continent)) +
  geom_text(data = process |> filter(annotation=="yes"), aes(label=country), size=4) +
  scale_x_log10() +
  xlab("Einkommen") +
  ylab("Lebenserwartung") +
  ggtitle("Roslings Bubble Chart für das Jahr 1800") +
  scale_size(range = c(1, 20)) +
  scale_color_viridis(discrete=TRUE) +
  theme(legend.position="none")
```

Code für die Grafik aus dem Jahr 1800



Animation für mehrere Jahre



Zusätzliche Pakete

- Weiterhin gibt es bei `ggplot2` Zusatzpakete, durch welche wir noch einige Dinge verändern können
- Durch die zusätzlichen Pakete `ggthemes` und `ggrepel` können wir unserer Grafik noch final bearbeiten
- Beispielsweise können wir unsere Grafik mit einem anderen Grundlayout ausstatten
  - Hier dem Thema `theme_fivethirtyeight()`
  - Hier finden Sie weitere Themen für ihre Grafik

Zusätzliche Pakete

- Das Paket `ggrepel` stellt sicher, dass die Label eines Punktes nicht übereinander gezeigt werden
- In unserem Beispiel schwer umzusetzen, da hier sehr viele Labels vorhanden sind
- Grundsätzlich kann dies jedoch durch das laden des Pakets `ggrepel` erreicht werden
  - In der Grafik muss dann nur noch `geom_text` durch `geom_text_repel` ersetzt werden

Mehrere Grafiken nebeneinander

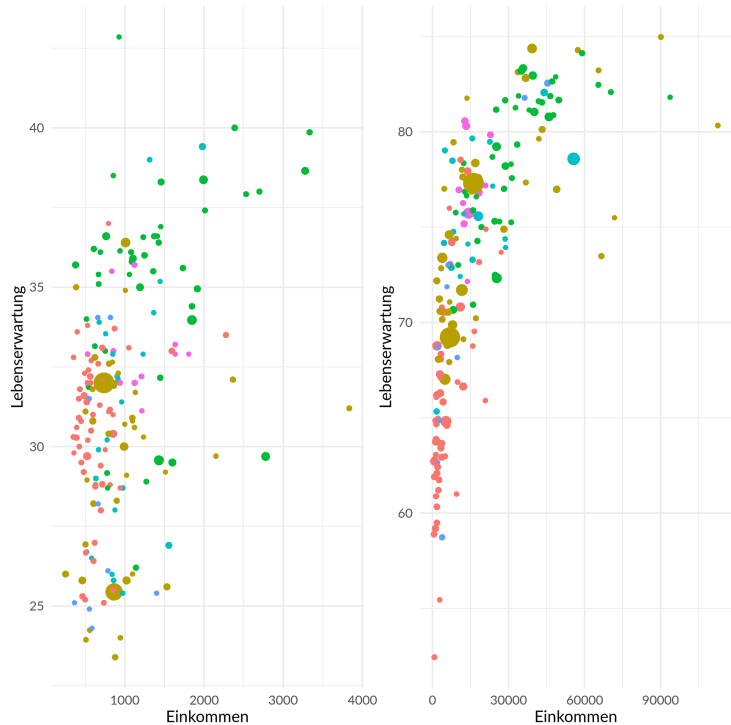
Sie wollen mehrere Grafiken nebeneinander platzieren

- Hier hilft ihnen das Paket `gridExtra`
- Mit der Funktion `grid.arrange` aus `gridExtra` werden die Schaubilder nebeneinander gezeigt:

```
p1 <- Datensatz |>
  filter(Jahr == 1800) |>
  ggplot(aes(x = Einkommen, y = Lebenserwartung, size=Bevoelkerung, color=continent)) +
  geom_point() +
  theme(legend.position="none")

p2 <- Datensatz |>
  filter(Jahr == 2018) |>
  ggplot(aes(x = Einkommen, y = Lebenserwartung, size=Bevoelkerung, color=continent)) +
  geom_point() +
  theme(legend.position="none")
```

Mehrere Grafiken nebeneinander

```
library(gridExtra)
grid.arrange(p1,p2, ncol = 2)
```



Achtung: Hier sind die Achsen für beide Grafiken nicht gleich! Diese müssen bei vergleichenden Schaubildern immer einheitlich formatiert sein!