6. Facets

Plots nach Gruppen aufteilen mit facet_wrap und individueller Achsenformatierung

Dr. Paul Schmidt

```
for (pkg in c("tidyverse", "gapminder", "showtext", "ggtext", "ggh4x", "scales")) {
  if (!require(pkg, character.only = TRUE)) install.packages(pkg)
}

showtext::showtext_opts(dpi = 300)
```

Wenn man mit Daten arbeitet, die mehrere numerische Variablen oder natuerliche Untergruppen enthalten, moechte man oft denselben Plottyp fuer jede Variable oder Gruppe nebeneinander zeigen. Diese Plots manuell zu erstellen - Code kopieren und jedes Mal den Variablennamen anpassen - ist muehsam und fehleranfaellig. Facetting loest dieses Problem elegant: `facet_wrap()` teilt die Daten automatisch nach einer oder mehreren Variablen auf und erstellt fuer jede Gruppe ein Panel, alle mit denselben aesthetischen Mappings, Skalen und dem gleichen Theme.

In diesem Kapitel wenden wir Facetting auf den gapminder-Datensatz aus dem vorherigen Kapitel an und erstellen einen facettierten Dumbbell Plot, der Lebenserwartung, Bevoelkerung und BIP pro Kopf in einer einzigen Grafik vergleicht. Dabei begegnen wir einer gaengigen Herausforderung: Wenn Facetten Variablen auf sehr unterschiedlichen Skalen darstellen, wird die Standard-Achsenformatierung unlesbar. Wir loesen dies mit benutzerdefinierter Label-Formatierung und dem {ggh4x}-Paket fuer individuelle Achsensteuerung pro Facette.

Setup

Wir verwenden die Datenvorbereitung und das Theme aus Kapitel 5:

```
dat <- gapminder::gapminder %>%
  filter(year == 1952 | year == 2007) %>%
  filter(country %in% c(
    "Canada", "Germany", "Japan",
    "Netherlands", "Nigeria", "Vietnam", "Zimbabwe"
  )) %>%
  mutate(year = as.factor(year)) %>%
  droplevels()

sorted_countries <- dat %>%
  filter(year == "2007") %>%
  arrange(lifeExp) %>%
  pull(country) %>%
  as.character()

dat <- dat %>%
  mutate(country = fct_relevel(country, sorted_countries))

sysfonts::font_add_google("Kanit", "kanit")
showtext::showtext_auto()

year_colors <- c("1952" = "#F7AA59", "2007" = "#37A9E1")
```

```
theme_nature <- function(base_size = 12) {
  theme_minimal(base_size = base_size) +
    theme(
      text = element_text(family = "kanit"),
      plot.title.position = "plot",
      plot.title = element_text(size = 15, face = "bold"),
      plot.subtitle = ggtext::element_textbox_simple(
        size = 10, margin = margin(0, 0, 10, 0)
      ),
      axis.line.y = element_blank(),
      axis.text.x = element_text(color = "#AAAAAA"),
      axis.ticks.x = element_line(color = "#AAAAAA", linewidth = 0.4),
      axis.ticks.length.x = unit(4, "pt"),
      axis.line.x = element_line(color = "black", linewidth = 0.6),
      panel.grid.minor = element_blank(),
      panel.grid.major.y = element_blank(),
      panel.grid.major.x = element_line(
        linetype = "dotted", color = "#AAAAAA", linewidth = 0.3
      )
    )
}
```

Datenstruktur fuer Facets

Um pro Variable eine Facette zu erstellen, muessen alle numerischen Werte in einer einzigen Spalte stehen - das ist das “Long-Format”, mit dem ggplot2 am besten arbeitet. Die gapminder-Daten haben aktuell drei separate Spalten (`lifeExp` , `pop` , `gdpPercap`), also muessen wir sie in zwei Spalten pivotieren: eine fuer den Variablennamen (`statistic`) und eine fuer den Wert (`value`).

```
dat_long <- dat %>%
  pivot_longer(
    cols = c(lifeExp, pop, gdpPercap),
    names_to = "statistic",
    values_to = "value"
  )

dat_long
```

```
# A tibble: 42 × 5
   country continent year  statistic      value
   <fct>   <fct>     <fct> <chr>          <dbl>
 1 Canada  Americas  1952  lifeExp         68.8
 2 Canada  Americas  1952  pop       14785584
 3 Canada  Americas  1952  gdpPercap    11367.
 4 Canada  Americas  2007  lifeExp         80.7
 5 Canada  Americas  2007  pop       33390141
 6 Canada  Americas  2007  gdpPercap    36319.
 7 Germany Europe    1952  lifeExp         67.5
 8 Germany Europe    1952  pop       69145952
 9 Germany Europe    1952  gdpPercap     7144.
10 Germany Europe    2007  lifeExp         79.4
# i 32 more rows
```

Ausserdem brauchen wir eine breite Version fuer die Dumbbell-Segmente und Differenzlabels:

```
dat_wide <- dat_long %>%
  pivot_wider(
    names_from = year,
    values_from = value,
```

```
    names_prefix = "year_"
  ) %>%
  mutate(
    max_x = pmax(year_2007, year_1952),
    diff = year_2007 - year_1952
  )
```

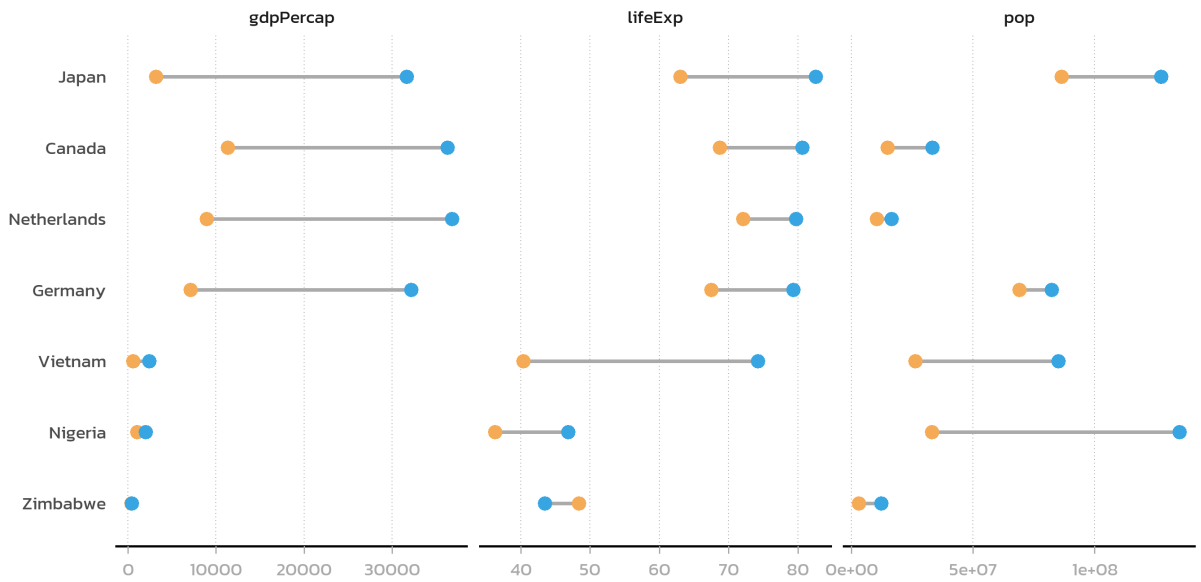
Einfaches Facetting

Mit `facet_wrap(~ statistic)` wird der Plot in drei Panels aufgeteilt, eines pro Variable. Die entscheidende Ergaenzung hier ist `scales = "free_x"`, das jedem Panel einen eigenen x-Achsenbereich erlaubt. Ohne diese Einstellung wuerden alle drei Panels eine gemeinsame Achse teilen, und Lebenserwartungswerte (40-80) waeren neben Bevoelkerungswerten (in Millionen) unsichtbar.

Das `scales`-Argument akzeptiert vier Optionen: `"fixed"` (der Standard - alle Panels teilen dieselben Achsen), `"free_x"` (jedes Panel bekommt seinen eigenen x-Achsenbereich), `"free_y"` (jedes Panel bekommt seinen eigenen y-Achsenbereich) und `"free"` (beide Achsen sind unabhaengig). Die richtige Wahl haengt davon ab, was man betonen moechte: Gemeinsame Skalen erleichtern Vergleiche zwischen Panels, waehrend freie Skalen verhindern, dass Variablen auf unterschiedlichen Groessenordnungen sich gegenseitig zusammenstauchen.

```
subtitle_text <- glue::glue(
  "Im Jahr <b style='color:{year_colors[['2007']]};'>2007</b>, ",
  "verglichen mit <b style='color:{year_colors[['1952']]};'>1952</b>, ",
  "spiegeln Lebenserwartung, BIP pro Kopf und Bevoelkerung bedeutende
Veraenderungen wider."
)

ggplot(data = dat_long) +
  aes(x = value, y = country, color = fct_rev(year)) +
  facet_wrap(~ statistic, scales = "free_x") +
  geom_segment(
    data = dat_wide,
    aes(x = year_1952, xend = year_2007, y = country, yend = country),
    color = "#AAAAAA", linewidth = 1
  ) +
  geom_point(size = 3) +
  scale_color_manual(
    limits = c("1952", "2007"),
    values = year_colors, guide = "none"
  ) +
  scale_y_discrete(name = NULL) +
  scale_x_continuous(name = NULL) +
  labs(
    title = "BIP, LEBENSERWARTUNG & BEVOELKERUNG",
    subtitle = subtitle_text
  ) +
  theme_nature()
```



Das ergibt bereits einen nuetzlichen Ueberblick, aber zwei Dinge fallen auf: Die Achsenbeschriftungen fuer Bevoelkerung und BIP sind Rohzahlen in Millionen (unlesbar), und die Standard-Facetten-Strip-Labels zeigen nur die Variablennamen aus den Daten (`gdpPercap`, `lifeExp`, `pop`). Beide Probleme lassen sich leicht beheben.

Benutzerdefinierte Facetten- und Datenlabels

Ein Grundprinzip guter Visualisierung ist, dass der Leser niemals im Kopf rechnen muss. “130000000” auf einer Achse anzuzeigen, wenn man “130m” schreiben koennte, ist eine unnoetige kognitive Belastung. Das {scales}-Paket bietet `number()` fuer genau diesen Zweck - es formatiert Zahlen mit benutzerdefinierter Genauigkeit, Skalierung und Suffixen.

Wir erstellen formatierte Wertlabels fuer jede Statistik und berechnen ausserdem Differenzlabels (“+38”, “+2.1k”) fuer den Dumbbell Plot:

```
dat_long <- dat_long %>%
  mutate(value_lab = case_when(
    statistic == "lifeExp"   ~ number(value, accuracy = 1),
    statistic == "pop"       ~ number(value, accuracy = 1, scale = 1/1e6, suffix =
"m"),
    statistic == "gdpPercap" ~ number(value, accuracy = 0.1, scale = 1/1e3, suffix
= "k")
  ))

dat_wide <- dat_wide %>%
  mutate(
    diff_lab = case_when(
      statistic == "lifeExp" ~ number(
        diff, style_positive = "plus", style_negative = "minus", accuracy = 1
      ),
      statistic == "pop" ~ number(
        diff, style_positive = "plus", style_negative = "minus",
        accuracy = 1, scale = 1/1e6, suffix = "m"
      ),
      statistic == "gdpPercap" ~ number(
        diff, style_positive = "plus", style_negative = "minus",
        accuracy = 0.1, scale = 1/1e3, suffix = "k"
```

```
      )
    ),
    x_pos_lab = case_when(
      statistic == "lifeExp"   ~ max_x + 3,
      statistic == "pop"       ~ max_x + 10000000,
      statistic == "gdpPercap" ~ max_x + 3000
    )
  )
```

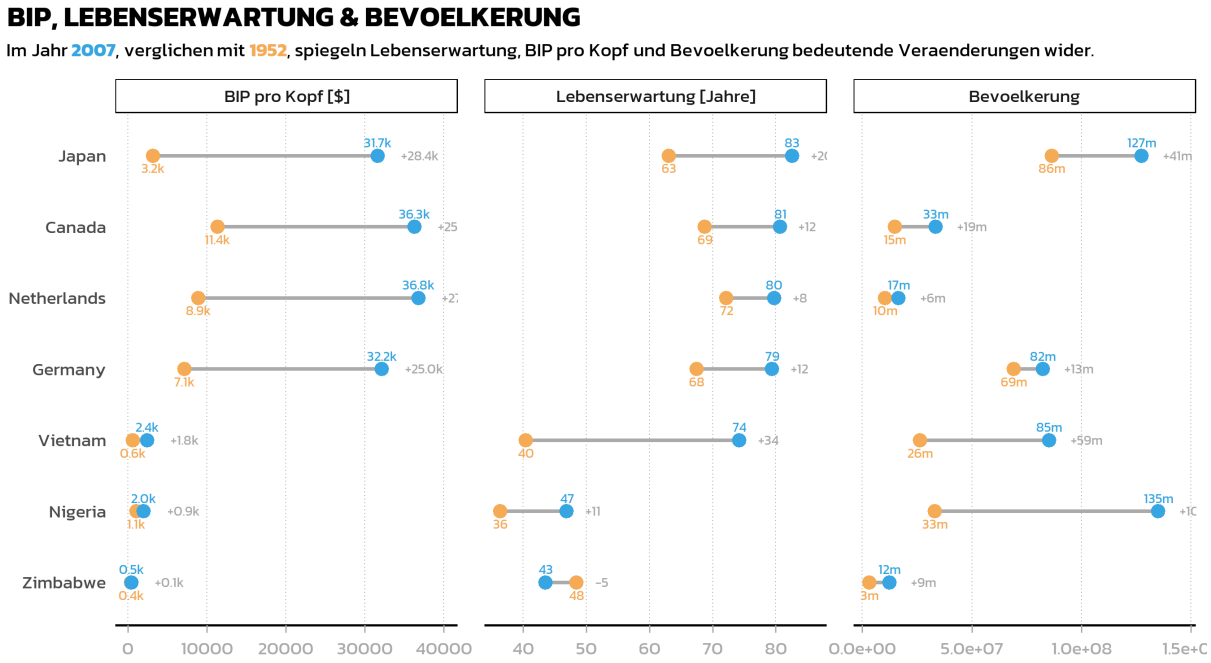
Als naechstes definieren wir lesbare Namen fuer die Facetten-Strip-Labels. Die `labeller()`-Funktion akzeptiert einen benannten Vektor, der interne Variablennamen auf Anzeige-Labels abbildet:

```
facet_labels <- c(
  lifeExp = "Lebenserwartung [Jahre]",
  pop = "Bevoelkerung",
  gdpPercap = "BIP pro Kopf [$]"
)
```

Nun fuegen wir den vollstaendigen facettierten Dumbbell Plot zusammen und kombinieren alle Elemente aus dem vorherigen Kapitel (Segmente, Punkte, Labels, Custom Theme) mit der Facetting-Schicht:

```
p3 <- ggplot(data = dat_long) +
  aes(x = value, y = country, color = fct_rev(year)) +
  facet_wrap(
    ~ statistic,
    scales = "free_x",
    labeller = labeller(statistic = facet_labels)
  ) +
  geom_segment(
    data = dat_wide,
    aes(x = year_1952, xend = year_2007, y = country, yend = country),
    color = "#AAAAAA", linewidth = 1
  ) +
  geom_point(size = 3) +
  geom_text(
    mapping = aes(
      label = value_lab,
      vjust = if_else(year == "1952", 2, -1)
    ),
    size = 2.5, family = "kanit"
  ) +
  geom_text(
    data = dat_wide,
    mapping = aes(x = x_pos_lab, label = diff_lab),
    size = 2.5, hjust = 0,
    color = "#AAAAAA", family = "kanit"
  ) +
  scale_color_manual(
    limits = c("1952", "2007"),
    values = year_colors, guide = "none"
  ) +
  scale_y_discrete(name = NULL) +
  scale_x_continuous(name = NULL) +
  labs(
    title = "BIP, LEBENSERWARTUNG & BEVOELKERUNG",
    subtitle = subtitle_text
  ) +
  theme_nature() +
  theme(
    panel.spacing = unit(1, "lines"),
    strip.background = element_rect(fill = NA, color = "black")
  )
```

```
p3
```



Ein nuetzlicher Trick in diesem Code ist `vjust = if_else(year == "1952", 2, -1)` innerhalb von `aes()`: Damit werden 1952-Labels unter und 2007-Labels ueber ihren Punkten platziert, was Ueberlappungen bei nahe beieinanderliegenden Werten verhindert. Da `vjust` innerhalb von `aes()` steht, wird es pro Beobachtung berechnet, was eine bedingte Positionierung ermoeglicht.

Die Anpassungen `panel.spacing` und `strip.background` in `theme()` sorgen fuer visuelle Trennung zwischen den Facetten: mehr Abstand zwischen den Panels und ein dezenter Rahmen um die Strip-Labels.

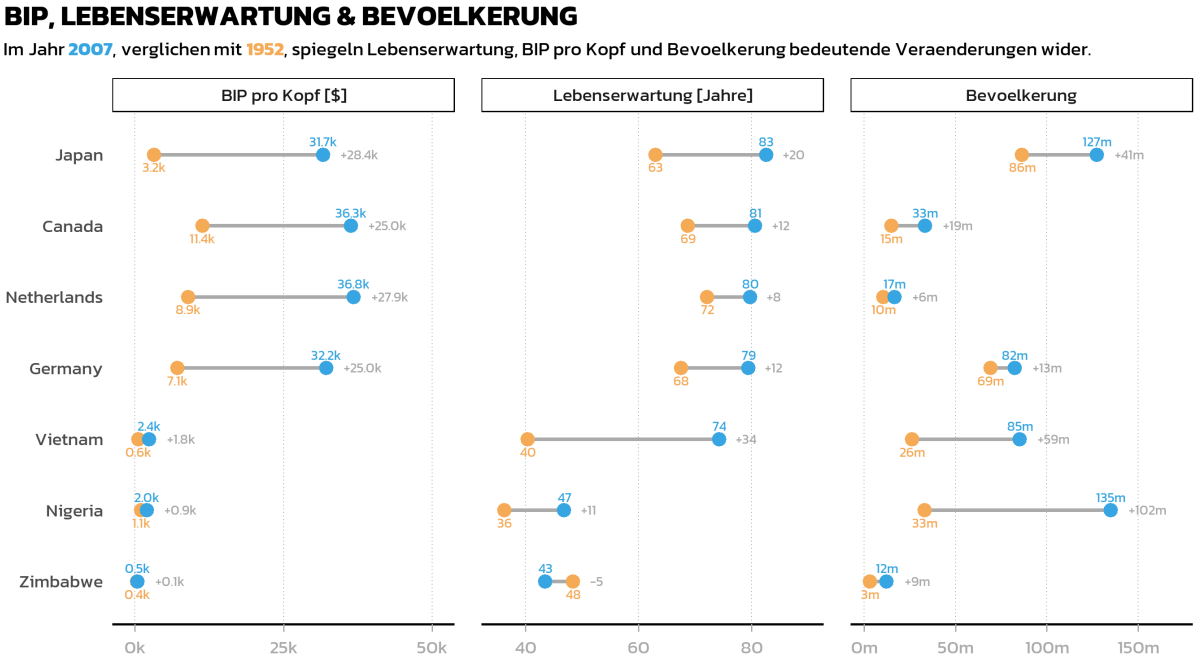
Individuelle Skalen pro Facette mit ggh4x

Waehrend `scales = "free_x"` jeder Facette ihren eigenen Achsenbereich gibt, wendet Standard-ggplot2 dieselbe Achsen-*Formatierung* auf alle Facetten an. In unserem Fall brauchen wir grundlegend unterschiedliche Formatierungen: einfache Zahlen fuer Lebenserwartung (35, 55, 75), Millionen mit “m”-Suffix fuer Bevoelkerung (0m, 50m, 100m) und Tausend mit “k”-Suffix fuer BIP (0k, 25k). Das {ggh4x}-Paket bietet `facetted_pos_scales()` fuer genau diesen Zweck - es erlaubt individuelle `scale_x_continuous()`-Definitionen pro Facette.

```
p3 <- p3 + facetted_pos_scales(
  x = list(
    statistic == "lifeExp" ~ scale_x_continuous(
      limits = c(35, 90),
      breaks = breaks_width(20),
      labels = number_format(accuracy = 1)
    ),
    statistic == "pop" ~ scale_x_continuous(
      limits = c(0, 150000000),
      expand = expansion(mult = c(0.05, 0.2)),
      breaks = breaks_width(50000000),
```

```
      labels = number_format(accuracy = 1, scale = 1/1e6, suffix = "m")
    ),
    statistic == "gdpPercap" ~ scale_x_continuous(
      limits = c(0, 50000),
      expand = expansion(mult = c(0.075, 0.075)),
      breaks = breaks_width(25000),
      labels = number_format(accuracy = 1, scale = 1/1e3, suffix = "k")
    )
  )
) + xlab(NULL)

p3
```



Jede Facette hat nun korrekt formatierte Achsenlabels: “35, 55, 75” fuer Lebenserwartung, “0m, 50m, 100m” fuer Bevoelkerung und “0k, 25k” fuer BIP. Die `limits`- und `expand`-Argumente pro Facette sind individuell abgestimmt, damit keine Datenlabels an den Raendern abgeschnitten werden.

Die Formelsyntax (`statistic == "lifeExp" ~ scale_x_continuous(...)`) macht den Code gut lesbar: Jede Zeile gibt klar an, welche Facette welche Skalenkonfiguration erhaelt. Das ist eine der nuetzlichsten Funktionen von ggh4x und lohnt sich zu merken, wann immer Facetten Variablen mit grundlegend unterschiedlichen Einheiten oder Groessenordnungen darstellen.

> **Tipp**
>
> Das {scales}-Paket ist das Rueckgrat der Achsenlabel-Formatierung. Wichtige Funktionen sind:
>
> - `number_format()` - allgemeine Zahlenformatierung mit `accuracy`, `scale`, `suffix`
> - `breaks_width()` - Break-Intervalle nach Breite setzen
> - `label_percent()`, `label_comma()`, `label_dollar()` - spezialisierte Formatierer

Bibliography