9. patchwork

Mehrere Plots in einer Abbildung kombinieren

Dr. Paul Schmidt

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

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

Wissenschaftliche Publikationen und Praesentationen erfordern oft mehrere zusammengehoerige Plots in einer einzigen Abbildung - ein Balkendiagramm neben einem Dumbbell Plot oder eine Zeitreihe ueber einem Streudiagramm. Waehrend `facet_wrap()` Faelle abdeckt, in denen derselbe Plottyp fuer verschiedene Teilmengen wiederholt wird, kann es keine grundlegend verschiedenen Plottypen oder Datensaetze kombinieren. Das {patchwork}-Paket von Thomas Lin Pedersen fuellt diese Luecke mit einer intuitiven operatorbasierten Syntax zum Zusammensetzen von ggplot2-Objekten.

In diesem Kapitel erstellen wir das Balkendiagramm und den Dumbbell Plot aus Kapitel 5 neu und kombinieren sie mit patchworks Layout-Operatoren zu Multi-Panel-Abbildungen.

Setup

Wir reproduzieren die Plots aus Kapitel 5. Der Code ist eingeklappt, da er eine Wiederholung frueheren Materials ist:

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

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))

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),
      legend.position = "top",
      legend.box.just = "left",
      legend.justification = "left",
      legend.title = element_text(face = "bold"),
      legend.key.size = unit(0.4, "cm"),
      legend.margin = margin(-5, 0, 0, 0),
      panel.grid.minor = element_blank(),
      panel.grid.major.y = element_blank(),
      panel.grid.major.x = element_line(
        linetype = "dotted", color = "#AAAAAA", linewidth = 0.3
      )
    )
}

long_subtitle <- "Die Daten zeigen eine Welt, in der die Lebenserwartung im Jahr <b
style='color:#37A9E1;'>2007</b> oft eine verbesserte Lebensqualitaet im Vergleich
zu <b style='color:#F7AA59;'>1952</b> widerspiegelt."

# Balkendiagramm
p_bar <- ggplot(data = dat) +
  aes(x = lifeExp, y = country, fill = fct_rev(year)) +
  geom_col(position = position_dodge()) +
  geom_text(
    mapping = aes(label = round(lifeExp), group = fct_rev(year)),
    position = position_dodge(width = 0.9),
    hjust = 1.1, color = "white", family = "kanit"
  ) +
  scale_y_discrete(name = NULL, expand = c(0, 0)) +
  scale_x_continuous(name = NULL, expand = expansion(mult = c(0, 0.05))) +
  scale_fill_manual(
    guide = "none", name = "Year",
    limits = names(year_colors), values = year_colors
  ) +
  labs(title = "LEBENSERWARTUNG", subtitle = long_subtitle) +
  theme_nature()

# Dumbbell Plot
dat_wide <- dat %>%
  select(country, year, lifeExp) %>%
  pivot_wider(
    names_from = year, values_from = lifeExp, names_prefix = "year_"
  ) %>%
  mutate(
    max_x = pmax(year_2007, year_1952),
    diff = year_2007 - year_1952,
    diff_lab = sprintf("%+d", round(diff))
  )

p_dumbbell <- ggplot(data = dat) +
  aes(x = lifeExp, y = country, color = fct_rev(year)) +
  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 = round(lifeExp)),
    size = 2.5, vjust = -1, family = "kanit"
```

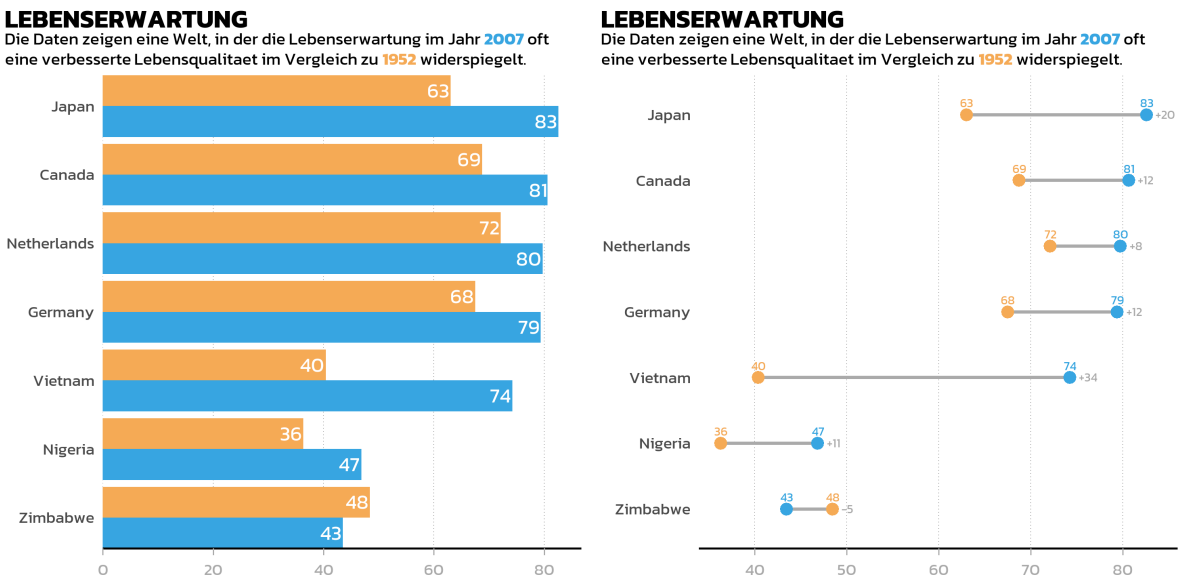
```
  ) +
  geom_text(
    data = dat_wide,
    mapping = aes(x = max_x, label = diff_lab),
    size = 2.5, hjust = 0,
    position = position_nudge(x = 1),
    color = "#AAAAAA", family = "kanit"
  ) +
  scale_color_manual(
    name = "Year", limits = c("1952", "2007"),
    values = year_colors, guide = "none"
  ) +
  scale_y_discrete(name = NULL) +
  scale_x_continuous(name = NULL) +
  labs(title = "LEBENSERWARTUNG", subtitle = long_subtitle) +
  theme_nature()

# Einfacher Scatter Plot fuer BIP
p_gdp <- ggplot(
  data = gapminder::gapminder %>% filter(year == 2007),
  aes(x = gdpPercap, y = lifeExp, size = pop, color = continent)
) +
  geom_point(alpha = 0.7) +
  scale_x_log10() +
  scale_size_continuous(guide = "none") +
  labs(x = "BIP pro Kopf", y = "Lebenserwartung", color = "Kontinent") +
  theme_minimal(base_size = 12) +
  theme(text = element_text(family = "kanit"))
```

Wir haben nun drei Plot-Objekte: `p_bar` (gruppiertes Balkendiagramm), `p_dumbbell` (Dumbbell Plot) und `p_gdp` (Streudiagramm).

Plots kombinieren

Der einfachste Weg, zwei Plots zu kombinieren, ist der `+`-Operator. Standardmaessig ordnet patchwork Plots nebeneinander an:

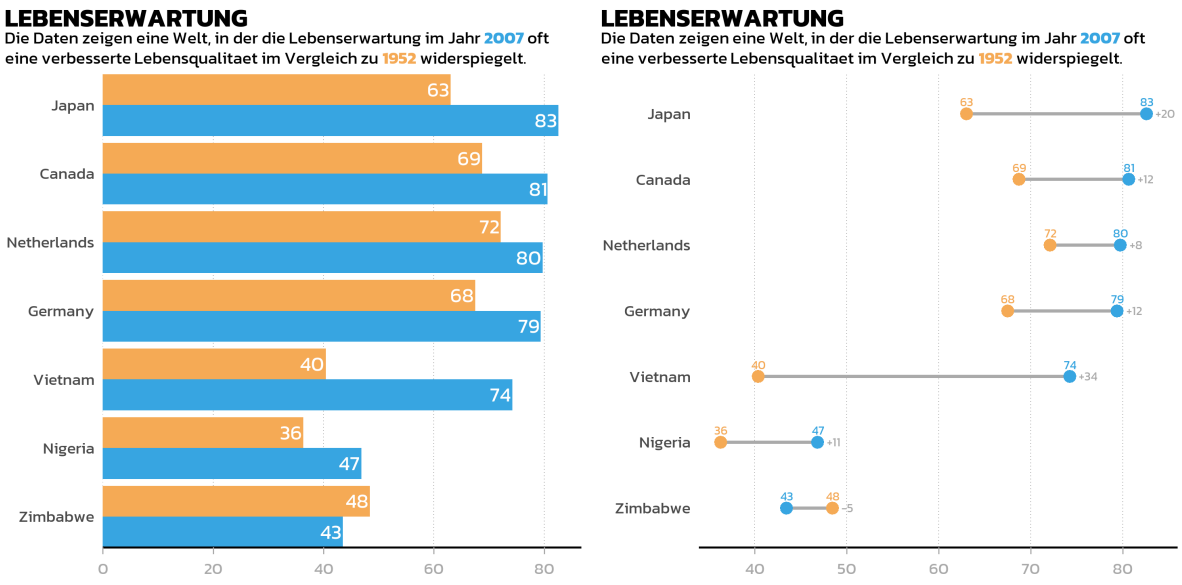
```
p_bar + p_dumbbell
```



Nebeneinander: der `|`-Operator

Der `|`-Operator ist die explizite Version der Nebeneinander-Anordnung. Er verhaelt sich identisch zu `+` fuer zwei Plots, macht aber die Absicht in komplexen Layouts klarer:

```
p_bar | p_dumbbell
```



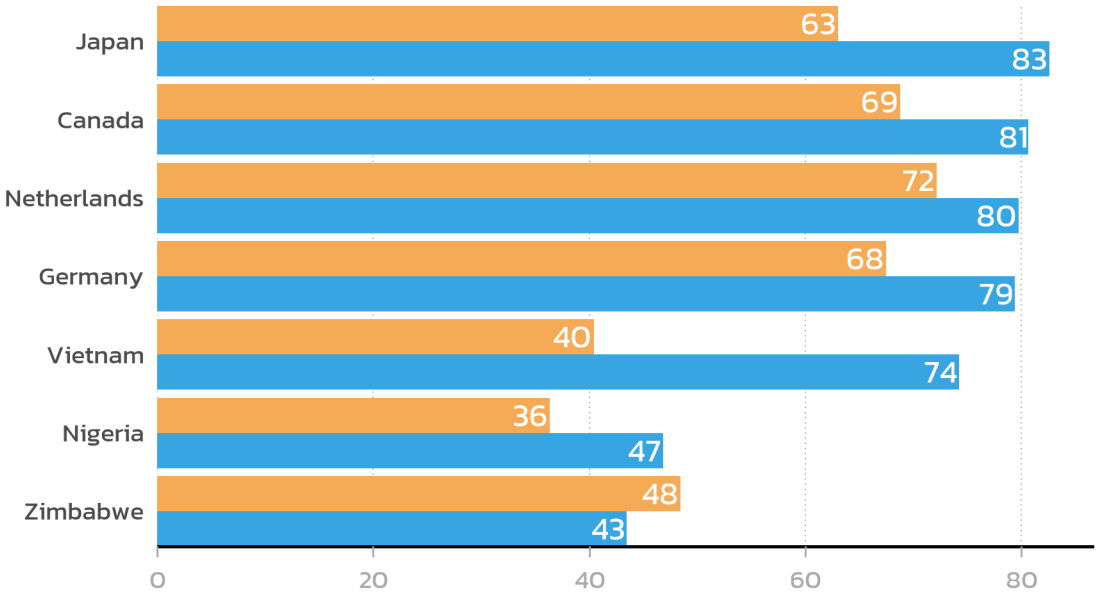
Uebereinander: der `/`-Operator

Der `/`-Operator platziert Plots uebereinander:

```
p_bar / p_dumbbell
```

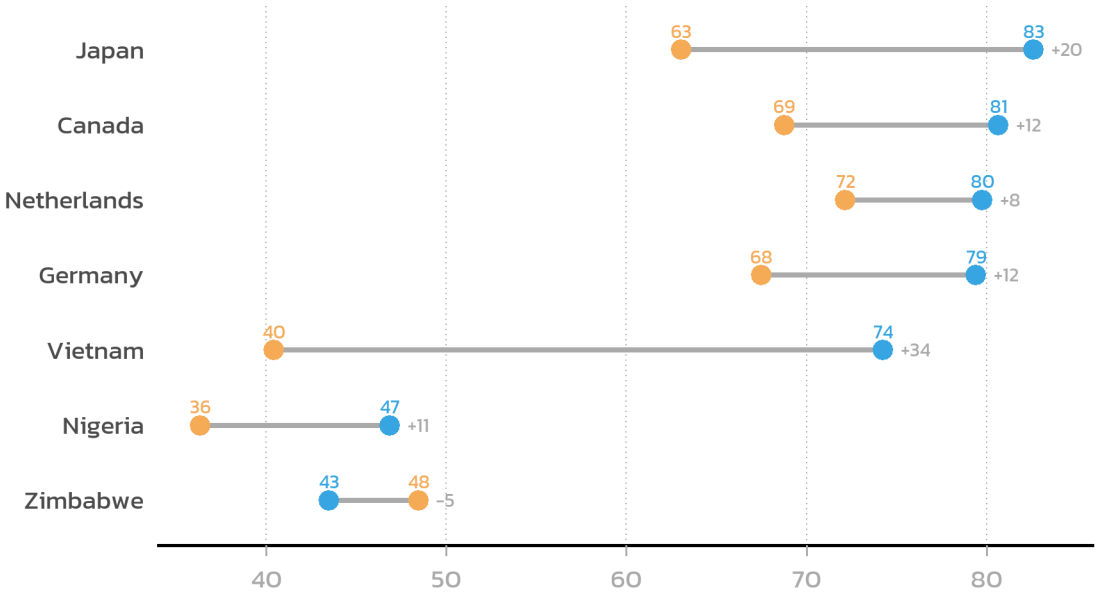
LEBENSERWARTUNG

Die Daten zeigen eine Welt, in der die Lebenserwartung im Jahr 2007 oft eine verbesserte Lebensqualitaet im Vergleich zu 1952 widerspiegelt.



LEBENSERWARTUNG

Die Daten zeigen eine Welt, in der die Lebenserwartung im Jahr 2007 oft eine verbesserte Lebensqualitaet im Vergleich zu 1952 widerspiegelt.



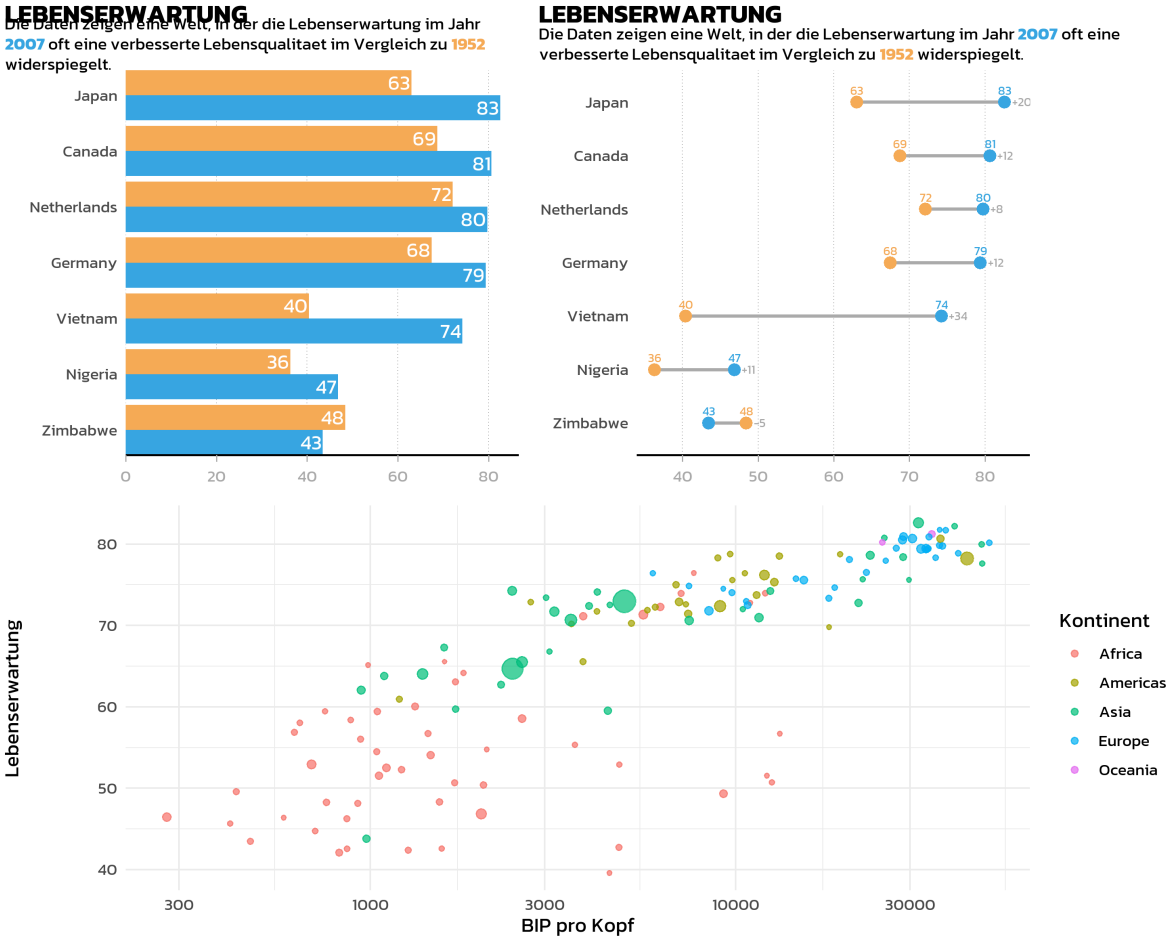
Operator-Praezedenz

Beim Kombinieren von Operatoren spielt die **Praezedenz** eine Rolle. Der `/`-Operator (Stapeln) bindet staerker als `|` (Nebeneinander), sodass `p1 | p2 / p3` als `p1 | (p2 / p3)` interpretiert wird statt als `(p1 | p2) / p3`. Der `+`-Operator hat die niedrigste Praezedenz. In der Praxis bedeutet das: Man sollte immer **Klammern** verwenden, um das beabsichtigte Layout explizit zu machen — sich auf Praezedenzregeln zu verlassen fuehrt zu Ueberraschungen.

Verschachtelte Layouts

Klammern machen die Layout-Struktur explizit und lesbar. Zum Beispiel zwei Plots nebeneinander oben, mit einem dritten Plot ueber die volle Breite unten:

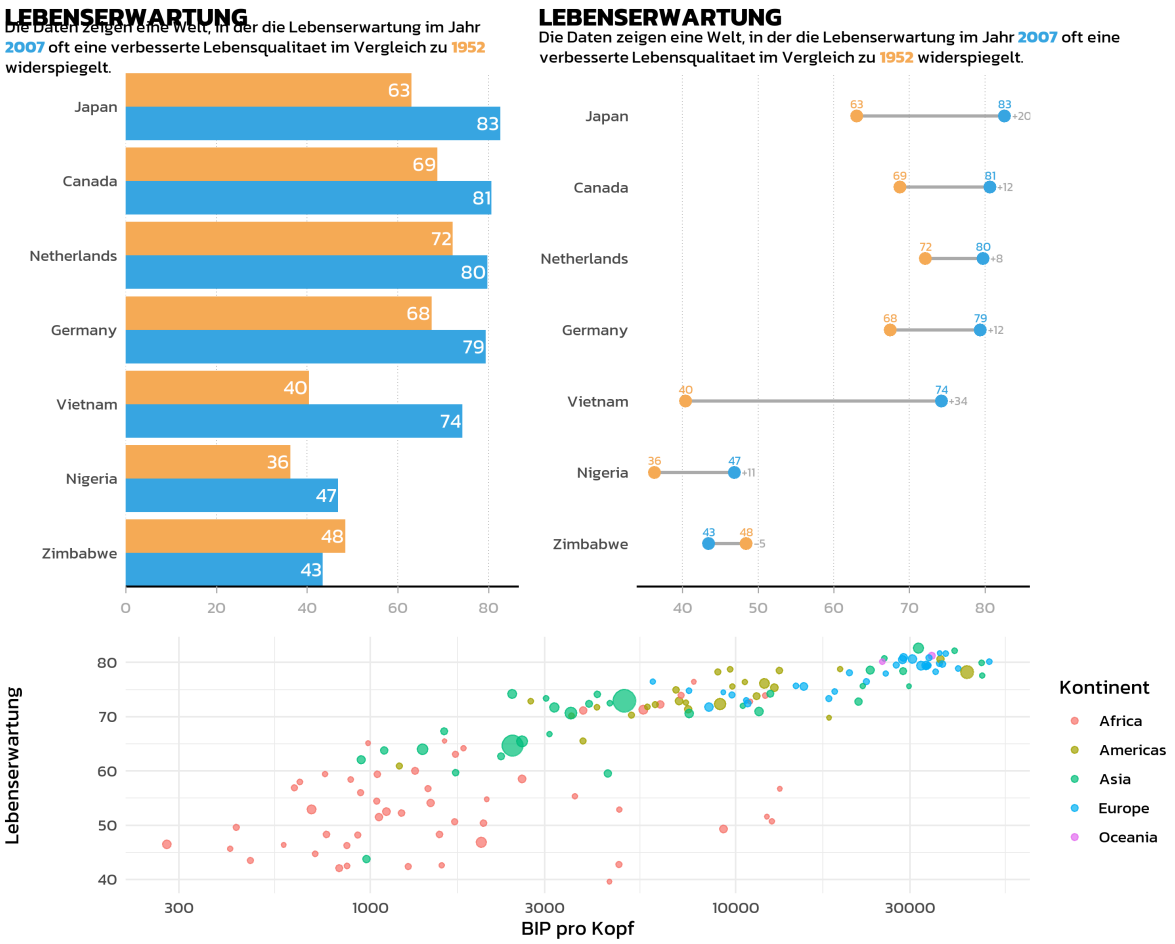
```
(p_bar | p_dumbbell) / p_gdp
```



Layout-Steuerung

Die `plot_layout()`-Funktion bietet feingranulare Kontrolle ueber die Anordnung:

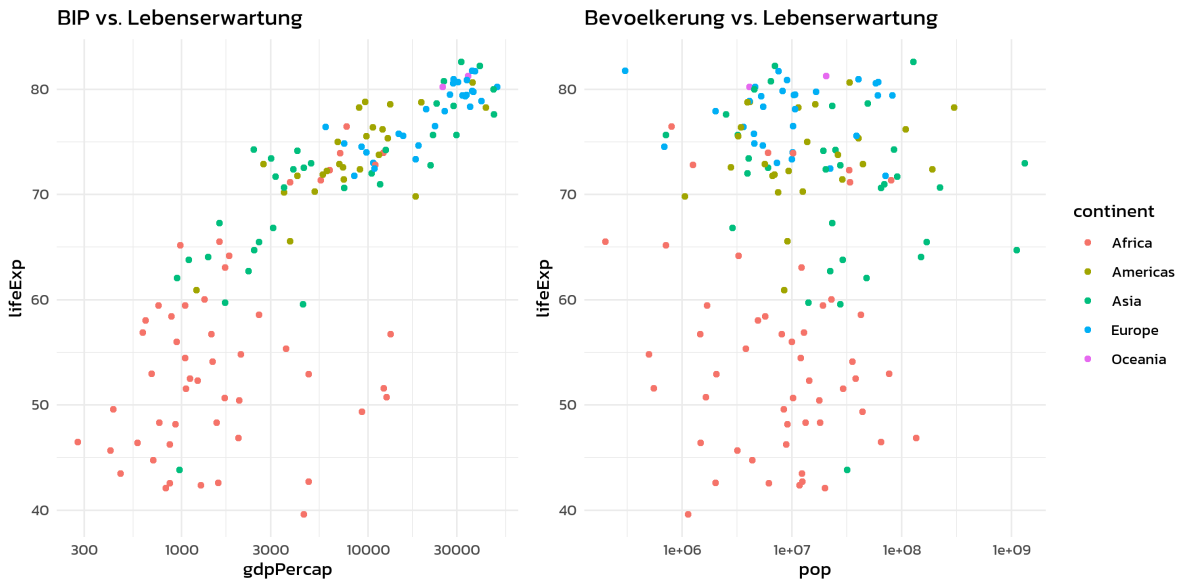
```
(p_bar | p_dumbbell) / p_gdp +
  plot_layout(heights = c(2, 1))
```



Wichtige Argumente:

- **`widths`** und **`heights`**: Relative Groessen von Spalten und Zeilen. `heights = c(2, 1)` macht die obere Reihe doppelt so hoch wie die untere.
- **`ncol`** und **`nrow`**: Erzwingt ein bestimmtes Gitterlayout.
- **`guides = "collect"`**: Sammelt identische Legenden aus allen Plots und zeigt sie einmal. Besonders nuetzlich, wenn mehrere Plots dieselbe Farbskala teilen.

```
p1 <- ggplot(gapminder::gapminder %>% filter(year == 2007),
  aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point() + scale_x_log10() + theme_minimal(base_size = 11) +
  theme(text = element_text(family = "kanit")) +
  labs(title = "BIP vs. Lebenserwartung")

p2 <- ggplot(gapminder::gapminder %>% filter(year == 2007),
  aes(x = pop, y = lifeExp, color = continent)) +
  geom_point() + scale_x_log10() + theme_minimal(base_size = 11) +
  theme(text = element_text(family = "kanit")) +
  labs(title = "Bevoelkerung vs. Lebenserwartung")

p1 + p2 + plot_layout(guides = "collect")
```

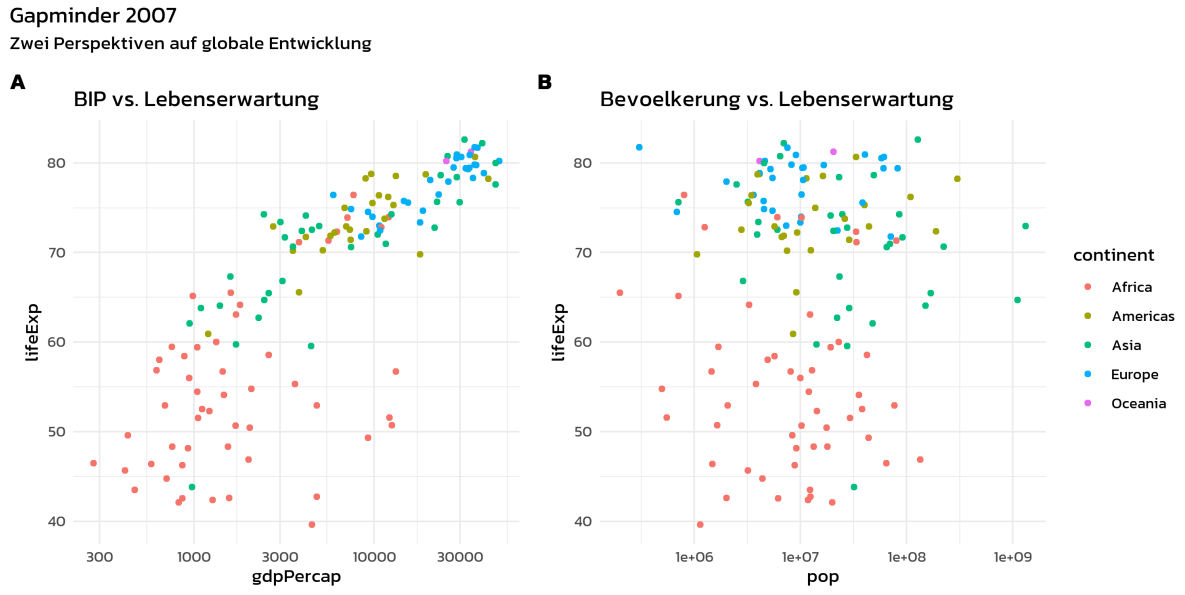


Ohne `guides = "collect"` wuerde jeder Plot seine eigene Kontinent-Legende zeigen - redundant und platzverschwendend.

Annotationen

`plot_annotation()` fuegt Titel, Untertitel, Quellenangaben und automatische Panel-Tags zur kombinierten Abbildung hinzu:

```
p1 + p2 +
  plot_layout(guides = "collect") +
  plot_annotation(
    title = "Gapminder 2007",
    subtitle = "Zwei Perspektiven auf globale Entwicklung",
    tag_levels = "A"
  ) &
  theme(
    text = element_text(family = "kanit"),
    plot.tag = element_text(face = "bold")
  )
```



Das `tag_levels`-Argument beschriftet Panels automatisch als “A”, “B”, “C” (oder “1”, “2”, “3”, “a”, “b”, “c”, “I”, “II”, “III”). Das ist essenziell fuer Journal-Abbildungen, in denen Panels im Text referenziert werden muessen.

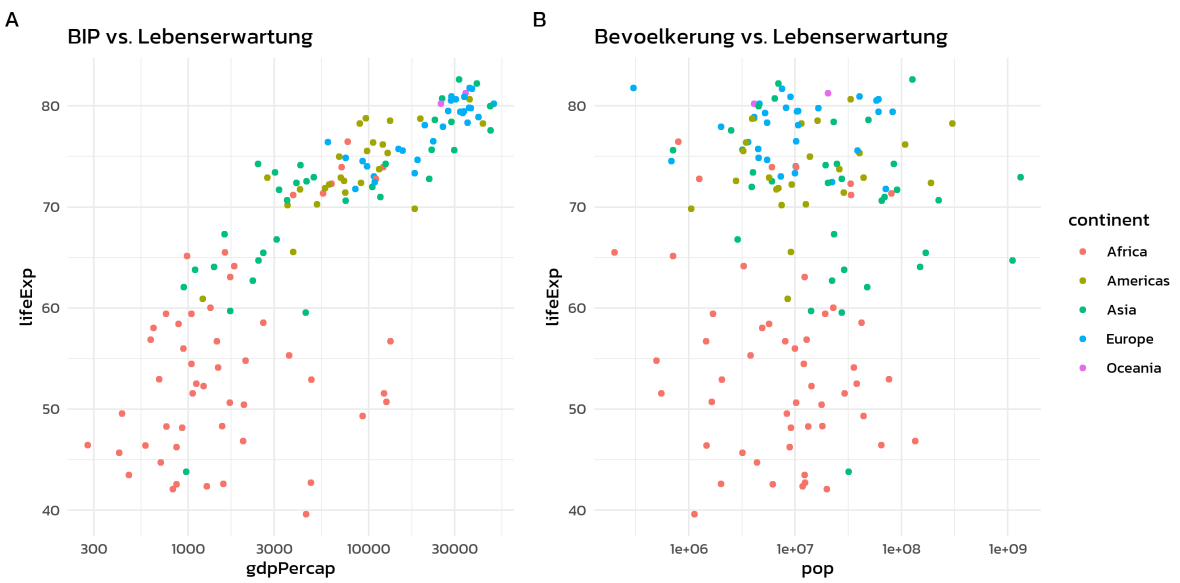
Der `&`-Operator

Man beachte den `&`-Operator im letzten Beispiel. Das ist eine der nuetzlichsten Funktionen von patchwork:

- `+` fuegt eine Theme-Aenderung nur zum **letzten** Plot hinzu
- `&` wendet eine Theme-Aenderung auf **alle** Plots in der Komposition an

Diese Unterscheidung ist wichtig, wenn man ein einheitliches Theme ueber alle Panels hinweg anwenden moechte:

```
p1 + p2 +
  plot_layout(guides = "collect") +
  plot_annotation(tag_levels = "A") &
  theme_minimal(base_size = 11) &
  theme(text = element_text(family = "kanit"))
```



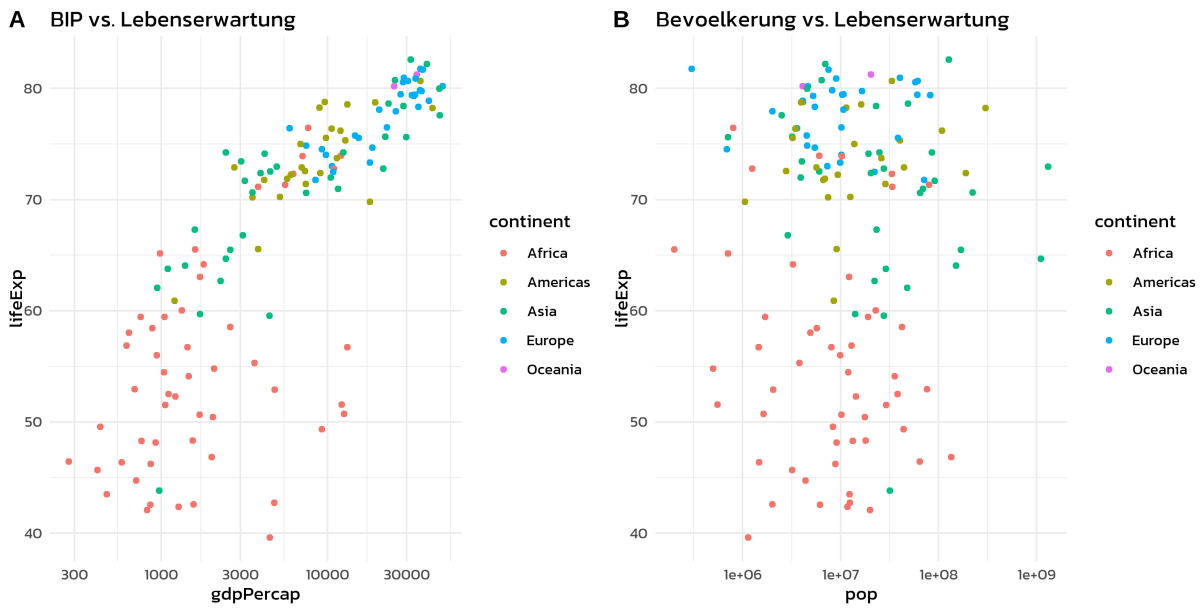
Ueber patchwork hinaus: cowplot

Das {cowplot}-Paket von Claus Wilke bietet ergaenzende Funktionalitaet zum Kombinieren von Plots. Waehrend patchwork bei gitterbasierten Layouts mit seiner Operatorsyntax glaenzt, punktet cowplot in zwei spezifischen Bereichen: automatische Panel-Beschriftungen und Inset-Plots.

Einfaches Gitter mit cowplot

Die `cowplot::plot_grid()`-Funktion arbeitet aehnlich wie patchworks `+`-Operator, bietet aber Panel-Beschriftungen direkt ueber das `labels`-Argument:

```
cowplot::plot_grid(p1, p2, labels = "AUTO")
```

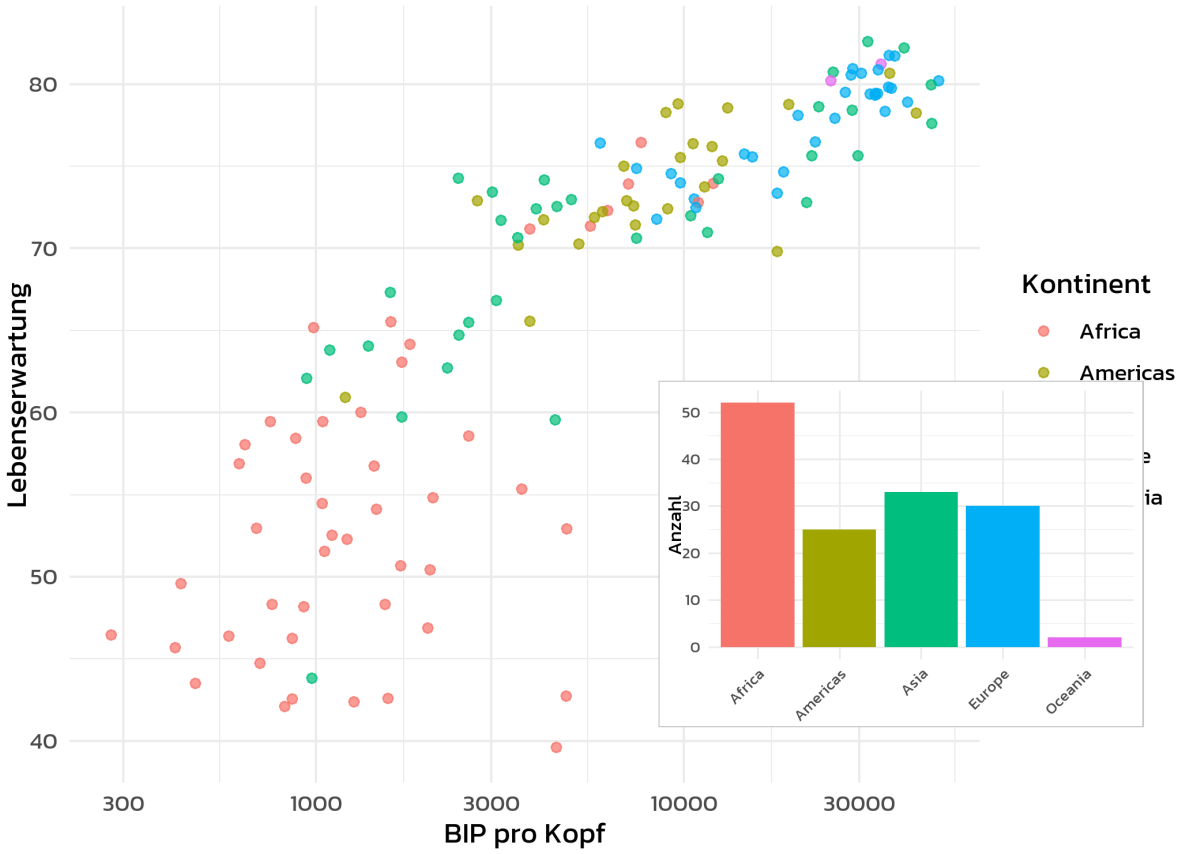


`labels = "AUTO"` erzeugt Grossbuchstaben-Labels (A, B, C, …), waehrend `labels = "auto"` Kleinbuchstaben liefert (a, b, c, …).

Inset-Plots

Eine Funktion, die patchwork nicht bietet, ist das Platzieren eines kleinen Plots *innerhalb* eines groesseren - ein Inset-Plot. Das ist nuetzlich, um ein vergroessertes Detail, eine zusammenfassende Statistik oder eine ergaenzende Ansicht innerhalb derselben Abbildung zu zeigen. Die Funktionen `cowplot::ggdraw()` und `cowplot::draw_plot()` machen das unkompliziert:

```
dat_2007 <- gapminder::gapminder %>% filter(year == 2007)

p_main <- ggplot(dat_2007, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point(alpha = 0.7) +
  scale_x_log10() +
  labs(x = "BIP pro Kopf", y = "Lebenserwartung", color = "Kontinent") +
  theme_minimal(base_size = 12) +
  theme(text = element_text(family = "kanit"))

p_inset <- ggplot(dat_2007, aes(x = continent, fill = continent)) +
  geom_bar(show.legend = FALSE) +
  labs(x = NULL, y = "Anzahl") +
  theme_minimal(base_size = 8) +
  theme(
    text = element_text(family = "kanit"),
    axis.text.x = element_text(angle = 45, hjust = 1),
    plot.background = element_rect(fill = "white", color = "grey80")
  )

cowplot::ggdraw(p_main) +
  cowplot::draw_plot(p_inset, x = 0.55, y = 0.15, width = 0.4, height = 0.4)
```



Die Argumente `x`, `y`, `width` und `height` in `draw_plot()` verwenden relative Koordinaten (0 bis 1), wobei (0, 0) die untere linke Ecke ist. Durch Anpassen dieser Werte positioniert und dimensioniert man den Inset-Plot innerhalb der Hauptabbildung.

Fuer die meisten Multi-Panel-Layouts bleibt patchwork aufgrund seiner saubereren Syntax und automatischen Ausrichtung die bessere Wahl. cowplot lohnt sich, wenn man Inset-Plots benoetigt oder das `plot_grid()`-Interface fuer einfache Nebeneinander-Anordnungen mit automatischen Labels bevorzugt.

> Wann Facets und wann patchwork?
>
> Beide Werkzeuge erstellen Multi-Panel-Abbildungen, aber sie dienen unterschiedlichen Zwecken:
>
> - **Facets** (`facet_wrap()`, `facet_grid()`): Gleicher Plottyp, gleiche Datenstruktur, aufgeteilt nach einer Gruppierungsvariable. Achsen werden geteilt oder individuell skaliert. Am besten fuer systematische Vergleiche innerhalb eines Datensatzes.
> - **Patchwork**: Verschiedene Plottypen, verschiedene Datensaetze oder verschiedene aesthetische Mappings. Jedes Panel ist ein voellig unabhaengiger ggplot. Am besten zum Kombinieren komplementaerer Sichten auf die Daten.
>
> Eine gute Faustregel: Wenn man die Panels als “denselben Plot fuer verschiedene Gruppen” beschreiben kann, nimmt man Facets. Wenn die Panels grundlegend verschiedene Dinge zeigen, nimmt man patchwork.

Bibliography