

Publication-ready ggplot2

Scientific workflows: Tools and Tips 

2026-05-21

What is this lecture series?

Scientific workflows: Tools and Tips

 Every 3rd Thursday  4-5 p.m.  Webex

- One topic from the world of scientific workflows
- Material provided [online](#)
- If you don't want to miss a lecture
 - [Subscribe to the mailing list](#)
- For credit points: Send me a short message (Email or Webex)

From default to publication-ready

- Making a first plot is easy, but making it publication-ready takes more work, choices and iterations

So many things you can tune:

- The plot itself: which geoms to pick
- Themes, colours, fonts
- How to highlight your results
- Creating multi-panel plots
- Exporting at a readable size

Different outputs (poster/talk/journal figure) require different tweaks

Today

We'll focus on:

1. **Custom themes** for a consistent look
2. **Colour** choice
3. **Multi-panel layouts**: how to combine plots
4. **Exporting** plots at the right size and resolution

Note

This is not an exhaustive list, there is so much more ggplot extension packages and tricks out there.

How it works

- There's a **GitHub repo** you download and open in your IDE (I'll use Positron)
- I demonstrate each topic: **follow along or just watch**
- Each module has a short **exercise** to play with the concept
- **Questions** anytime: in the chat or just unmute

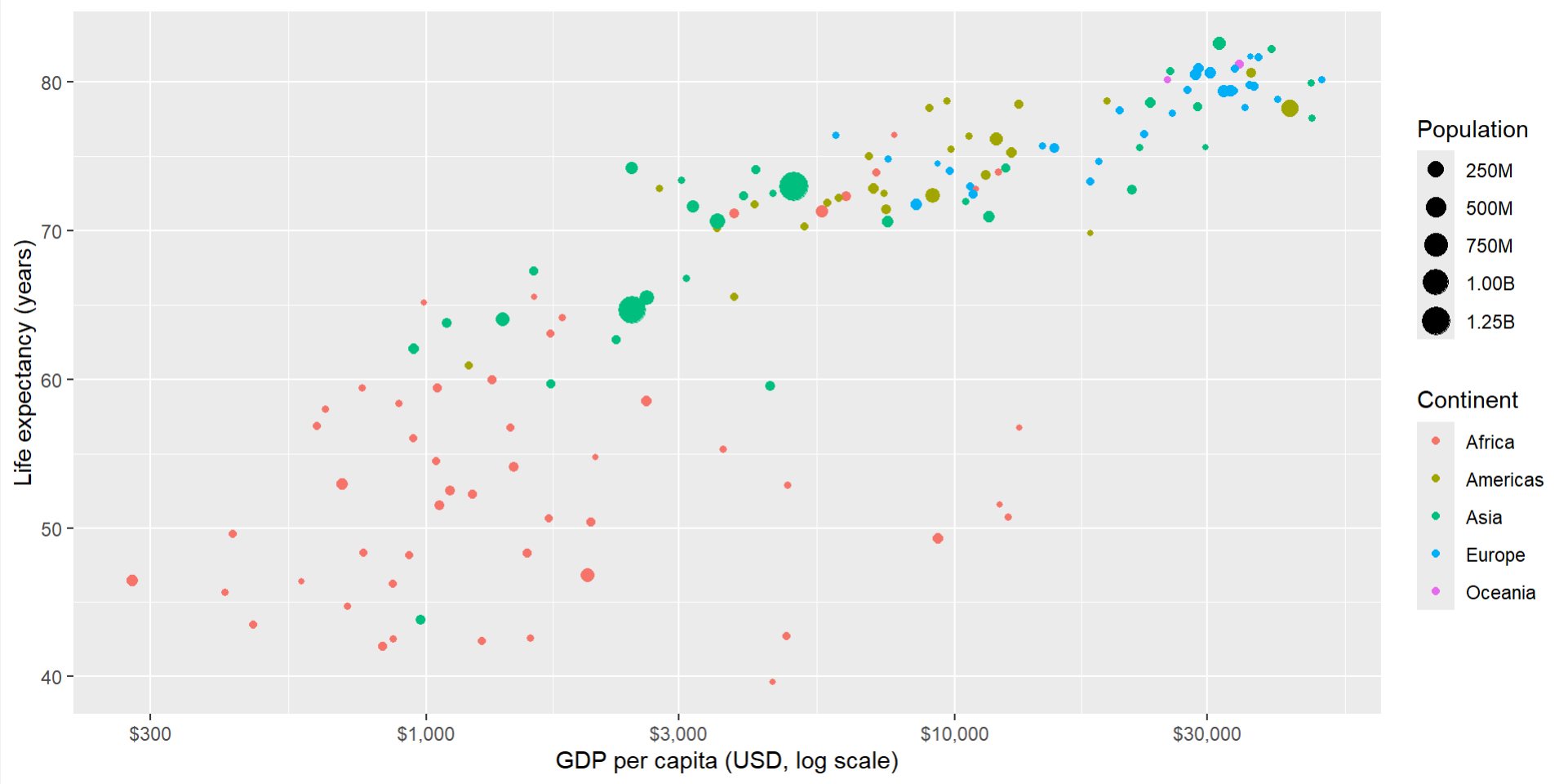
Get set up

1. Click the green **Code** button → **Download ZIP**
2. Unzip it somewhere you can find it
3. Double-click the `.Rproj` file → opens in RStudio
 - *(Positron/VS code users: open the folder)*
4. Run `install_packages.R` to install today's packages

Repo: <https://github.com/selinaZitrone/advanced-ggplot-workshop>

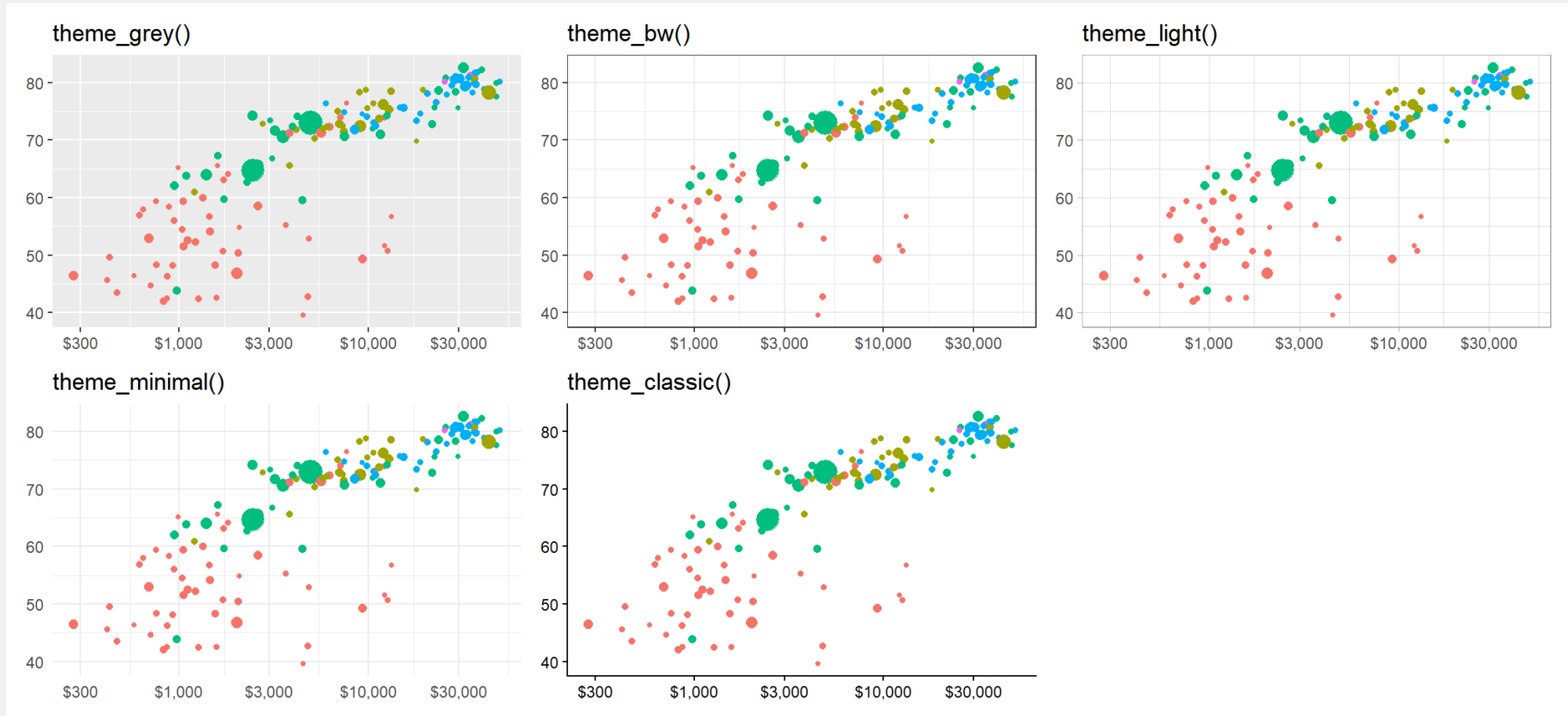
1. Custom themes

The default isn't publication-ready



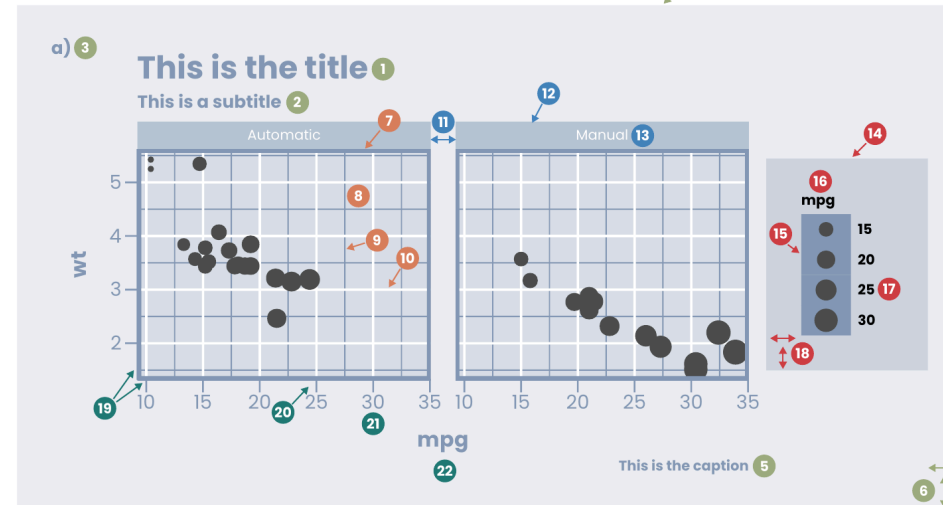
Built-in themes: a quick win

Add a `theme_*()` layer and the whole look changes.



Further customization with `theme()`

A theme controls everything that isn't data: Fonts, sizes, colours, grid lines, spacing, legends, ...



Plot elements	Panel elements	Legend elements	Axis elements
1 plot.title element_text()	7 panel.border element_rect()	14 legend.background element_rect()	19 axis.line element_line()
2 plot.subtitle element_text()	8 panel.background element_rect()	15 legend.key element_rect()	20 axis.ticks element_line()
3 plot.tag element_text()	9 panel.grid.minor element_line()	16 legend.title element_text()	axis.ticks.length unit()
plot.tag.position "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright" or a coordinate	10 panel.grid.major element_line()	legend.title.align Numeric between 0 to 1, where: 0=left, 1=right	21 axis.text element_text()
4 plot.background element_rect()	aspect.ratio numeric	17 legend.text element_text()	22 axis.title element_text()
5 plot.caption element_text()	Facet elements	legend.text.align Numeric between 0 to 1, where: 0=left, 1=right	Global These affect all elements of same type in the plot. Useful to define defaults.
6 plot.margin margin()	11 panel.spacing unit()	18 legend.margin margin()	text element_text() line element_line() rect element_rect() title element_title()
	12 strip.background element_rect()	legend.position "none", "left", "right", "bottom", "top", or two-element numeric vector	
	13 strip.text element_text()		

Element functions

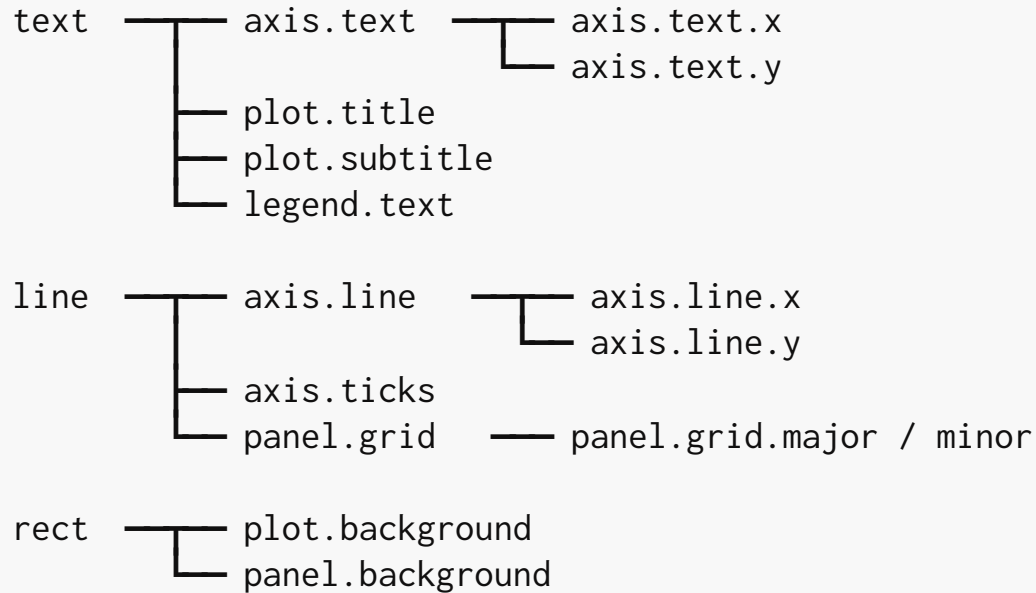
- element_text()**
(font) family
(font) face
(font) colour
(font) size (in points)
hjust [0..1] (0=left, 1=right)
vjust [0..1] (0=bottom, 1=top)
angle (in degrees)
lineheight (as ratio of fontcase)
margin
margin (t, r, b, l)
#remember trouble
- element_line()**
(line) colour
size (width of line)
linetype
An integer (0.8)
A name ("blank", "solid",
"dashed", "dotted", "dotdash",
"longdash", "twodash")
lineend
"round", "butt", "square"
arrow
An arrow specification: arrow()
- element_rect()**
fill
colour
size (width of border)
linetype (of border) (see element_line)
- element_blank()**
Eliminates element
Doesn't take parameters

Note.
Of those elements that have two components, the way to access is by appending *x* or *y* at the end. e.g. *axis.line.y* will change only the "y" axis line. Idem with "x". If nothing is specified (e.g. *axis.line*), both elements (*x* and *y*) will be changed.

This material is part of the course: [Learn ggplot2 in R for Data Visualization](https://www.udemy.com/course/learn-ggplot2-in-r-for-data-visualization/)
<https://www.udemy.com/course/learn-ggplot2-in-r-for-data-visualization/?referralCode=E98592E4E943F800ECB8>
See the full list of Theme Elements here: <https://ggplot2.tidyverse.org/reference/theme.html>

ggplot2 theme-system cheatsheet (by Clara Granell)

Inheritance of theme elements



- Set a **parent** and it affects all its **children**
- E.g. Change **text** once and every piece of text updates

How to change a theme

```
p_bubble +  
  theme_light(base_size = 18) +  
  theme(  
    legend.text = element_text(size = rel(0.85)),  
    panel.grid.minor = element_blank()  
  )
```

Live demo

Open `demo/01_themes.R`

Your turn

 Exercise (~7 min):

Open `exercises/01_themes_exercise.R`

2. Colour with intent

Pick intuitive colours

Use colours your reader already associates with the thing.

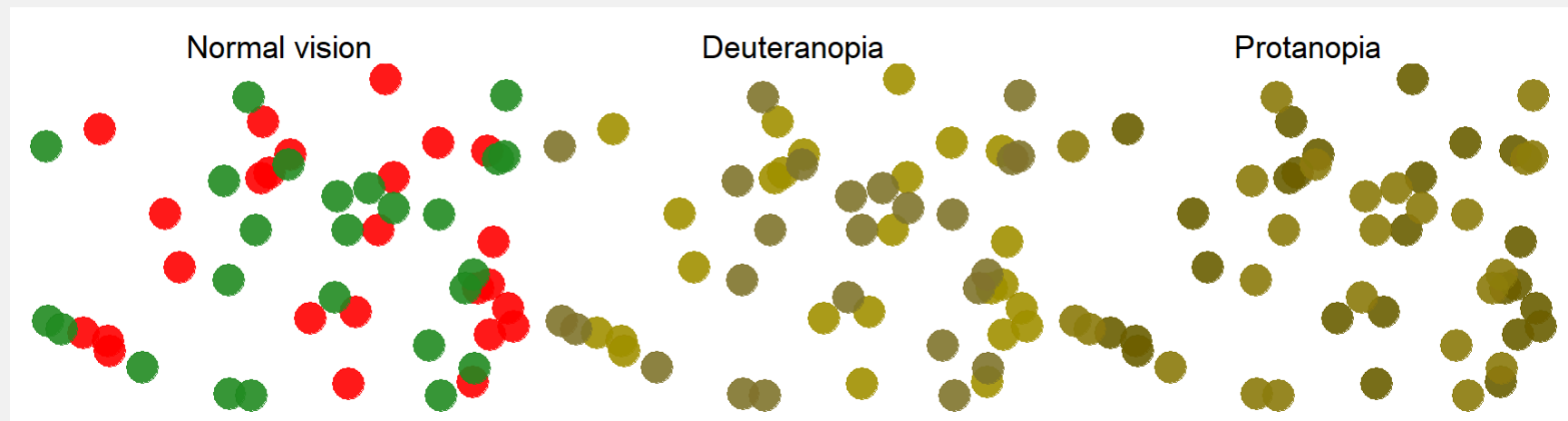


Blogpost on colours by Lisa Charlotte Muth (Datawrapperr)

Two things that matter

For scientific figures:

1. **Intuitive** colours that match what your reader expects
2. **Colourblind-safe** colours



Three kinds of palettes

Qualitative



Sequential



Diverging



Distinct categories



Ordered, low to high

Above / below a midpoint

Okabe-Ito (base R)

A colourblind-safe qualitative palette, built into base R.



- 8 distinct hues plus black and grey, separable under all common CVD types
- access with `palette.colors(palette = "Okabe-Ito")`

[See here for details](#)

viridis (built into ggplot2)

Perceptually uniform, colourblind- and greyscale-safe. Best for ordered or continuous data.

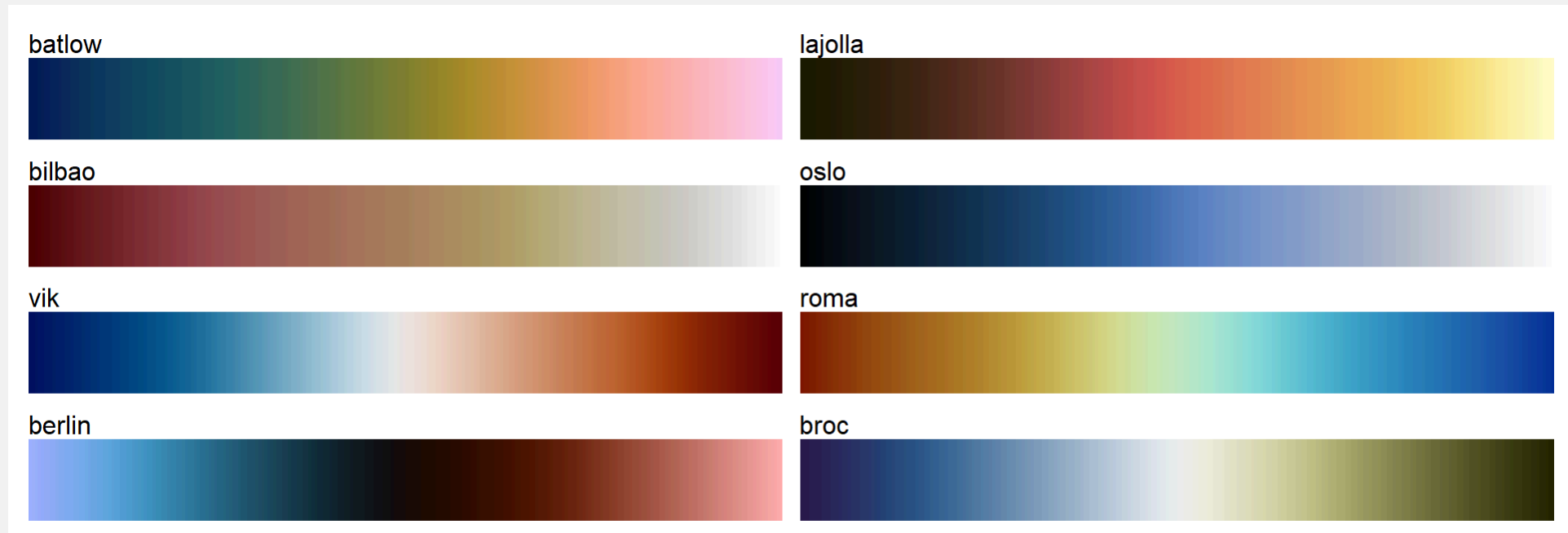


- access with `scale_colour_viridis_d()` / `_c()`

See here for details

scico R package

Perceptually uniform and colourblind-safe. Made for **scientific data** and designed to be fair, readable, and citable



- access with `scale_colour_scico_d()` / `_c()`

scico R package, the colour maps by Fabio Crameri

Live demo

Open `demo/02_color.R`.

Your turn

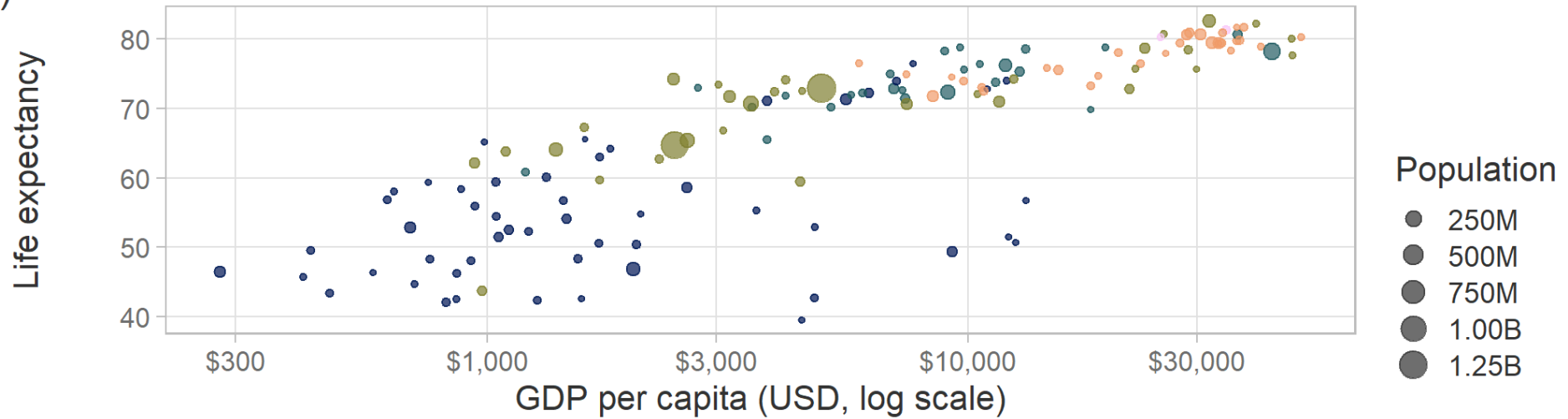
 Exercise (~5 min):

Open `exercises/02_color_exercise.R`

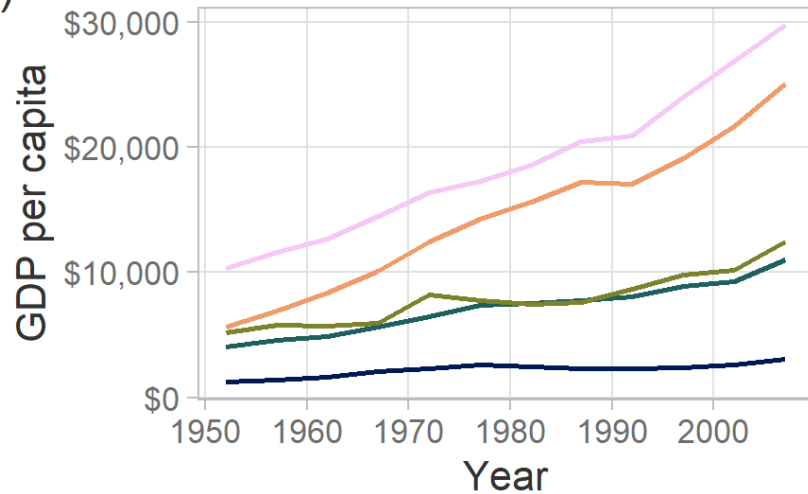
3. Multi-panel layouts

Common multi-panel layouts

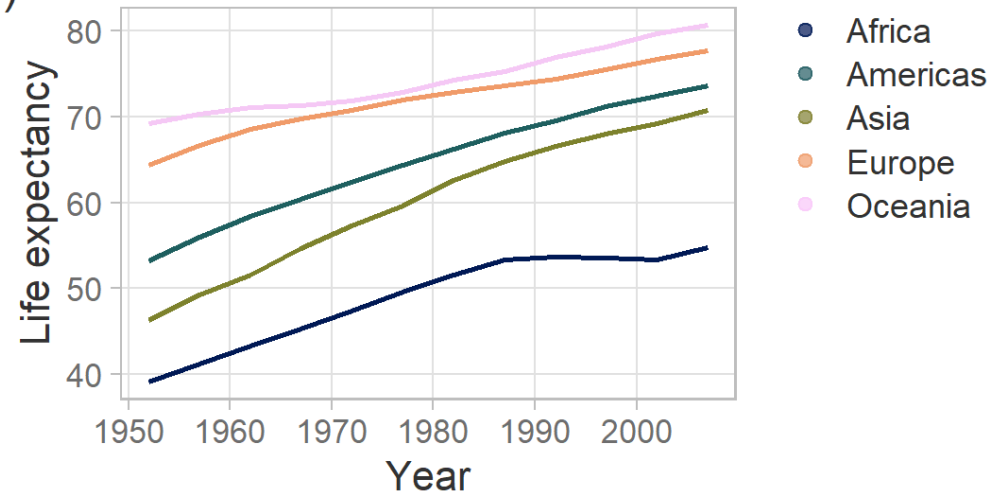
A)



B)



C)



The patchwork package

- Compose plots
- Collect shared legends and axes
- Tag panels
- Apply additional plot layers to every panel
- Place plots inside plots as insets

The [patchwork documentation](#) is excellent

Live demo

Open `demo/03_patchwork.R`.

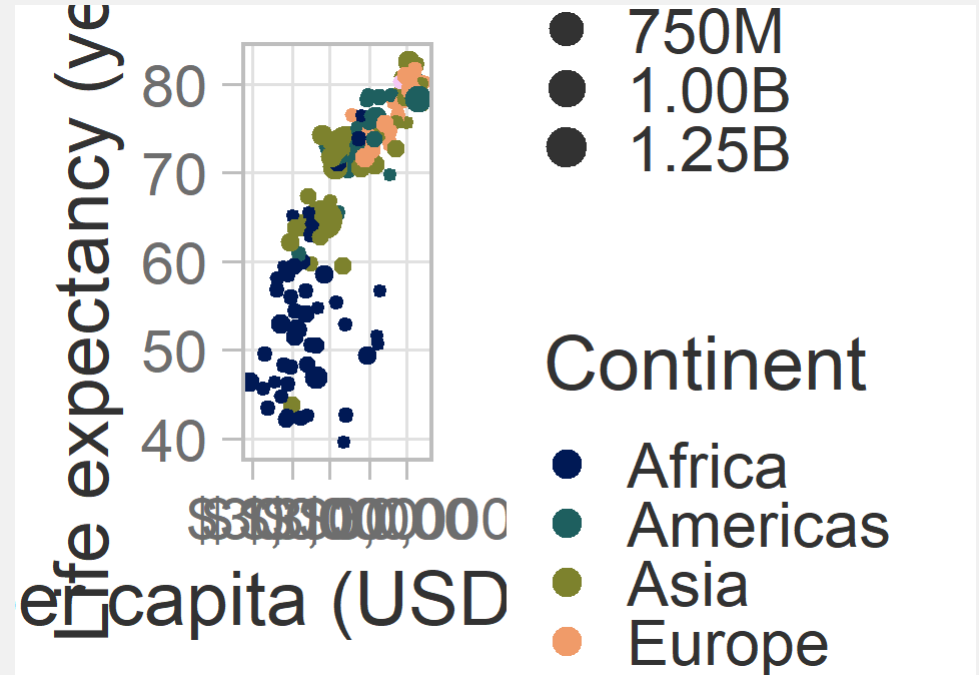
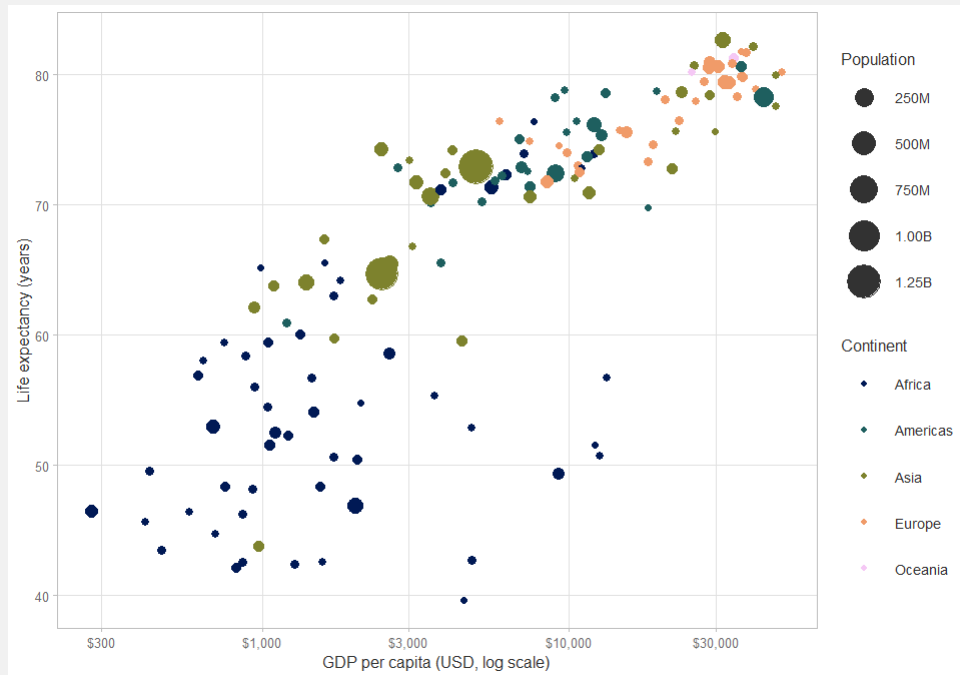
Your turn

 Exercise (~8 min):

Open `exercises/03_patchwork_exercise.R`

4. Exporting figures

Same plot, different canvas



Rule of thumb

Starting points for Canvas and font sizes:

Outlet	Canvas width	base_size
Paper, single column	~89 mm	8–10
Paper, double column	~120–180 mm	10–12
Slide, half	~150 mm	14–16
Slide, full	~250 mm	18–22
Poster	depends on layout	18–24

Live demo

Open `demo/04_export.R`.

Your turn

 Exercise (~7 min):

Open `exercises/04_export_exercise.R`

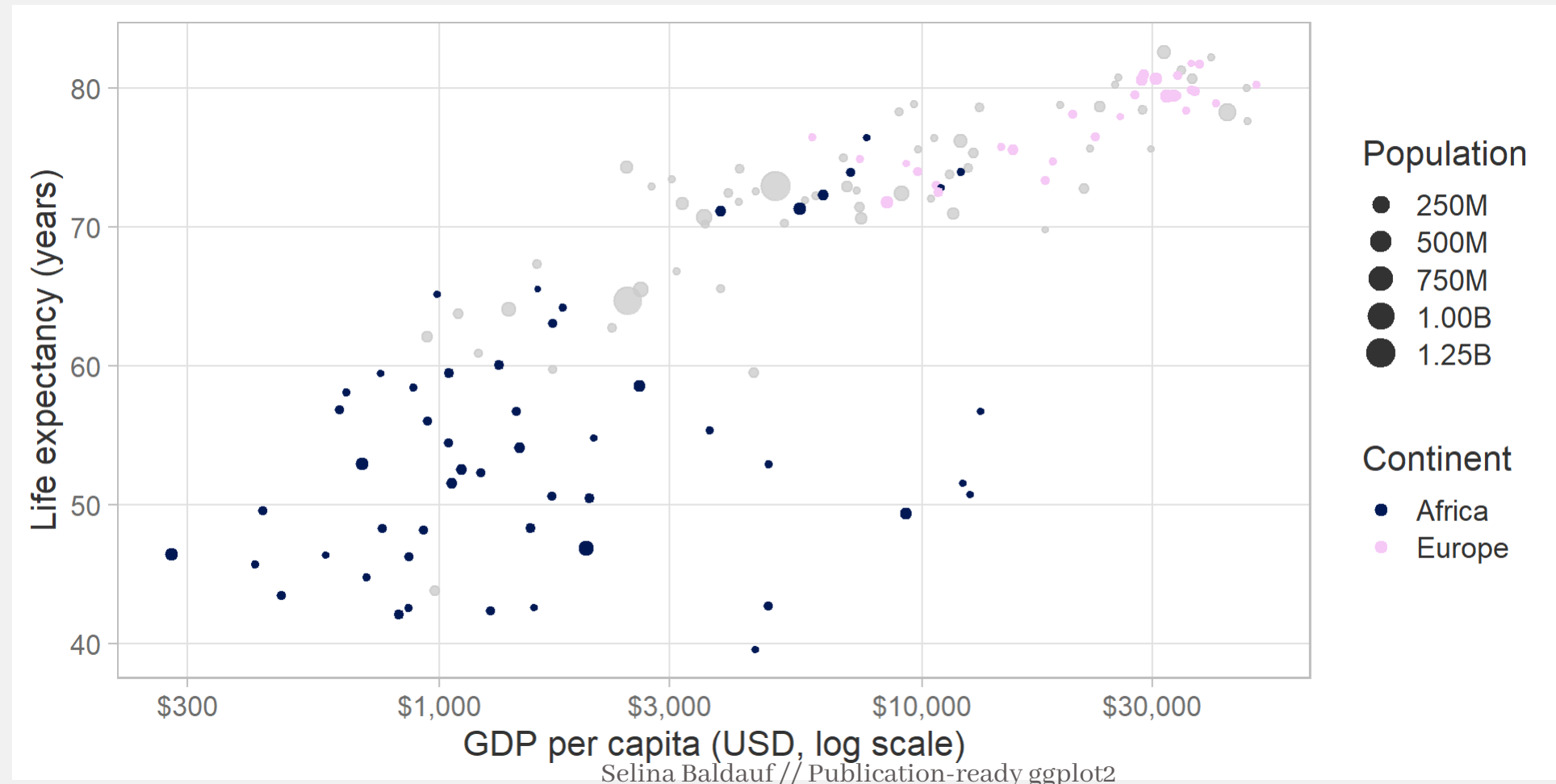
Other handy packages

The official [ggplot extension gallery](#) lists ~120 community extension packages to browse

gghighlight

`gghighlight` to fade data and highlight specific elements.

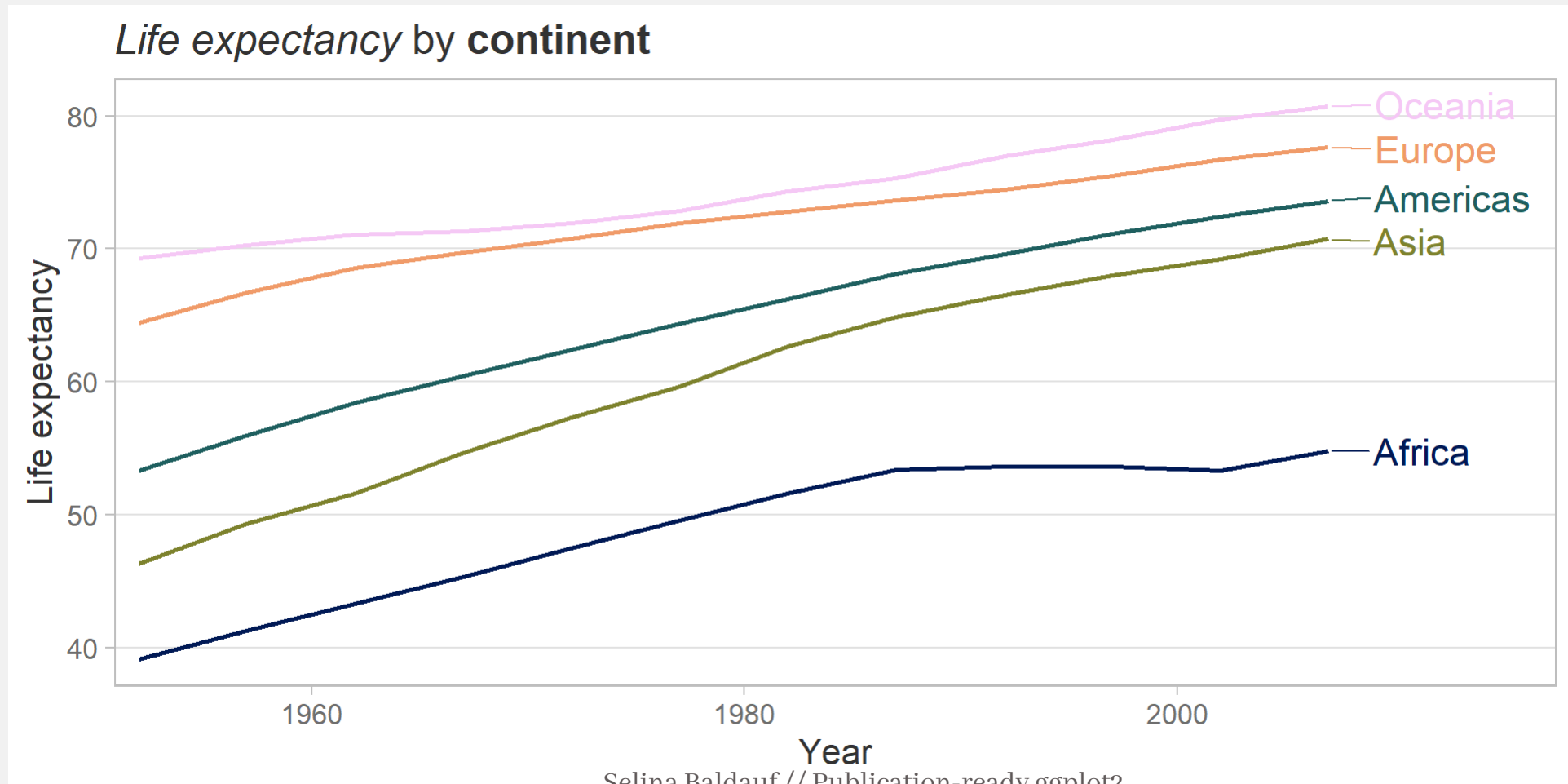
► Code



ggrepel + ggtext

ggrepel and ggtext for labels and markdown text

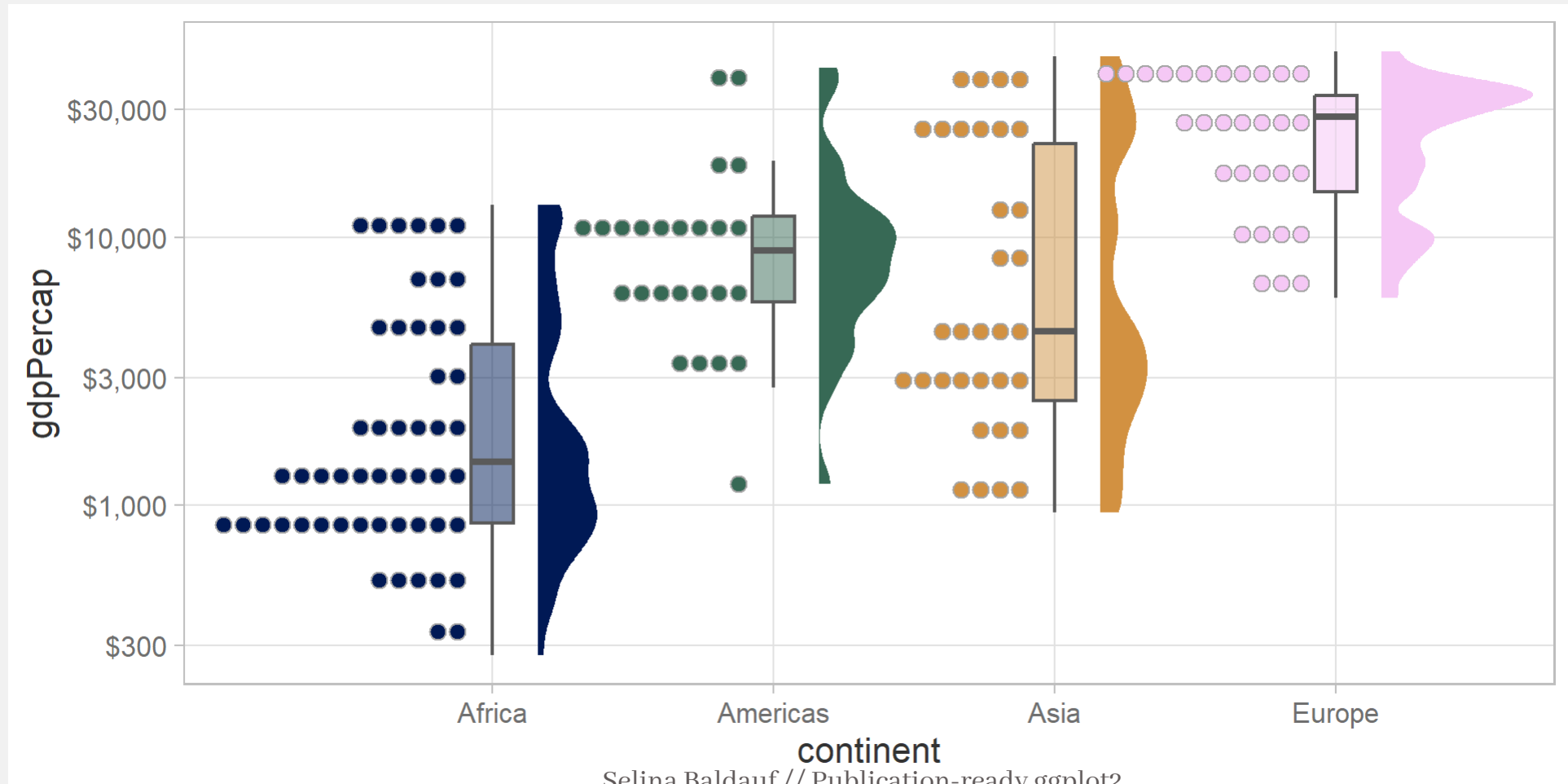
► Code



ggdist raincloud plots

ggdist to show distributions and uncertainty -> Barplot alternatives

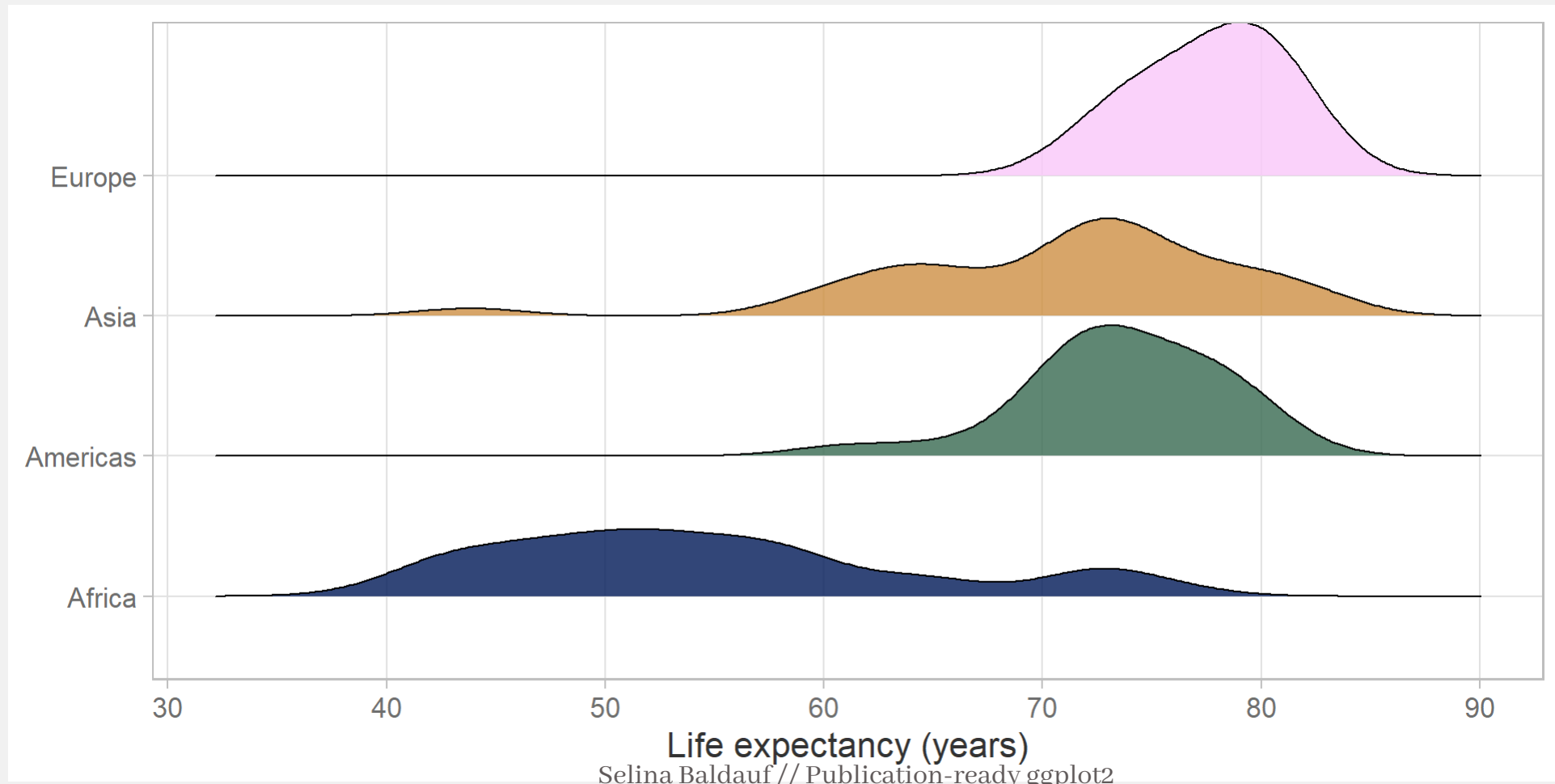
► Code



ggridges

`ggridges` for ridgeline distribution plots

► Code





Takeaways

What you can take back to your own plots:

1. **Custom themes:** write one `theme_*()` function, source it, `theme_set()` it everywhere it's needed
2. **Colour with intent:** pick intuitive, colourblind-safe palettes (Okabe-Ito, viridis, scico), and check with `cvdPlot()`
3. **Multi-panel layouts:** compose plots with `patchwork`, collect shared legends, add tag panels, ...
4. **Export:** pick the canvas size, preview with `ggview::canvas()`, tweak `base_size` and geoms, and save with `ragg::ragg_png` or `cairo_pdf`.

Next lecture/workshop

Topic tba

 18.06.2026  4-5 p.m.  Webex

 [Subscribe to the mailing list](#)

 For topic suggestions and/or feedback [send me an email](#)

The end :)

Questions?