Visualisation

An introduction to R, day 3

Prof. Dr. Claudius Gräbner-Radkowitsch
Europa-University Flensburg, Department of Pluralist Economics
www.claudius-graebner.com | @ClaudiusGraebner | claudius@claudius-graebner.com

Europa-Universität Flensburg

Europa-Universität Flensburg
International Institute of Management and Economic Education
Department of Pluralist Economics

Goals for today

I. Understand how plots are created layer-wise via the `ggplot2` package

II. Learn how to map variables in data frames to visual aspects of a plot

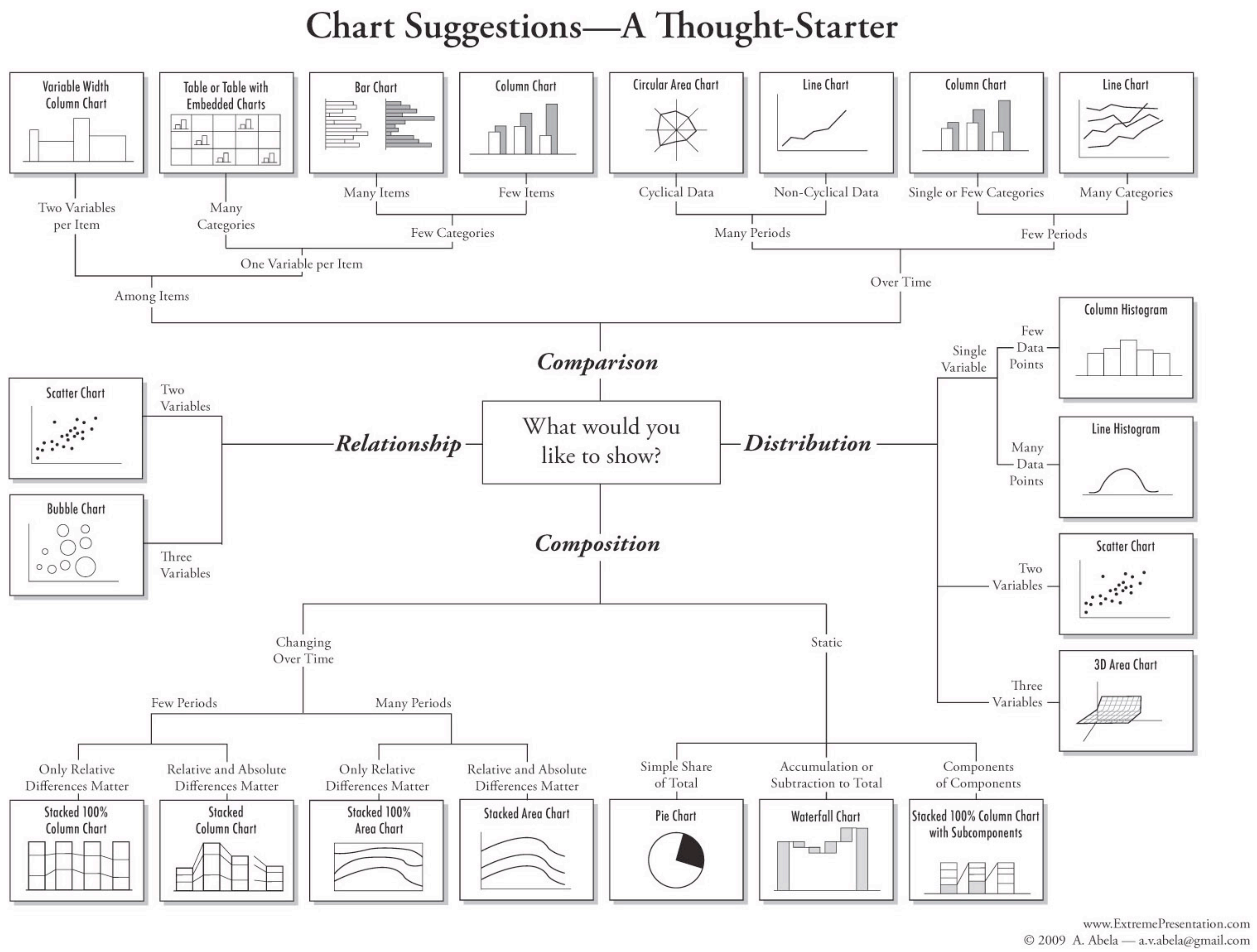
III. Figure out how you can re-use code across different visualisation tasks

Basics of visualization

About visualisations

- Visualisations can be used for many purposes
    - **Exploratory data analysis** → understand your data → prepare/refine models
    - **Communication** → inform others about your results
    - **Manipulation** → convince others or recognise others trying to convince you
- Here we will learn about how to create visualisations using the package `ggplot2`
- An easy-to-read, widely-used and powerful visualisation engine
- Many great extensions, e.g. for animated GIFs, control charts, and many more…

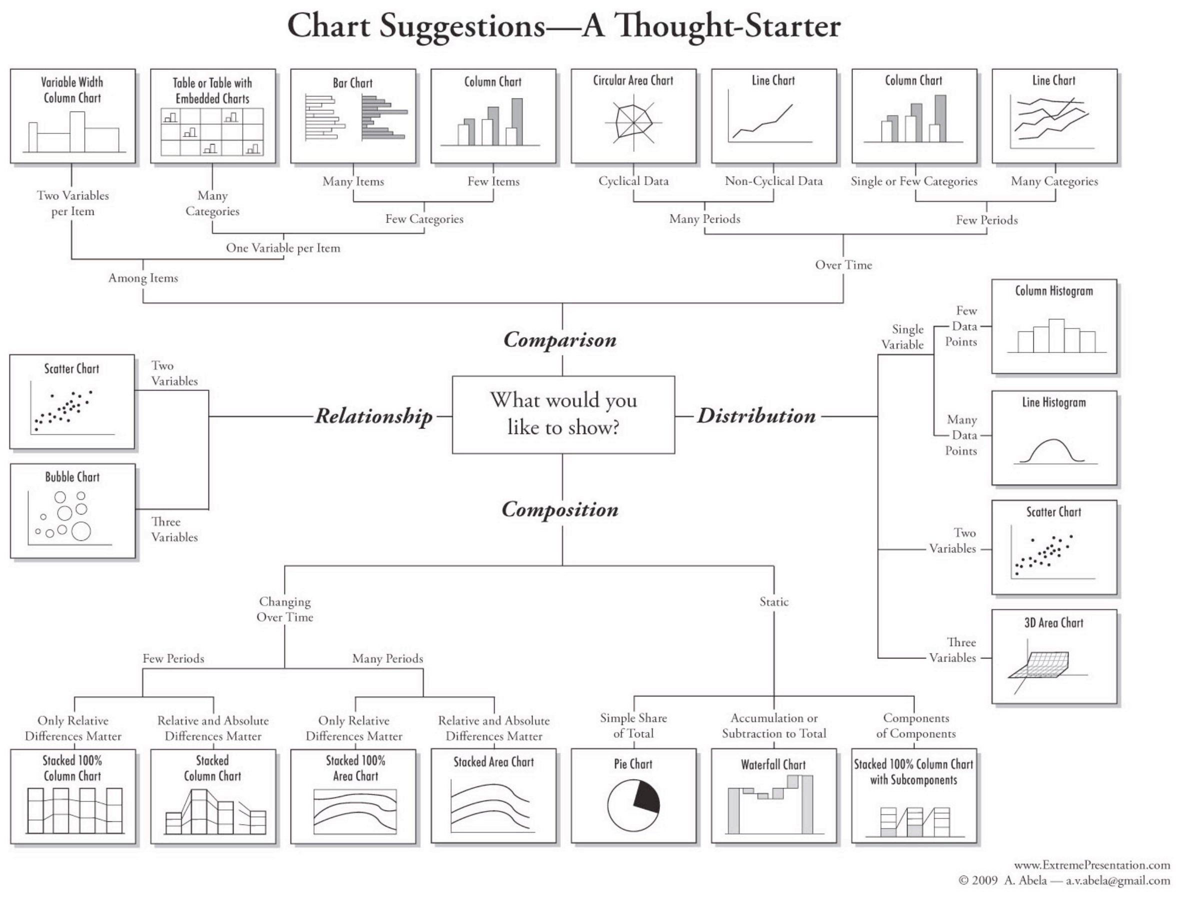
What kind of plot do you want?



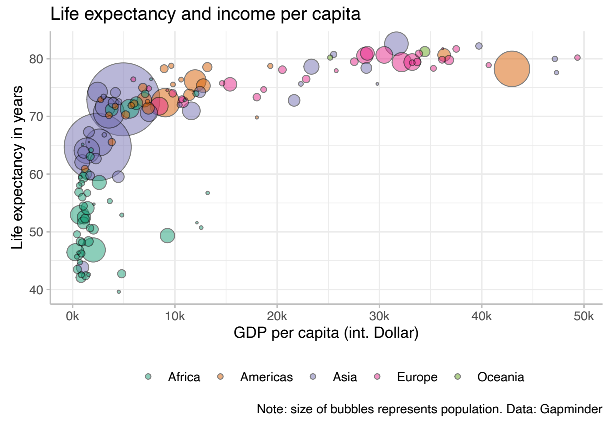
What kind of plot do you want?

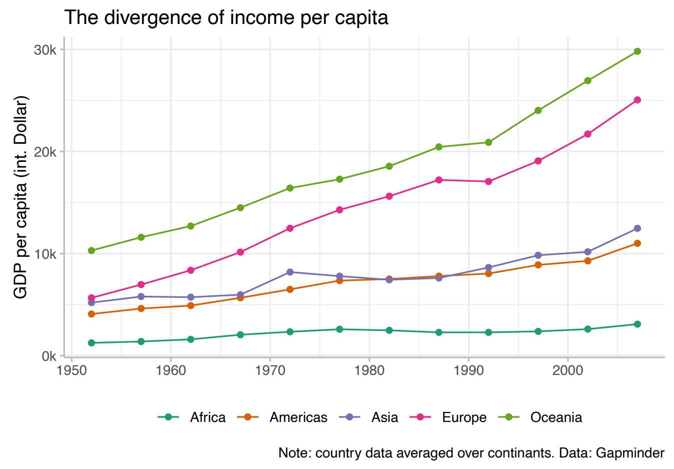
- Visualisation always involves prior thinking and theory

- The great thing about `ggplot2` is that the syntax is the same for all graphs



- During our lecture we focus on two examples:
  - A scatterplot/bubble chart
  - A line chart

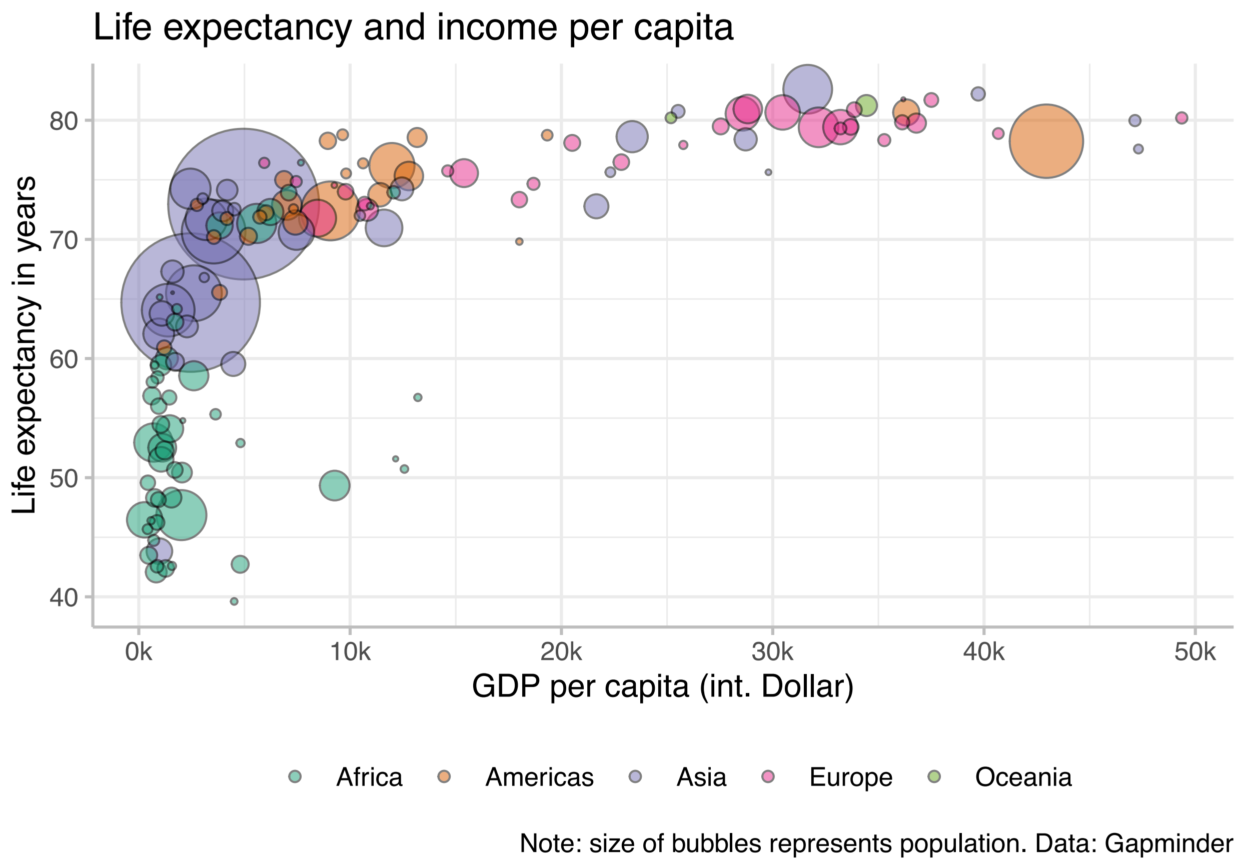


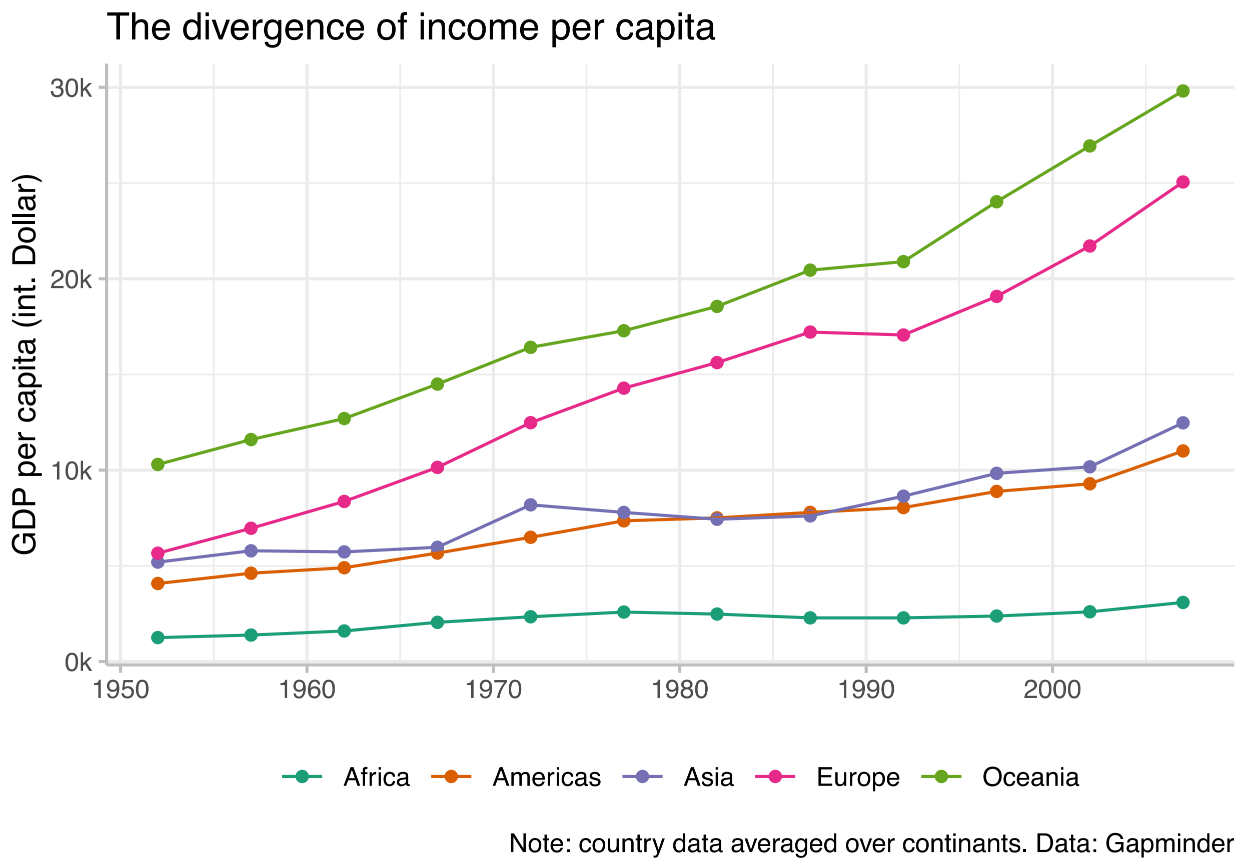


The practical workflow

Where we want to go:

- We illustrate the functioning of `ggplot2` by creating these two plots:





- We will see that the mechanics are very similar for different plots
  - Based on the readings you will be able to make even much more plots already now!

The general idea

- Every plot in `ggplot2` is generated in two major steps
  - You describe the plot in all its details via a list ← This is where all the work gets done 😅
  - You call the list and R renders the plot for you ← This is where errors become apparent 🤬
- To create the list-like description, `ggplot2` offers you a ton of helper functions
- You always start with an empty plot, then add layers above this empty plot, adjust details and that's it!
- Lets illustrate this using a subset of the gapminder data set only containing data for the year 2017
  - Readymade available to you via the DataScienceExercises package as `DataScienceExercises::gdplifexp2007`

Summary & outlook

Summary

- Visualisations serve many purposes, including the exploration of your data and the communication of your results
- We learned how to visualise data stored in data frames via `ggplot2`
- While there are many different plot variants, their syntax is very similar

```
ggplot() +
     <GEOM_FUNCTION>(
       data = <DATA>
       mapping =aes(<MAPPINGS>),
          stat = <STAT>,
          position = <POSITION>
          ) +
     <COORDINATE_FUNCTION> +
     <FACET_FUNCTION> +
     <THEME_ADJUSTMENTS>
```

The geometric forms used to represent the data (points, lines, shades,…)

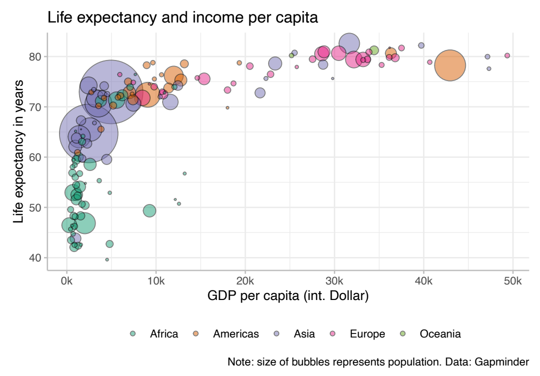
The data to be visualized

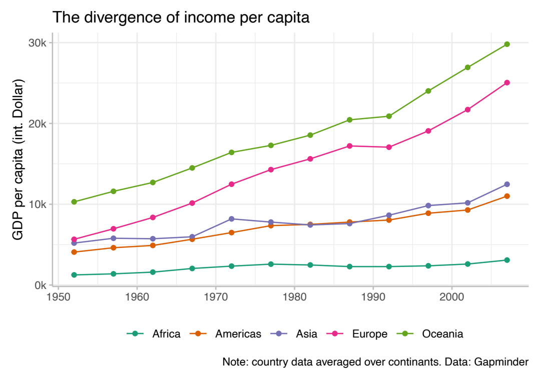
The mapping of the variables in data to the plot aesthetics (x/y-axis, size, form,…)

May be set as defaults within `ggplot()`, or separately for each geom

Adjustment to look, labels, etc.

From the bubble to the line chart





```r
gdp_data <- DataScienceExercises::gdplifexp2007

gdp_plot <- ggplot(
  data = gdp_data,
  mapping = aes(
    y = lifeExp,
    fill = continent,
    size = pop,
    x = gdpPercap)
) +
  geom_point(alpha=0.65, shape = 21) +
  scale_fill_brewer(palette = "Dark2") +
  scale_size_continuous(range = c(0.1, 21), guide = "none") +
  scale_x_continuous(
    labels = label_number(scale = 0.001, suffix = "k")
  ) +
  labs(
    x="GDP per capita",
    y = "Life expectancy in years",
    title = "Life expectancy and income per capita",
    caption = "Data: Gapminder.") +
  theme_bw() +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    panel.border = element_blank(),
    axis.line = element_line(colour = "grey"),
    axis.ticks = element_blank()

  )
```

Change data set

Adjust mappings

Use different shape

←Not required

Switch from x to y

Adjust labels

```r
gdp_data_time <- DataScienceExercises::aggGDPlifexp

gdp_line_plot <- ggplot(
  data = gdp_data_time,
  mapping = aes(
    y = gdpPercap,
    color = continent,
    x = year)
) +
  geom_point(alpha=0.65) +
  geom_line() +
  scale_color_brewer(palette = "Dark2") +
  scale_y_continuous(
    labels = scales::label_number(scale = 0.001, suffix = "k")
  ) +
  labs(
    y="GDP per capita",
    title = "Divergences in income",
    caption = "Data: Gapminder.") +
  theme_bw() +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    panel.border = element_blank(),
    axis.line = element_line(colour = "grey"),
    axis.ticks = element_blank(),
    axis.title.x = element_blank()
  )
gdp_line_plot
```

←New geom added

←Remove title of x axis

Summary

- Code for different plots differs mainly by the aesthetic mappings and the geoms used → allows you to re-use a lot of coding heuristics
  - We produced two nice plots: a bubble plot and a line graph
- The readings introduce you to other types of plots, which you can easily make more beautiful using the techniques you learned today
  - A great way to learn how to plot is to replicate examples from the internet, and adjust them to your own data:

from Data to Viz

The R Graph Gallery