Intro to Programming with R for Political Scientists

Session 4: Data Visualization

Markus Freitag
Geschwister Scholl Institute of Political Science, LMU

19.07.2021

Overview

1. Intro + R-Studio and (Git)Hub
2. Base R & Tidyverse Basics
3. Data Wrangling I
4. Data Wrangling II
5. **Data Viz**
6. Writing Functions

Workflow

- Navigate to `Session Scripts > Session 4` and open `Session_4_script.R`.
- You will see a pre-formatted Script with all the steps I do on the slides.
- Explore as you follow.
- If you have a second monitor, great! If not, split your screen.

Why is data viz important?

Why is data viz important?

- We have always tried to represent information in an (abstract) visual form...

The Babylonian Map of the World. Oldest known world map.

Why is data viz important?

Communication

- Useful for...
  - exploring data structure/cleaning data (e.g. spotting missing or outliers)
  - identifying trends, clusters, descriptive patterns
  - descriptive as well as for statistical and causal inference

BUT: Visual Inferences can also mis-guide in several ways (for one perspective on this, see e.g. **here**).

- More generally, graphical abstractions can aid in other aspects when working with data (e.g. graphical representations of database structures or DAGs as a tool to see conditions for causal identification more easily).

Why is data viz important?

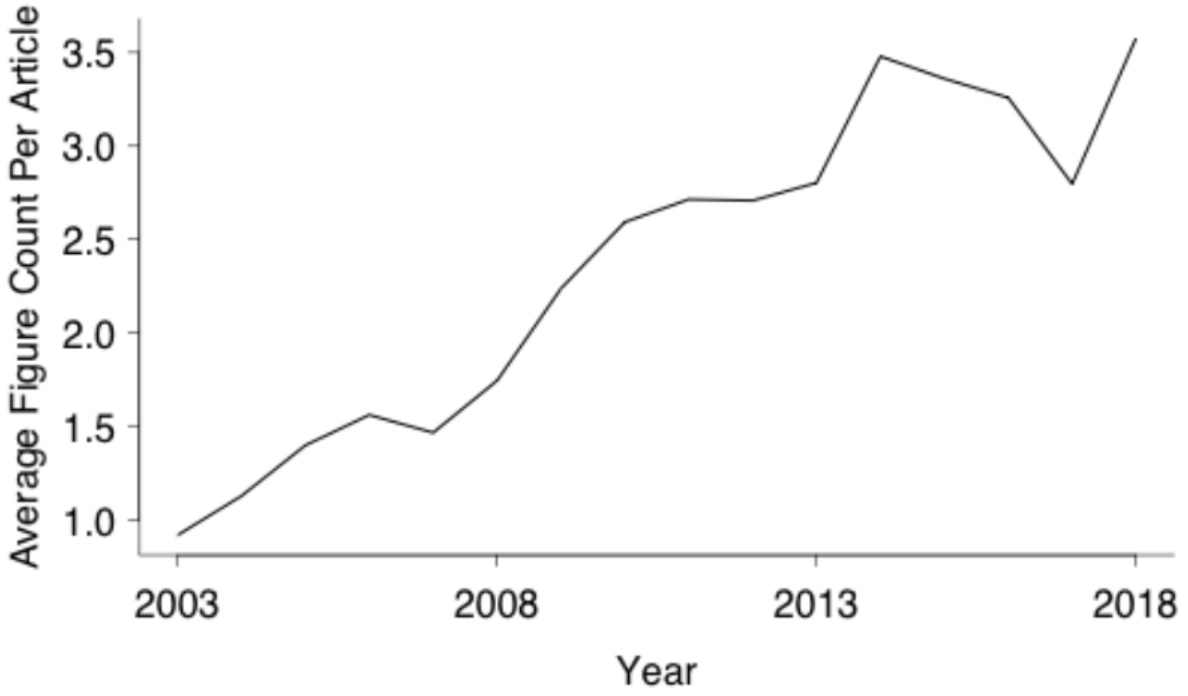
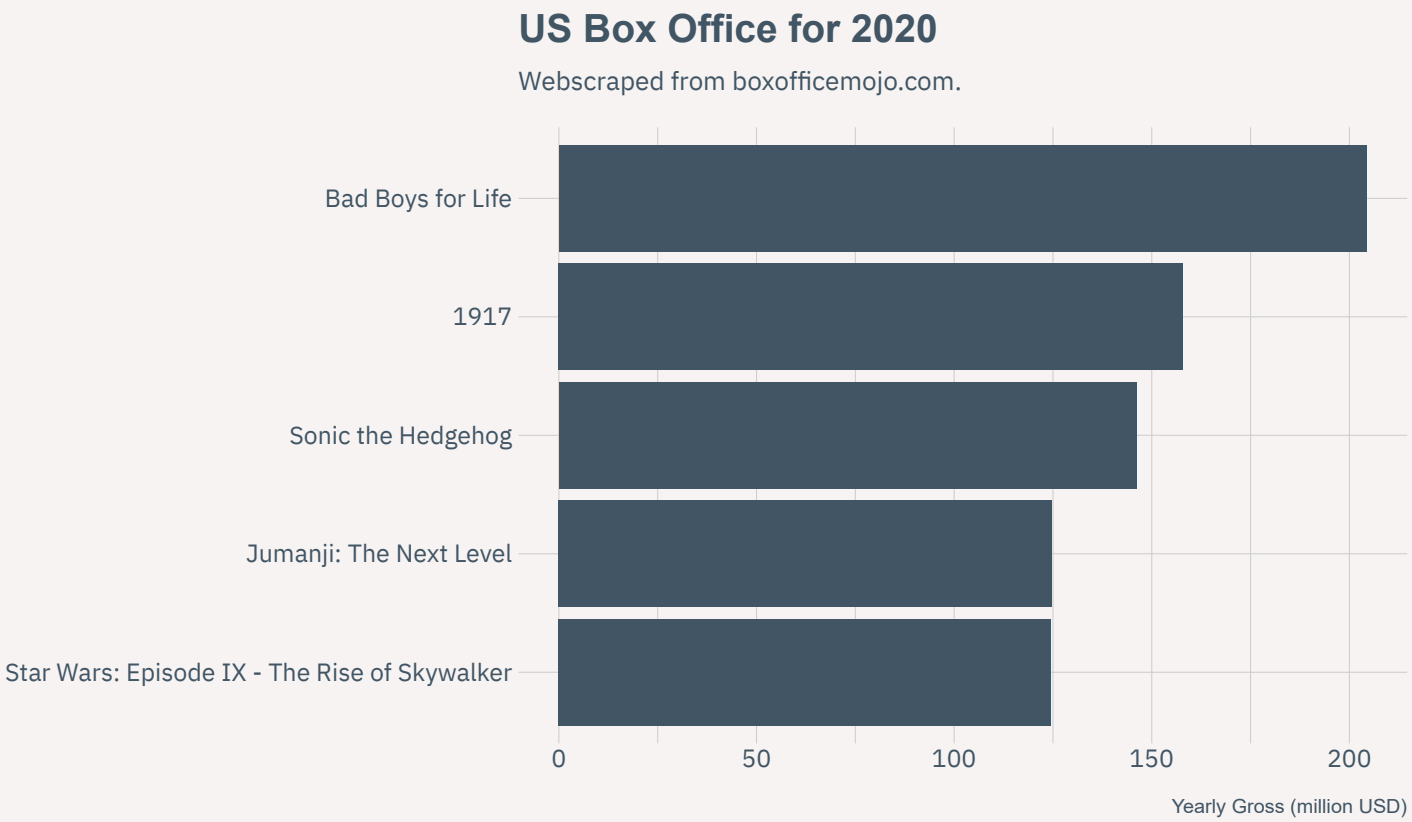


Figure 1: Average Number of Figures in all Articles Published in the AJPS, 2003-2018.

Source: **Traunmueller, 2018**.

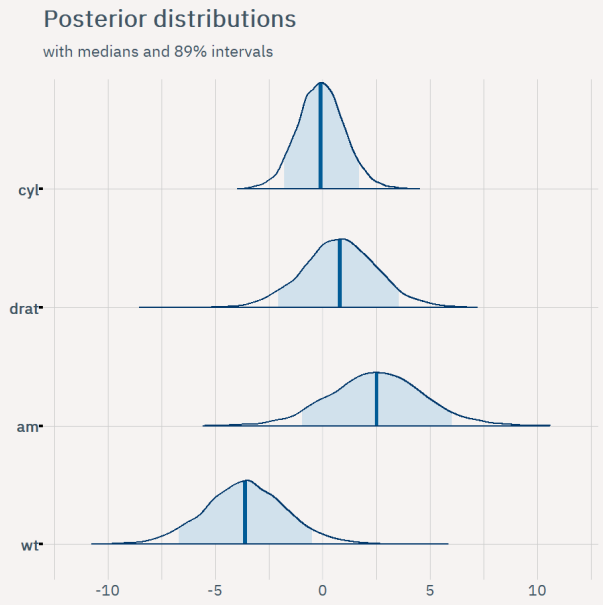
Useful for Description...

- We observe (see) some data and have various ways to transport summarising information to our eyes, e.g.:



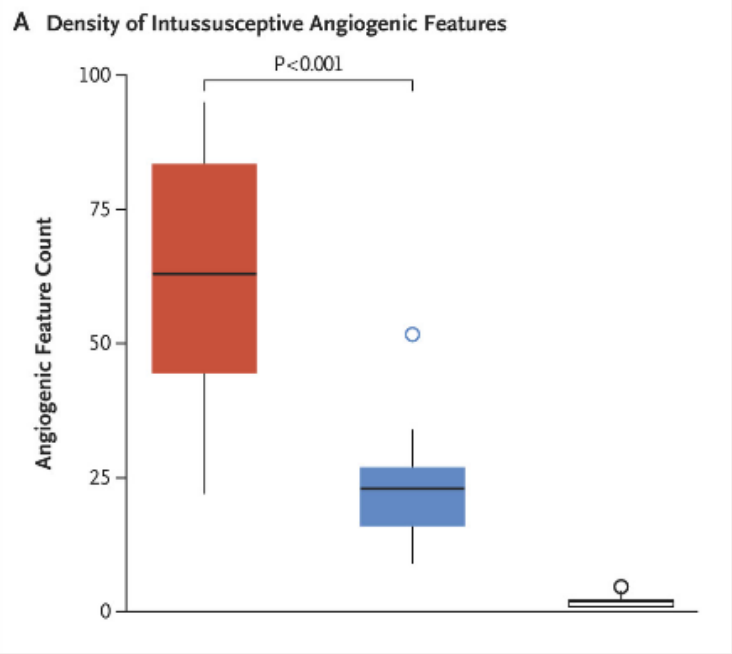
Useful for Statistical Inference...

- Inferring the unkown (some population quantity/parameter, e.g. $E[Y|X]$) from the known (the data at hand).
- Does what we see in the data accurately represent what we would see in the population we are interested in? How certain are we?
- Visualization is important to express our uncertainty about parameters...

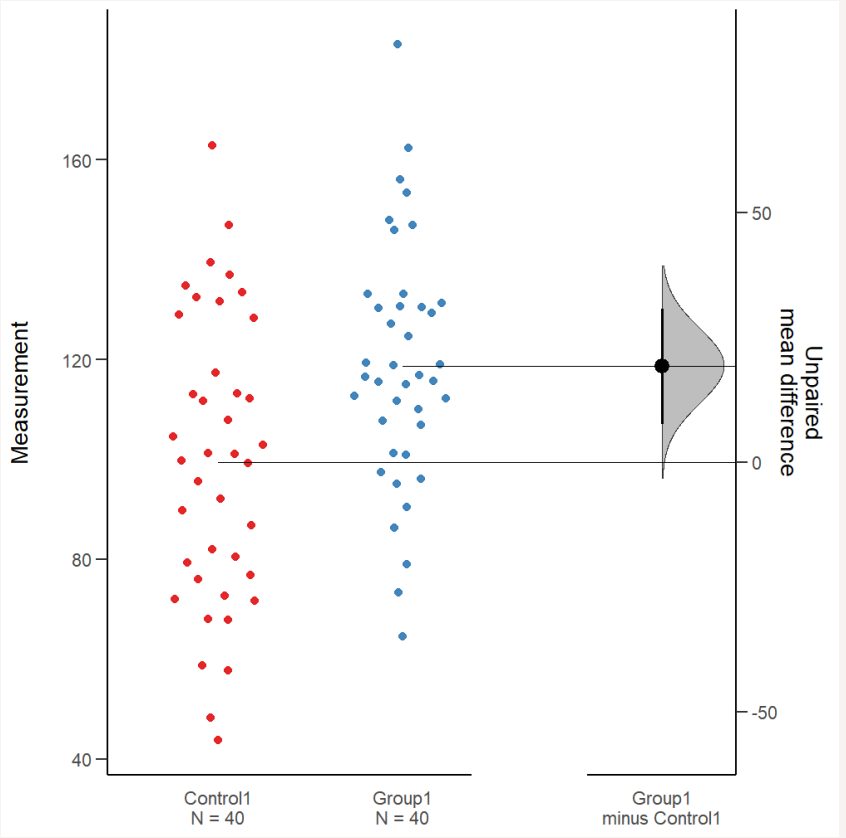


Useful for Statistical Inferences...

Not optimal (**NJEM 2020, Comparing Covid and Influenza Lungs**):

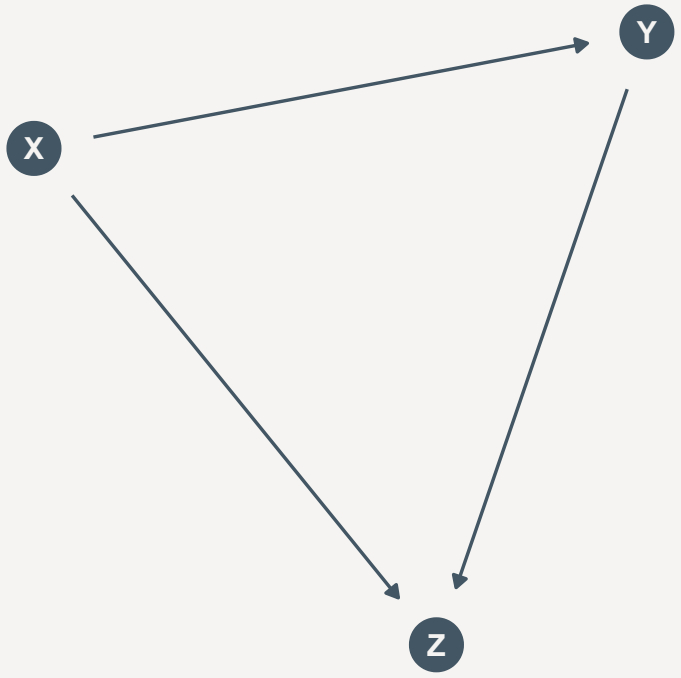


A little **better** (not the same data, just a sim. example):



Visualizations are also useful for Causal Inference...

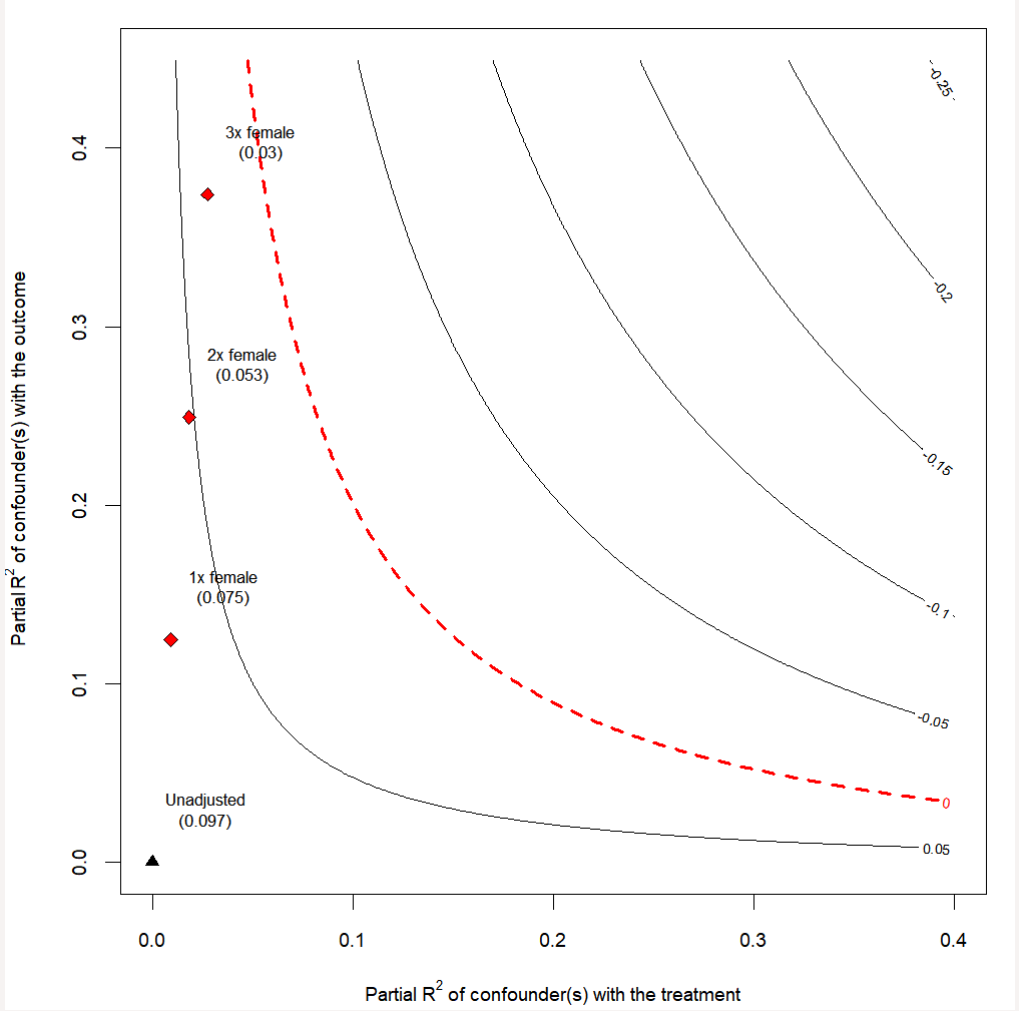
- We can represent our theoretical knowledge about (the absense) of causal relationships between with graphical models:



A DAG.

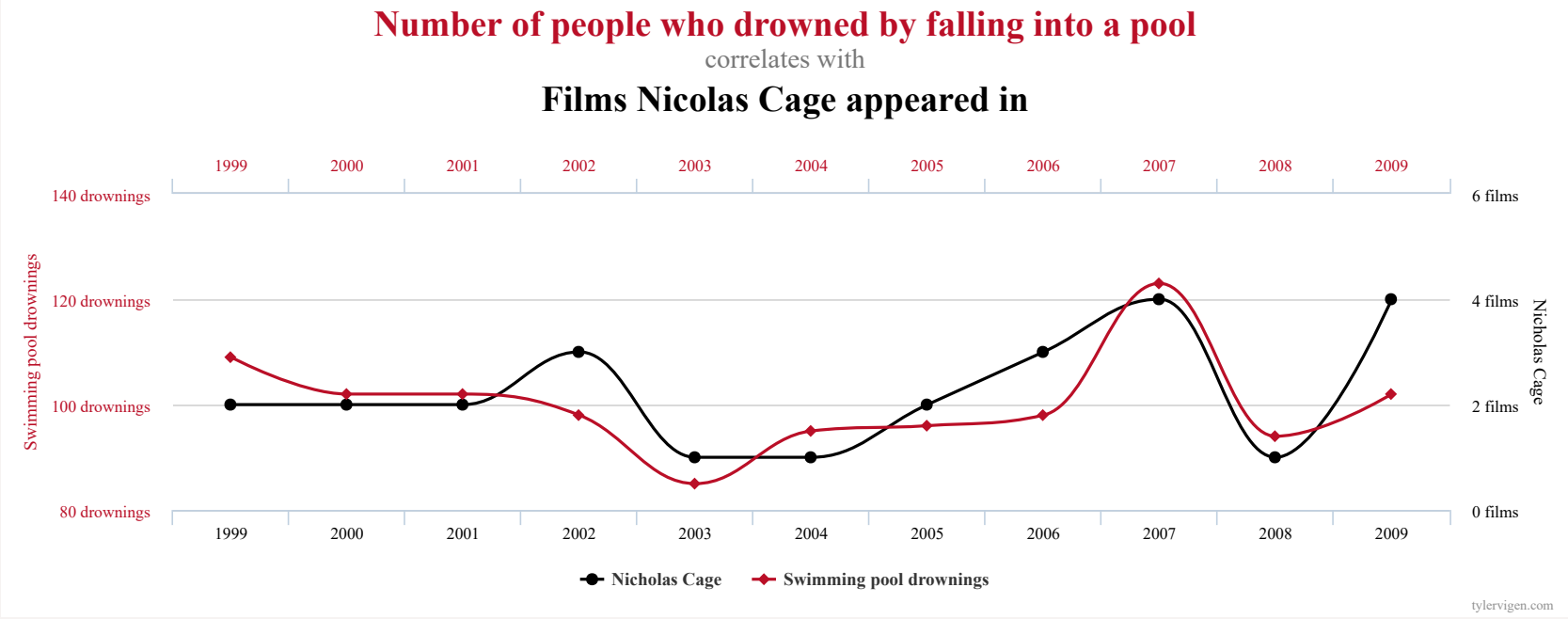
Visualizations are also useful for Causal Inference...

Or for sensitivity analysis:

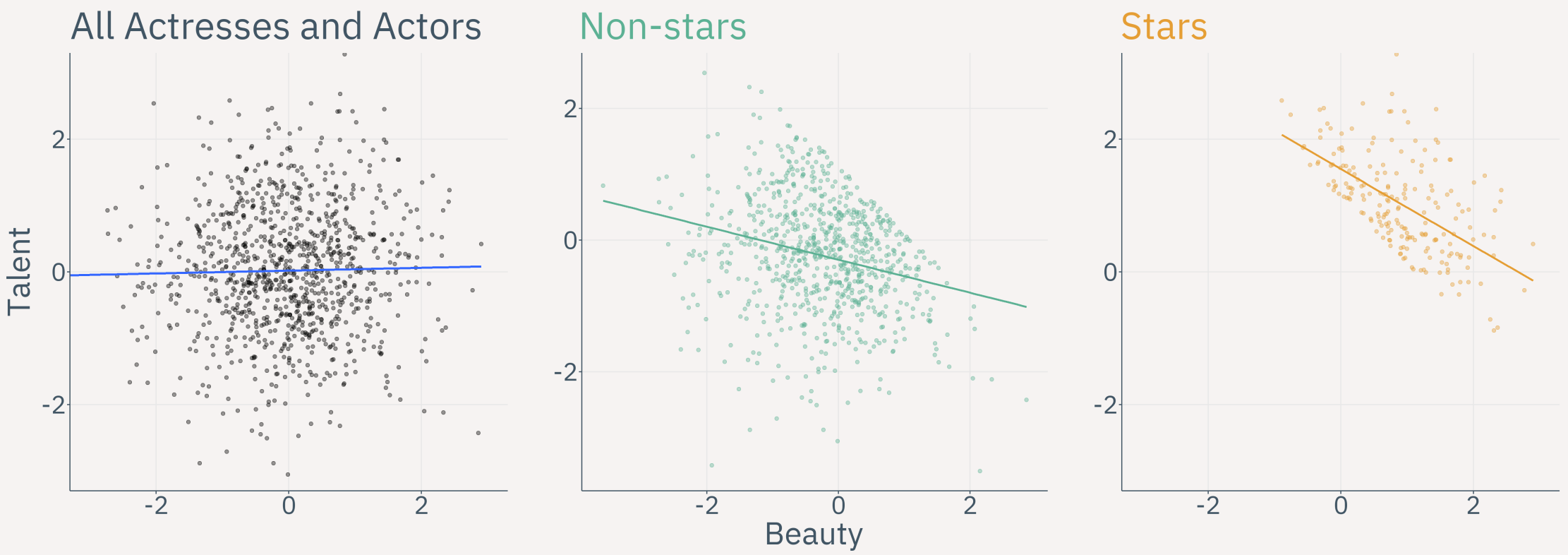


But beware...

- If you conduct exploratory data analysis and plot relationships, e.g. in a scatter plot:
- What you see in the data does not tell you anything about causation. You need theory and assumptions.
- $E[Y|X] \neq E[Y|do(X = x')]$ (correlation is not causation!)



But beware...



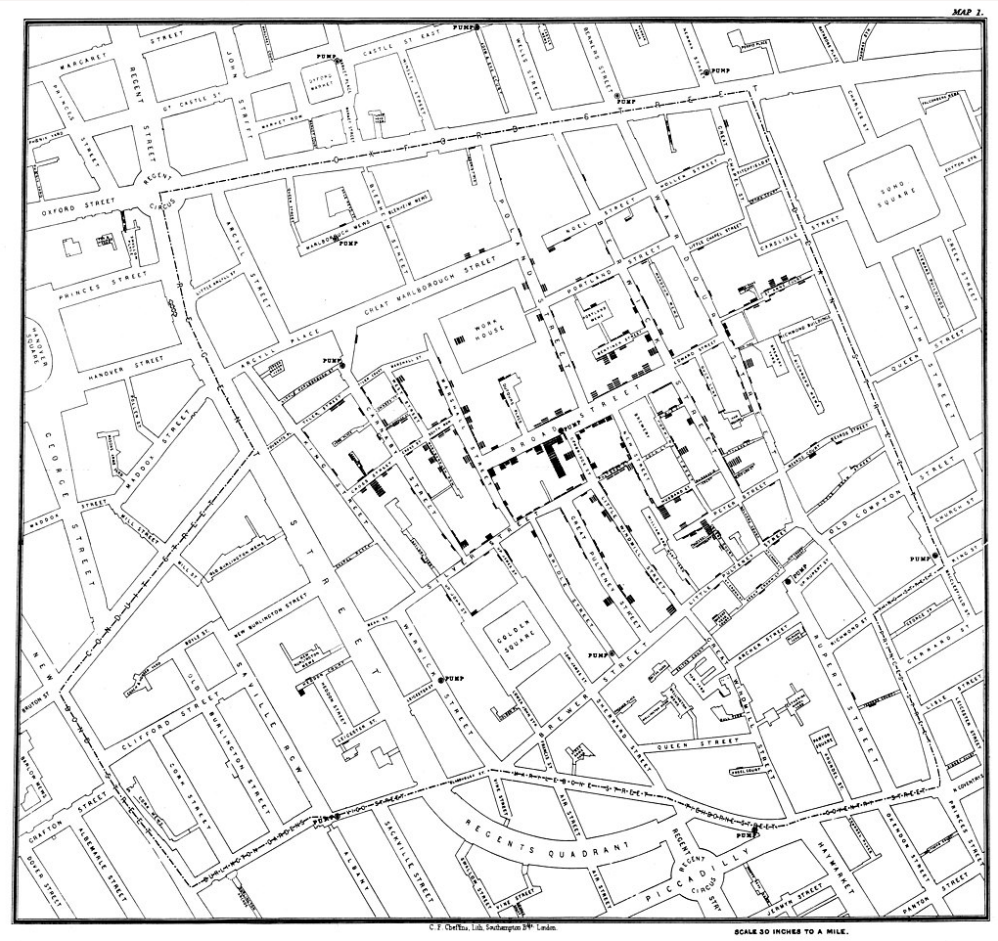
- Conditioning on the collider „Star“ (filtering our data and looking at subsamples of the „star“ variable) would lead us to wrongly infer a negative relationship between talent and beauty, even if there is, in fact, no relationship.

Still, theory + data + viz can put you on the right tracks...

John Snow and Cholera in London:

- In the 1840s and 50s, most scientists thought Cholera was transmitted via air.
- John Snow did not and forensically identified contaminated water as the cause of infection.
- During the pandemic, a water company moved their production up the Thames, resulting in a **natural experiment (diff-in-diff)**.
- He also visualized that Cholera deaths were concentrated around a contaminated pump (from another company).

Still, theory + data + viz can put you on the right tracks...



Some people still did not believe him. E.g. **Max Pettenkofer** in Munich...

Some Principles of Data Viz

- Raw data first. Erase everything unnecessary.

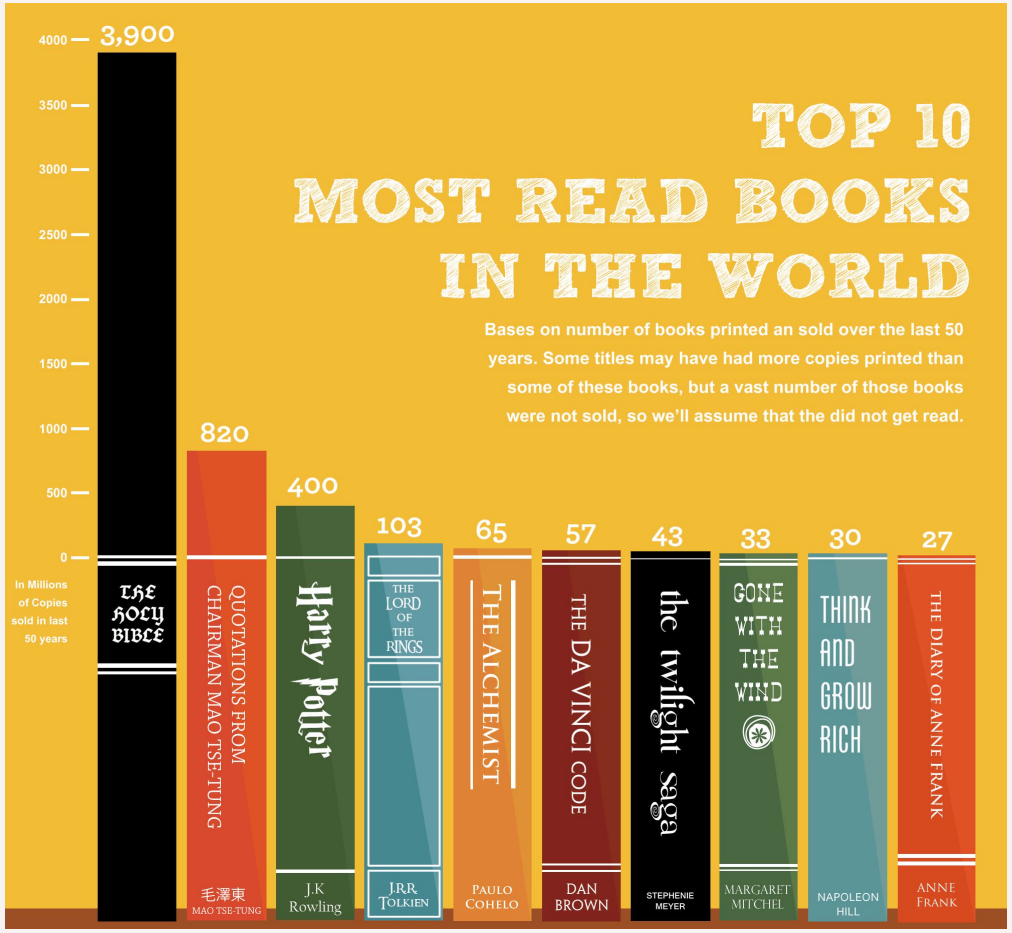
The principle of proportional ink (PPI):

> The representation of numbers, as physically measured on the surface of the graphic itself, should be directly proportional to the numerical quantities represented. (**Tufte, 1983**)

- Use color (incl. brightness, hue, saturation, etc.), but use it wise.
- Most tables should be figures (except for the case when there is little data)

Some Principles of Data Viz

Ex. of PPI violation: not showing 0 in bar plots or other weird things...

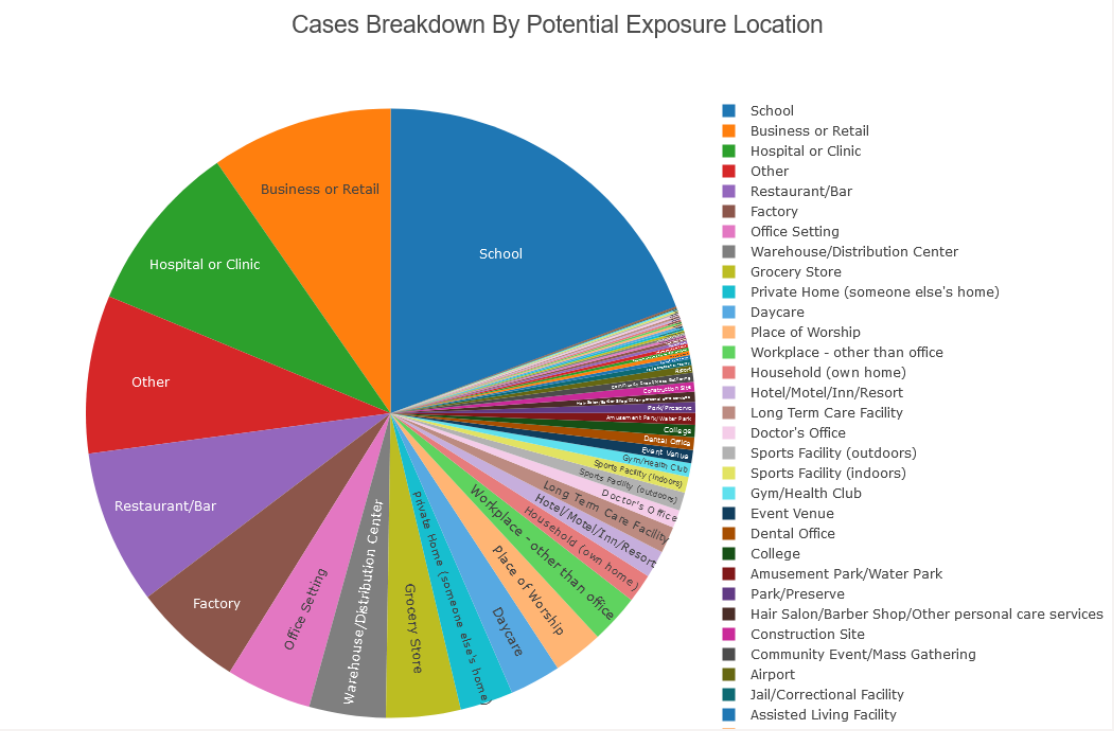


Gestalt Principles

- **Gestalt psychology** is a somewhat antiquated school of perceptual theory
- However, some of the "laws"/cognitive principles are pretty uncontroversial for **figure** or webdesign:
    - Proximity: Objects or shapes that are spatially close appear to be related.
    - Similarity: Humans group things that look alike (e.g. use of color in figures).
    - Change Blindness: We have a hard time comparing multiple visually similar plots or facets of plots.
    - Common region: Things in a common region are related, easy to grasp (e.g. confidence intervals)

Some No-Noes

Pie charts. Almost always bad (except for maybe dist. of seats in a parlianment with <= 4 parties).



Source: **https://www.dph.illinois.gov/covid19/location-exposure?regionID=0**.

Some No-Noes

3D when there is no third dimensions. Also, avoid 3D charts in general and break them down into multiple 2D charts.

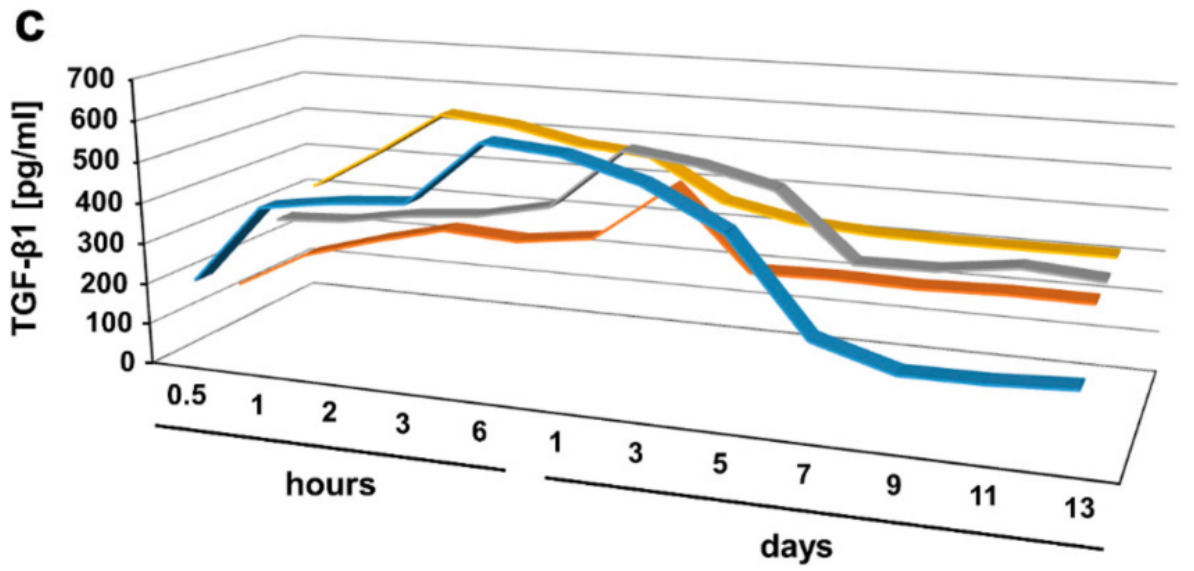
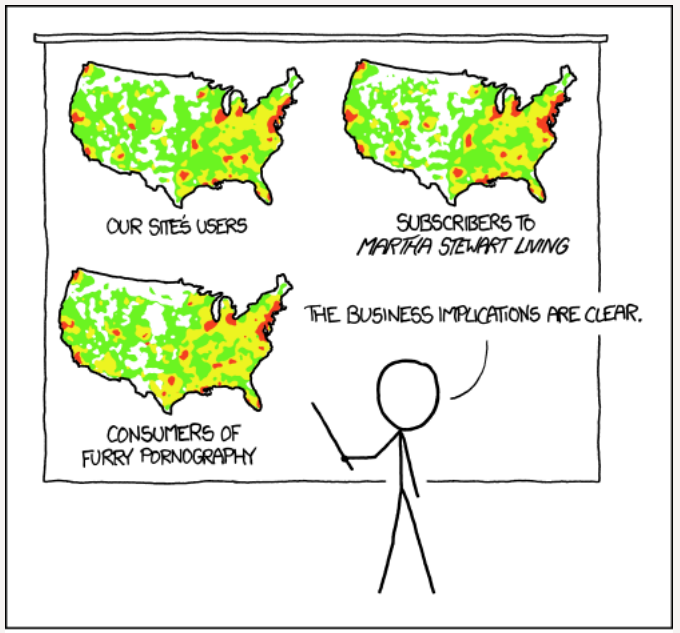


Figure 3. In vitro release of TGF-β1 from EMD-coated collagen matrices over time.

Some No-Noes



Source: **https://xkcd.com/1138/**.

From Data to the Eye

Grammar of Graphics:

- Data can come in many types: num. continious, num. discrete, categorical (ordered/unordered), time/date, text, spatial.
- In order to make a Figure, we need to map the data onto components of graphical elements: onto **aesthetics** that form geometric geometric objects.

> Some aesthetics are: coordinate positions, shapes, colors, line widths, etc.

We need program which cells of our data correspond to which aesthetics values. I.e. specify a **scale** that connects those two "layers".

Which Plot to Choose?

- Depends, ofc., on your data (make sure you know it well) and the question you want to answer.
- We will not go through types of charts systematically.
- Take a look at **this** book and this nice **descision tree** with code examples!

Some Tipps: **Cleveland dot plots** are an often useful alternative to bar charts. Line graphs need an ordered variable on the x axis. Scatter plots need cont. y and x and can be hard to read with big data. For a single numeric or numeric x categorical variables I like **raincloud plots**.

Workflow:

> Theory/understanding of the data → think about appropriate aesthetics and tyes of geometric objects (i.e. plot type) → sketch it by hand! → compute it → detail last!!

Data Viz with ggplot

ggplot's Grammar

- `ggplot` **builds on** the grammar of graphics.
- `ggplot(data = df, mapping = aes(x = x, y = y, color = group))` is the core function where we specify the aesthetic mapping.
- We then add layers with `+`.

A Plot from Scratch

We will use the UN voting data again, but this time the ideal point estimates of **Bailey et al. 2017**.

- Based on an IRT, they compute session-year ideal points that can also be used to compute dyadic similarity measures.
- We will replicate and extend one of their time series figures to the post 2010 period.

Preprocessing:

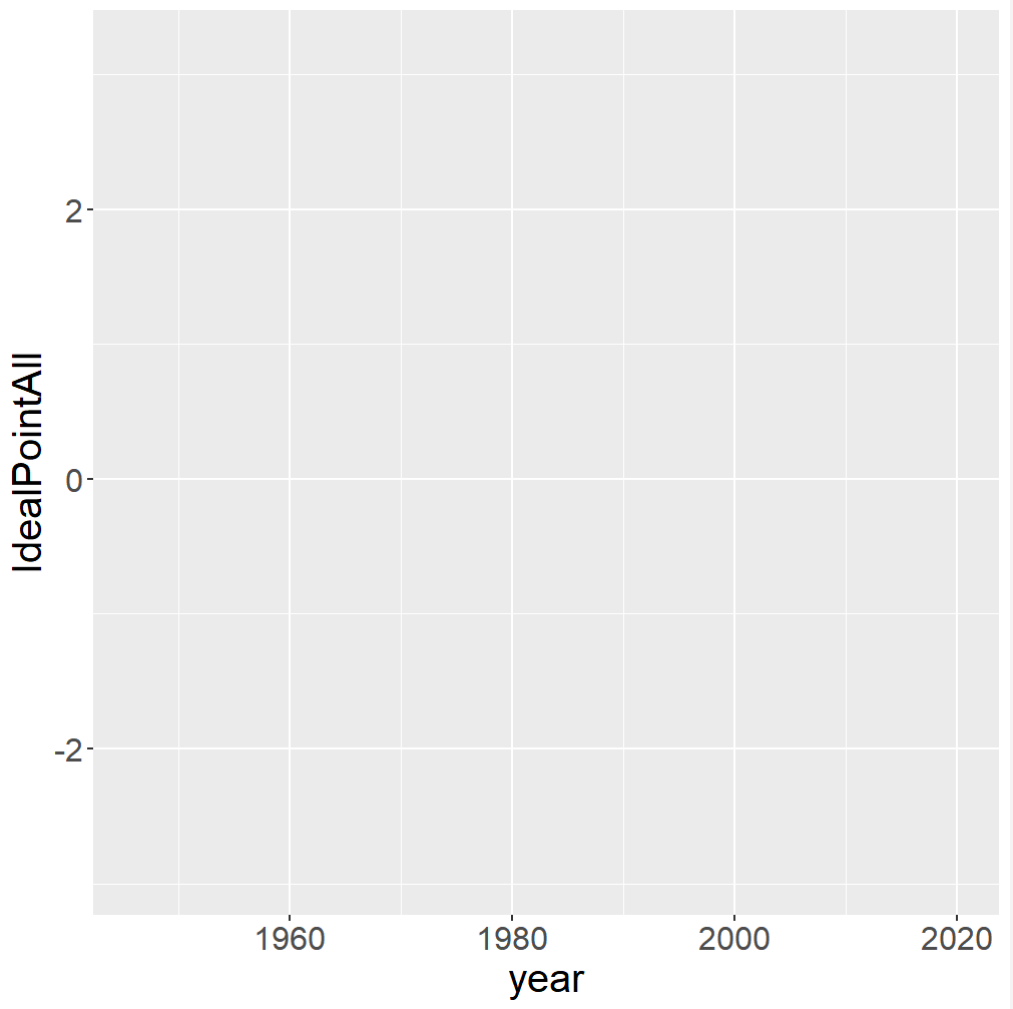
```
unvotes <- import("data/UNVotes2.parquet")
sessions <- unvotes %>%
  select(session, date) %>%
  group_by(session) %>%
  summarise(year = year(min(date)))
ideal <- import("data/ideal.csv")
ideal <- left_join(ideal, sessions, by = "session")
ideal <- filter(ideal, iso3c %in% c("USA", "GBR", "FRA", "CHN", "RUS", "DEU"))
```

A Plot from Scratch

Step 1: Specify aesthetic mapping

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c))
```

Other aesthetics are, e.g.: size, shape, fill, alpha.

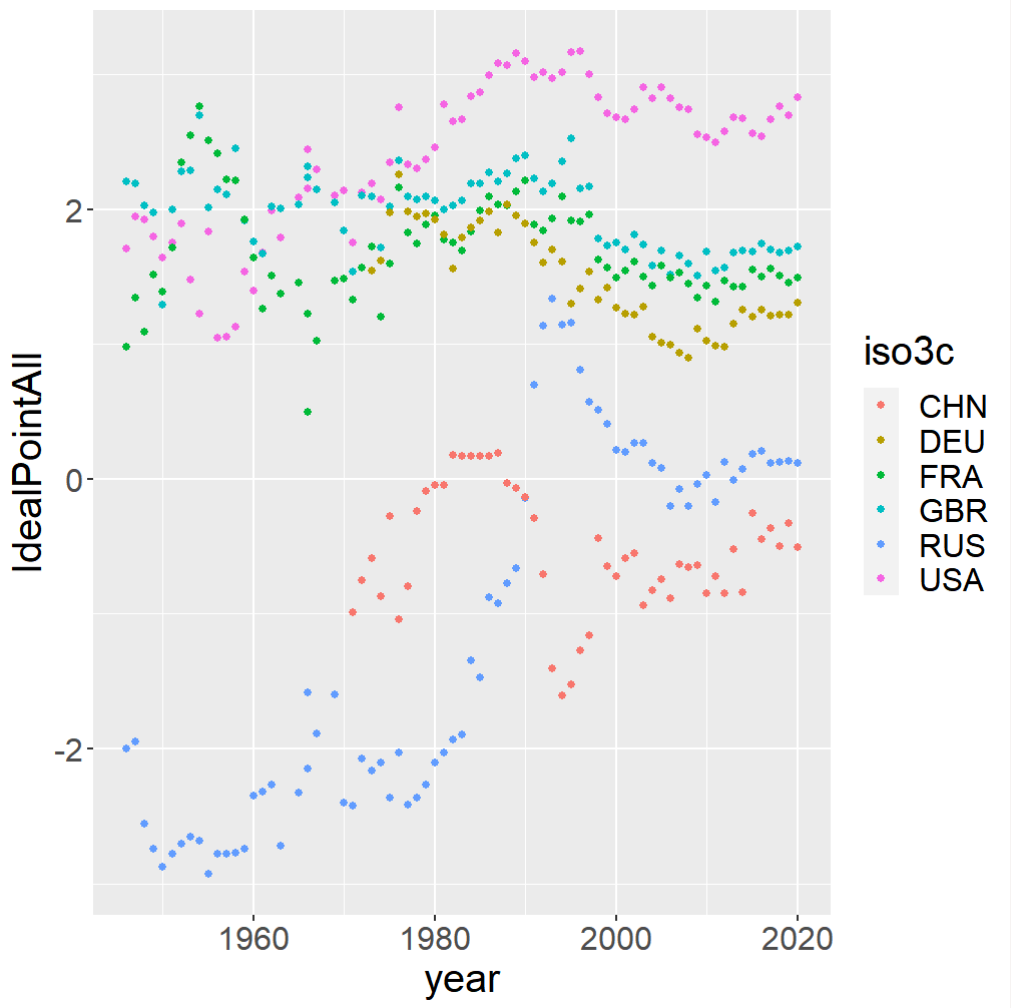


A Plot from Scratch

Step 2: Choose a geometry

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point()
```

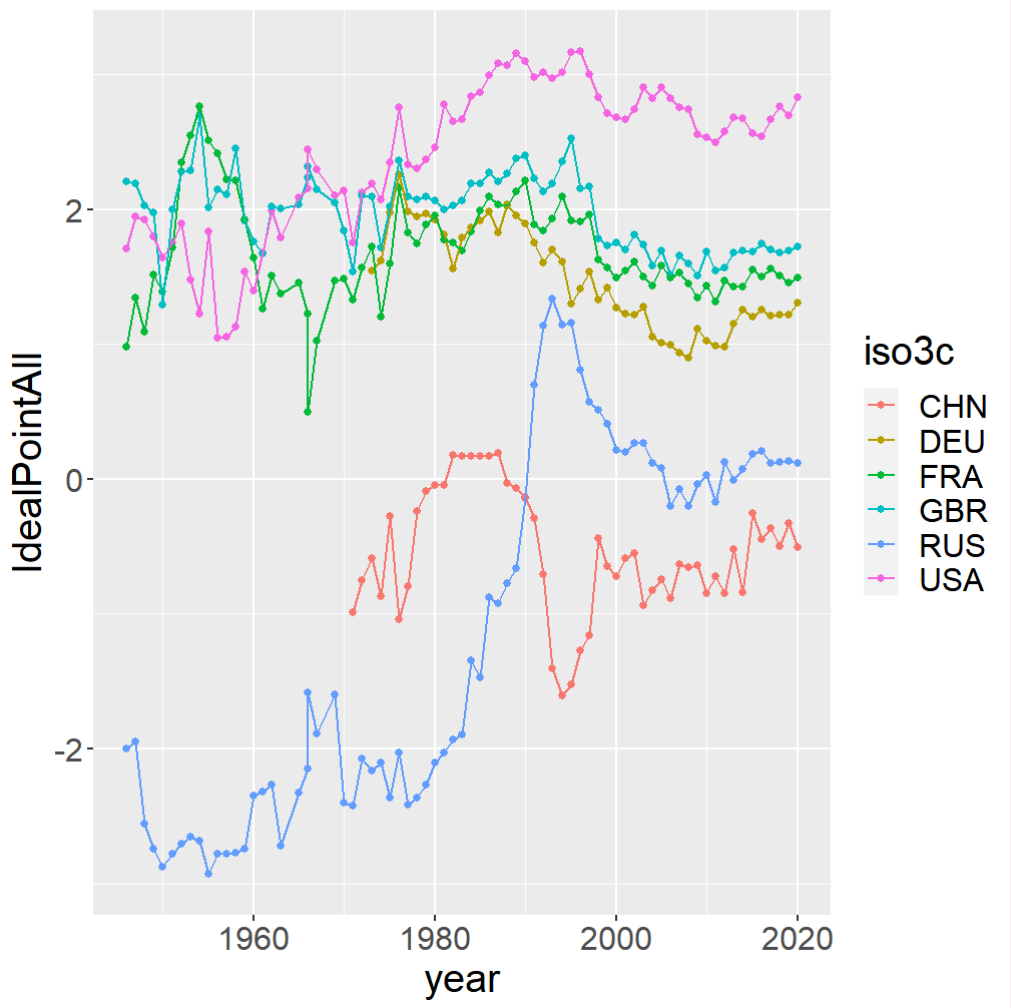
We prefer to show every single data point. Since the data is yearly this works ok with points. But we want lines too...



A Plot from Scratch

Step 3: Adding a line geom

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point() +
  geom_line()
```

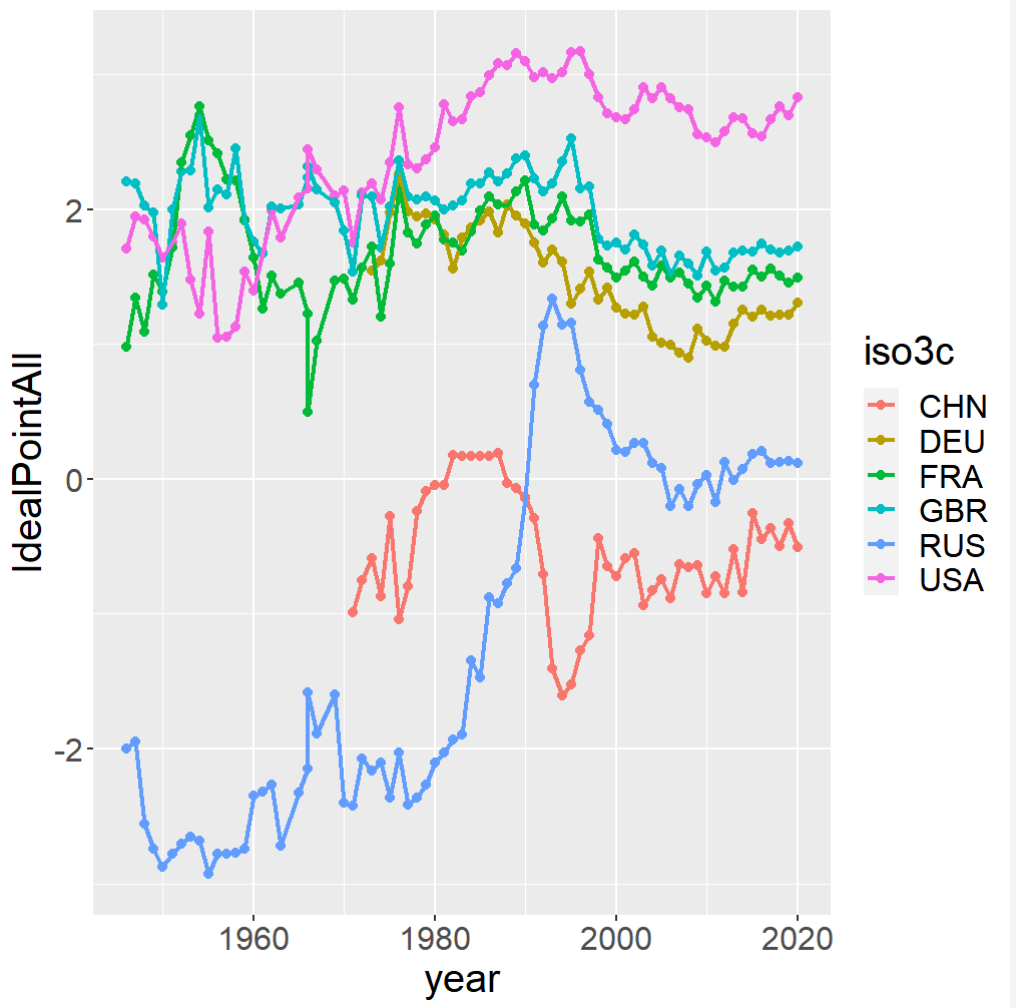


A Plot from Scratch

Step 4: Resize lines and points

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1)
```

If we dont want the size to map to our data, we can omit `aes()` and set to some value.



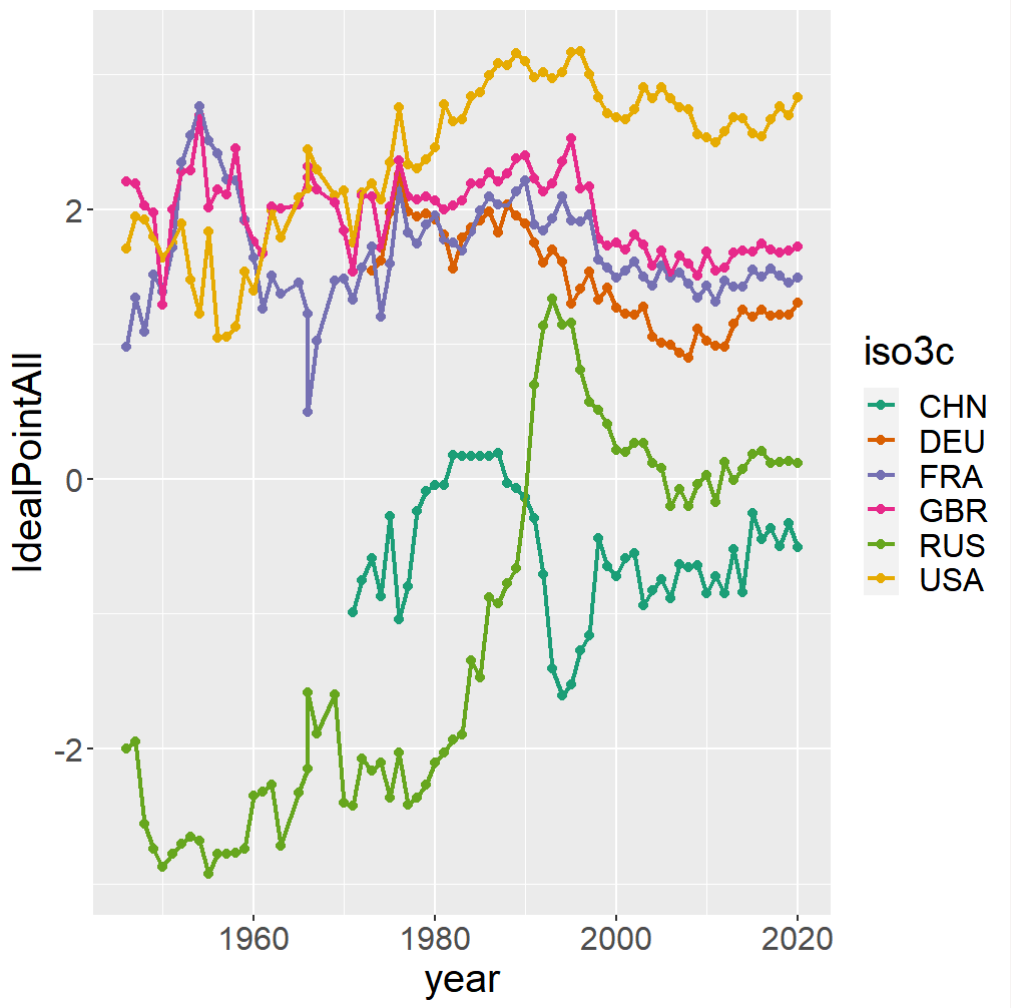
A Plot from Scratch

Step 5: Change color scale

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2")
```

We can add a decent color-blind friendly scale from **ColorBrewer**. We need a discrete color scale.

Tipp: MORE COLORS!?! Take a look **here**.

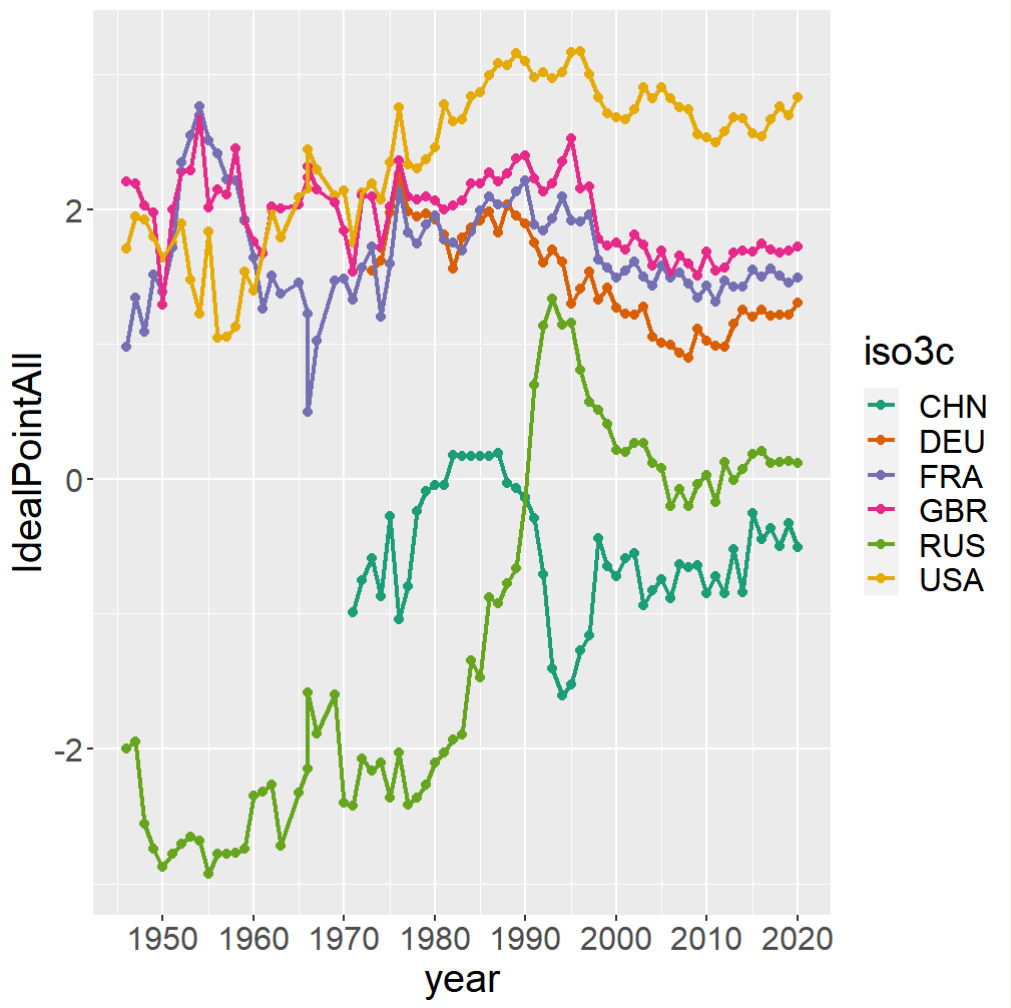


A Plot from Scratch

Step 6: x scale

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 202
```

As our date variable is simply of type integer, we use `scale_*_continuous` instead of `scale_*_date`.

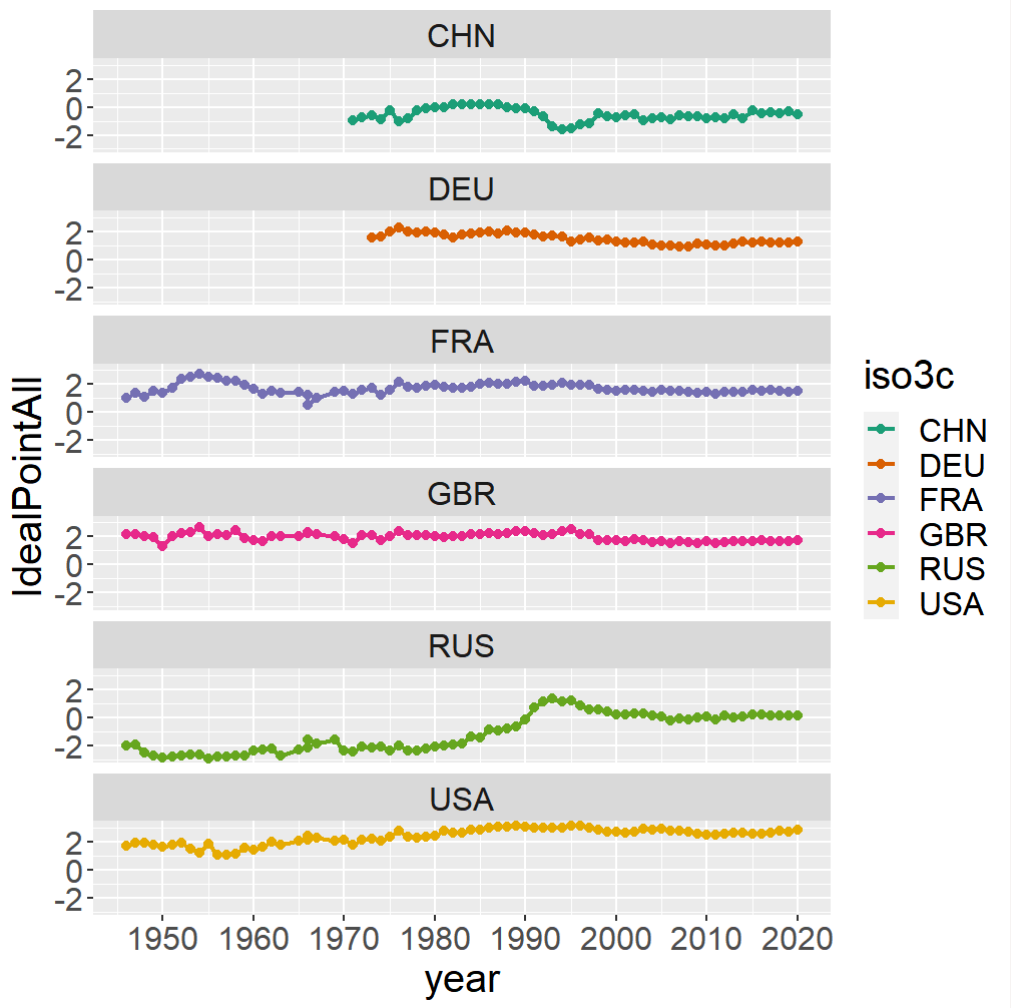


A Plot from Scratch

Step 7: (optional) Facets

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 202
  facet_wrap(vars(iso3c), ncol = 1)
```

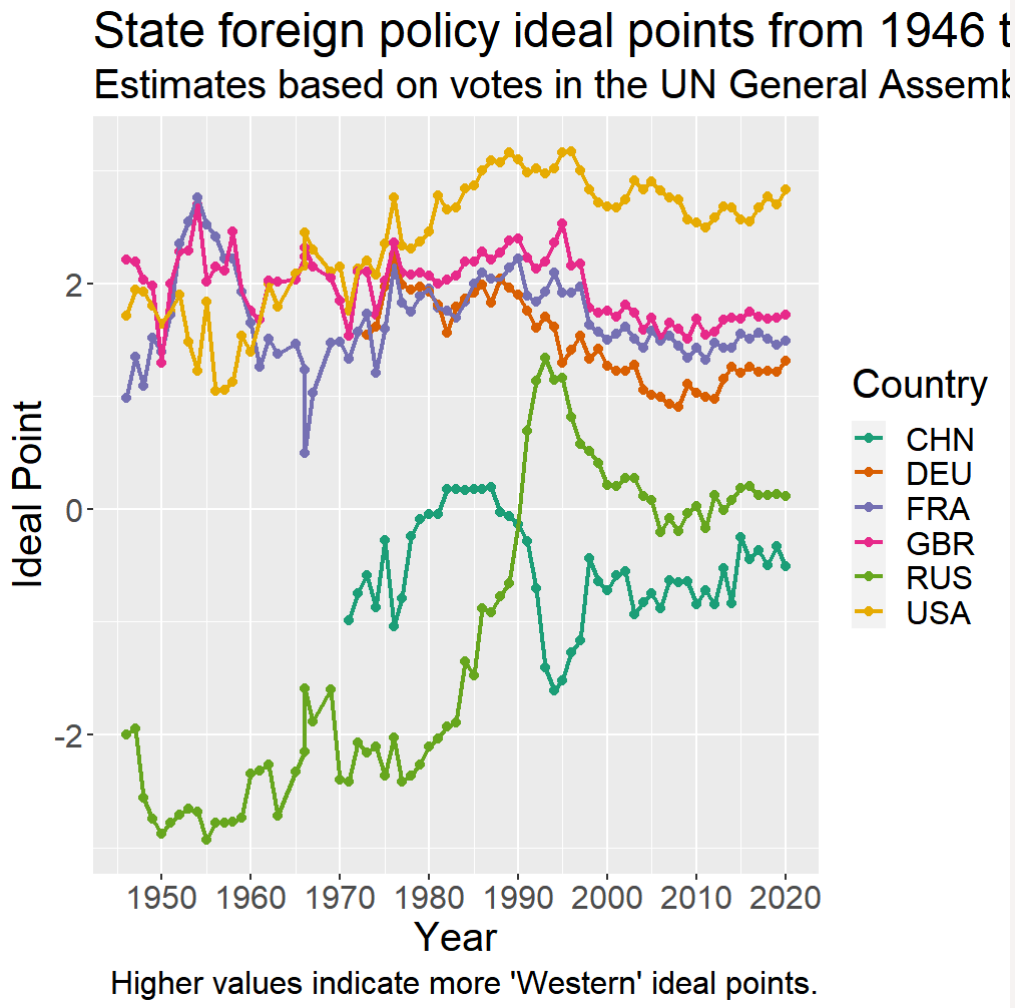
But we don't want this in our case.



A Plot from Scratch

Step 8: Labels

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 2020, 10)) +
  labs(x = "Year",
       y = "Ideal Point",
       color = "Country",
       title = "State foreign policy ideal points from 19
       subtitle = "Estimates based on votes in the UN Gen
       caption = "Higher values indicate more 'Western' i
```

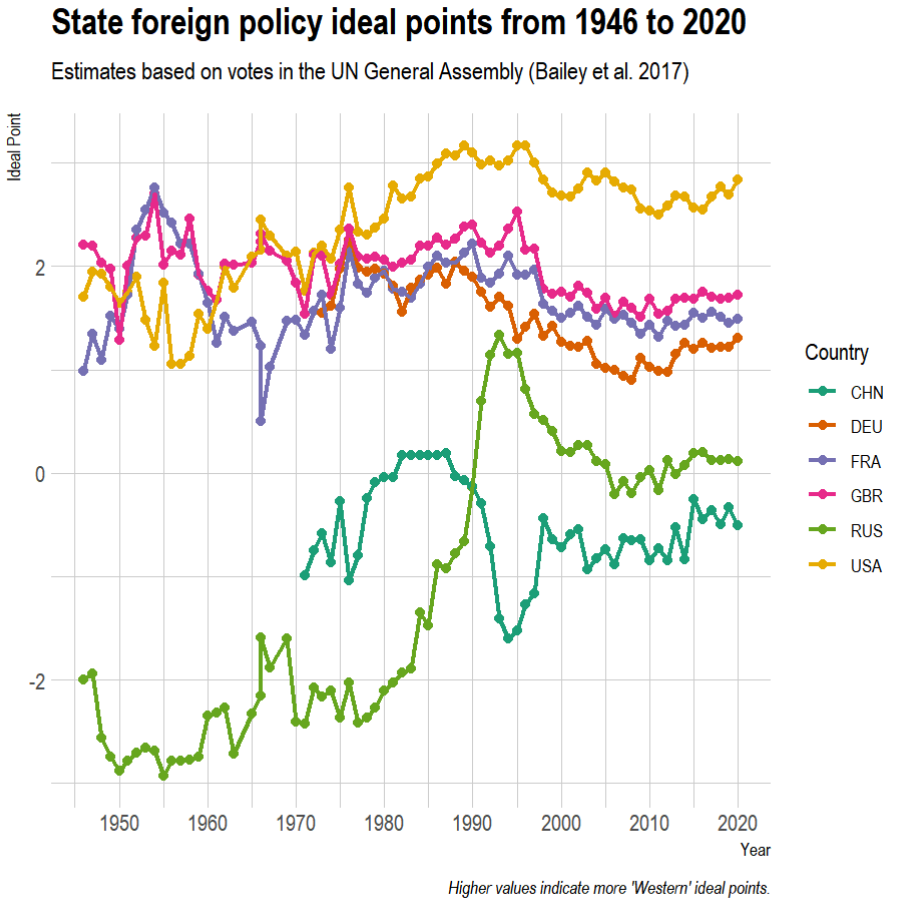


A Plot from Scratch

Step 9: Themes

```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 2020, 10)) +
  labs(x = "Year",
       y = "Ideal Point",
       color = "Country",
       title = "State foreign policy ideal points from 19
       subtitle = "Estimates based on votes in the UN Gen
       caption = "Higher values indicate more 'Western' i
  hrbrthemes::theme_ipsum() #<< # There are many ggplot t
```

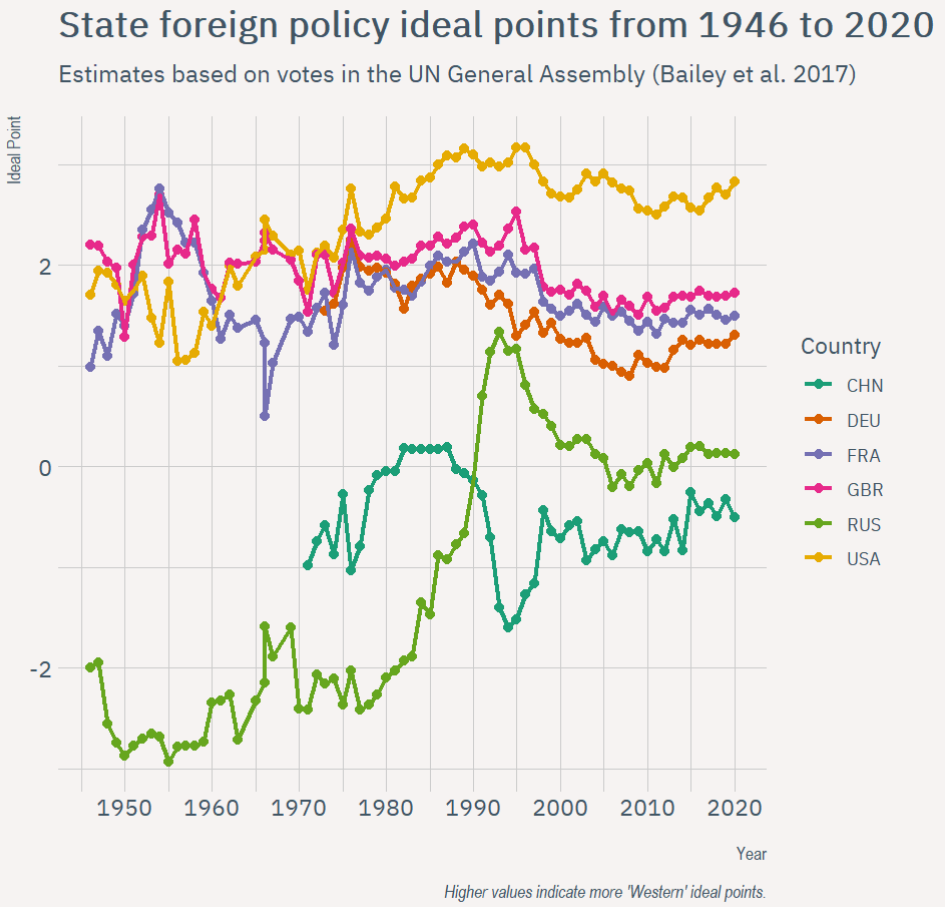
Tipp: MORE THEMES!?! Take a look **here**, **here**, **here**, **here** or **here**.



A Plot from Scratch

Step 10: Going wild - theme tuning

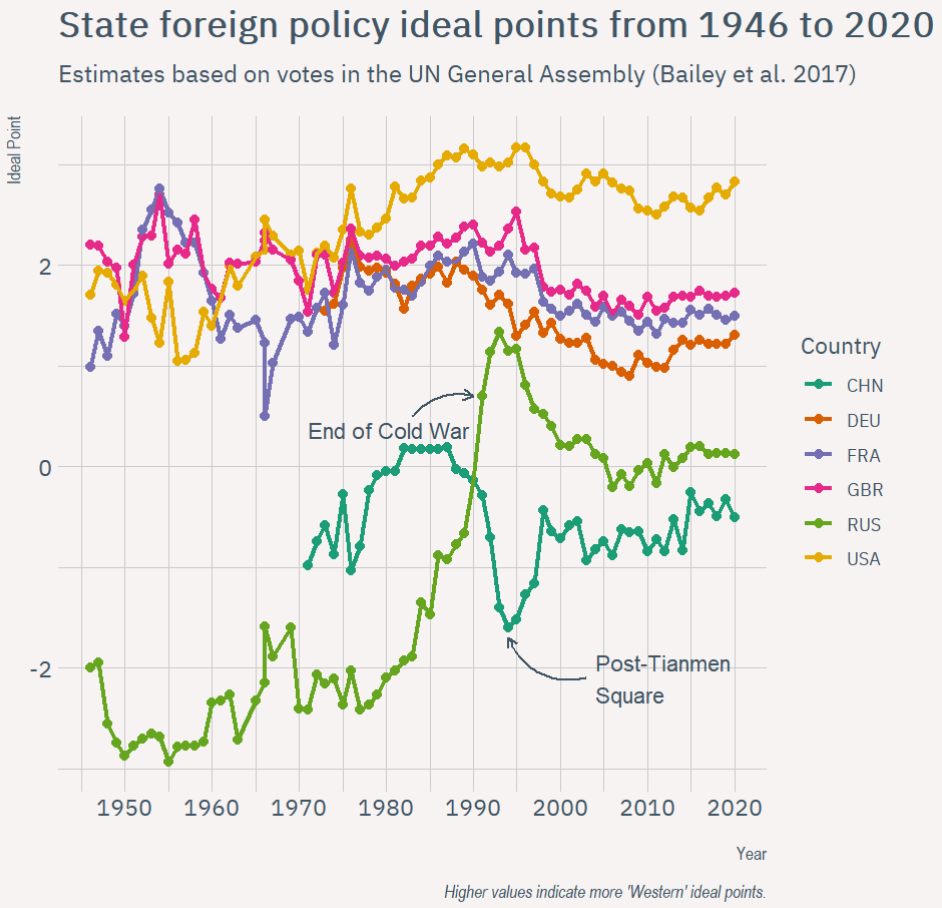
```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 2020, 10)) +
  labs(x = "\nYear",
       y = "Ideal Point",
       color = "Country",
       title = "State foreign policy ideal points from 1946 to 2020",
       subtitle = "Estimates based on votes in the UN General Assembl
       caption = "Higher values indicate more 'Western' ideal points.
  hrbrthemes::theme_ipsum() +
  theme(text = element_text(colour = "#415564", family = "IBM Plex Sa
        plot.title = element_text(colour = "#415564", family = "IBM P
        plot.subtitle = element_text(colour = "#415564", family = "IB
        plot.background = element_rect(fill = "#f6f3f2", color = "#f6
        panel.border = element_blank(),
        axis.text = element_text(colour = "#415564"),
        axis.title = element_text(colour = "#415564"))
```



A Plot from Scratch

Step 11: Going wild - Annotations

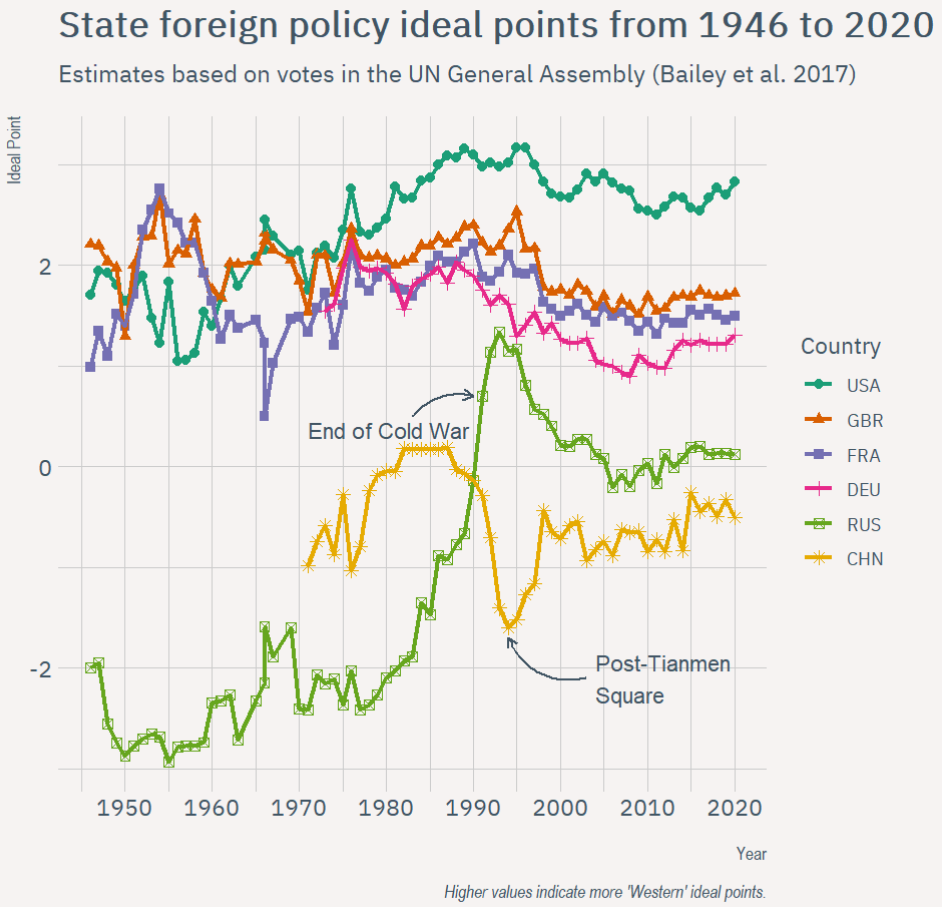
```
ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 2020, 10)) +
  labs(x = "\nYear",
       y = "Ideal Point",
       color = "Country",
       title = "State foreign policy ideal points from 1946 to 2020",
       subtitle = "Estimates based on votes in the UN General Assembly (Bailey et al. 
       caption = "Higher values indicate more 'Western' ideal points.") +
  hrbrthemes::theme_ipsum() +
  theme(text = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.title = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.subtitle = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.background = element_rect(fill = "#f6f3f2", color = "#f6f3f2"),
        panel.border = element_blank(),
        axis.text = element_text(colour = "#415564"),
        axis.title = element_text(colour = "#415564")) +
  annotate(geom = "curve", xend = 1990, yend = 0.7, x = 1983, y = 0.5,
           curvature = -.3, arrow = arrow(length = unit(2, "mm")), color = "#415564") 
  annotate(geom = "text", x = 1971, y = 0.37, label = "End of Cold War", hjust = "left
  annotate(geom = "curve", xend = 1994, yend = -1.7, x = 2003, y = -2.1,
           curvature = -.4, arrow = arrow(length = unit(2, "mm")), color = "#415564") 
  annotate(geom = "text", x = 2004, y = -2.1, label = "Post-Tianmen \nSquare", hjust =
```



A Plot from Scratch

Step 12: Fine tuning the legend and shapes

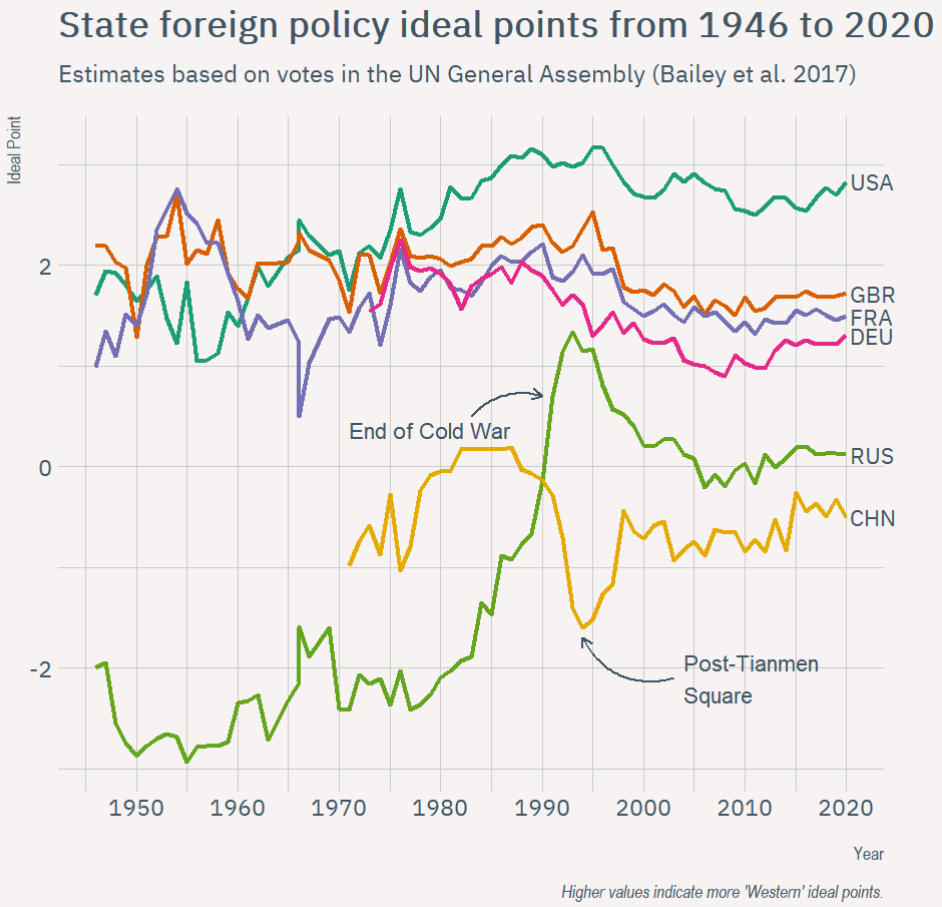
```
ideal <- ideal %>%
  mutate(iso3c = fct_relevel(iso3c, c("USA", "GBR", "FRA", "DEU", "RUS", "CHN")))

ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c,
                     shape = iso3c)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 2020, 10)) +
  labs(x = "\nYear",
       y = "Ideal Point",
       color = "Country",
       shape = "Country",
       title = "State foreign policy ideal points from 1946 to 2020",
       subtitle = "Estimates based on votes in the UN General Assembly (Bailey et al. 2017)",
       caption = "Higher values indicate more 'Western' ideal points.") +
  hrbrthemes::theme_ipsum() +
  theme(text = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.title = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.subtitle = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.background = element_rect(fill = "#f6f3f2", color = "#f6f3f2"),
        panel.border = element_blank(),
        axis.text = element_text(colour = "#415564"),
        axis.title = element_text(colour = "#415564")) +
  annotate(geom = "curve", xend = 1990, yend = 0.7, x = 1983, y = 0.5,
           curvature = -.3, arrow = arrow(length = unit(2, "mm")), color = "#415564") +
  annotate(geom = "text", x = 1971, y = 0.37, label = "End of Cold War", hjust = "left", color = "#
  annotate(geom = "curve", xend = 1994, yend = -1.7, x = 2003, y = -2.1,
           curvature = -.4, arrow = arrow(length = unit(2, "mm")), color = "#415564") +
  annotate(geom = "text", x = 2004, y = -2.1, label = "Post-Tianmen \nSquare", hjust = "left", colo
```



A Plot from Scratch

Alternative "Legend"

```
ideal_fin <- filter(ideal, year == 2020)

ggplot(data = ideal,
       mapping = aes(x = year,
                     y = IdealPointAll,
                     color = iso3c)) +
  geom_line(size = 1) +
  scale_colour_brewer(palette = "Dark2") +
  scale_x_continuous(breaks = seq(1950, 2020, 10)) +
  labs(x = "\nYear",
       y = "Ideal Point",
       color = "Country",
       title = "State foreign policy ideal points from 1946 to 2020",
       subtitle = "Estimates based on votes in the UN General Assembly (Bailey et al. 2017)",
       caption = "Higher values indicate more 'Western' ideal points.") +
  hrbrthemes::theme_ipsum() +
  theme(text = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.title = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.subtitle = element_text(colour = "#415564", family = "IBM Plex Sans"),
        plot.background = element_rect(fill = "#f6f3f2", color = "#f6f3f2"),
        panel.border = element_blank(),
        axis.text = element_text(colour = "#415564"),
        axis.title = element_text(colour = "#415564"),
        legend.position = "none",
        axis.text.y.right = element_text(margin = margin(0, 0, 0, -20))) +
  annotate(geom = "curve", xend = 1990, yend = 0.7, x = 1983, y = 0.5,
           curvature = -.3, arrow = arrow(length = unit(2, "mm")), color = "#415564") +
  annotate(geom = "text", x = 1971, y = 0.37, label = "End of Cold War", hjust = "left", color = "#
  annotate(geom = "curve", xend = 1994, yend = -1.7, x = 2003, y = -2.1,
           curvature = -.4, arrow = arrow(length = unit(2, "mm")), color = "#415564") +
  annotate(geom = "text", x = 2004, y = -2.1, label = "Post-Tianmen \nSquare", hjust = "left", colo
  scale_y_continuous(sec.axis = dup_axis(breaks = ideal_fin$IdealPointAll, labels = c("USA", "GBR",
```



Setting the theme globally

Adding a theme to every plot can be cumbersome.

Luckily, we can use `theme_set()` to set it globally.

Changing default fonts is also a bit fiddly (esp. on windows).

- You can use the `extrafont` package and see, e.g., **here**.
- Alternatively, you can use the package `showtext` to load fonts directly from Google fonts!

```
theme_set(plex)
```

Saving your work

The last step is now to save your plot.

We can do this with `ggsave`. Exports the last graphic you plotted or the plot object u specify.

```
ggsave("ideal_points.png", width = 9, height = 7)
```

If we work with `.rmd` documents, we can just control the output using chunk options (but more on this later).

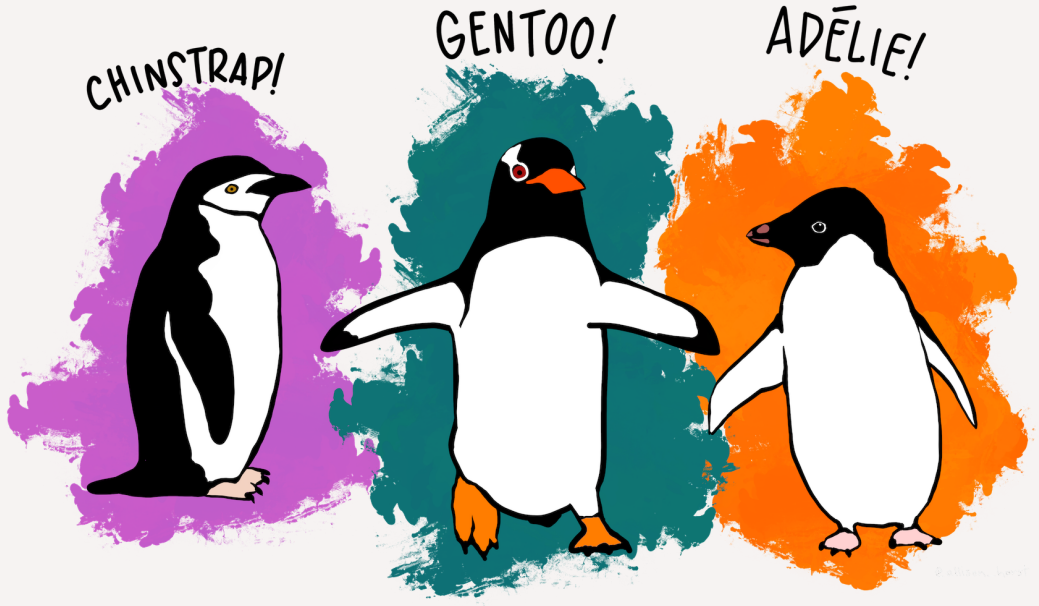
Other Geoms

You can find all geoms here: **https://ggplot2.tidyverse.org/reference/#section-geoms**.

More examples: Amounts, Props and Distributions

To explore some other geoms and plots, we will just use data on the **Palmer penguins**, a popular exemplary data set.

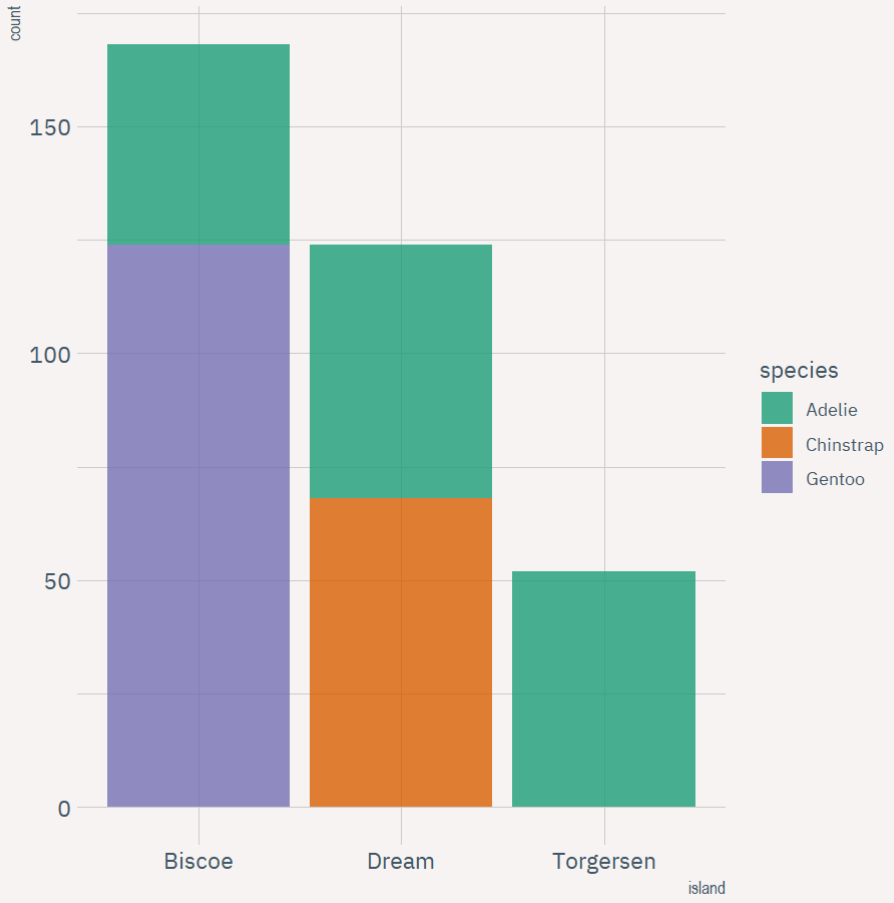
> The `palmerpenguins` data contains size measurements for three penguin species observed on three islands in the Palmer Archipelago, Antarctica.



More examples: Amounts, Props and Distributions

Visualizing Amounts

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(alpha = 0.8) +
  scale_fill_brewer(palette = "Dark2")
```



Q What are the pros and cons?

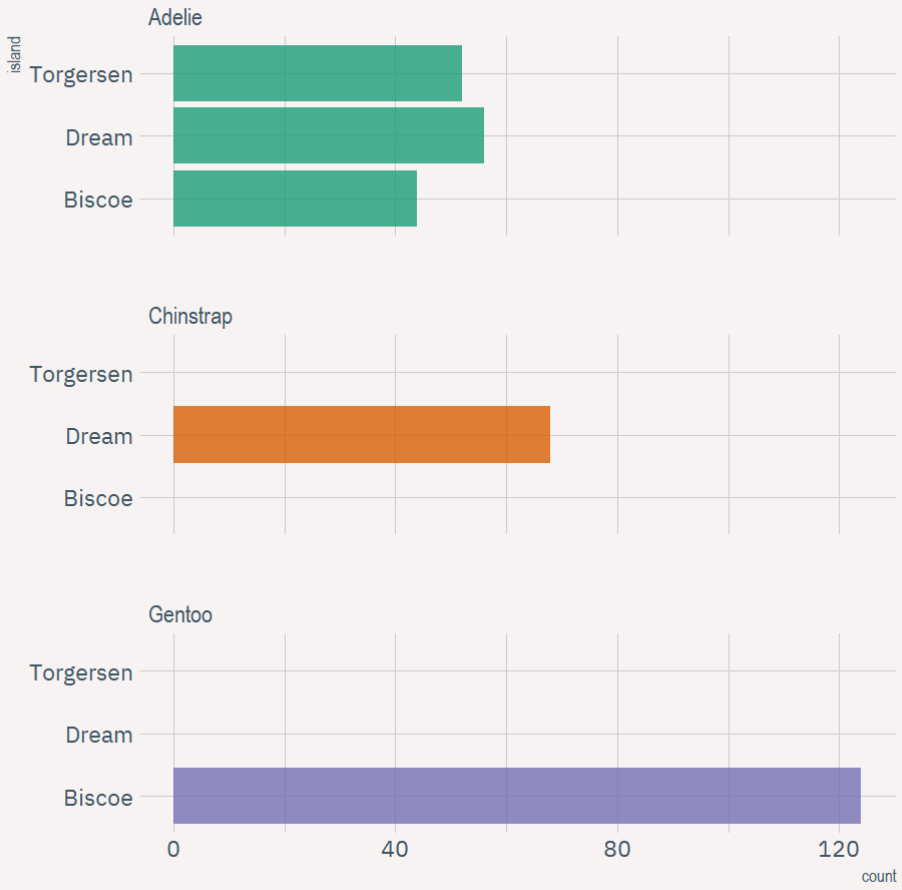
Fine Point: `geom_bar()` automatically counts the number of cases and displays the height of the bars accordingly. If you want the height to represent other values in the data use `geom_col()`.

More examples: Amounts, Props and Distributions

Visualizing Amounts

```
ggplot(penguins, aes(x = island, fill = species)) +
  geom_bar(alpha = 0.8) +
  scale_fill_brewer(palette = "Dark2") +
  facet_wrap(~species, ncol = 1) +
  coord_flip() +
  theme(legend.position = "none")
```

Q What are the pros and cons?



More examples: Amounts, Props and Distributions

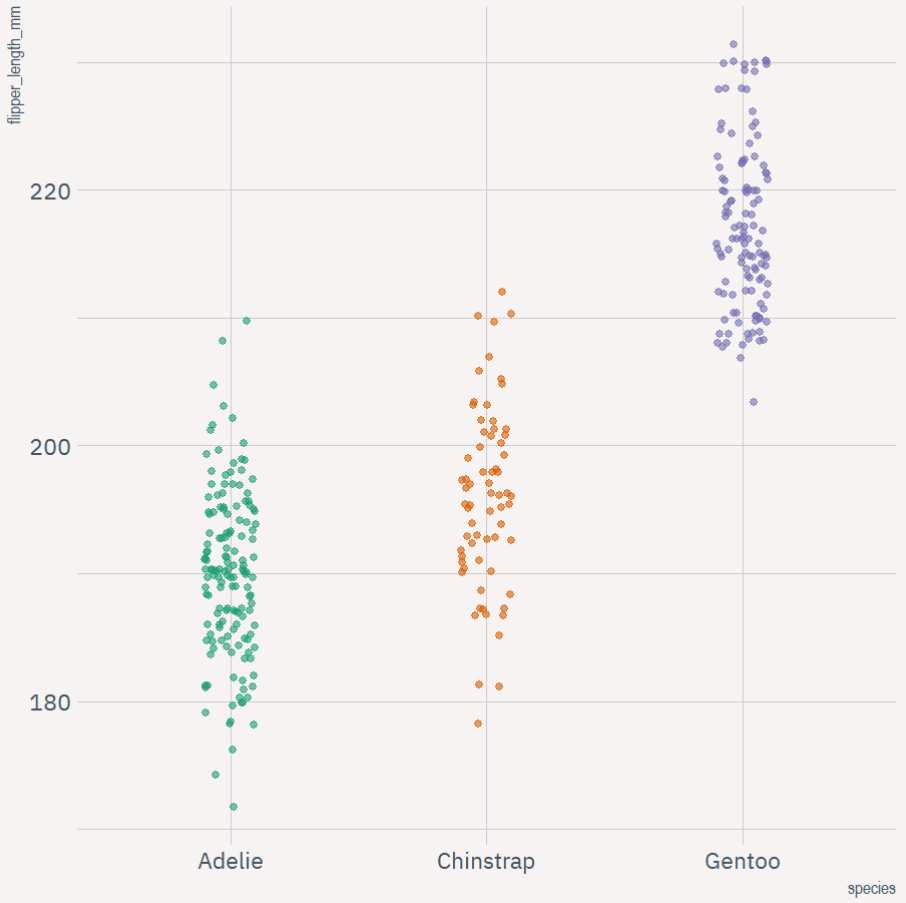
Visualizing Cont. Distributions

```
ggplot(penguins, aes(x = species, y = flipper_length_mm)) +
  geom_point(alpha = 0.6,
             aes(color = species),
             show.legend = FALSE) +
  scale_color_brewer(palette = "Dark2")
```

More examples: Amounts, Props and Distributions

Visualizing Cont. Distributions

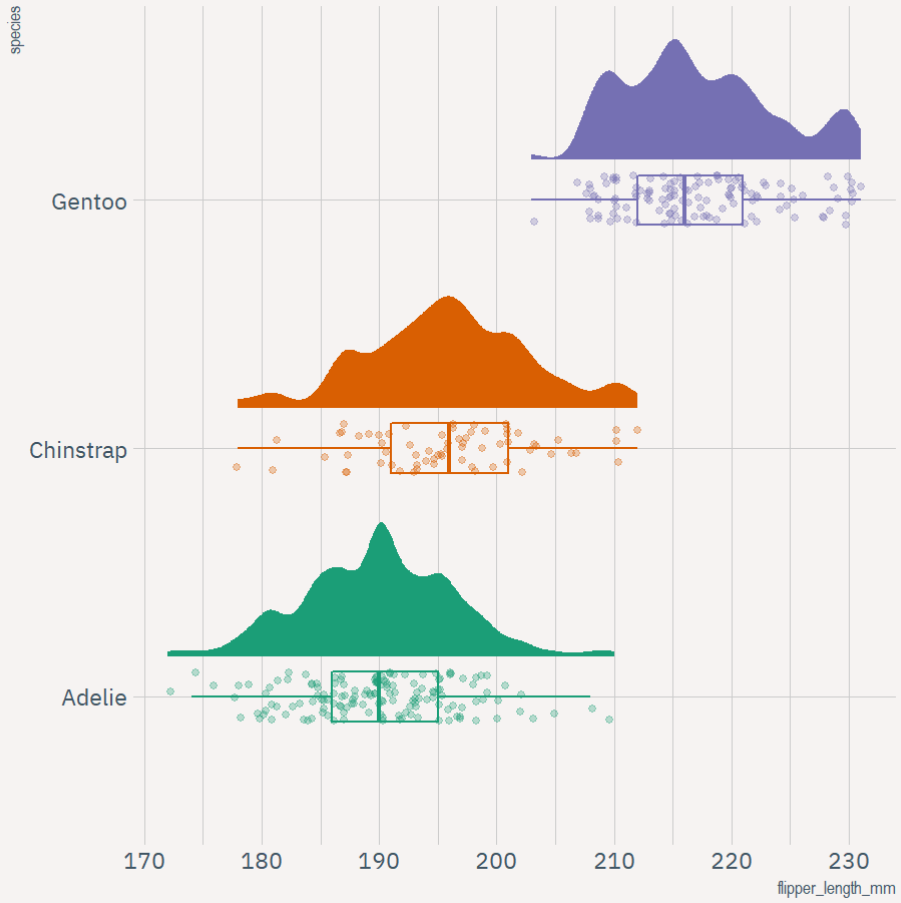
```
set.seed(213)
ggplot(penguins, aes(x = species, y = flipper_length_mm, color = species)) +
  geom_point(alpha = 0.6,
             show.legend = FALSE,
             position = position_jitter(seed = 213, width = .1)) +
  scale_color_brewer(palette = "Dark2")
```



More examples: Amounts, Props and Distributions

Visualizing Cont. Distributions

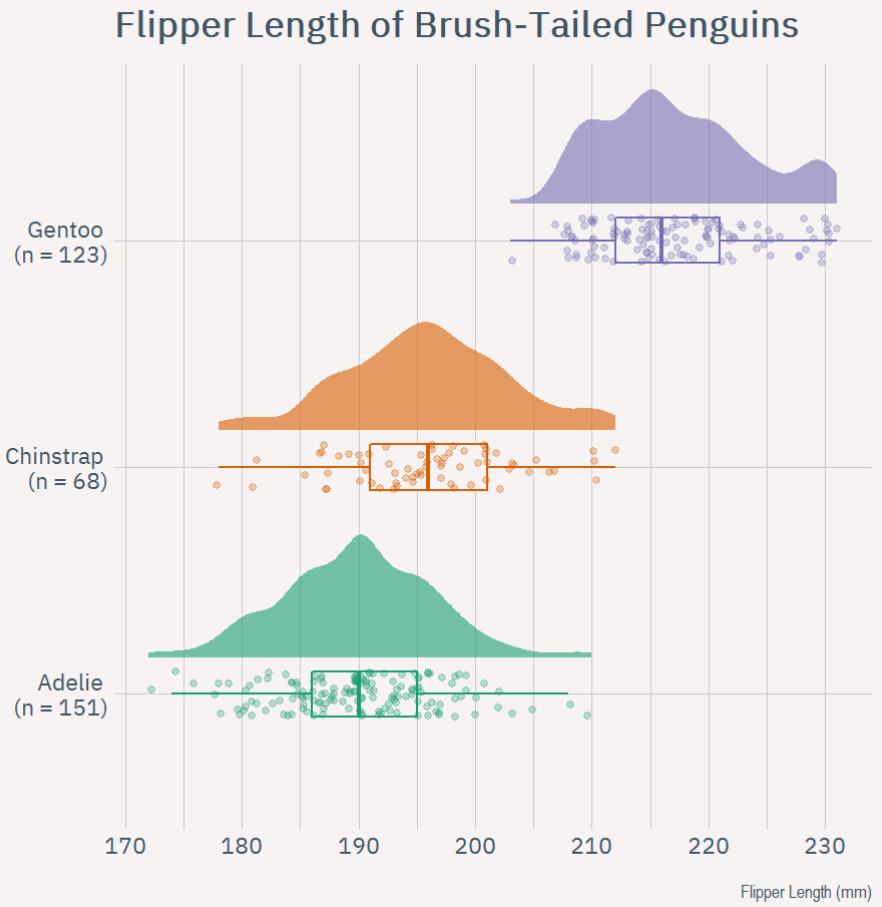
```
set.seed(213)
penguins2 <- filter(penguins, flipper_length_mm != is.na(flipper_length_mm))
ggplot(penguins2, aes(x = species, y = flipper_length_mm, color = species)) +
  ggdist::stat_halfeye(adjust = .5,
                       width = .6,
                       .width = 0,
                       justification = -.3,
                       point_colour = NA,
                       aes(fill = species)) +
  geom_boxplot(width = .2,
               outlier.shape = NA, fill = "#f6f3f2") +
  geom_point(alpha = 0.3,
             position = position_jitter(seed = 213, width = .1)) +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette = "Dark2") +
  theme(legend.position = "none") +
  coord_flip()
```



More examples: Amounts, Props and Distributions

Visualizing Cont. Distributions

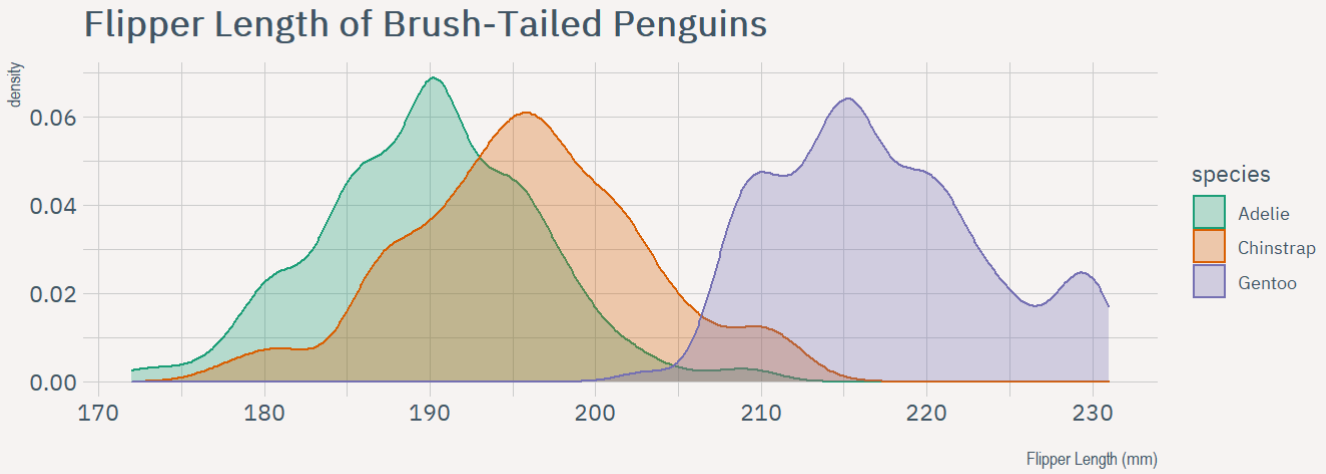
```
set.seed(213)
penguins2 <- filter(penguins, flipper_length_mm != is.na(flipper_length_mm))

penguins2 <- penguins2 %>%
  group_by(species) %>%
  mutate(n = n()) %>%
  ungroup() %>%
  mutate(spec_n = paste0(species, " \n(n = ", n, ")"))

ggplot(penguins2, aes(x = spec_n, y = flipper_length_mm, color = spec_n)) +
  ggdist::stat_halfeye(adjust = .7,
                       width = .6,
                       .width = 0,
                       justification = -.3,
                       point_colour = NA,
                       alpha = 0.6,
                       aes(fill = spec_n)) +
  geom_boxplot(width = .2,
               outlier.shape = NA, fill = "#f6f3f2") +
  geom_point(alpha = 0.3,
             position = position_jitter(seed = 213, width = .1)) +
  scale_color_brewer(palette = "Dark2") +
  scale_fill_brewer(palette = "Dark2") +
  theme(legend.position = "none") +
  coord_flip() +
  labs(title = "Flipper Length of Brush-Tailed Penguins",
       x = "",
       y = "\nFlipper Length (mm)")
```



More examples: Amounts, Props and Distributions

Visualizing Cont. Distributions

```
ggplot(penguins, aes(x = flipper_length_mm, fill = species, color = species)) +
  geom_density(alpha = 0.3, adjust = .7) +
  labs(title = "Flipper Length of Brush-Tailed Penguins",
       x = "\nFlipper Length (mm)") +
  scale_fill_brewer(palette = "Dark2") +
  scale_color_brewer(palette = "Dark2")
```



Q Pros and cons?

Maps

Wrangling and visualizing spatial data (i.e. **GIS stuff**) is a pretty complex subject.

You got to think of **projections**, the **use of color**, **spurious/meaningless spatial correlations**, and a lot more...

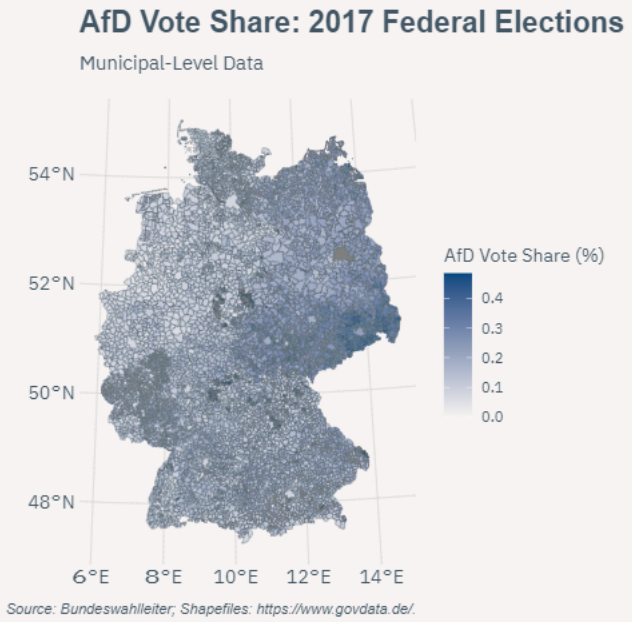
We won't be able to dive into it but here are some pointers:

- `sf` is probably the most complete package to do GIS stuff in R.
- Check out **this** book on spatial data science with R.
- Check out `leaflet` for interactive maps.

Maps

Once you get a hang of **shapefiles and the information they store**, **projections/coordinate reference systems**, and figured out how to join your administrative data with it, it's not too difficult...

```
ger_map <- sf::read_sf("data/.shapes/VG250_GEM.shp")
elec_res <- rio::import("data/bundeswahlleiter_17.rds")
afd_plot <- dplyr::left_join(ger_map, elec_res, by = c("AGS" = "ags"))

ggplot(afd_plot) +
  geom_sf(aes(fill = afd_share),
          size = 0.1,
          color = "#415564") +
  scale_fill_gradient(low = "#f6f4f2",
                      high = "#014980") +
  labs(title = "AfD Vote Share: 2017 Federal Elections",
       subtitle = "Municipal-Level Data",
       fill = "AfD Vote Share (%)",
       caption = "Source: Bundeswahlleiter; Shapefiles: https://www.govdata.d
```



Combining Plots: patchwork

My favorite little data viz helper: `patchwork`.

```
p1 <- ggplot(data = penguins, aes(x = bill_length_mm, y = bill_depth_mm)) +
  geom_point(aes(color = species, shape = species), size = 2)

p2 <- ggplot(data = penguins, aes(x = flipper_length_mm)) +
  geom_histogram(aes(fill = species), alpha = 0.5, position = "identity")

p1 / p2
```

Enables you to connect plots with `+`. And to stack `/` and pack them.

Data Viz Challenge

Next Up: Writing Functions and Clean Code