Day 5: Data visualization and communication

Data Visualization Tools and Techniques

Sebastian Ramirez-Ruiz
Hertie School

Table of contents

1. Principles of good data visualization[1]
2. Good practices in data visualization
3. The best statistical graph of all times
4. Click- and code-based visualization tools
5. Visualization with R and `ggplot2`

[1] Much of this section draws on materials from Claus Wilke's excellent book *Fundamentals of Data Visualization*.

Principles of good data visualization

On the shoulder of giants...

"Visualization is surprisingly difficult. Even the most simple matters can easily go wrong."

"No matter how clever the choice of the information, and no matter how technologically impressive the encoding, a visualization fails if the decoding fails."

William Cleveland

"Although nothing can replace a good graphical idea applied to an interesting set of numbers, editing and revision are as essential to sound graphical work as they are to writing."

"Design cannot rescue failed content."

"Above all else show the data."

Edward Tufte

Fundamental principles of analytic design

In Chapter 5 of his book "Beautiful Evidence", Edward Tufte outlines six fundamental principles of analytic design:

1. **Comparisons.** Show *comparisons*, *contrasts*, *differences*.
2. **Causality, mechanism, structure, explanation.** Show causality, mechanism, explanation, *systematic structure*.
3. **Multivariate analysis.** Show data; that is show more than 1 or 2 variables (*Use additional facets to add dimensions to the 2d plane*.)
4. **Integration of evidence.** Completely integrate *words*, *numbers*, *images*, *diagrams*.
5. **Documentation.** *Thoroughly describe* the evidence. Provide a detailed title, indicate the authors and sponsors, document the data sources, show complete measurement scales, point out relevant issues.
6. **Content counts most of all.** Analytical presentations ultimately stand or fall depending on the quality, relevance and integrity of their content ("What is the problem you want to solve?")

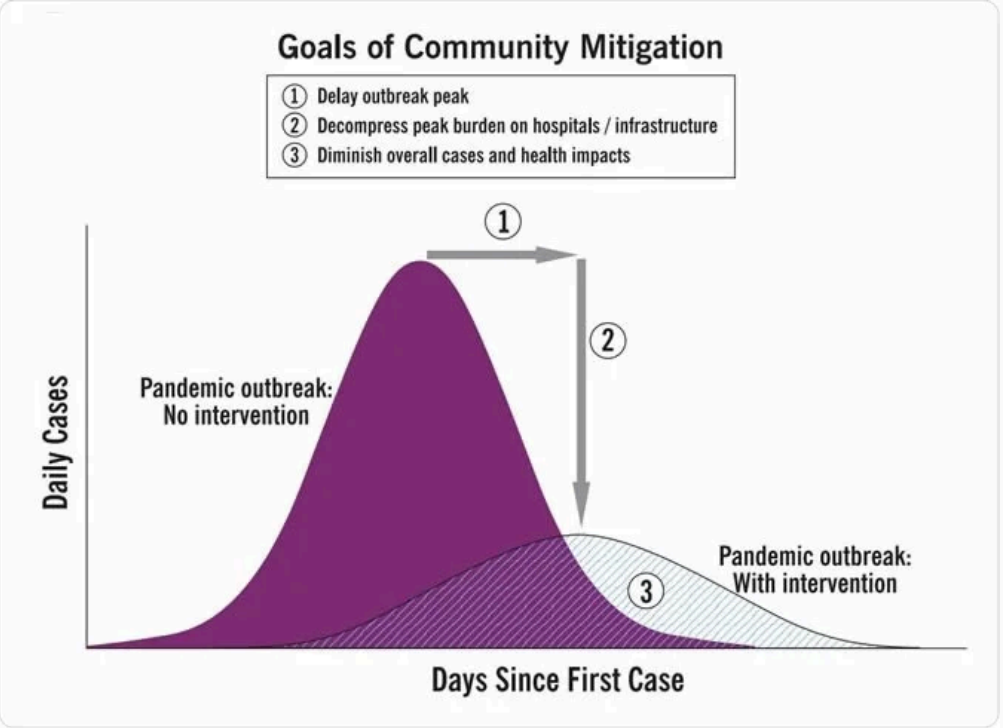
Flattening the curve

Thinking through visuals

We can illustrate these with one the **most influential graphs** of the past couple of years. This visual traveled across the world in the start of 2020.

Can you spot how these *principles were implemented in the different versions of the graph*?

1. **Comparisons.**
2. **Causality, mechanism, structure, explanation.**
3. **Multivariate analysis.**
4. **Integration of evidence.**
5. **Documentation.**
6. **Content counts most of all.**



`Source` CDC

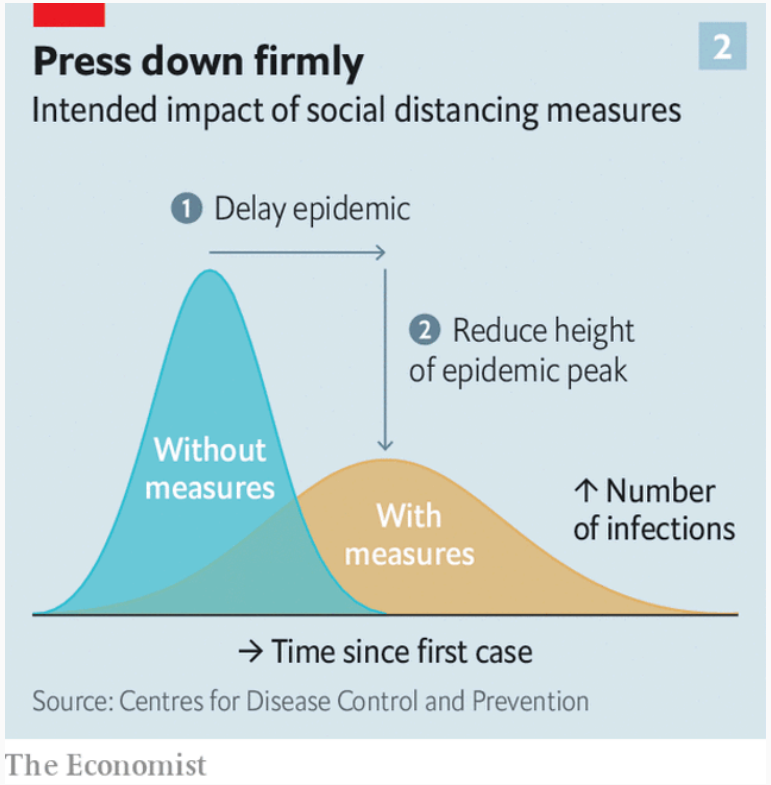
Flattening the curve

Thinking through visuals

We can illustrate these with one the **most influential graphs** of the past couple of years. This visual traveled across the world in the start of 2020.

Can you spot how these *principles were implemented in the different versions of the graph*?

1. **Comparisons.**
2. **Causality, mechanism, structure, explanation.**
3. **Multivariate analysis.**
4. **Integration of evidence.**
5. **Documentation.**
6. **Content counts most of all.**



`Source` The Economist

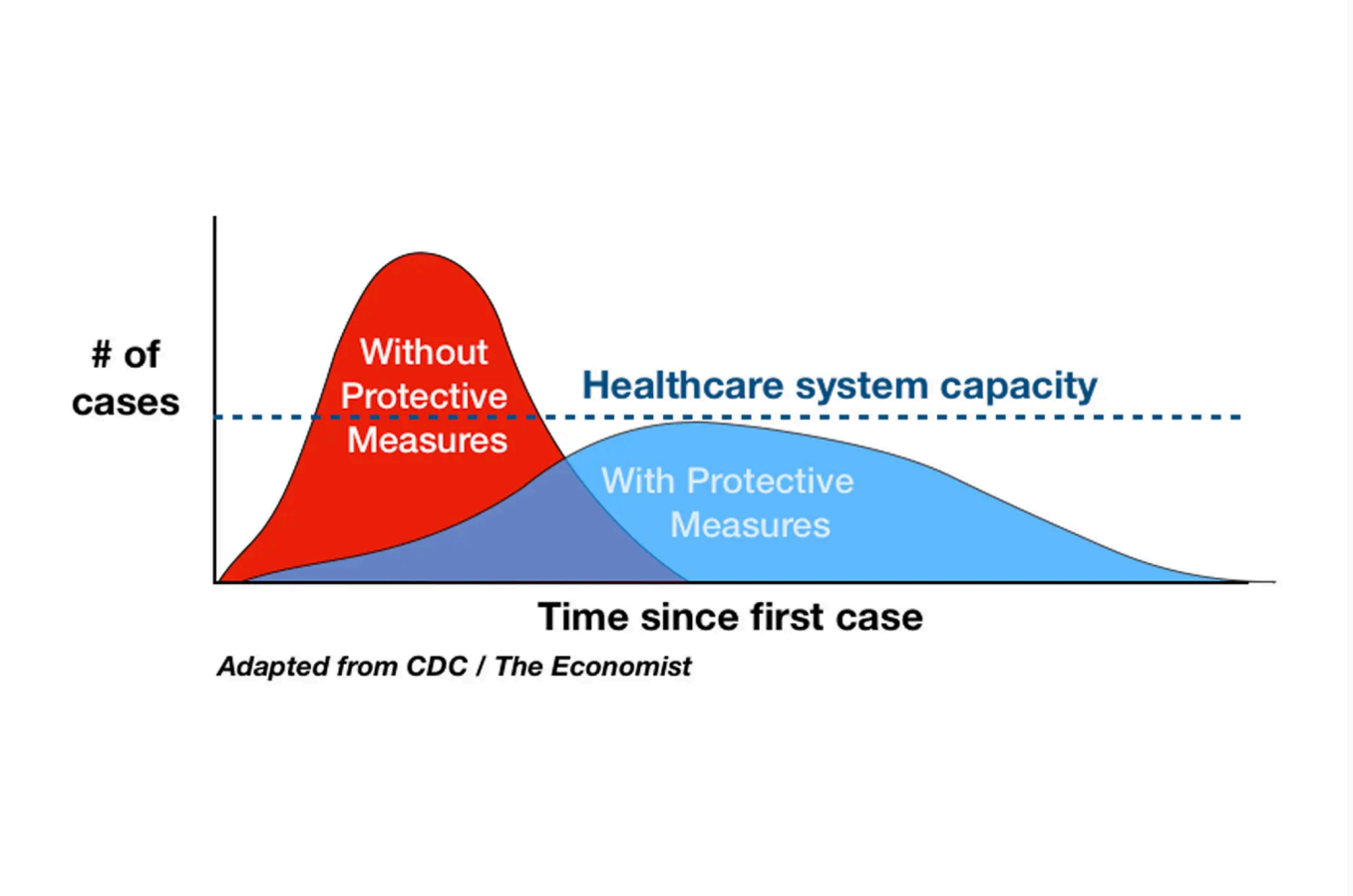
Flattening the curve

Thinking through visuals

We can illustrate these with one the **most influential graphs** of the past couple of years. This visual traveled across the world in the start of 2020.

Can you spot how these *principles were implemented in the different versions of the graph*?

1. **Comparisons.**
2. **Causality, mechanism, structure, explanation.**
3. **Multivariate analysis.**
4. **Integration of evidence.**
5. **Documentation.**
6. **Content counts most of all.**



`Source` NYT

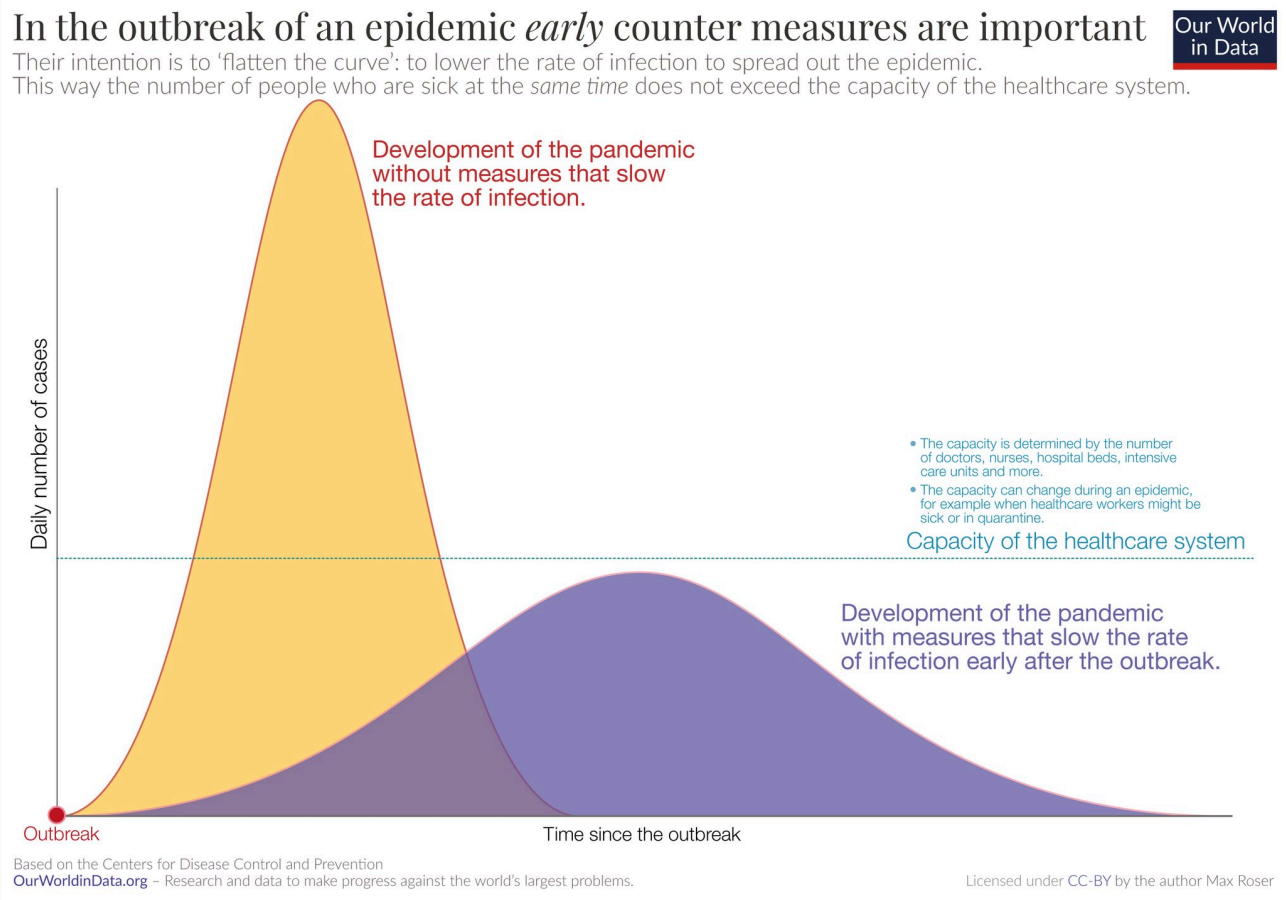
Flattening the curve

Thinking through visuals

We can illustrate these with one the **most influential graphs** of the past couple of years. This visual traveled across the world in the start of 2020.

Can you spot how these *principles were implemented in the different versions of the graph*?

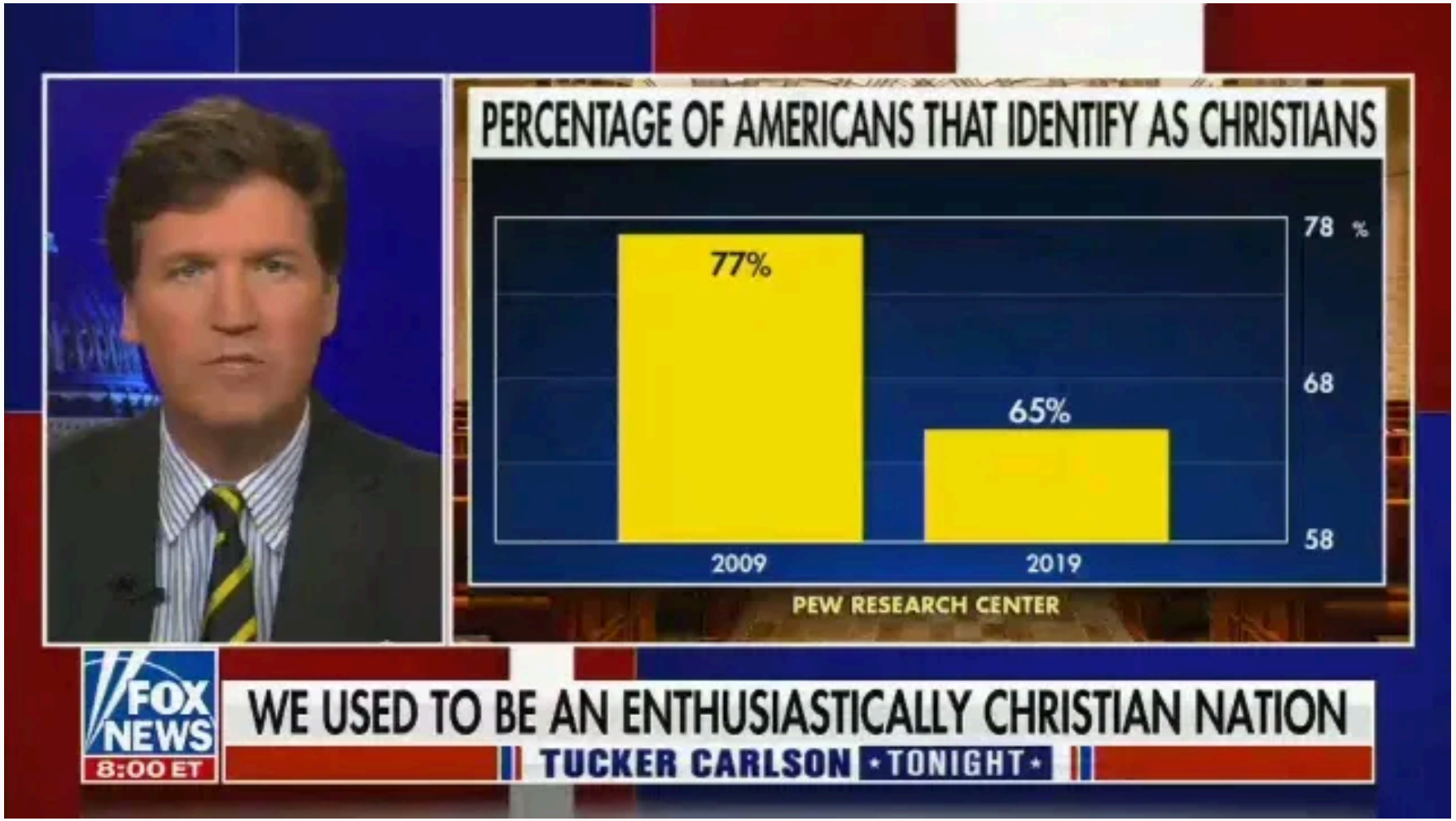
1. **Comparisons.**
2. **Causality, mechanism, structure, explanation.**
3. **Multivariate analysis.**
4. **Integration of evidence.**
5. **Documentation.**
6. **Content counts most of all.**



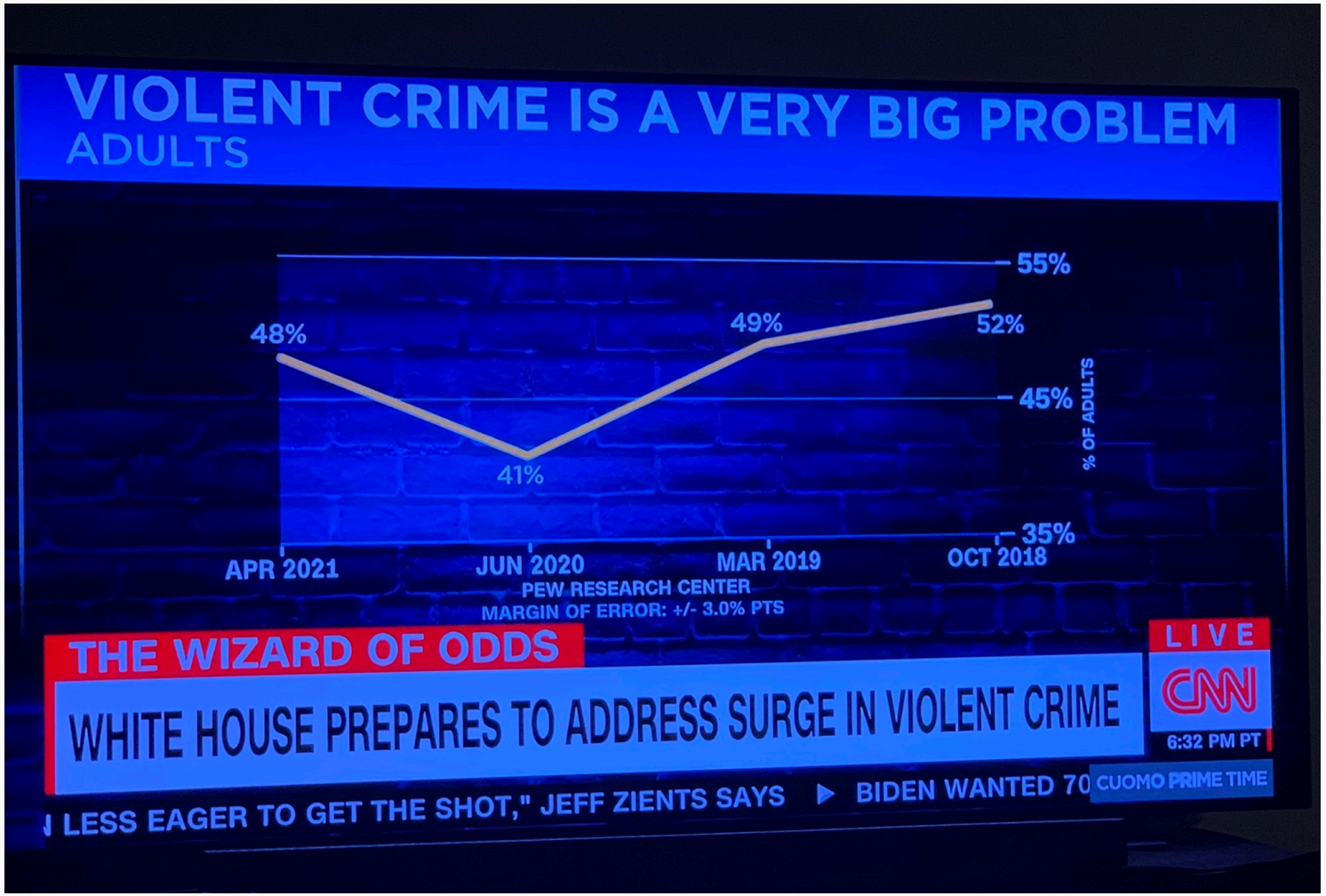
`Source` Our world in data

Good practices in data visualization

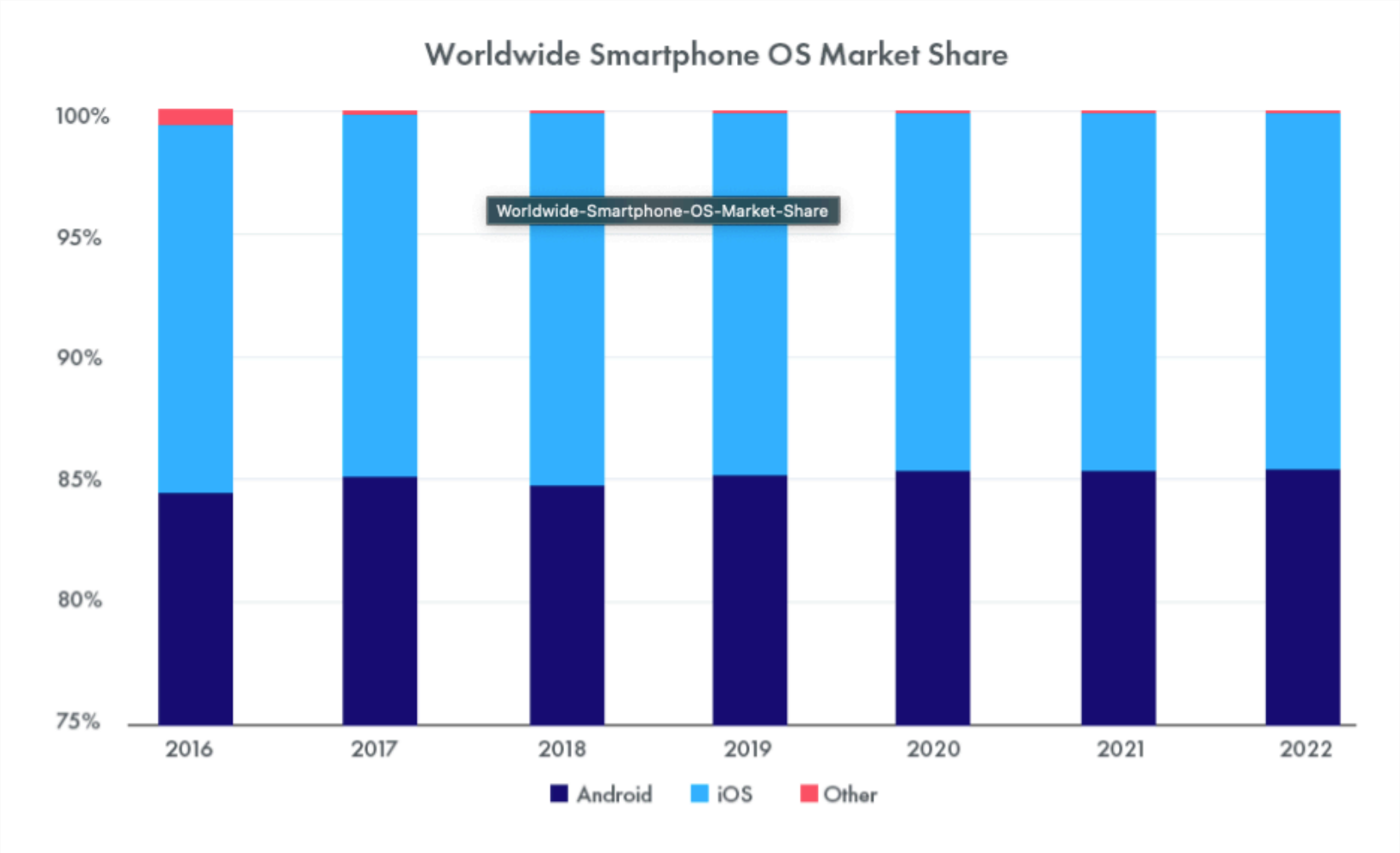
What is happening here?



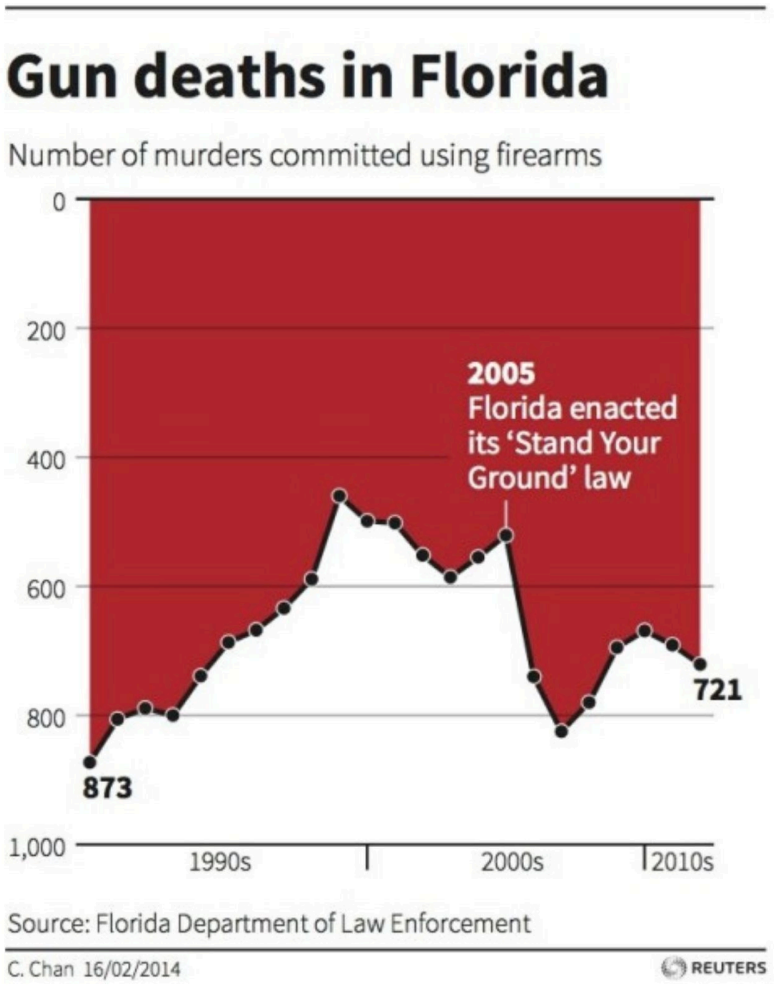
What is happening here? (cont.)



What is happening here? (cont.)

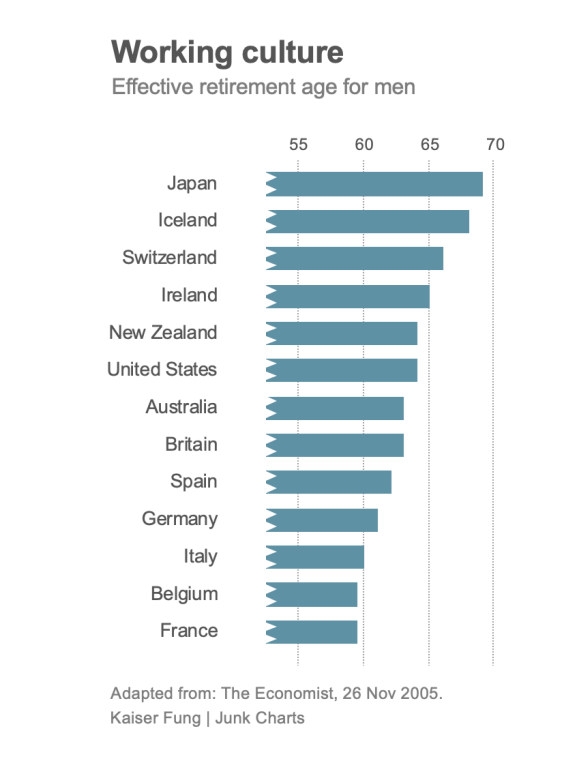


What is happening here? (cont.)



Friends don't let friends...

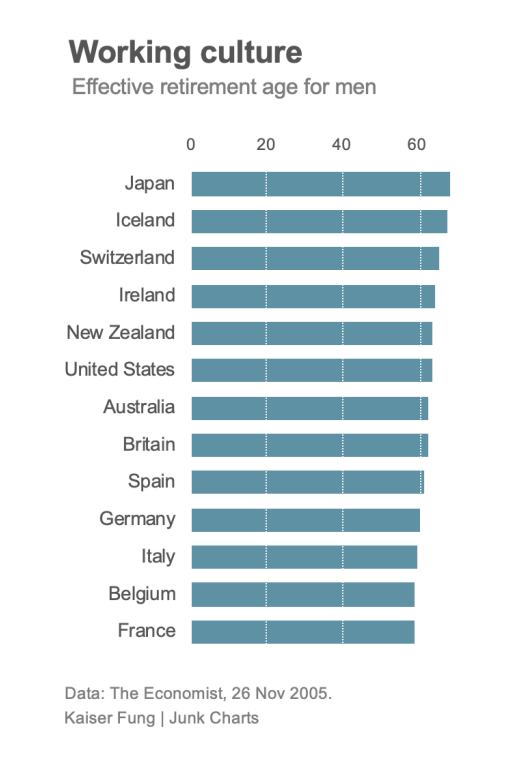
Truncate the axes of a plot at arbitrary points...



Misleading

instead, friends encourage...

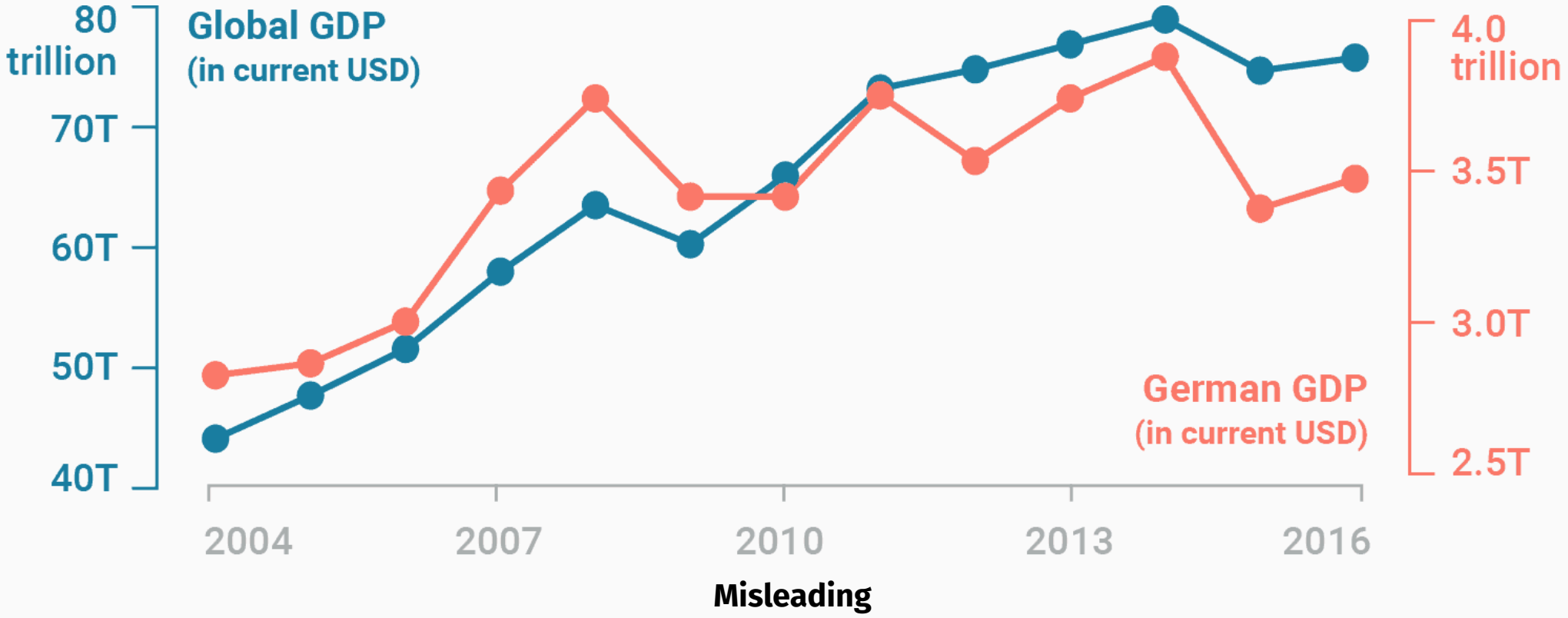
The use of **informative axis spaces**.



Better

Friends don't let friends...

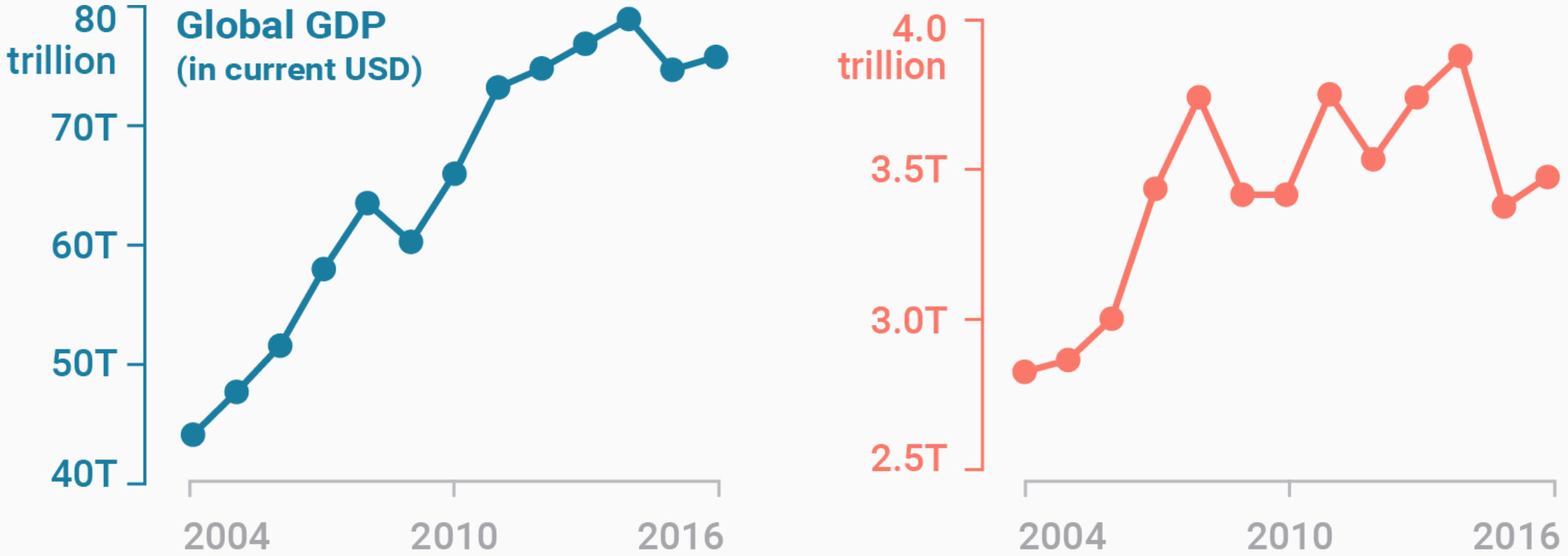
Use **dual axis** charts...



Misleading

instead, friends encourage...

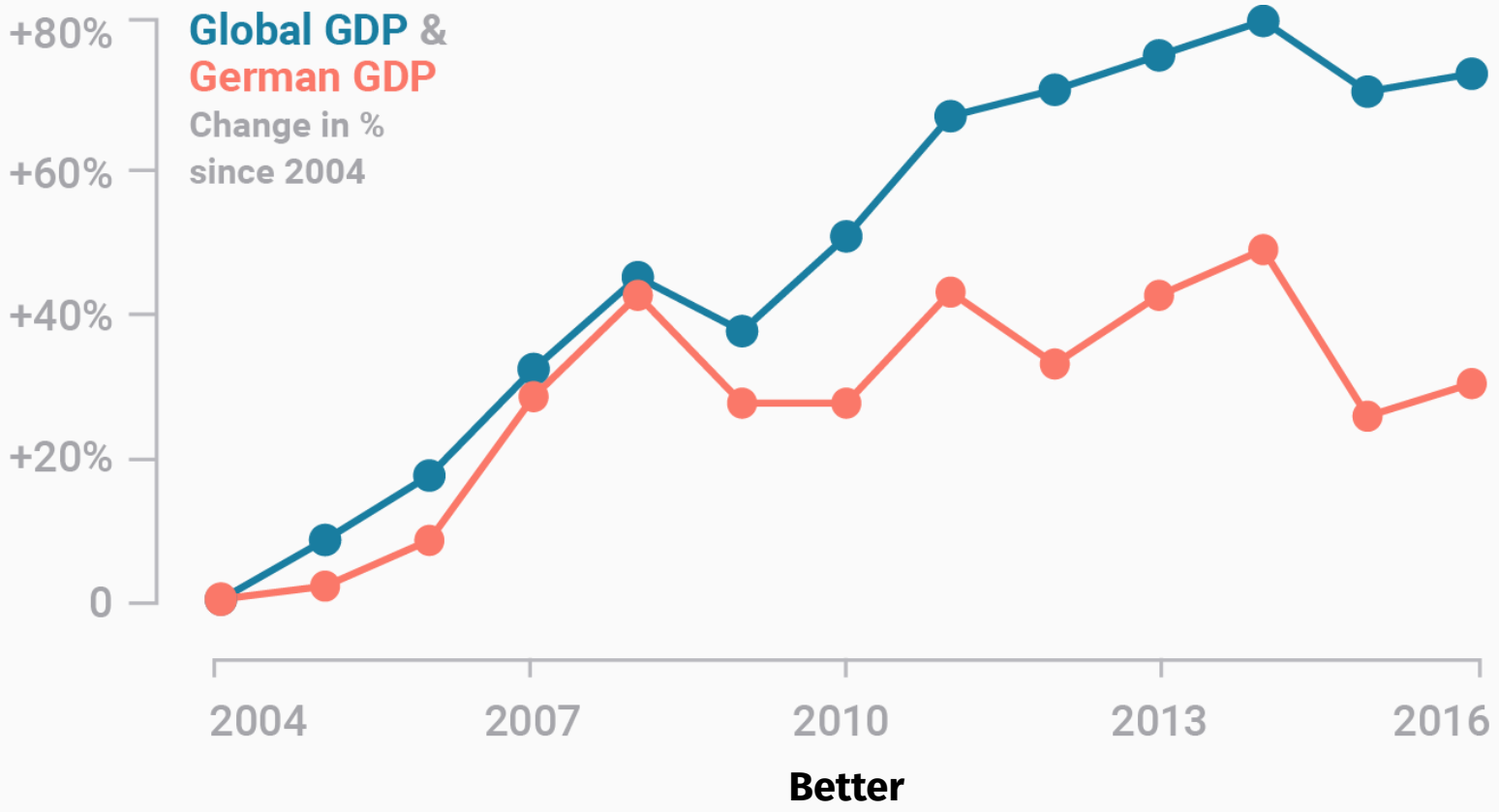
The use of **side-by-side** comparisons.



Better

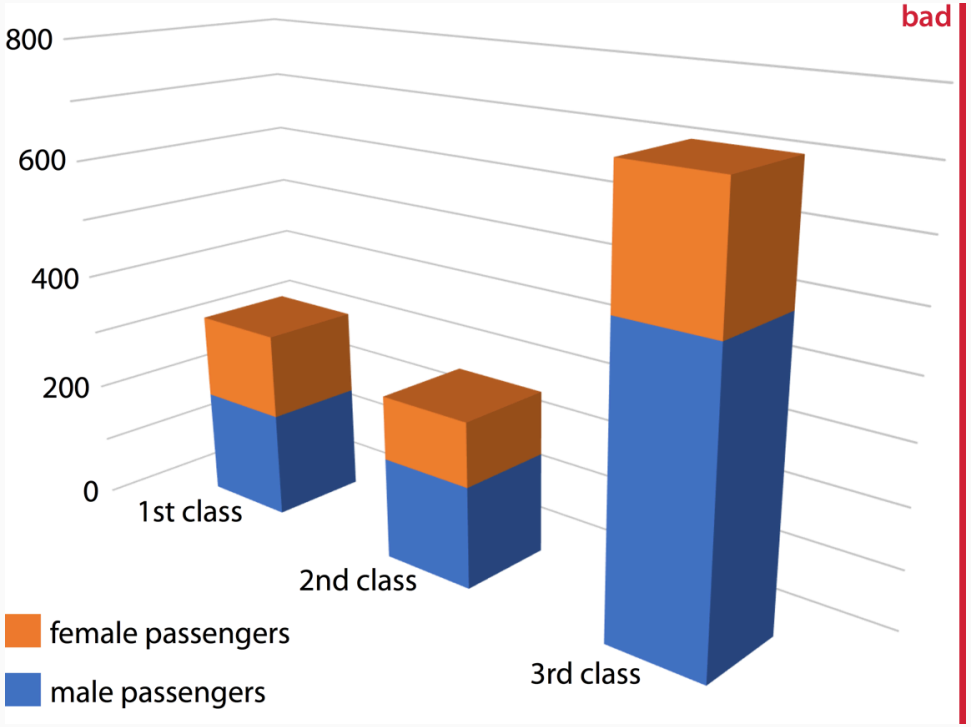
instead, friends encourage...

The use of **indexed** comparisons.



Friends don't let friends...

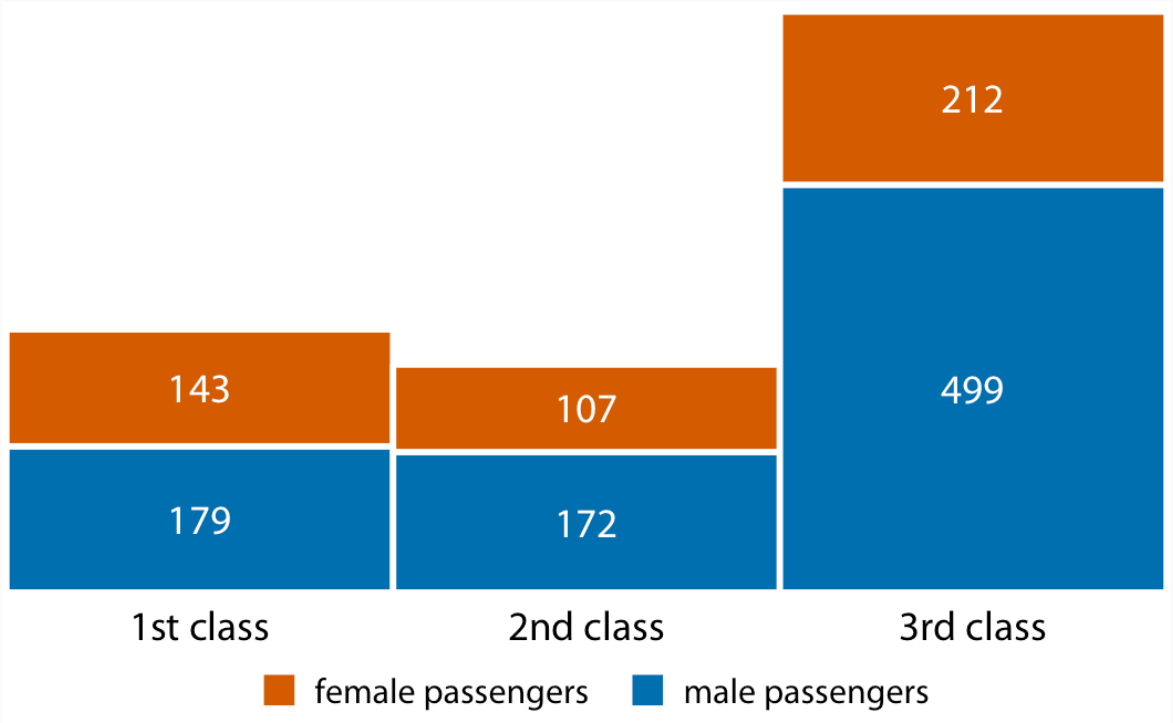
Use unnecesarily *spruced up* **3D** plots...



Misleading

instead, friends encourage...

The prioritization of **content** over apparent aesthetics.



Better

A Short List of Dos and Don'ts

1. Follow the principle of proportional ink.[1]
2. Maximize the data–ink ratio, within reason.
3. Avoid invisible overplotting.
4. Drop all the unimportant stuff.
5. Don't overload graphs. Instead, use several.
6. Use color scales that match the logic of the data scale.
7. Use color-vision deficient-friendly colors.
8. Only use a legend when you need one.
9. Pay attention to legend order.
10. Label axes properly (but avoid trivial information).
11. Use grids and helper lines, within reason.
12. Don't order alphabetically ("Alabama first"). Use natural orders instead.
13. Bar chart axes should include zero.
14. Put the explanatory variable on the x axis, the outcome on the y axis.
15. Axes have canonical directions. Larger values are placed above/right of smaller values.
16. Avoid multiple y axes at all cost.
17. Don't do pie charts. (Or maybe do?)
18. Don't go 3D.
19. Use readable fonts and font sizes.
20. Sometimes a table is just enough.

[1] Follow the links for more information.

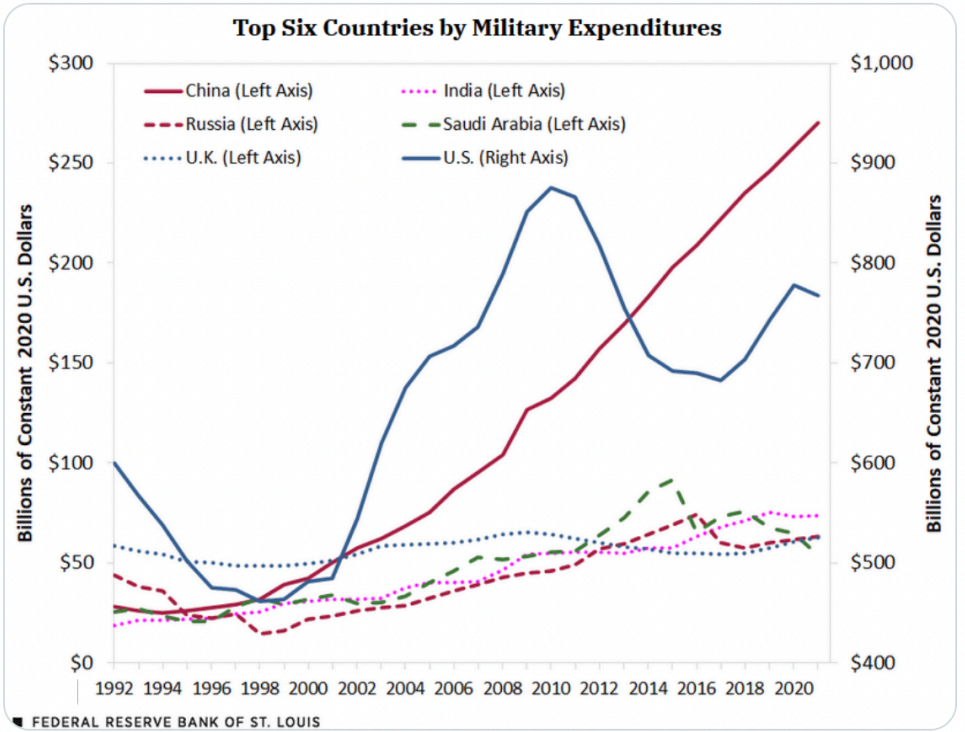
Break out into pairs

Let's take a couple of minutes in pairs:

- **What do you think is the aim of the visualization?**
  - What does the designer want to convey?
  - Are they effective in presenting their message?
- **Do you find any underlying issues with the visualization?**
  - Is this it the best plot type?
  - Are there any features that may mislead the viewer?
- **Can you think of ways to improve the visualization?**

St. Louis Fed
@stlouisfed

An analysis looks at how defense spending among the nations with the highest expenditures has changed since 1992 and what may have driven the changes ow.ly/MyOx50MwEyF

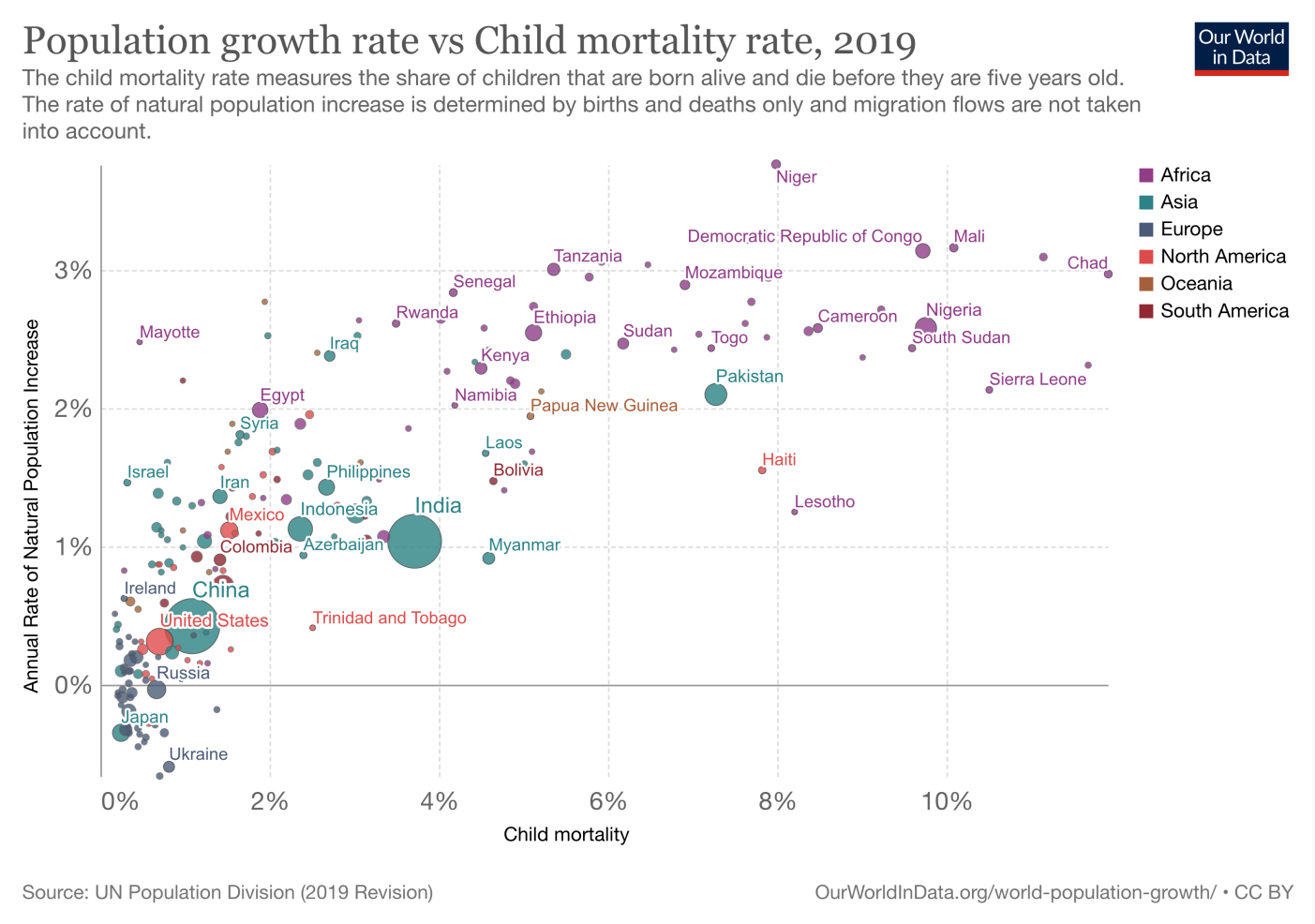


The best statistical graph of all times

The best statistical graph of all times

Hertie School

- "Indeed, among all the forms of statistical graphics, **the humble scatterplot** may be considered the most versatile, polymorphic, and generally useful invention in the entire history of statistical graphics." (Friendly & Denis 2005)
- There's another lesson to learn here: **Keep it simple.** Reading visualizations is a skill, and most people exposed to your work will be worse at it than you.



The best statistical graph of all times

- "Indeed, among all the forms of statistical graphics, **the humble scatterplot** may be considered the most versatile, polymorphic, and generally useful invention in the entire history of statistical graphics." (Friendly & Denis 2005)
- There's another lesson to learn here: **Keep it simple.** Reading visualizations is a skill, and most people exposed to your work will be worse at it than you.

Click- and code-based visualization tools

Click- and code-based visualization tools

Click-based

These tools are like *easy-to-use apps* where you don't need to know how to code. You simply *click buttons*, *drag elements*, and *select options to create visualizations*. They're great if you want to quickly make charts and graphs without getting into the nitty-gritty of coding. Think of it like using a paint-by-numbers kit where you fill in colors without needing to mix them yourself.

Code-based

These are like toolkits for *artists who prefer to paint from scratch*. Instead of clicking buttons, you write *instructions in a programming language* like `Python` or `R` to create your visualizations. This gives you a lot of **flexibility** and **control** over every detail of your chart or graph. It's like being able to mix your own colors and create any kind of painting you want, but it requires learning a bit of programming.

Click-based visualization tools

Advantages

1. **Ease of Use:** Click-based tools like Tableau or Power BI offer *intuitive interfaces* that make it easy for users to create visualizations without any programming knowledge.

2. **Rapid Prototyping:** Users can *quickly prototype* and iterate on visualizations using click-based tools, enabling faster exploration of data and insights.

3. **Interactive Features:** Click-based tools often come with *built-in interactive features* that allow users to explore data dynamically, such as filtering, drilling down, or zooming in on specific data points.

4. **Pre-built Templates:** Click-based tools often provide a variety of *pre-built templates* and visualizations, saving users time and effort in design and formatting.

Disadvantages

1. **Limited Customization:** Click-based tools may have limitations in terms of customization compared to code-based tools, *restricting* users' ability to create highly tailored visualizations.

2. **Scalability Issues:** Click-based tools may *struggle with large datasets* or complex visualizations, leading to performance issues or limitations in functionality.

3. **Vendor Lock-in:** Users may become dependent on specific click-based tools, leading to vendor lock-in and potential *limitations in flexibility or compatibility* with other tools.

4. **Cost:** Click-based tools often come with *licensing fees or subscription costs*, which can be prohibitive for individuals or organizations with budget constraints.

Code-based visualization tools

Advantages

1. **Flexibility:** With code-based tools like `ggplot2`, users have *full control* over customization, allowing them to create highly customized and intricate visualizations.

2. **Reproducibility:** Code-based tools facilitate reproducibility since the code can be shared and reused easily, ensuring *consistent results across different environments*.

3. **Scalability:** These tools are often more suitable for handling *large datasets and complex visualizations*, as they offer more advanced programming capabilities.

4. **Integration:** Code-based tools can be seamlessly *integrated into data analysis workflows*, allowing for direct manipulation of data and integration with statistical analysis.

Disadvantages

1. **Learning Curve:** Users without programming experience may find it *challenging to learn* the syntax and concepts required to use code-based visualization tools effectively.

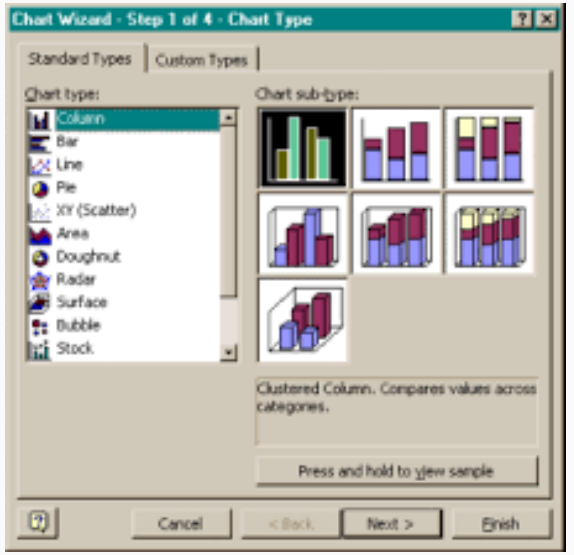
2. **Time-Consuming:** Creating visualizations from scratch using code can be *time-consuming*, especially for users who are not proficient in programming.

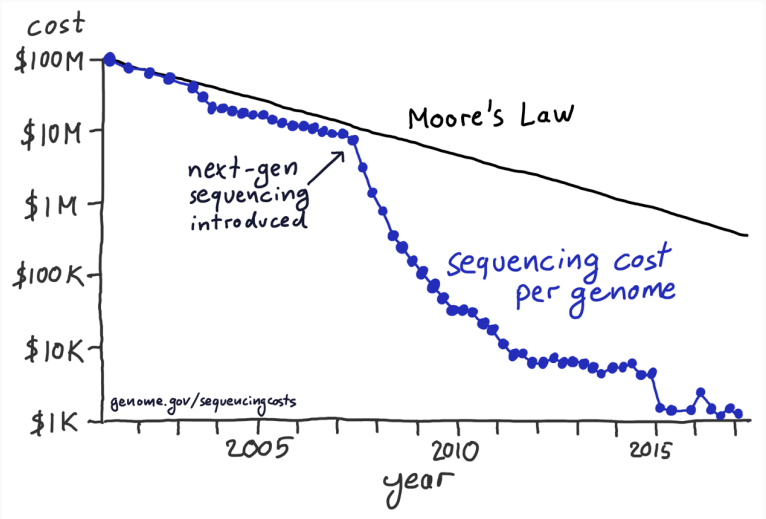
3. **Debugging Complexity:** Debugging code-based visualizations can be more *complex* compared to click-based tools, especially when dealing with intricate plots or errors in the code.

4. **Maintenance:** Code-based visualizations may require more *maintenance and updates over time*, especially when libraries or dependencies change.

Some notes on how to get visualization right

1. **Always conceptualize first**.
2. Prioritize programmatic solutions to stay reproducible (i.e., `R` over Excel).
3. For conceptual charts (not: data viz!), other tools might be just fine (e.g., Powerpoint or even hand-drawn figures).
4. Don't be distracted by interactives (as offered by, e.g., highcharts.com, Tableau, and others).
5. Designing good graphs is a learnable skill. Study how others do it in your software of choice!
6. But a good visualization takes time, even if you're experienced. Working a full day on the key plot of your analysis? No problem! If you go the coding route, this is a good investment for Future-You.

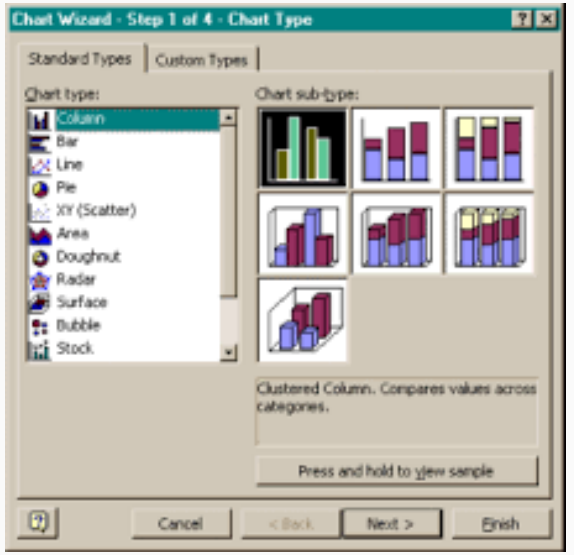


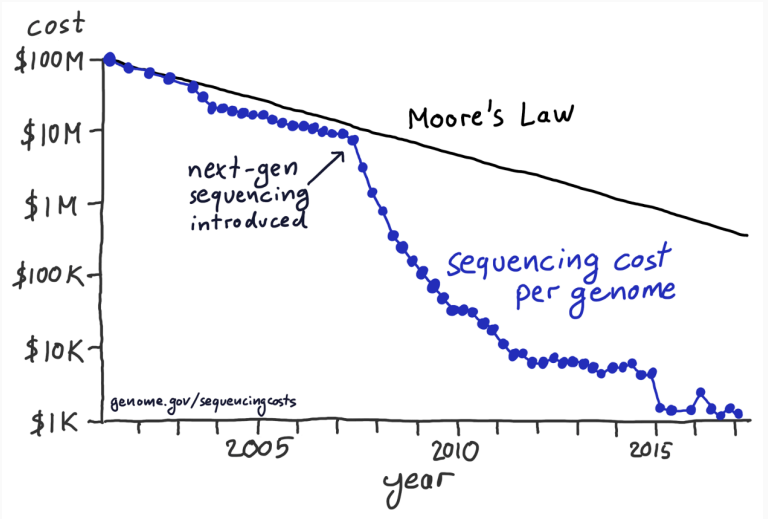


Some notes on how to get visualization right

1. **Always conceptualize first**.

2. **Prioritize programmatic solutions to stay reproducible** (i.e., `R` over Excel).



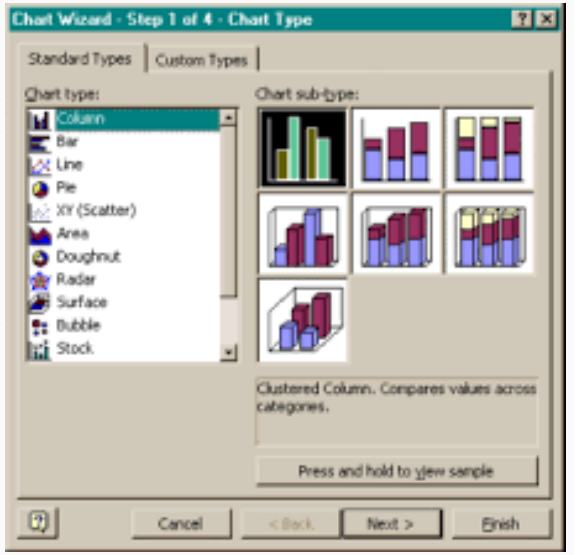


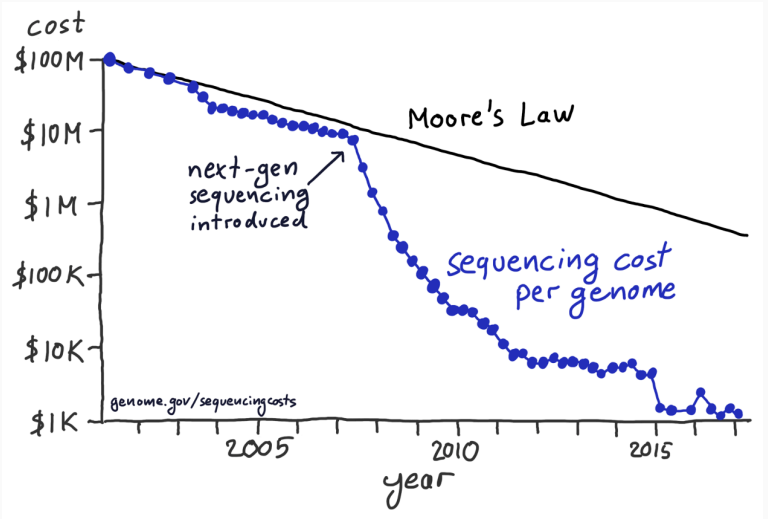
Some notes on how to get visualization right

1. **Always conceptualize first**.

2. Prioritize programmatic solutions to stay reproducible (i.e., `R` over Excel).

3. For conceptual charts (not: data viz!), **other tools might be just fine** (e.g., Powerpoint or even hand-drawn figures).





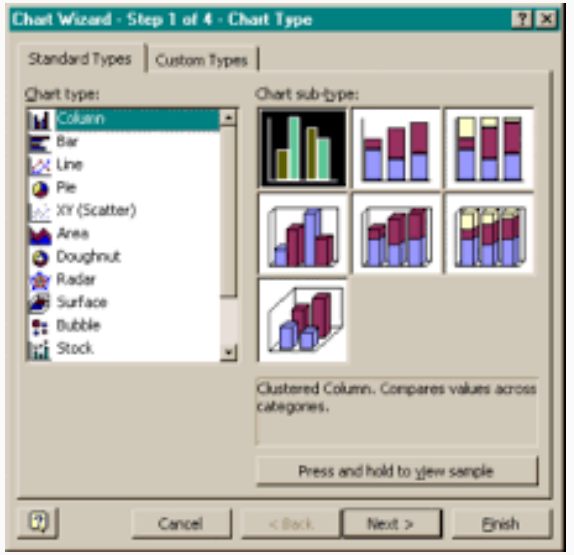
Some notes on how to get visualization right

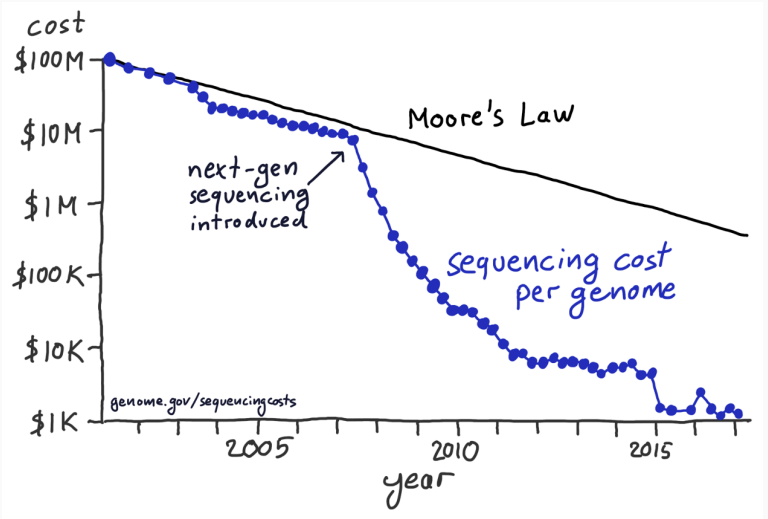
1. **Always conceptualize first**.

2. Prioritize programmatic solutions to stay reproducible (i.e., `R` over Excel).

3. For conceptual charts (not: data viz!), other tools might be just fine (e.g., Powerpoint or even hand-drawn figures).

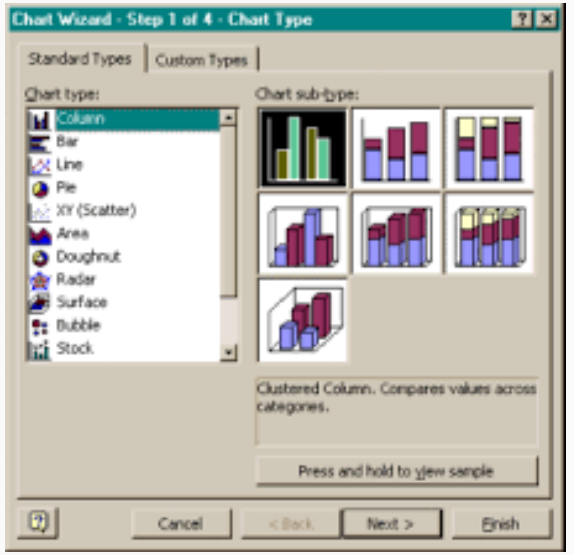
4. **Don't be distracted by interactives** (as offered by, e.g., highcharts.com, Tableau, and others).

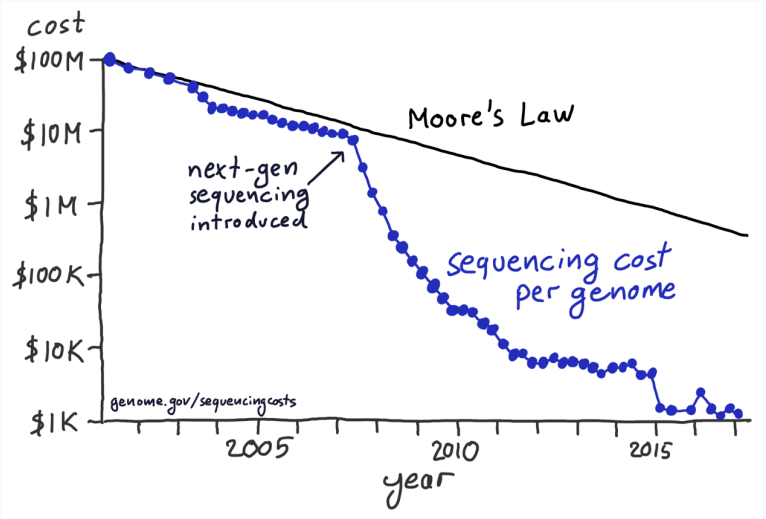




Some notes on how to get visualization right

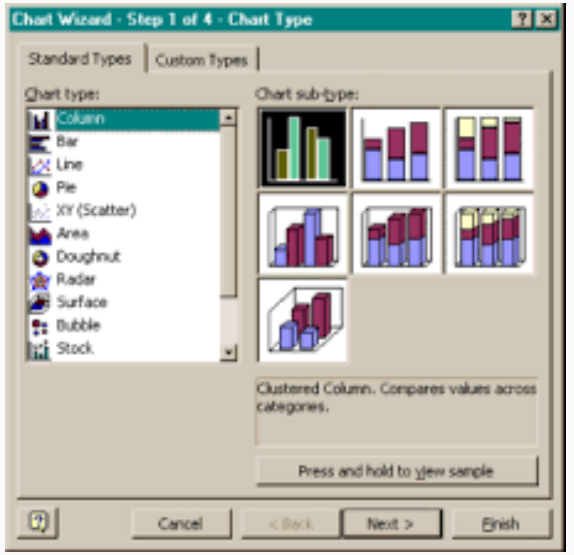
1. **Always conceptualize first**.
2. Prioritize programmatic solutions to stay reproducible (i.e., `R` over Excel).
3. For conceptual charts (not: data viz!), other tools might be just fine (e.g., Powerpoint or even hand-drawn figures).
4. Don't be distracted by interactives (as offered by, e.g., highcharts.com, Tableau, and others).
5. Designing good graphs is a **learnable skill**. Study how others do it in your software of choice!

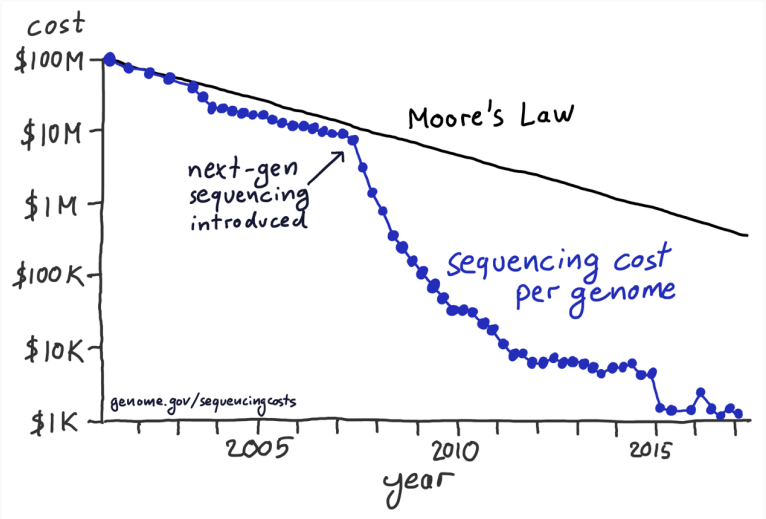




Some notes on how to get visualization right

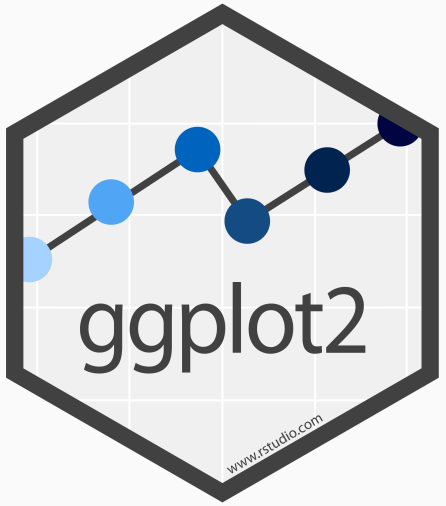
1. **Always conceptualize first**.
2. Prioritize programmatic solutions to stay reproducible (i.e., `R` over Excel).
3. For conceptual charts (not: data viz!), other tools might be just fine (e.g., Powerpoint or even hand-drawn figures).
4. Don't be distracted by interactives (as offered by, e.g., highcharts.com, Tableau, and others).
5. Designing good graphs is a learnable skill. Study how others do it in your software of choice!
6. But **a good visualization takes time**, even if you're experienced. Working a full day on the key plot of your analysis? No problem! If you go the coding route, this is a good investment for Future-You.





Visualization with R and ggplot2

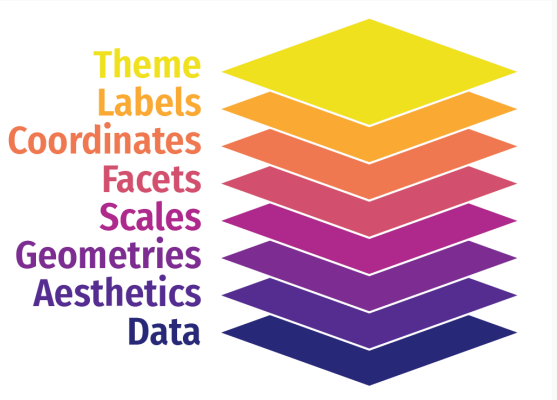
`ggplot2` is a system for declaratively creating graphics, based on *The Grammar of Graphics* (Leland Wilkinson). You provide the data, tell `ggplot2` how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. It is part of the tidyverse.



The grammar of graphics

The easy logic of graphical grammar

- Coding a graphic requires us to **describe each element** of it very precisely.

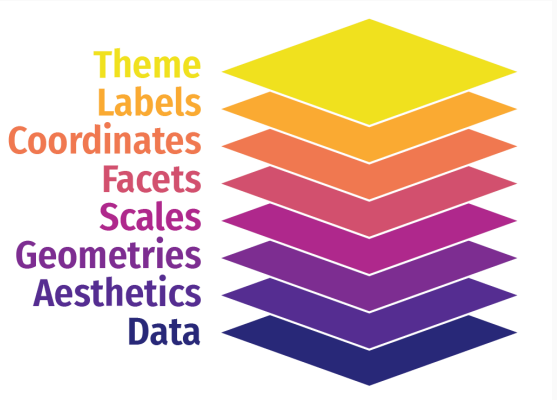


Grammar components as layers in ggplot2

The grammar of graphics

The easy logic of graphical grammar

- Coding a graphic requires us to describe each element of it very precisely.
- To do so, `ggplot` and its creator Hadley Wickham build on and extend a particular grammar - the **"Grammar of Graphics"** by Wilkinson, Anand, and Grossmann.

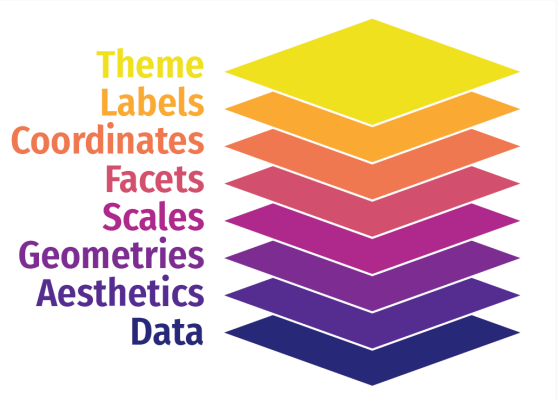


Grammar components as layers in ggplot2

The grammar of graphics

The easy logic of graphical grammar

- Coding a graphic requires us to describe each element of it very precisely.
- To do so, `ggplot` and its creator Hadley Wickham build on and extend a particular grammar - the "Grammar of Graphics" by Wilkinson, Anand, and Grossmann.
- Building plots with `ggplot2` emphasizes the **concept of layers**, which are specified function by function and assembled with `+`.

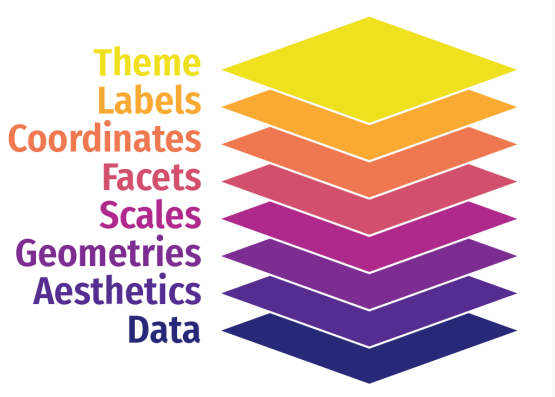


Grammar components as layers in ggplot2

The grammar of graphics

The easy logic of graphical grammar

- Coding a graphic requires us to describe each element of it very precisely.
- To do so, `ggplot` and its creator Hadley Wickham build on and extend a particular grammar - the "Grammar of Graphics" by Wilkinson, Anand, and Grossmann.
- Building plots with `ggplot2` emphasizes the **concept of layers**, which are specified function by function and assembled with `+`.
- With this particular grammar, we talk less about chart **types** and more about specific chart **elements**.



Grammar components as layers in ggplot2

The grammar of graphics

The easy logic of graphical grammar

- Coding a graphic requires us to describe each element of it very precisely.
- To do so, `ggplot` and its creator Hadley Wickham build on and extend a particular grammar - the "Grammar of Graphics" by Wilkinson, Anand, and Grossmann.
- Building plots with `ggplot2` emphasizes the **concept of layers**, which are specified function by function and assembled with `+`.
- With this particular grammar, we talk less about chart *types* and more about specific chart *elements*.

- For instance, **we don't say**:

  "R, give me a small multiple scatter plot."

Grammar components as layers in ggplot2

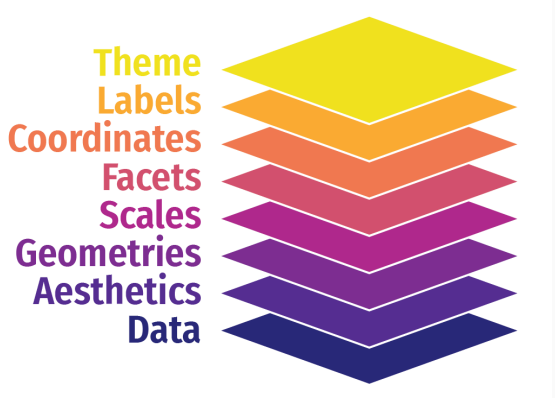
The grammar of graphics

The easy logic of graphical grammar

- Coding a graphic requires us to describe each element of it very precisely.
- To do so, `ggplot` and its creator Hadley Wickham build on and extend a particular grammar - the "Grammar of Graphics" by Wilkinson, Anand, and Grossmann.
- Building plots with `ggplot2` emphasizes the **concept of layers**, which are specified function by function and assembled with `+`.
- With this particular grammar, we talk less about chart *types* and more about specific chart *elements*.
- For instance, we don't say:

  "R, give me a small multiple scatter plot."
- Instead, **we say**:

  "Using this dataset, map wealth to the x-axis, health to the y-axis, add points, color by continent, size by population, scale the y-axis with a log, and facet by year."



Grammar components as layers in ggplot2

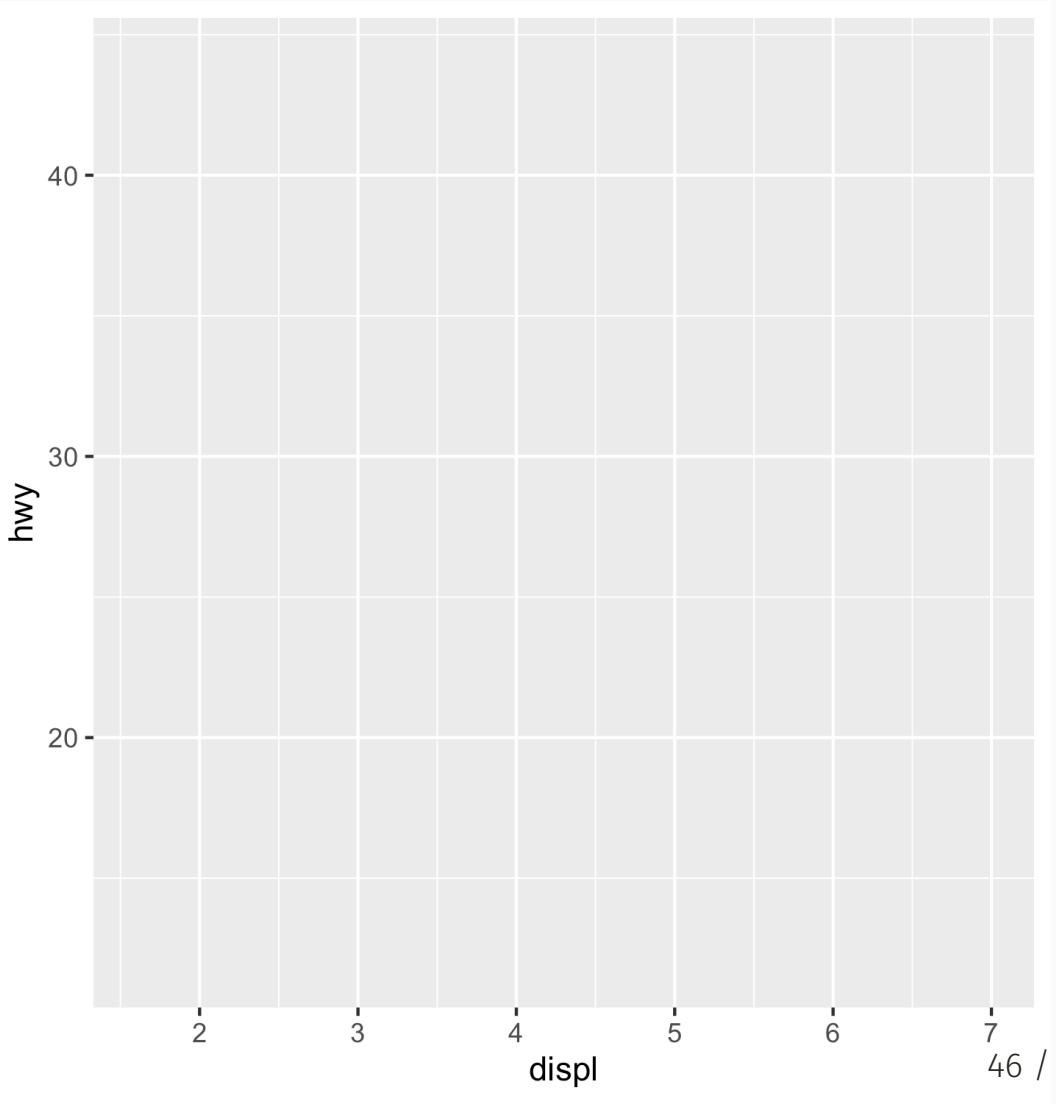
Components of grammar of graphics

Element	Description	Essential?	Example
Data	This is the dataset being plotted containing the variables to be plotted on the graph.	Yes	`ggplot(data = mpg)`
Aesthetics	Aesthetics refers to the scales on which we map our data. Some common aesthetics to consider are axis (x,y), shape, size and color.	Yes	`aes(x = displ, y = hwy, color = drv)`
Geometries	Geom refers to the actual visual elements used for the data in the plot, such as points, lines, and bars.	Yes	`geom_point()`
Scales	Scales map data values to the visual values of an aesthetic. This can be used to change a default mapping	No	`scale_colour_manual(values = c("red", "blue", "green"))`
Facets	Faceting refers to splitting the data into multiple subsets and then displaying plots for the specific subsets in a panel (small multiples).	No	`facet_grid(vars(drv), ncol = 1)`
Statistics	This refers to representing statistical information about the data, such as mean and variance, to help in understanding the data.	No	`stat_count(geom="bar")`
Coordinates	This refers to the space on which the data is plotted (e.g., Cartesian coordinates).	No	`coord_polar()`
Labels	This refers to additional descriptions of your plot, such as title, subtitle, caption, x and y axis label	No	`labs(x = "Height", y = "Weight", title = "Look at my plot")`
Themes	Themes are used to change the appearance of non-data elements, such as fonts, color, or legends.	No	`theme_bw()`

Building a plot step by step with ggplot2

Start with data and aesthetics[1]

```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv))
```

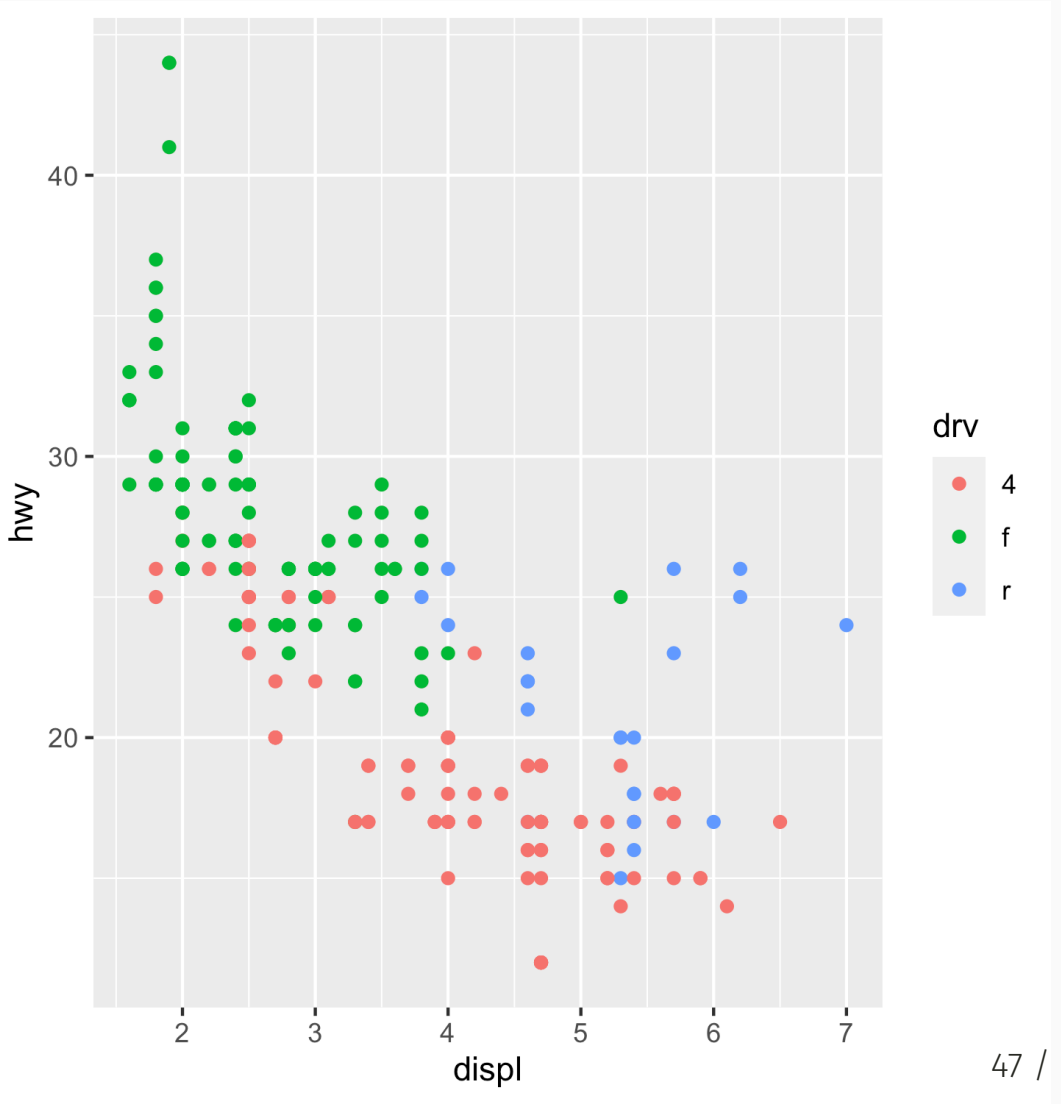


[1]This example is borrowed from Andrew Heiss.

Building a plot step by step with ggplot2 (cont.)

Add a point geom

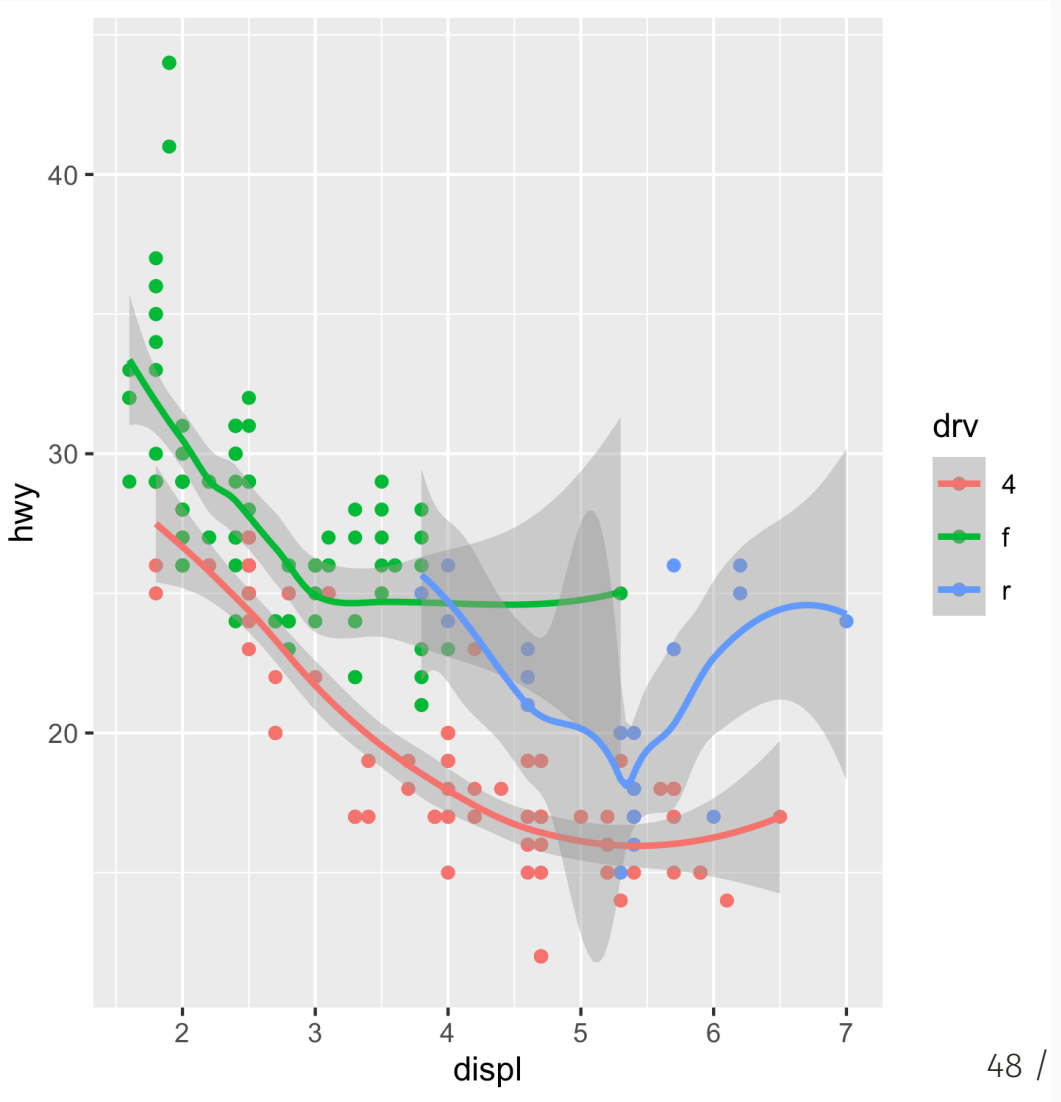
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+   geom_point()
```



Building a plot step by step with ggplot2 (cont.)

Add a smooth geom

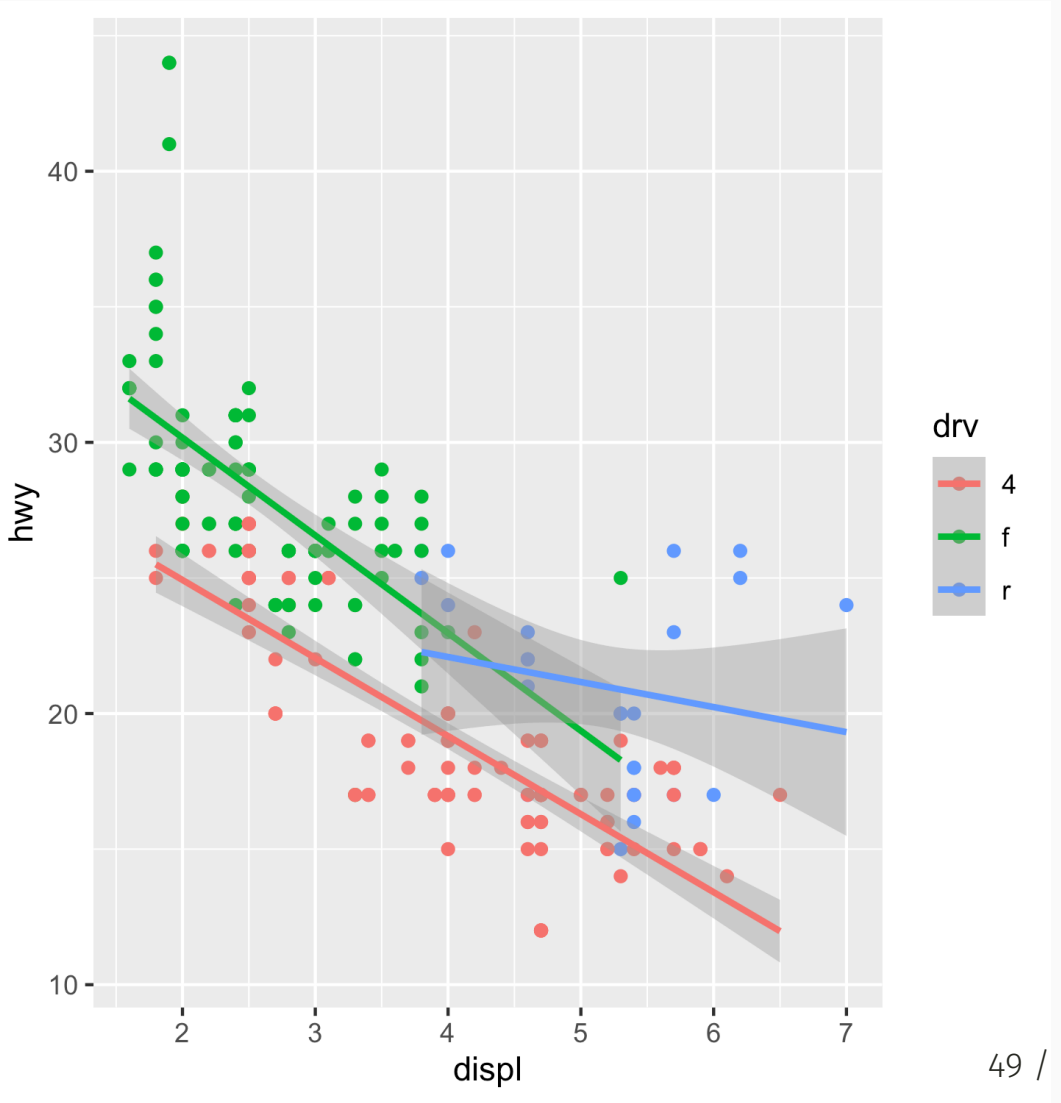
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth()
```



Building a plot step by step with ggplot2 (cont.)

Make it straight

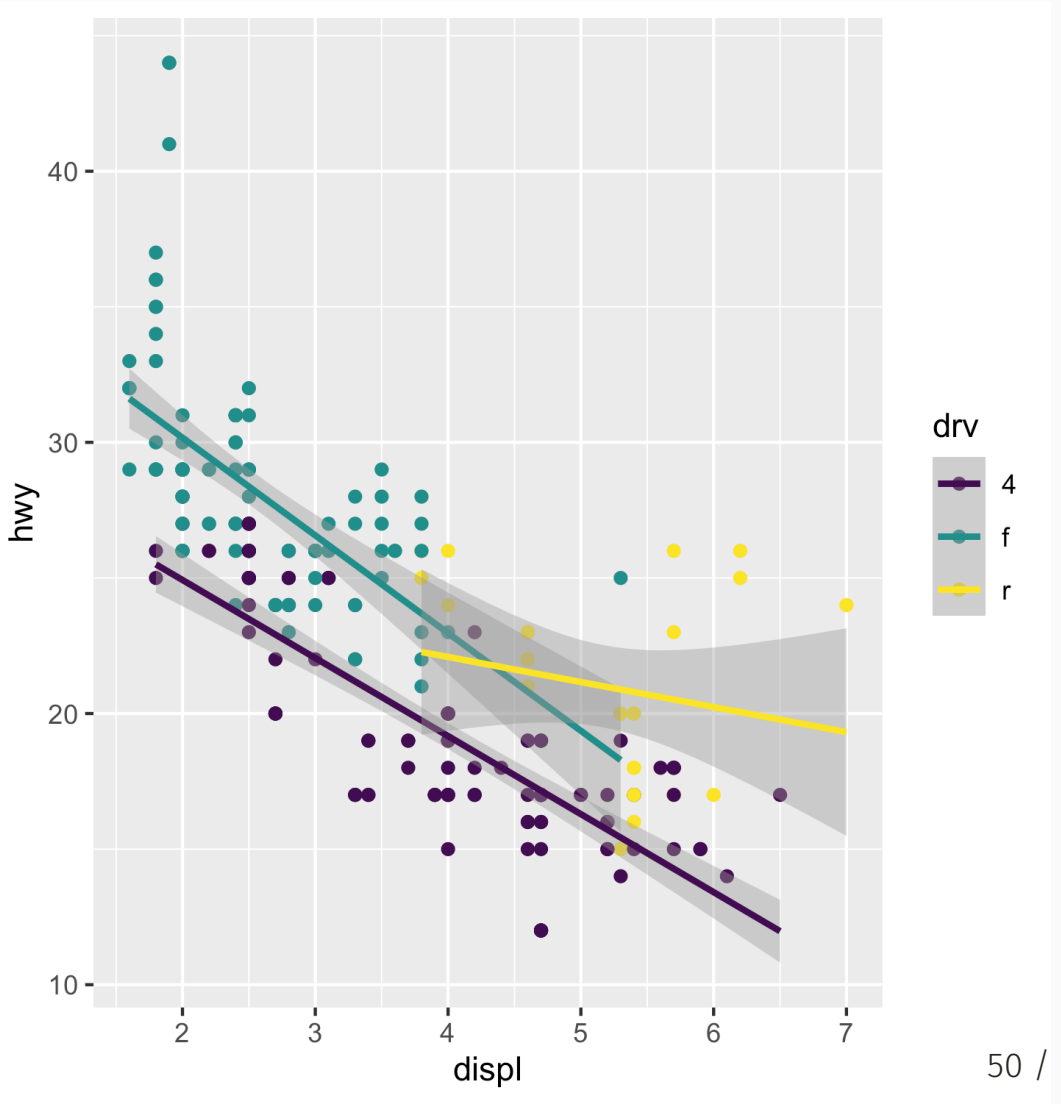
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm")
```



Building a plot step by step with ggplot2 (cont.)

Use a viridis color scale

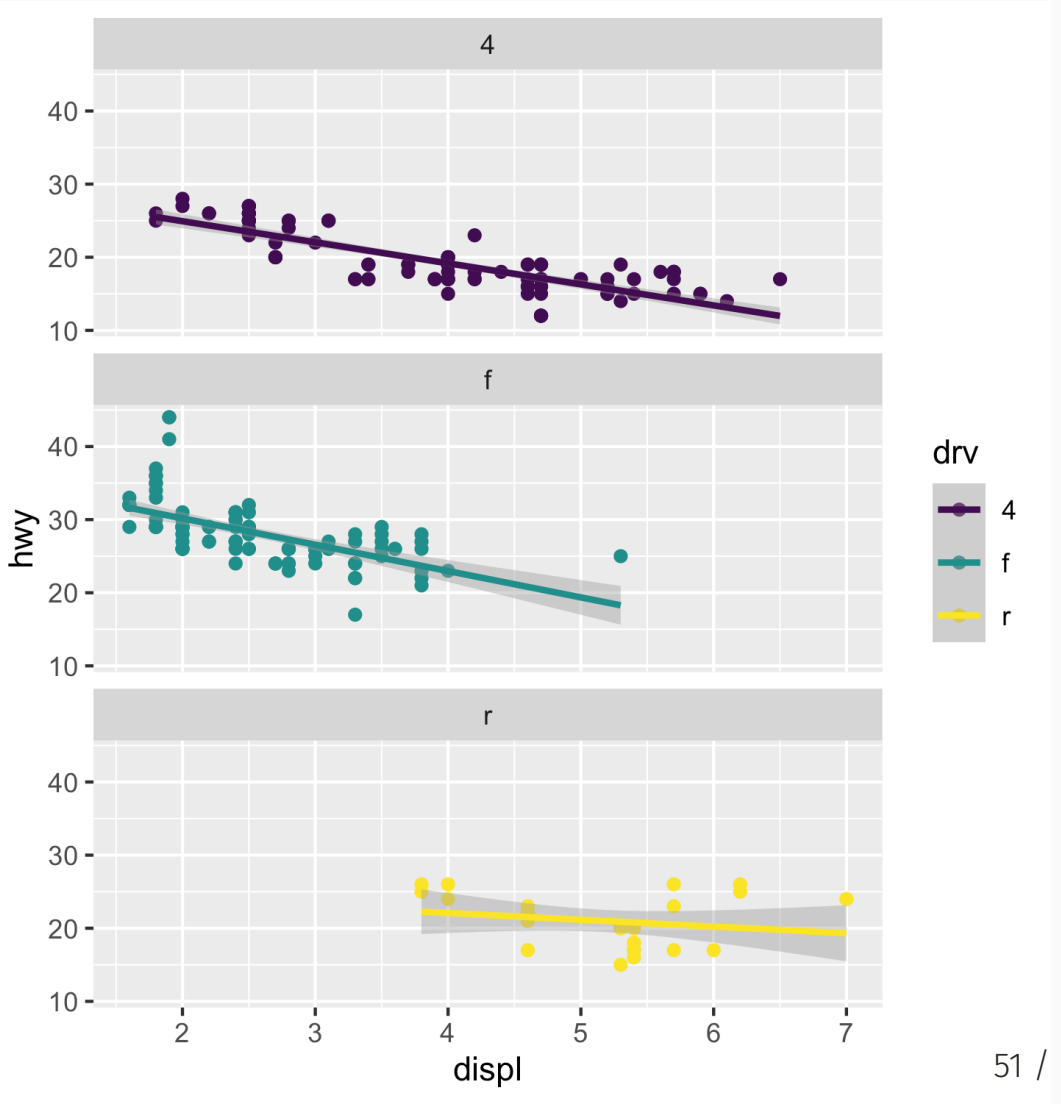
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm") +
+    scale_color_viridis_d()
```



Building a plot step by step with ggplot2 (cont.)

Facet by drive

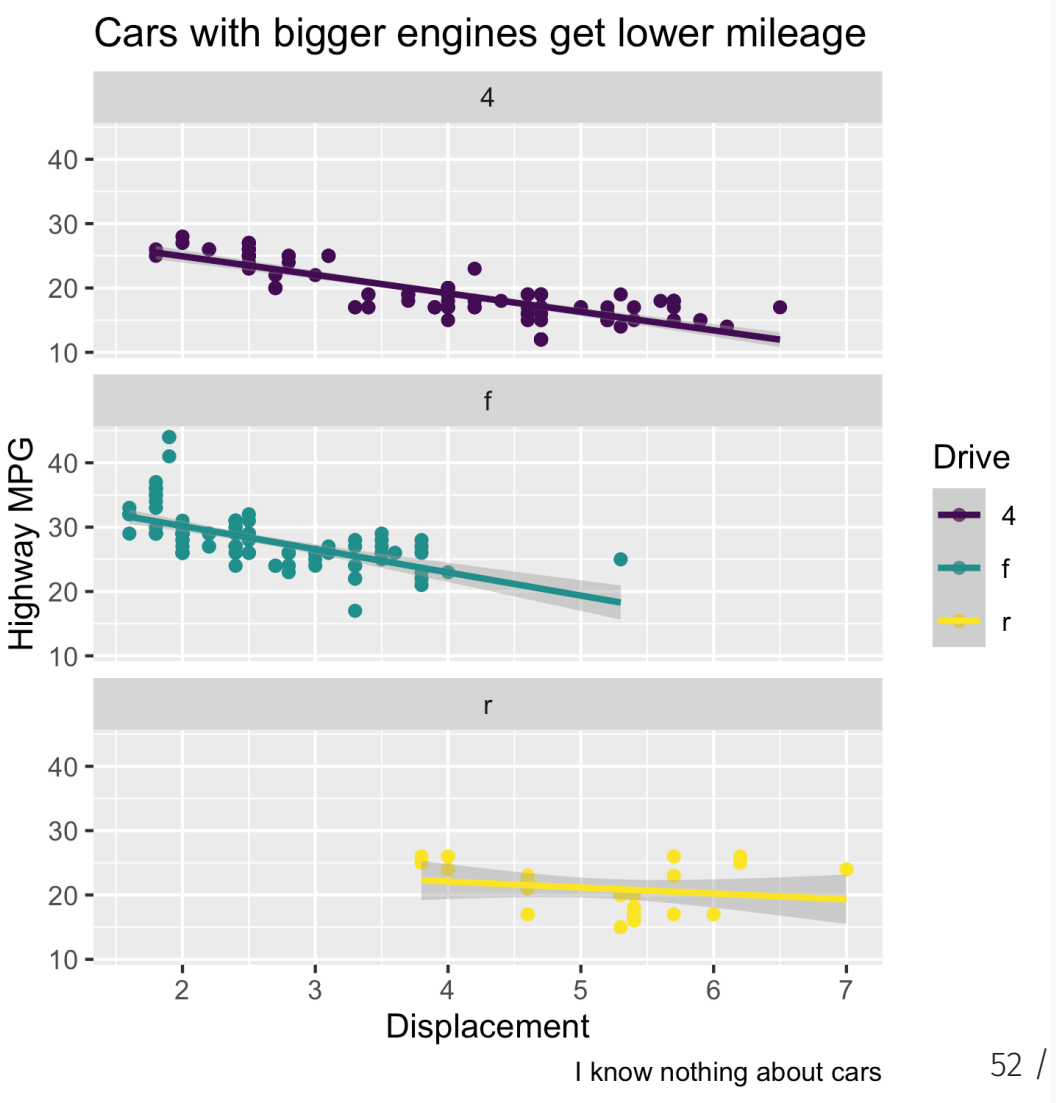
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm") +
+    scale_color_viridis_d() +
+    facet_wrap(vars(drv), ncol = 1)
```



Building a plot step by step with ggplot2 (cont.)

Add labels

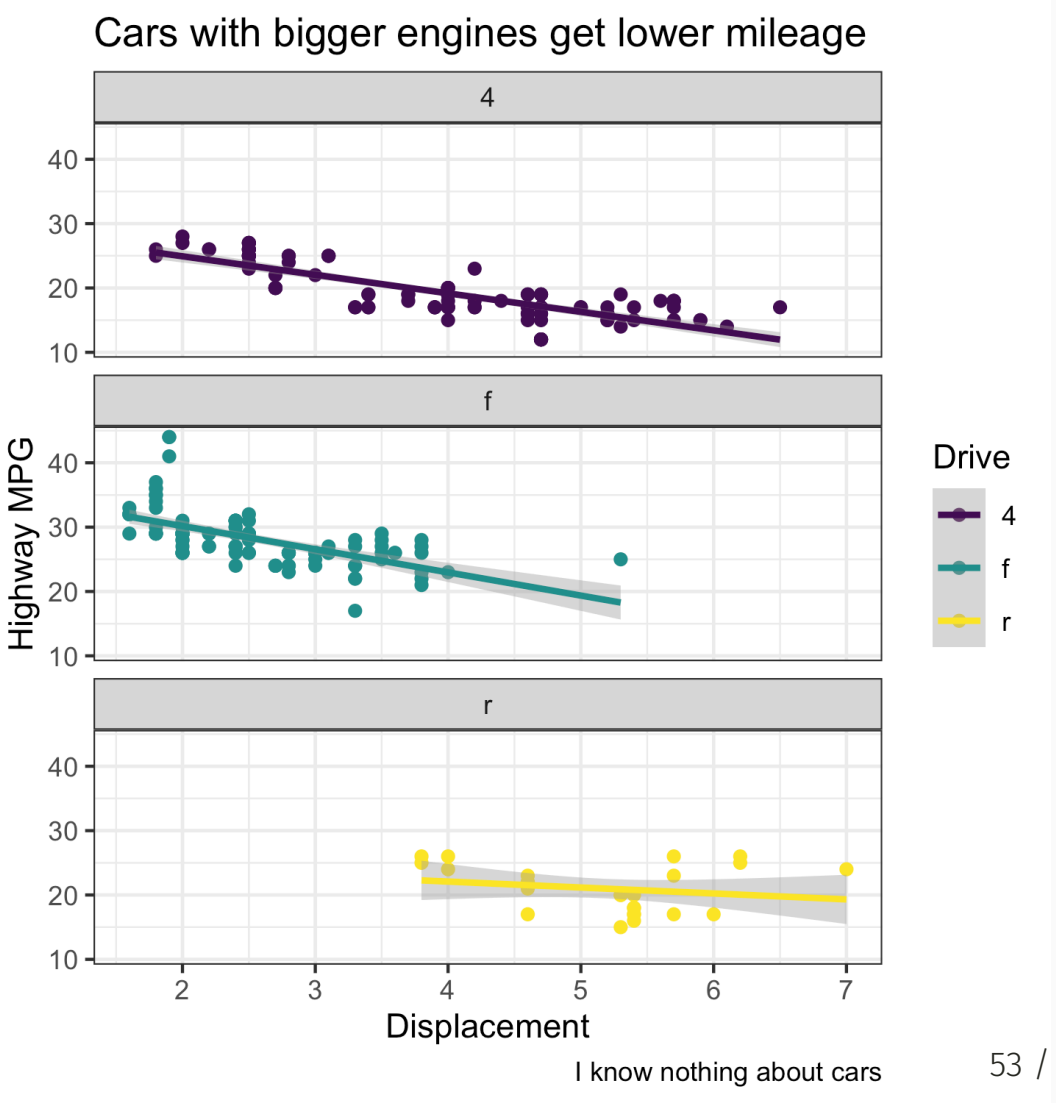
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm") +
+    scale_color_viridis_d() +
+    facet_wrap(vars(drv), ncol = 1) +
+    labs(x = "Displacement", y = "Highway MPG",
+         color = "Drive",
+         title = "Cars with bigger engines get lower
+         caption = "I know nothing about cars")
```



Building a plot step by step with ggplot2 (cont.)

Add a theme

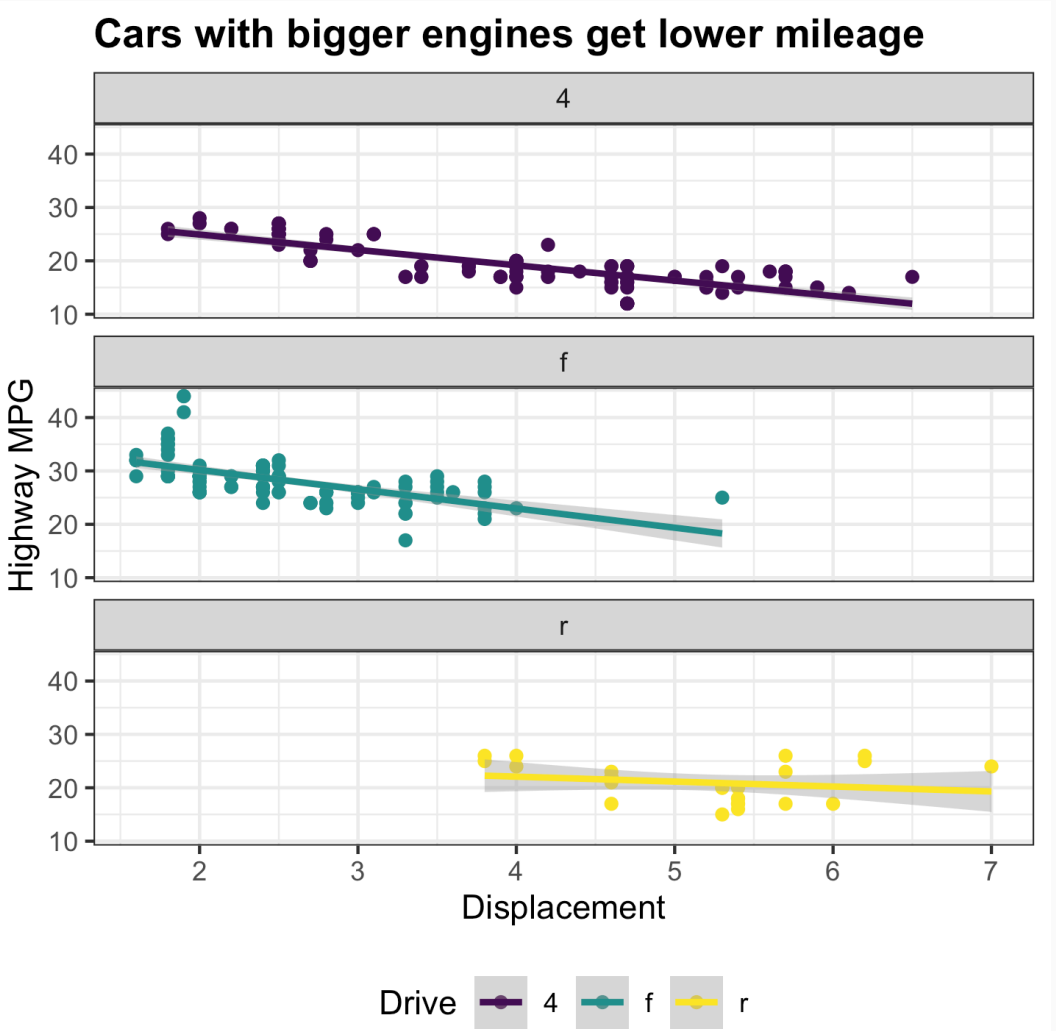
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm") +
+    scale_color_viridis_d() +
+    facet_wrap(vars(drv), ncol = 1) +
+    labs(x = "Displacement", y = "Highway MPG",
+         color = "Drive",
+         title = "Cars with bigger engines get lower
+         caption = "I know nothing about cars") +
+    theme_bw()
```



Building a plot step by step with ggplot2 (cont.)

Modify the theme

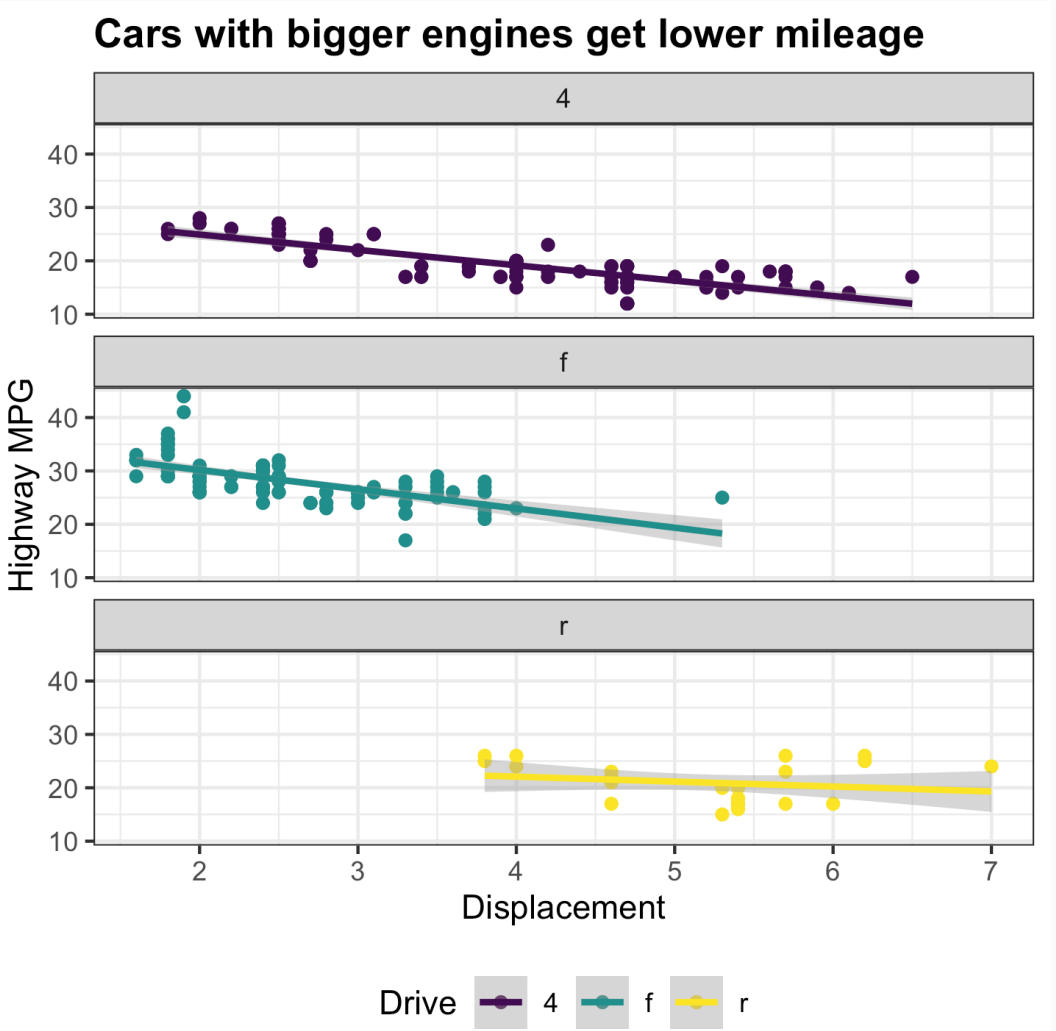
```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm") +
+    scale_color_viridis_d() +
+    facet_wrap(vars(drv), ncol = 1) +
+    labs(x = "Displacement", y = "Highway MPG",
+         color = "Drive",
+         title = "Cars with bigger engines get lower
+         caption = "I know nothing about cars") +
+    theme_bw() +
+    theme(legend.position = "bottom",
+          plot.title = element_text(face = "bold"))
```



Building a plot step by step with ggplot2 (cont.)

Finished!

```
R> ggplot(data = mpg,
+         mapping = aes(x = displ,
+                       y = hwy,
+                       color = drv)) +
+    geom_point() +
+    geom_smooth(method = "lm") +
+    scale_color_viridis_d() +
+    facet_wrap(vars(drv), ncol = 1) +
+    labs(x = "Displacement", y = "Highway MPG",
+         color = "Drive",
+         title = "Cars with bigger engines get lower
+         caption = "I know nothing about cars") +
+    theme_bw() +
+    theme(legend.position = "bottom",
+          plot.title = element_text(face = "bold"))
```



Picking the right image file format

What are the formats?

- At the end of the visualization workflow, you have to decide how to store/export/publish the figures.
- There are many different file formats, but the most important difference is whether they are:
  - **Bitmap (raster graphics)**: store information as a grid of pixels
  - **Vector**: store information as geometric arrangement of individual graphical elements

Which format to pick?

- Use `pdf/svg` whenever possible.
- Use `png` for online documents/presentations.
- Use `jpeg` as last resort (in particular if pngs are too large and loss by compression is acceptable).

Table 27.1: Commonly used image file formats

Acronym	Name	Type	Application
pdf	Portable Document Format	vector	general purpose
eps	Encapsulated PostScript	vector	general purpose, outdated; use pdf
svg	Scalable Vector Graphics	vector	online use
png	Portable Network Graphics	bitmap	optimized for line drawings
jpeg	Joint Photographic Experts Group	bitmap	optimized for photographic images
tiff	Tagged Image File Format	bitmap	print production, accurate color reproduction
raw	Raw Image File	bitmap	digital photography, needs post-processing
gif	Graphics Interchange Format	bitmap	outdated for static figures, Ok for animations

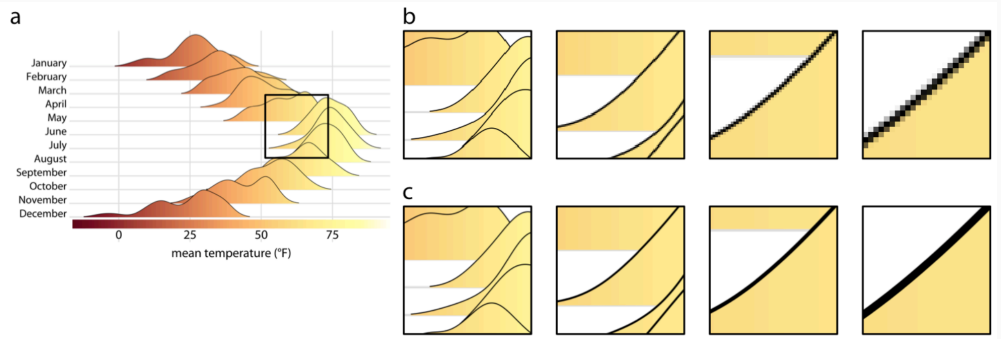


Figure 27.1: Illustration of the key difference between vector graphics and bitmaps. (a) Original image. The black square indicates the area we are magnifying in parts (b) and (c). (b) Increasing magnification of the highlighted area from part (a) when the image has been stored as a bitmap graphic. We can see how the image becomes increasingly pixelated as we zoom in further. (c) Increasing magnification of a vector representation of the image. The image maintains perfect sharpness at arbitrary magnification levels.

Why go through the trouble of programmatic solutions?

Hertie School

tidytuesday.rocks

Tidy Tuesday (https://github.com/rfordatascience/tidytuesday) is a weekly data visualization challenge for R users hosted on Twitter with #TidyTuesday.

As of January 21, 2021, there are

- 7,117 tweets from
- 1,508 users across
- 147 datasets!

Use the options below to filter the tweets by dataset or user and sort them by likes, retweets, and when they were posted. Happy plotting! 🏌️‍♀️

(Made by @nsgrantham (https://twitter.com/nsgrantham). Source code on GitHub (https://github.com/nsgrantham/tidytuesdayrocks).)

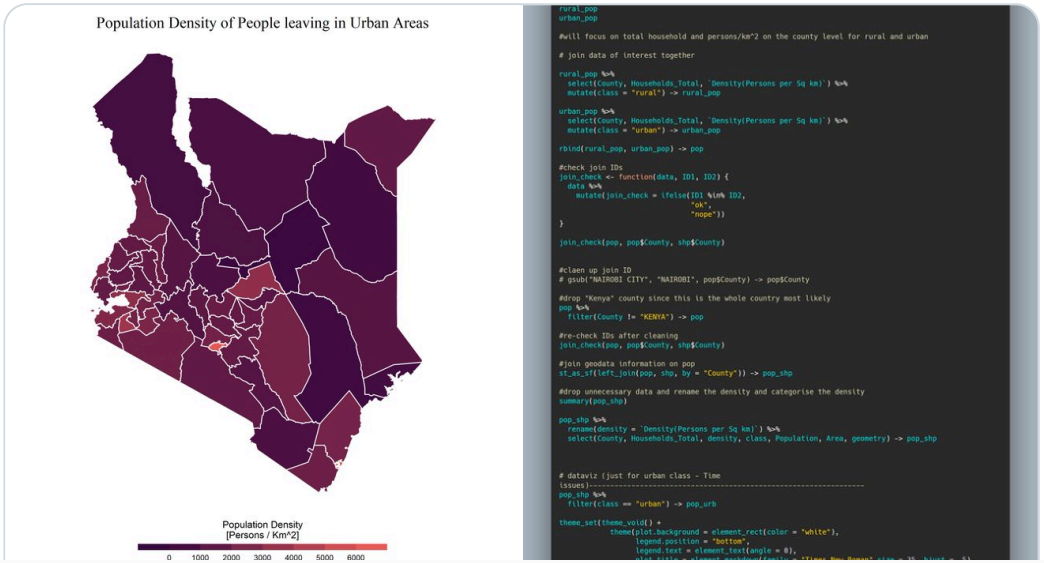
Dataset tweets | User tweets

Tobias Stalder
@toeb18 · **Follow**

Wanted to make something fancy for this week's #TidyTuesday but ran out of time. So I'm semi-proud to present this map which is definitely a compromise =P. Hope you enjoy. Done in #Rstats #tidyverse #ggplot2.

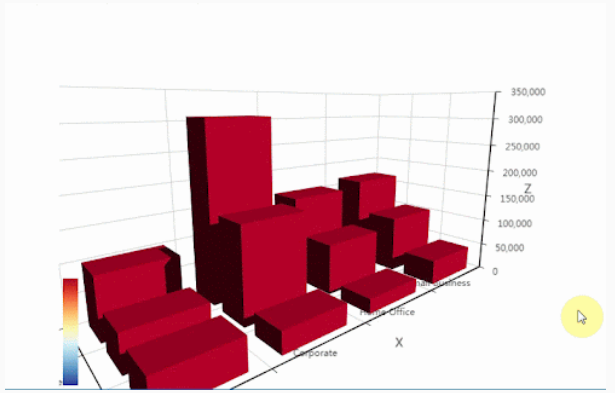
#dataviz #rspatial #gischat

code: git.io/Jtm6L



Going interactive

- In online presentations of your data and analyses, you might want to go interactive.
- Interactive webpages are mainly run with JavaScript, and we can use `R` to draw on JavaScript libraries to create interactive content.
- In particular, the `htmlwidgets` package provides a framework to bind `R` commands to various JavaScript libraries, including those that create data graphs.
- Many "widgets" are already available - check out http://gallery.htmlwidgets.org/.
- Other JavaScript libraries for interactive graphics can be created with R, too:
  - `leaflet` to connect to the Leaflet library and create interactive maps
  - `plotly` to connect to Plotly and create graphs of all kinds
- For the record: interactive ≠ animated. On the right you see animated charts. Please don't do this at home.



(Let's give data visualization a try in `R`)