Lecture 5: Intro to Data Visualization in R

James Sears*
AFRE 891 SS 24
Michigan State University

*Parts of these slides are adapted from "Advanced Data Analytics" by Nick Hagerty and "Data Science for Economists" by Grant McDermott, used under CC BY-NC-SA 4.0.

Table of Contents

This Lecture:

Next Lecture:

Prologue

Prologue

Packages we'll use for today's examples:

```
pacman::p_load(dslabs, gapminder, tidyverse)
```

Additional packages if you want to replicate the plots in the "Principles of Data Visualization" section:

```
pacman::p_load(ggrepel, readxl)
```

Data Visualization

Being able to **visualize data well** is a critical skill for economists and data scientists.

- Effectively communicate a message
- Engage a reader/audience
- Discover patterns in the underlying data

To do this in R, we're going to use the incredibly powerful **ggplot2** package, but before we do, let's chat about...

Principles of Data Visualization

Principles of Data Visualization

We're going to focus on the **punchline** of the principles of data visualization today.

If you want more of the **setup**, there are excellent resources available:

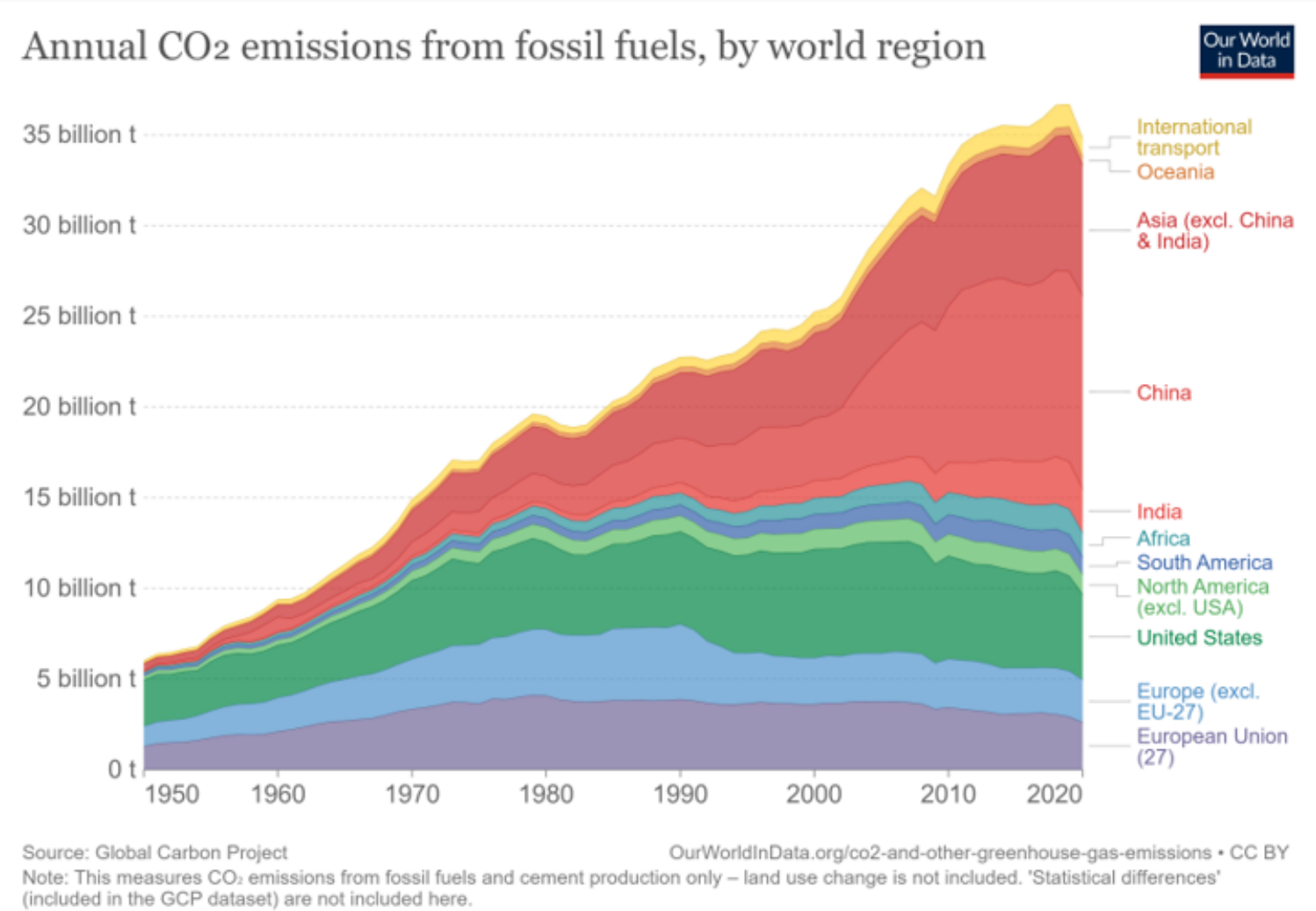
- **Introduction to Data Science** by Rafael A. Irizarry
- **Data Visualization a Practical Introduction** by Kieran Healy
- **"Creating Effective Figures and Tables" talk** by Karl Broman
- **An Economist's Guide to Visualizing Data** by Jonathan A. Schwabish (JPE 2014)
- **Modern Data Science with R, 2nd Ed.** by Baumer, Kaplan, and Horton
- **from Data to Viz (finding the appropriate graph)**
- **R Graph Gallery**

Effective Charts

An **effective graph or chart**

- Conveys information clearly
- Summarizes data quickly
- Helps identify salient features or patterns
- Conveys complex relationships in easy-to-communicate visuals

Effective Charts

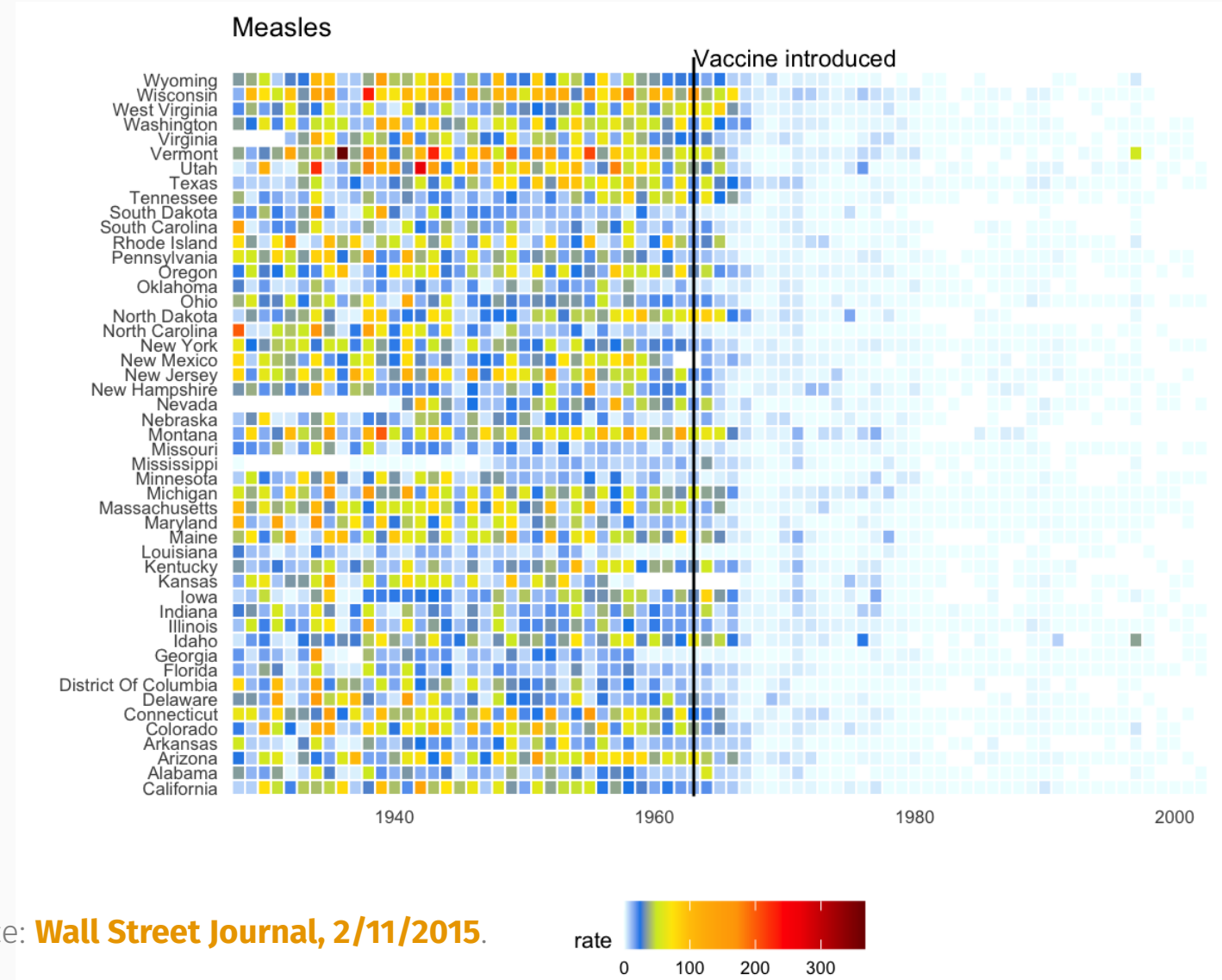


Poorly Constructed Charts

On the other hand, **poorly constructed charts**

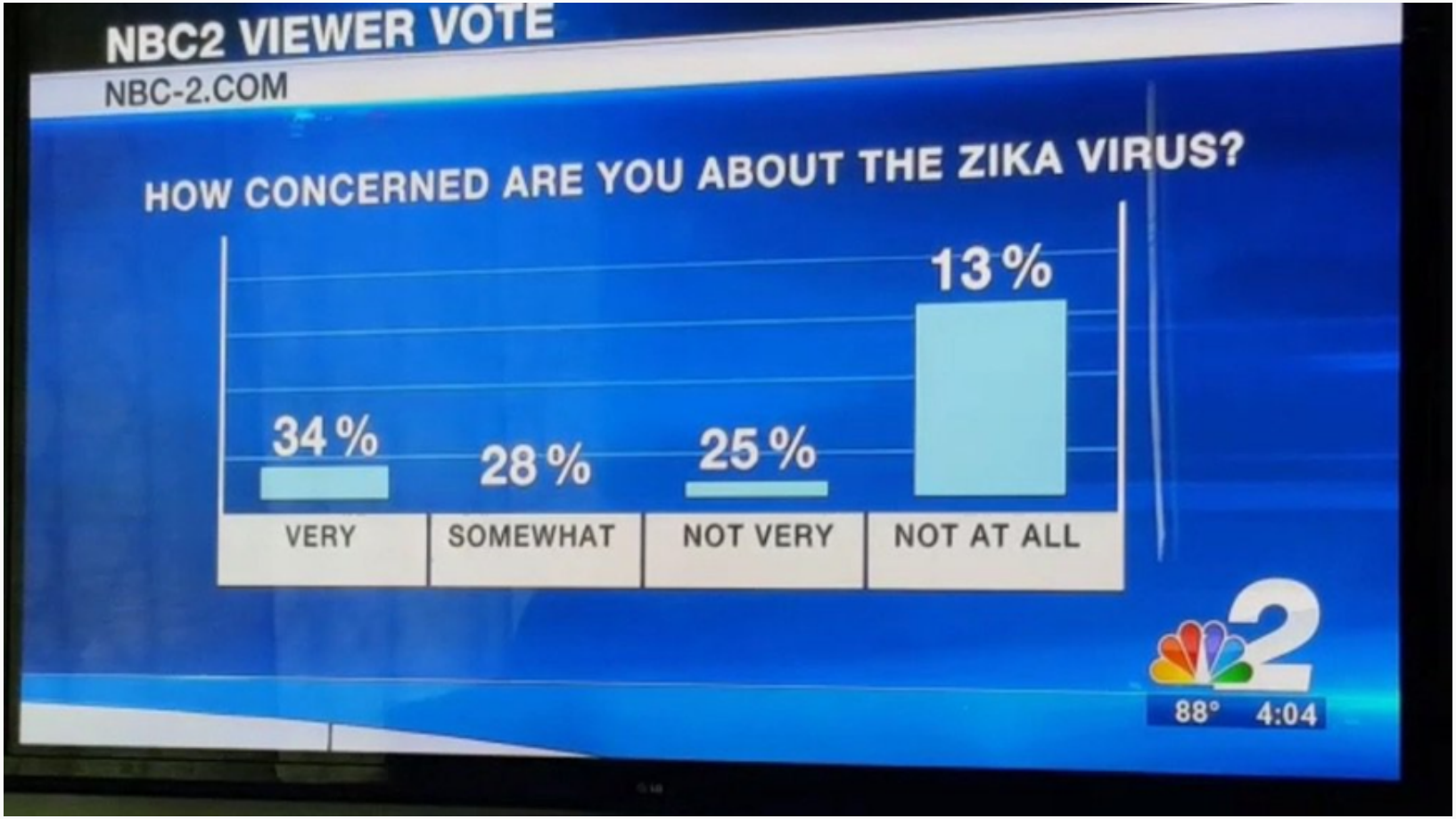
- Unintentionally obscure or purposefully misreport true relationships
- Convey too much information without the necessary orienting information easily accessible
- Are completely uninterpretable

Imperfect Charts



Source: **Wall Street Journal, 2/11/2015**.

Dishonest Charts



Source: **McGarry, 8/15/2016 (Mashable)**.

Principles of Data Visualization

When constructing charts, we need to keep in mind several **key principles:**

1. Show the Data

2. Reduce the Clutter

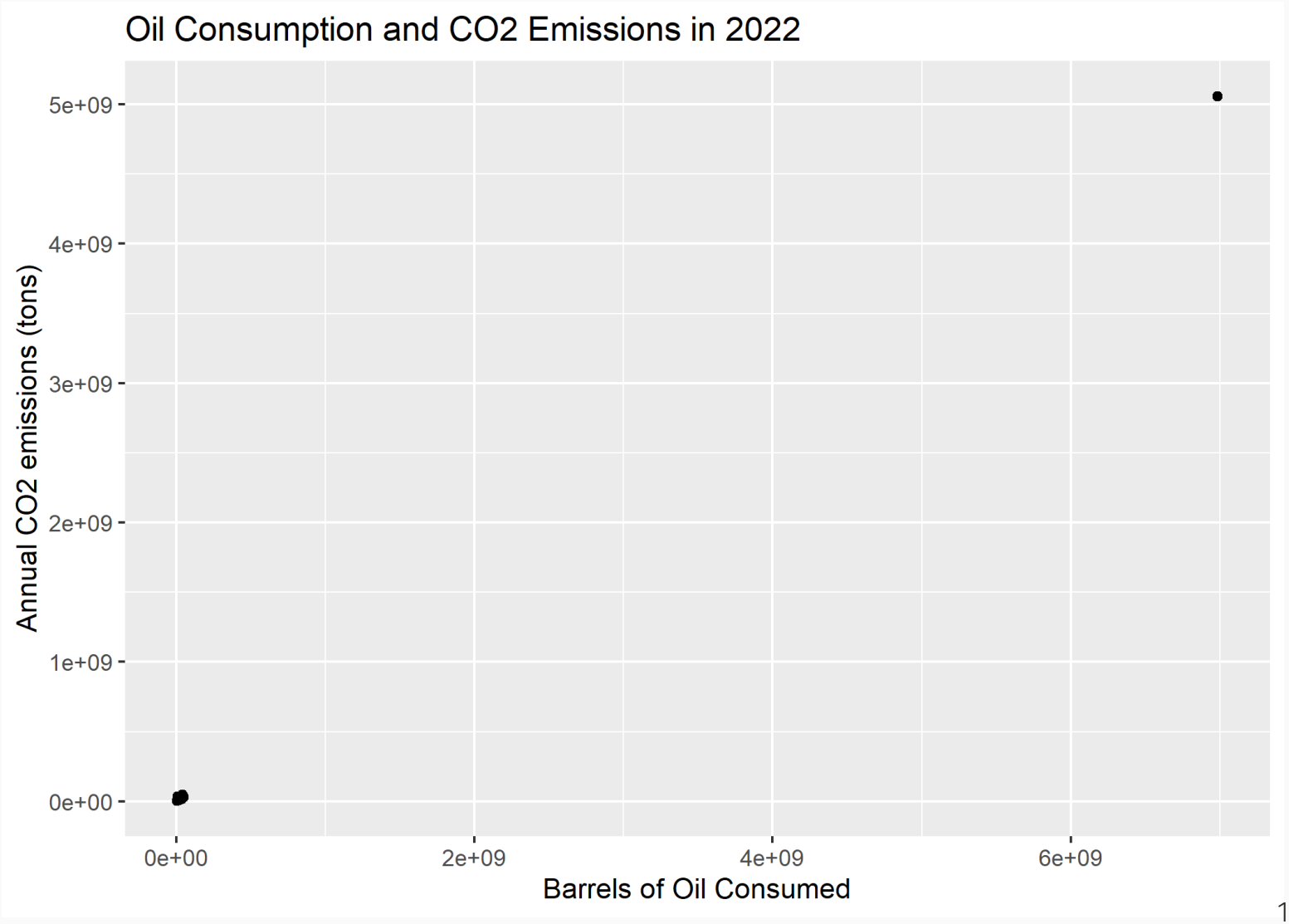
3. Integrate the Text and the Graph

Show the Data

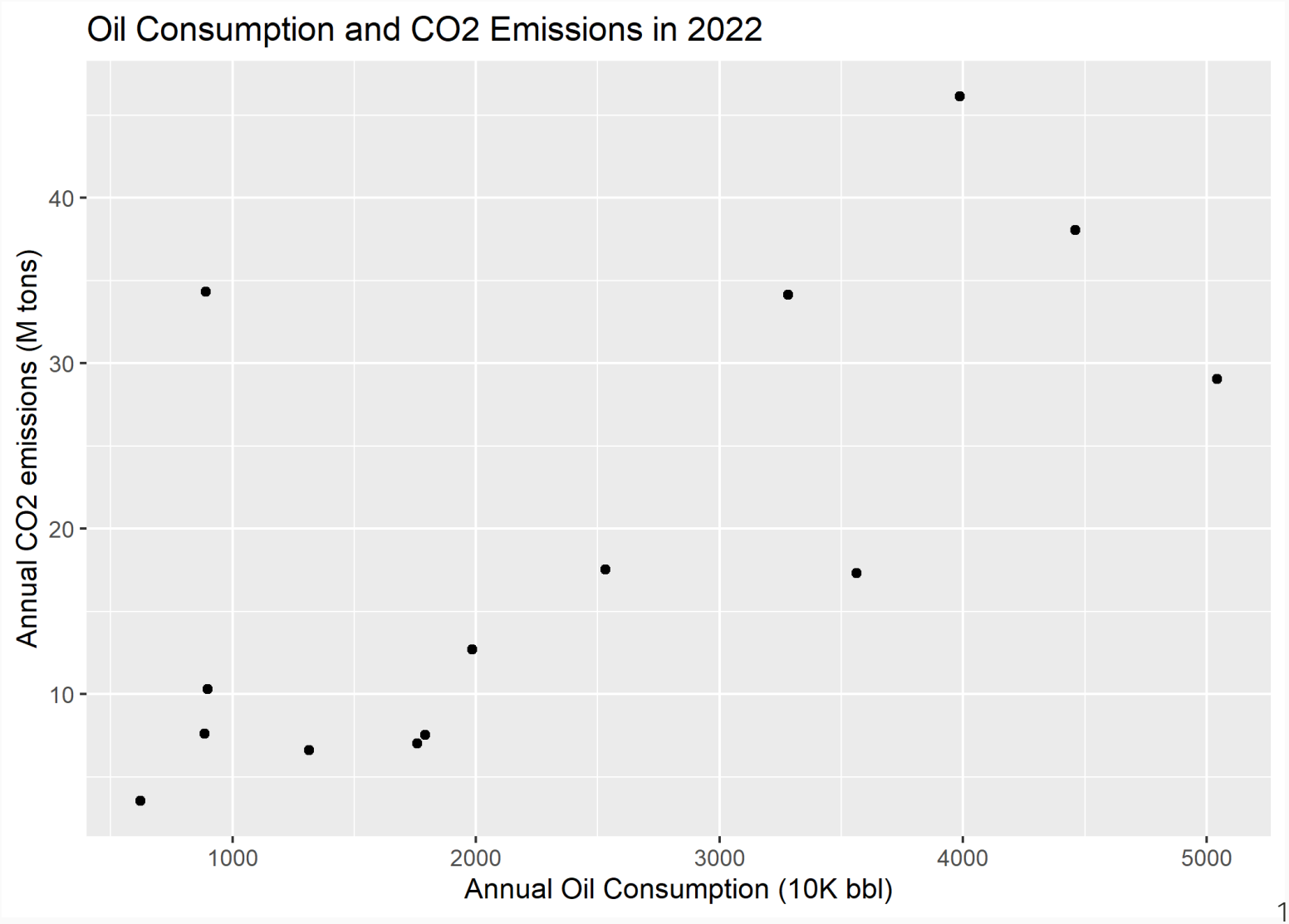
We use graphs to help readers/listeners understand a story. The data are the **most important part** of this.

- Present the data in the **clearest way possible**
- However, that doesn't mean we have to show **all the data**

Show the Data



Show the Data (Clearly)



Reduce the Clutter

Ask yourself: What is the **central message** you are trying to communicate?

Decide, then build your plot around that message.

- Make that message as **easy to see as you can**.
- **Remove the clutter:** get rid of any features of the visualization that do not contribute to the central message.

Example of A "Clutterplot"

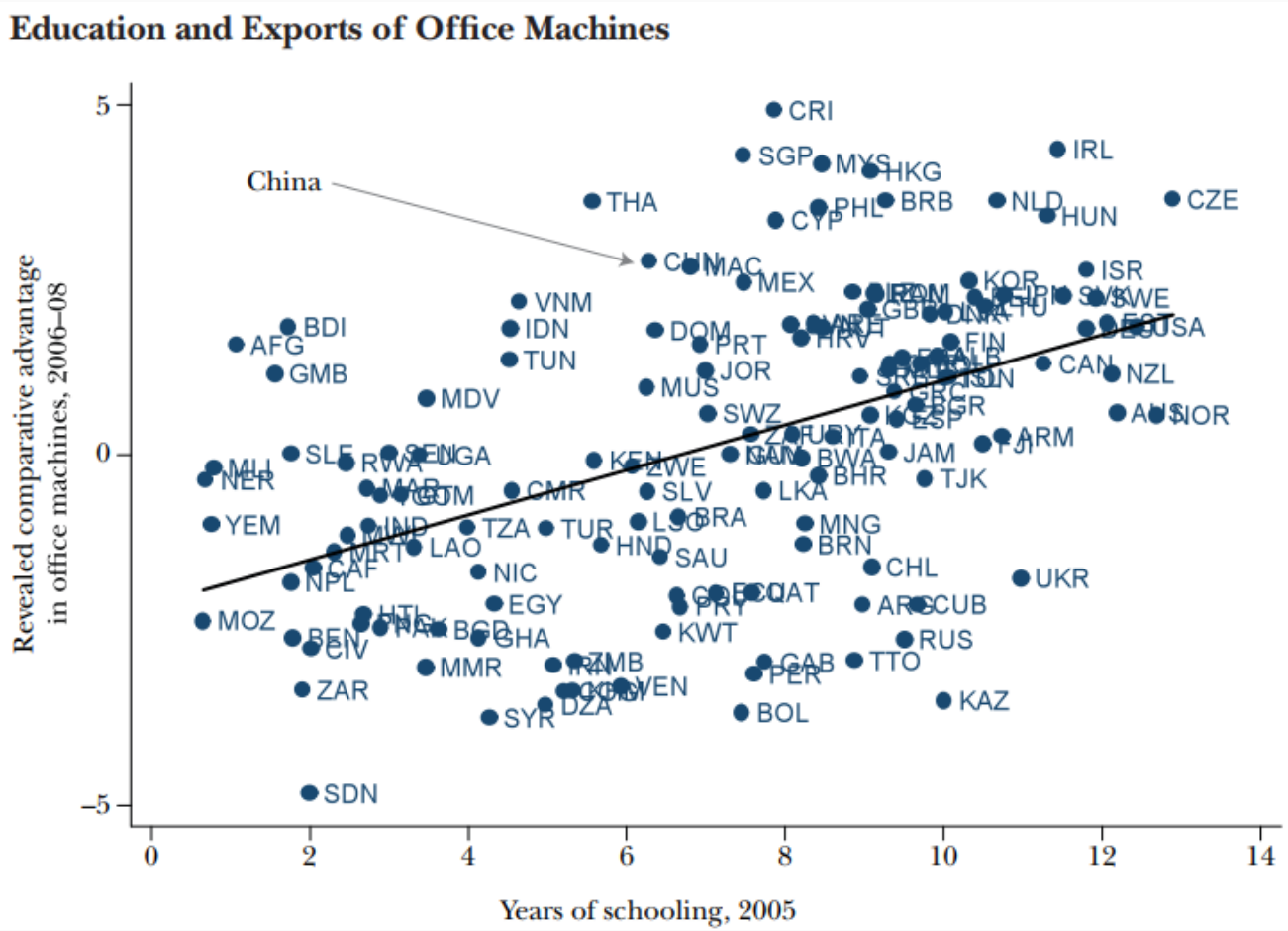


Image is from **"An Economist's Guide to Visualizing Data" by Jonathan Schwabish** and excluded from the overall CC license.

Reduce the Clutter

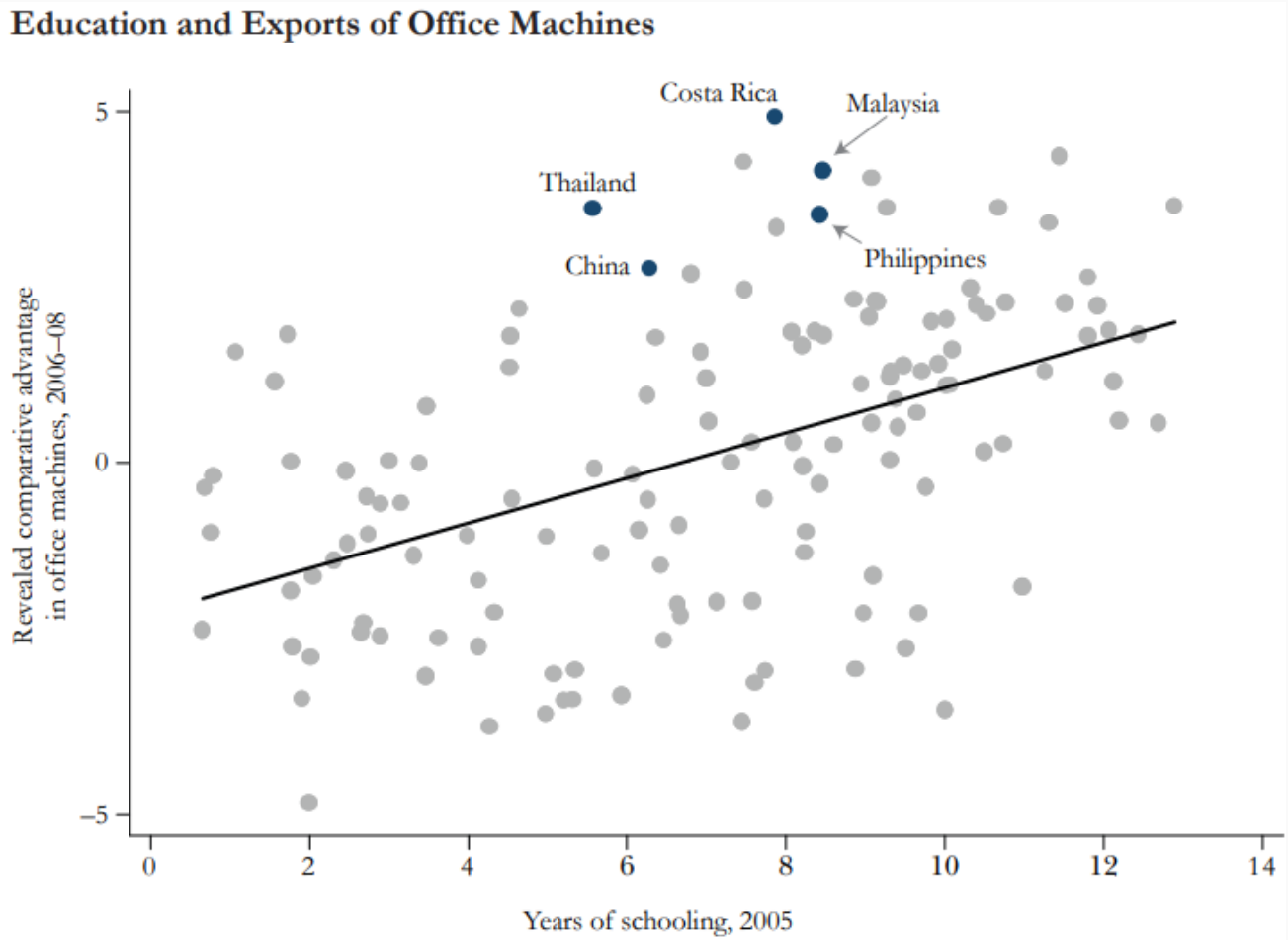


Image is from **"An Economist's Guide to Visualizing Data" by Jonathan Schwabish** and excluded from the overall CC license.

Integrate the Text and the Graph

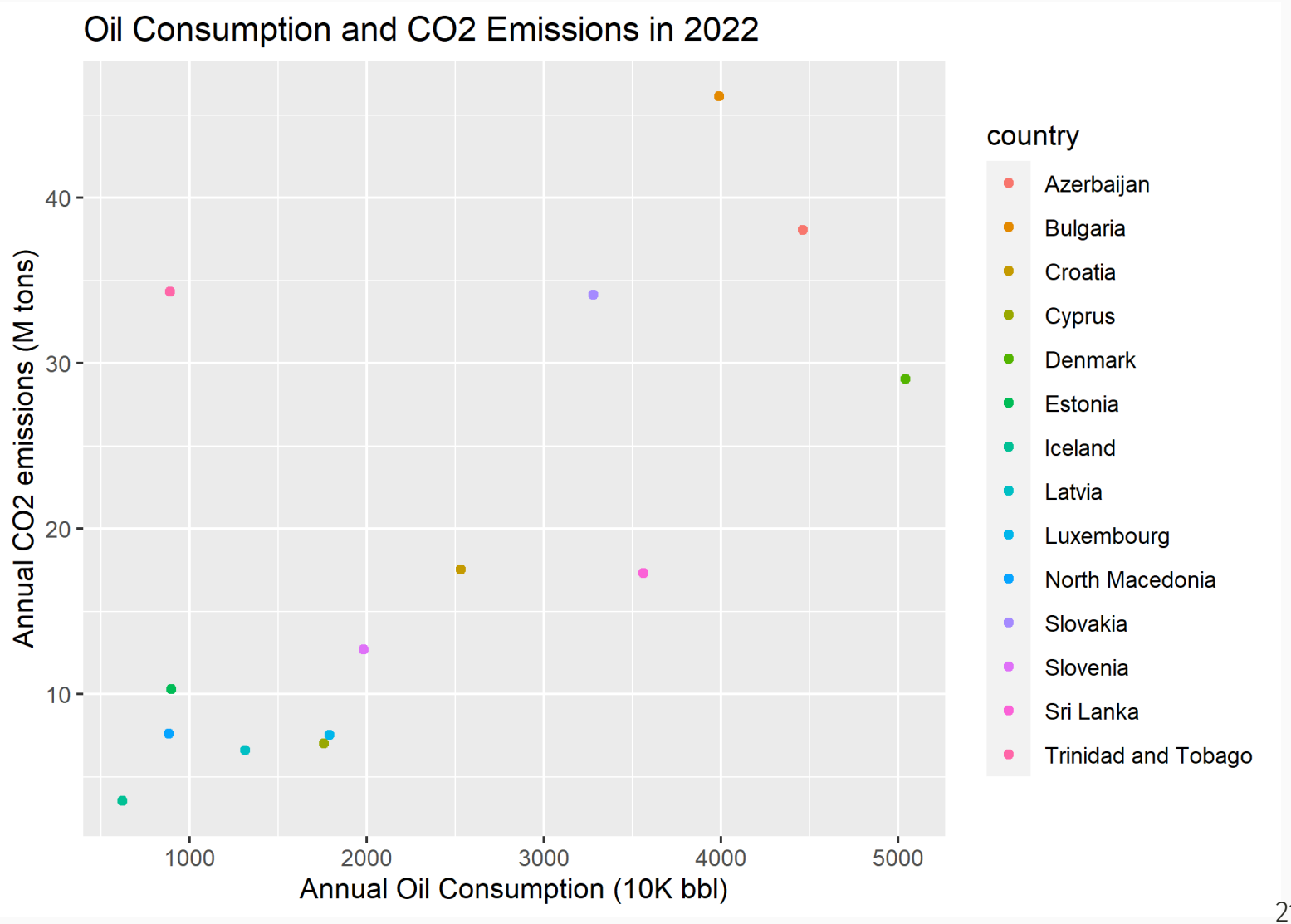
Research reports often suffer from the **slideshow effect**

- Writer narrates the graph's text elements

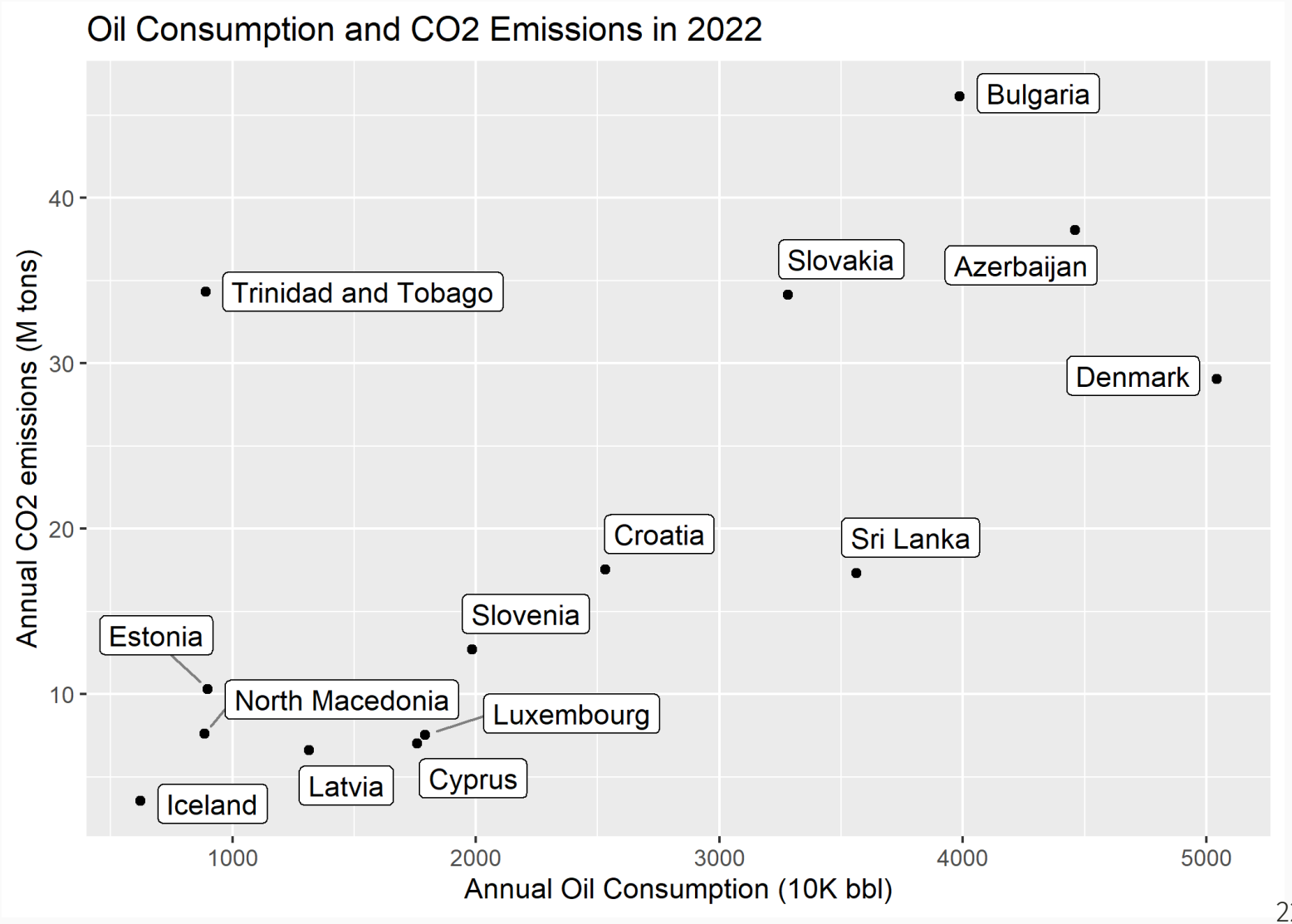
Better: integrate the text and the graph!

- Visuals are built to **complement the text**
- Charts contain enough information to **fully stand alone**
- Place labels/legend elements **close to the element its referencing**

Disconnected Text and Graph



Integrate the Text and the Graph



Principles of Data Visualization

Many of the charts that *don't* follow these principles often violate either

1. Area Principle: the area occupied by the chart element reflects its full value

- **Common violation:** bar/column chart axis doesn't start at zero

2. White Space Rule: if there's too much white space, **refocus the plot**

- Remove unnecessary outliers
- Split into multiple panels
- Change axis scales (e.g. logarithmic - but know your audience!)

Violating the Area Principle

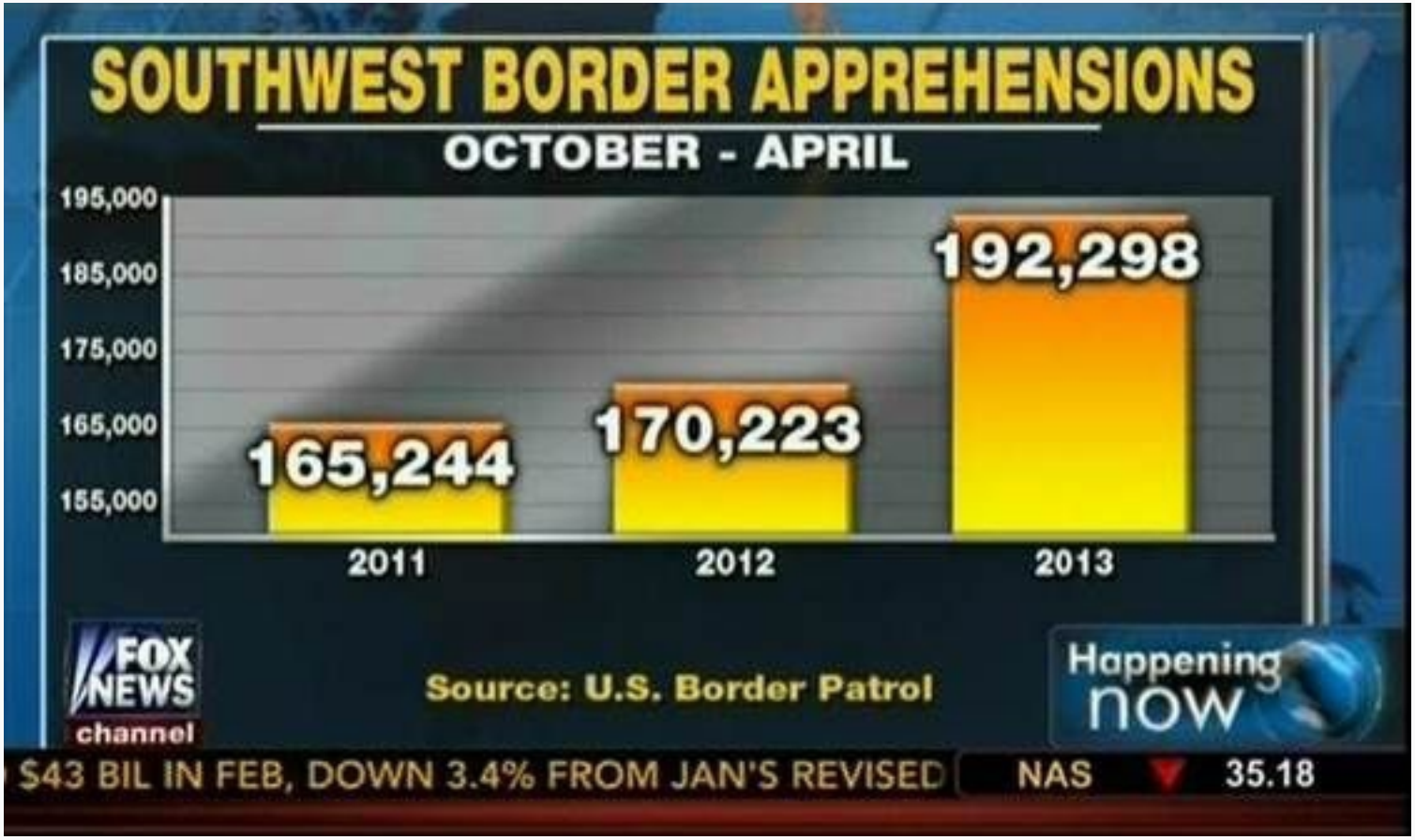


Image is from **"Introduction to Data Science" by Rafael A. Irizarry**

Violating the Area Principle

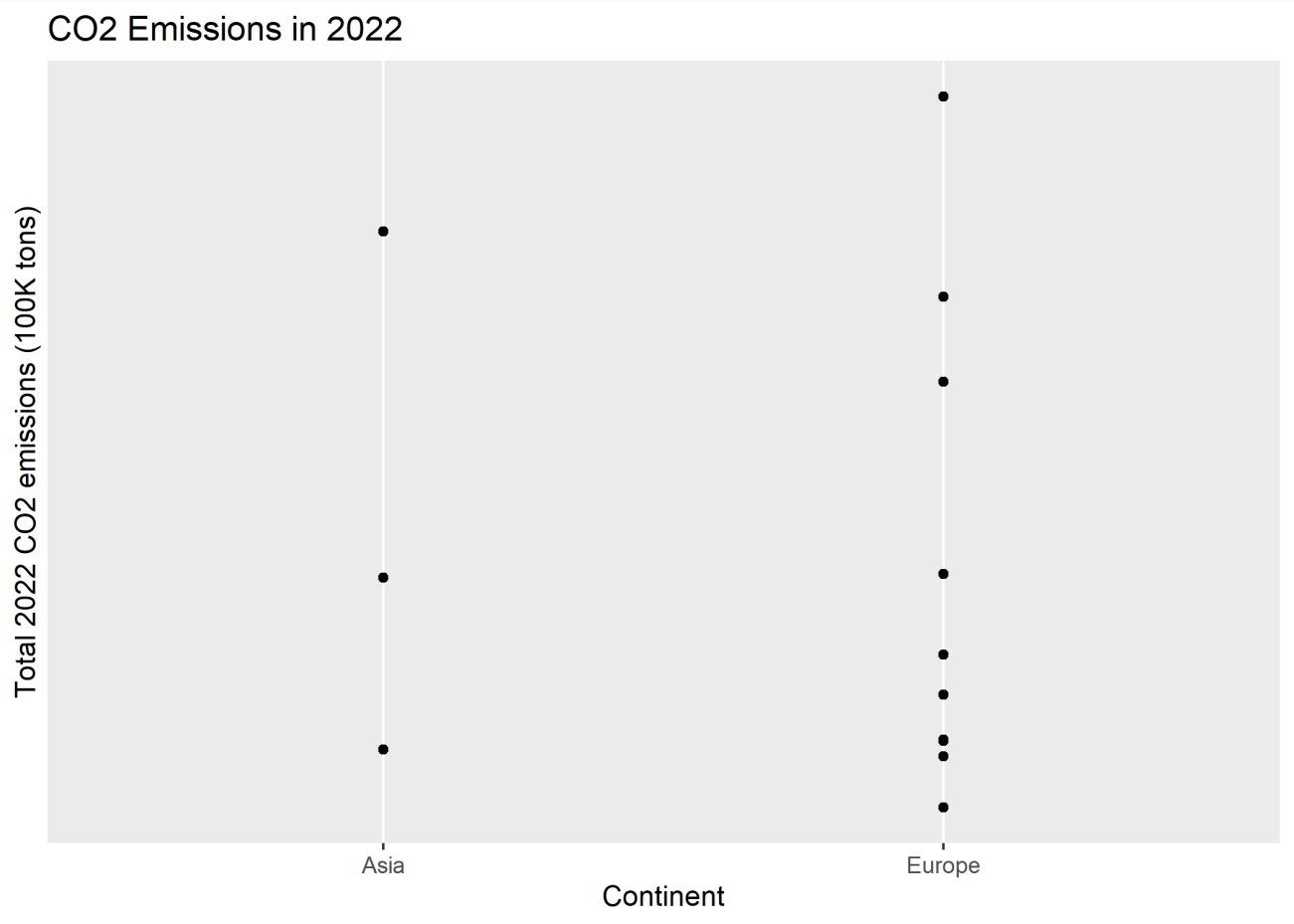
Image is from **"Introduction to Data Science" by Rafael A. Irizarry**

Violating the Area Principle (Again)

Image is from **"Introduction to Data Science" by Rafael A. Irizarry**

NOT an Area Principle Violation

Note that we *don't* need to include 0 if we use **position** rather than length:



Getting Started with ggplot2

Elements of ggplot2

ggplot2 is one of the most popular packages in the entire R canon.

- Built upon some deep visualization theory: i.e. **Leland Wilkinson's "The Grammar of Graphics"**.

There's a lot to say about **ggplot2**'s implementation of this "grammar of graphics" (gg) approach, but the three key elements are:

1. Your plot ("the visualization") is linked to your variables ("the data") through various **aesthetic mappings.**

2. Once the aesthetic mappings are defined, you can represent your data in different ways by choosing different **geoms** (i.e. "geometric objects" like points, lines or bars).

3. You build your plot in **layers**.

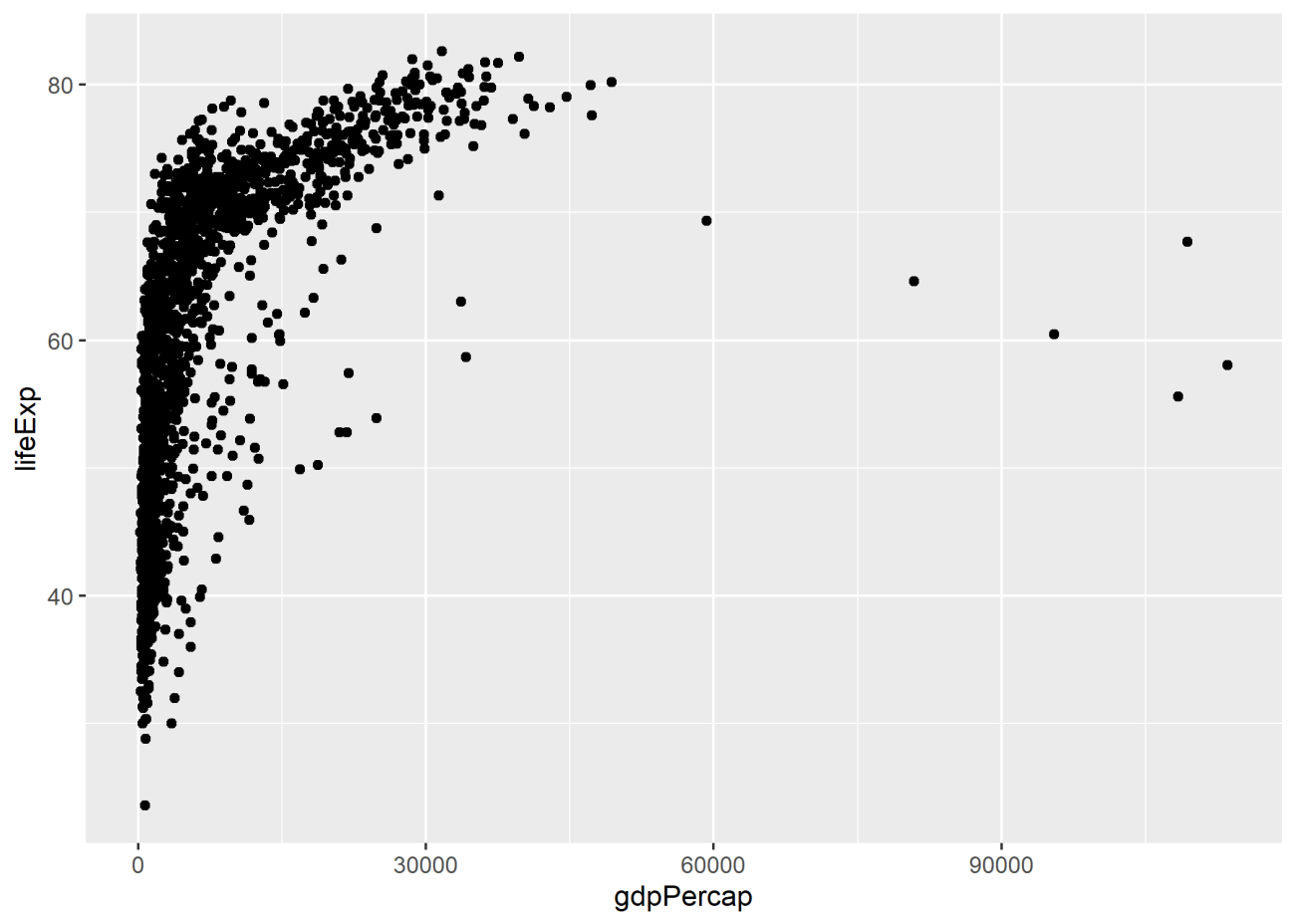
Elements of ggplot2

1. Link plots to variables through **aesthetic mappings**
2. Represent data using **geoms**
3. Build your plot in **layers**

That's kind of abstract. Let's review each element in turn with some actual plots, using panel data on life expectancy, population size, and GDP per capita for 142 countries since the 1950s from the **gapminder** package

1. Aesthetic Mappings

```
ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



1. Aesthetic Mappings

```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```

Focus on the top two lines, which contain the initialising `ggplot()` function call. This function accepts various arguments, including:

- Where the data come from
  - i.e. `data = gapminder`
- What the aesthetic mappings are
  - i.e. `mapping = aes(x = gdpPercap, y = lifeExp)`
  - Here we're setting the mapping **globally** (applies to all subsequent geom layers)

1. Aesthetic Mappings

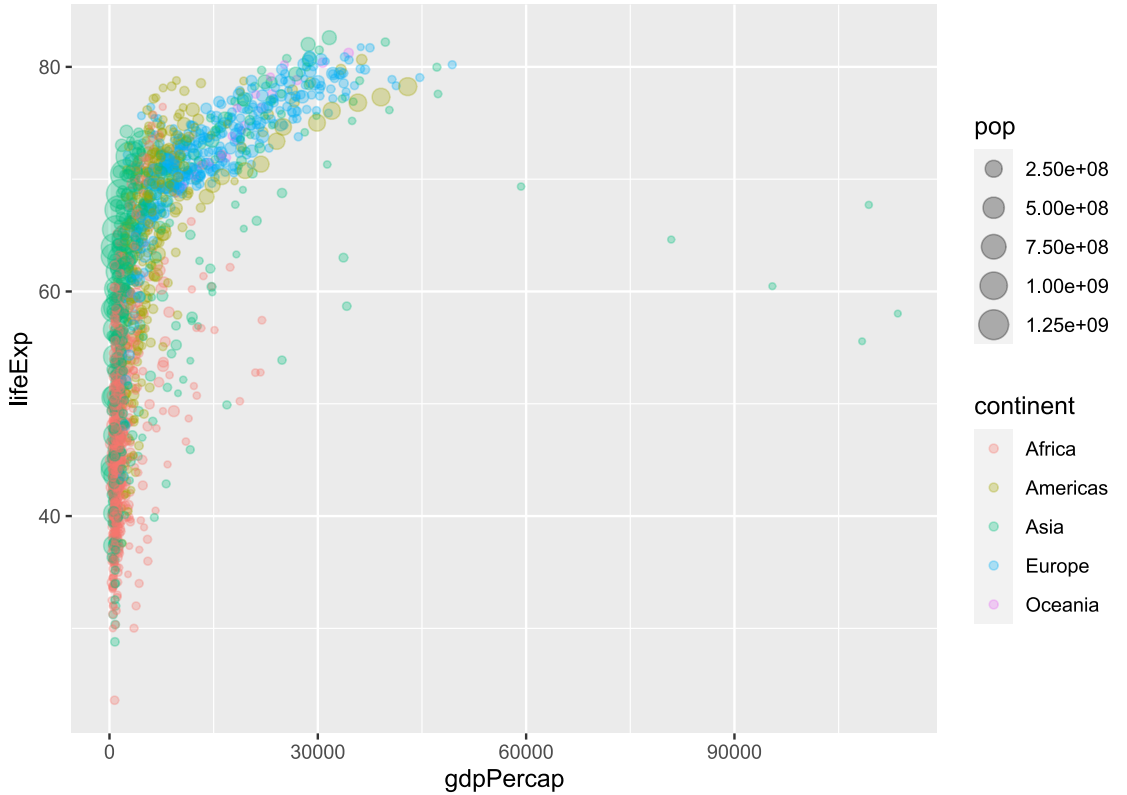
```
ggplot(data = gapminder,
       mapping = aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```

The aesthetic mappings here are pretty simple: They just define an x-axis (GDP per capita, `x`) and a y-axis (life expectancy, `y`).

- To get a sense of the power and flexibility that comes with this approach, however, consider what happens if we add more aesthetics to the plot call...

1. Aesthetic Mappings

```
ggplot(data = gapminder,
       aes(x = gdpPercap, y = lifeExp,
           size = pop, col = continent)) +
  geom_point(alpha = 0.3) ## "alpha" controls transparency. Takes a value
```



1. Aesthetic Mappings

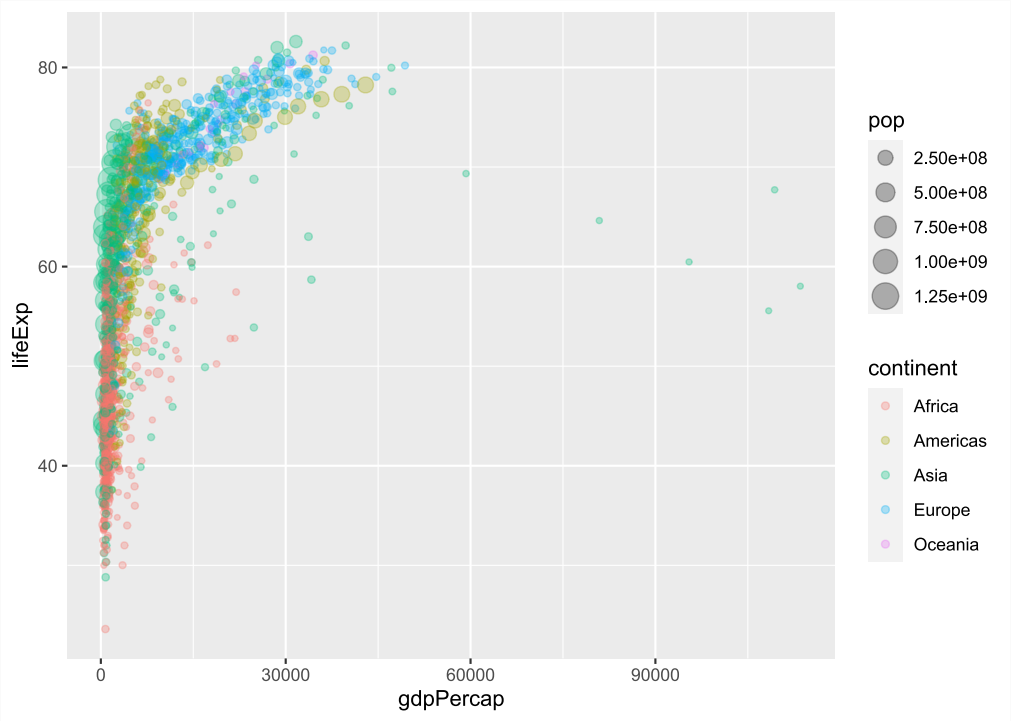
```
ggplot(data = gapminder,
       aes(x = gdpPercap,
           y = lifeExp, # set x/y variables
           size = pop, # scale point size with value of "pop"
           col = continent # change color on value of "continent"
           )
       ) +
  geom_point(alpha = 0.3) # "alpha" controls transparency in [0,1]
```

Note that I've dropped the `mapping =` part of the ggplot call. Most people just start with `aes( ... )`, since **ggplot2** knows the order of the arguments.

1. Aesthetic Mappings

We can specify aesthetic mappings **locally in the geom layer too**.

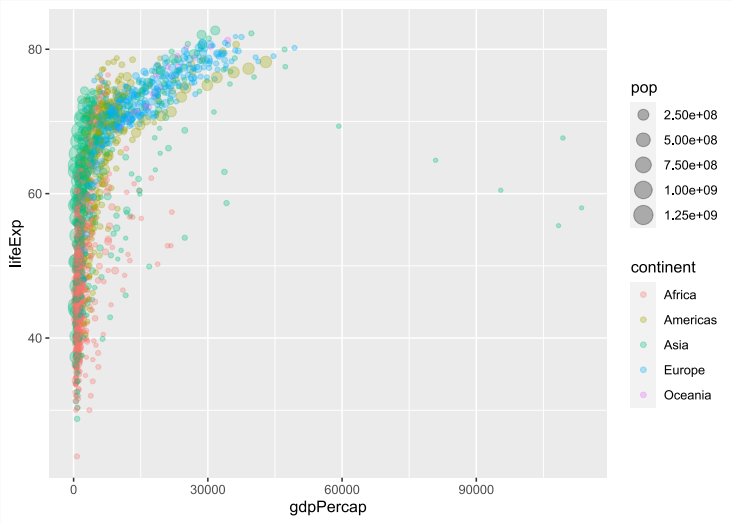
```
## First aes: applicable to all geoms
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
## Next aes: specific to this geom only
  geom_point(aes(size = pop, col = continent), alpha = 0.3)
```



1. Aesthetic Mappings

Data can be declared **locally too**.

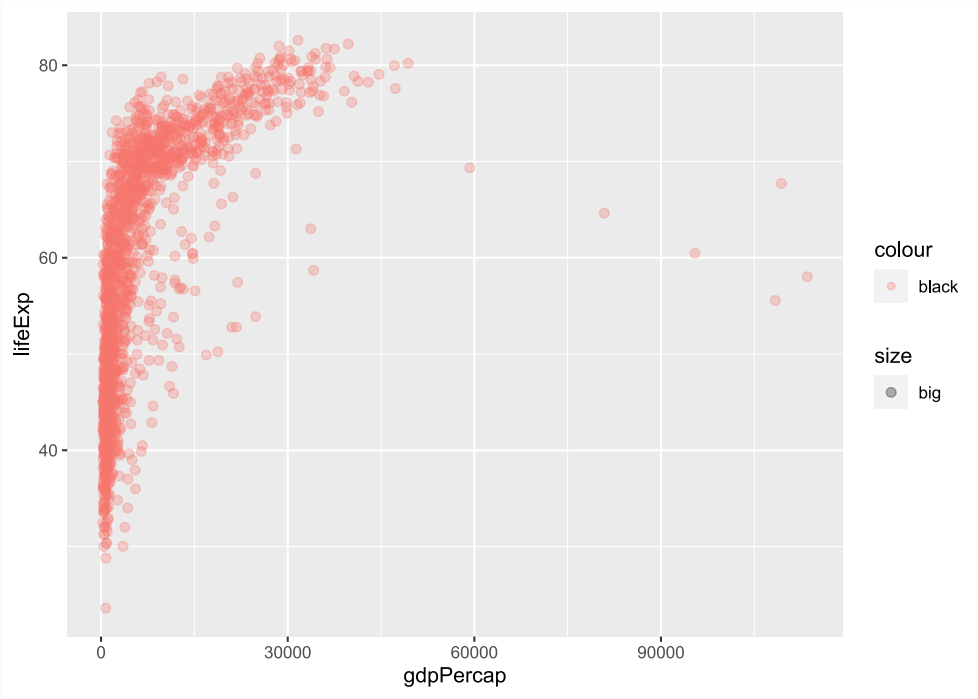
```
## First aes: applicable to all geoms
ggplot() +
## Next aes: specific to this geom only
  geom_point(data = gapminder,
             aes(x = gdpPercap, y = lifeExp, size = pop, col = continent)
```



1. Aesthetic Mappings

Oops. What went wrong here?

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    aes(size = "big", col="black"),
        alpha = 0.3)
```



1. Aesthetic Mappings

Oops. What went wrong here?

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp)) +
  geom_point(
    aes(size = "big", col="black"),
    alpha = 0.3)
```

Answer: Aesthetics must be mapped to **variables**, not descriptions!

1. Aesthetic Mappings

Instead of repeating the same ggplot2 call every time, we can**store it in memory** as an intermediate plot object that we can re-use.[1]

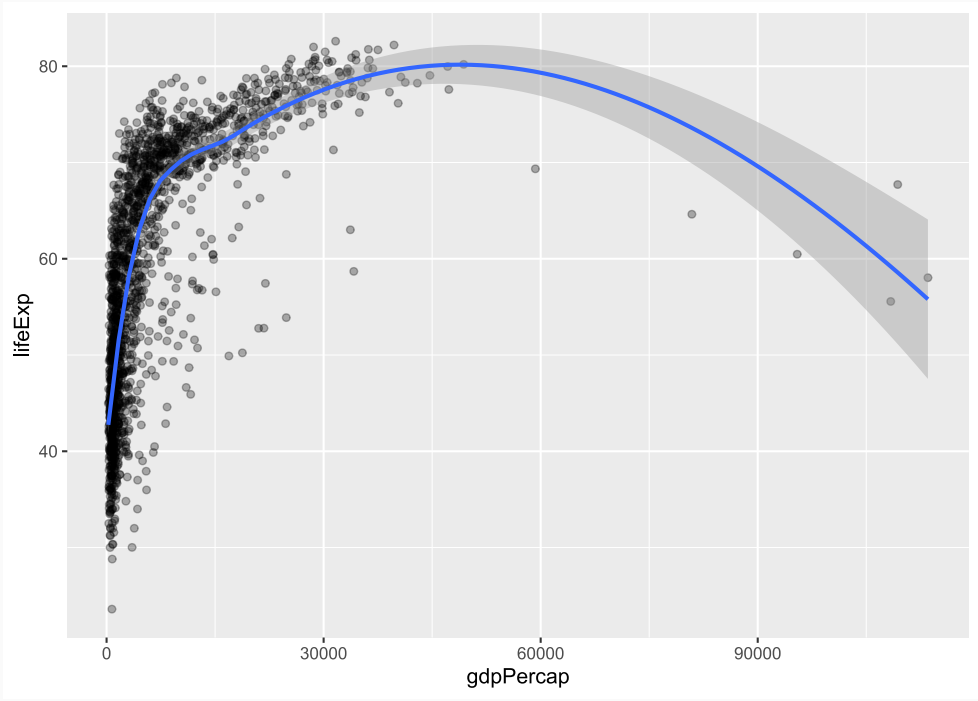
```
p ← ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp))
```

[1] We can store anything in memory! It's like R's version of the **Portlandia "put a bird on it" sketch**

2. Geoms

You can invoke and combine different **geoms** to generate **different visualizations**.

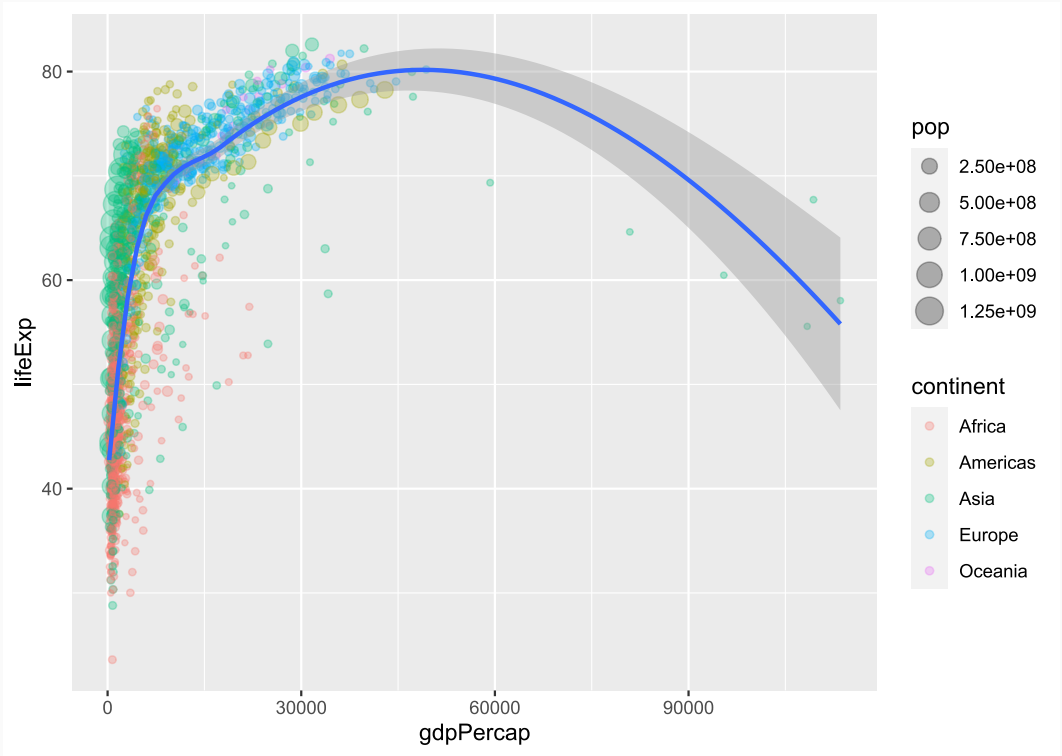
```
p +
  geom_point(alpha = 0.3)  +
  geom_smooth(method = "loess") # add local polynomial regression fit
```



2. Geoms

Aesthetics can be applied **differentially across geoms**.

```
p +
  geom_point(aes(size = pop, col = continent), alpha = 0.3)  +
  geom_smooth(method = "loess")
```



2. Geoms

The previous plot provides a good illustration of the power (or effect) that comes from assigning aesthetic mappings "globally" vs in the individual geom layers "locally".

- Compare: What happens if you run the below code chunk?

```
ggplot(data = gapminder, aes(x = gdpPercap, y = lifeExp, size = pop, col =
  geom_point(alpha = 0.3)  +
  geom_smooth(method = "loess")
```

2. Geoms (cont.)

Similarly, note that some geoms only accept a subset of mappings. E.g. `geom_density()` doesn't know what to do with the "y" aesthetic mapping.

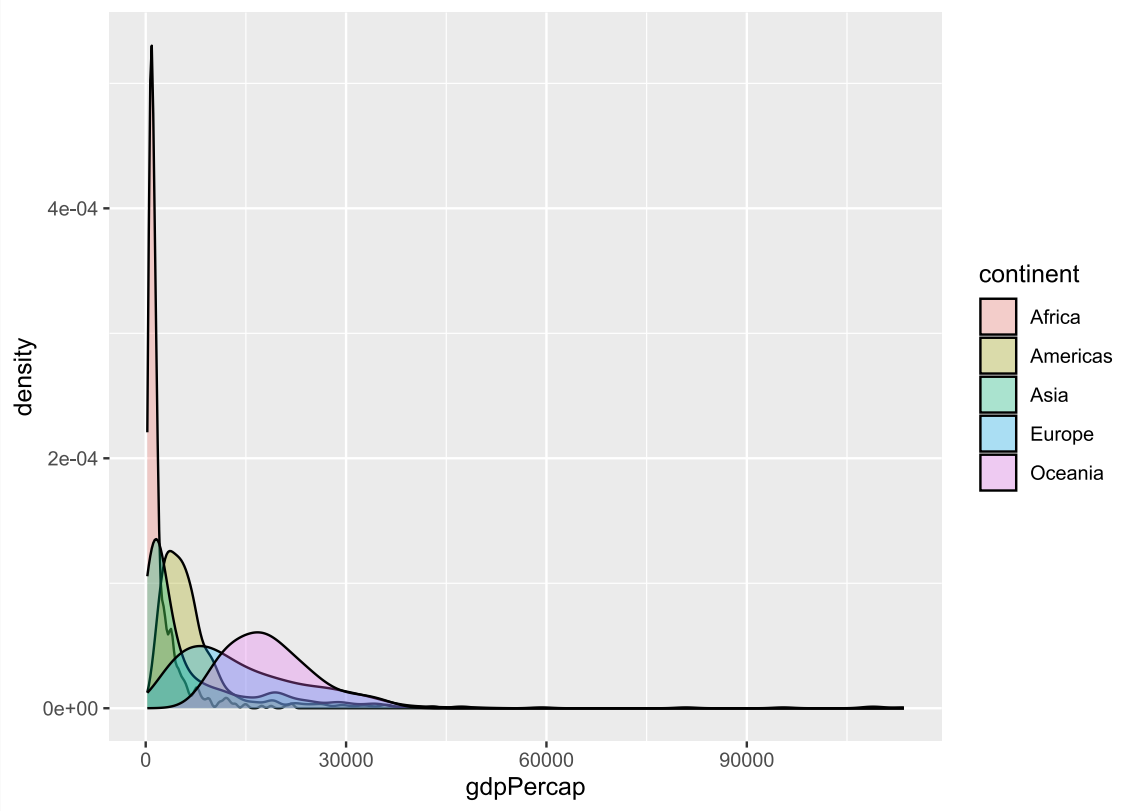
```
p + geom_density()
```

```
## Error in geom_density():
## ! Problem while setting up geom.
## ℹ Error occurred in the 1st layer.
## Caused by error in compute_geom_1():
## ! geom_density() requires the following missing aesthetics: y
```

2. Geoms (cont.)

We can fix that by being more careful about how we build the plot.

```
ggplot(data = gapminder) + ## i.e. No "global" aesthetic mappings"
  geom_density(aes(x = gdpPercap, fill = continent), alpha=0.3)
```



3. Build your Plot in Layers

We've already seen how we can chain (or "layer") consecutive plot elements using the `+` connector.

- The fact that we can create and then re-use an intermediate plot object (e.g. `p`) is testament to this.

But it bears repeating: you can build out some **truly impressive complexity and transformation** of your visualization through this simple layering process.

- You don't have to transform your original data; **ggplot2** takes care of all of that.
- For example...

3. Build your Plot in Layers

(See next slide for figure)

```
p2 ← p +
  geom_point(aes(size = pop, col = continent), alpha = 0.3) +
  scale_color_brewer(name = "Continent", palette = "Set1") + ## Different
  scale_size(name = "Population", labels = scales::comma) + ## Different
  scale_x_log10(labels = scales::dollar) + ## Switch to logarithmic scale
  labs(x = "Log (GDP per capita)", y = "Life Expectancy") + ## Better axi
  theme_minimal() ## Try a minimal (b&w) plot theme
```

3. Build your Plot in Layers

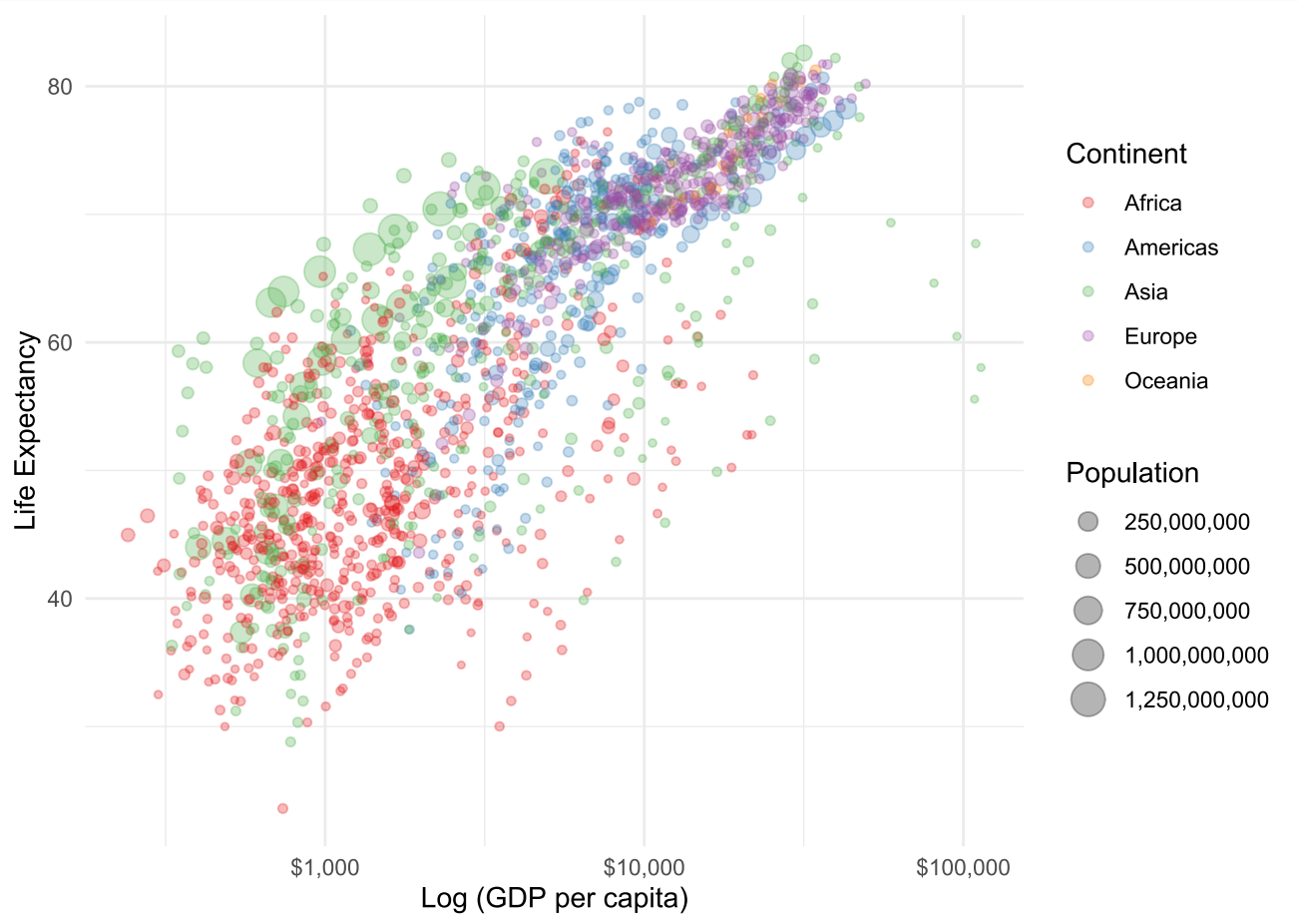


Table of Contents

This Lecture:

Next Lecture: