Introduction to Data Science and Analytics (DSC510)

Data Visualization

George Pallis

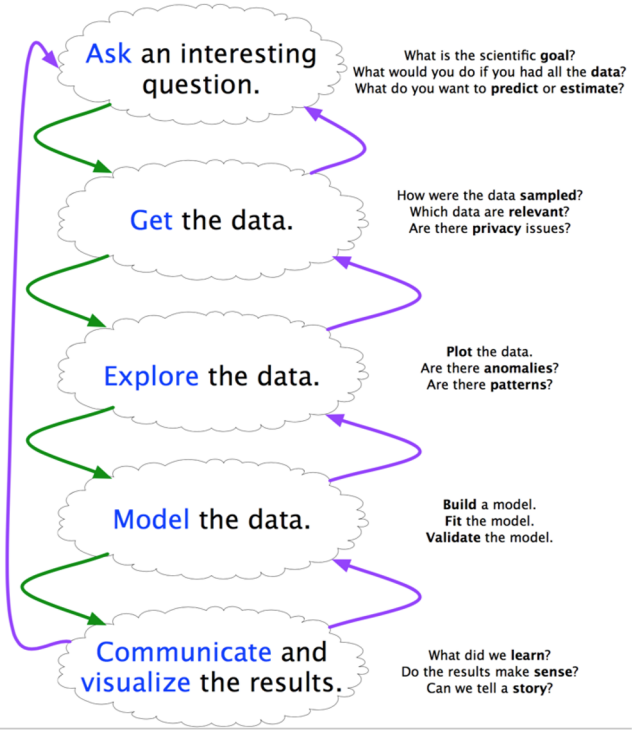
Uses for data visualization

Support reasoning about information (**analysis**)

- Finding relationships
- Discover structure
- Quantifying values and influences
- Should be part of a query/analyze cycle

Inform and persuade others (**communication**)

- Capture attention, engage
- Tell a story visually
- Can focus on certain aspects and omit othe



Static viz

Great for data exploration, developed throughout the last few centuries…

Interactive viz

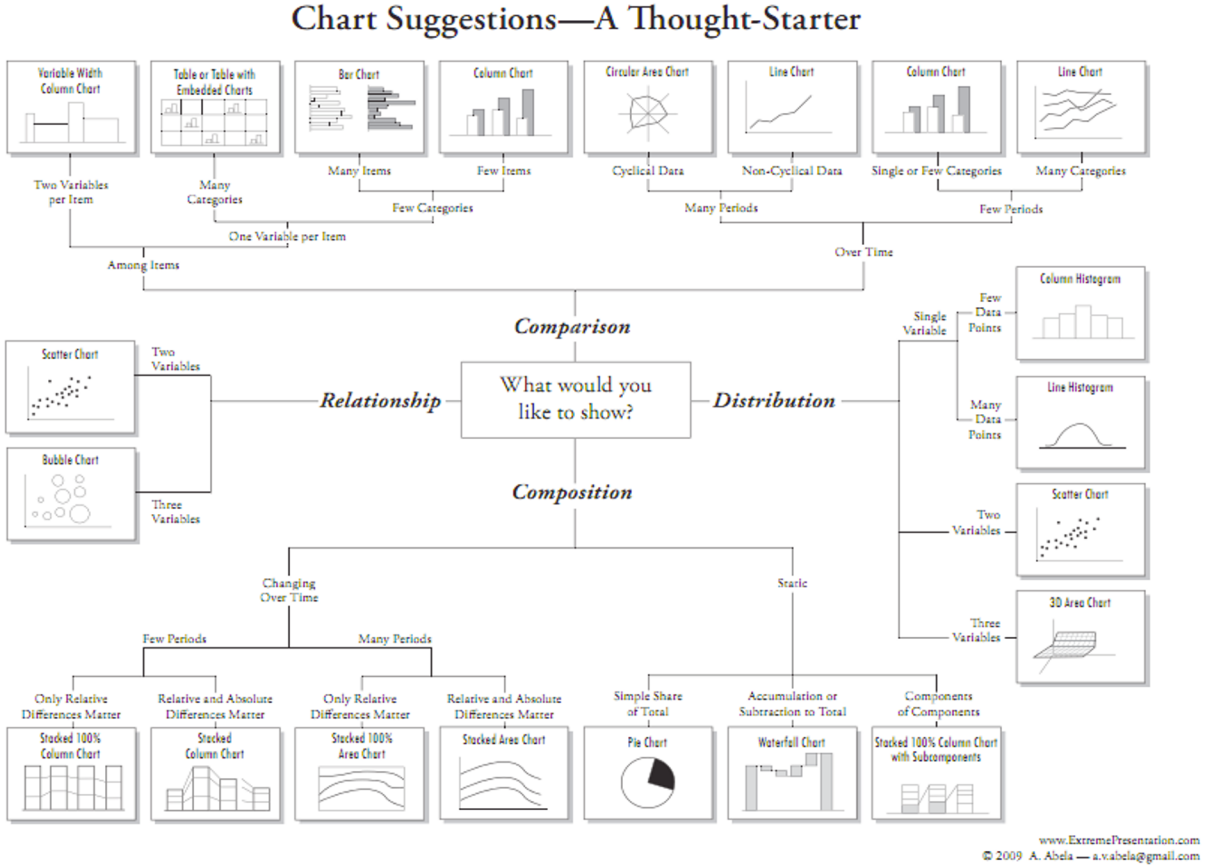
More and more common when delivering the results (and also during exploration). New frameworks are the key enabler.

Today's lecture

- Part 1: Navigating the chart landscape
- Part 2: Principles and best practices
- Part 3: A (small) selection of use cases for data visualization

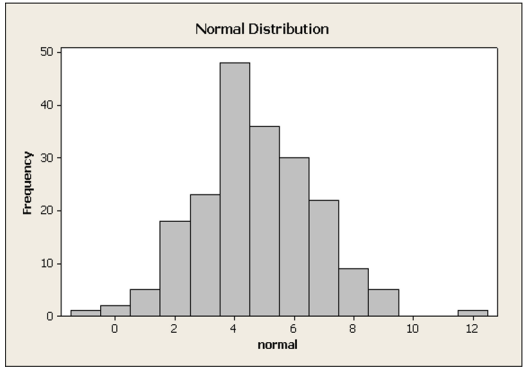
Part 1
Navigating the chart landscape

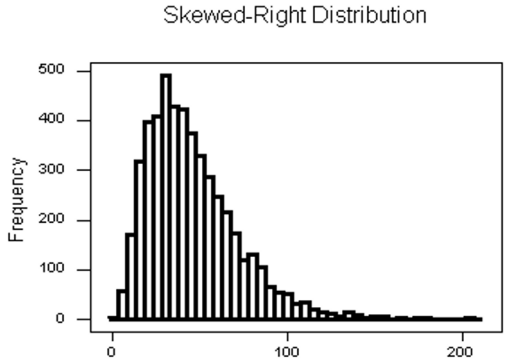
Chart selection



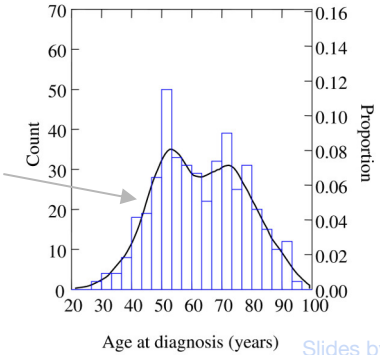
One variable: **histograms**

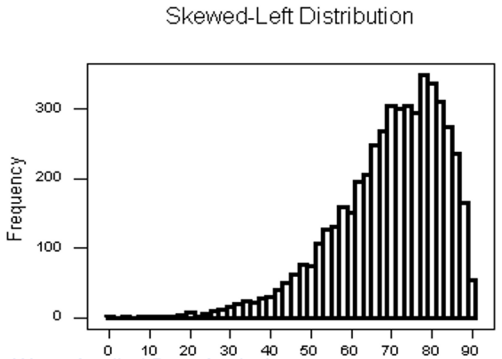
Histograms can tell you a lot about a single variable, discrete or continuous





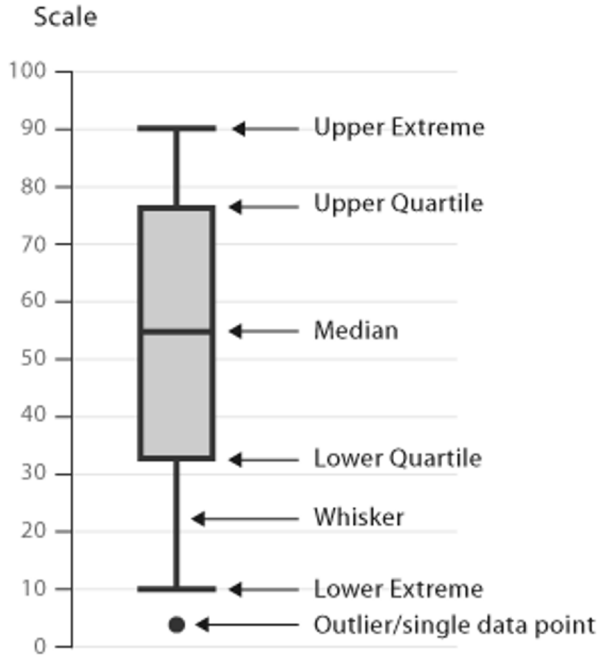
Smoothed histogram (a.k.a. kernel density estimate)

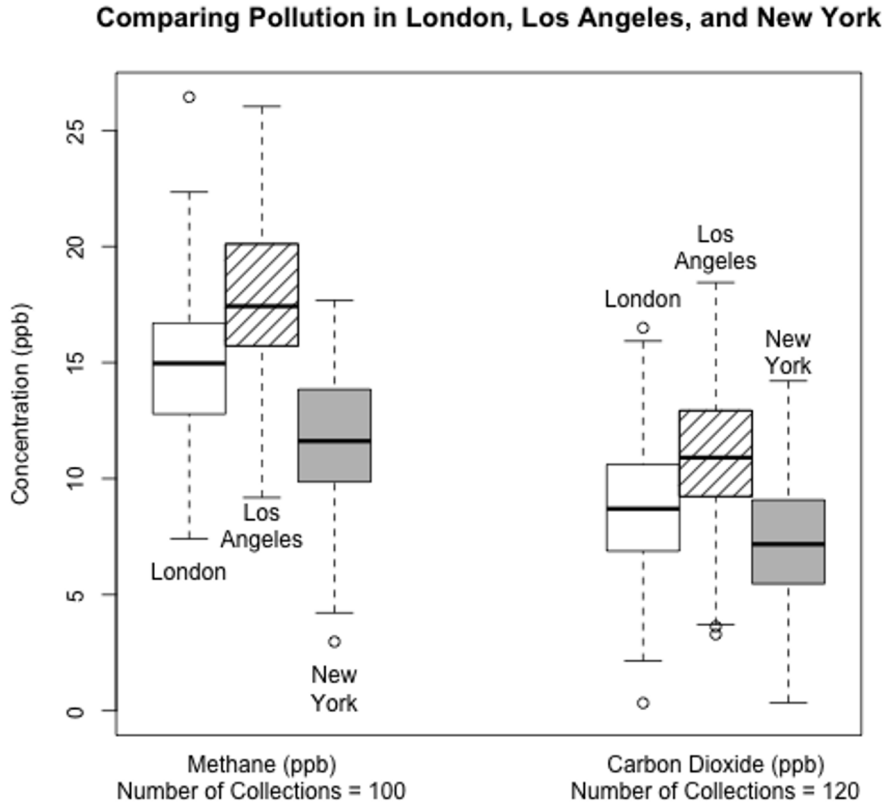




Easy to recognize skewed distributions!

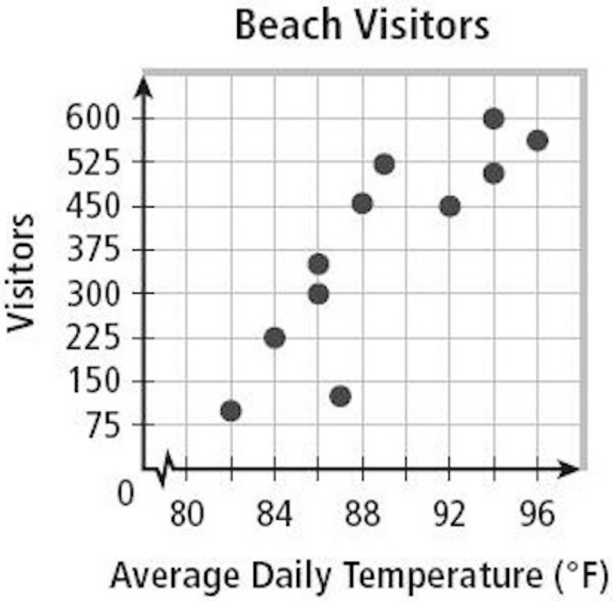
One variable: box plots





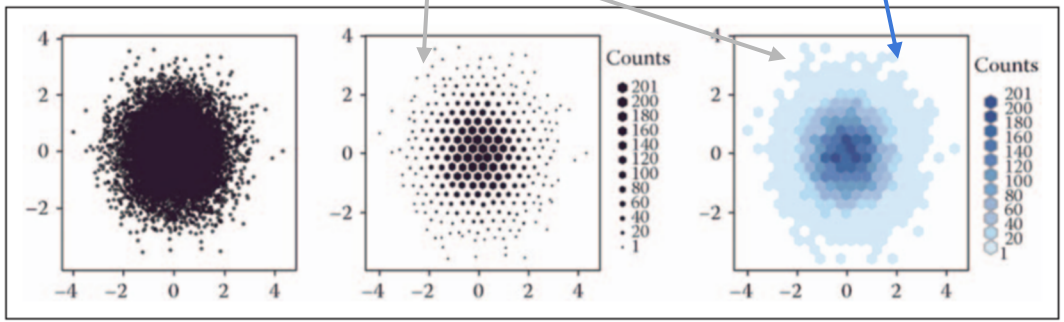
Two variables: scatter plots

Scatter plots quickly expose the relationships between two variables



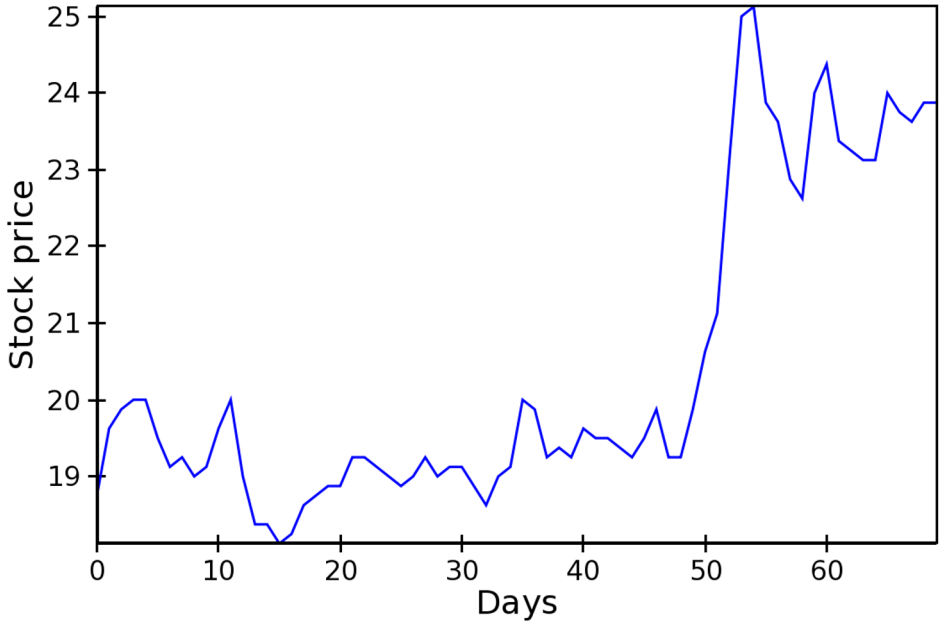
2D histograms

a.k.a. heatmap

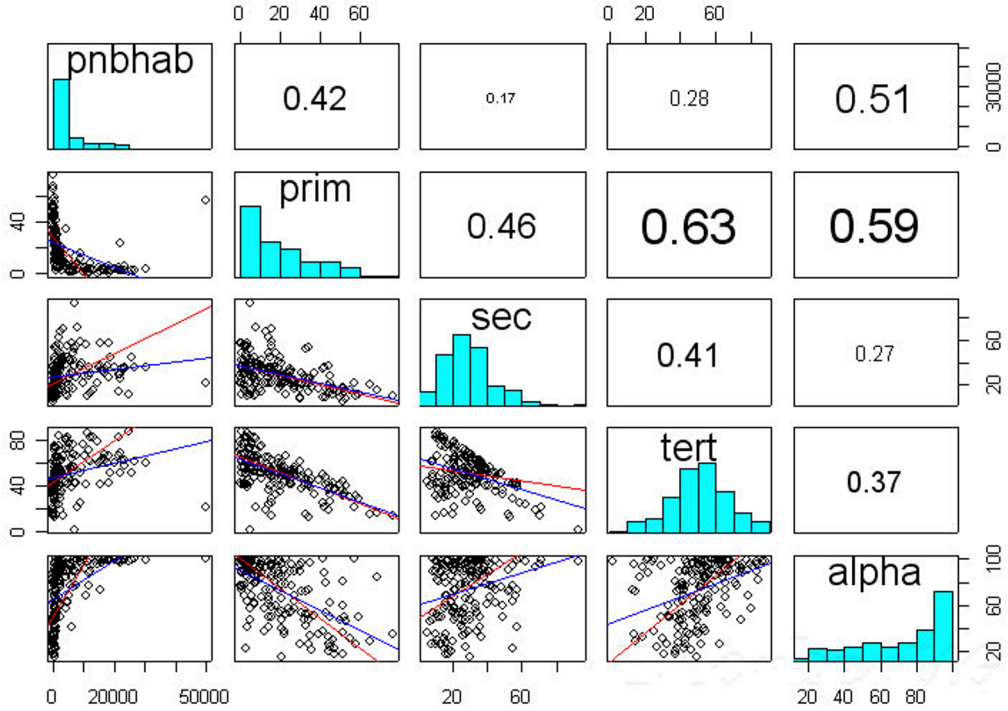


Two variables: line plots

If relationship is functional (e.g., after binning and aggregating)



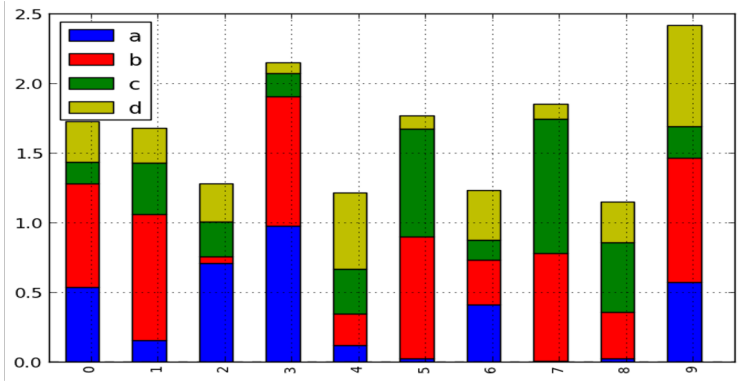
> 2 variables: scatter plot matrix



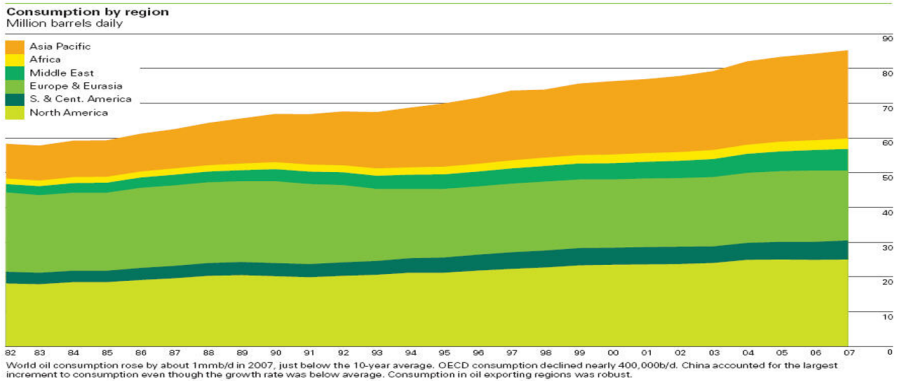
> 2 variables: stacked plots

Here: 3 variables: stack index, color, height

Stack variable and color variables categorical, height variable continuous:

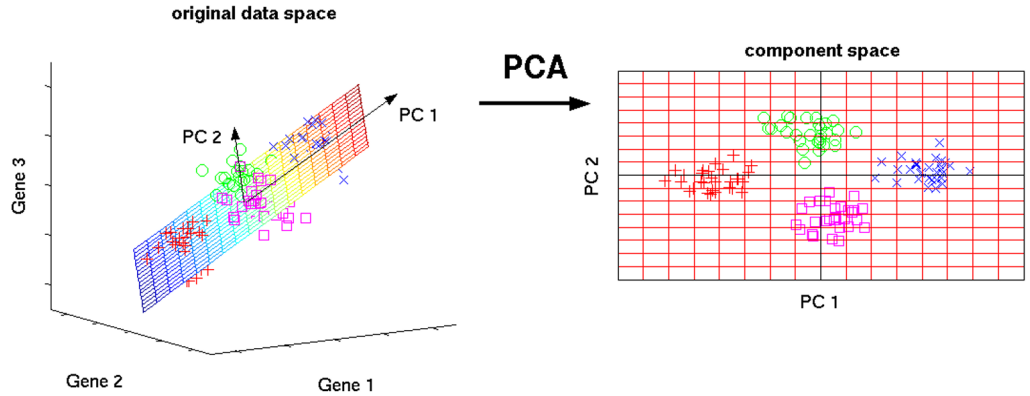


Color variable categorical, stack and height variables continuous:

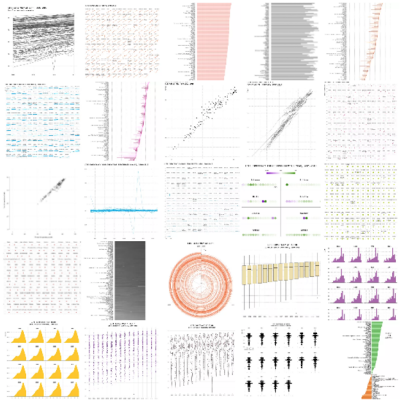


Dimensionality reduction

- For example, **PCA:** allows visualization of high-dimensional continuous data in 2D using principal components
- The principal components are the strongest (highest variation) dimensions in the dataset, and are orthogonal



One dataset, visualized 25 ways

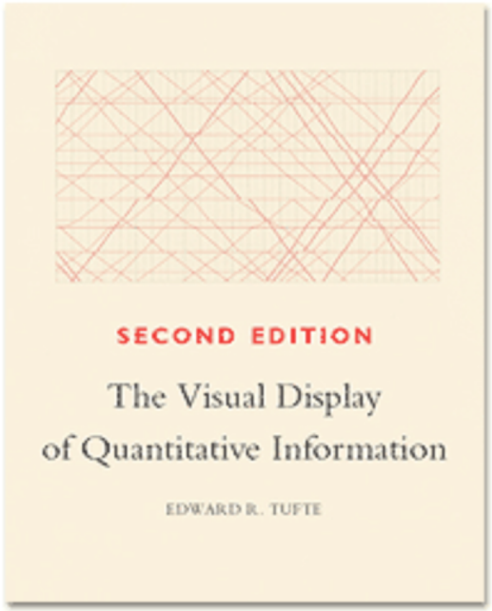


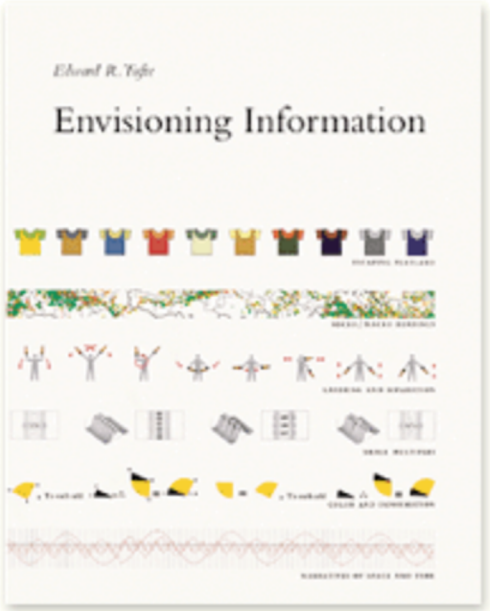
http://flowingdata.com/2017/01/24/one-dataset-visualized-25-ways

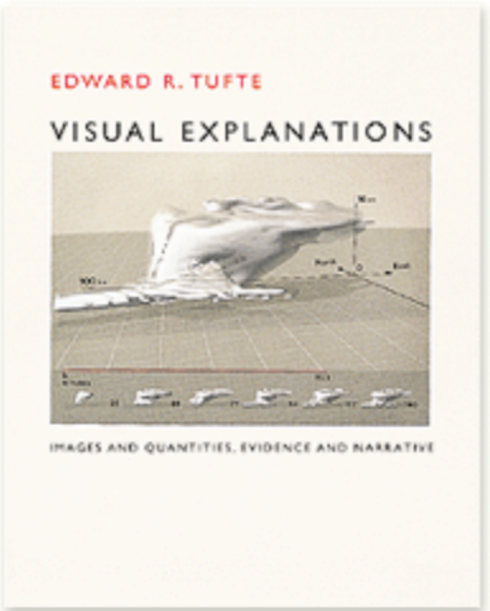
"You must help the data focus and get to the point. Otherwise, it just ends up rambling about what it had for breakfast this morning and how the coffee wasn't hot enough."

Part 2
Principles and best practices

Instructive coffee table books by Edward Tufte







Perception of magnitudes

(128, 128, 128)

(144, 144, 144)

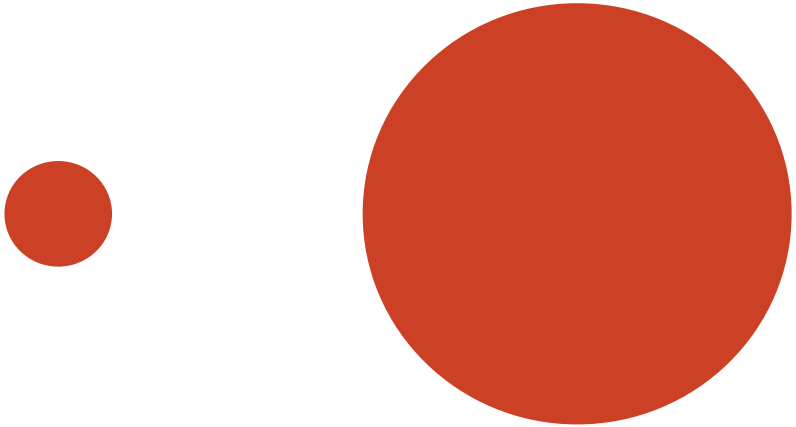
Which is brighter?

Just noticeable difference (JND)

- Weber's Law:

$$\frac{\Delta I}{I} = k,$$

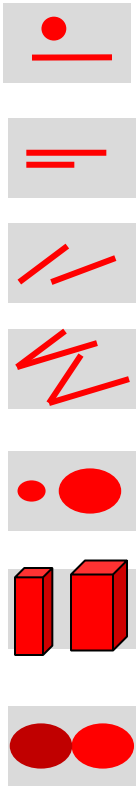
- I: intensity; ΔI: increase from I to notice a difference; k: constant
- Required increase ΔI depends on original intensity I
- **Most continuous variations in stimuli are perceived in discrete (multiplicative) steps**



Compare area of circles

Perception of magnitudes

Most accurate



Least accurate

Position

Length
Slope

Angle

Area

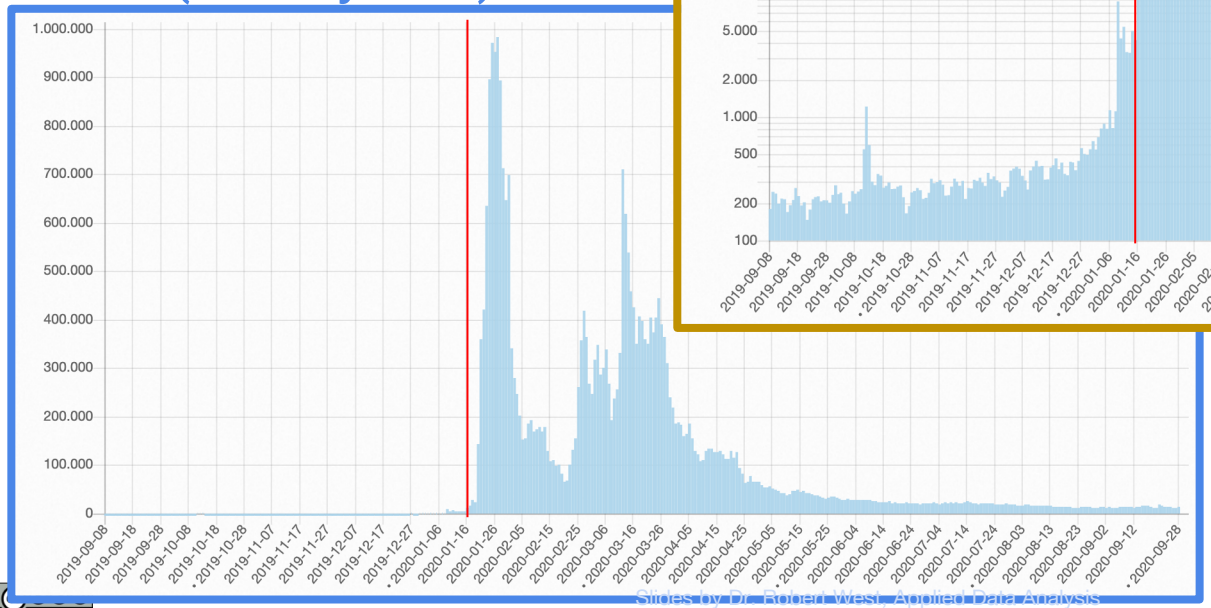
Volu

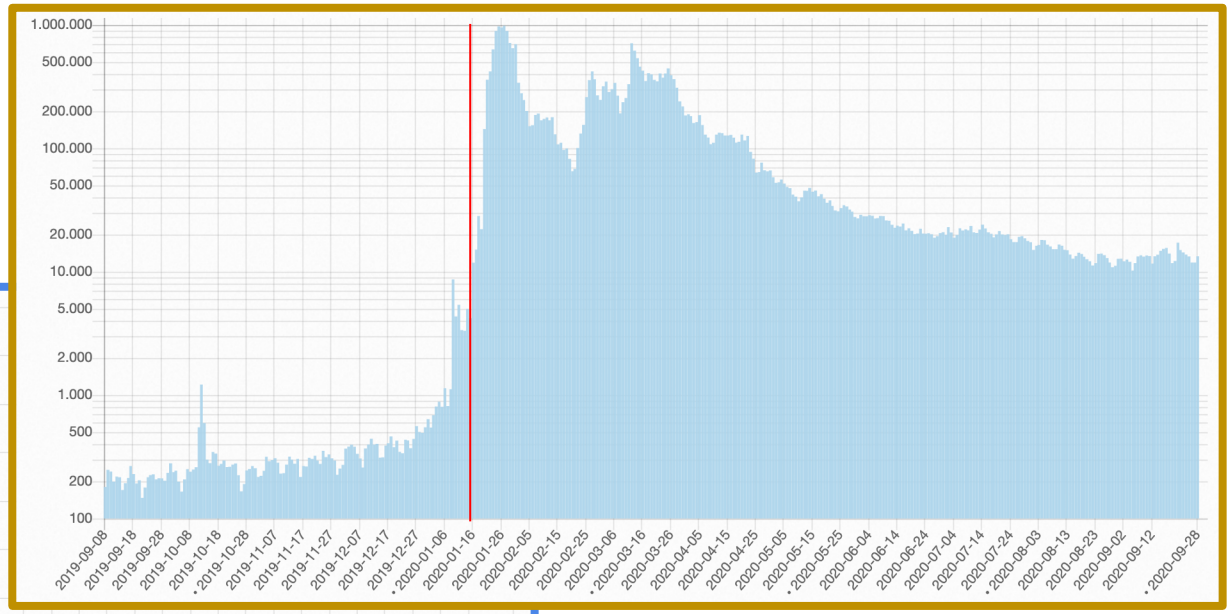
Color hue-saturation-density

Cleveland & McGill (1984)
Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods

Choose your axes wisely!

Time series of pageviews of Wikipedia article about “Coronavirus” (linear y-axis)

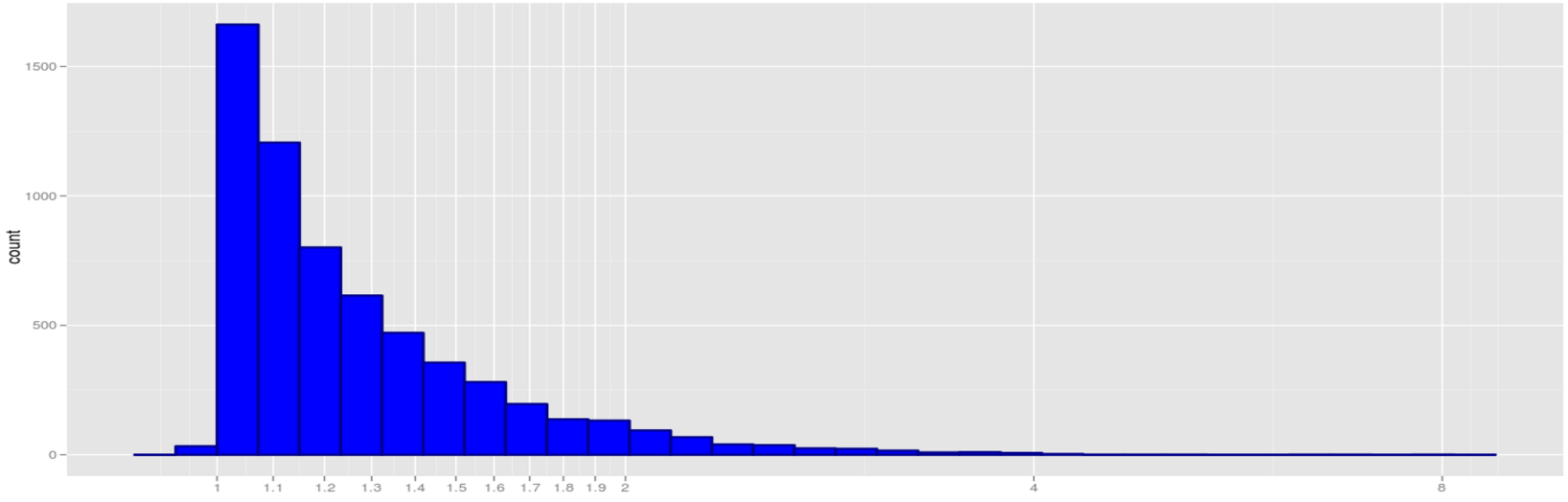




(logarithmic y-axis)

[link]

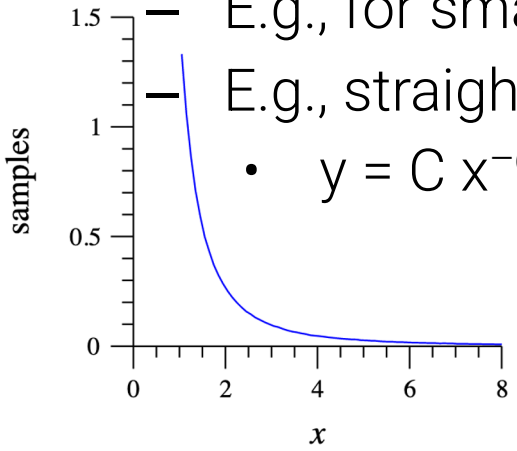
Choose your axes wisely: Visualizing heavy-tailed distributions

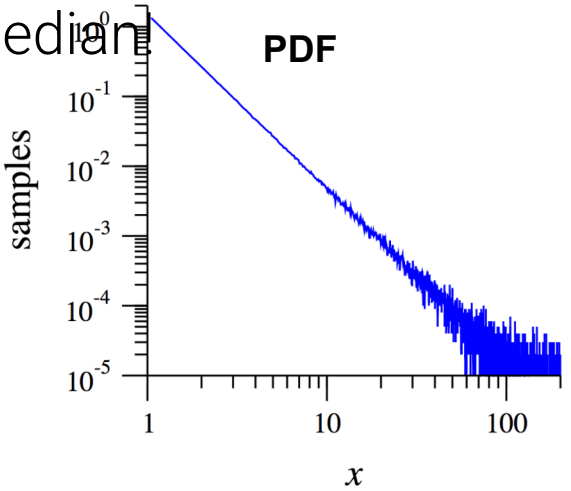


Heavy-tailed data: power laws

$$p(x) = Cx^{-\alpha},$$

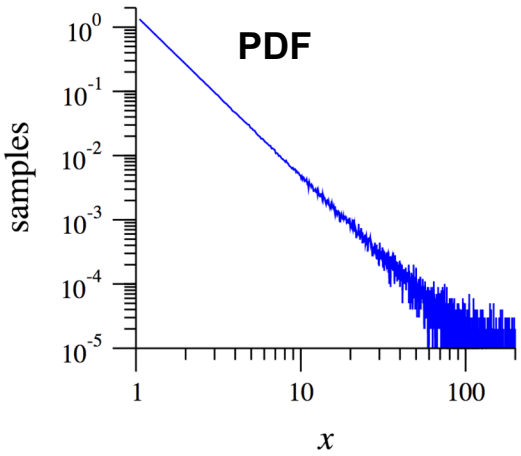
- Very large values are rare, “but not very rare”
- Body size vs. city size
- Many natural phenomena are power laws (e.g., # of friends)
- For dealing with them, need to know some tricks
- E.g., for small α, mean & var = ∞ → use median
- E.g., straight line on log-log axes:
  - $y = C\,x^{-\alpha}$ ↔ $\log(y) = \log(C) - \alpha \log(x)$

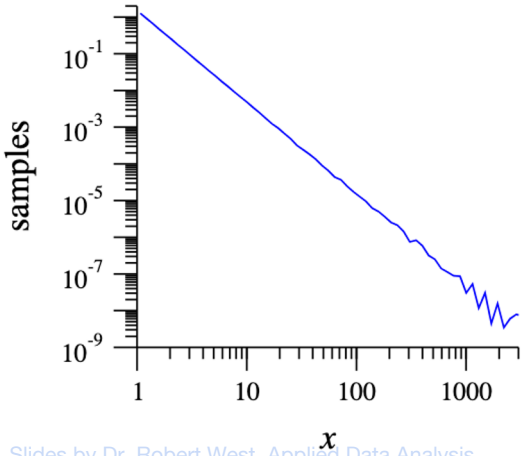




Heavy-tailed data: power laws

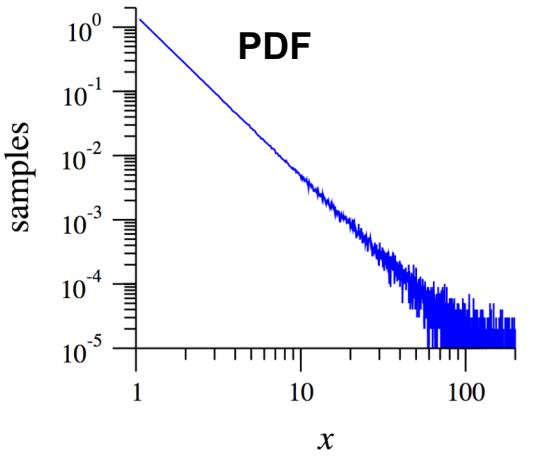
- Complementary cumulative distribution function (CCDF):
  - P(x) := Pr{X >= x}

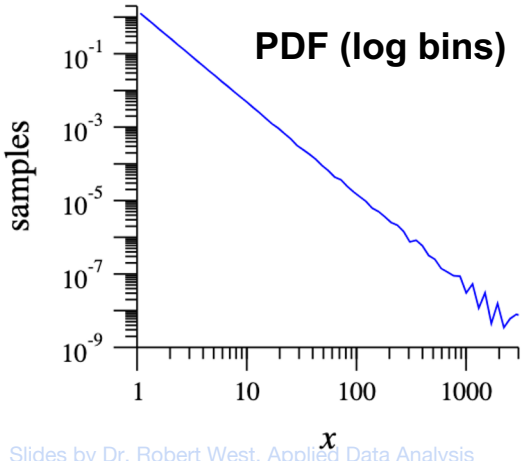


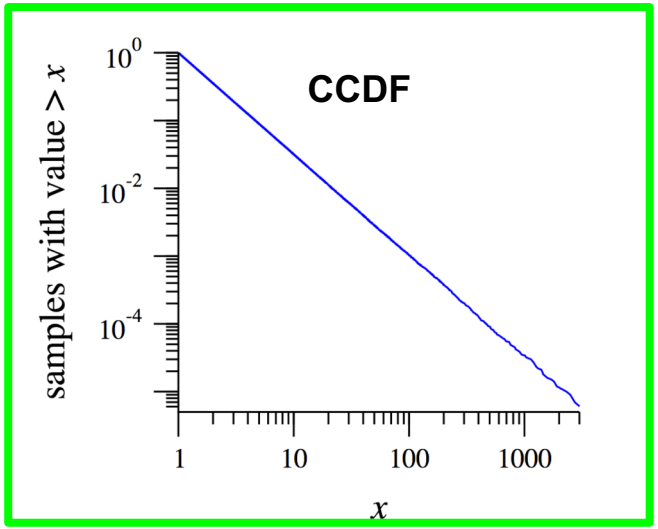


Heavy-tailed data: power laws

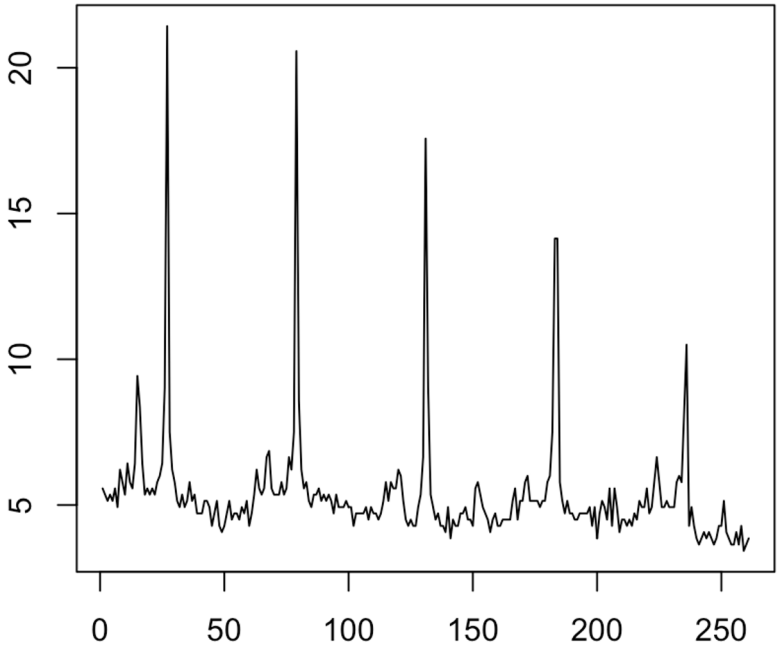
- Complementary cumulative distribution function (CCDF):
  - P(x) := Pr{X >= x}
- CCDF of power law is also a power law (with exponent $\alpha - 1$)
  - $$P(x) = C \int_x^\infty x'^{-\alpha}\, \mathrm{d}x' = \frac{C}{\alpha - 1} x^{-(\alpha-1)}.$$

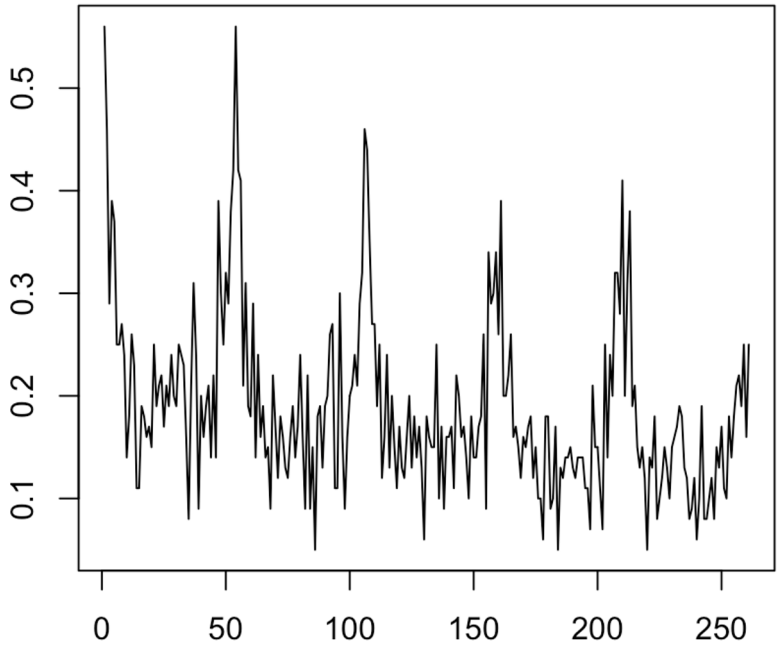






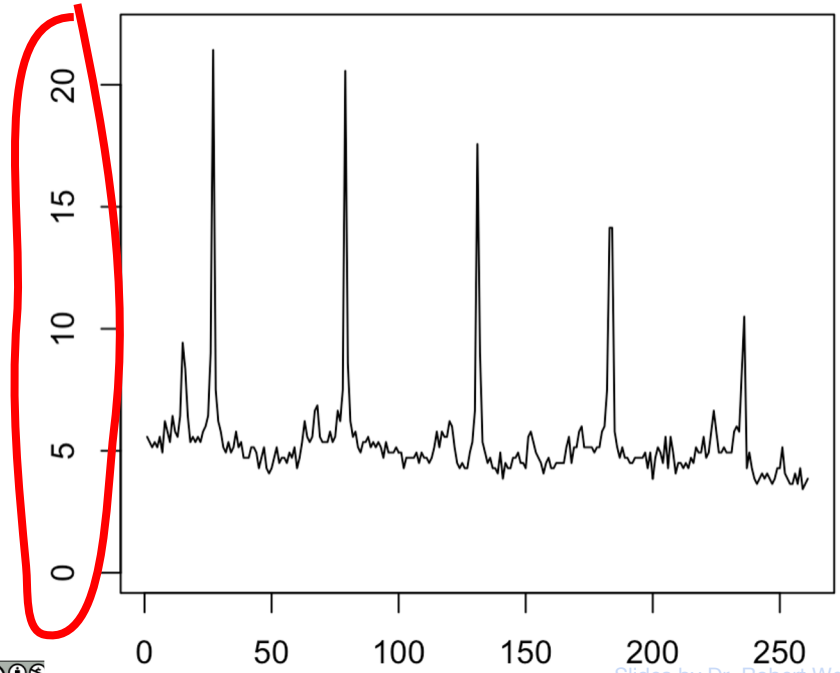
Answer fast: which time series has a higher mean value?

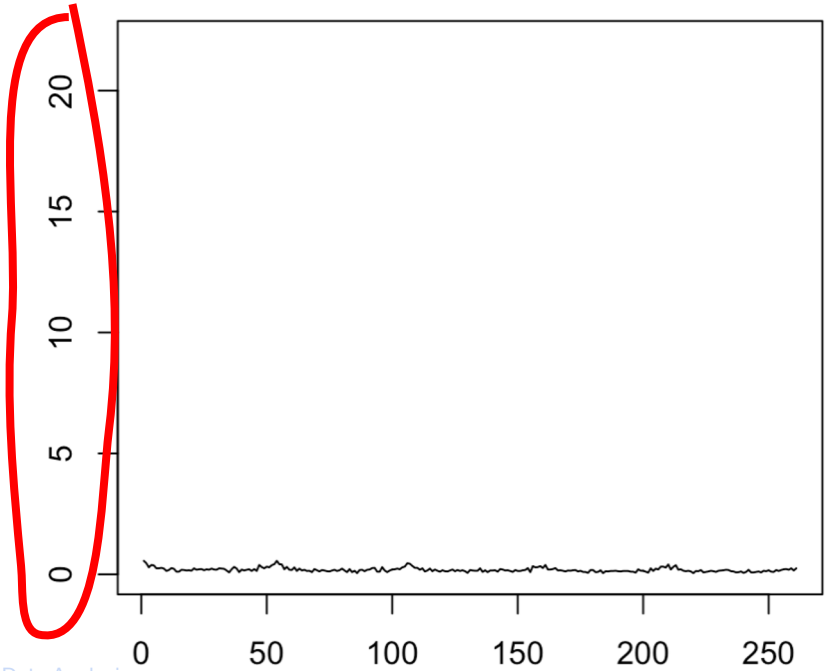




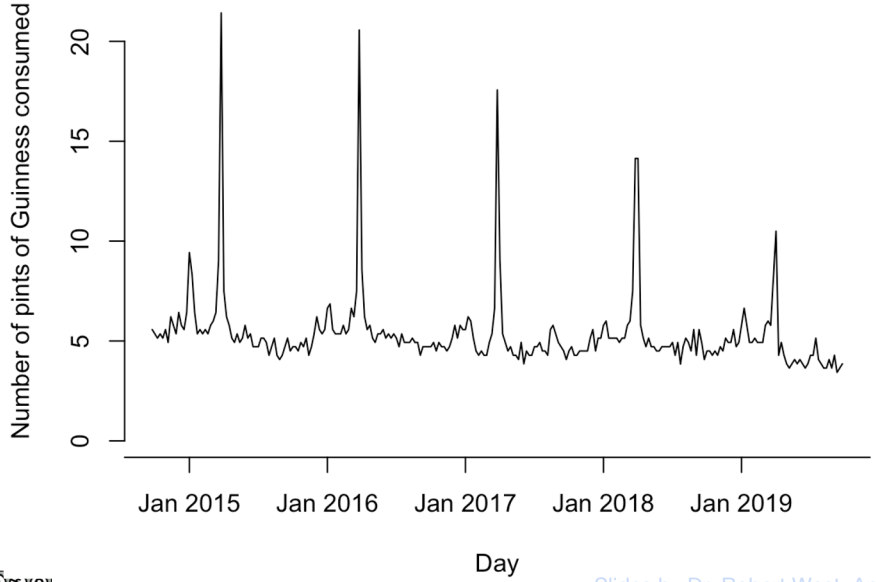
Use consistent axes!

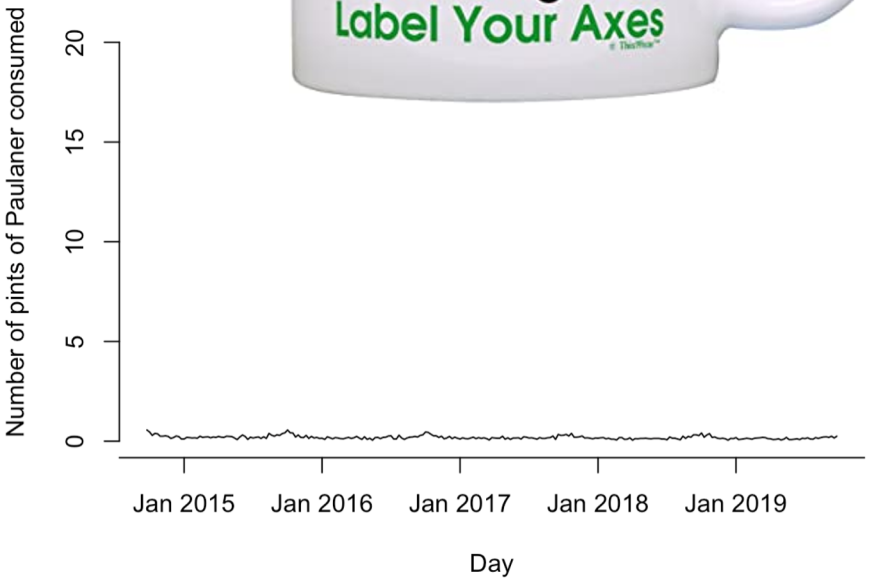
Answer fast: which time series has a higher mean value?

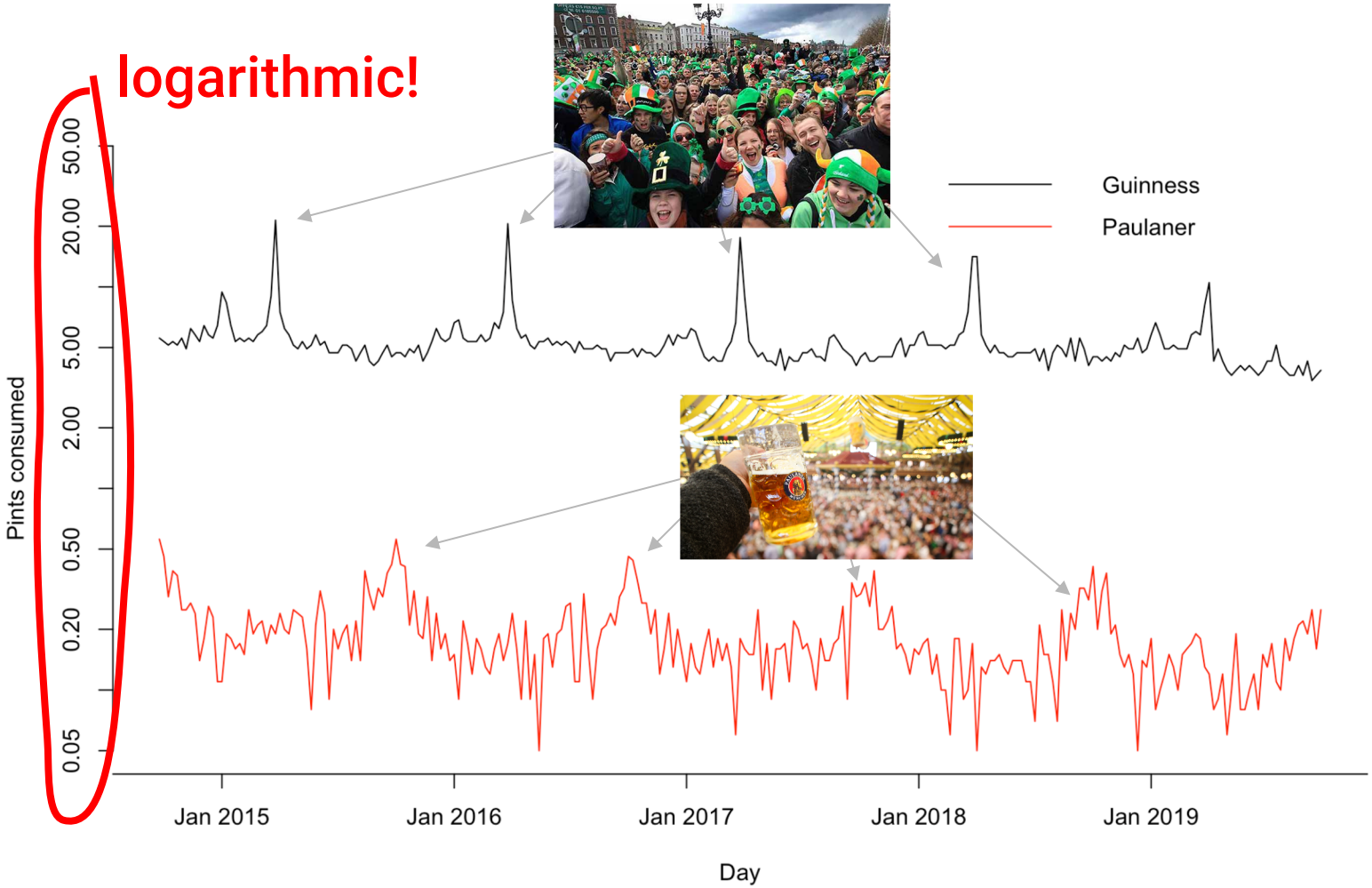




Label your axes

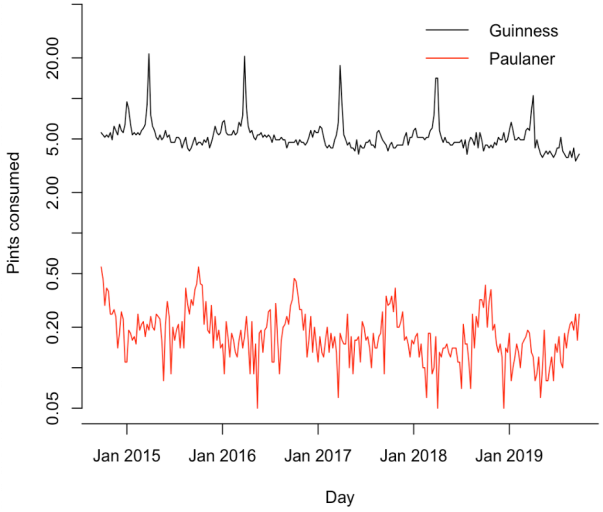


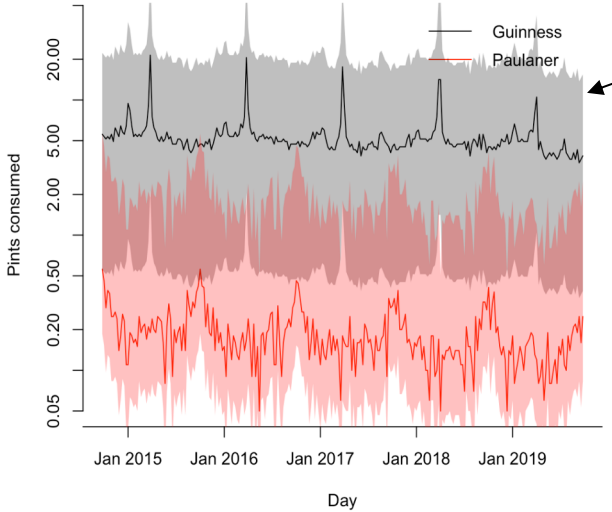



logarithmic!
Guinness
Paulaner
Pints consumed
50.00
20.00
5.00
2.00
0.50
0.20
0.05
Jan 2015
Jan 2016
Jan 2017
Jan 2018
Jan 2019
Day

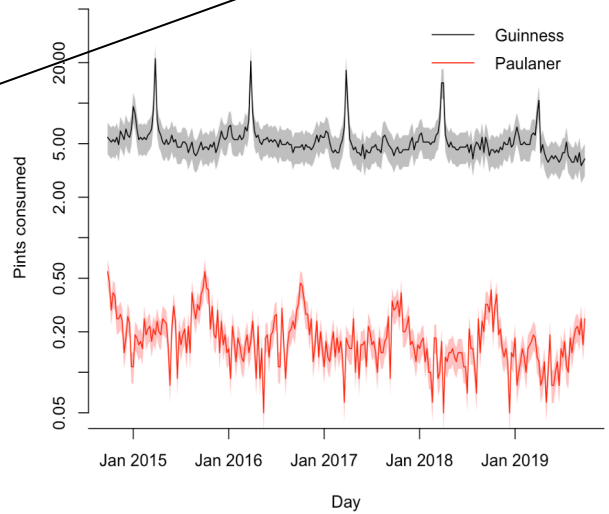
Show data uncertainty!

Which beer is more popular, Guinness or Paulaner?

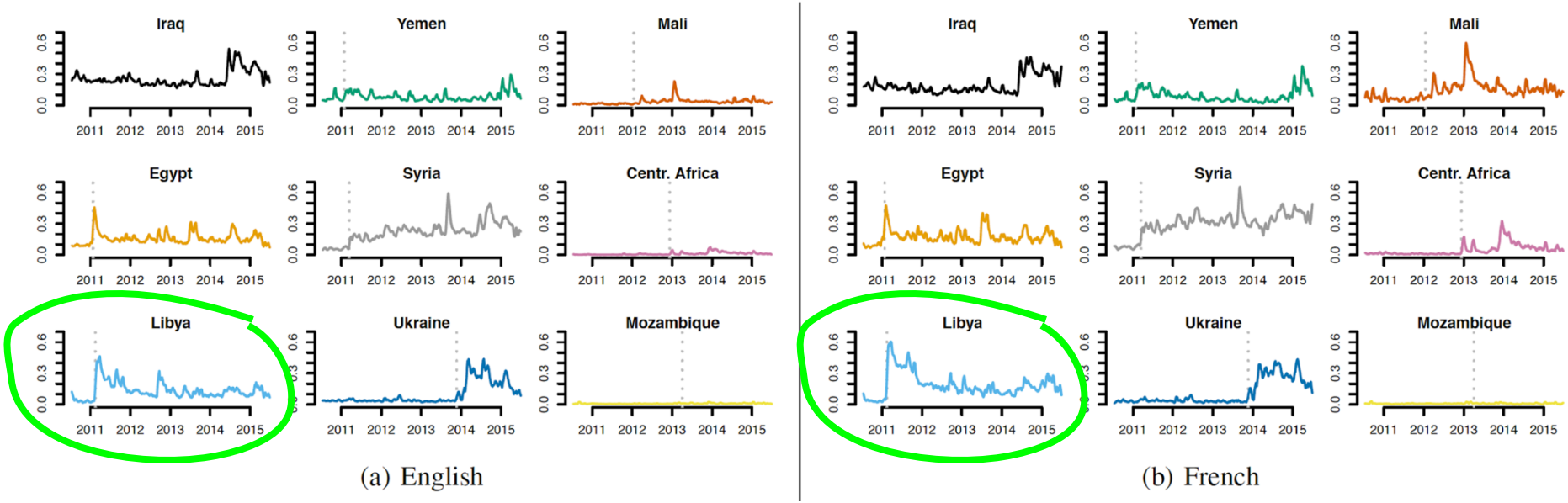




"error bars"



Consider using small multiples!



Use colors consistently!

Use colors wisely!

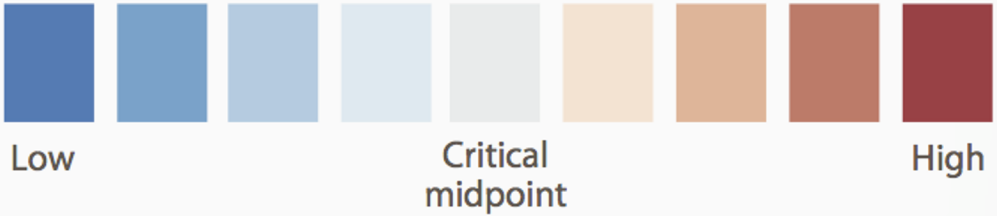
Choose colors based on the information you want to convey

Sequential

Colors can be ordered from low to high

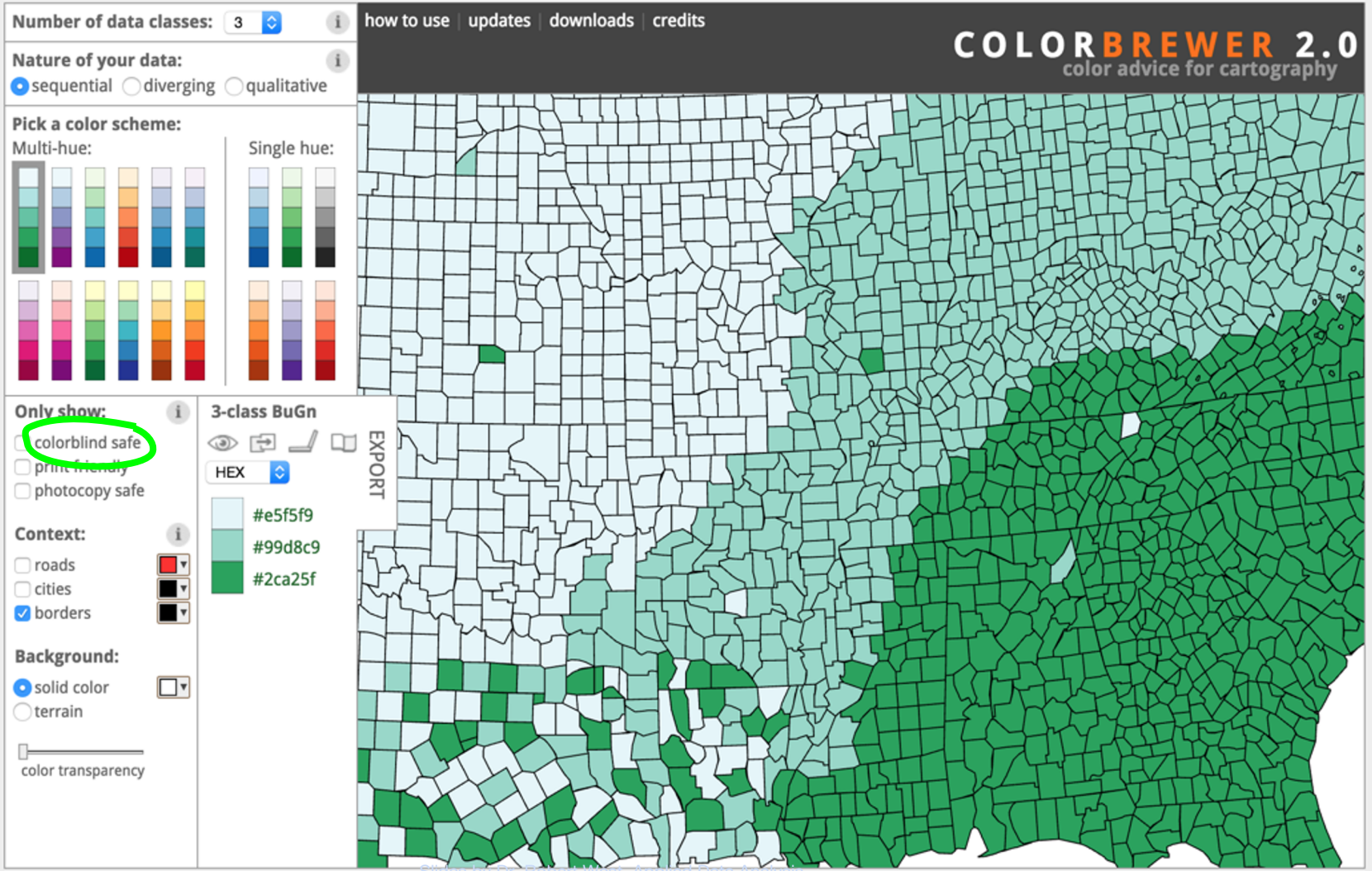
Diverging

Two sequential schemes extended out from a critical midpoint value



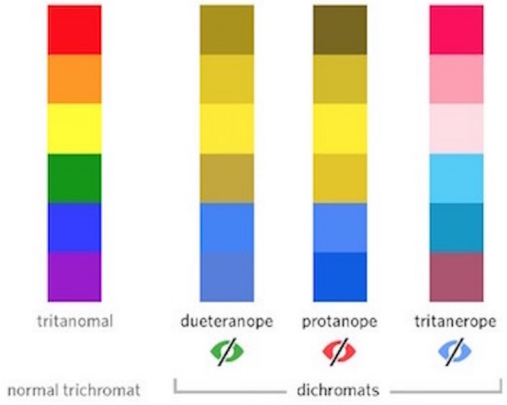
Categorical

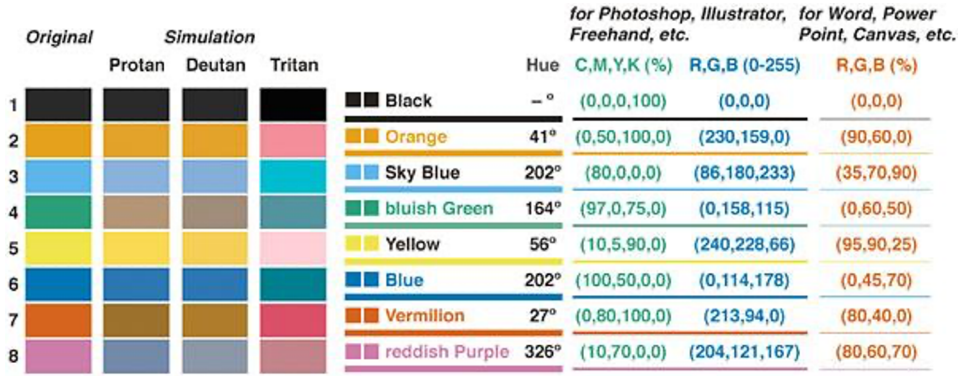
Lots of contrast between each adjacent color



Use colorblind-safe palettes!

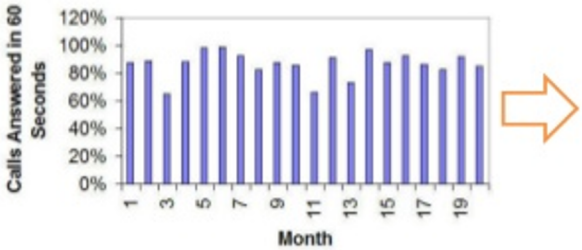
- 10% of males (i.e., ~10% of your reviewers...) have some form of colorblindness

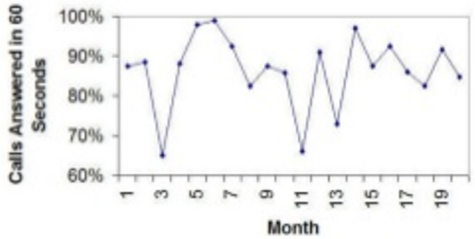




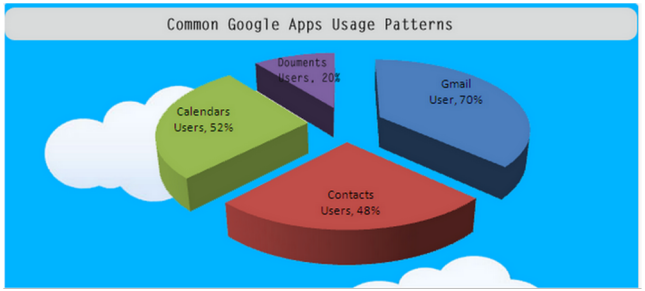
Use data ink wisely! Avoid chart junk!

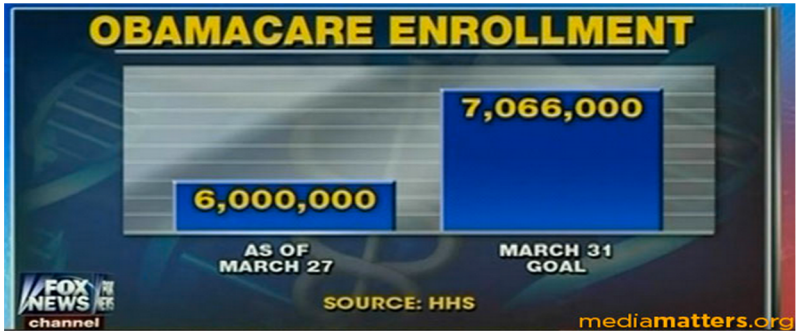
Graphical excellence gives the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space.
-Edward Tufte

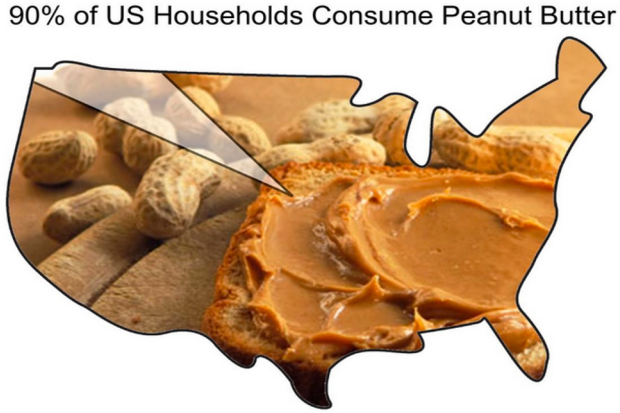


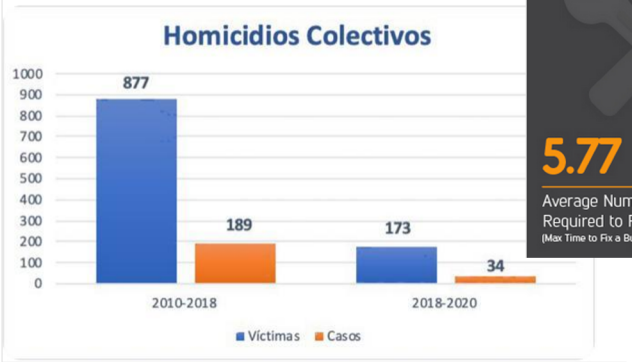


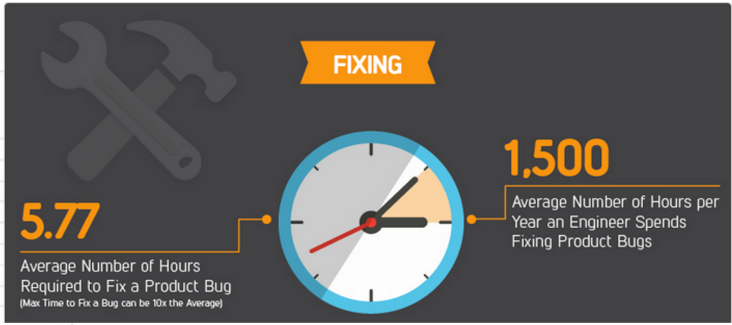
Which principles and best practices do these graphics violate?







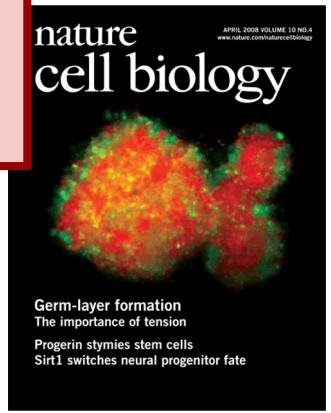


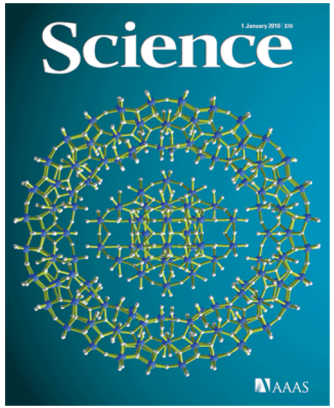


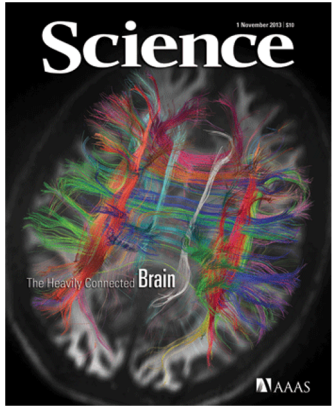
Courtesy of viz.wtf

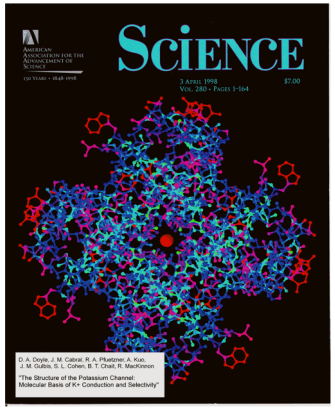
Part 3
A (small) selection of use cases for data visualization

Use case: Presenting scientific results





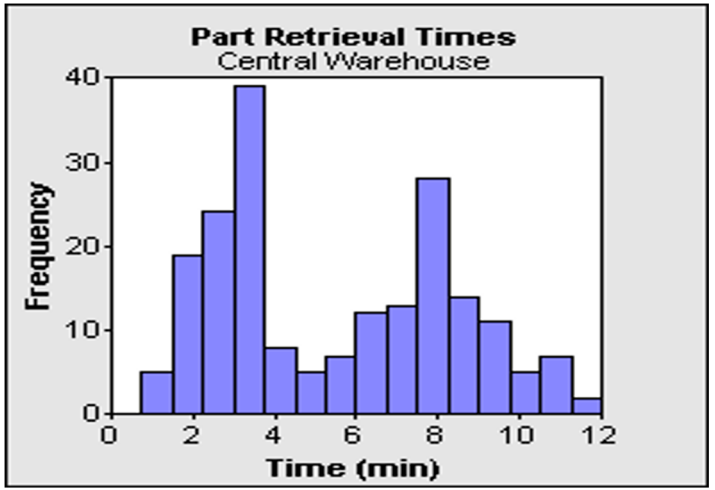




Use case: Data wrangling

Multimodal data

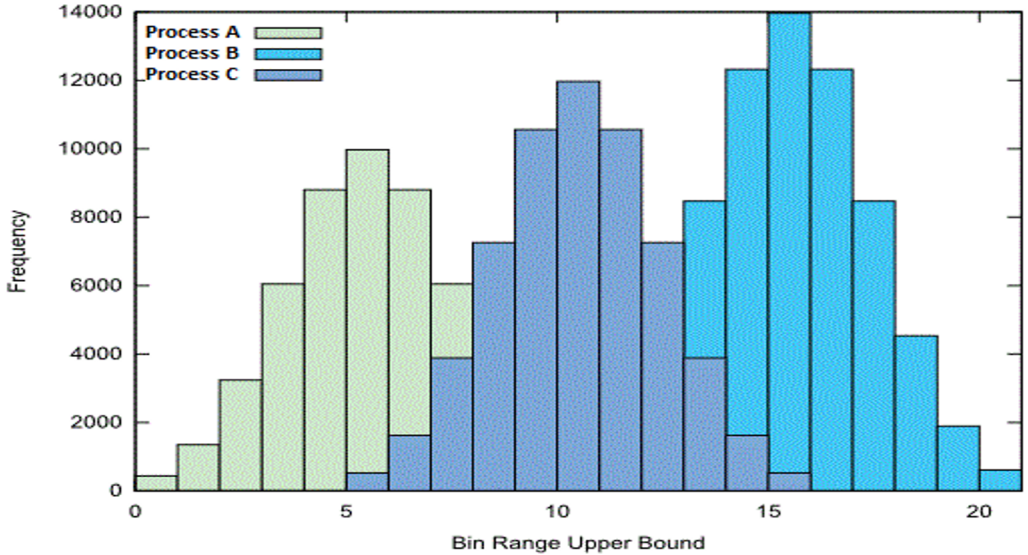
- Two or more distinct peaks in a histogram often **suggest** 2 **or more distinct populations** of samples.
- Often arises from binary factors such as gender or political views.
- But don't guess! Explore further by using, e.g., color and a histogram of multiple populations (p.t.o.).



Use case:
Data wrangling

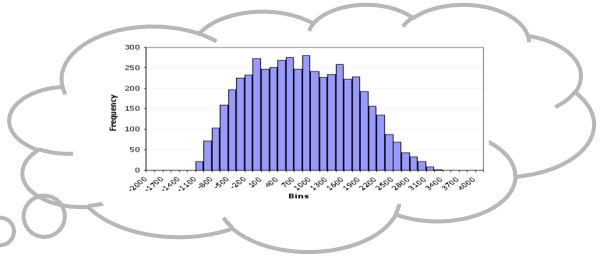
Multimodal data

Explore further by using, e.g., color and a histogram of multiple populations

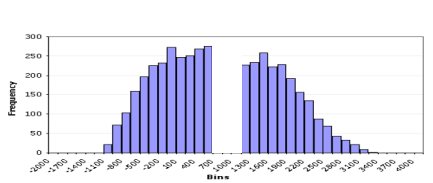


Use case:
Data wrangling

Weird data



- Maintain a theory of what the data should look like.
- Some data is **very hard to explain**.
- **Never just blink it away!**
- First, assume a bug. Try to fix it.
- If not a bug: you might have made an interesting discovery!
- Some of science's most important findings were made by not ignoring weird data, but dwelling on it!



[link]

Use case: Journalism

NY Times interactive visualizations (recession/recovery 2014)
http://www.nytimes.com/interactive/2014/06/05/upshot/how-the-recession-reshaped-the-economy-in-255-charts.html

And 2014 “the year in interactive storytelling”
http://www.nytimes.com/interactive/2014/12/29/us/year-in-interactive-storytelling.html?_r=0

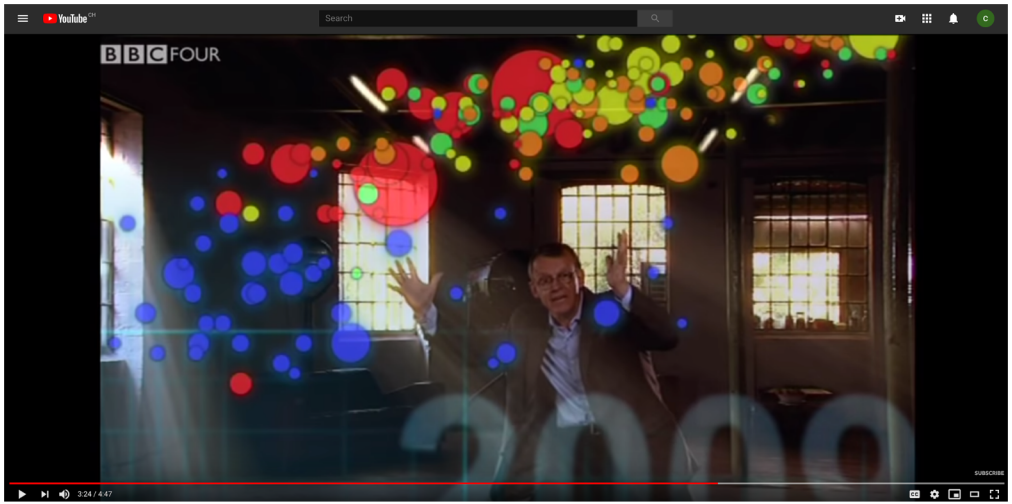
NY Times graphics are a great source of best practices in viz (except for when they’re not…)

Use case: Educating the public

Hans Rosling:

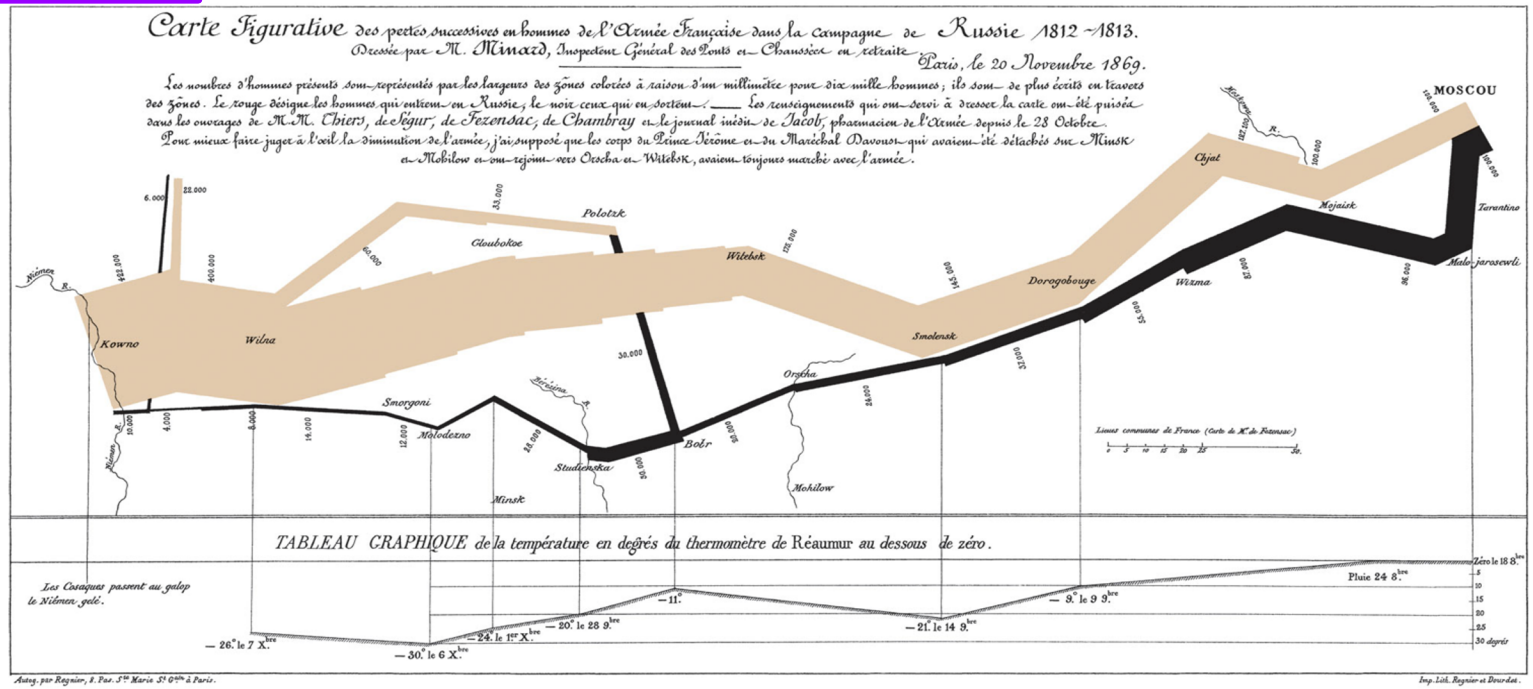
200 countries, 200 years, 4 minutes

https://www.youtube.com/watch?v=jbkSRLYSojo



Use case: Give new perspectives

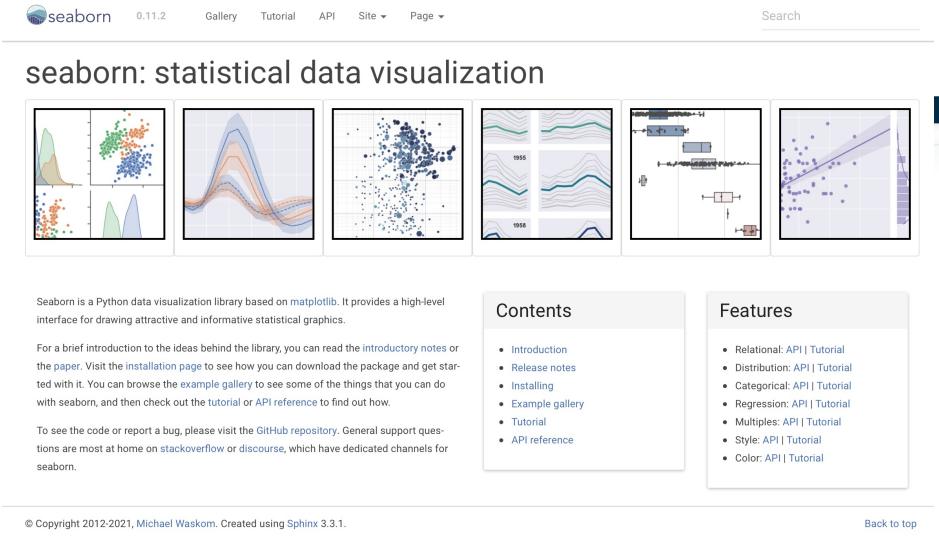
Charles Joseph Minard 1869
Napoleon's march

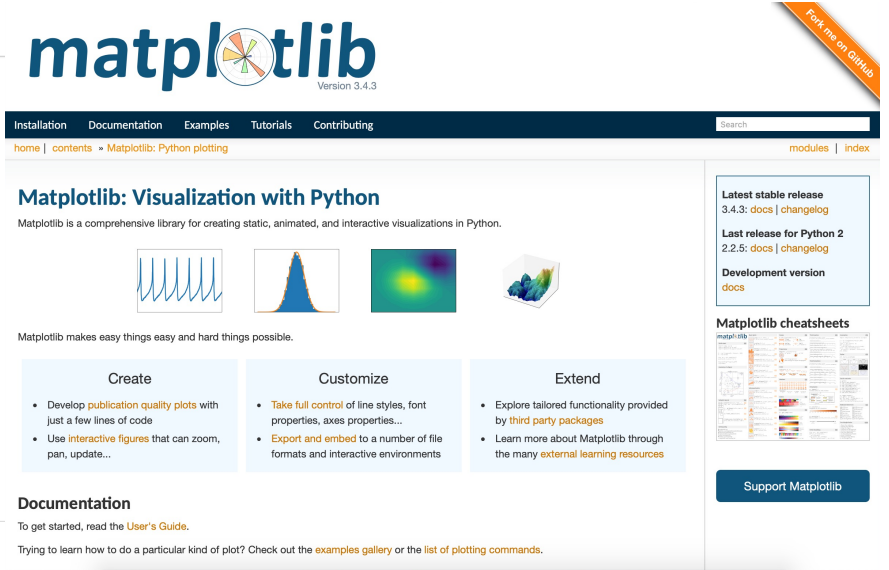


According to Tufte: "*It may well be the best statistical graphic ever drawn.*"
5 variables: army size, location, dates, direction, temperature during retreat

Tools
(remaining slides for your personal perusal)

In the lab you will see





Interactive toolkits: D3

One of the most widely used interactive visualization framework is **D3**.

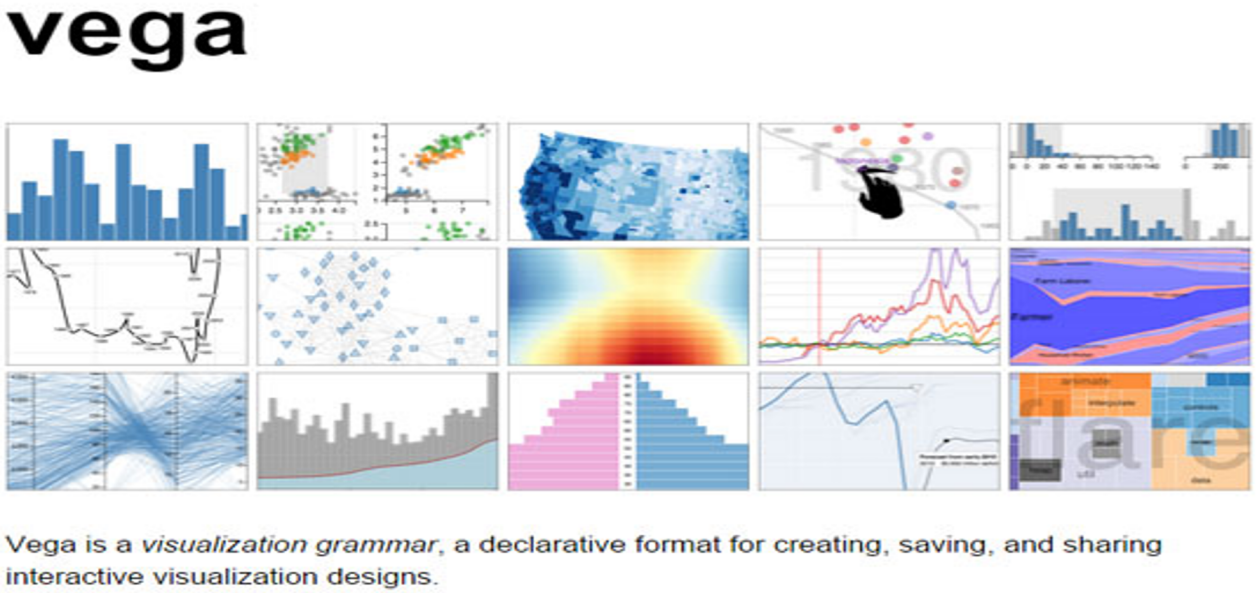
Note from the authors: *D3 is intentionally a low-level system. During the early design of D3, we even referred to it as a "visualization kernel" rather than a "toolkit" or "framework"*

https://d3js.org

Interactive toolkits: Vega

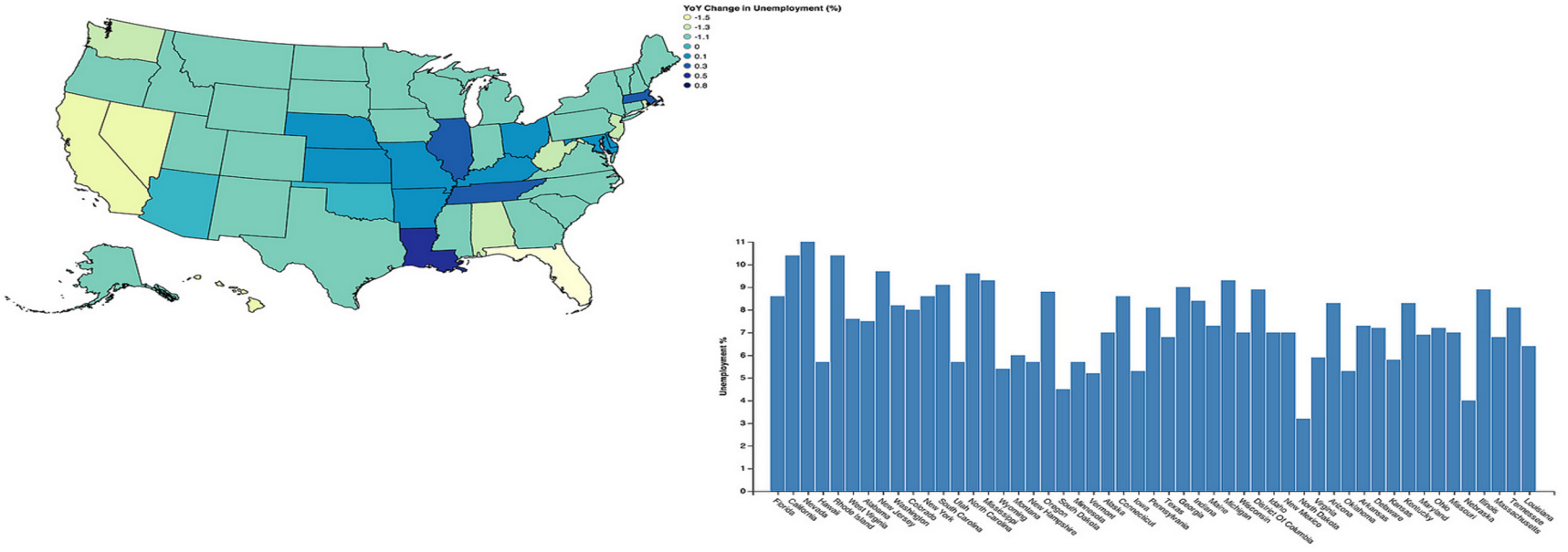
Vega is a “visualization grammar” developed on top of D3.js

It specifies graphics in JSON format.



Interactive toolkits: Vincent

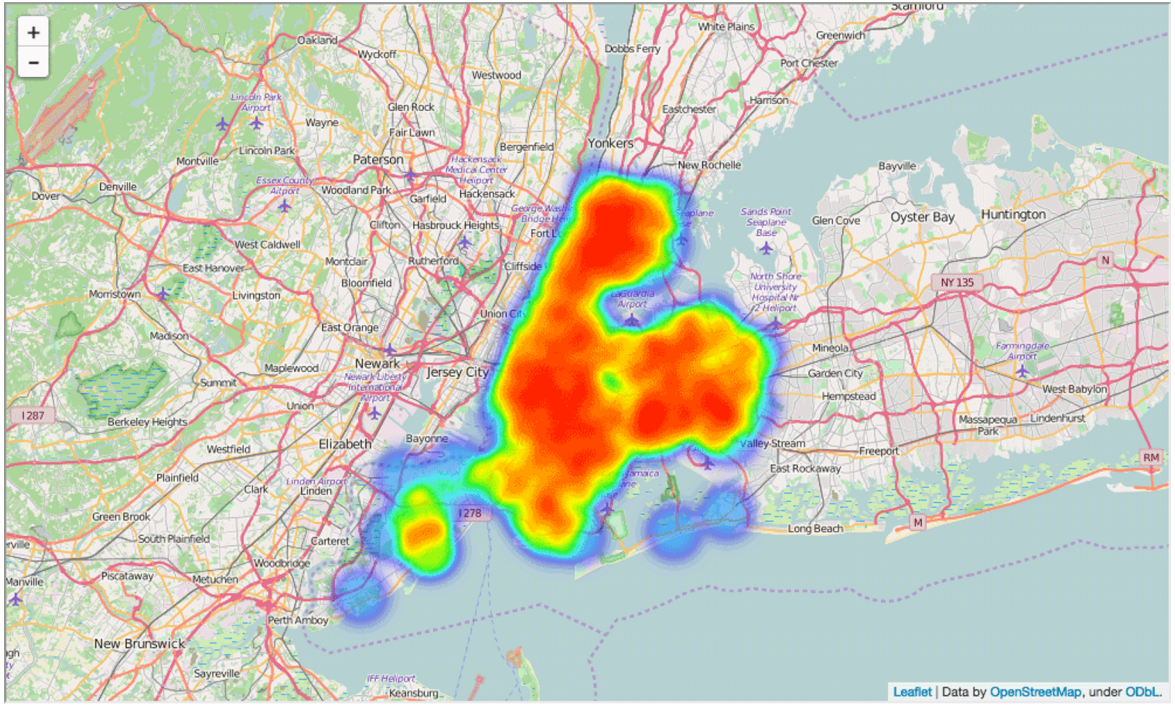
Vincent is a Python-to-Vega translator.



Bokeh: another interactive viz library

Bokeh is an independent Viz library focused more heavily on big data visualization. Has both Python and Scala bindings.

Visualizing maps: Folium



Visualization

Visualizing distributions and relationships; applying transformations and KDEs

Goals for this Lecture

Understand the theories behind effective visualizations and start to generate plots of our own

- The necessary “pre-thinking” before creating a plot
- Python libraries for visualizing data

Agenda

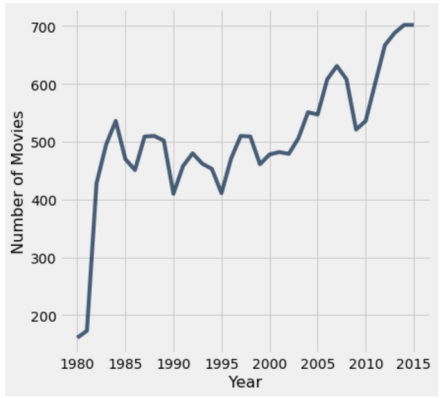
- Goals of visualization
- Visualizing distributions
- Kernel density estimation
- Visualizing relationships
- Transformations

Goals of Visualization

- **Goals of visualization**
- Visualizing distributions
- Kernel density estimation
- Visualizing relationships
- Transformations

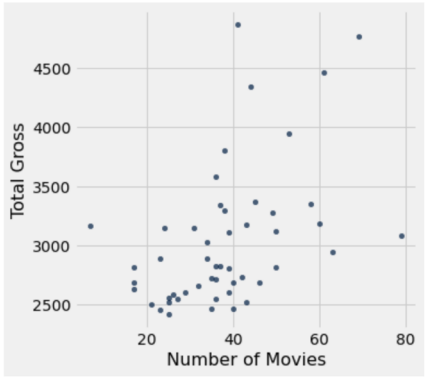
Visualizations in DSC510 so far

```
# The call is
# t.plot(x_label, y_label)

movies.plot('Year', 'Number of Movies')
```



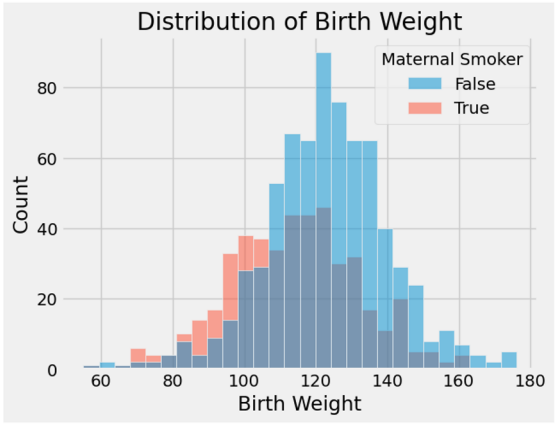
Line plot

```
# The call is
# t.scatter(x_label, y_label)

actors.scatter('Number of Movies', 'Total Gross')
```



Scatter plot

```
import seaborn as sns
sns.histplot(births_df, x="Birth Weight", hue="Maternal Smoker");
plt.title("Distribution of Birth Weight");
```



Histogram from Homework #1

What did these achieve?

- Provide a high-level overview of a complex dataset
- Communicated trends to viewers

Goals of Data Visualization

Goal 1: To **help your own understanding** of your data/results.

- Key part of exploratory data analysis.
- Summarize trends visually before in-depth analysis
- Lightweight, iterative and flexible.

Goal 2: To **communicate results/conclusions to others**.

- Highly editorial and selective.
- Be thoughtful and careful!
- Fine-tuned to achieve a communications goal.
- Considerations: clarity, accessibility, necessary context

What do these goals imply?

Visualizations aren't a matter of making "pretty" pictures.

We need to do a lot of thinking about what stylistic choices communicate ideas most effectively.

Goals of Data Visualization

What do these goals imply?

Visualizations aren't a matter of making "pretty" pictures.

We need to do a lot of thinking about what stylistic choices communicate ideas most effectively.

Choosing the "right" plot

- Introducing plots for different variable types
- Generating these plots through code

Stylizing plots appropriately

- Smoothing and transforming visual data
- Providing context through labeling and color

Visualizing Distributions

- Goals of visualization
- **Visualizing distributions**
- Kernel density estimation
- Visualizing relationships
- Transformations

Distributions

A distribution describes…

- The set of values that a variable can possibly take
- The frequency with which each value occurs

…for a **single** variable

Example: distribution of students across discussion sections in DSC 510

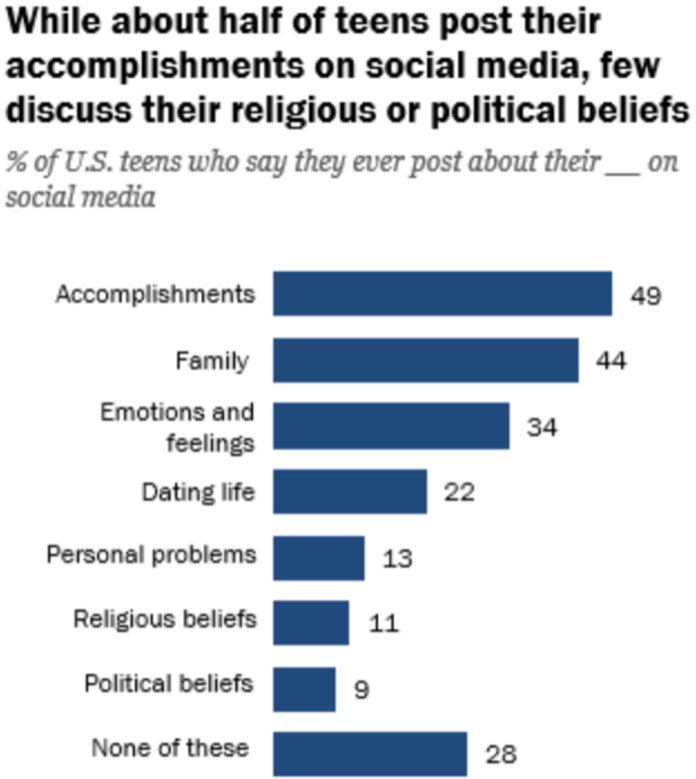
- The list of discussion sections (10-11 am, 11-12 pm, etc.)
- The number of students enrolled in each section

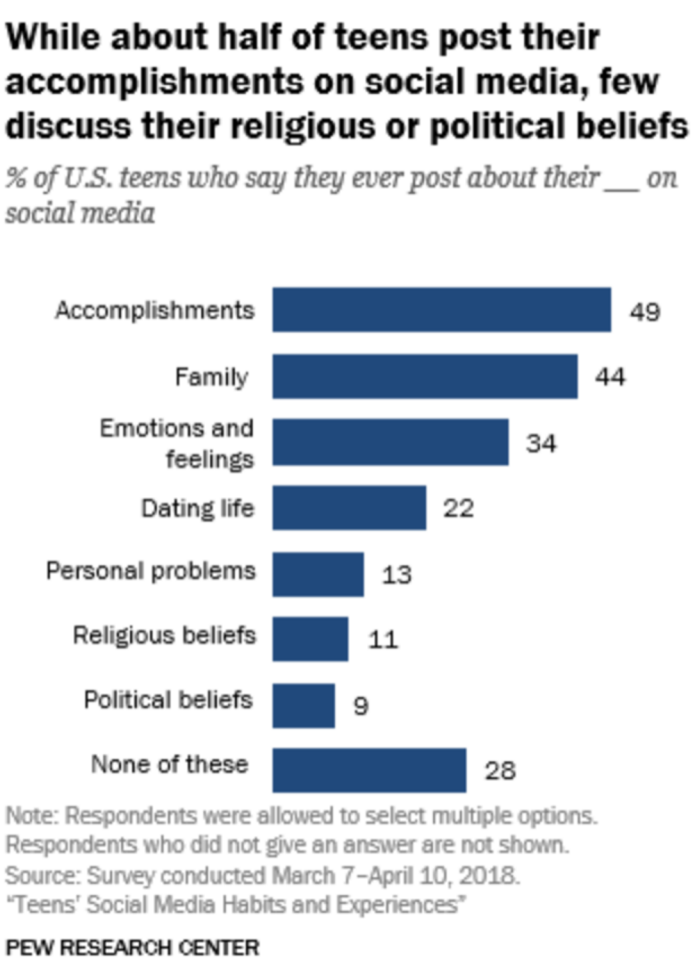
In other words: how is the variable distributed across all out its possible values?

This means that percentages **should sum to 100%** (if using proportions) and counts should **sum to the total number of datapoints** (if using raw counts).

Let's see some examples.

Does this chart show a distribution?



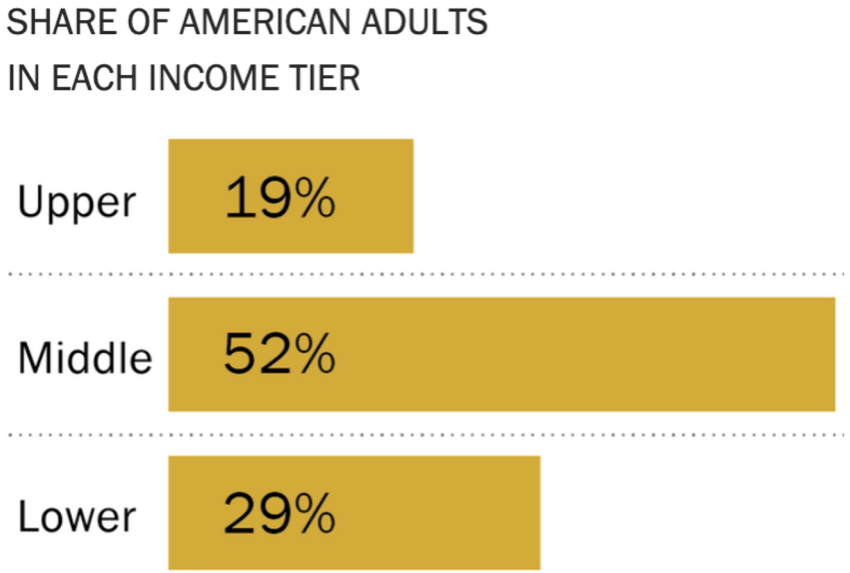


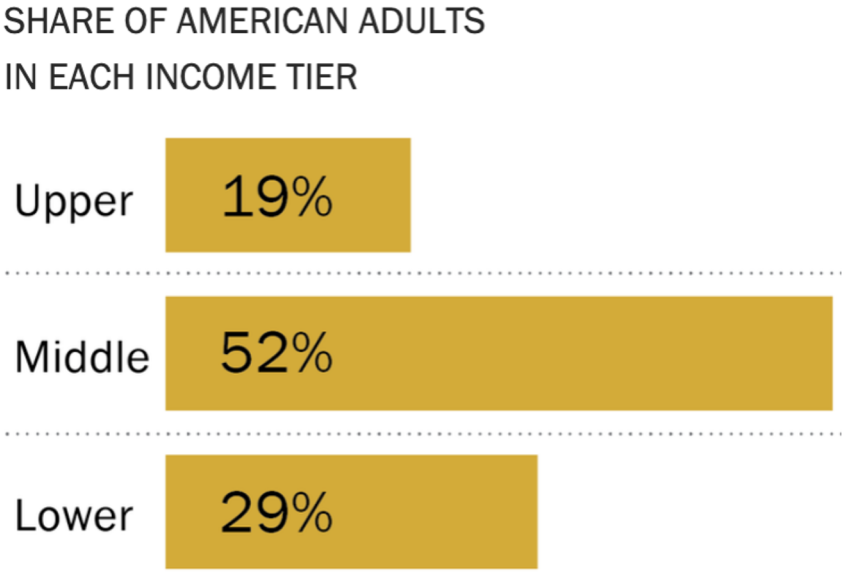
Does this chart show a distribution?

No.

- The chart does show percents of individuals in different categories!
- But, this is not a distribution because individuals can be in more than one category (see the fine print).

Does this chart show a distribution?





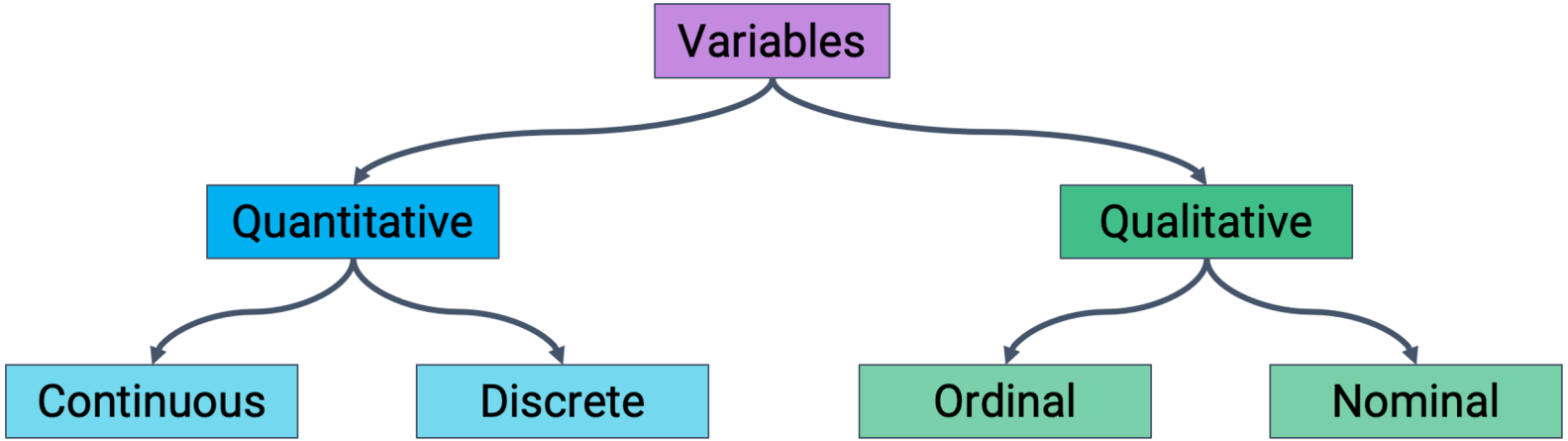
Does this chart show a distribution?

Yes!

- This chart shows the distribution of the qualitative ordinal variable "income tier."
- Each individual is in exactly one category.
- The values we see are the proportions of individuals in that category.
- Everyone is represented, as the total percentage is 100%.

Variable types should inform plot choice

Different plots are more or less suited for displaying particular types of variables

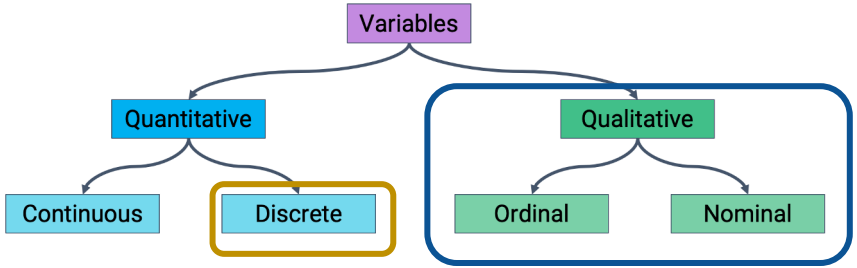


First step of visualization: identify the variables being visualized. Then, select a plot type accordingly.

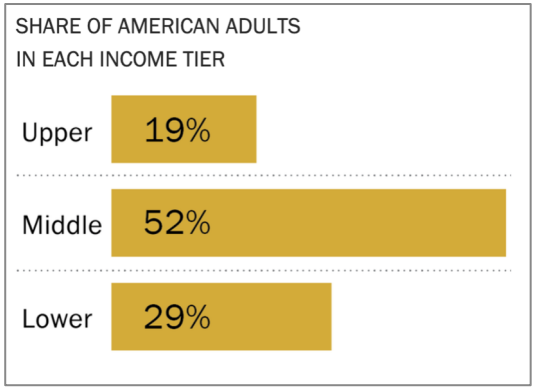
Bar plots: distributions of qualitative variables

Bar plots are the most common way of displaying the **distribution** of a **qualitative** variable.

*Sometimes quantitative discrete data too, if there are few unique values.



- For example, the proportion of adults in the upper, middle, and lower classes.
- Lengths encode values.
  - *Widths* encode *nothing*!
  - *Color* could indicate a sub-category (but not necessarily).



World Bank Dataset

We will be using the **`wb`** dataset about world countries for most of our work today.

	Continent	Country	Primary completion rate: Male: % of relevant age group: 2015	Primary completion rate: Female: % of relevant age group: 2015	Lower secondary completion rate: Male: % of relevant age group: 2015	Lower secondary completion rate: Female: % of relevant age group: 2015	Youth literacy rate: Male: % of ages 15-24: 2005-14	Youth literacy rate: Female: % of ages 15-24: 2005-14	Adult literacy rate: Male: % ages 15 and older: 2005-14	Adult literacy rate: Female: % ages 15 and older: 2005-14
0	Africa	Algeria	106.0	105.0	68.0	85.0	96.0	92.0	83.0	68.0
1	Africa	Angola	NaN	NaN	NaN	NaN	79.0	67.0	82.0	60.0
2	Africa	Benin	83.0	73.0	50.0	37.0	55.0	31.0	41.0	18.0
3	Africa	Botswana	98.0	101.0	86.0	87.0	96.0	99.0	87.0	89.0
5	Africa	Burundi	58.0	66.0	35.0	30.0	90.0	88.0	89.0	85.0

Generating bar plots: Matplotlib

We will mainly use two libraries for generating plots: Matplotlib and Seaborn.

Most Matplotlib plotting functions follow the same structure: we pass in a sequence (list, array, or Series) of values to be plotted on the x-axis, and a second sequence of values to be plotted on the y-axis.

```
import matplotlib.pyplot as plt
plt.plotting_function(x_values, y_values)
```

Matplotlib is typically given the alias `plt`

To add labels and a title:

```
plt.xlabel("x axis label")
plt.ylabel("y axis label")
plt.title("Title of the plot");
```

Generating bar plots: Matplotlib

To create a bar plot in Matplotlib: `plt.bar( )`

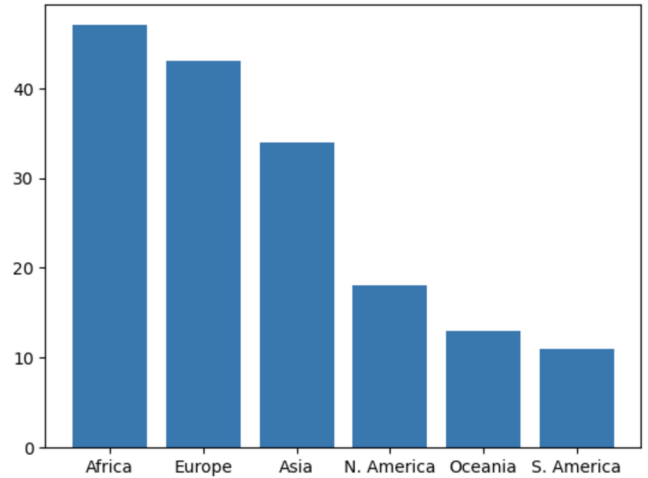
```
continents = wb["Continent"].value_counts()
```

```
Africa          47
Europe          43
Asia            34
N. America      18
Oceania         13
S. America      11
Name: Continent, dtype: int64
```

```
plt.bar(continents.index, continents.values);
```

x values

y values



Generating bar plots: Seaborn

Seaborn plotting functions use a different structure: pass in an entire *DataFrame*, then specify what column(s) to plot.

Seaborn is typically given the alias `sns`

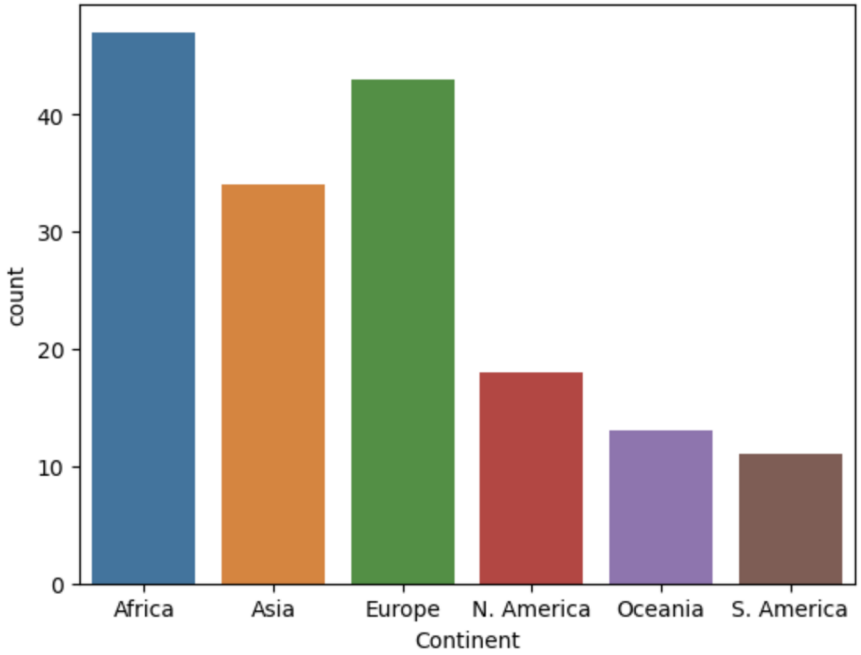
```
import seaborn as sns
sns.plotting_function(data=df, x="x_col", y="y_col")
```

To add labels and a title, use the same syntax as before:

```
plt.xlabel("x axis label")
plt.ylabel("y axis label")
plt.title("Title of the plot");
```

Generating bar plots: Seaborn

To create a bar plot in Seaborn: `sns.countplot( )`



`countplot` operates at a higher level of abstraction!

You give it the entire DataFrame and it does the counting for you.

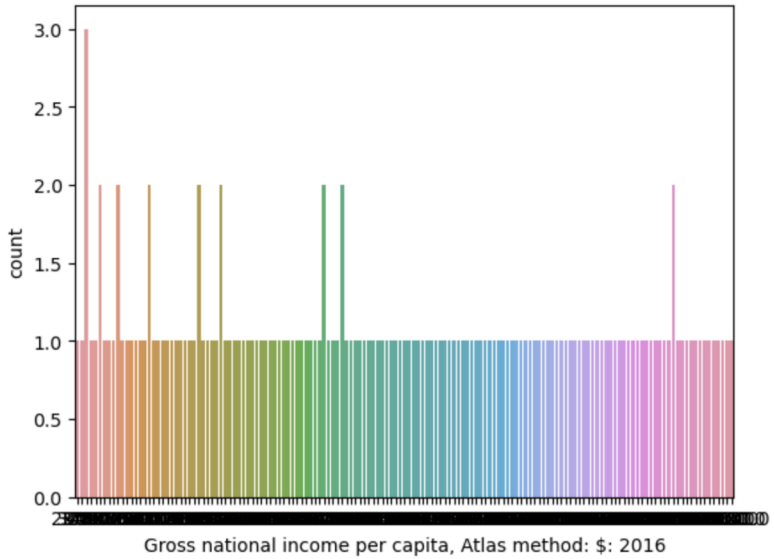
```
import seaborn as sns

sns.countplot(data = wb, x = 'Continent');
```

Distributions of quantitative variables

Earlier, we said that bar plots are appropriate for distributions of qualitative variables.

Why only qualitative? Why not quantitative as well?

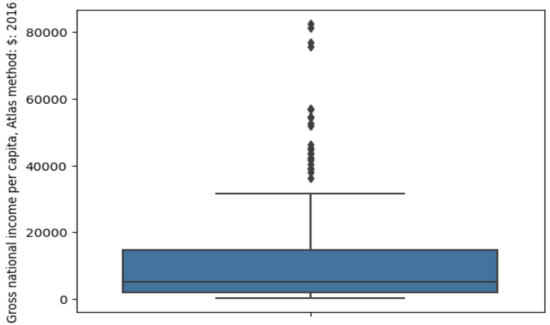
- For example: the distribution of gross national income per capita



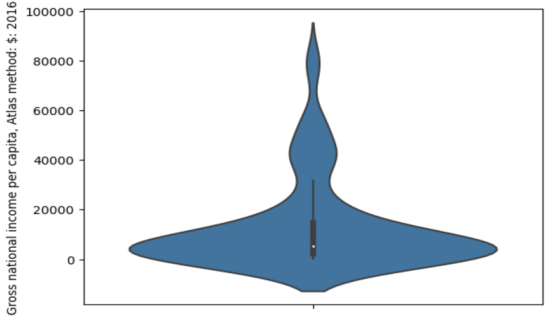
A bar plot will create a separate bar for each unique value. This leads to too many bars for continuous data!

Distributions of quantitative variables

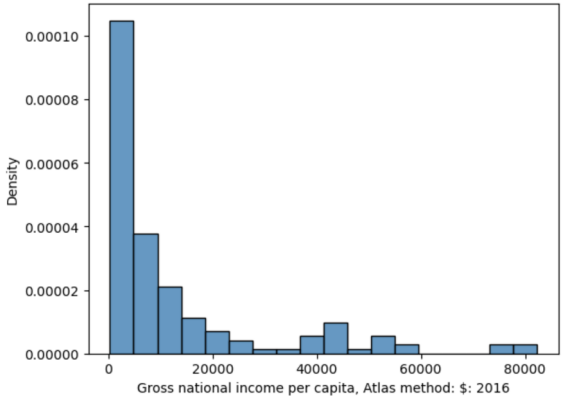
To visualize the distribution of a continuous quantitative variable:



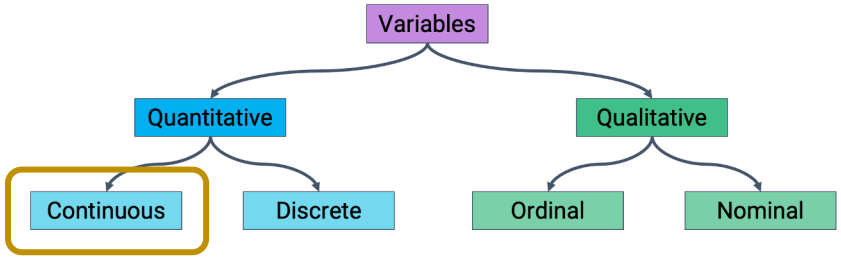
Box plot



Violin plot



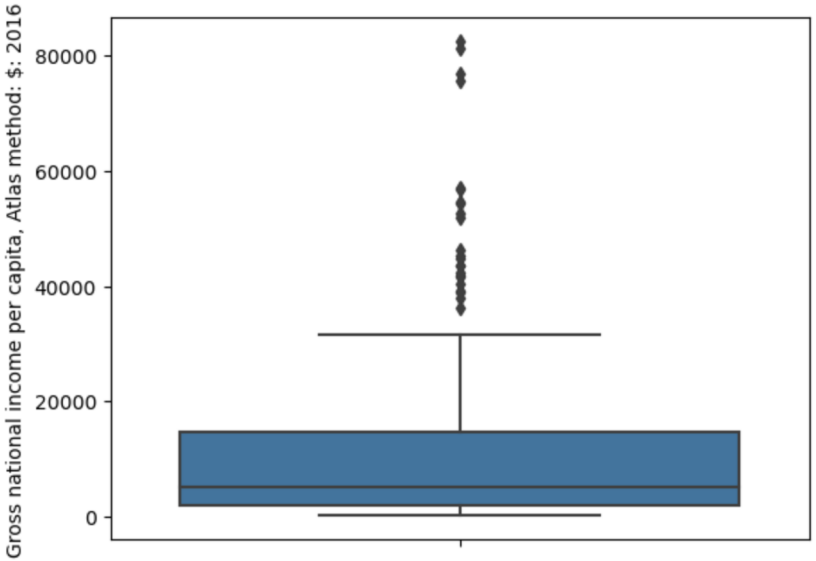
Histogram

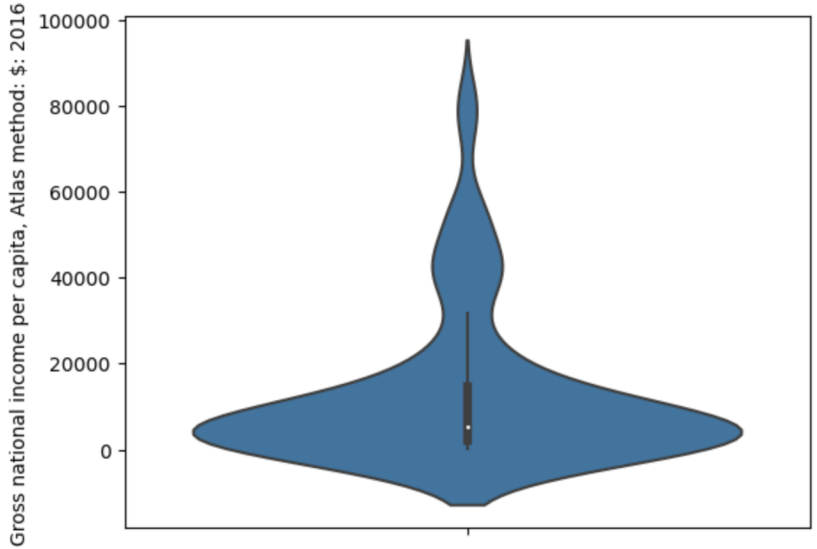


Box plots and violin plots

Box plots and violin plots display distributions using information about **quartiles**.

- In a box plot, the width of the box encodes no meaning
- In a violin plot, the width of the "violin" indicates the density of datapoints at each value






```
sns.boxplot(data = df, y = "y_variable");
```

```
sns.violinplot(data = df, y = "y_variable");
```

Quartiles

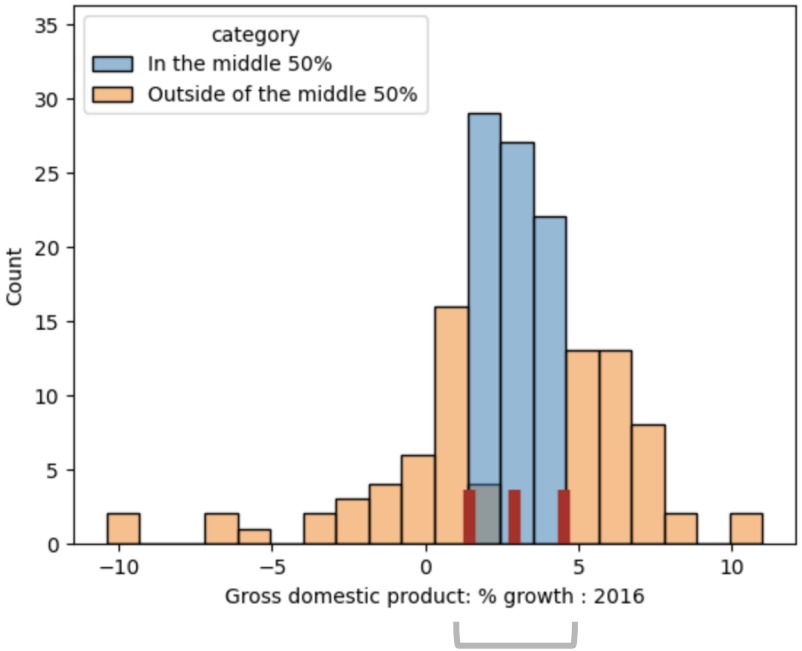
For a quantitative variable:

- First or lower quartile: 25th percentile
- Second quartile: 50th percentile (median)
- Third or upper quartile: 75th percentile

The interval [first quartile, third quartile] contains the "middle 50%" of the data.

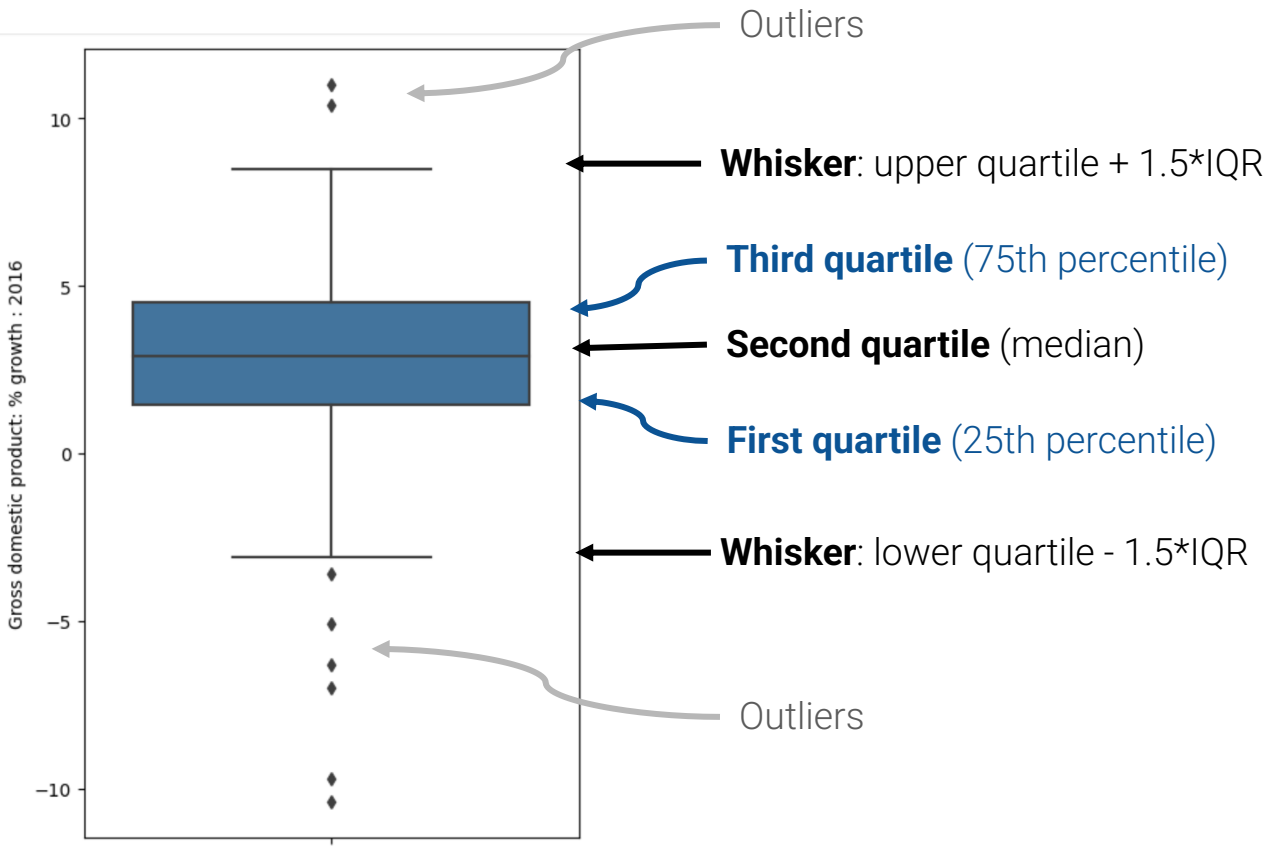
Interquartile range (IQR) measures spread.

- IQR = third quartile – first quartile.



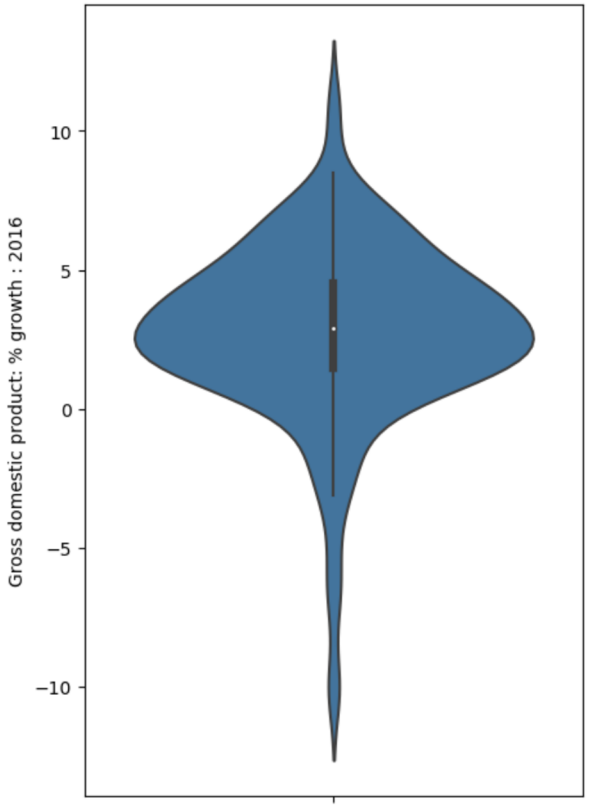
The length of this region is the IQR

Box Plots




```
sns.boxplot(data = wb, y = 'Gross domestic product: % growth : 2016')
```

Violin Plots



Violin plots are similar to box plots, but also show smoothed density curves.

- The "width" of our "box" now has meaning!
- The three quartiles and "whiskers" are still present – look closely.

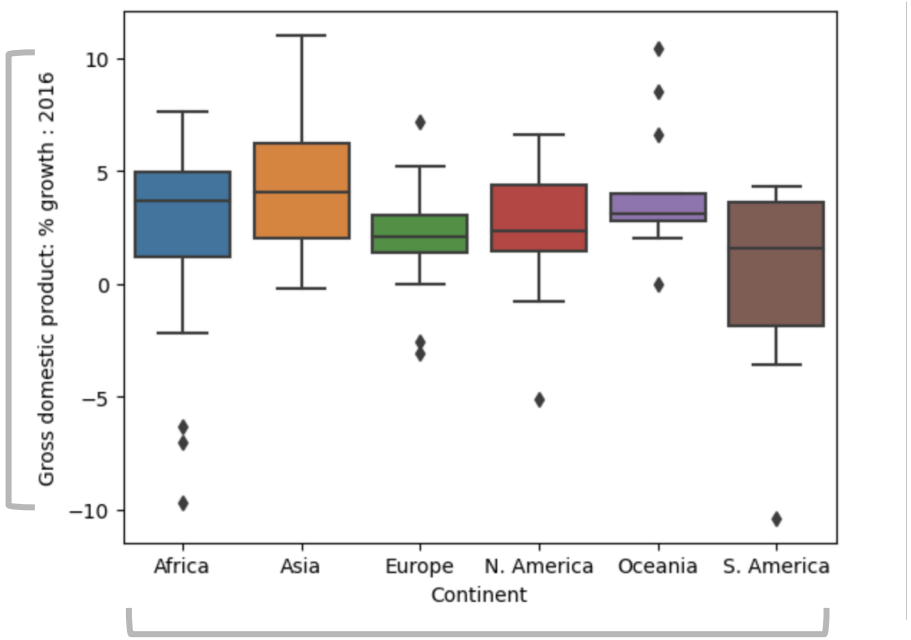
```
sns.violinplot(data = wb, y = 'Gross domestic product: % growth : 2016')
```

Side-by-side box and violin plots

What if we wanted to incorporate a *qualitative* variable as well? For example, compare the distribution of a quantitative continuous variable *across* different qualitative categories.

```
sns.boxplot(data=wb, x= “Continent”, y= “Gross domestic product: % growth : 2016”);
```

GDP growth: quantitative continuous

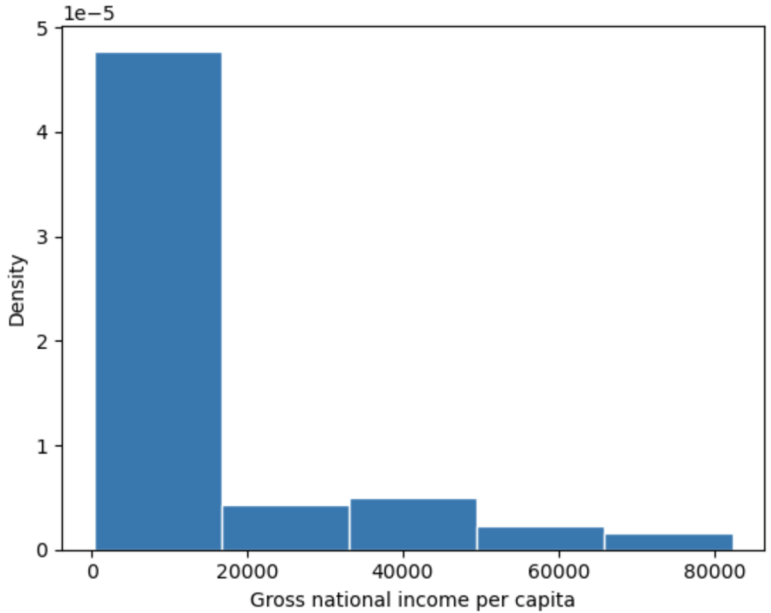


Continent: qualitative nominal

Histograms

A histogram:

- Collects datapoints with similar values into a shared “bin”
- Scales the bins such that the **area** of each bin is equal to the **percentage** of datapoints it contains



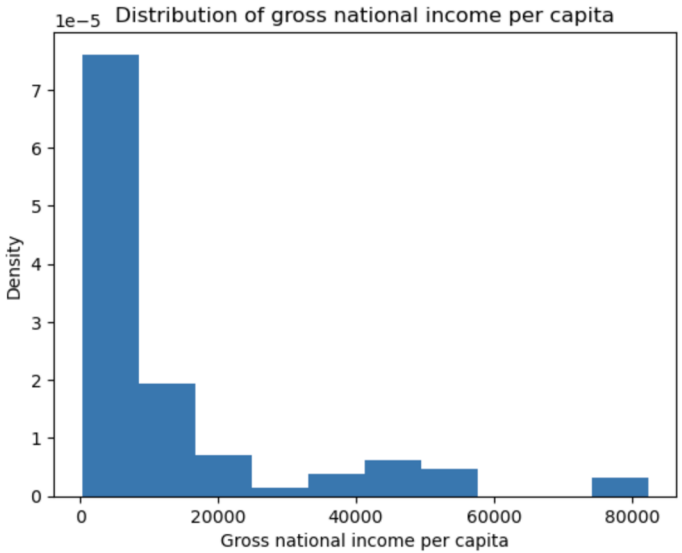
The first bin has a width of $16410
height of 4.77×10^5

This means that it contains $16410 \times (4.77 \times 10^5) = 78.3\%$ of all datapoints in the dataset

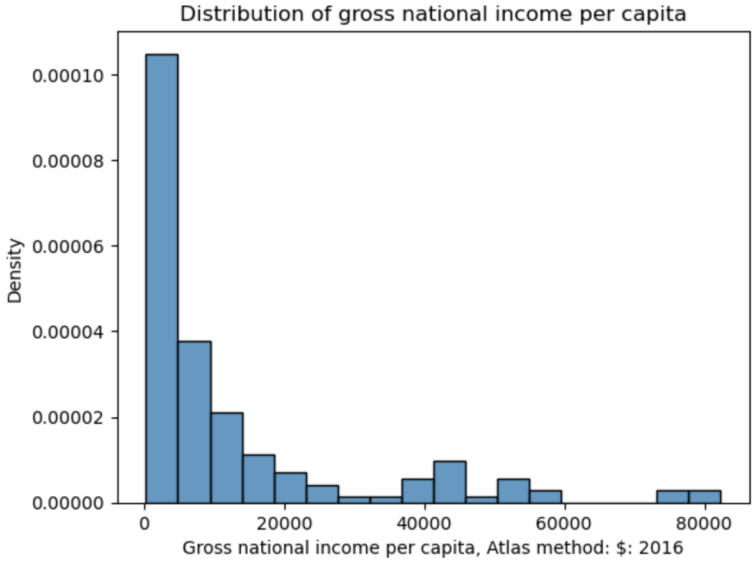
Histograms in code

In Matplotlib: `plt.hist(x_values, density = True)`

In Seaborn: `sns.histplot(data = df, x = "x_column", stat = "density")`



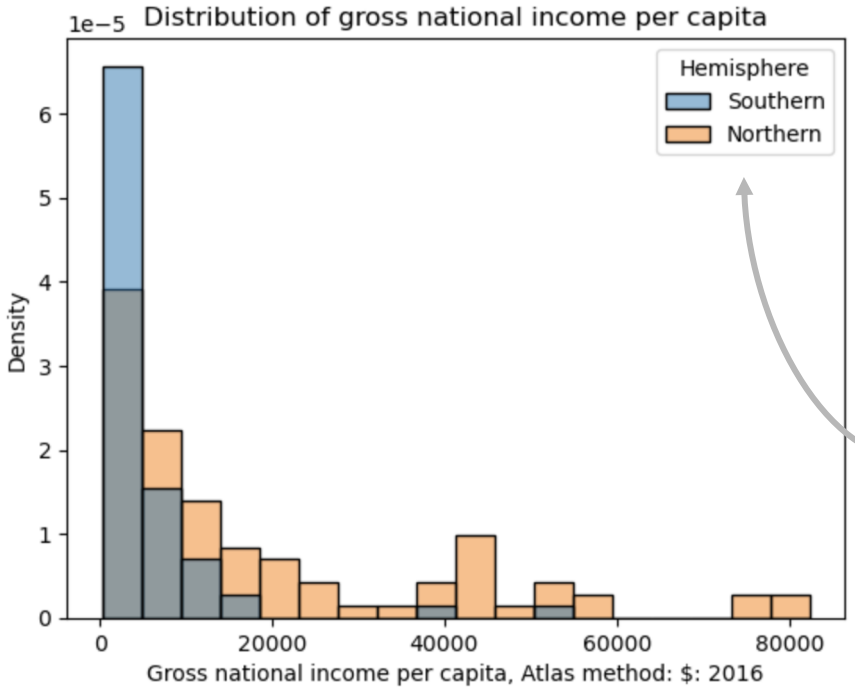
Matplotlib



Seaborn

Overlaid histograms

To compare a quantitative variable's distribution across qualitative categories, overlay histograms on top of one another.



The **hue** parameter of Seaborn plotting functions sets the column that should be used to determine color.

```
sns.histplot(data = wb, hue="Hemisphere",
x="Gross national income…")
```

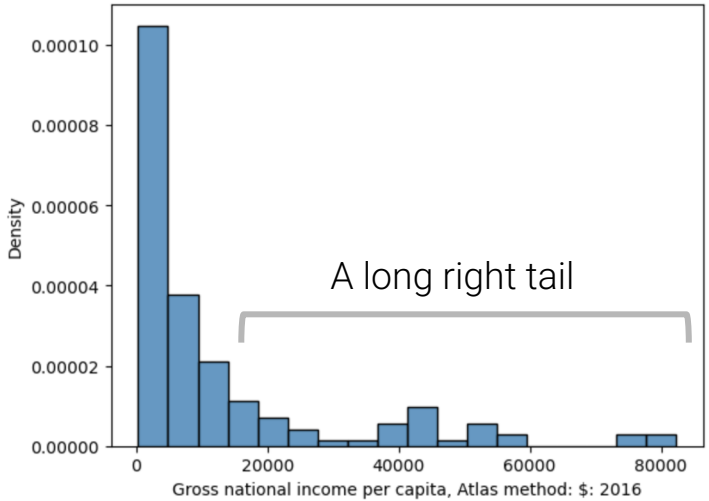
Always include a legend when color is used to encode information!

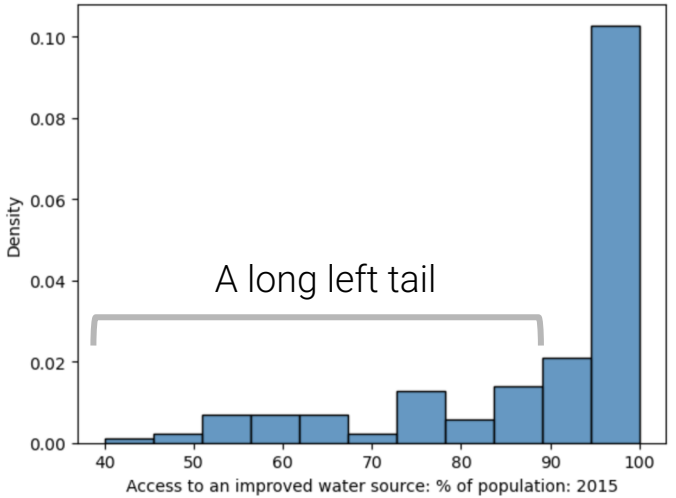
Interpreting histograms

The **skew** of a histogram describes the direction in which its "tail" extends

- A distribution with a long right tail is skewed right
- A distribution with a long left tail is skewed left

A histogram with no clear skew is called symmetric.

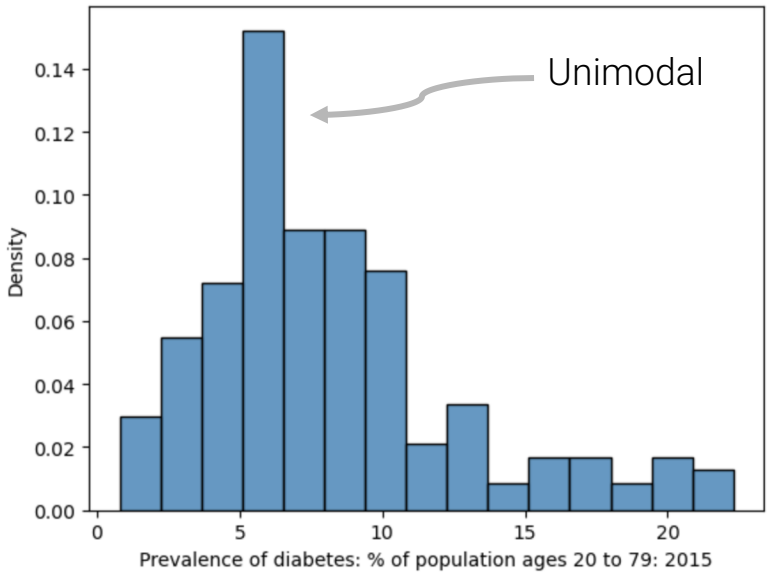


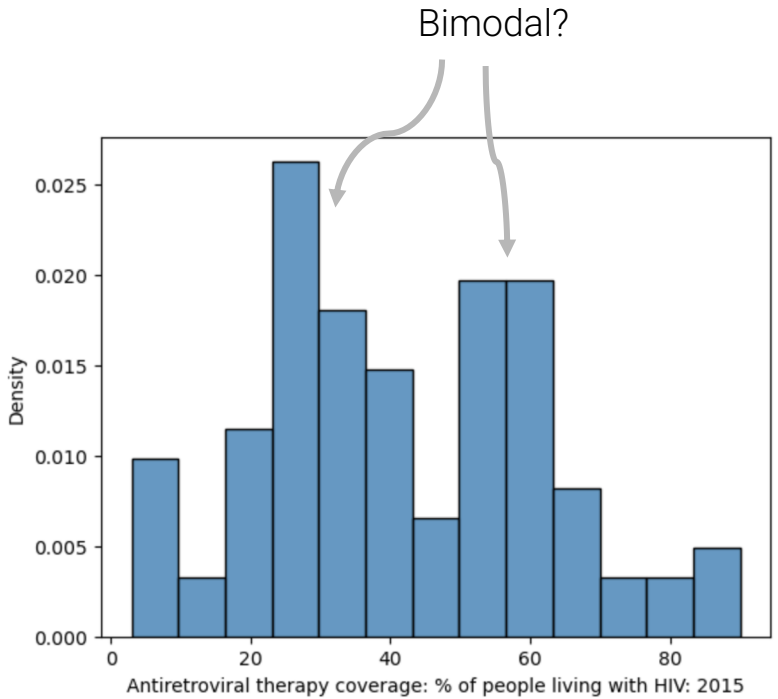


Interpreting histograms

The **mode(s)** of a histogram are the peak values in the distribution

- A distribution with one clear peak is called unimodal
- Two peaks: bimodal
- More peaks: multimodal



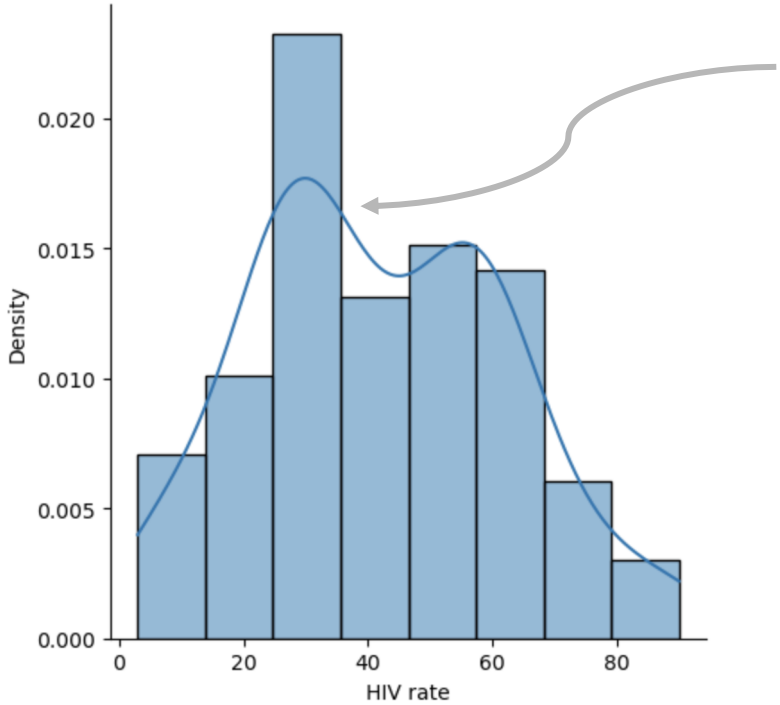


Kernel Density Estimation

- Goals of visualization
- Visualizing distributions
- **Kernel density estimation**
- Visualizing relationships
- Transformations

Kernel density estimation: intuition

Often, we want to identify *general* trends across a distribution, rather than focus on detail. Smoothing a distribution helps generalize the structure of the data and eliminate noise.



A KDE curve

Idea: approximate the probability distribution that generated the data

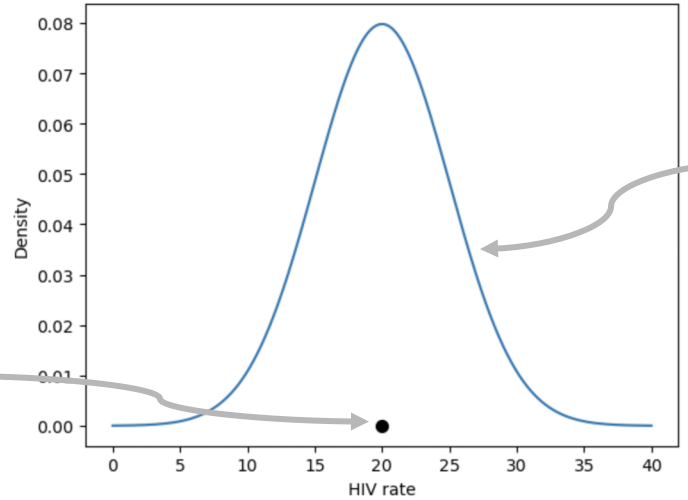
- Assign an “error range” to each data point in the dataset – if we were to sample the data again, we might get a different value
- Sum up the error ranges of all data points
- Scale the resulting distribution to integrate to 1

Kernel density estimation: process

Idea: approximate the probability distribution that generated the data

- Place a kernel at each data point.
- Normalize kernels so that total area = 1.
- Sum all kernels together.

A **kernel** is a function that tries to capture the randomness of our sampled data.



The **kernel** models the probability of us sampling that datapoint

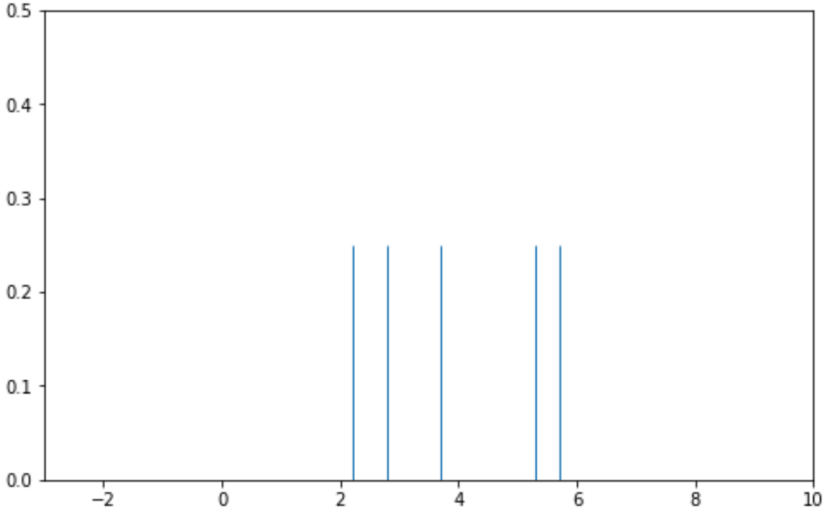
Area below integrates to 1

A **datapoint** in our dataset

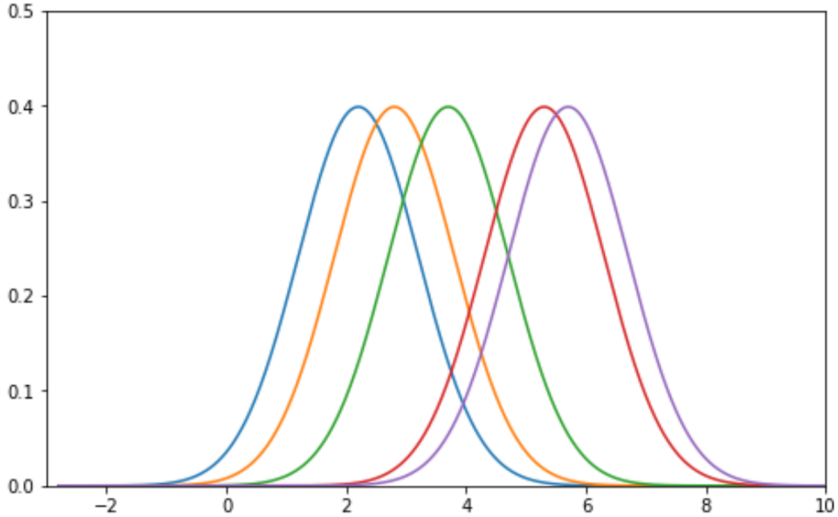
Step 1 – Place a kernel at each data point

Consider a fake dataset with just five collected datapoints.

- Place a **Gaussian kernel** with **bandwidth** of **alpha = 1**.
- We will precisely define both the **Gaussian kernel** and **bandwidth** in a few slides.



Each line represents a datapoint in the dataset (e.g. one country's HIV rate)

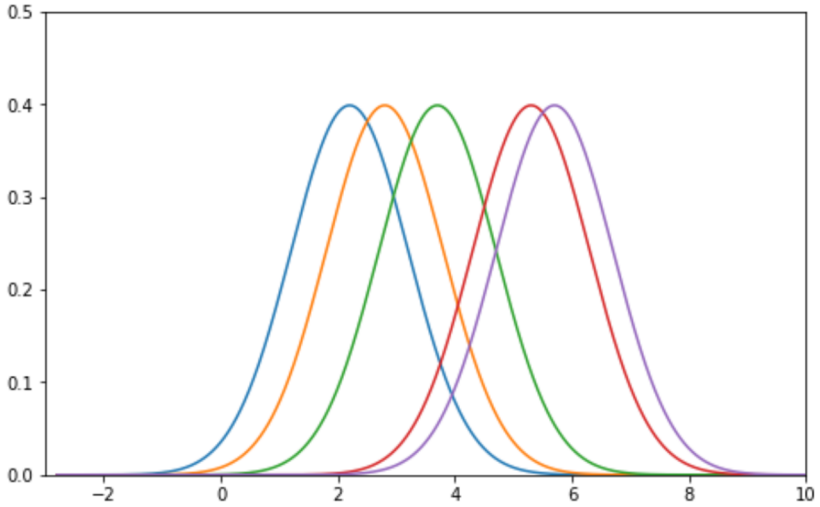


Place a kernel on top of each datapoint

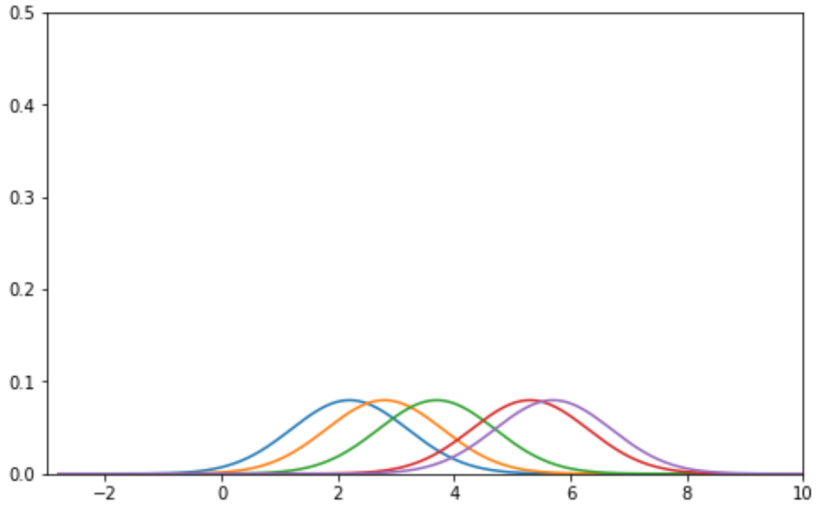
Step 2 – Normalize kernels

In Step 3, we will be summing each of these kernels to produce a probability distribution.

- We want the result to be a valid probability distribution that has area 1.
- We have 5 different kernels, each with an area 1.
- So, we normalize by multiplying each kernel by 1/5.



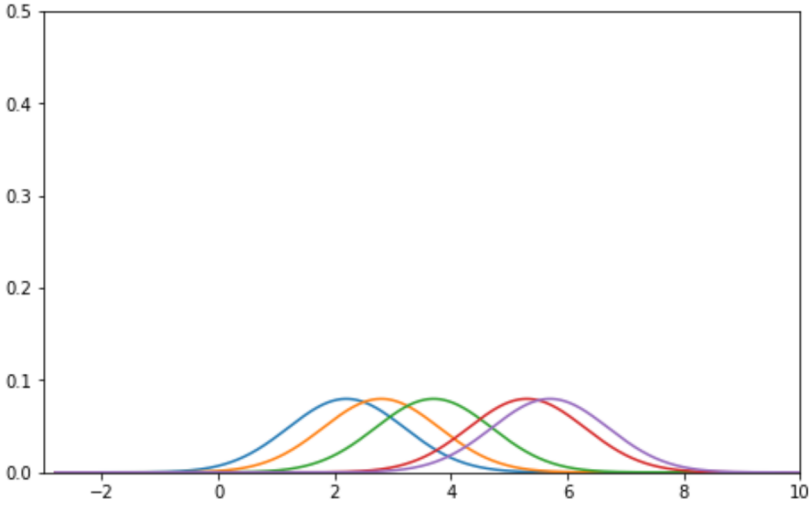
Each kernel has area 1



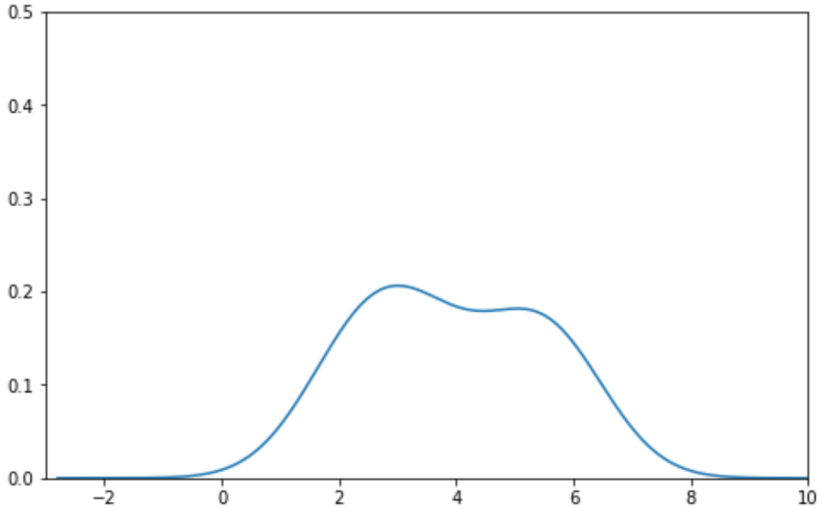
Each normalized kernel has density 1/5

Step 3 – Sum the normalized kernels

At each point in the distribution, add up the values of all kernels. This gives us a smooth curve with area 1 – an approximation of a probability distribution!



Sum these five normalized curves together

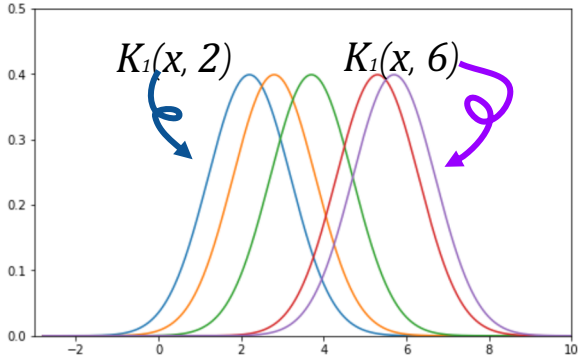


The final KDE curve

Summary of KDE

$$f_\alpha(x) = \frac{1}{n}\sum_{i=1}^{n} K_\alpha(x, x_i)$$

② ③ ①



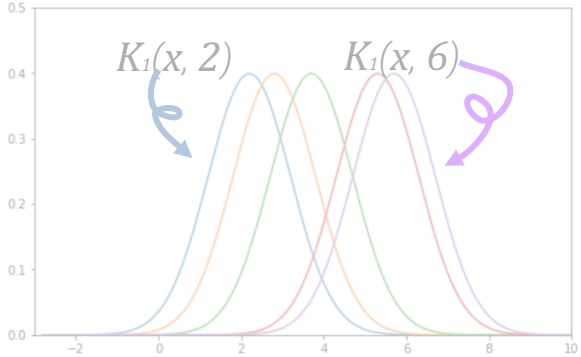
A general "KDE formula" function is given above.

1. $K_\alpha(x, x_i)$ is the **kernel** function centered on the observation i.
   - Each kernel individually has area 1.
   - K represents our kernel function of choice. We'll talk about the math of these functions soon.

Summary of KDE

$$f_\alpha(x) = \frac{1}{n} \sum_{i=1}^{n} K_\alpha(x, x_i)$$

② ③ ①



A general "KDE formula" function is given above.

1. $K_\alpha(x, x_i)$ is the **kernel** centered on the observation i.
   - Each kernel individually has area 1.
   - $\mathbf{x}$ represents any number on the number line. It is the input to our function.
2. $\mathbf{n}$ is the number of observed data points that we have.
   - We multiply by $\mathbf{1/n}$ to normalize the kernels so that the total area of the KDE is still 1.
3. Each $\mathbf{x}_i$ ($\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_n$) represents an observed data point. We sum the kernels for each datapoint to create the final KDE curve

α is the **bandwidth** or **smoothing parameter**.

Kernels

A **kernel** (for our purposes) is a valid density function, meaning:

- It must be non-negative for all inputs.
- It must integrate to 1(area under curve = 1).

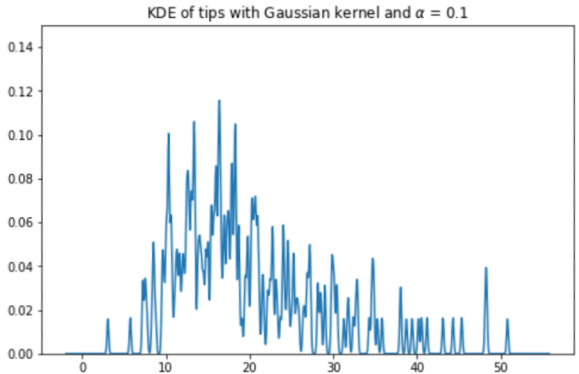
The most common kernel is the **Gaussian kernel**.

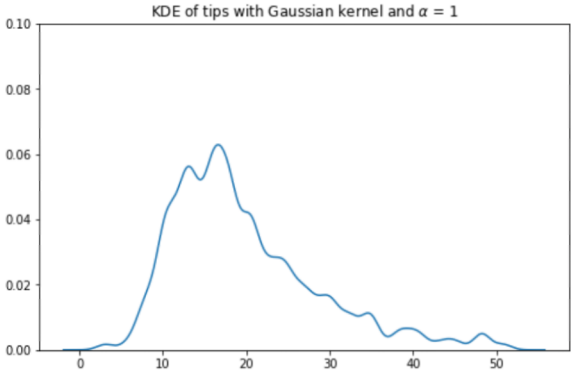
- Gaussian = normal distribution = bell curve
- Here, x represents any input, and x_i represents the ith observed value (datapoint).
- Each kernel is **centered** on our observed values (and so its distribution mean is x_i).
- **α** is the **bandwidth parameter**. It controls the smoothness of our KDE. Here, it is also the standard deviation of the Gaussian.

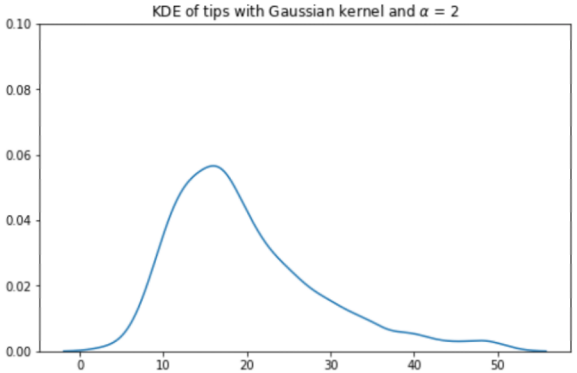
$$K_\alpha(x, x_i) = \frac{1}{\sqrt{2\pi\alpha^2}} e^{-\frac{(x-x_i)^2}{2\alpha^2}}$$

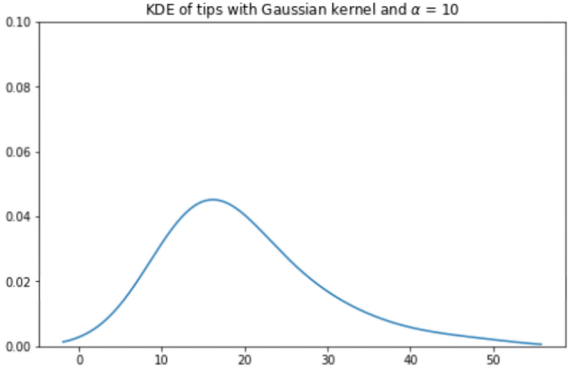
Memorizing this formula is less important than knowing the shape and how the bandwidth parameter **α** smoothes the KDE.

Effect of bandwidth on KDEs



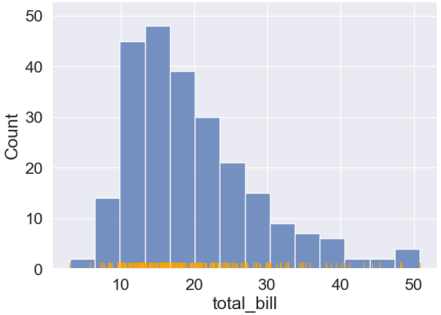






Bandwidth is analogous to the width of each bin in a histogram.

- As **α** increases, the KDE becomes more smooth.
- Large **α** KDE is simpler to understand, but gets rid of potentially important distributional information (e.g. multimodality).



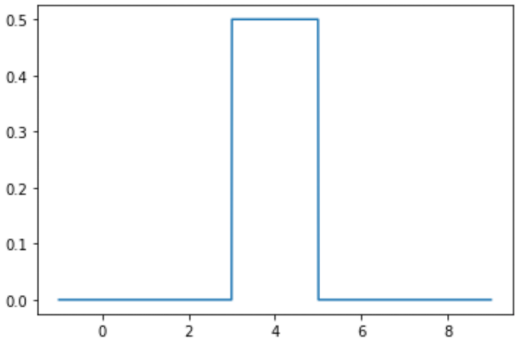
Other kernels: boxcar

As an example of another kernel, consider the **boxcar kernel**.

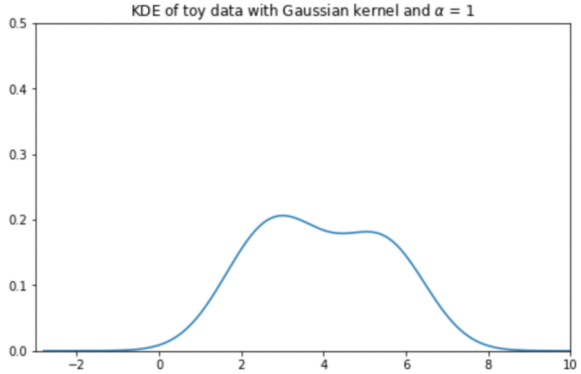
- It assigns uniform density to points within a "window" of the observation, and 0 elsewhere.
- Resembles a histogram… *sort of*.

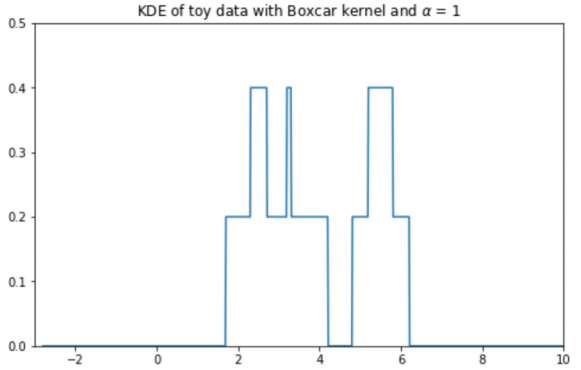
$$K_\alpha(x, x_i) = \begin{cases} \frac{1}{\alpha}, & |x - x_i| \leq \frac{\alpha}{2} \\ 0, & \text{else} \end{cases}$$

- Not of any practical use in DSC 510! Presented as a simple theoretical alternative.



A boxcar kernel centered on $x_i = 4$ with $\alpha = 2$.





Visualizing Relationships

- Goals of visualization
- Visualizing distributions
- Kernel density estimation
- **Visualizing relationships**
- Transformations

From distributions to relationships

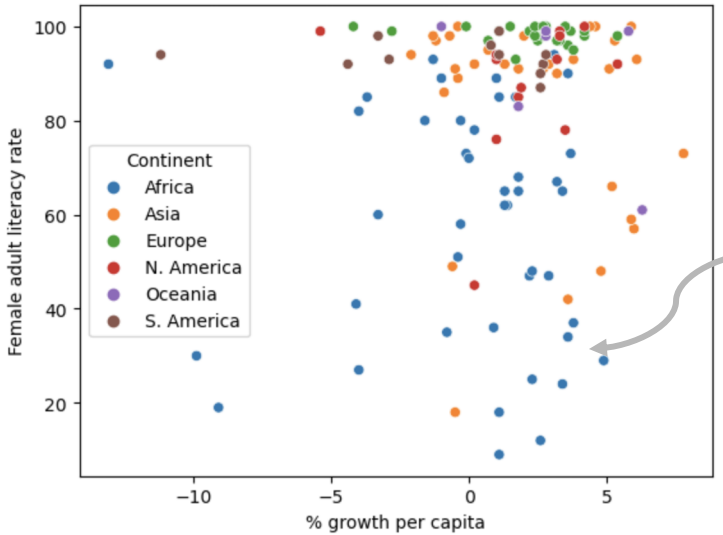
Up until now, we focused exclusively on visualizing variable distributions.

Now we will visualize **relationships** between variables. In other words, how do sets of two (or more) variables vary in relation to one another?

Scatter plot

Scatter plots are used to reveal relationships between two quantitative variables

- Plot one quantitative continuous variable on the x-axis, and second quantitative continuous variable on the y-axis
- Each scatter point represents one datapoint in the dataset



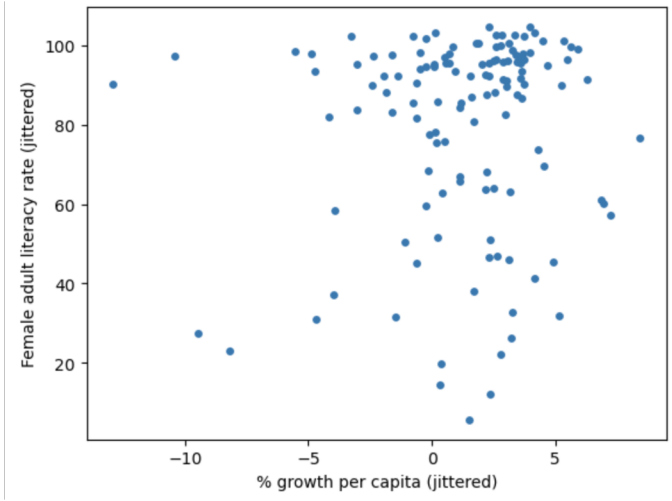

```
plt.scatter(x_values, y_values)
```

```
sns.scatterplot(data = df, x = “x_column”, \
                y = “y_column”, hue = “hue_column”)
```

Overplotting

The plot on the previous slide suffered from **overplotting** – scatter points all stacked on top of one another are difficult to see.

Jittering: adding a small amount of random noise to all x and y values to slightly move each scatter point. Main trends are still present, but individual datapoints are easier to distinguish.

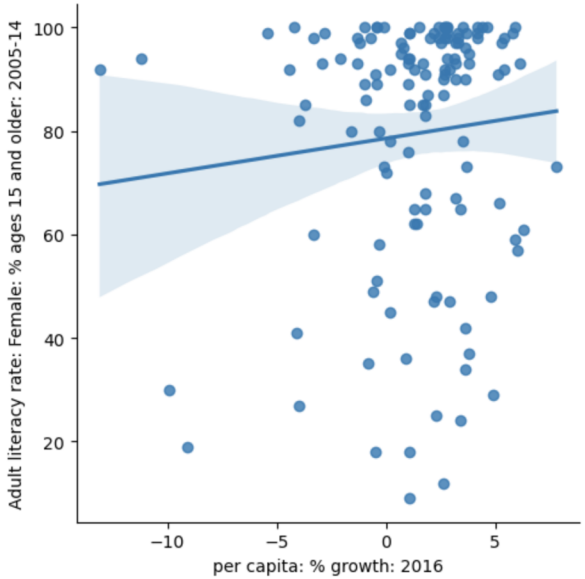



```
x_noise = np.random.uniform(-1, 1, len(wb))
y_noise = np.random.uniform(-5, 5, len(wb))

plt.scatter(wb['% growth'] + x_noise, \
            wb['Literacy rate: Female'] + y_noise, \
            s=15);
```

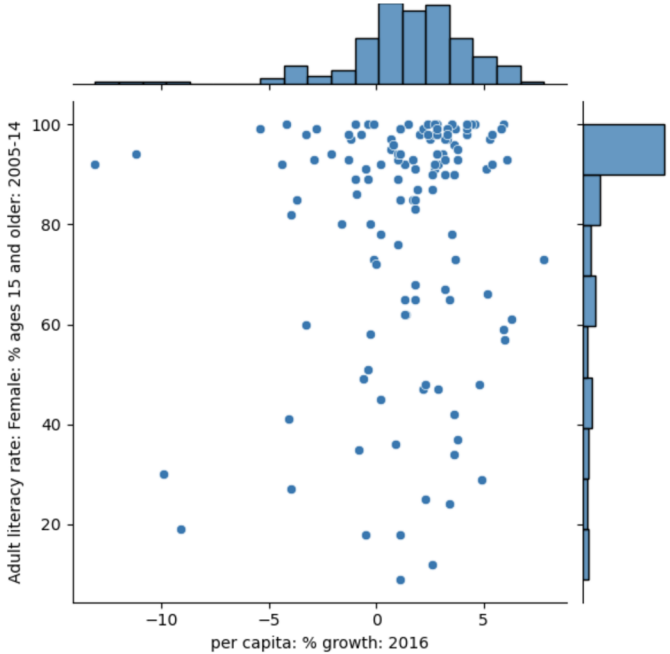
Decreasing point size also helps. `s` specifies the marker size in Matplotlib.

Scatter plot alternatives

Seaborn includes several built-in functions for making more complex scatter plots.




```
sns.lmplot(data = df, \
x = “x_column”, y = “y_column”)
```




```
sns.jointplot(data = df, \
x = “x_column”, y = “y_column”)
```

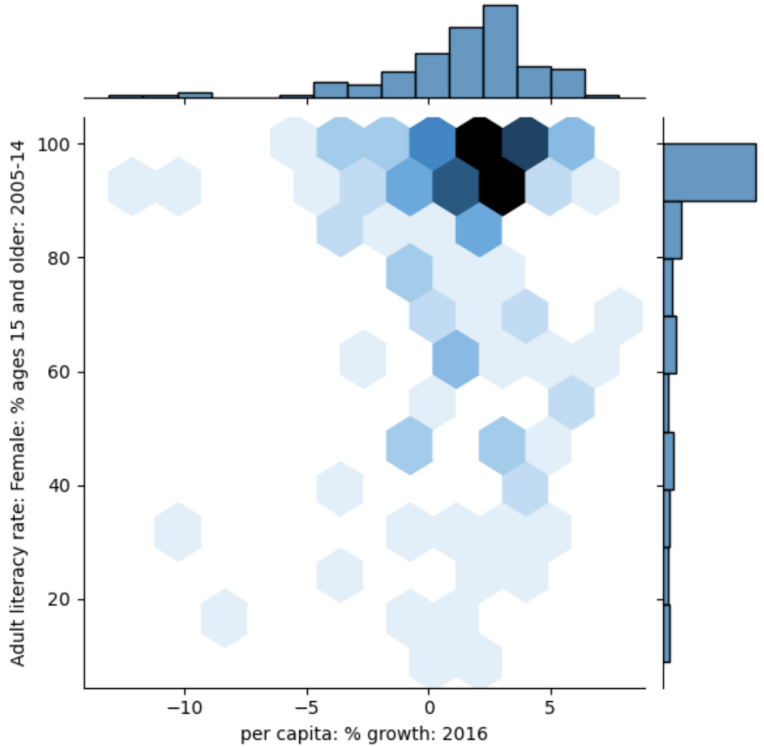
Hex plots

Rather than plot individual datapoints, plot the *density* of their joint distribution.

Can be thought of as a two dimensional histogram.

- The xy plane is binned into hexagons.
- More shaded hexagons typically indicate a greater density/frequency = more datapoints lie in that spot

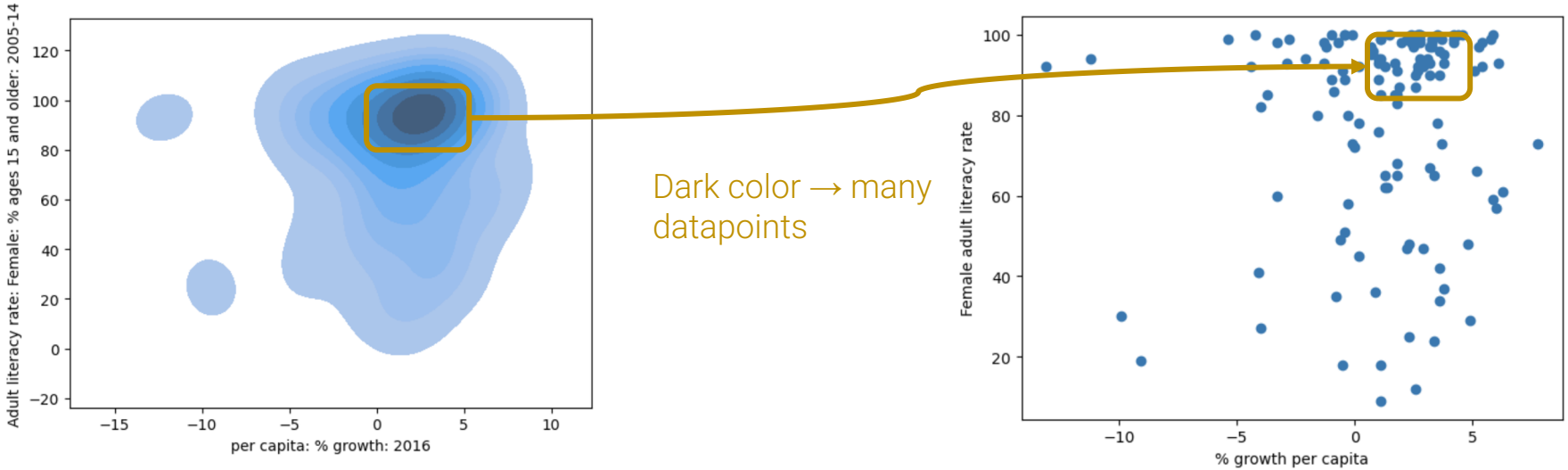
```
sns.jointplot(data=df, x='x_column', \
y='y_column', kind='hex')
```



Contour plots

2-dimensional version of a KDE plot

Similar to a topographic map – contour lines represent an area that has the *same density* of datapoints throughout. Darker colors indicate more datapoints in the region.




```
sns.kdeplot(data = df, x='x_column', y='y_column', fill=True)
```

Transformations

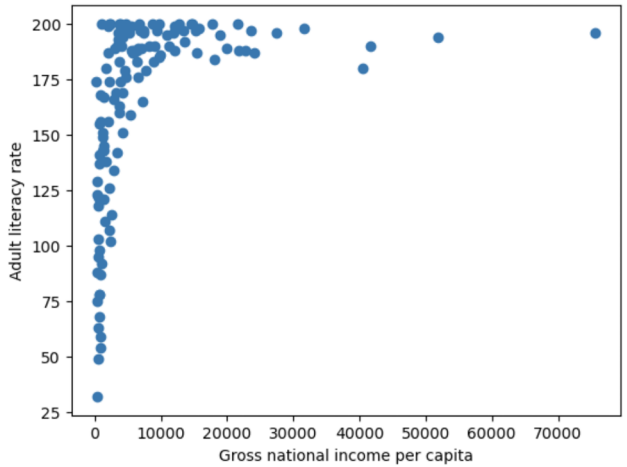
- Goals of visualization
- Visualizing distributions
- Kernel density estimation
- Visualizing relationships
- **Transformations**

Visualization theory

Remember our goals of visualization:

1. To help your own understanding of your data/results
2. To communicate results/conclusions to others

These are influenced by our choice of visualization and our choices in *how to prepare data for visualization*.



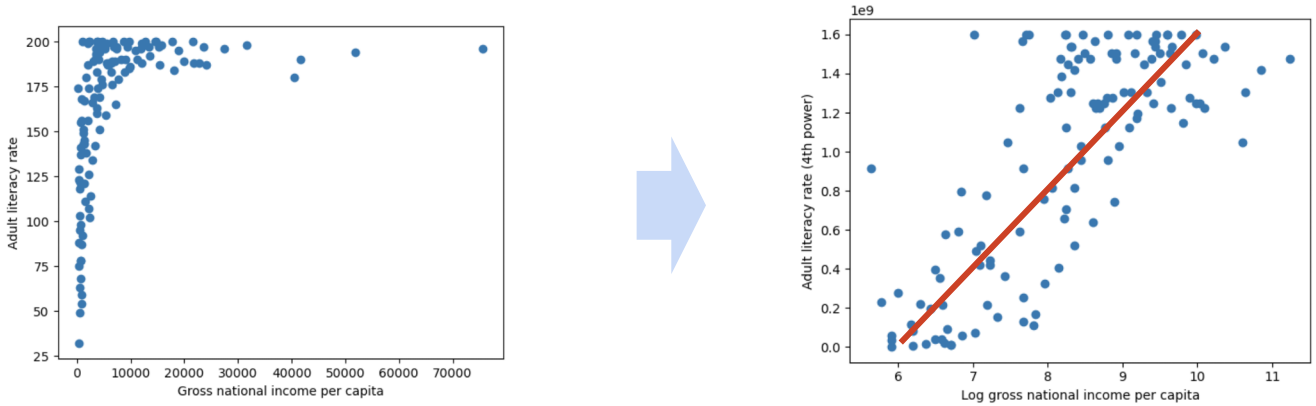
What problems are there here?

- Data is “smushed” – hard to interpret, even if we jittered
- Difficult to generalize a clear relationship between the variables

We often **transform** a dataset to help prepare it for being visualized.

Linearization

When applying transformations, we often want to **linearize** the data – rescale the data so the x and y variables share a linear relationship.

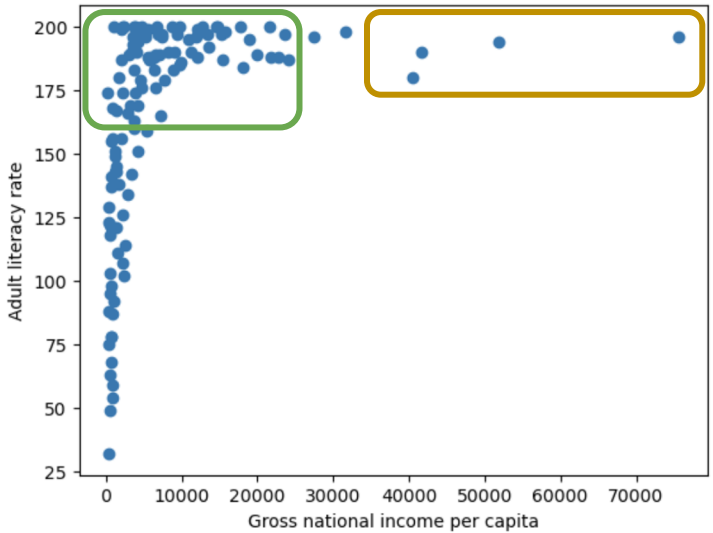


Why?

- Linear relationships are simple to interpret – we know how to work with slopes and intercepts to understand how two variables are related

Applying transformations

What makes this plot non-linear?



1. A few large outlying x values are distorting the horizontal axis
2. Many large y values are all clumped together, compressing the vertical axis

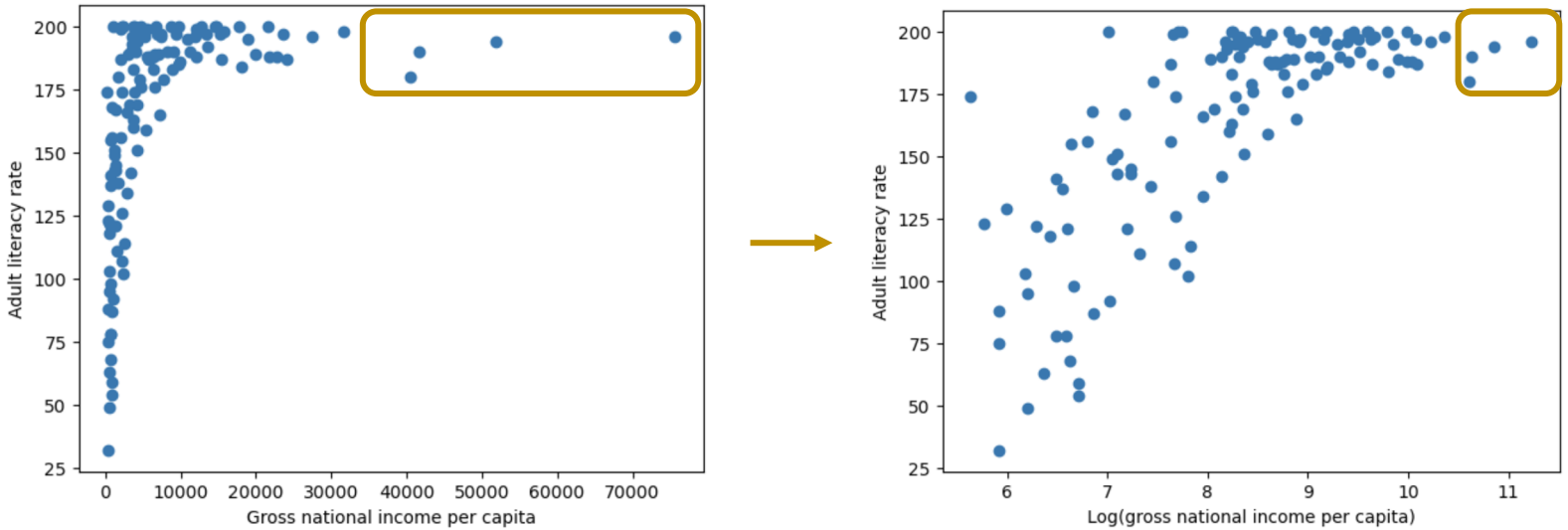
Applying transformations

What makes this plot non-linear?

1. A few large outlying x values are distorting the horizontal axis

Resolve by log-transforming the x data:

- Taking the log of a large number decreases its value significantly
- Taking the log of a small number does not change its value as significantly



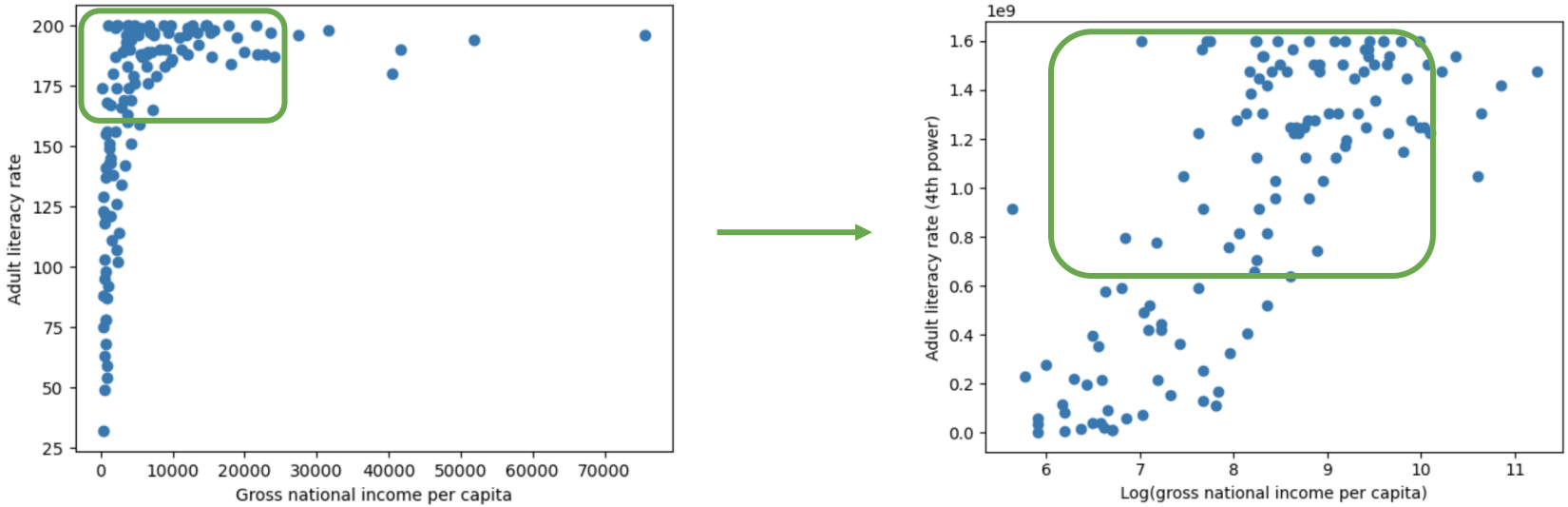
Applying transformations

What makes this plot non-linear?

2. Many large y values are all clumped together, compressing the vertical axis

Resolve by power-transforming the x data:

- Raising a large number to a power increases its value significantly
- Raising a small number to a power does not change its value as significantly

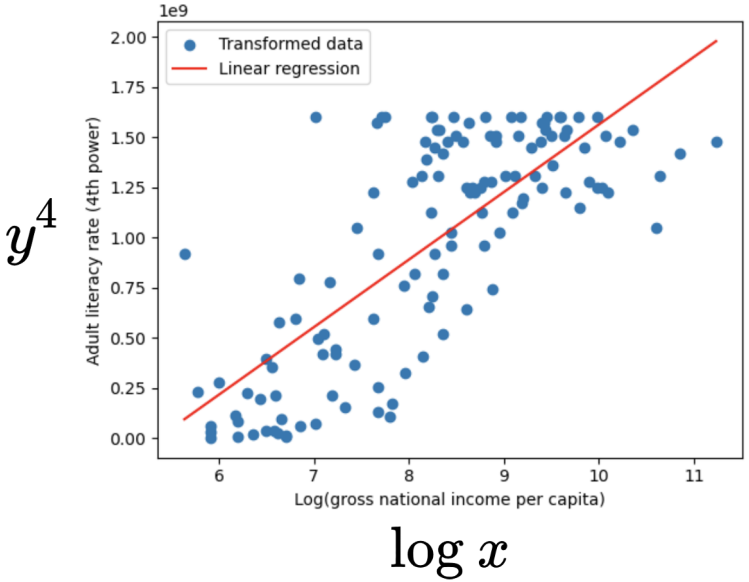


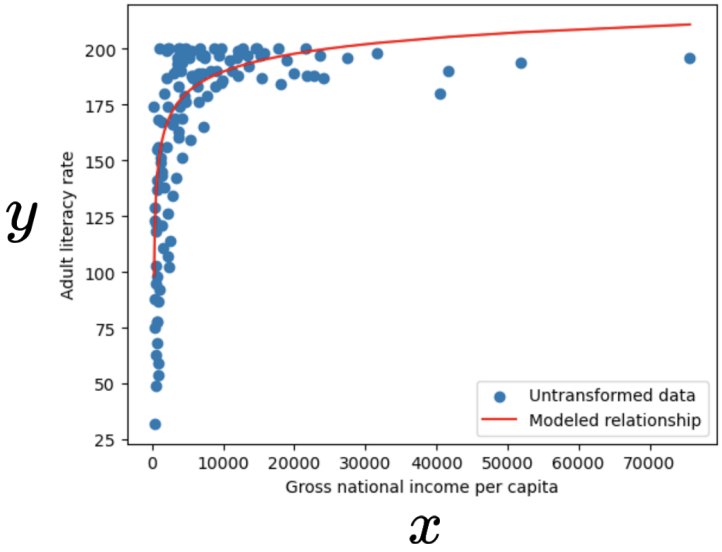
Interpreting transformed data

Now, we see a linear relationship between the transformed variables

This tells us about the underlying relationship between the *original* x and y!

$$y^4 = m(\log x) + b \quad \longrightarrow \quad y = [m(\log x) + b]^{1/4}$$



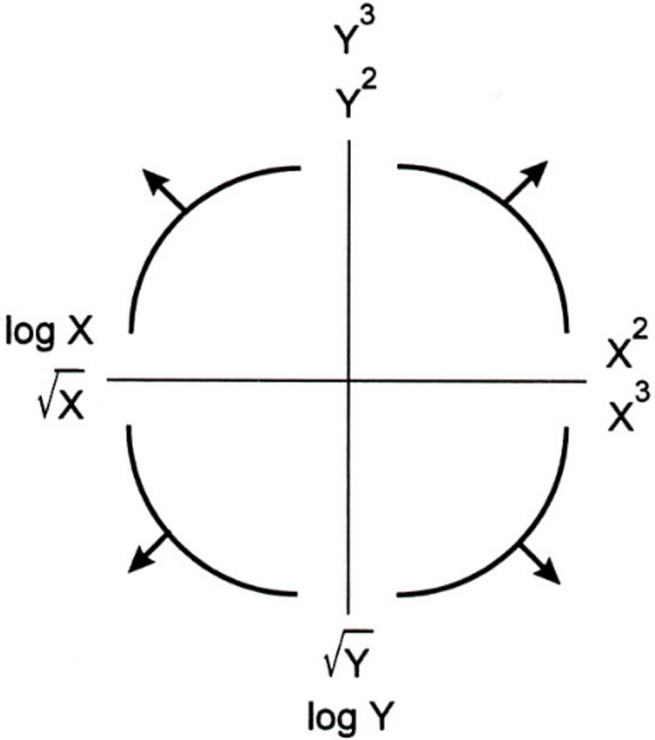


Tukey-Mosteller Bulge Diagram

The **Tukey-Mosteller Bulge Diagram** is a guide to possible transforms to try to get linearity.

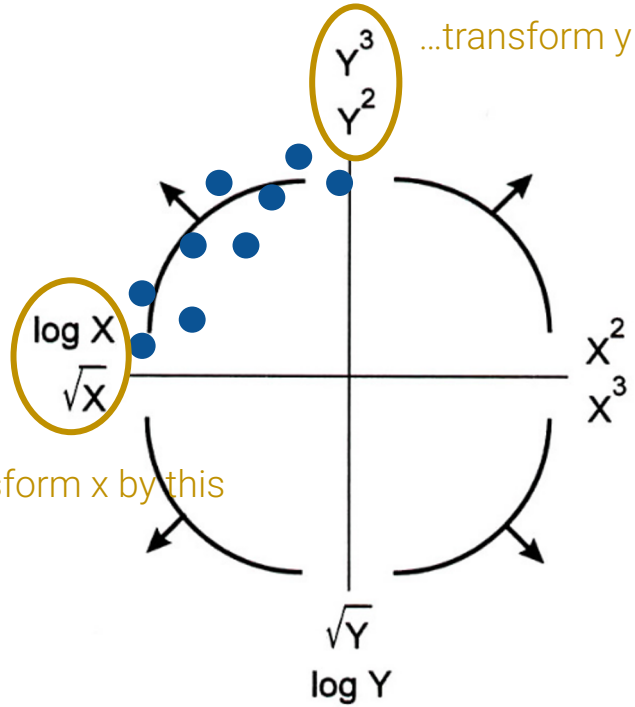
- A visual summary of the reasoning we just worked through
- sqrt and log make a value “smaller”.
- Raising to a value to a power makes it “bigger”.
- There are multiple solutions. Some will fit better than others.

You should still understand the *logic* we just worked through to decide how to transform the data. The bulge diagram is just a summary.



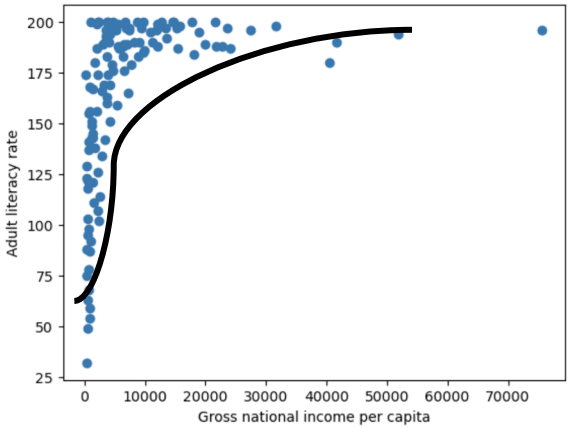
Tukey-Mosteller Bulge Diagram

If the data bulges like this…



Applying to the data from before:

Could have transformed y by y^2, y^3



Could have transformed x by log(x), sqrt(x)