Data Visualization

(Practice) Quiz

- Who was the doctor who drew the cholera map during the 1854 cholera outbreak in London?
- Draw a rectangular map and draw two “wells” in the map as dots. Can you draw a line so that we can easily figure out which well is the closest for every location in the map?
- What’s the name of this method?

Can’t we just use **numbers** and **statistics**?

I		II		III		IV	
X	Y	X	Y	X	Y	X	Y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

Download anscombe.csv file in

[In-class Exercise - Thu] Why can't we simply use summary statistics?

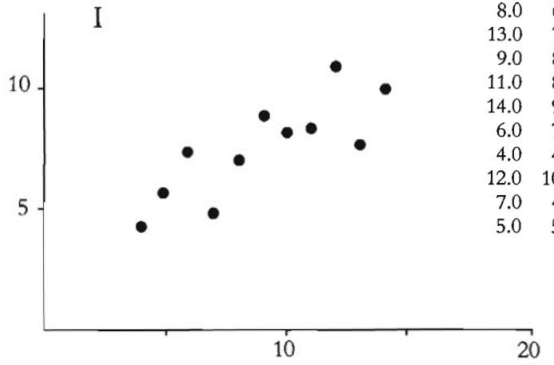
I		II		III		IV	
X	Y	X	Y	X	Y	X	Y
10.0	8.04	10.0	9.14	10.0	7.46	8.0	6.58
8.0	6.95	8.0	8.14	8.0	6.77	8.0	5.76
13.0	7.58	13.0	8.74	13.0	12.74	8.0	7.71
9.0	8.81	9.0	8.77	9.0	7.11	8.0	8.84
11.0	8.33	11.0	9.26	11.0	7.81	8.0	8.47
14.0	9.96	14.0	8.10	14.0	8.84	8.0	7.04
6.0	7.24	6.0	6.13	6.0	6.08	8.0	5.25
4.0	4.26	4.0	3.10	4.0	5.39	19.0	12.50
12.0	10.84	12.0	9.13	12.0	8.15	8.0	5.56
7.0	4.82	7.0	7.26	7.0	6.42	8.0	7.91
5.0	5.68	5.0	4.74	5.0	5.73	8.0	6.89

“Anscombe's quartet”

Property	Value	Accuracy
Mean of x	9	exact
Sample variance of x : s_x^2	11	exact
Mean of y	7.50	to 2 decimal places
Sample variance of y : s_y^2	4.125	±0.003
Correlation between x and y	0.816	to 3 decimal places
Linear regression line	$y = 3.00 + 0.500x$	to 2 and 3 decimal places, respectively
Coefficient of determination of the linear regression : R^2	0.67	to 2 decimal places

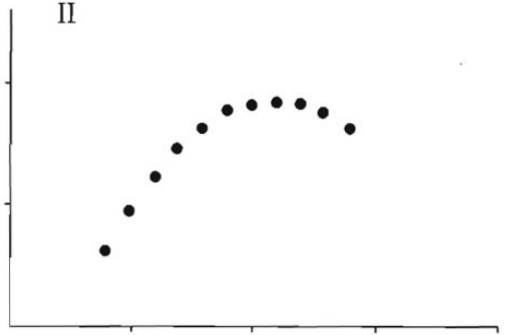
Francis Anscombe

https://en.wikipedia.org/wiki/Anscombe%27s_quartet



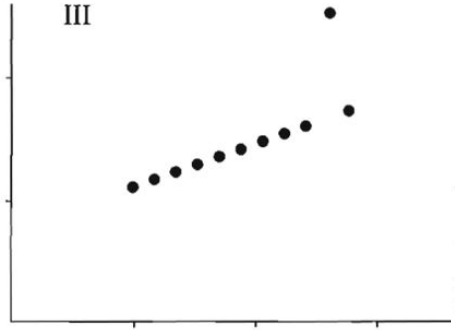
I

X	Y
10.0	8.04
8.0	6.95
13.0	7.58
9.0	8.81
11.0	8.33
14.0	9.96
6.0	7.24
4.0	4.26
12.0	10.84
7.0	4.82
5.0	5.68



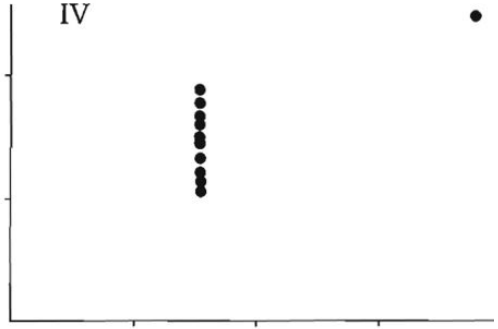
II

X	Y
10.0	9.14
8.0	8.14
13.0	8.74
9.0	8.77
11.0	9.26
14.0	8.10
6.0	6.13
4.0	3.10
12.0	9.13
7.0	7.26
5.0	4.74



III

X	Y
10.0	7.46
8.0	6.77
13.0	12.74
9.0	7.11
11.0	7.81
14.0	8.84
6.0	6.08
4.0	5.39
12.0	8.15
7.0	6.42
5.0	5.73

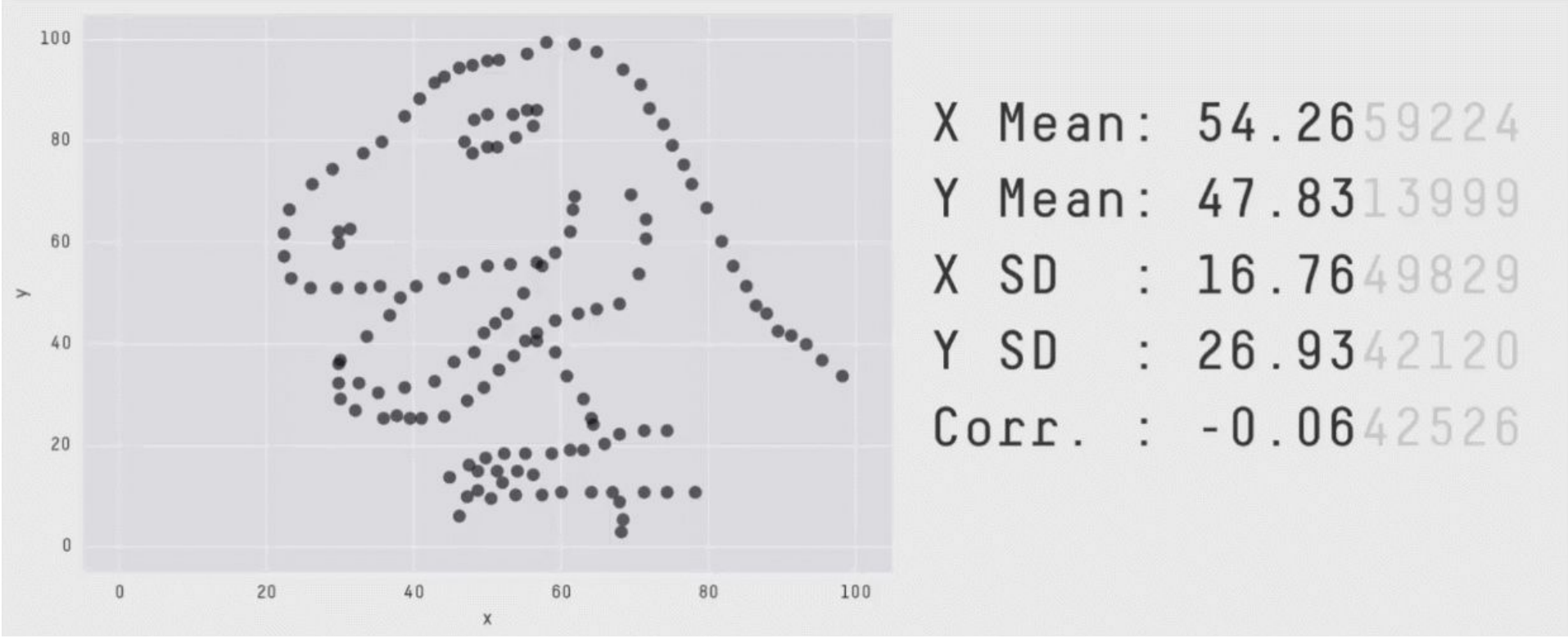


IV

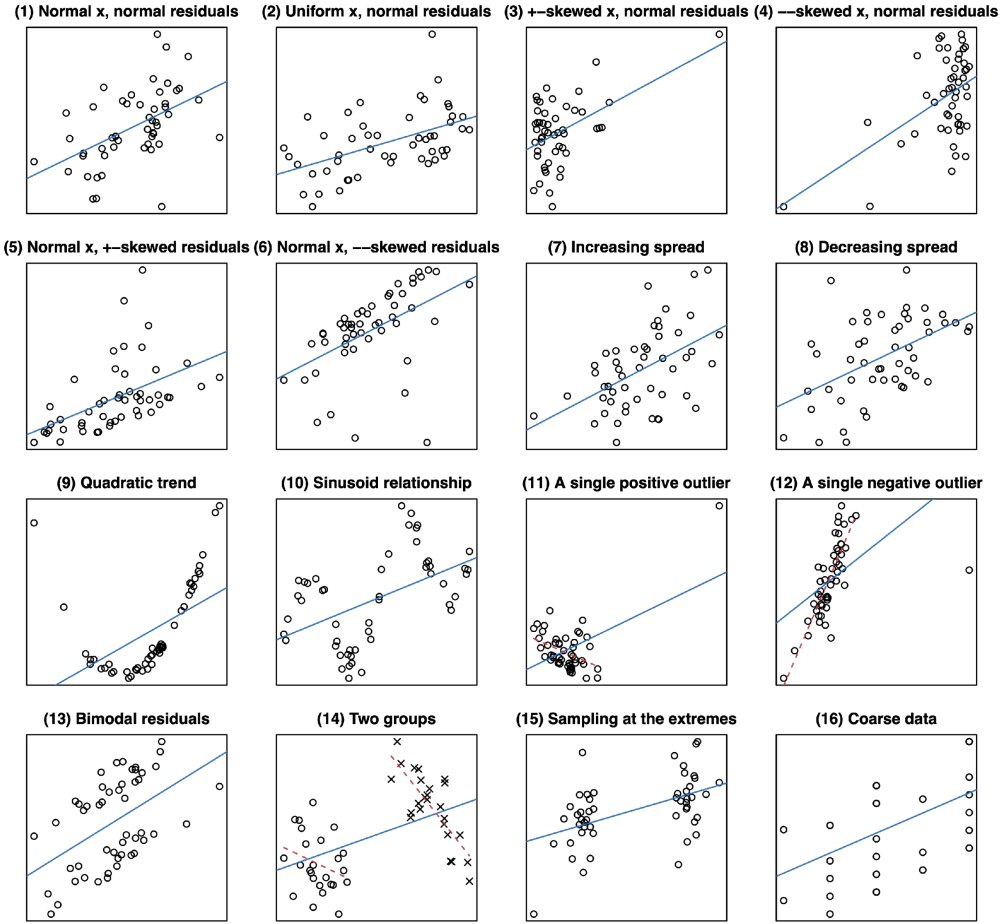
X	Y
8.0	6.58
8.0	5.76
8.0	7.71
8.0	8.84
8.0	8.47
8.0	7.04
8.0	5.25
19.0	12.50
8.0	5.56
8.0	7.91
8.0	6.89

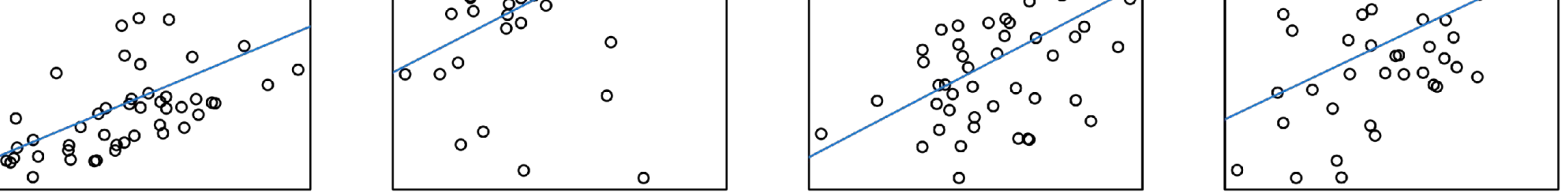
“Same Stats, Different Graphs”

https://www.research.autodesk.com/publications/same-stats-different-graphs

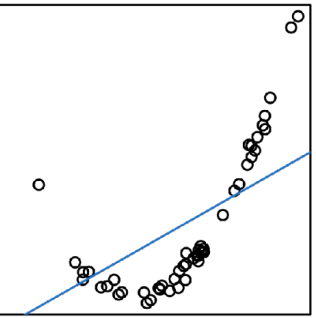


Correlation of 0.5?

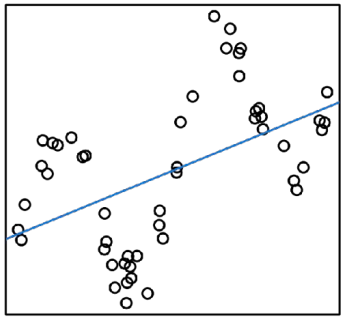

All correlations: r(50) = 0.5
(1) Normal x, normal residuals
(2) Uniform x, normal residuals
(3) +–skewed x, normal residuals
(4) ––skewed x, normal residuals
(5) Normal x, +–skewed residuals
(6) Normal x, ––skewed residuals
(7) Increasing spread
(8) Decreasing spread
(9) Quadratic trend
(10) Sinusoid relationship
(11) A single positive outlier
(12) A single negative outlier
(13) Bimodal residuals
(14) Two groups
(15) Sampling at the extremes
(16) Coarse data



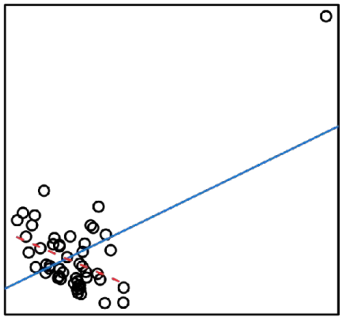
(9) Quadratic trend



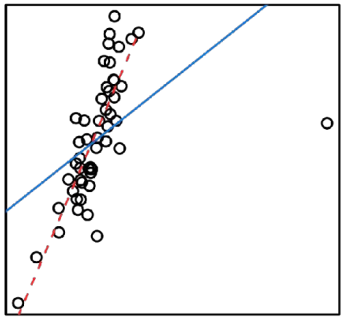
(10) Sinusoid relationship



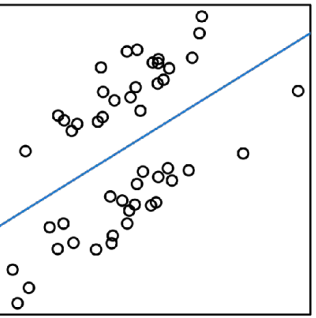
(11) A single positive outlier



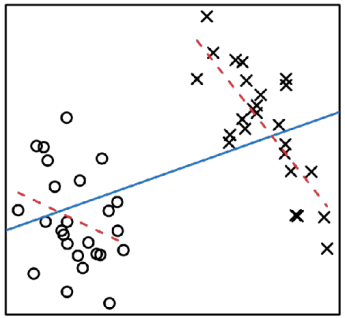
(12) A single negative outlier



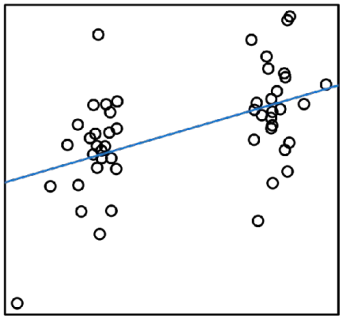
(13) Bimodal residuals



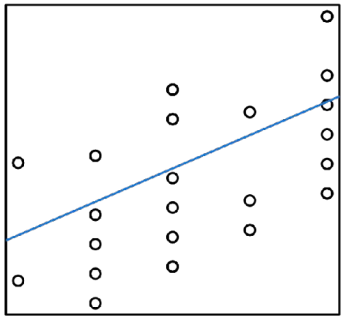
(14) Two groups



(15) Sampling at the extremes

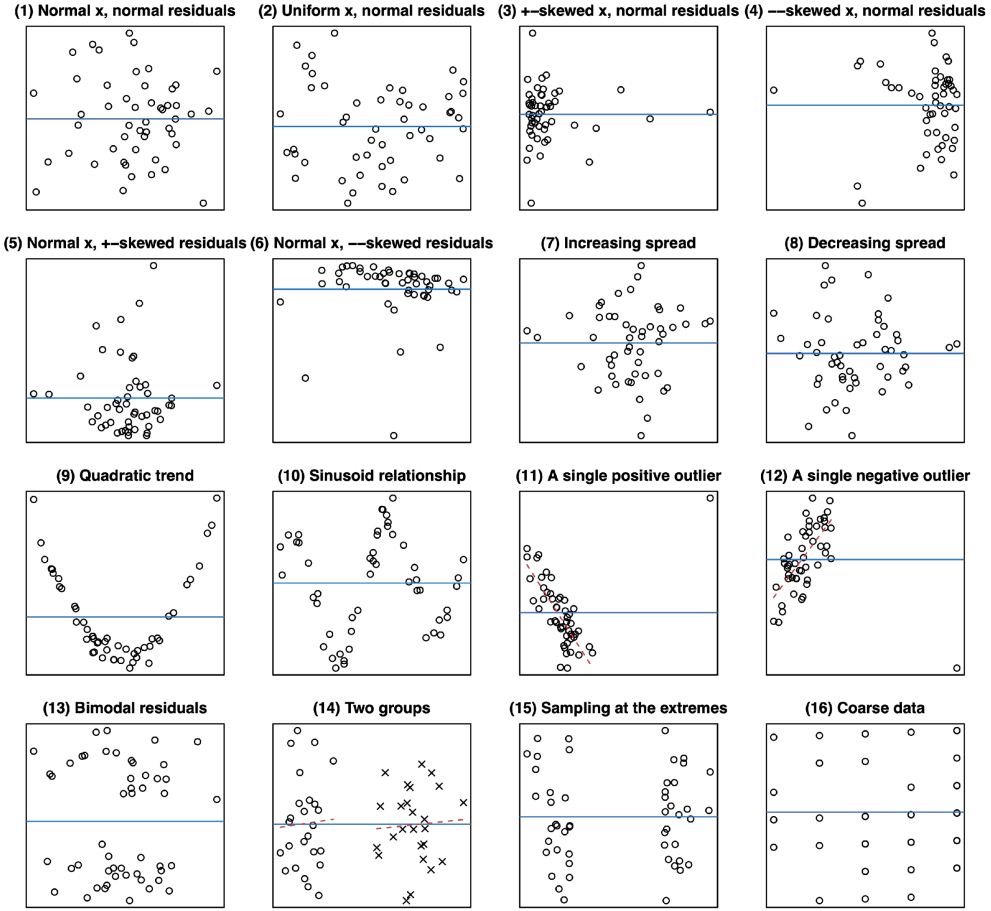


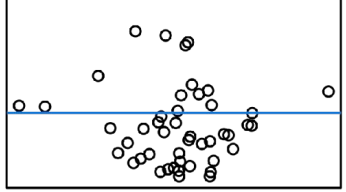
(16) Coarse data

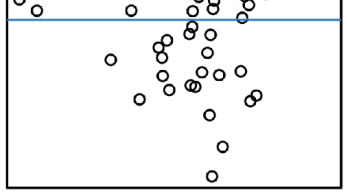


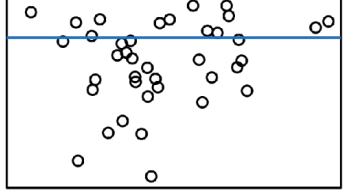
Zero correlation?

All correlations: r(50) = 0

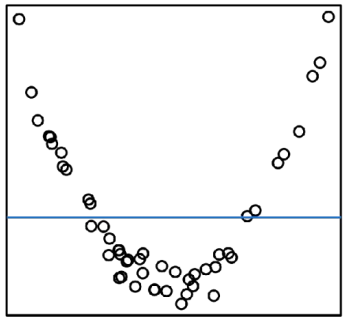




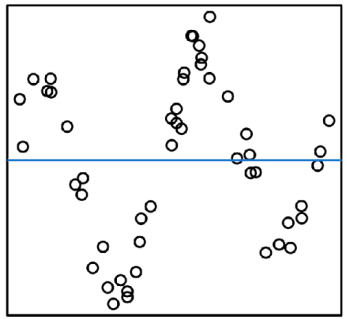




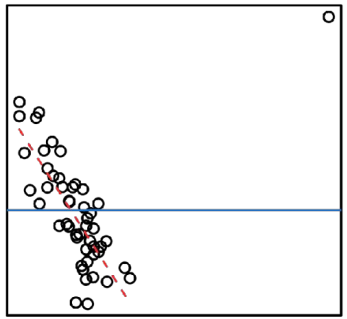
(9) Quadratic trend



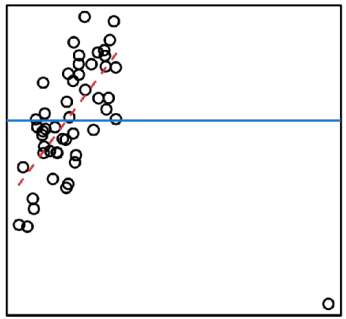
(10) Sinusoid relationship



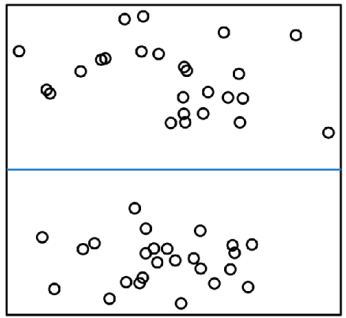
(11) A single positive outlier



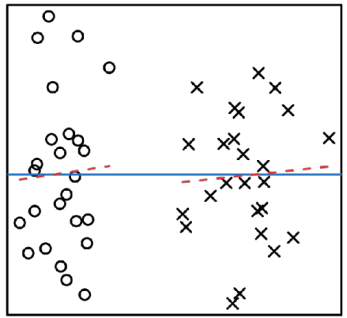
(12) A single negative outlier



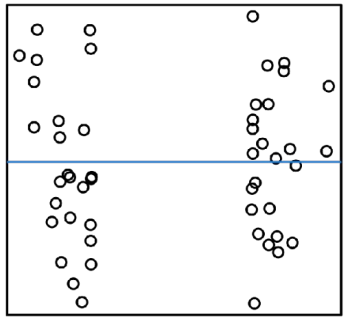
(13) Bimodal residuals



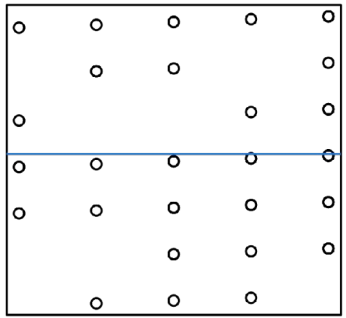
(14) Two groups



(15) Sampling at the extremes



(16) Coarse data



How many 5s?

19256871927316358162315295723051926391270174961923610270137506934162947103701263916

How many 5s?

192**5**6871927316**5**8162315295723050 19

263912701749619236102701375069341

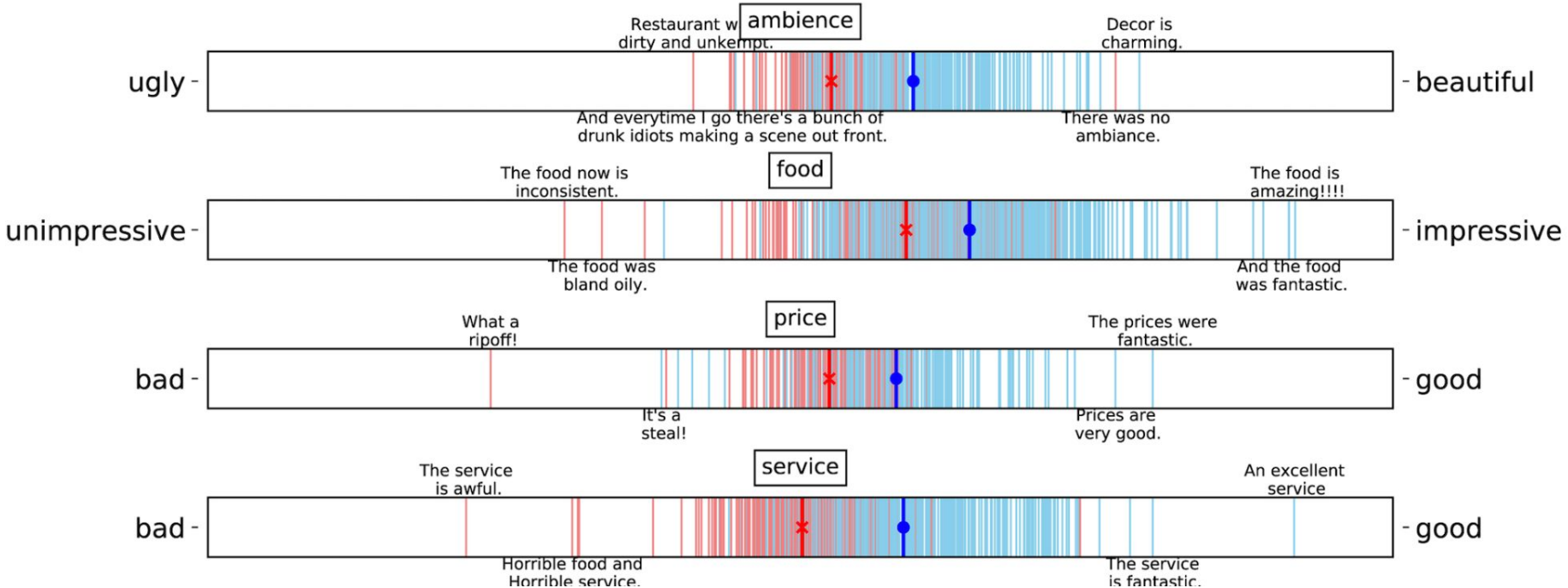
629471037012639161

Our cognitive ability is **limited.**

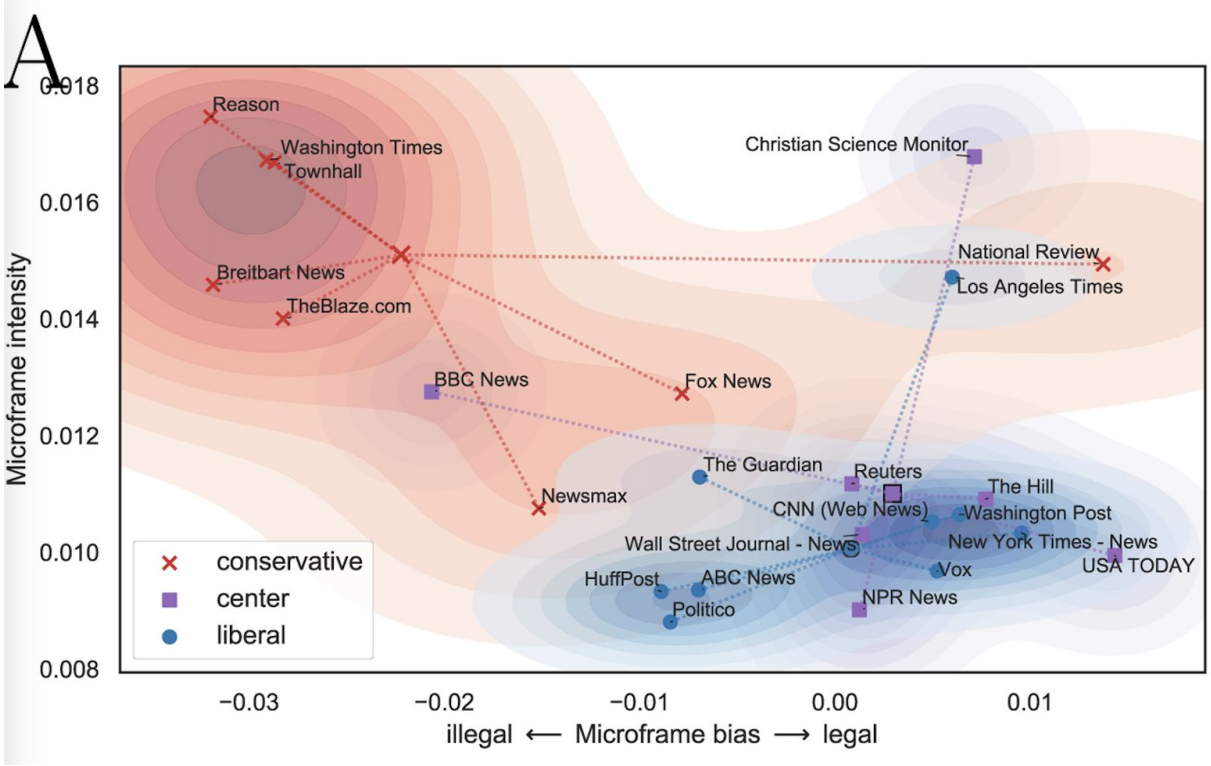
Visual aids free up our mental capacity.

My visualization zoo

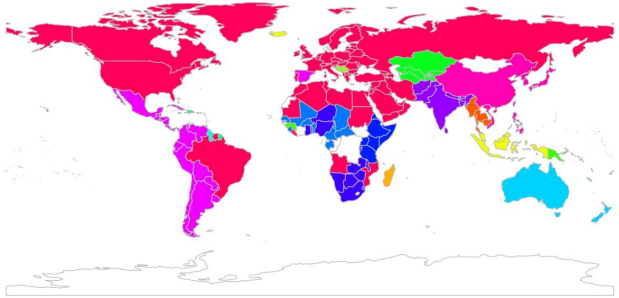
Restaurant reviews



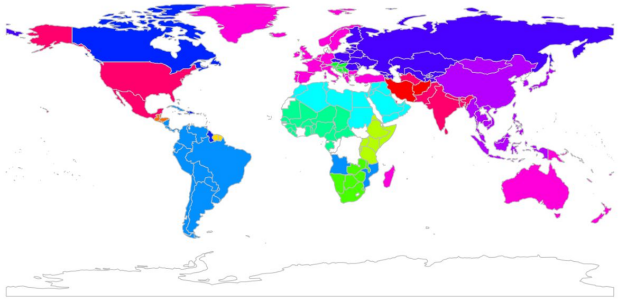
Media framing on immigration issues



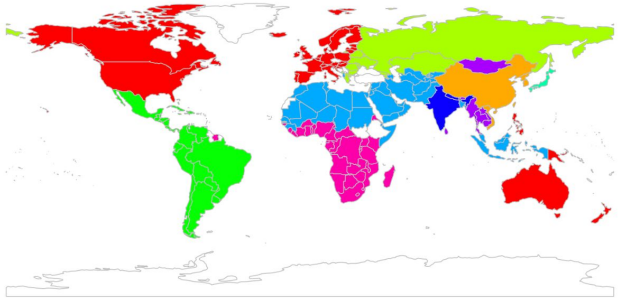
Media attention
Vs.
Public attention



(a) Media Attention

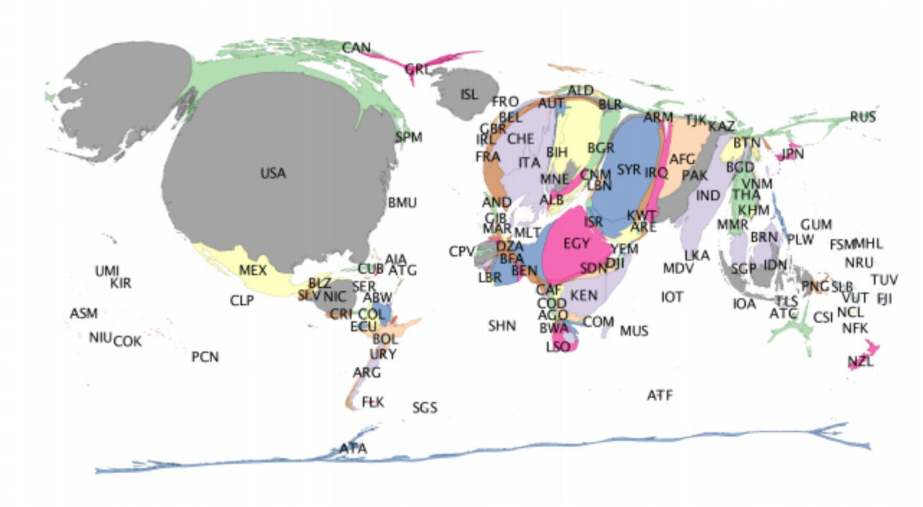


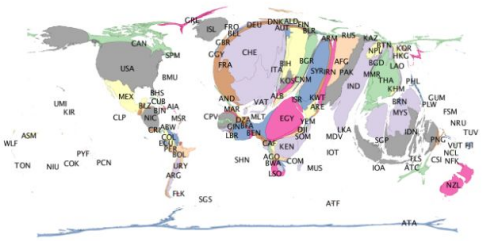
(b) Public Attention



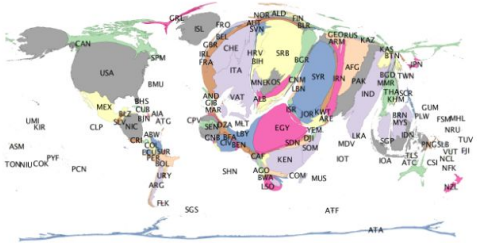
(c) Civilizations proposed by Huntington [15]

One interesting question about the nature of news is how well it reflects the pattern of real events around the world. It's natural to assume that people living in a certain part of the world are more likely to read, see and hear about news from their own region. But what of the international news they get—how does that compare to the international news that people in other parts of the world receive?

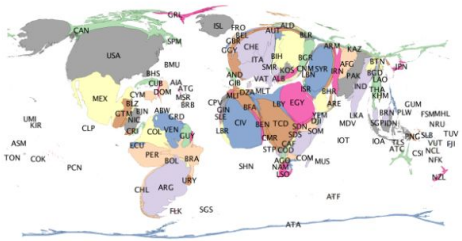




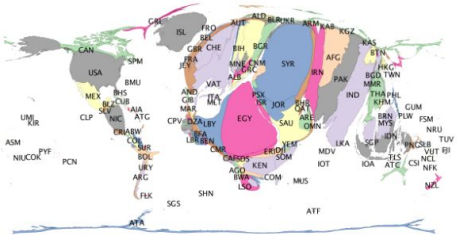
(a) East Asia & Pacific



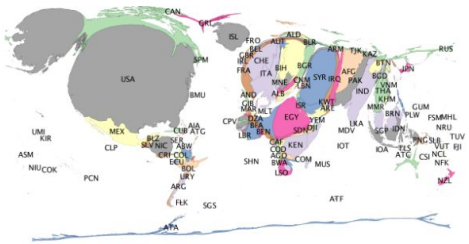
(b) Europe & Central Asia



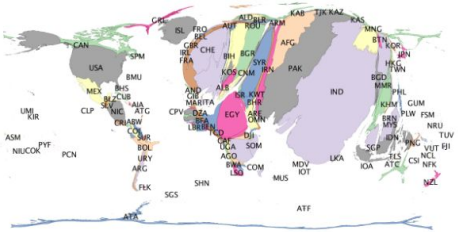
(c) Latin America & Caribbean



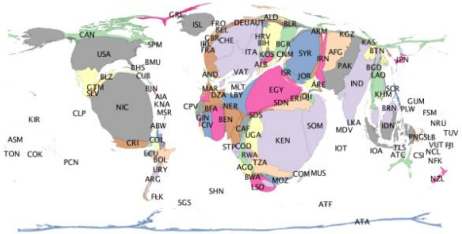
(d) Middle East & North Africa



(e) North America



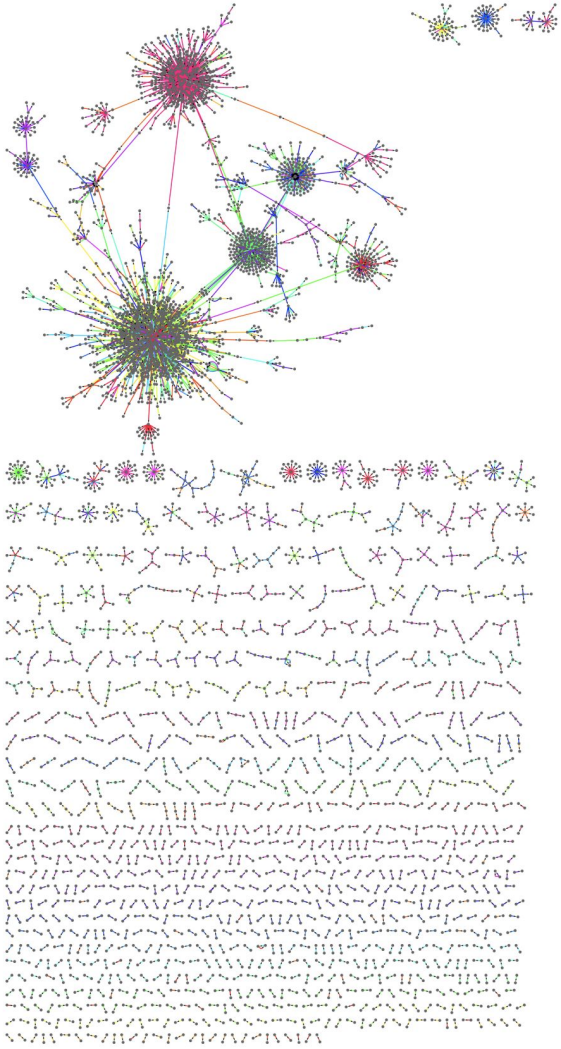
(f) South Asia



(g) Sub-Saharan Africa

Figure 1: News geography seen by each region

Retweet tree

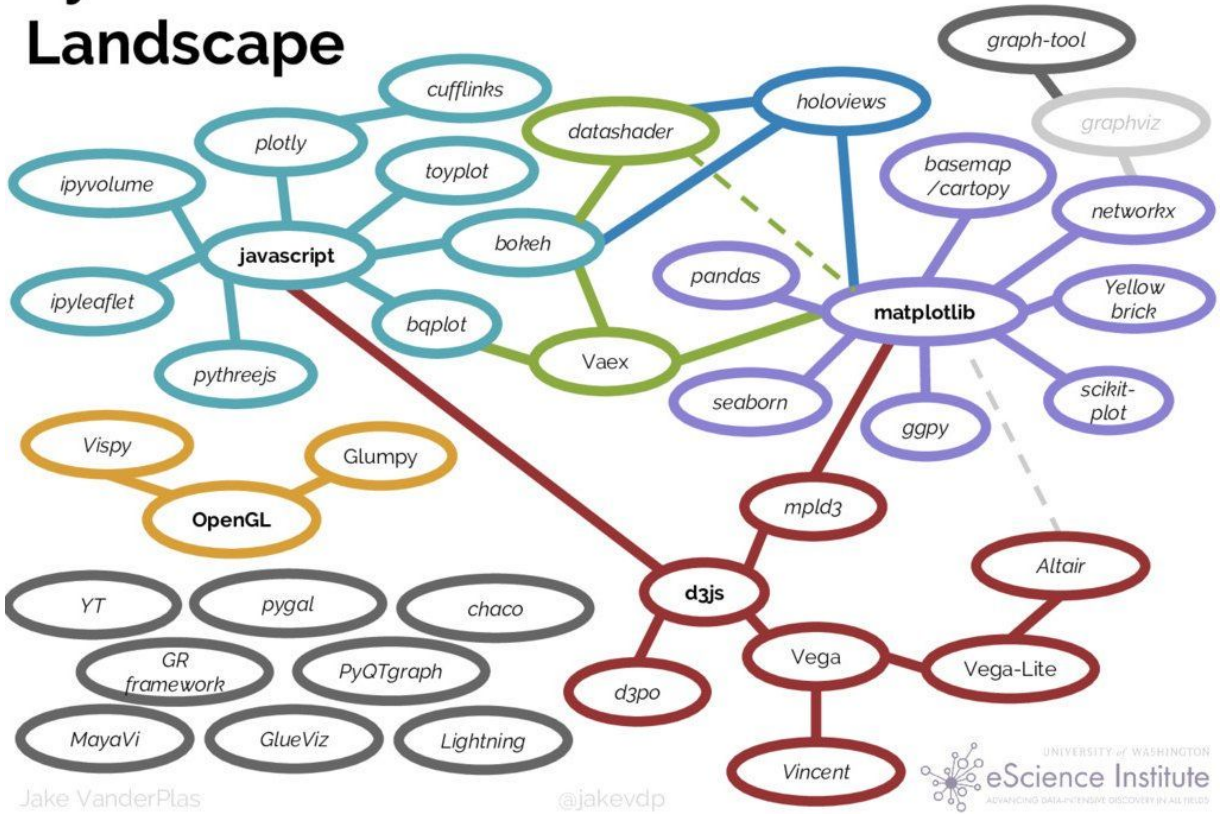


Some take-away points

- **Data is king**!
- There's always endless **data cleaning**.
- There are always lots of **iterations** to arrive at the good final product.
- Understanding pros & cons of visualization techniques and the principles of visualization makes a big difference.

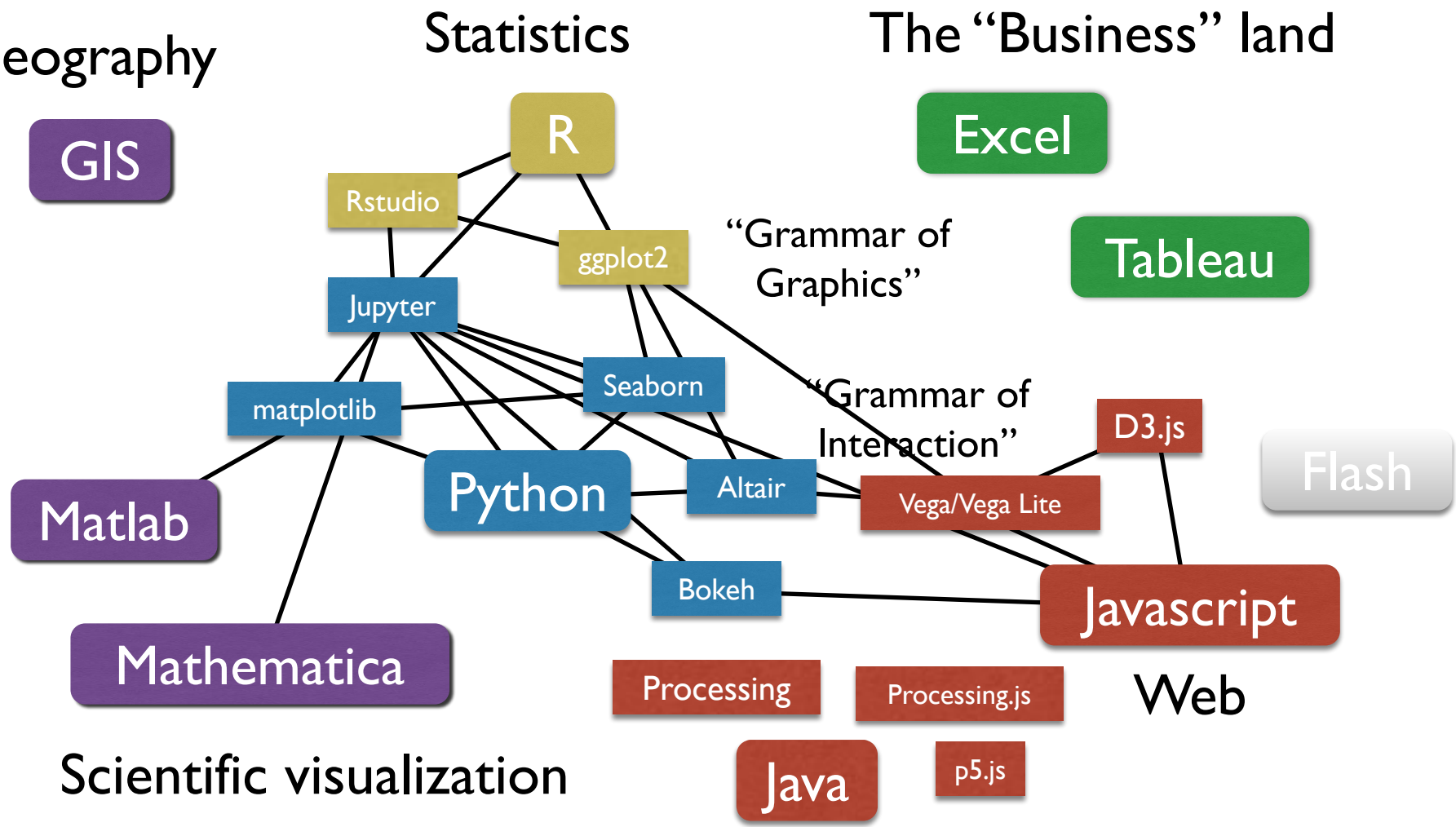
The Landscape

Python's Visualization Landscape



Jake VanderPlas @jakevdp

UNIVERSITY of WASHINGTON eScience Institute


Geography
Statistics
The "Business" land
GIS
R
Excel
Rstudio
ggplot2
"Grammar of Graphics"
Tableau
Jupyter
Seaborn
matplotlib
"Grammar of Interaction"
D3.js
Flash
Matlab
Python
Altair
Vega/Vega Lite
Bokeh
Javascript
Mathematica
Processing
Processing.js
Web
Java
p5.js
Scientific visualization

Python “stack” for data science

Jupyter notebook: interactive environment for manipulation, analysis, and visualization.

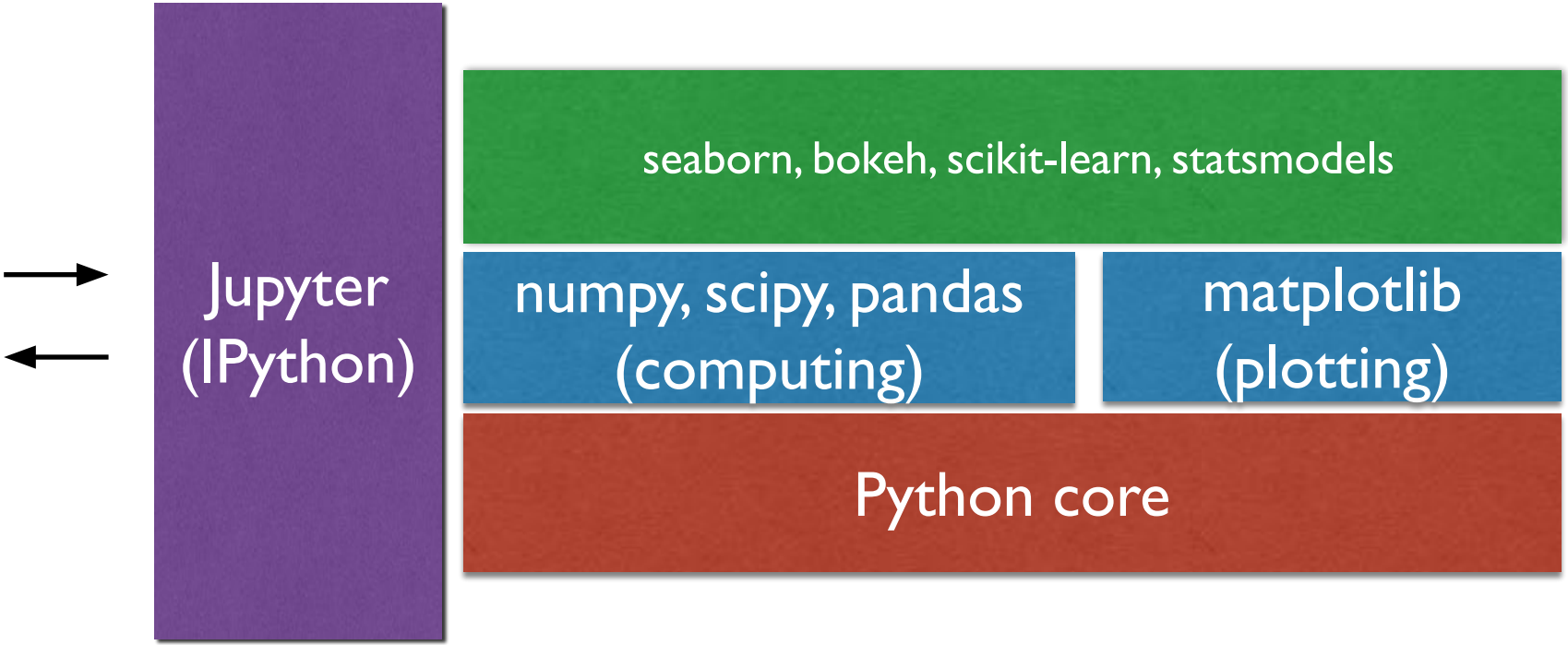
Numpy and **SciPy**: packages for scientific computing

Pandas: data analysis toolkit

Matplotlib, **Seaborn**, **Altair**, …: data visualization

Scikit-learn, **statsmodels, pytorch, tensorflow, …**: machine learning and statistics

Python data analysis stack

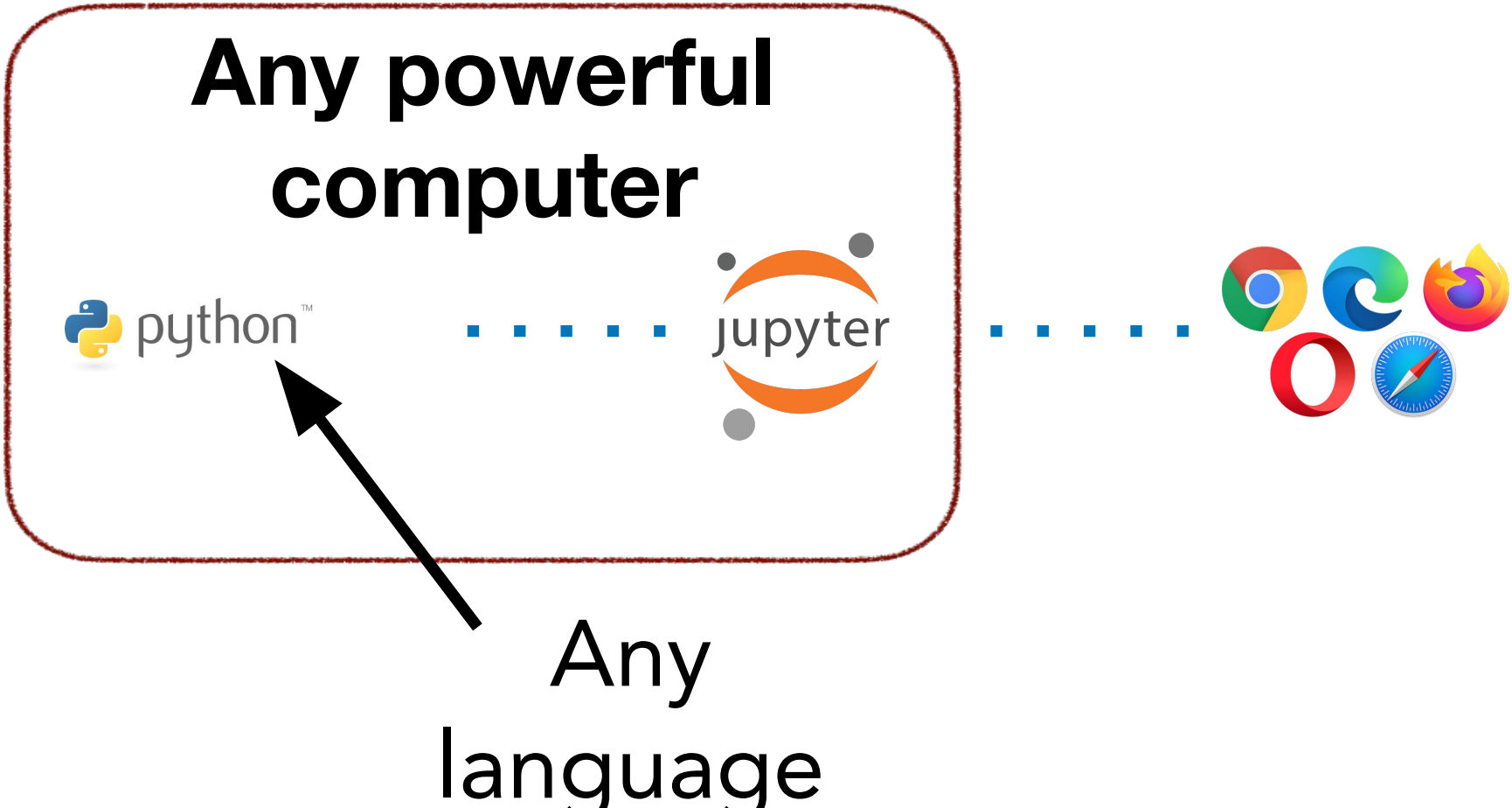


Kernel

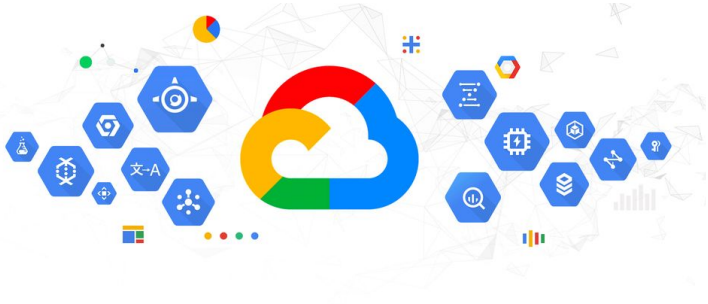
Web browser

Kernel

Web browser


Any powerful computer
python
jupyter
Any language

https://colab.research.google.com/notebooks/intro.ipynb



If you want to work locally… Probably the easiest way to use the full Python Stack:

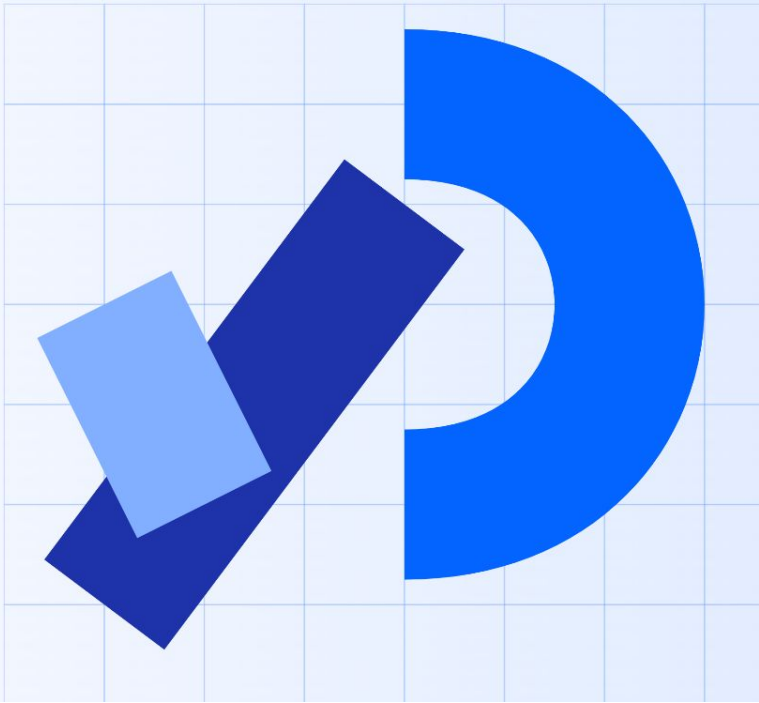
Anaconda distribution

- https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html (download python 3)
- Pre-installed packages
- conda: A decent package manager + virtual environments manager

Welcome to Processing!

Processing is a flexible software sketchbook and a language for learning how to code. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

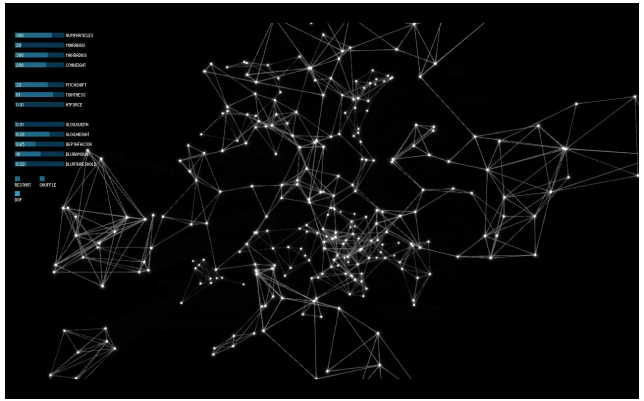
Download Reference Donate



Example (Processing)

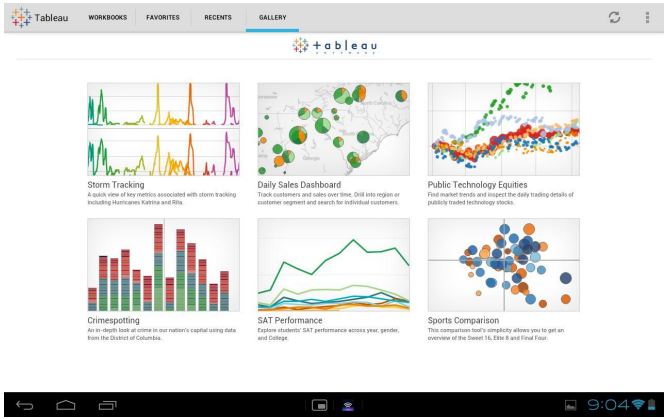
https://www.youtube.com/watch?v=4glhRkCcD4U

Why not Processing?



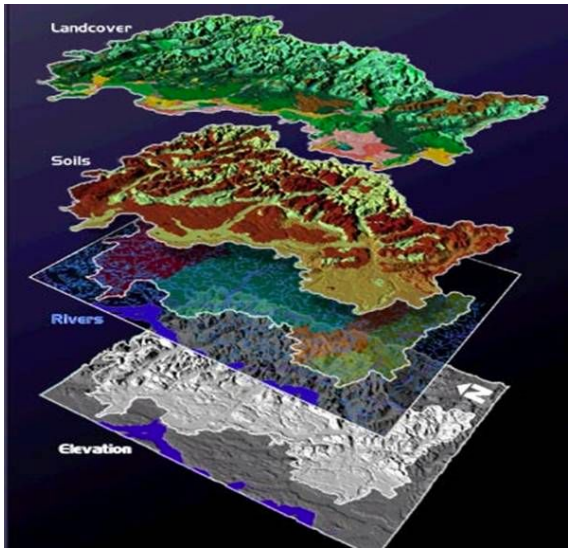
- Java-based (can use java libraries).
- Very good at creating high quality visualization and animations.
- Used a lot for artsy projects and movies.
- Much less development on the data visualization side → Pretty much everything should be created by yourself.

Why not Tableau?



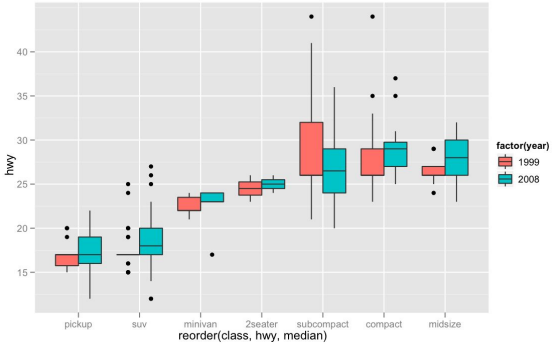
- It’s a pretty good tool!
- Good defaults, easy-to-use interface, sharing features.
- But it’s a proprietary tool and the skills less transferrable to other tools, especially because it does too much for you.

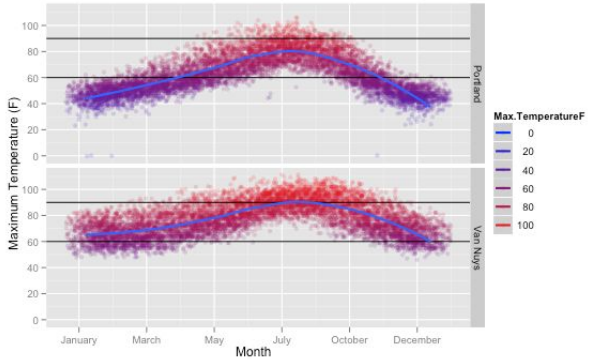
Why not GIS (arcgis, …)?



- A set of tools for geographical data
- Too specific to geo-data
- Python/R can do pretty decent jobs for most (simpler) tasks

Why not R?



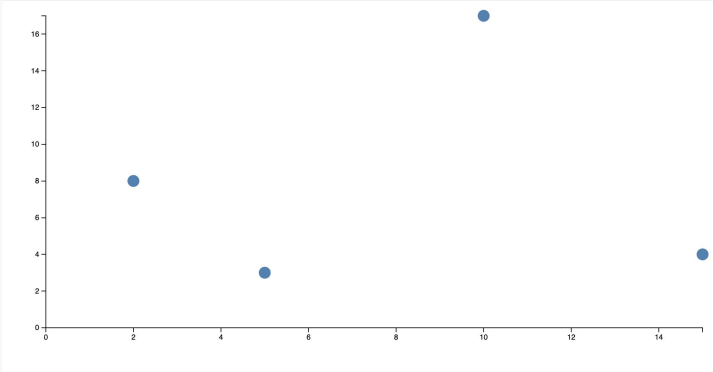


- Python is more widely used in scientific computing and data science in general; our curriculum uses Python as the de facto standard.
- Probably the best language for **statistics-focused applications and when you already have a dataset.**
- The language itself may not be as general-purpose as Python, but strong package ecosystem makes up for that.
- One of the most elegant statistical visualization package **ggplot2** & a really strong IDE (**RStudio**)

Example (R):

Why not D3.js?

Simple Scatter Chart Example



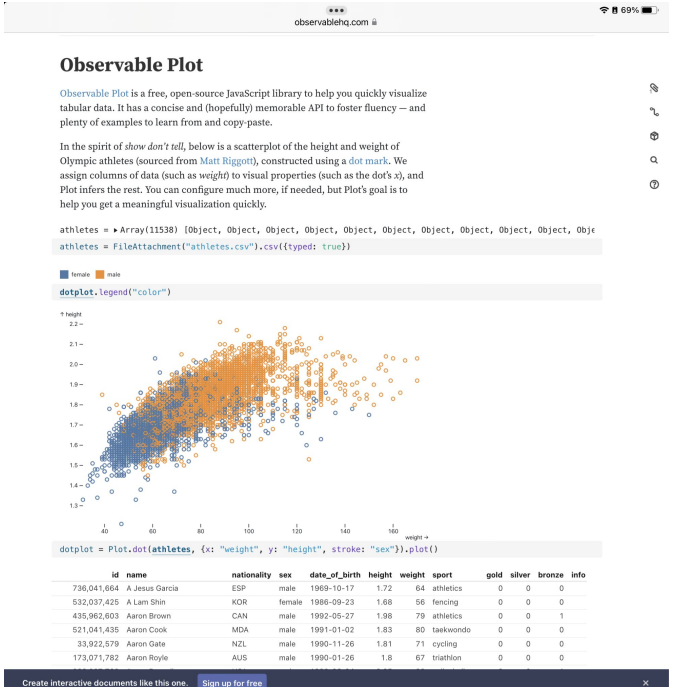

```
// data that you want to plot, I've used separate arrays for x and y values
var xdata = [5, 10, 15, 20],
    ydata = [3, 17, 4, 6];

// size and margins for the chart
var margin = {top: 20, right: 15, bottom: 60, left: 60}
  , width = 960 - margin.left - margin.right
  , height = 500 - margin.top - margin.bottom;

// x and y scales, I've used linear here but there are other options
// the scales translate data values to pixel values for you
var x = d3.scale.linear()
          .domain([0, d3.max(xdata)])  // the range of the values to plot
          .range([ 0, width ]);        // the pixel range of the x-axis

var y = d3.scale.linear()
          .domain([0, d3.max(ydata)])
          .range([ height, 0 ]);

// the chart object, includes all margins
var chart = d3.select('body')
.append('svg:svg')
.attr('width', width + margin.right + margin.left)
.attr('height', height + margin.top + margin.bottom)
.attr('class', 'chart')

// the main object where the chart and axis will be drawn
var main = chart.append('g')
.attr('transform', 'translate(' + margin.left + ',' + margin.top + ')')
.attr('width', width)
.attr('height', height)
.attr('class', 'main')

// draw the x axis
var xAxis = d3.svg.axis()
.scale(x)
.orient('bottom');

main.append('g')
.attr('transform', 'translate(0,' + height + ')')
.attr('class', 'main axis date')
.call(xAxis);

// draw the y axis
var yAxis = d3.svg.axis()
.scale(y)
.orient('left');

main.append('g')
.attr('transform', 'translate(0,0)')
.attr('class', 'main axis date')
.call(yAxis);

// draw the graph object
var g = main.append("svg:g");

g.selectAll("scatter-dots")
  .data(ydata)  // using the values in the ydata array
  .enter().append("svg:circle")  // create a new circle for each value
      .attr("cy", function (d) { return y(d); } ) // translate y value to a pixel
      .attr("cx", function (d,i) { return x(xdata[i]); } ) // translate x value
      .attr("r", 10) // radius of circle
      .style("opacity", 0.6); // opacity of circle
```

Data-Driven Documents

Why not D3.js?

- Javascript based: Web! Interactions!
- De-facto standard for **high-quality, highly custom** web-based visualizations
- It's procedural (you should draw points, axis, lines, etc. by using javascript) and too cumbersome to learn (lots of overhead).
- Emerging alternatives: declarative web-based visualization tools such as **Vega** (vega-lite, altair)

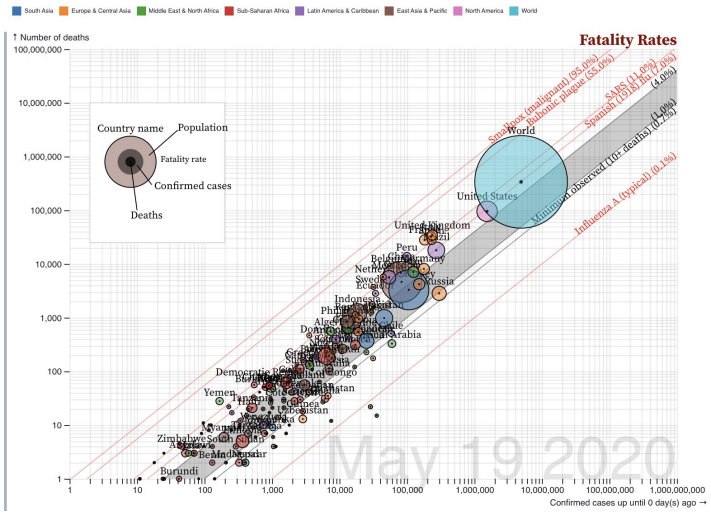
Observable (Platform)



- Reactive JavaScript notebook
- Both d3.js and Vega system can be used.
- New plotting library “Observable Plot”
- This is the recommended tool if you want to create **interactive visualizations on the web**.

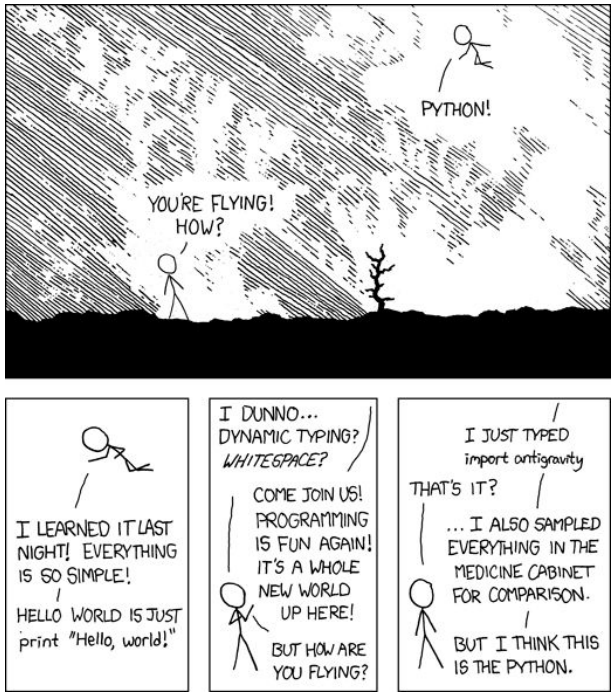
Observable + d3.js example

https://observablehq.com/@yy/covid-19-fatality-rate



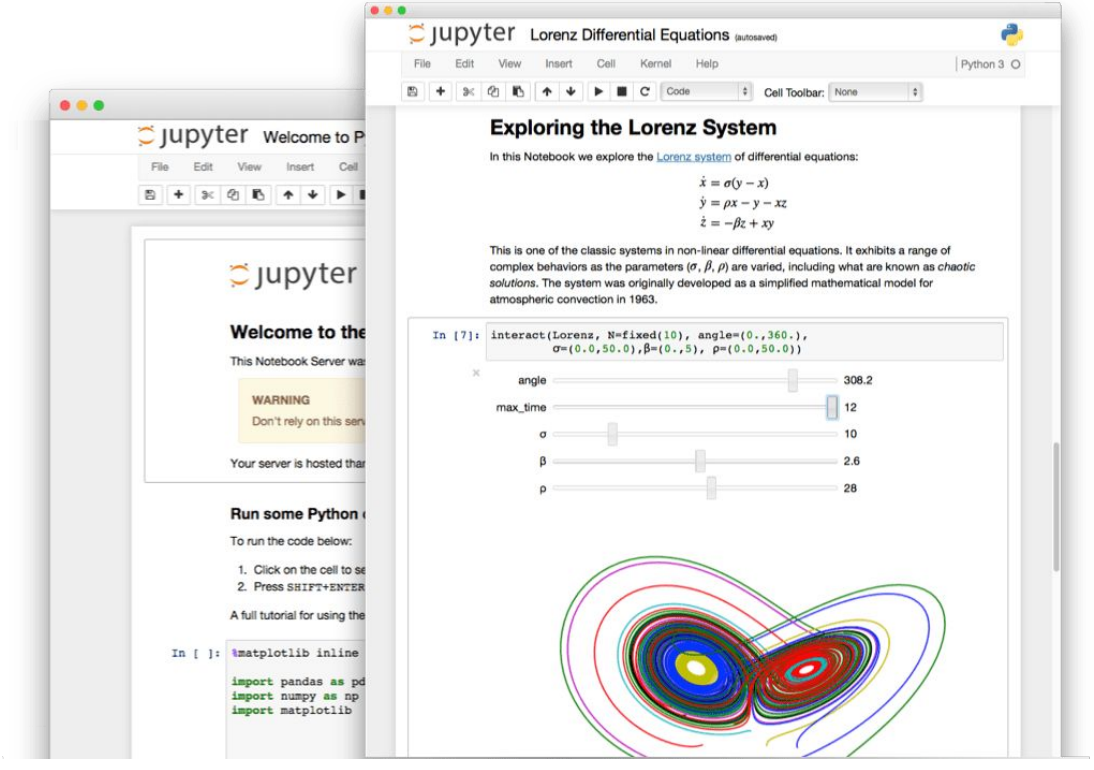
Then, why Python?

Why Python?



- A strong **general-purpose** language that can fit into various data pipelines.
- The mighty **Jupyter notebook/lab**
- Also mighty python data science stack that spans low-level high performance computing to data visualization and machine learning (numpy, scipy, pandas, scikit-learn, pytorch, tensorflow, …)
- **Huge** user-base (scientific computing and data science).
  - Many strong libraries
  - Fast development
  - Easy to get help

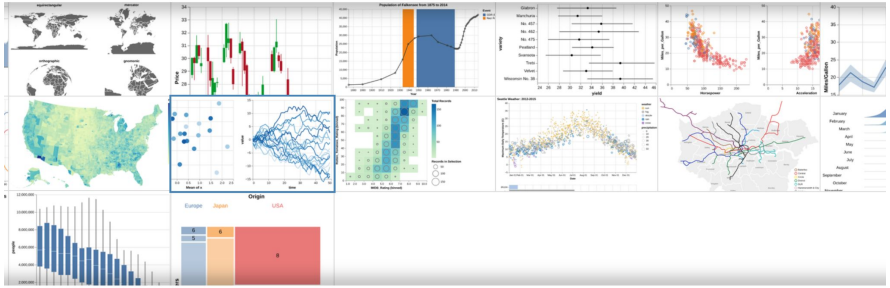
Jupyter notebook/lab



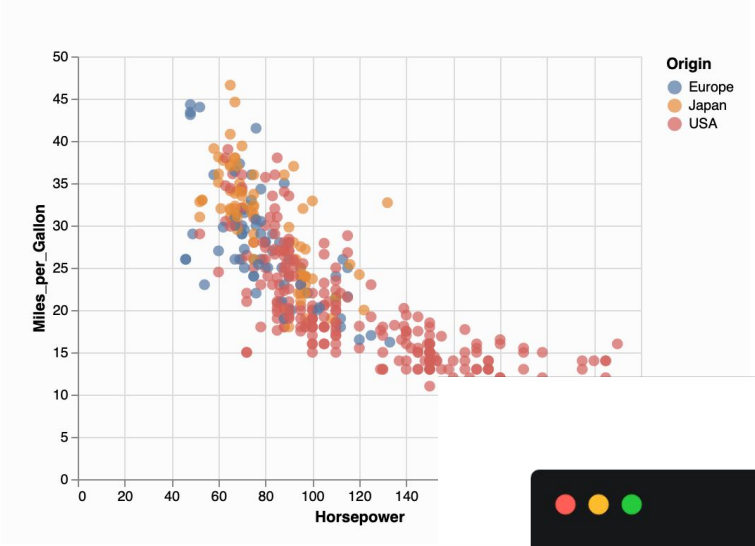
Why Altair (and Vega, Vega-lite)?

- **Declarative** (not system- or language-specific)
- Inter-operable
- High-level **grammar** rather than instructions for drawing individual elements.
- Less overhead to put standard visualizations on the web.
- Sufficiently developed (can produce many of the standard visualizations that you can create with D3.js)

Vega-Altair: Declarative Visualization in Python

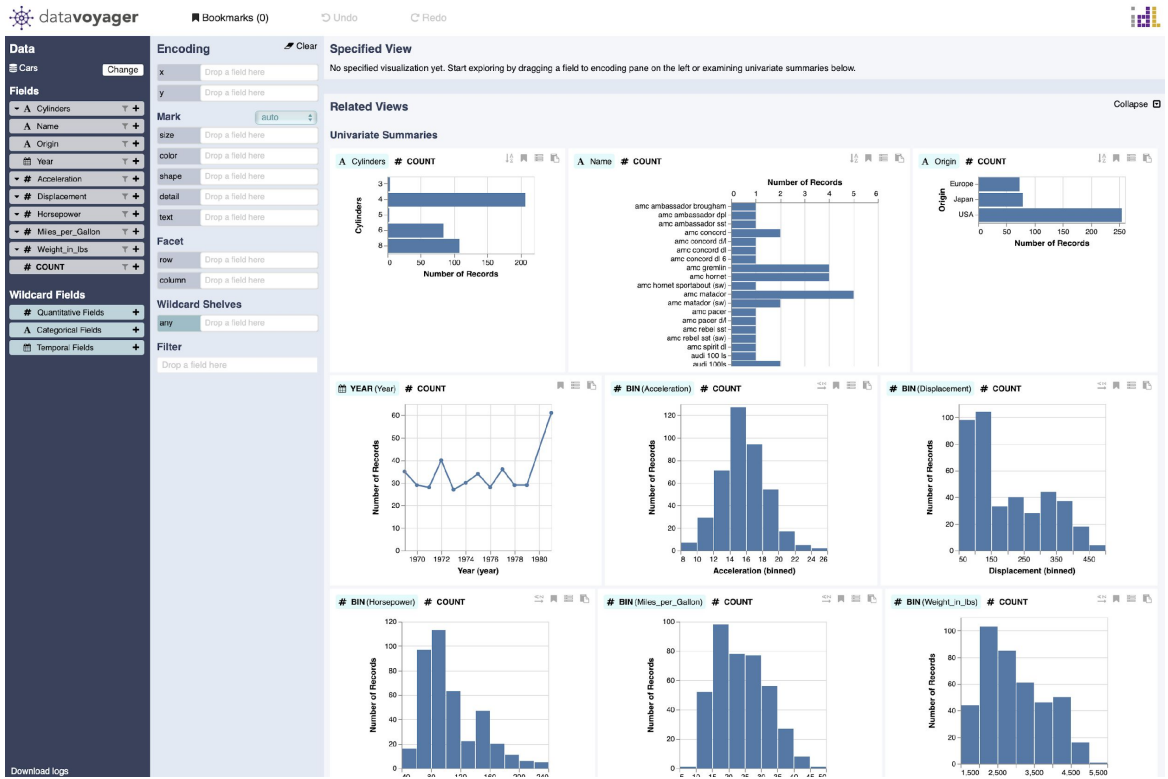


Vega-Altair is a declarative visualization library for Python. Its simple, friendly and consistent API, built on top of the powerful Vega-Lite grammar, empowers you to spend less time writing code and more time exploring your data.




```python
import altair as alt
from vega_datasets import data

source = data.cars()

alt.Chart(source).mark_circle(size=60).encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color='Origin',
    tooltip=['Name', 'Origin', 'Horsepower', 'Miles_per_Gallon']
).interactive()
```

Demo: Data Voyager
(Tableau-like interface & plot discovery)



https://github.com/vega/voyager

Vega vs. D3.js

- https://vega.github.io/vega/about/vega-and-d3/
- D3.js will still be useful to create novel visualizations with tailored details, but for many standard visualizations, Vega ecosystem provides a nicer way.
- You can use both Vega(lite) and D3.js in observablehq!

@Canvas: Week 1 ->
[Assignment] Python environment setup:

Install packages that we will use and submit the notebook (details are in the instruction).

Please ask help (Canvas, office hour) if you have any issues!

Reading for the next class

Tufte Ch. 1 (Module 2 on Canvas)