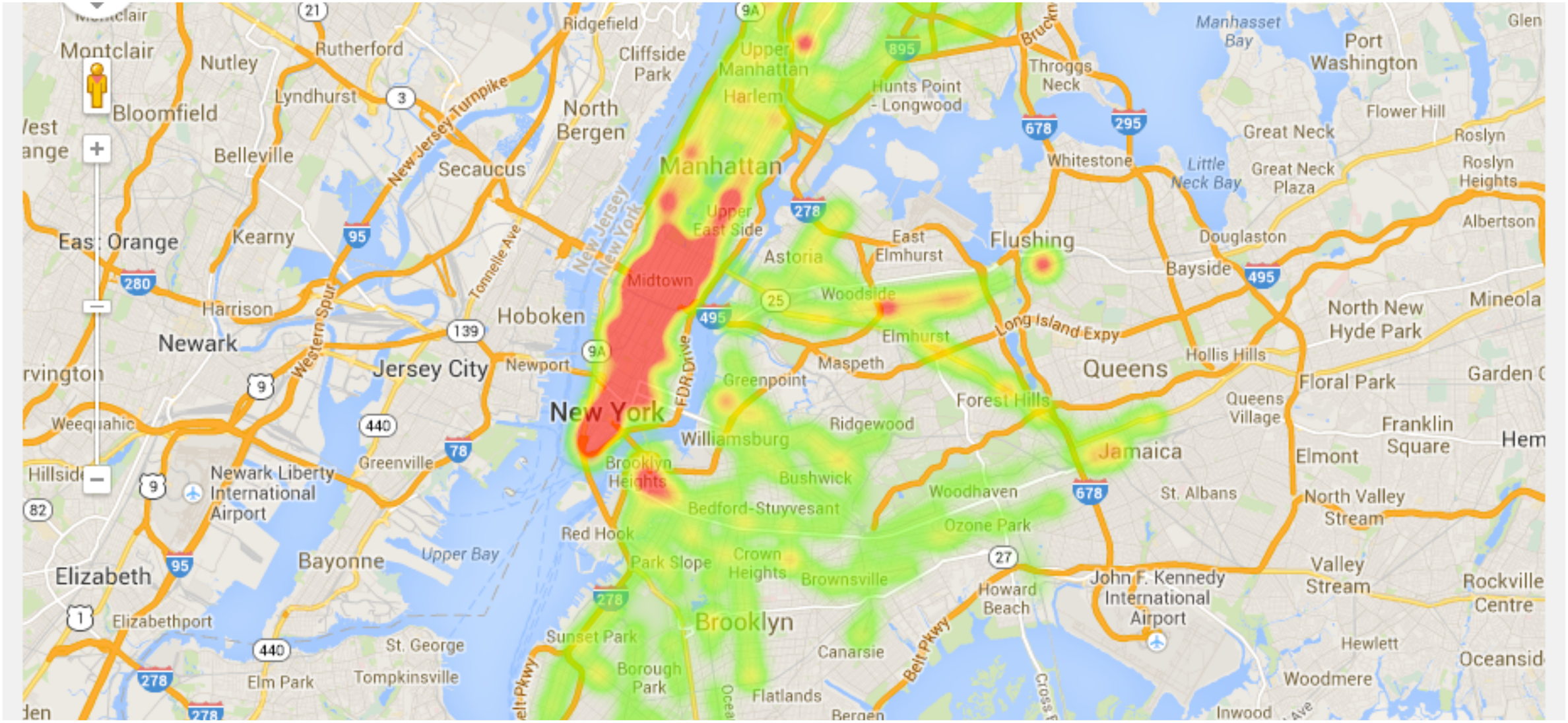
Programming Principles in Python (CSCI 503)

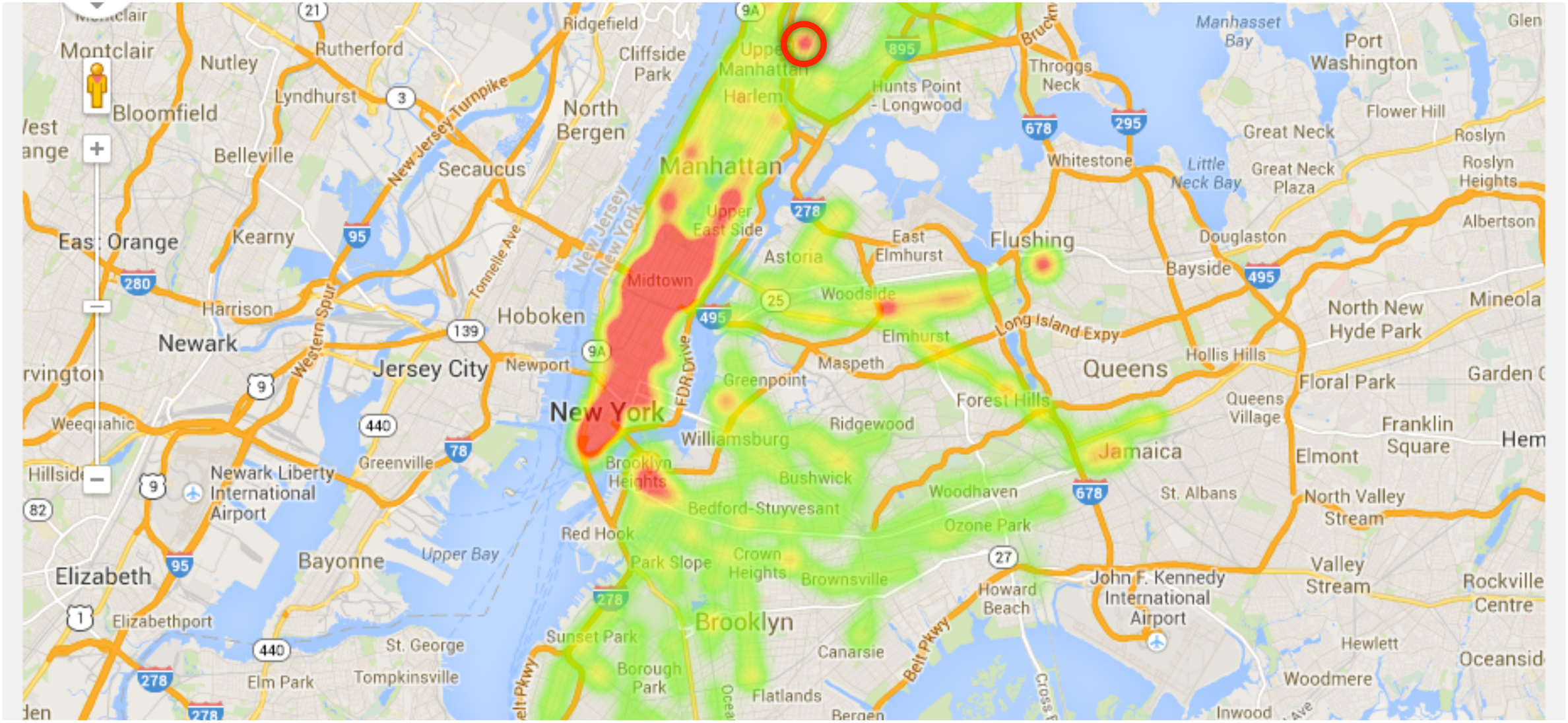
Visualization

Dr. David Koop

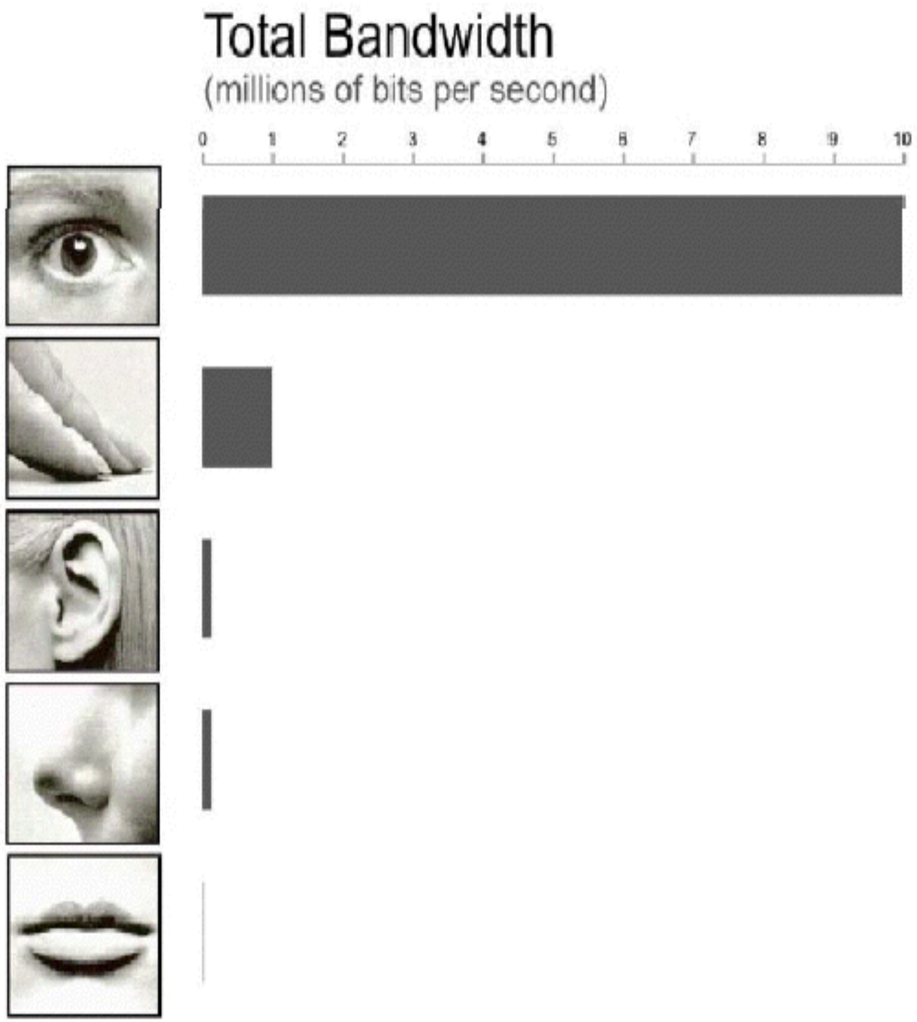
Exploring Data through Visualization



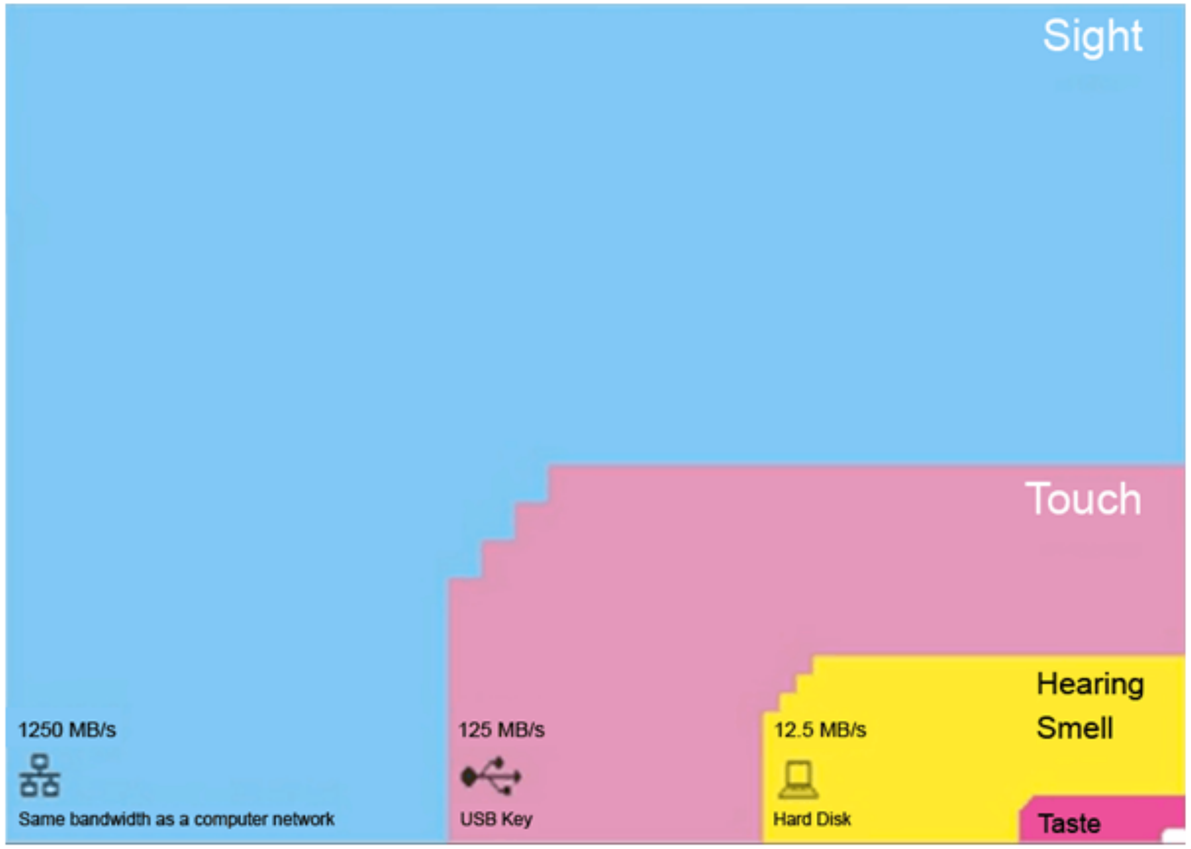
Exploring Data through Visualization



Why do we visualize data?



[via A. Lex]



[T. Nørretranders]

Why Visual?

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

[F. J. Anscombe]

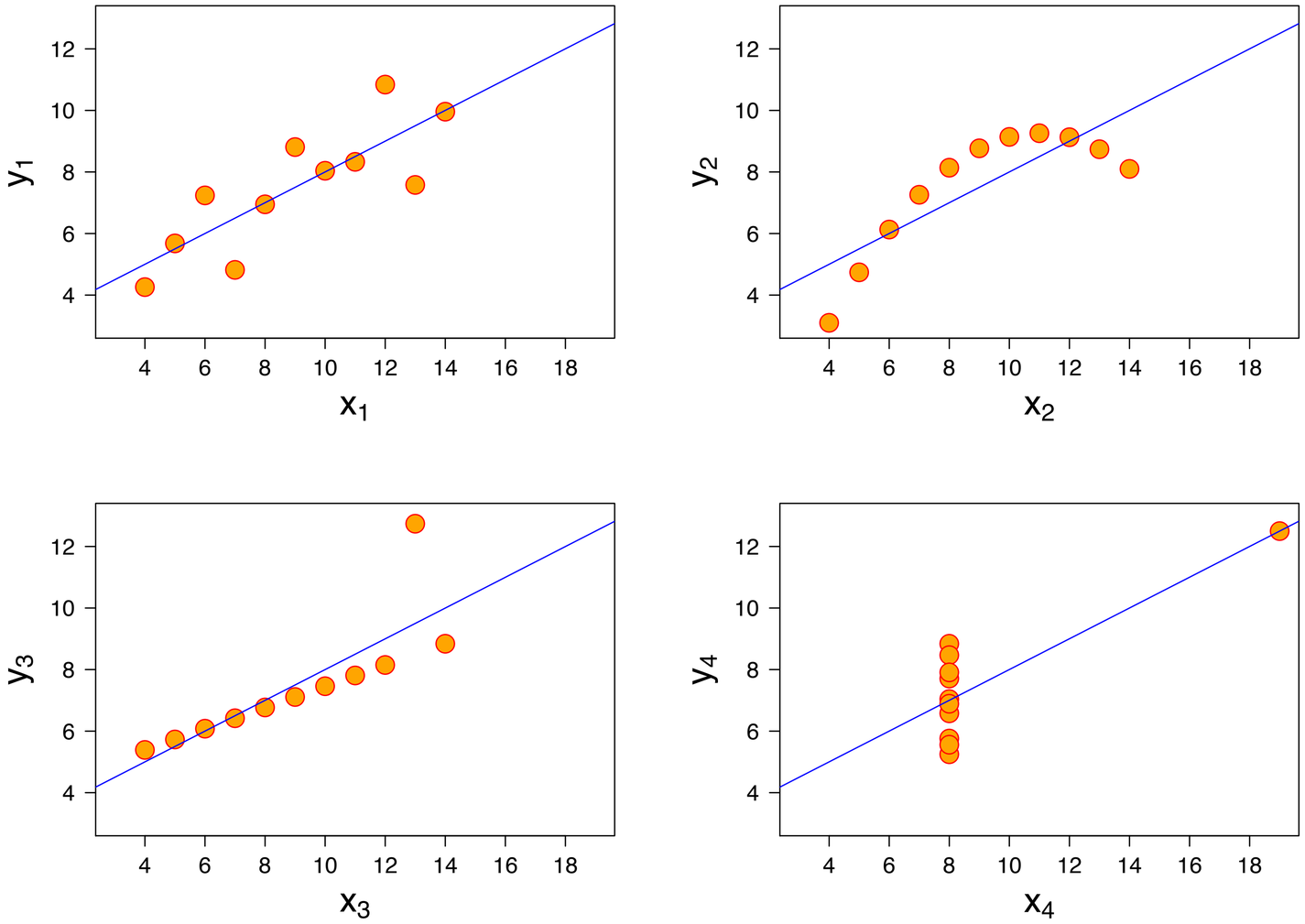
Why Visual?

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

Mean of x	9
Variance of x	11
Mean of y	7.50
Variance of y	4.122
Correlation	0.816

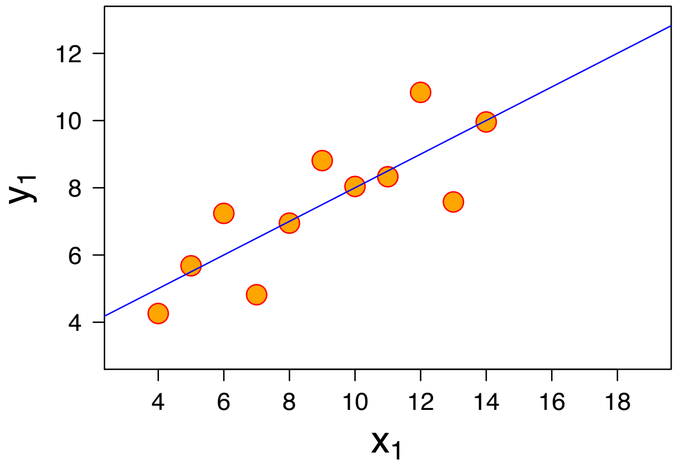
[F. J. Anscombe]

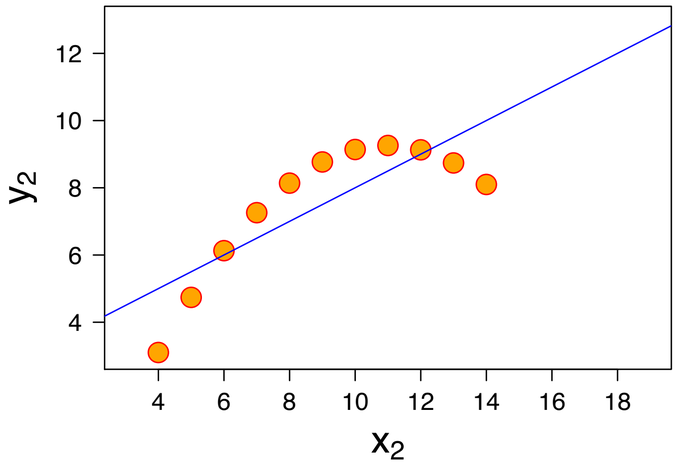
Why Visual?

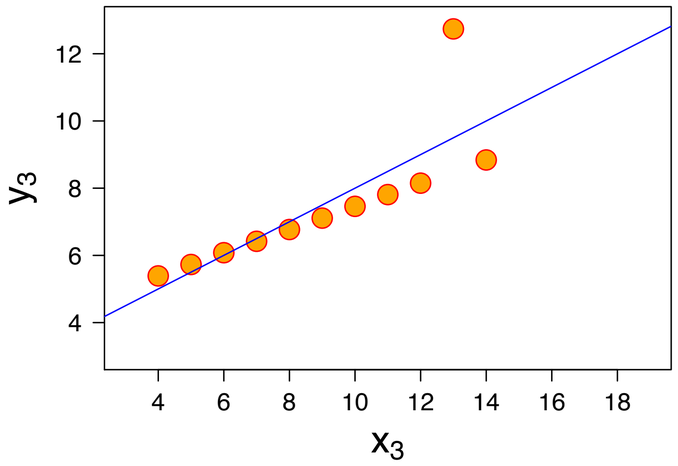


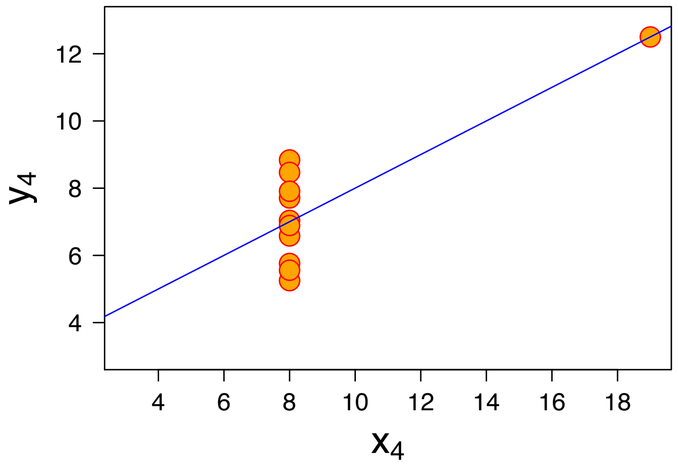
[F. J. Anscombe]

Why Visual?









Mean of x	9
Variance of x	11
Mean of y	7.50
Variance of y	4.122
Correlation	0.816

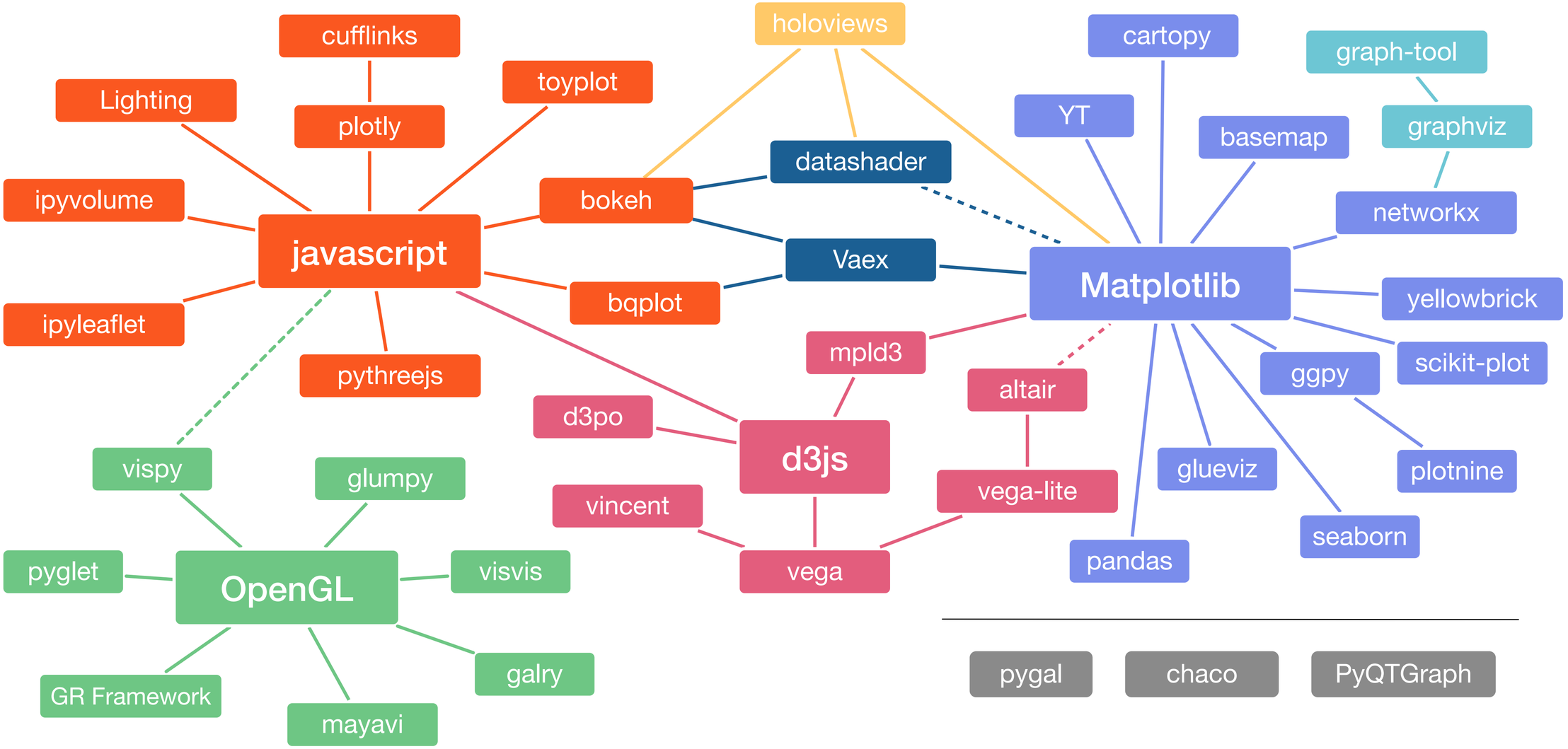
[F. J. Anscombe]

Visualization Goals

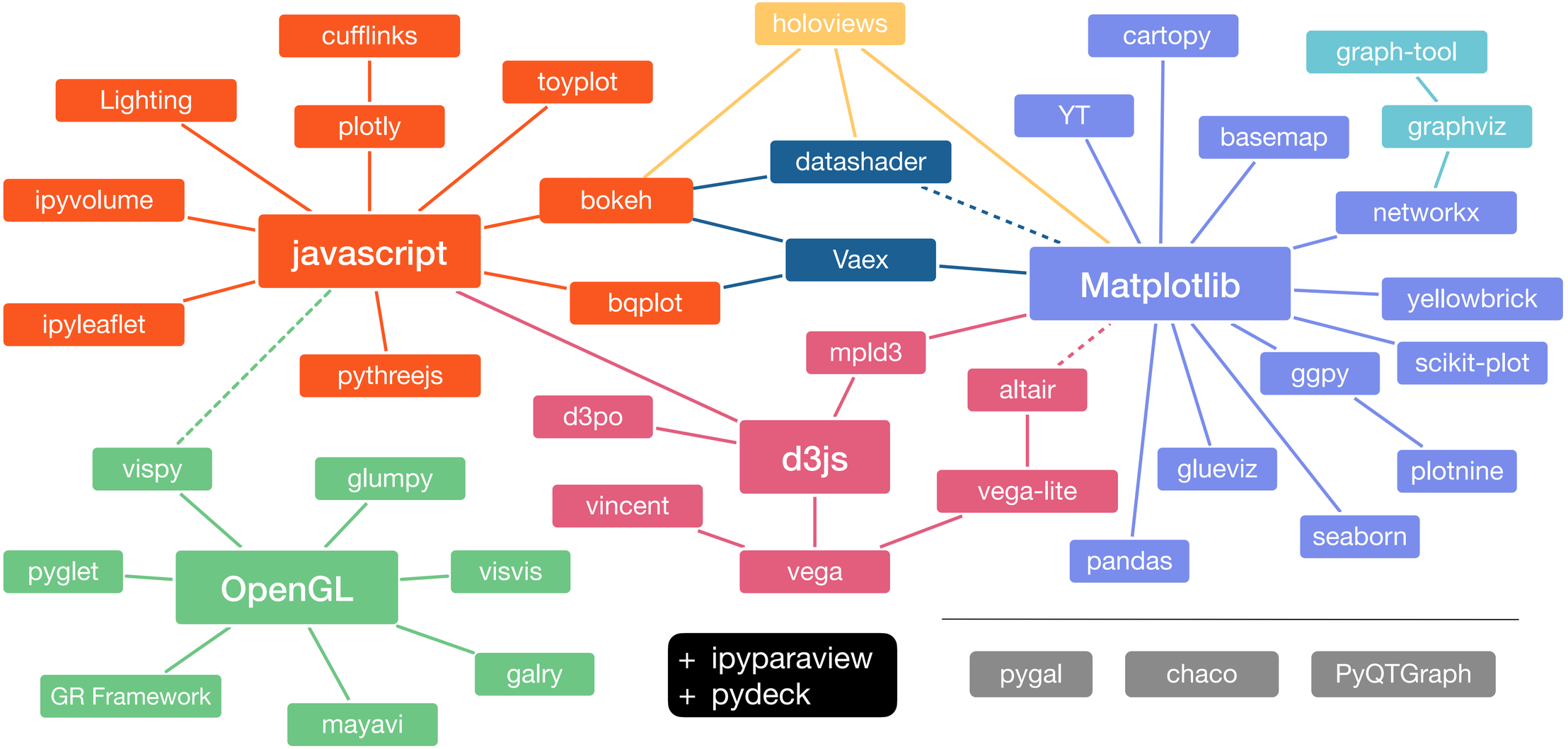
- "The purpose of visualization is **insight**, not pictures" – B. Schneiderman

- Identify patterns, trends
- Spot outliers
- Find similarities, correlation

The Python Visualization Landscape

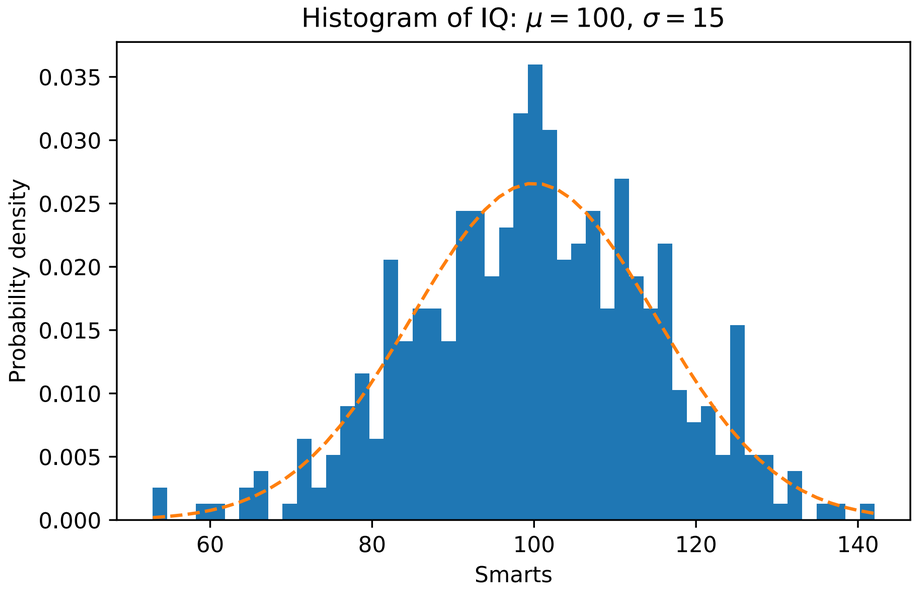


The Python Visualization Landscape



matplotlib

- Strengths:
  - Designed like Matlab
  - Many rendering backends
  - Can reproduce almost any plot
  - Proven, well-tested
- Weaknesses:
  - API is imperative
  - Not originally designed for the web
  - Dated styles

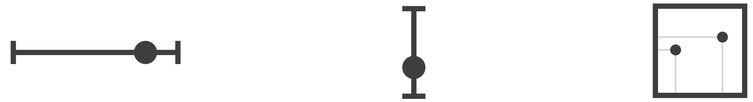


Basic Example

- `import matplotlib.pyplot as plt`
  `plt.plot([1,5,2,7,3])`
- Default is line plot
- x-values are implicit (`range(5)`)
- Can add x-values
  - `plt.plot([1,3,4,6,10],[1,5,2,7,3])`
- Can change type of plot
  - `plt.scatter([1,3,4,6,10],[1,5,2,7,3])`
  - `plt.plot([1,3,4,6,10],[1,5,2,7,3],'o') # format string`

Data is Encoded via Visual Channels

- **Position**
  - Horizontal
  - Vertical
  - Both



- **Color**

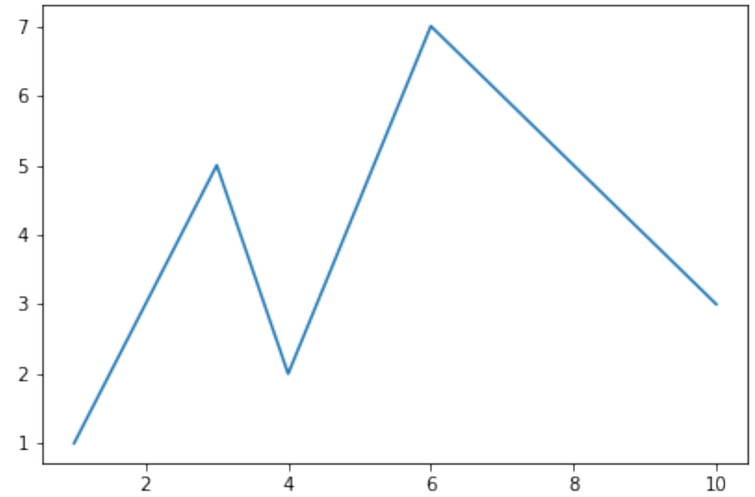
- **Shape**

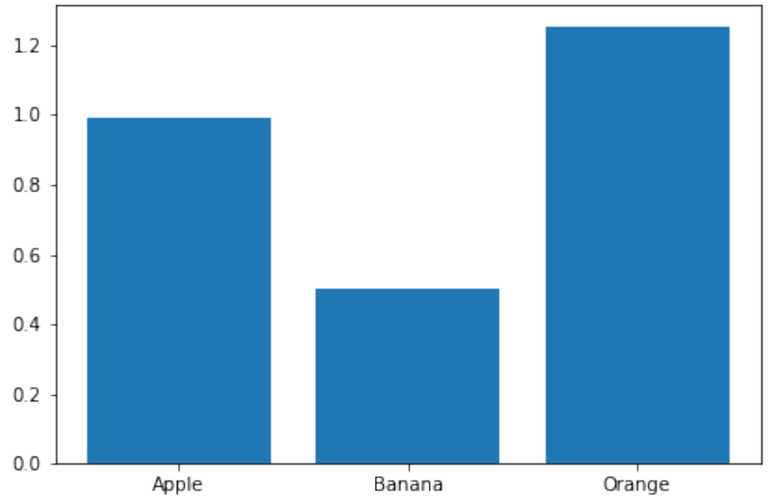
- **Tilt**

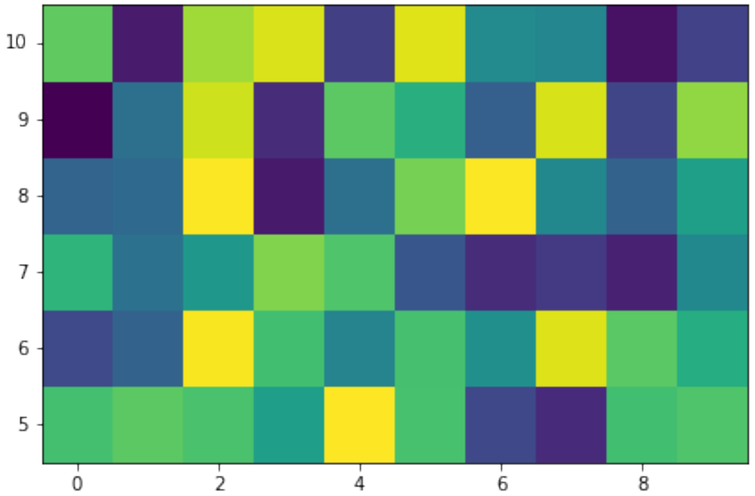
- **Size**
  - Length
  - Area
  - Volume

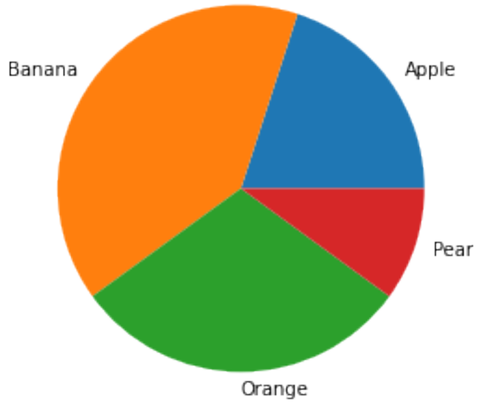
[Munzner (ill. Maguire), 2014]

Many different types of charts

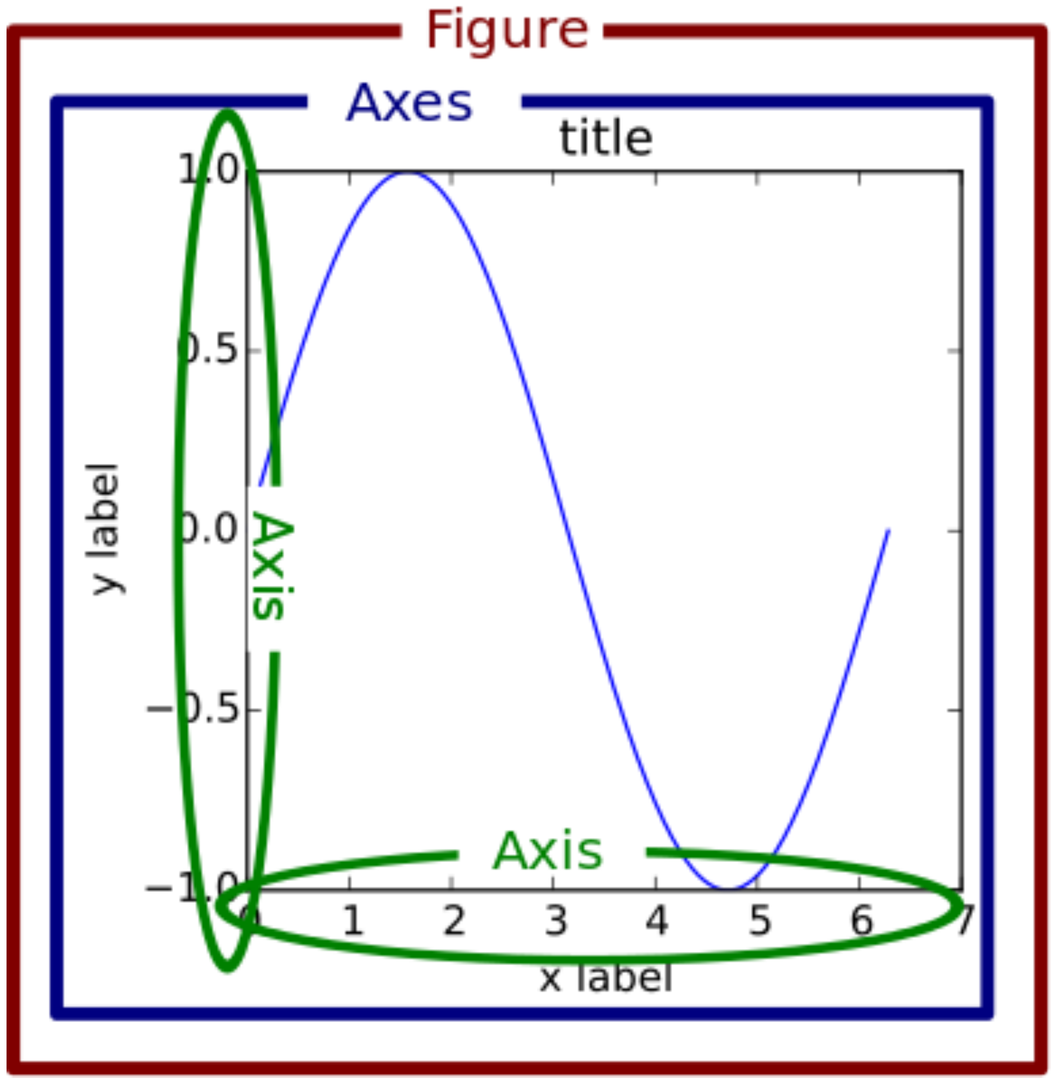


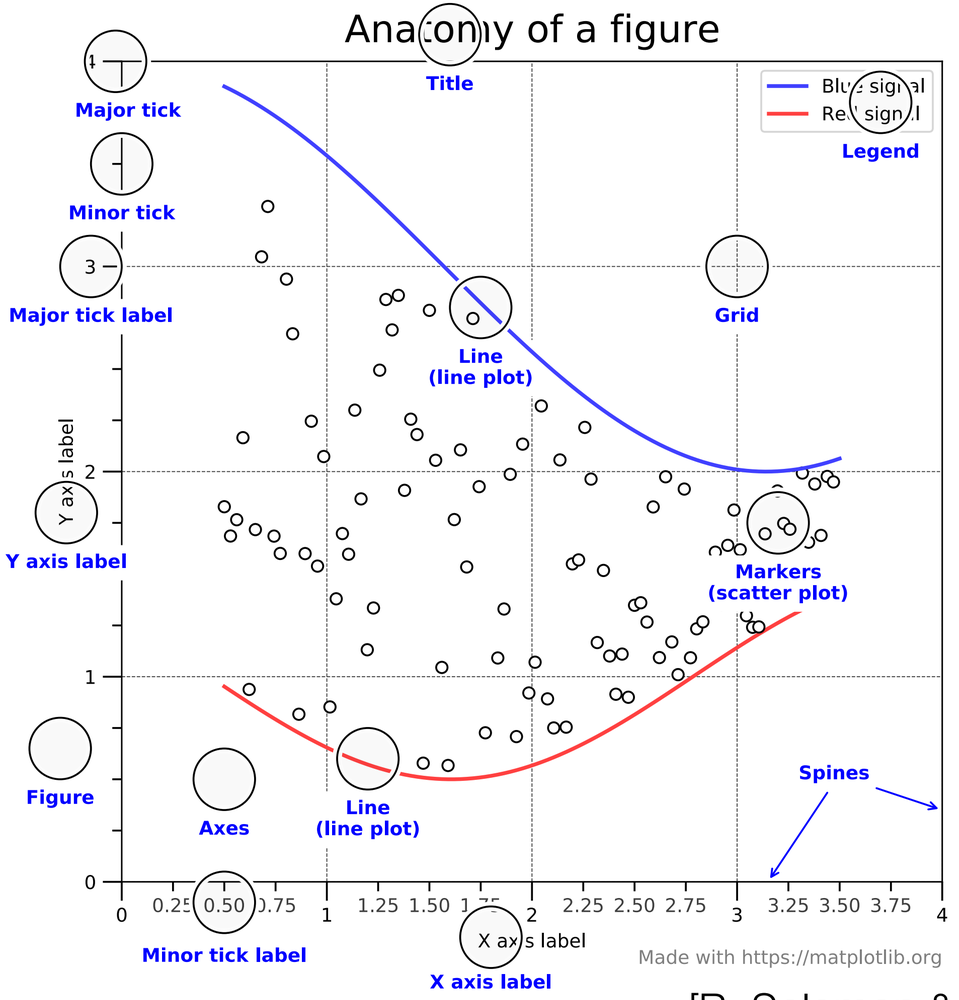






Anatomy of a Figure





[B. Solomon & matplotlib]

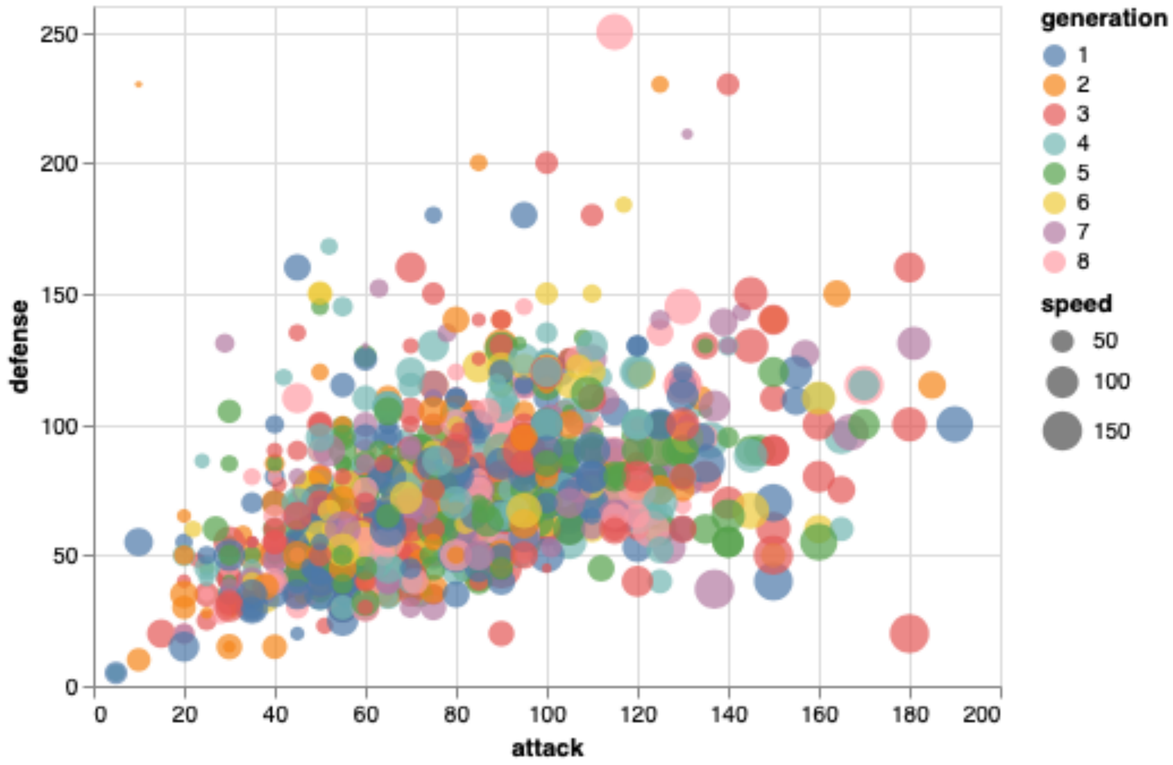
Object-Based Plotting

- `fig, ax = plt.subplots() # "constructor-like" method`
  `ax.scatter([1,3,4,6,10],[1,5,2,7,3])`
- Use getters/setters for labels and title
  - `ax.set_xlabel('Age')`
  - `ax.set_ylabel('Number of Jumps')`
  - `ax.set_title('Kangaroo Jumps Today')`
- We can also call methods on the figure:
  - `fig.tight_layout() # reduce margins`

pandas Integration

- Can call many of these methods directly from pandas
- Handled through `kind` kwarg or `.plot` accessor
- It will try to guess a reasonable visualization, but may fail:
  - `fruit.plot()`
- Instead, specify x and y and other parameters:
  - `fruit.plot(kind='bar',x='name',y='price')`
  - `plt.bar(x='name',height='price',data=fruit) # SIMILAR`
  - `fruit.plot.scatter(x='price',y='count',c='name') # ERROR`
  -
```
colors = {'Apple': 'red','Orange': 'orange',
          'Banana': 'yellow','Pear': 'green'}
fruit.plot.scatter(x='price',y='count',
                   c=fruit['name'].map(colors))
```

Assignment 8



- Back to Pokémon Data
- Calculate MaxCP in pandas and find highest per generation
- Analyze attack, defense, and speed by primary type and generation using visualizations created with matplotlib and altair

Final Exam

- Monday, April 26, 2:00-3:50pm, Online (Blackboard)
- **More** comprehensive than Test 2
- Expect questions from topics covered on Test 1 and 2
- Expect questions from the last three weeks of class (data, visualization, machine learning)
- Similar format

Test 2 Comments

Teaching Evaluations

- Online
- Check your email for instructions

History of Vega-Lite & Altair

- "Grammar of Graphics", L. Wilkinson
- "A Layered Grammar of Graphics", H. Wickham
- ggplot: plotting library for R
- Vega: similar idea for Javascript/JSON (U. Washington, A. Satyanarayan)
  - "Declarative language for creating, saving, and sharing interactive visualization designs"
  - More focus on interaction and reactive signals
  - Separation between specification and runtime
- Vega-Lite: higher-level language than Vega (U. Washington, D. Moritz)
  - uses carefully designed rules to default settings

History of Vega-Lite & Altair

- Altair: Python interface to Vega-Lite (J. VanderPlas)
  - "spend more time understanding your data and its meaning"
  - Specify the what, minimize the amount of code directing the how
  - Python can write JSON specification just as well as any other language
  - Bindings make it more Python-friendly, integrate with pandas, add support for Jupyter, etc.

Basic Example

```
import altair as alt
import pandas as pd
data = pd.DataFrame({'x': [1,3,4,6,10],'y': [1,5,2,7,3]})
alt.Chart(data).mark_line().encode(x='x', y='y')
```

- Easiest to use data from a pandas data frame
  - Another option is a csv or json file
  - Can support geo_interface, too
- `Chart` is the basic unit
- `mark_*()` indicates the geometry created for each data item
- `encode` allows visual properties to be set to data attributes

Visual Marks

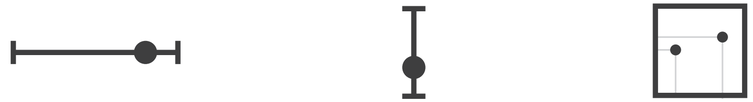
- **Marks** are the basic graphical elements in a visualization
- Marks classified by dimensionality:

Points **Lines** **Areas**

- Also can have surfaces, volumes
- Think of marks as a mathematical definition, or if familiar with tools like Adobe Illustrator or Inkscape, the path & point definitions
- Altair: area, bar, circle, geoshape, image, line, point, rect, rule, square, text, tick
  - Also compound marks: boxplot, errorband, errorbar

Encode via Visual Channels

- **Position**
  - Horizontal
  - Vertical
  - Both



- **Color**

- **Shape**

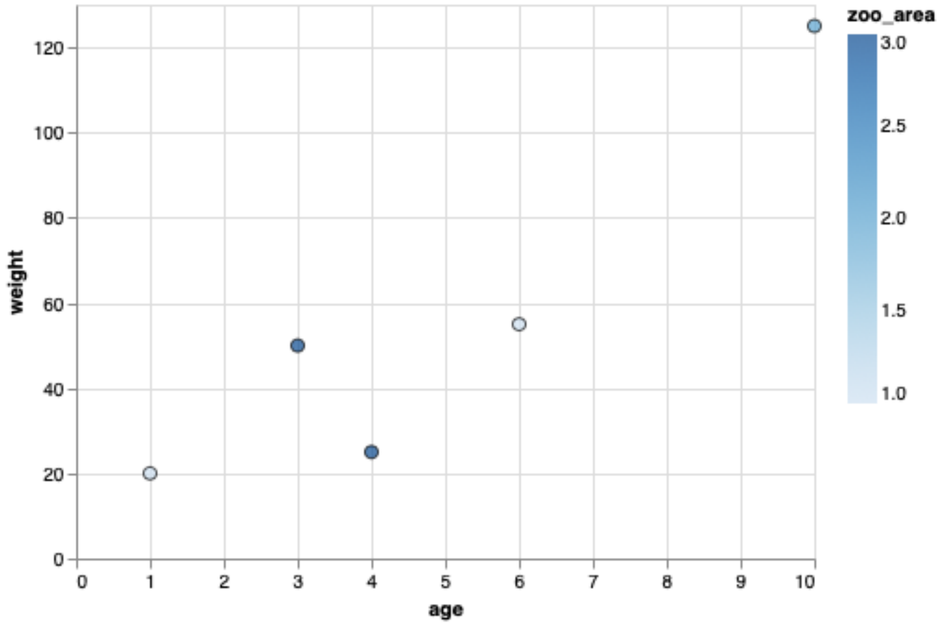
- **Tilt**

- **Size**
  - Length
  - Area
  - Volume

[Munzner (ill. Maguire), 2014]

Easily Explore Different Encodings

```
data = pd.DataFrame({
        'age': [1,3,4,6,10],
        'weight': [20,50,25,55,125],
        'zoo_area': [1,3,3,1,2],
        'num_scoops': [3,2,4,2,3]
})
alt.Chart(data).mark_point(
        filled=True, size=50,
        stroke='black',strokeWidth=1
).encode(
        x='age',
        y='weight',
        color='zoo_area'
)
```



Problem: zoo_area is not a continuous value,
nor is it ordered in any way!

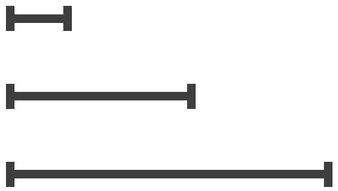
Data Attributes and Altair Types

➔ Categorical

➔ Ordered

➔ *Ordinal*

➔ *Quantitative*



[Munzner (ill. Maguire), 2014]

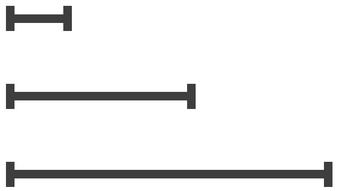
Data Attributes and Altair Types

➔ Categorical

➔ Ordered

➔ *Ordinal*

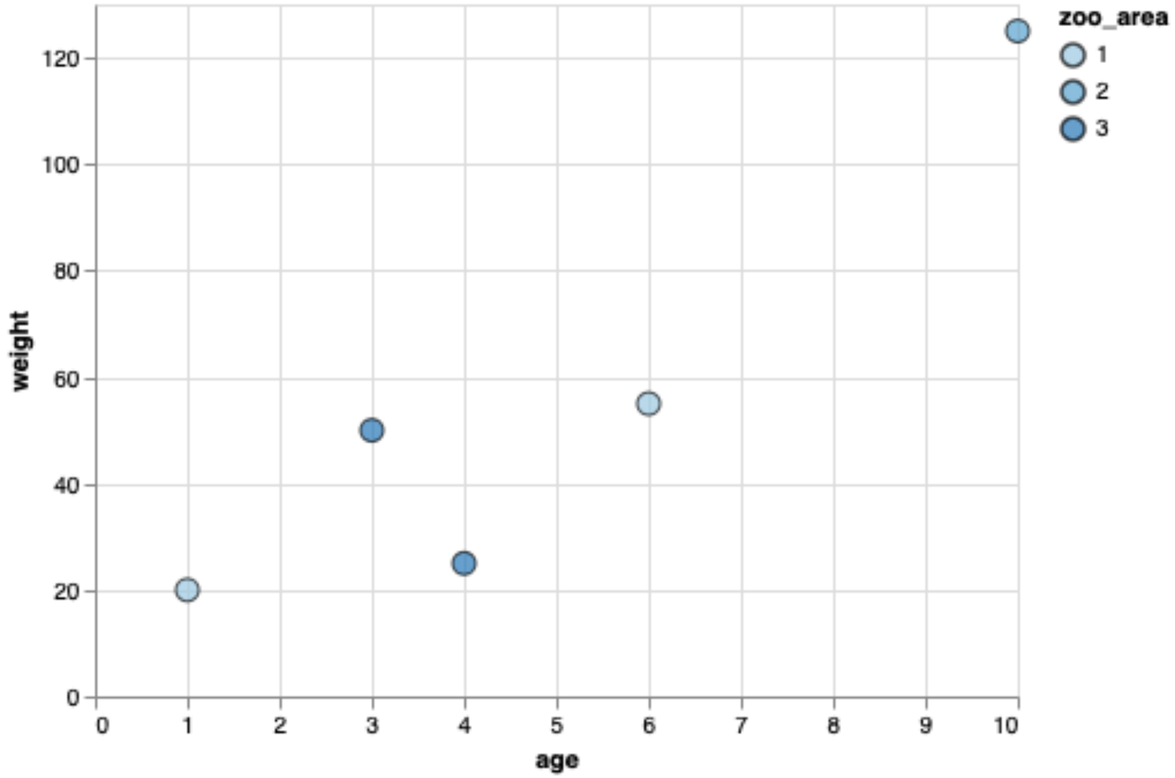
➔ *Quantitative*



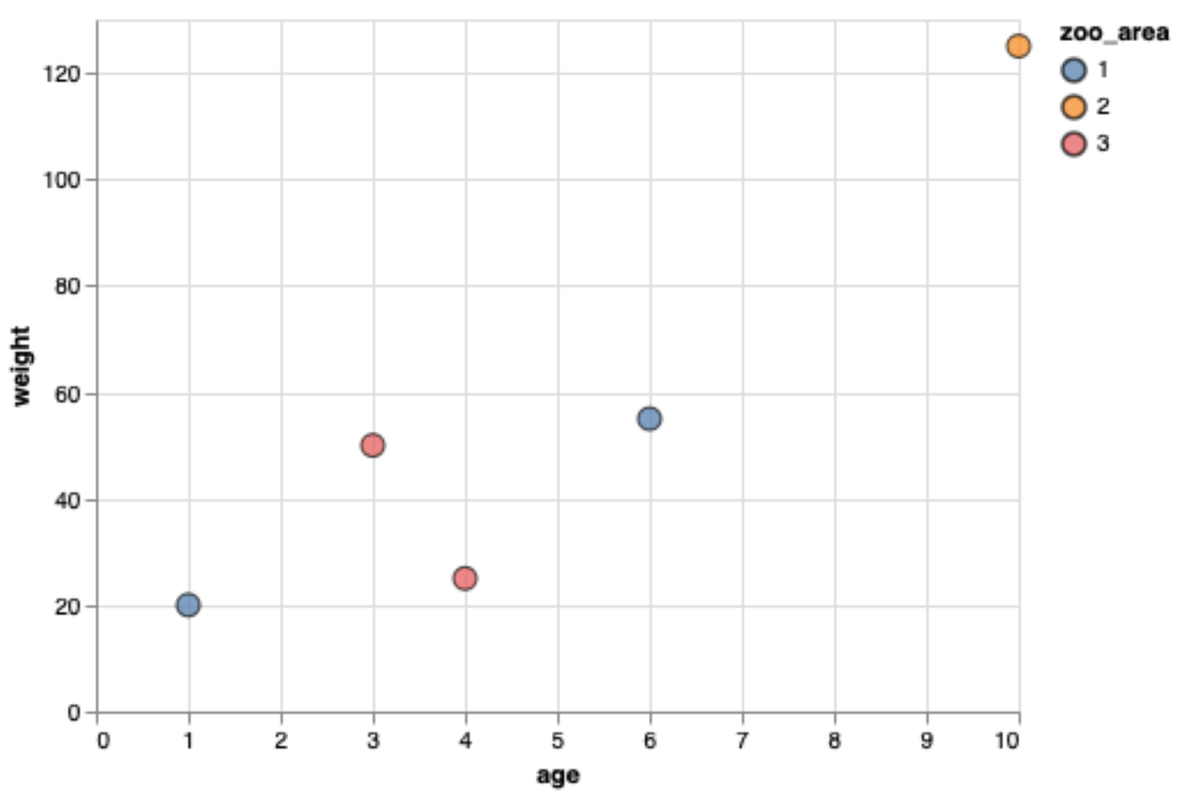
- Categorical data = Nominal (N)
- Ordinal data = Ordinal (O)
- Quantitative data = Quantitative (Q)
- Temporal data = Temporal (T)

[Munzner (ill. Maguire), 2014]

Specifying the Type



`zoo_area:O`

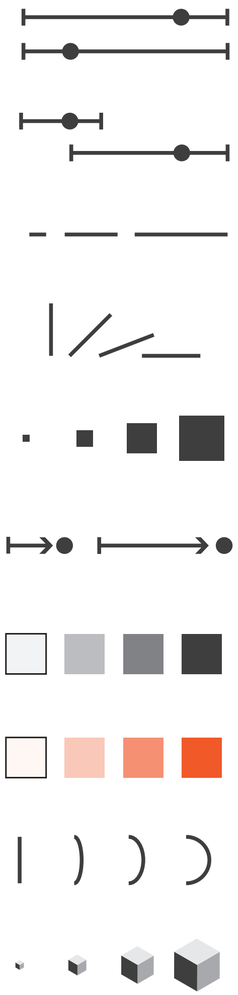


`zoo_area:N`

Different Channels for Different Attribute Types

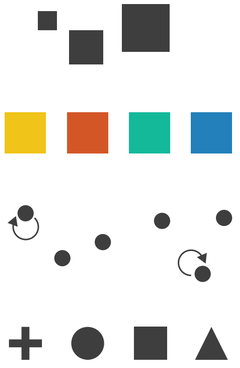
Magnitude Channels: Ordered Attributes

- Position on common scale
- Position on unaligned scale
- Length (1D size)
- Tilt/angle
- Area (2D size)
- Depth (3D position)
- Color luminance
- Color saturation
- Curvature
- Volume (3D size)



Identity Channels: Categorical Attributes

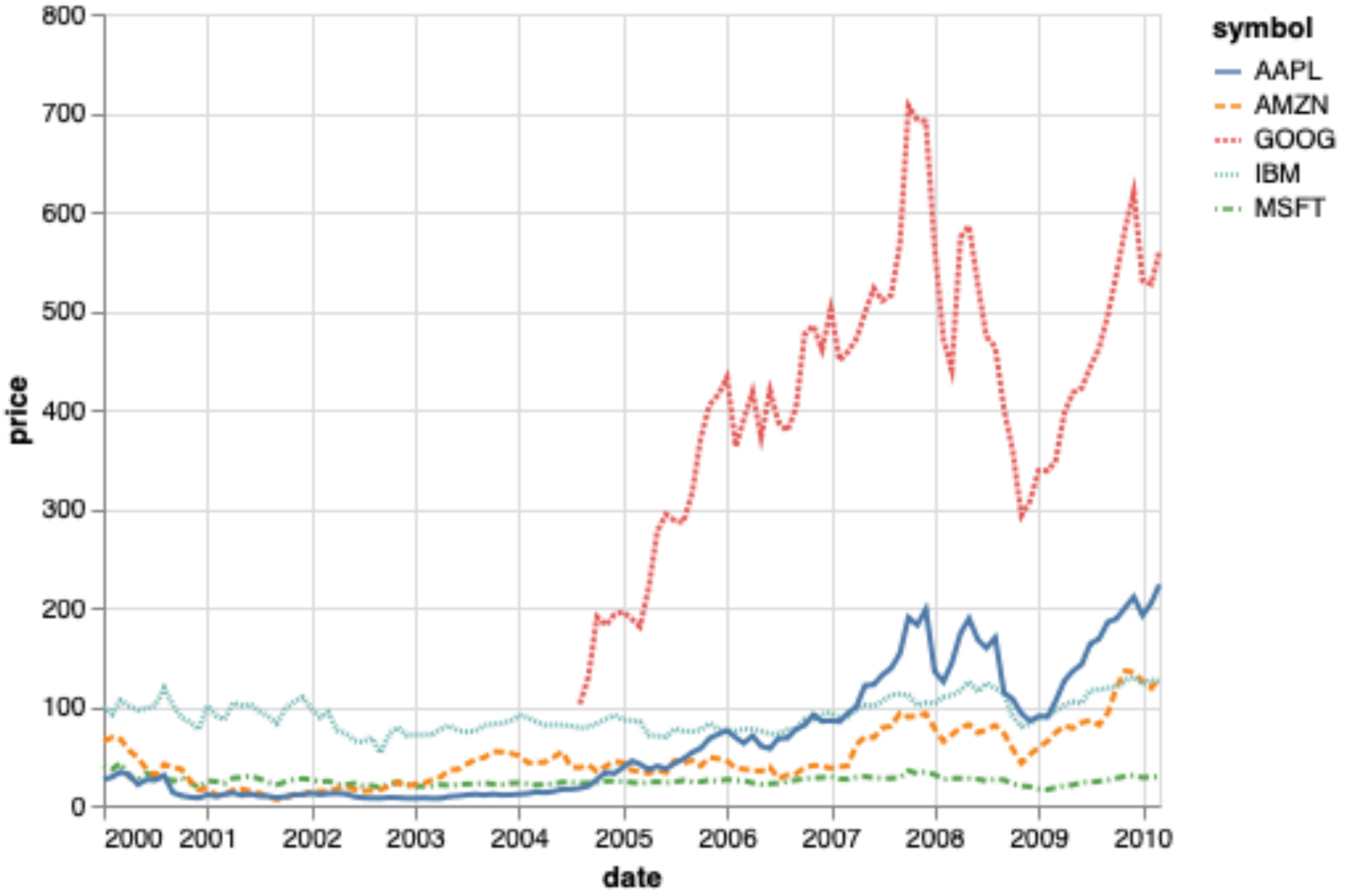
- Spatial region
- Color hue
- Motion
- Shape



Altair will use its rules to pick whether to use color hue or saturation based on the type

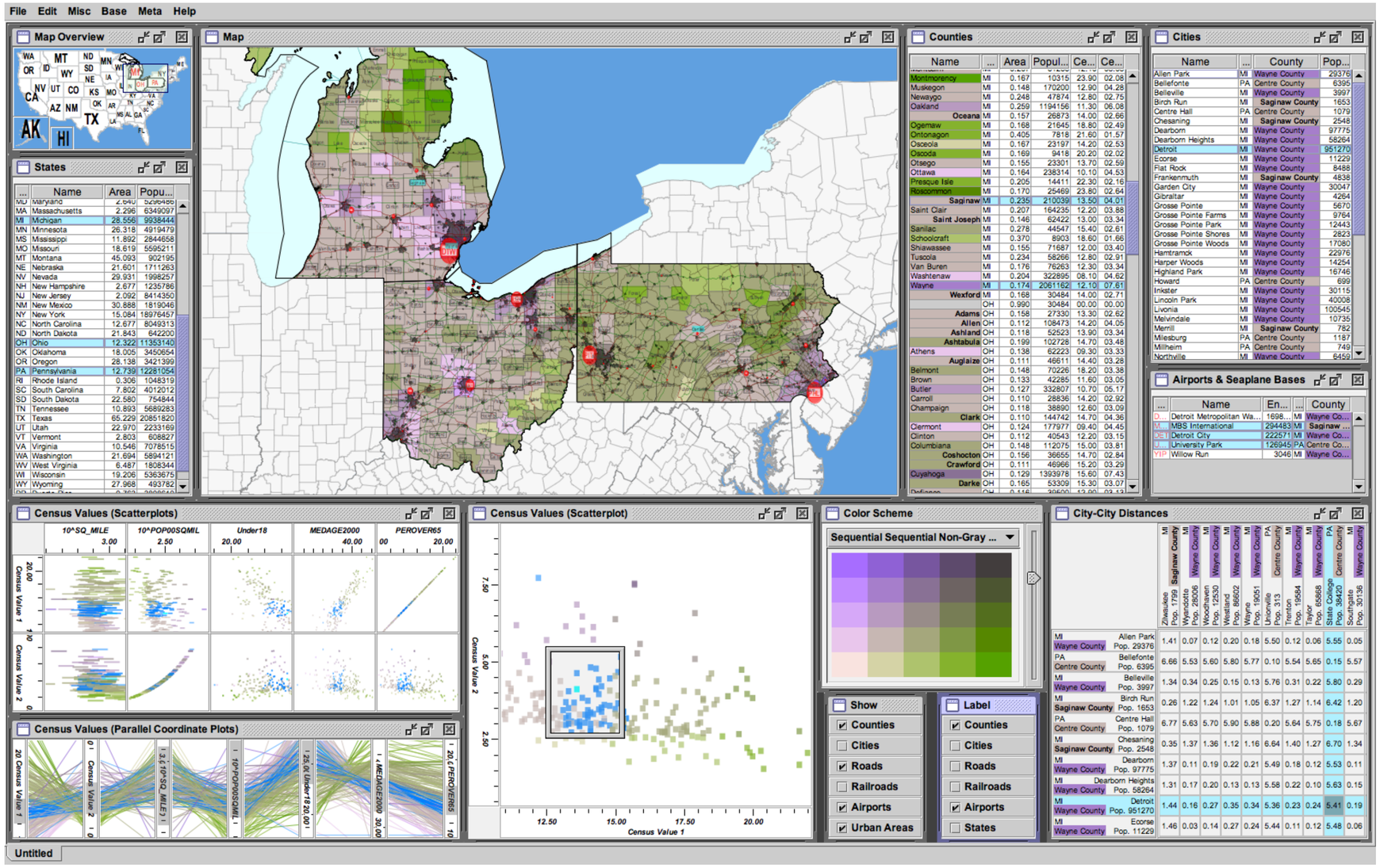
[Munzner (ill. Maguire), 2014]

Multiple Views in Visualization



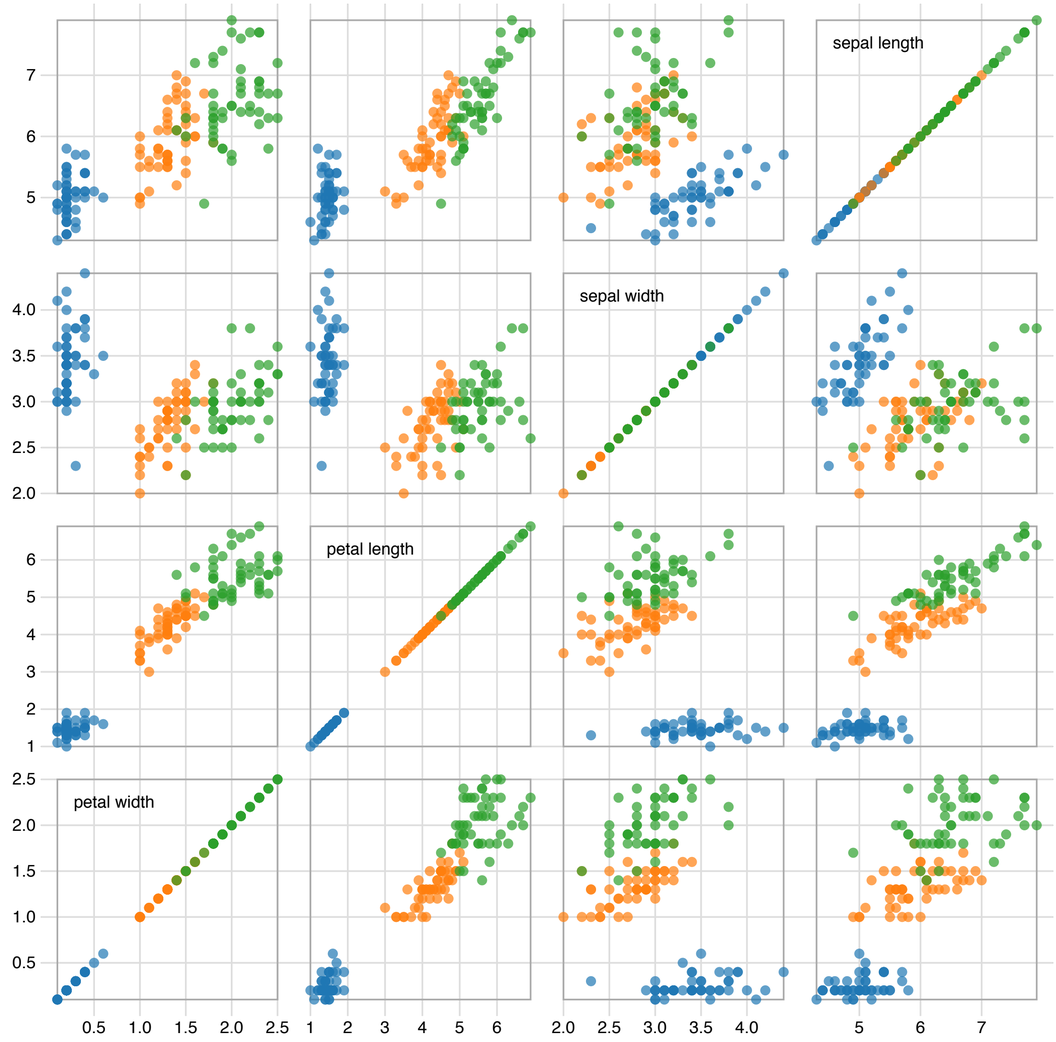
[Altair]

Multiple Views in Visualization



[Improvise, Weaver, 2004]

Multiple Views in Visualization

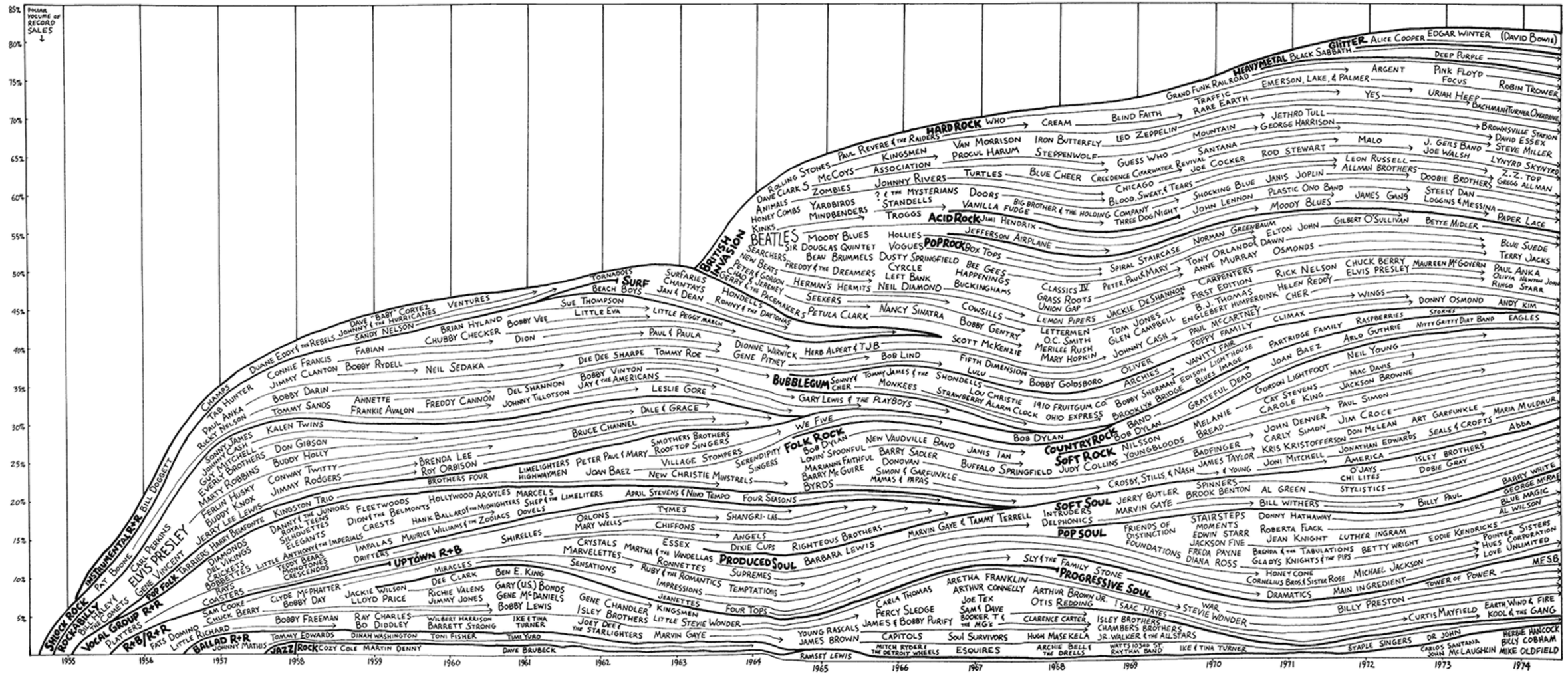


[M. Bostock]

Altair Supports Concatenation, Layering, & Repetition

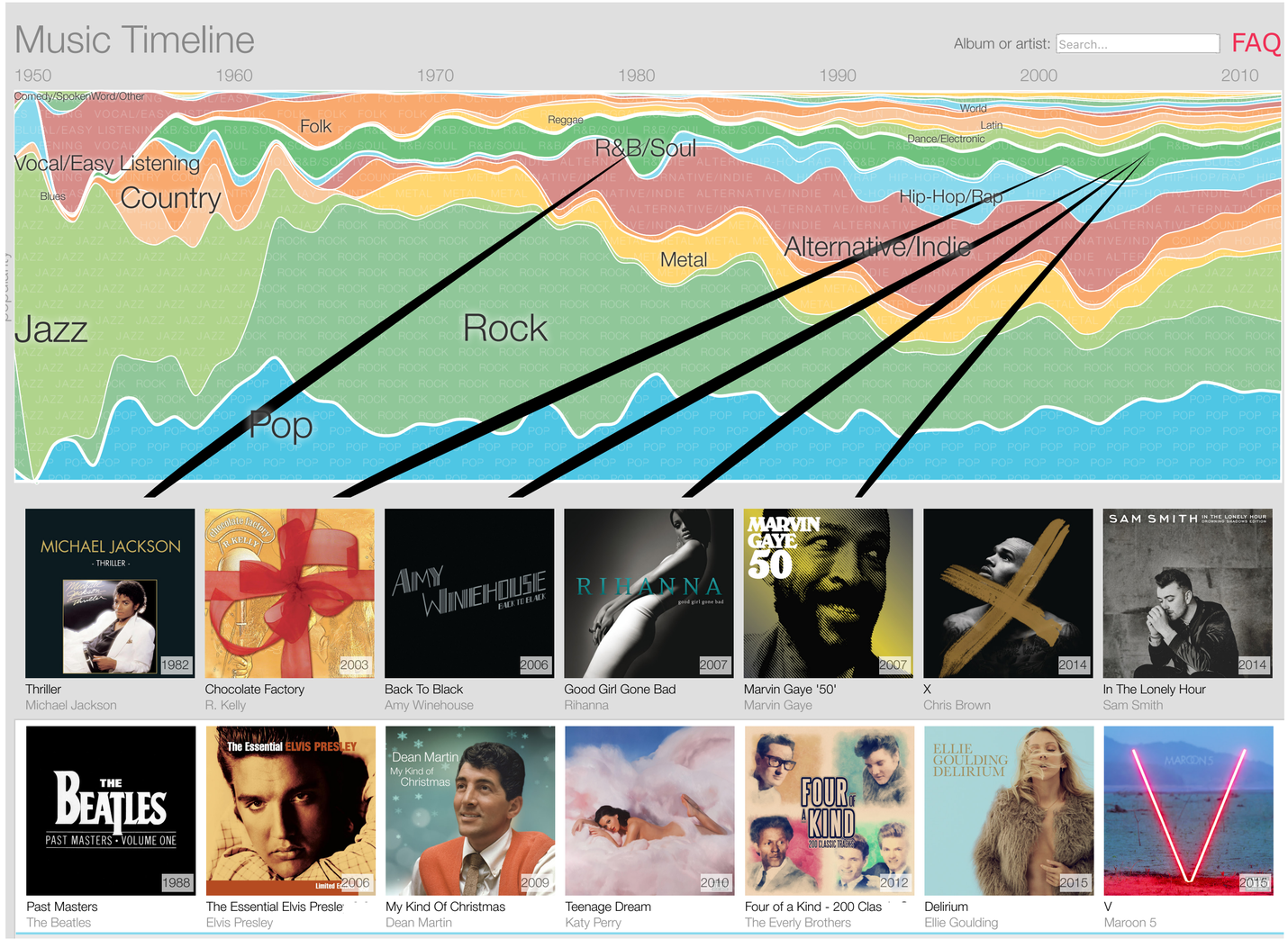
- Layering:
  - + Operator
- Concatenation:
  - Horizontal: | operator
  - Vertical: & operator
- Repetition
  - Use of .repeat for layout
  - Reference repeated variables in the encoding

Visualization



[Rock 'N' Roll is Here to Pay, R. Garofalo, 1977 (via Tufte)]

Also Visualization, but with Interaction



[Music Timeline, Google Research]

Interaction

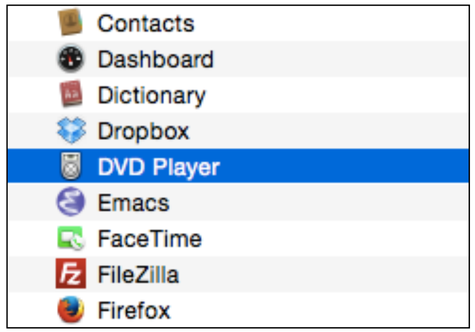
- Grammar of Graphics, why not Grammar of Interaction?
- Vega-Lite/Altair is about **interactive** graphics
- Types of Interactions:
  - Selection
  - Zoom
  - Brushing

Selection

- Selection is often used to initiate other changes
- User needs to select something to drive the next change
- What can be a selection target?
  - Items, links, attributes, (views)
- How?
  - mouse click, mouse hover, touch
  - keyboard modifiers, right/left mouse click, force
- Selection modes:
  - Single, multiple
  - Contiguous?

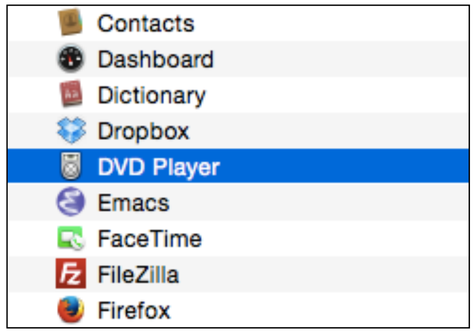
Highlighting

- Selection is the user action
- Feedback is important!
- How? Change selected item's visual encoding
  - Change color: want to achieve visual popout
  - Add outline mark: allows original color to be preserved
  - Change size (line width)
  - Add motion: marching ants



Highlighting

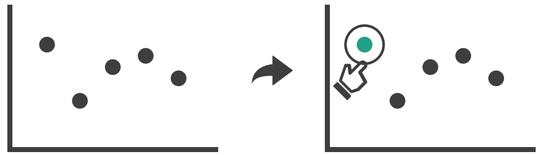
- Selection is the user action
- Feedback is important!
- How? Change selected item's visual encoding
  - Change color: want to achieve visual popout
  - Add outline mark: allows original color to be preserved
  - Change size (line width)
  - Add motion: marching ants



Interaction Overview

Change over Time

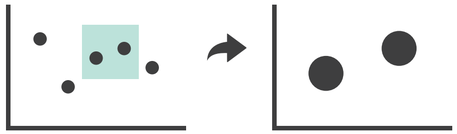
Select



Navigate

Item Reduction

- *Zoom*
  Geometric or *Semantic*

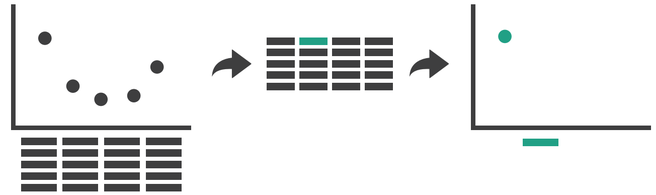


- *Pan/Translate*

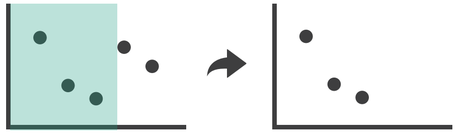
- *Constrained*

Attribute Reduction

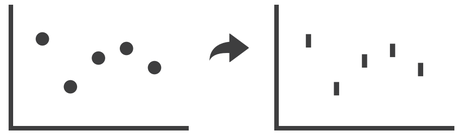
- *Slice*



- *Cut*



- *Project*



[Munzner (ill. Maguire), 2014]

Altair's Interactive Charts

- https://altair-viz.github.io/gallery/index.html#interactive-charts