Group: DSAA 5024 2024 FALL

Valid until 9/30 and will update upon joining group

Data Exploration & Visualization

Module 2

Data Transformation & Mapping

Dr. ZENG Wei

DSAA 5024

The Hong Kong University of Science and Technology (Guangzhou)

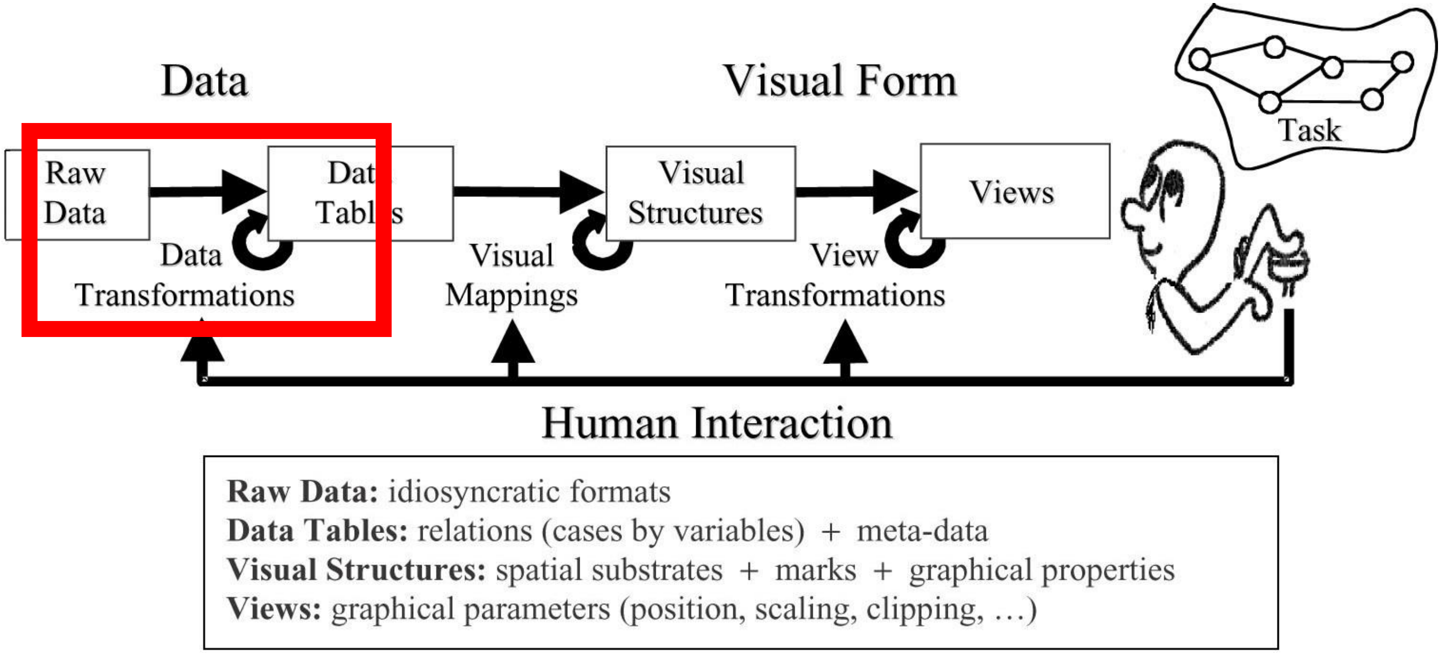
Data Exploration & Visualization

Module 2: Data Transformation & Mapping

- Data transformation
  - Aggregation, sampling, discretization, attribute transformation, arrangement, selection, feature creation
- Mapping data to visuals
  - Numbers to positions

Visualization process

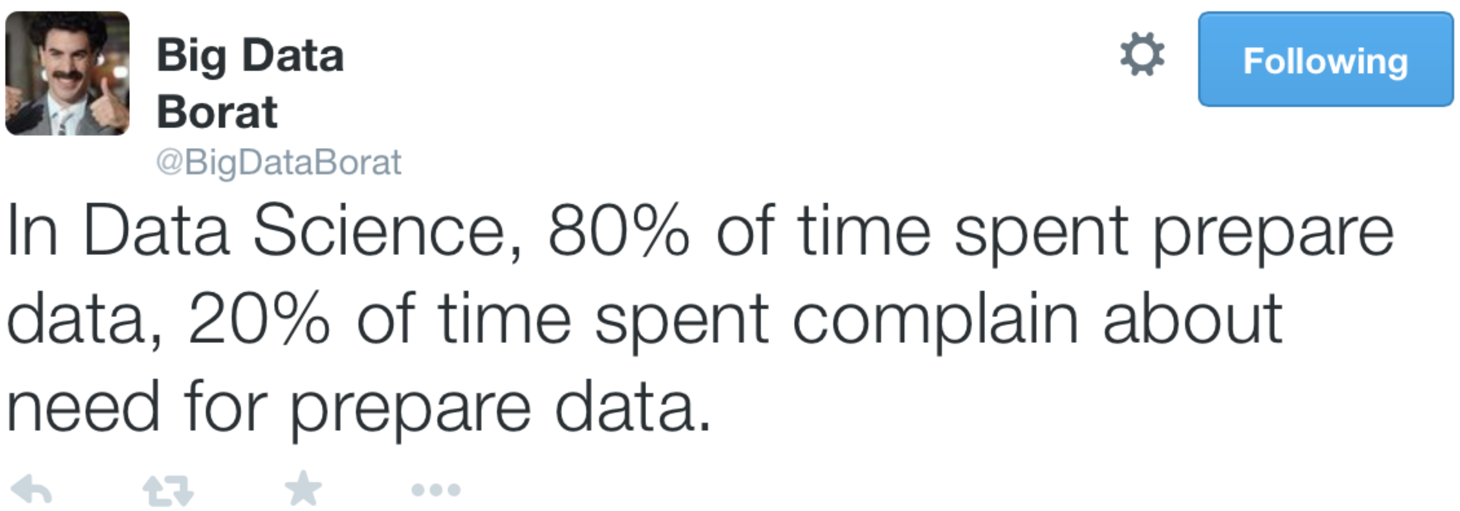
- Information visualization reference model
  - Data transformations
  - Visual mappings
  - View transformations



Credit: Stuart, Mackinlay, Shneiderman, 1999

Data processing

- The process of processing raw data to facilitate subsequent analysis
  - Data cleaning
    - Noise, missing or corrupted values



Data cleaning

- Data quality problems
  - Noise: modification of original values
  - Outliers: data objects with characteristics that are considerably different than most of the other data objects in the data set
  - Missing values: iinformation is not collected or Attributes may not be applicable to all cases
  - Dsuplicate values: data objects that are duplicates, or almost duplicates of one another
- What can we do about these problems?

Data transformation

- Why data transformation?
  - Raw data are too big and complex.
    - Too heavy for computation or storage
    - Data cube
  - Algorithm/model does not work for the raw format / works better for the format.
    - Graph vs. matrix representation
  - A visualization perspective: the display space is limited.
    - Displaying all the data will cause cluttering
    - Paper reading: A taxonomy of clutter reduction for information visualization

Data transformation

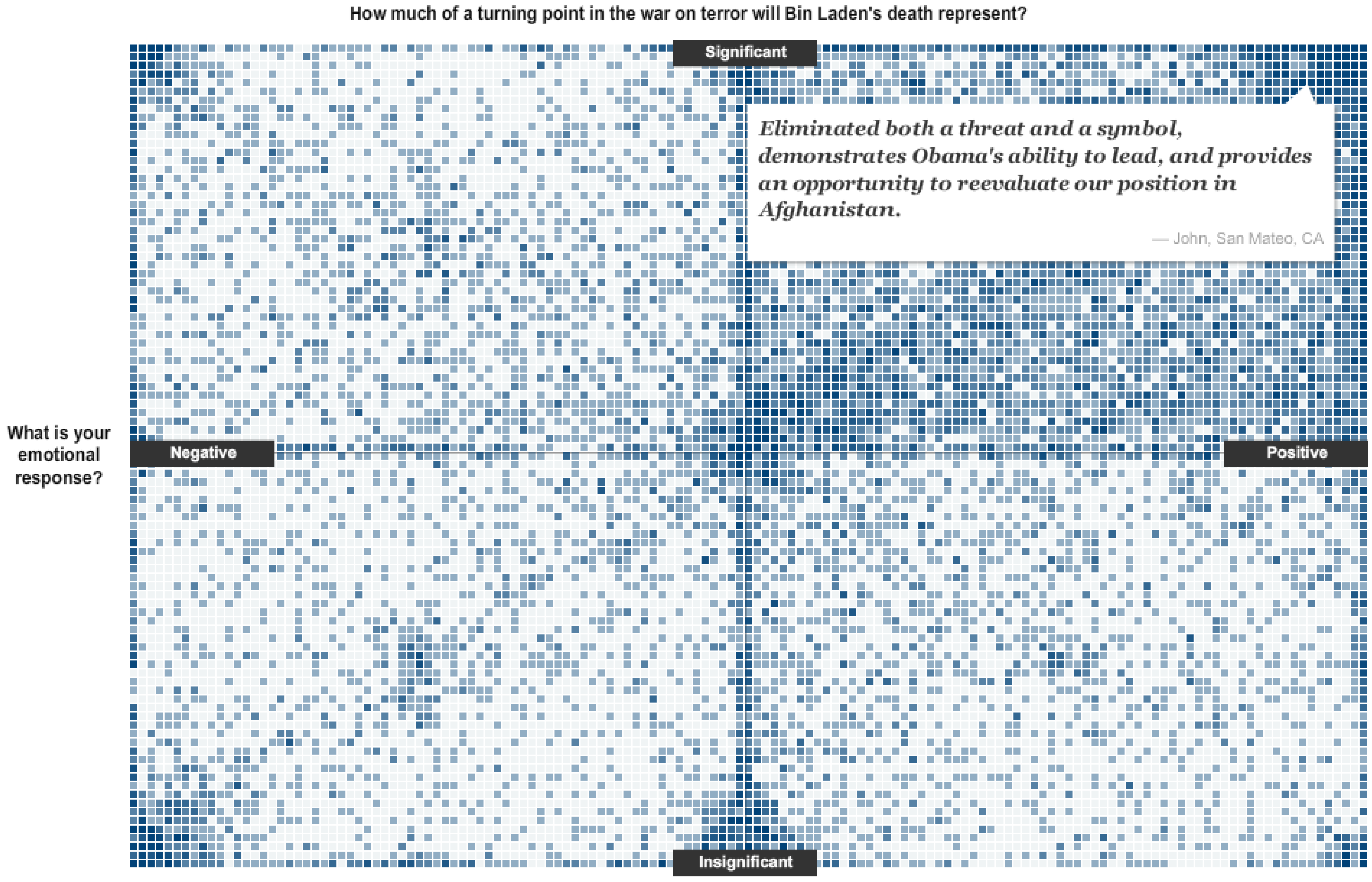
- Common methods
  - Aggregation
  - Sampling
  - Discretization
  - Attribute transformation
  - Arrangement
  - Selection
  - Feature creation

Aggregation

- **Aggregation** combines two or more attributes (or records) to a single attribute (or record)
  - Data reduction
    - Reduce the number of attributes or records
  - Change of scale
    - Cities aggregated into regions, states, countries, etc.
  - More ‘stable’ data
    - Aggregated data tend to have less variability

Aggregation

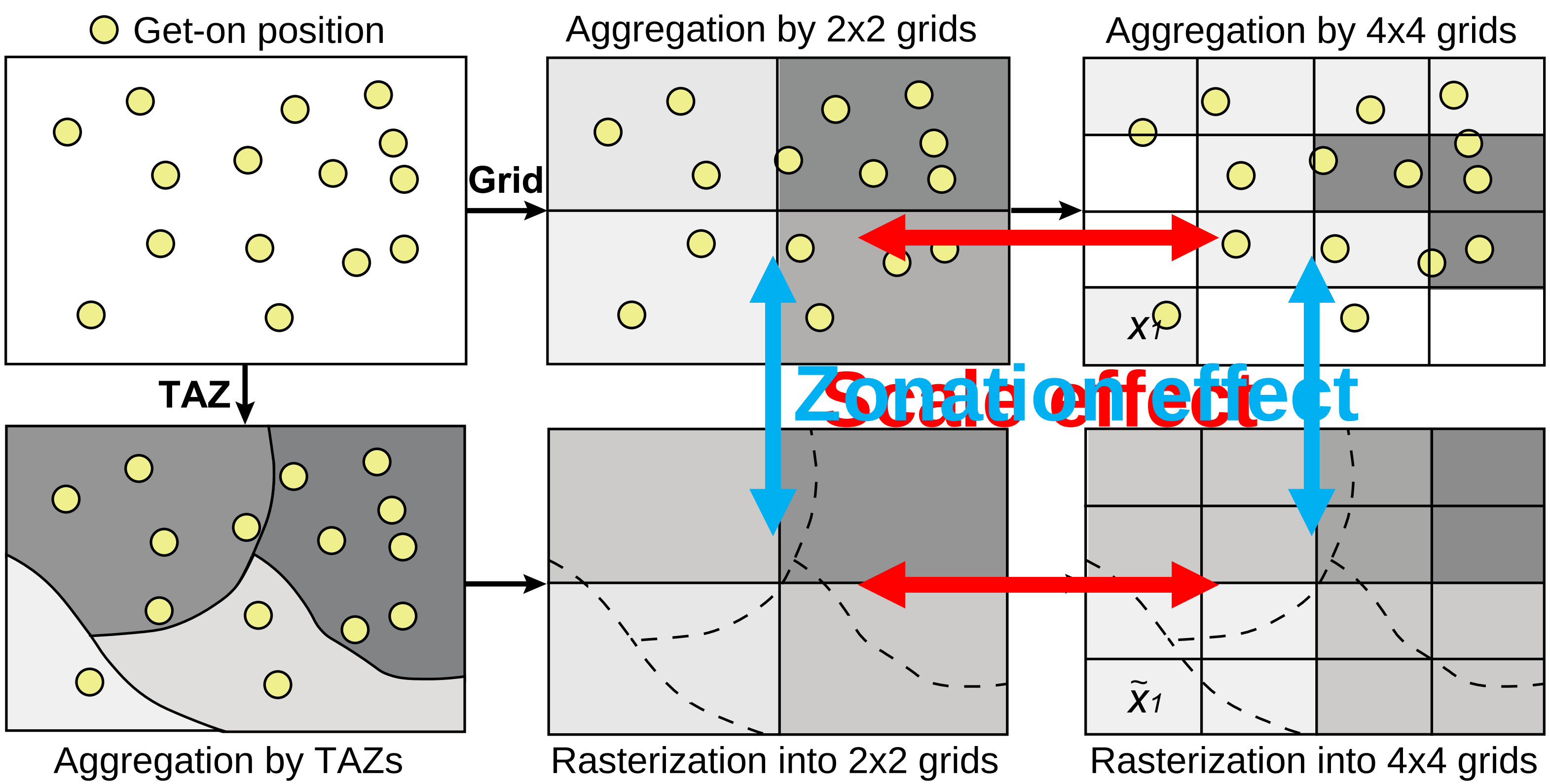
- Aggregating user clicks into the number of clicks.



'The Death of a Terrorist: A Turning Point?' - NYTimes, 2011

Why aggregation matters? MAUP as an example

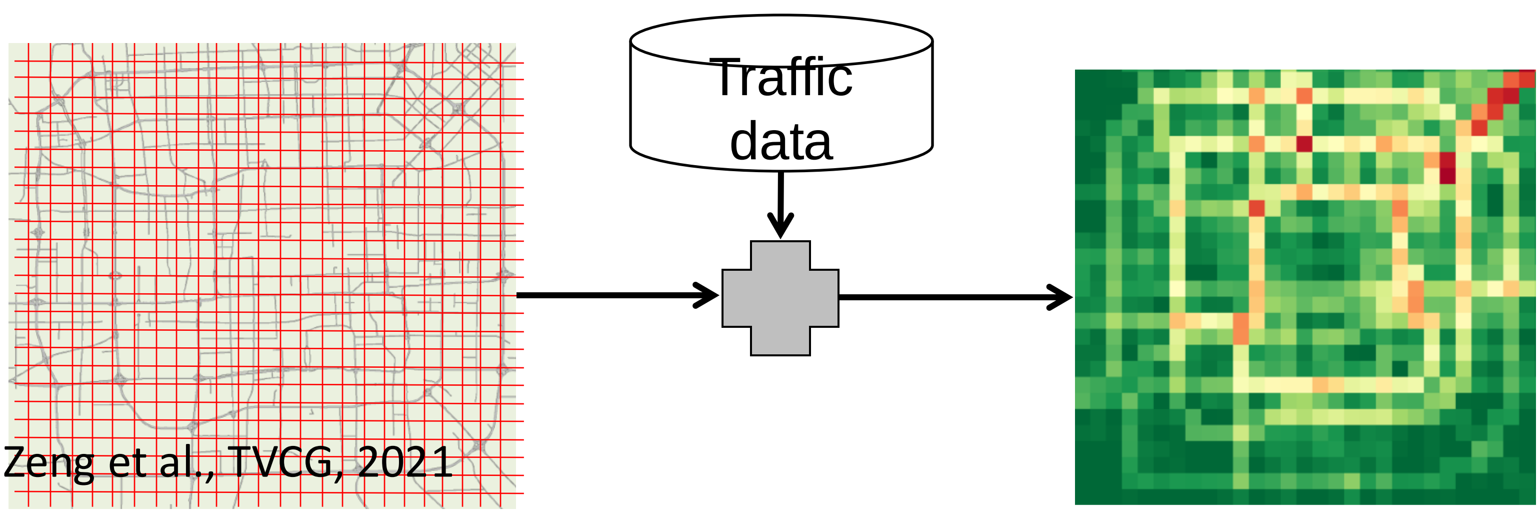
- The same basic data yield different results when aggregated in different ways.



Zeng et al., TVCG, 2021

Why aggregation matters? MAUP as an example

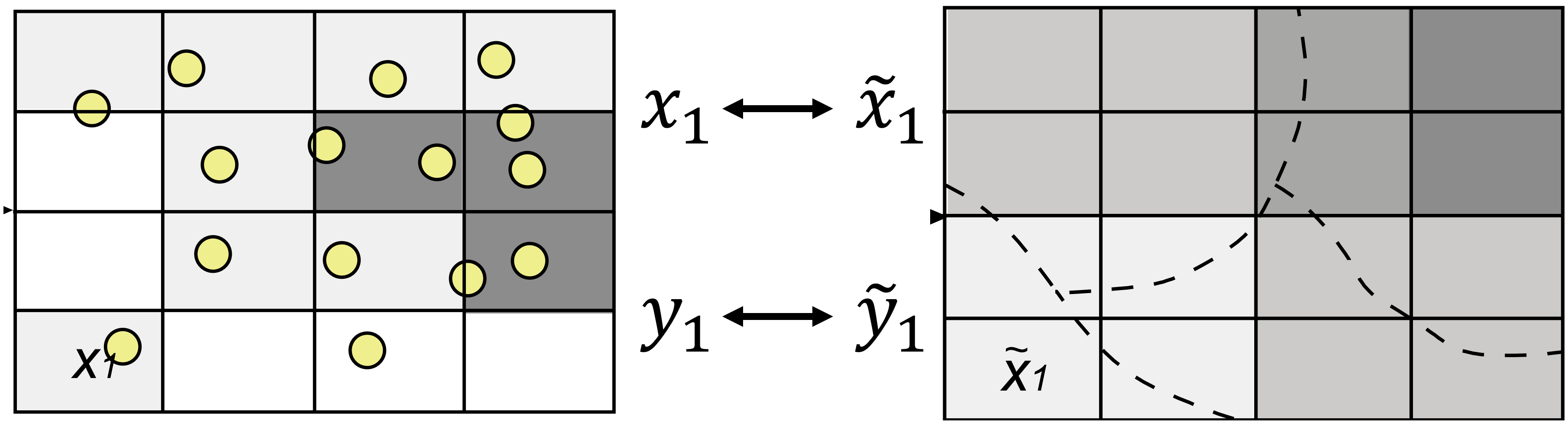
- Affects many types of analysis including correlation, regression, and prediction.
- ST-ResNet [Zhang et al., 2017]
  1. partition an underlying territory into grids
  2. aggregate in- and out-flows in each grid
  3. model as a sequence of raster images
  4. apply convolution-based residual neural network



Zeng et al., TVCG, 2021

Why aggregation matters? MAUP as an example

- Adversarial perturbation problem of DNNs.
  - The input difference may be marginal, but it may cause significant effects on the outputs.



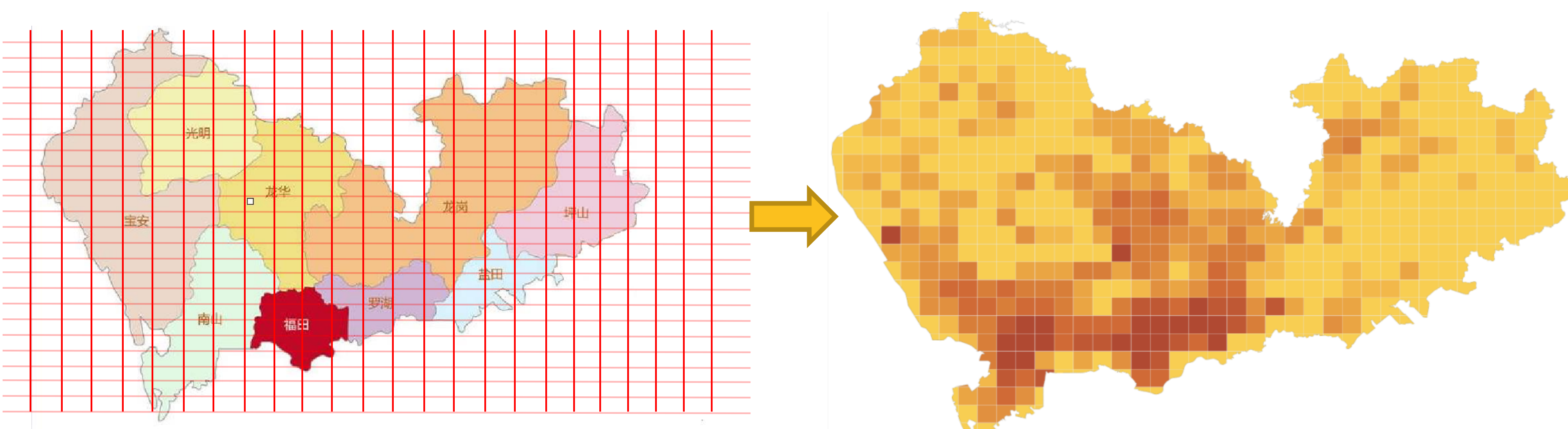
Aggregation by 4x4 grids

Aggregation by TAZs

Zeng et al., TVCG, 2021

Why aggregation matters? MAUP as an example

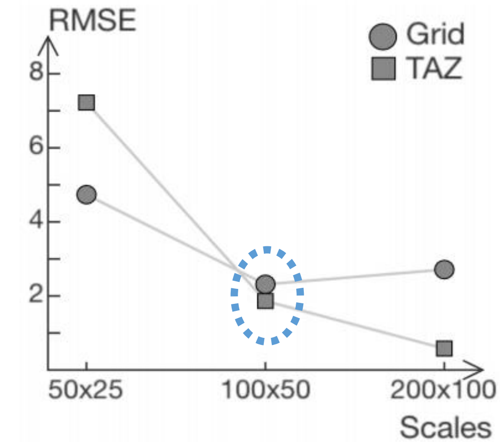
- Data and data transformation
  - Taxi transaction records in Shenzhen from 01 Jan. 2019 to 28 Feb. 2019 (59 days)
  - Traffic analysis zones (TAZs): 1066
  - Shapes: by grids vs. by TAZs followed by rasterization
  - Scales: 50x25, 100x50, 200x100

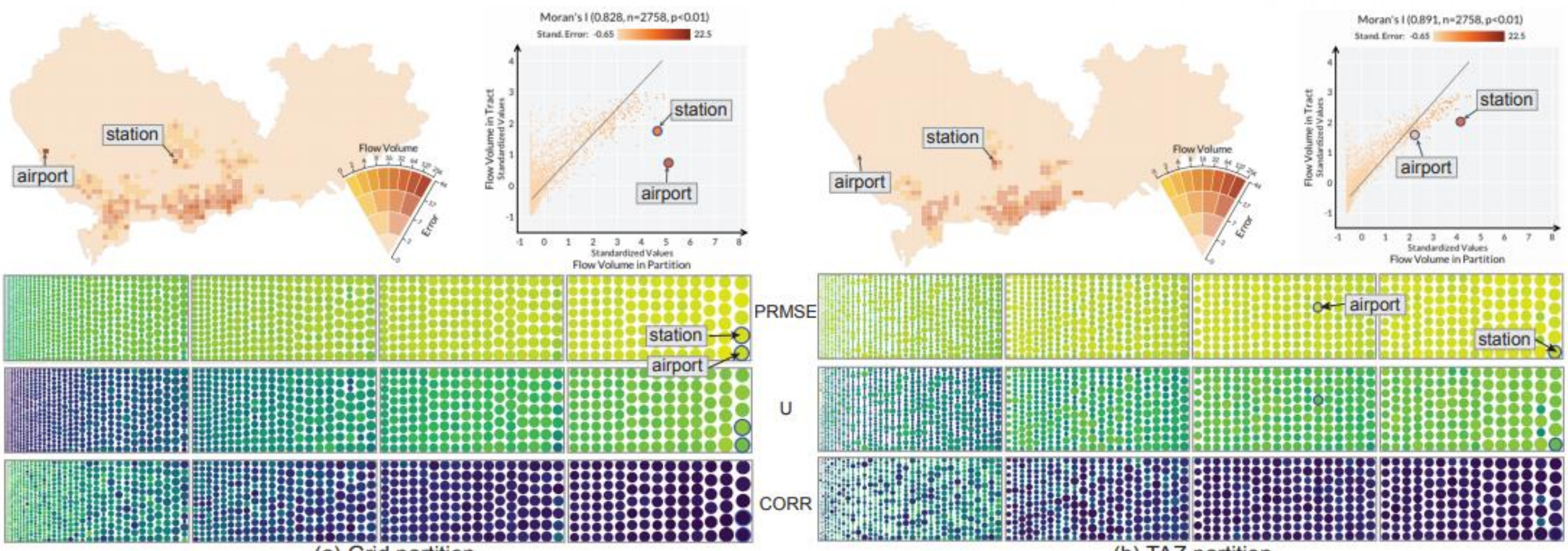


Zeng et al., TVCG, 2021

Why aggregation matters? MAUP as an example

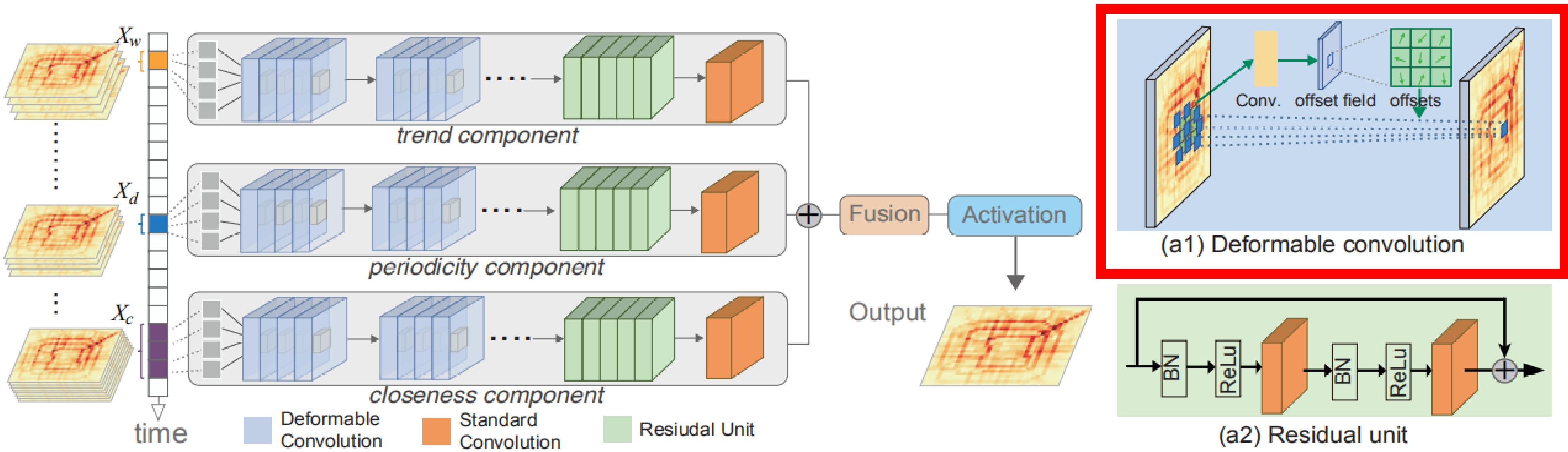
- RMSEs are similar for grid and TAZ partitions at scale 100x50.
- Scale-independent metrics show TAZ partition achieves better performance.
  - especially for the airport region





Why aggregation matters? MAUP as an example

- Lessons learnt: Spatial nonstationarity affects prediction accuracy.
- Solution: Modeling spatial nonstationarity via deformable convolutions.
- Deformable convolutions + RoI partition helps.



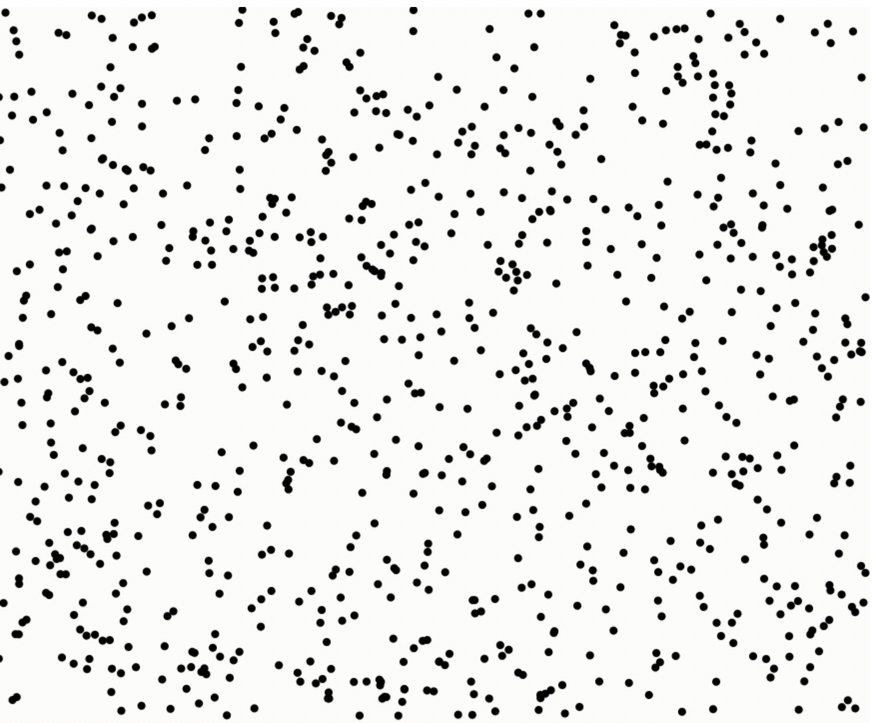
Zeng et al., TKDE, 2021

Sampling

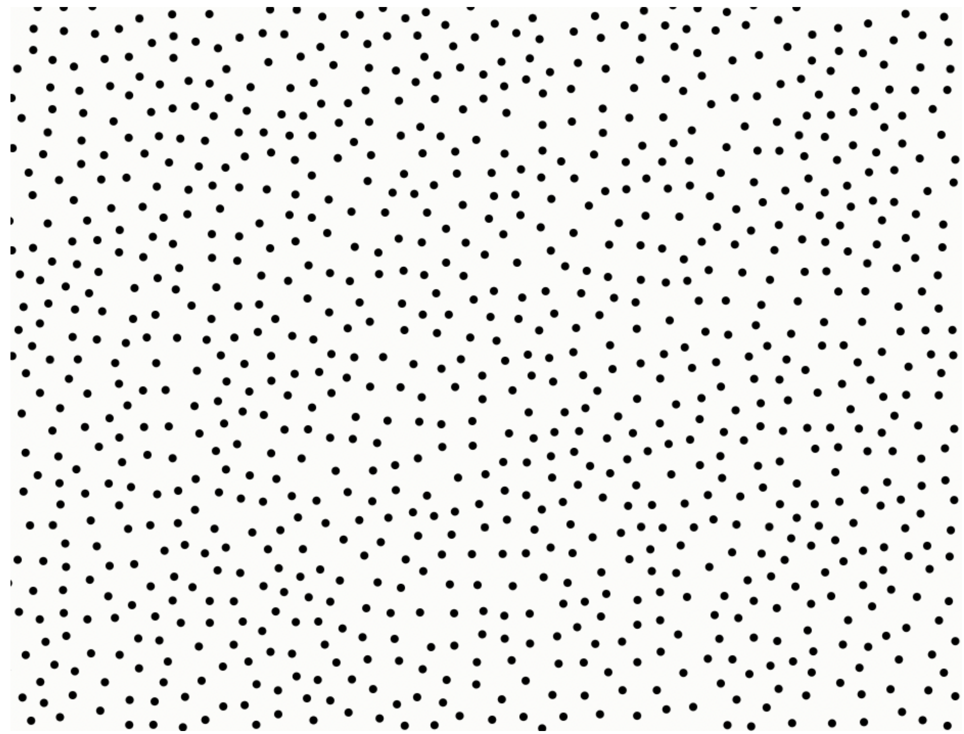
- **Sampling** is the main technique employed for data selection.
  - **Statisticians** sample because **obtaining** the entire set of data of interest is too expensive or time consuming.
  - Sampling is used in **data mining** because **processing** the entire set of data of interest is too expensive or time consuming.

Sampling

- Good sampling selects **representative** samples
  - Has approximately the same property (of interest) as the original set of data



Uniform random sampling

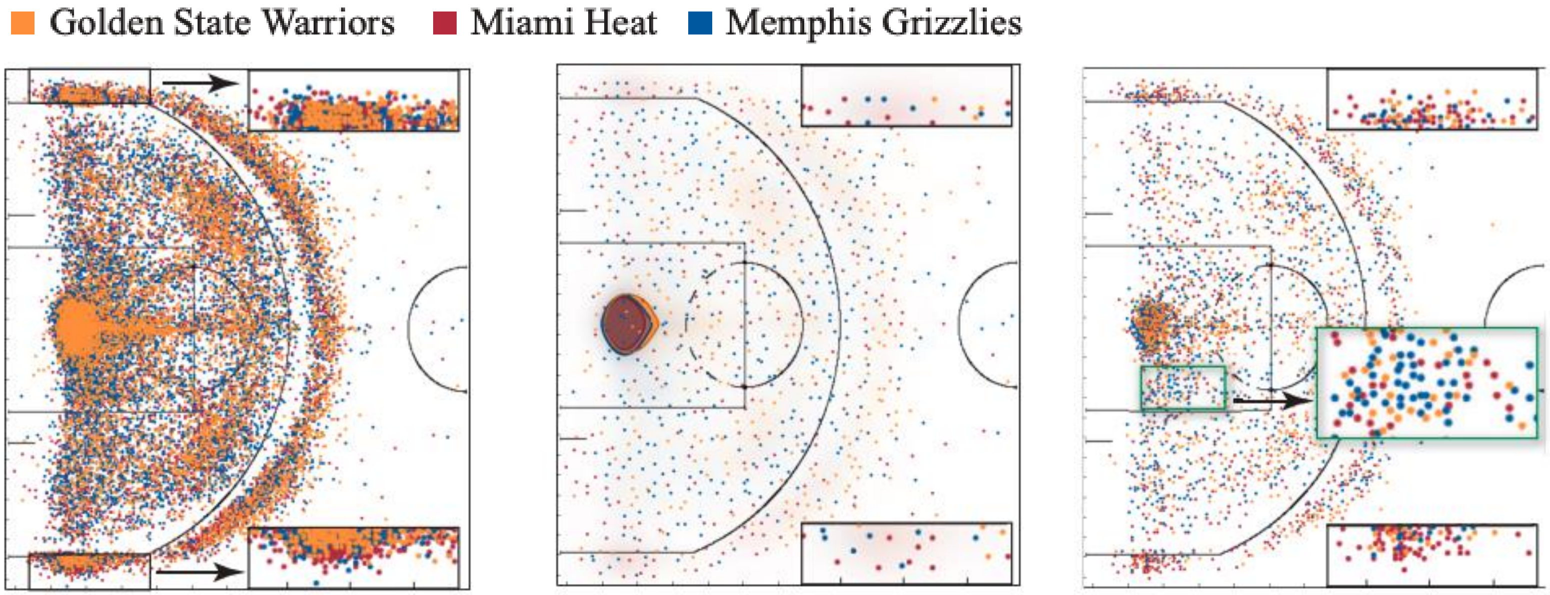


Poisson-disc sampling

Visualizing algorithms – Mike Bostock

Sampling

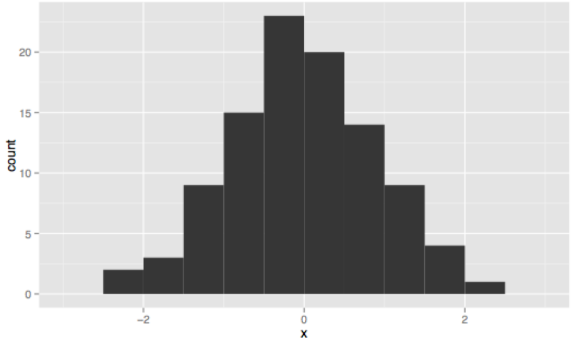
- Sampling for multi-class scatterplot
  - faithfully presenting relative data
  - faithfully presenting class densities
  - Preserving major outliers



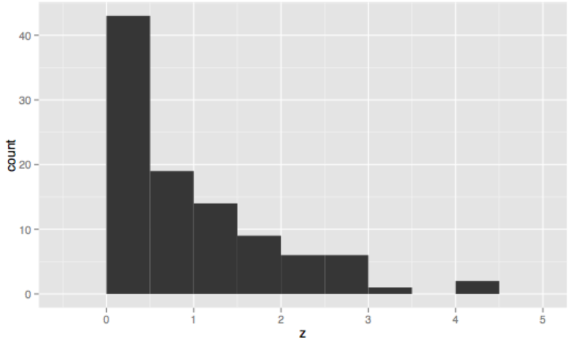
Splatterplots 2013 Chen et al. 2014

Discretization

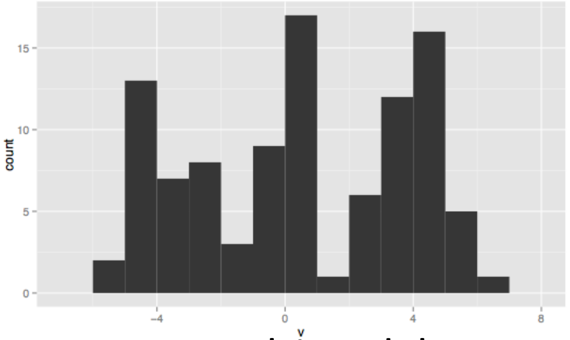
- **Discretization** divides the range of continuous variables into a finite set of intervals.
  - Usually a first step for numerical evaluation and implementation on digital computers.
- **Histogram** is a common technique for discretization.
  - Divide the values into *bins* and show a bar plot of the number of objects in each bin
  - The height of each bar indicates the number of objects



Symmetric, unimodal



Skewed

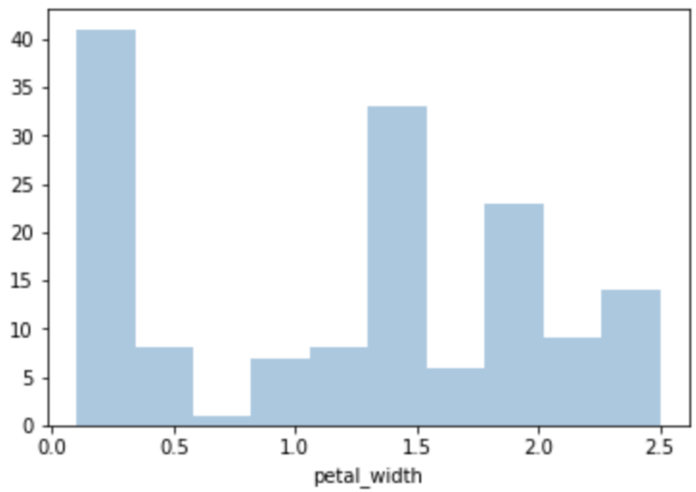


Multimodal

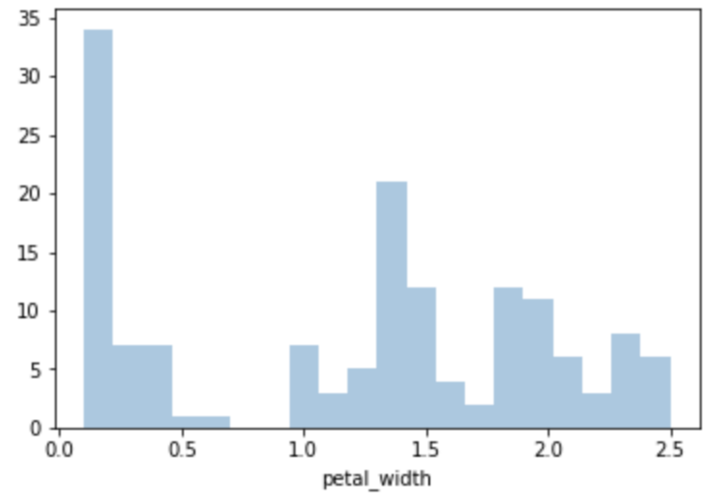
Discretization

- Shape of histogram depends on the number of bins

```python
sns.distplot(iris['petal_width'],\
                 bins=10, kde=False);
```



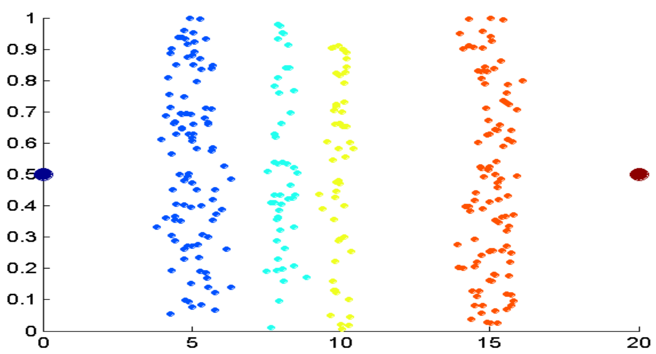

```python
sns.distplot(iris['petal_width'],\
                 bins=20, kde=False);
```



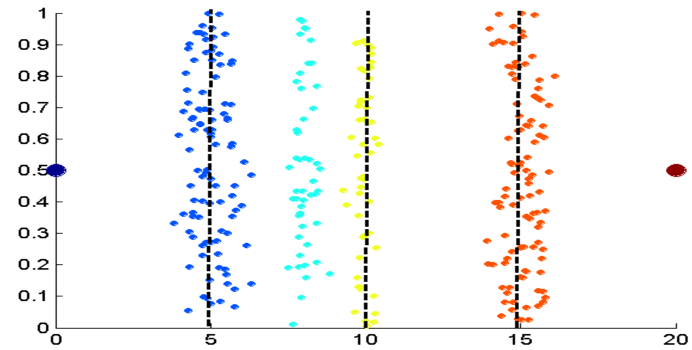
- Choose proper number of bins, not just for histogram?
  - Unsupervised: equal width, equal frequency
  - Supervised: entropy-based, binning

Discretization

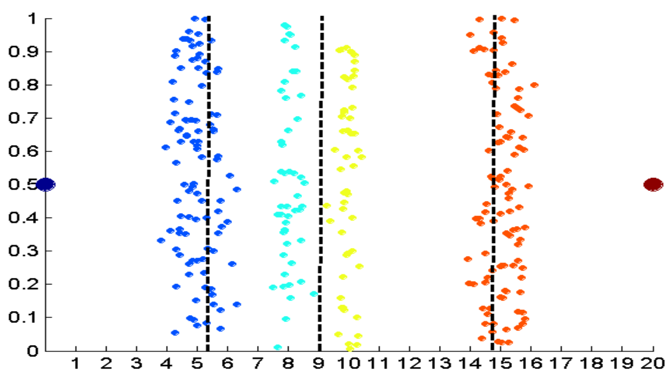
- Unsupervised approaches



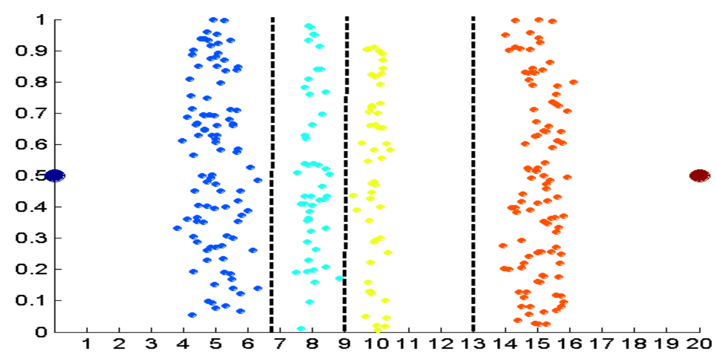
Data



Equal interval width



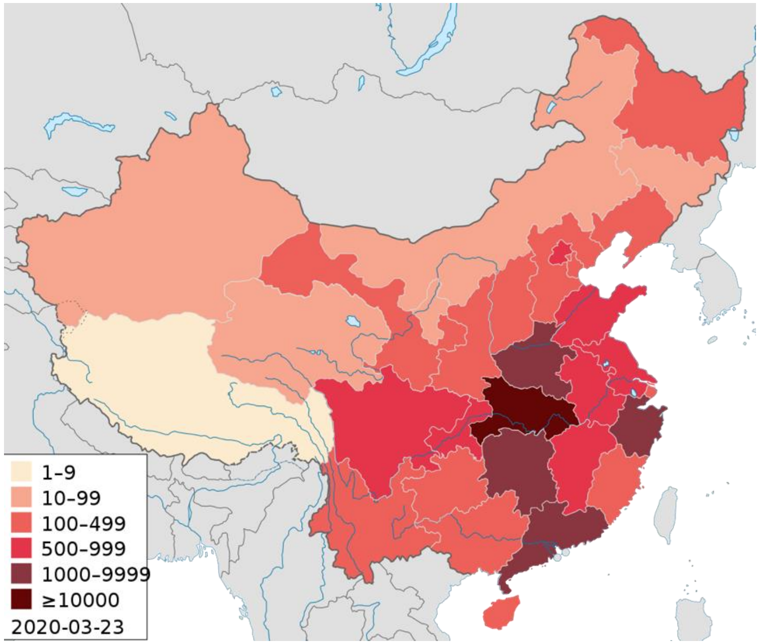
Equal frequency



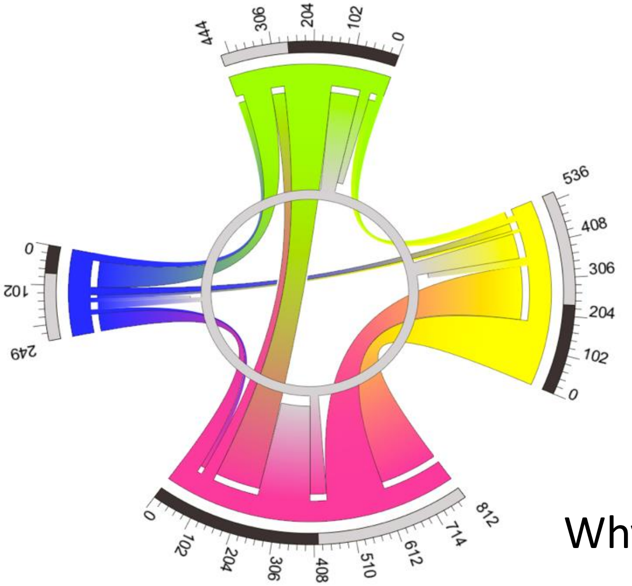
K-means

Discretization

- An important step for determining enumeration units in choropleth map.
  - Why Covid-19 cases (right) are divided into [1, 9], [1-99], [100, 999]…



COVID-19 cases in mainland China by provinces as of 7 March 2020



Why 102, not 100?

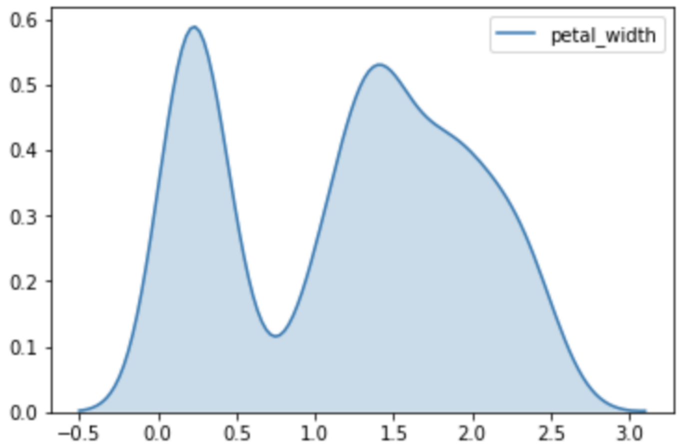
Kernel density estimation

- **Kernel density estimation** (KDE) estimates the probability density function of a random variable.

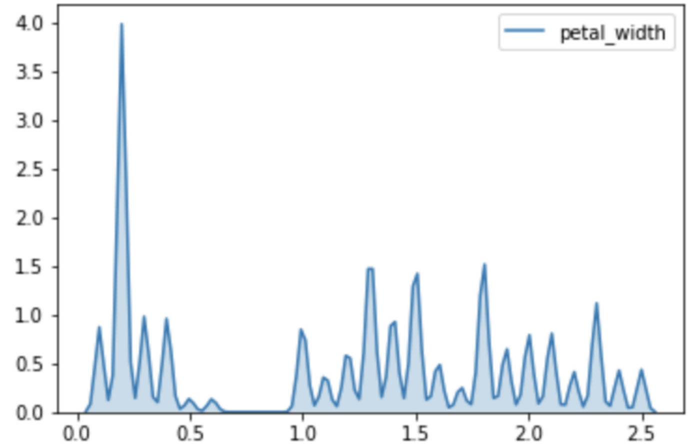
$$kde(x) = \frac{1}{mb}\sum_{i=1}^{m} K(\frac{x_i - x}{b})$$

where *K* is the chosen kernel (weight function), *b* is the bandwidth

```
sns.kdeplot(iris['petal_width'],\
            shade=True, bw=0.2);
```




```
sns.kdeplot(iris['petal_width'],\
            shade=True, bw=0.02);
```



https://github.com/mwaskom/seaborn/blob/master/seaborn/distributions.py

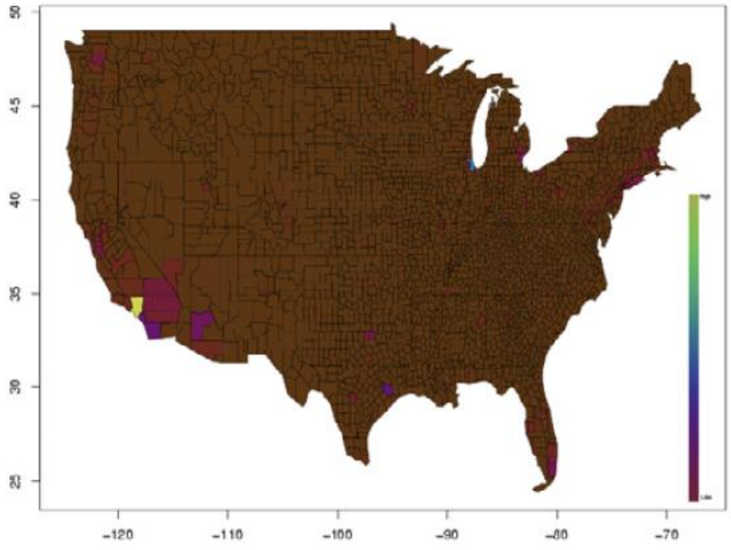
Attribute transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values
  - **Division by sum** $y_i \,/\, \sum_i y_i$
  - **Log** $\log y$
  - **Power** $y^{1/k}$
  - **Min-max** $\frac{(y_i - y_{min})}{(y_{max} - y_{min})}$
  - **z-score** $(y_i - \mu)/\sigma$

- In practice, the analyst does not know a priori which normalization function is best suited for a given dataset and he may test some preferred ones.

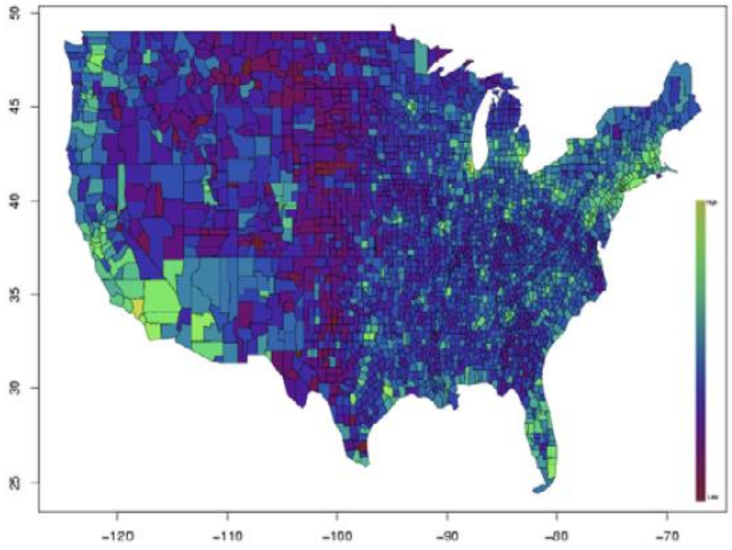
Attribute transformation

- **Normalization** changes the values of numeric columns in the dataset to a **common scale**, such that the visualization can better reveal patterns.



Linear colormap

$f: y \rightarrow c$



Logarithmic colormap

$f: y' \rightarrow c$, where y’ = log(y)

Schneidewind et al., 2006

Arrangement

- **Arrangement** is the placing of visual elements with a display
  - Arrangement can make a large difference in how easy it is to understand data
  - **Grouping** and **sorting** are common methods for data arrangement.

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	0	0	1
3	0	1	0	1	1	0
4	1	0	1	0	0	1
5	0	1	0	1	1	0
6	1	0	1	0	0	1
7	0	1	0	1	1	0
8	1	0	1	0	0	1
9	0	1	0	1	1	0

	6	1	3	2	5	4
4	1	1	1	0	0	0
2	1	1	1	0	0	0
6	1	1	1	0	0	0
8	1	1	1	0	0	0
5	0	0	0	1	1	1
3	0	0	0	1	1	1
9	0	0	0	1	1	1
1	0	0	0	1	1	1
7	0	0	0	1	1	1

Selection

- **Selection** is the elimination or the de-emphasis of certain objects / attributes
  - Choosing of a subset of *attributes,* aka *feature selection*
    - Feature selection is to remove redundant or irrelevant features
    - Dimensionality reduction is often used to reduce the number of dimensions to two or three

  - Choosing a subset of *objects*
    - You can only show so many points on the screen
    - Sampling – how to preserve points in sparse areas?

Feature creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes
- Three general methodologies:
  - Feature Extraction
    - domain-specific
  - Mapping Data to New Space
    - Embeddings
  - Feature Construction
    - combining features

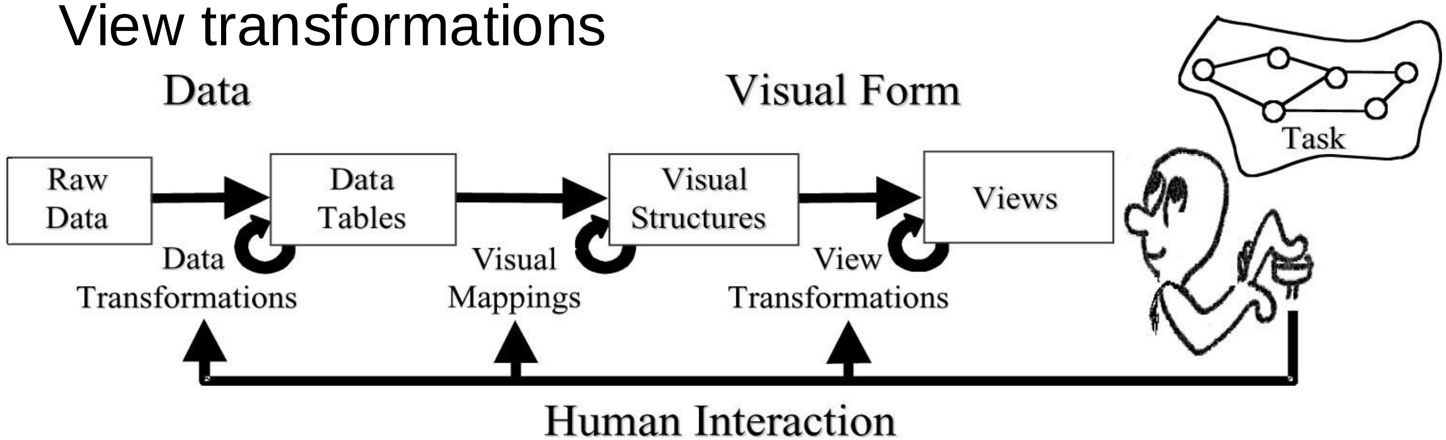
Data Exploration & Visualization

Module 2: Data Transformation & Mapping

- Data transformation
  - Aggregation, sampling, discretization, attribute transformation, arrangement, selection, feature creation
- Mapping data to visuals
  - Numbers to positions

Reference model

- Information visualization reference model
  - Data transformations
  - Visual mappings **: mapping the ‘features’ of a data set to the ‘features’ of visual perception.**
  - View transformations

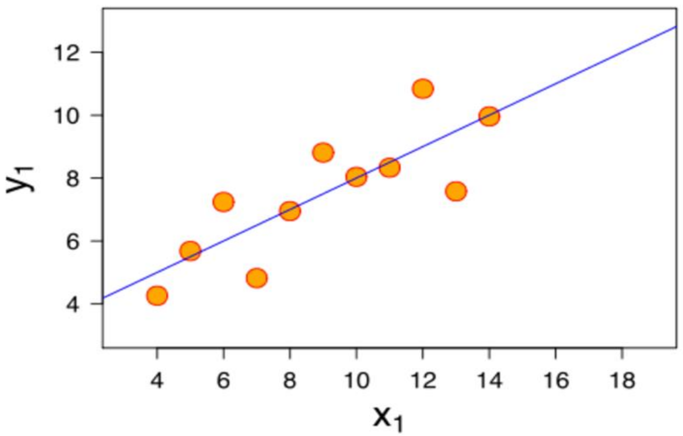


Raw Data: idiosyncratic formats
Data Tables: relations (cases by variables) + meta-data
Visual Structures: spatial substrates + marks + graphical properties
Views: graphical parameters (position, scaling, clipping, …)

Card, Stuart K., Jock D. Mackinlay, and Ben Shneiderman, eds. Readings in information visualization: using vision to think. Morgan Kaufmann, 1999.

Mapping data to position

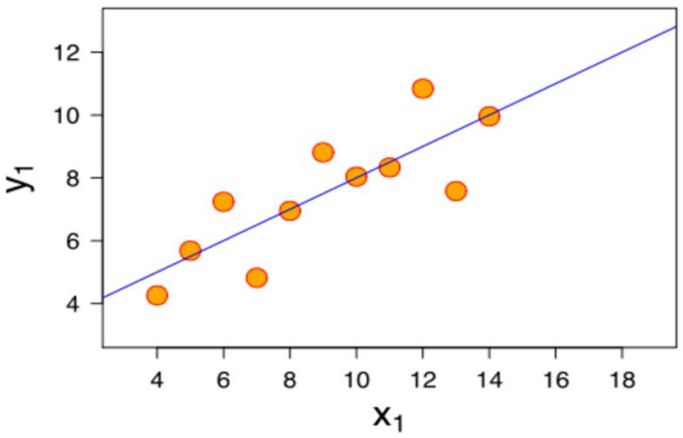
x	10.0	8.0	13.0	9.0	11.0
y	8.04	6.95	7.58	8.81	8.33
14.0	6.0	4.0	12.0	7.0	5.0
9.96	7.24	4.26	10.84	4.82	5.68



- Coordinate systems
  - Data coordinates: native coordinates of the data space
    - min, max, and average values of a given dataset
    - x: [4.0, 14.0], y: [4.26, 10.84]
  - View volume coordinates: the volume of the data space the user wants to view
    - can be defined by min & max, or a subset of the data, or origin & an extent
    - x: [2, 20], y: [2, 14]

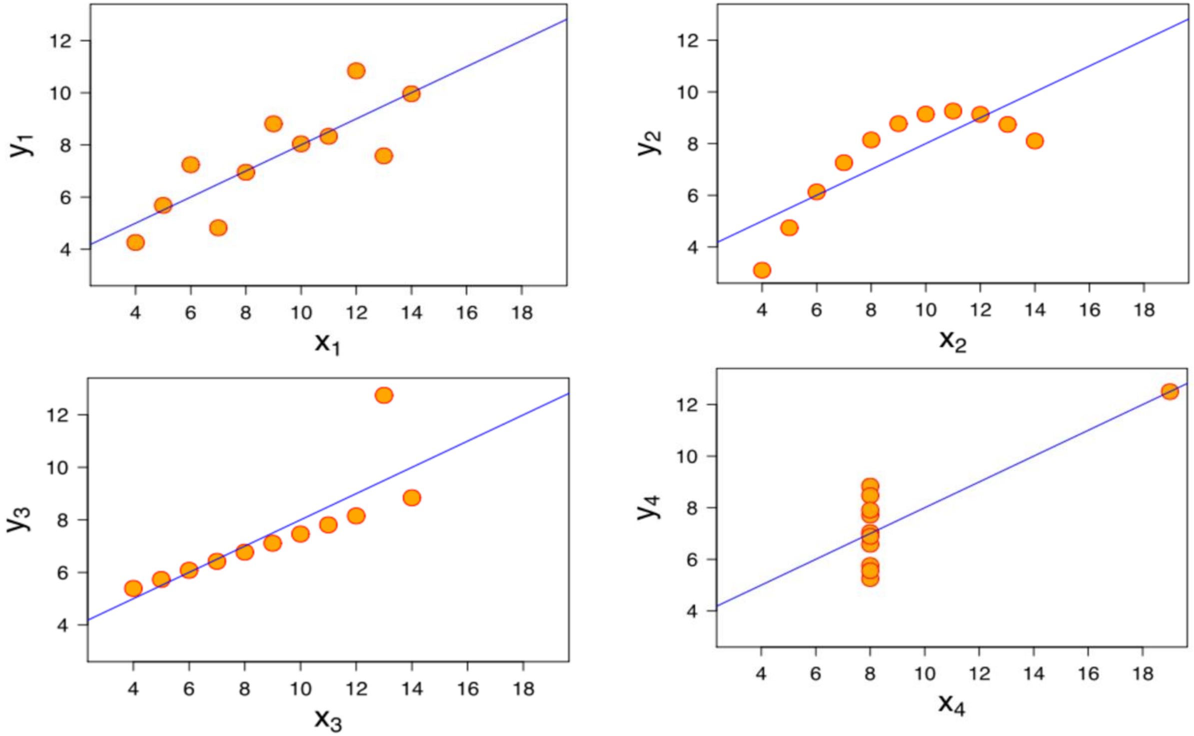
Mapping data to position

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



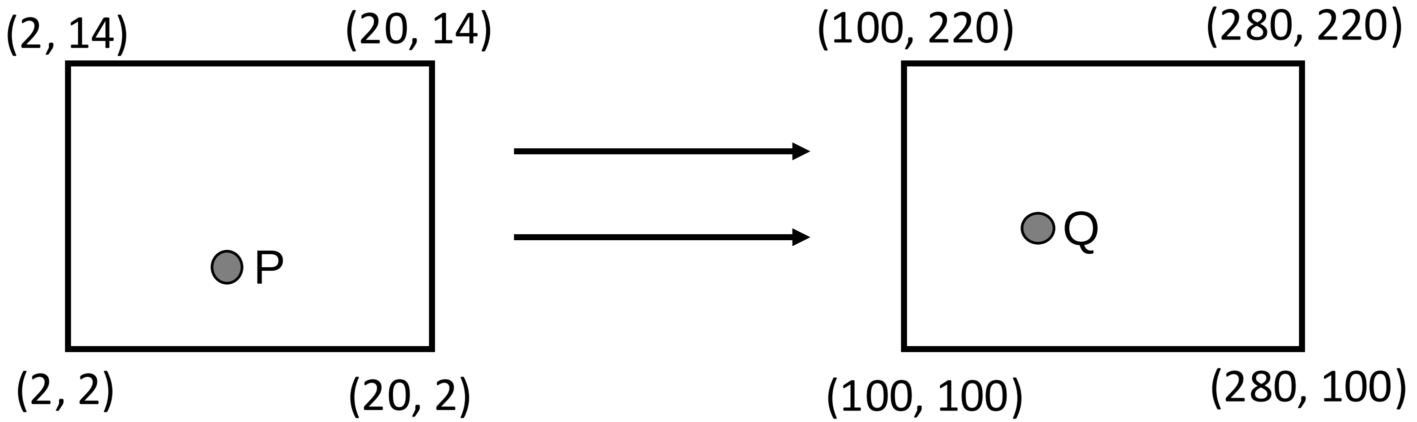
- Coordinate systems
  - Normalized view coordinates
    - A scaling of the data so that the data points within the view volume fit within the range [0, 1].
    - Linear scale: $f(x) = {(x - vol_{\min\_x})}/{(vol_{\max\_x} - vol_{\min\_x})}$, or Log scale
  - Screen coordinates
    - A scaling, possibly a translation, and a projection to convert the normalized view coordinates to viewport
    - 1080p: 1920x1080, 720p: 1280x720

Mapping data to position



- How to convert to screen coordinates? How can we arrange four plots as above?
  - **Geometric transformation**: a set of tools that aid in manipulating graphical objects and their coordinate systems

Affine transformation



- P(P_x, P_y) is transformed into Q(Q_x, Q_y) as follows:

$$Q_x = a\,P_x + c\,P_y + T_x$$
$$Q_y = b\,P_x + d\,P_y + T_y$$

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} P_x \\ P_y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\vec{Q} = \vec{M}\,\vec{P} + \vec{T}$$

2D primitive affine transformation

$$\vec{Q} = \vec{M}\,\vec{P} + \vec{T}$$

- Translation

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \text{ (Identity)}$$

$$T = \begin{bmatrix} T_x \\ T_y \end{bmatrix}$$

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} P_x + T_x \\ P_y + T_y \end{bmatrix}$$

2D primitive affine transformation

$$\vec{Q} = \vec{M}\,\vec{P} + \vec{T}$$

- Scale

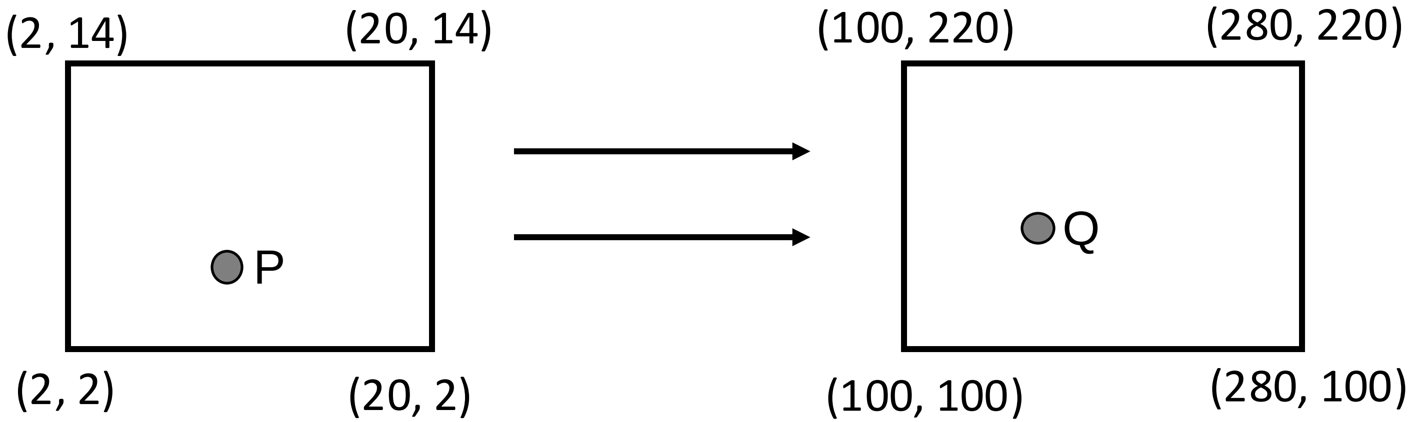
$$M = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}, T = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} Q_x \\ Q_y \end{bmatrix} = \begin{bmatrix} P_x S_x \\ P_y S_y \end{bmatrix}$$

$S_x = S_y$ (uniform scaling)

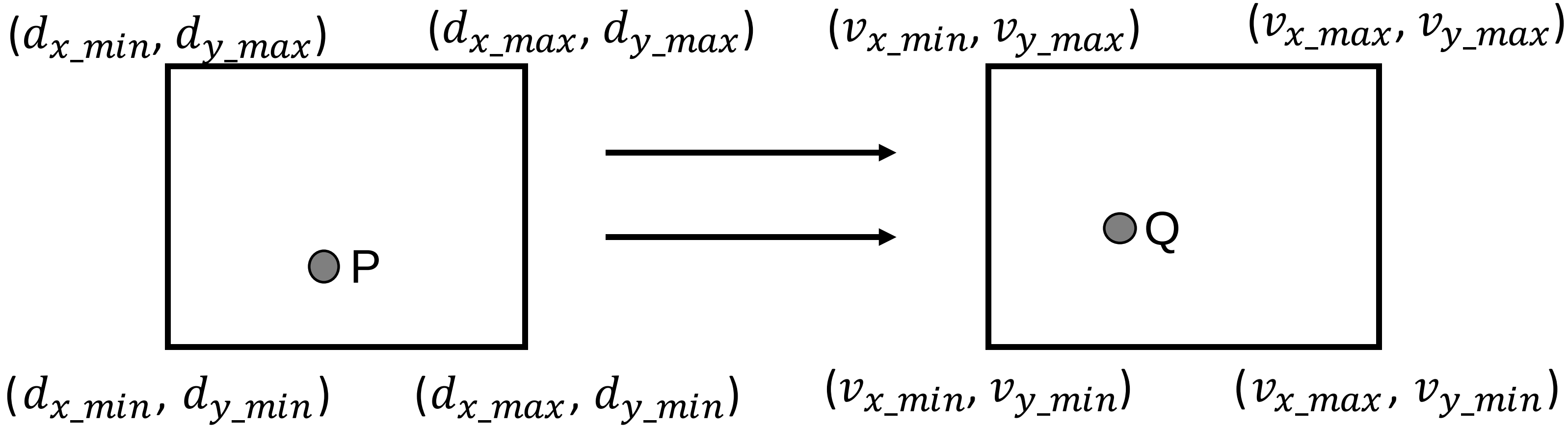
$S_x \neq S_y$(differential scaling)

Affine transformation



- View volume coordinates to screen coordinates
  - *Translate* a corner of view volume coordinates to the origin
  - *Scale* view volume coordinates to unit size
  - *Scale* normalized view coordinates to viewpoint size
  - *Translate* to viewpoint origin

Affine transformation



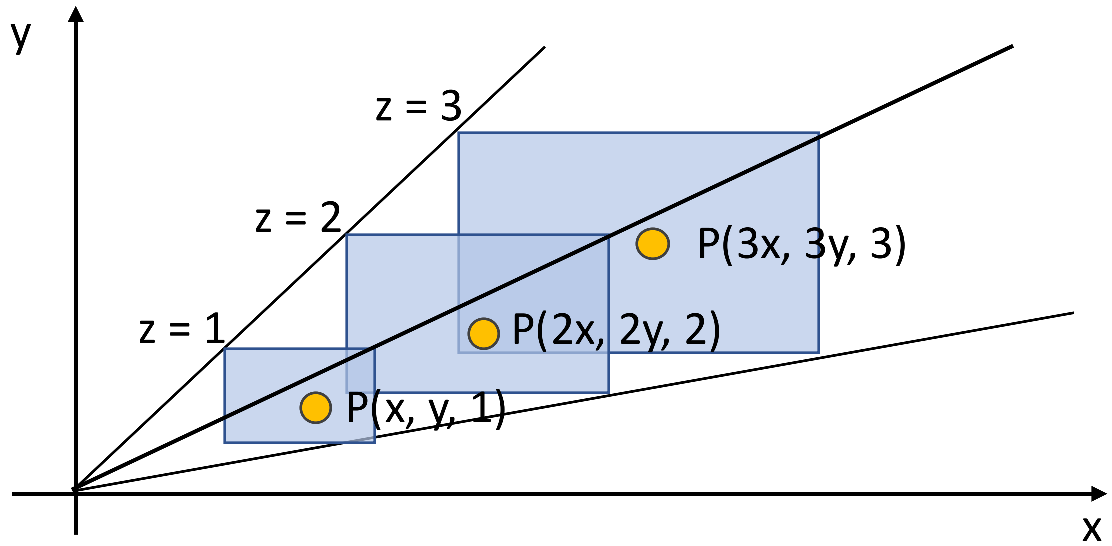
- *Translate* a corner of view volume coordinates to the origin: $\mathrm{T}(-d_{x\_min}, -d_{y\_min})$
- *Scale* view volume coordinates to unit size: $\mathrm{S}(S_{dx}, S_{dy})$
- *Scale* normalized view coordinates to viewpoint size: $\mathrm{S}(S_{vx}, S_{vy})$
- *Translate* to viewpoint origin: $\mathrm{T}(v_{x\_min}, v_{y\_min})$

$$\mathrm{Q} = \mathrm{T}(v_{x\_min}, v_{y\_min}) + \mathrm{S}(S_{vx}, S_{vy})\,\{\,\mathrm{S}(S_{dx}, S_{dy})\ [\mathrm{P} + \mathrm{T}(-d_{x\_min}, -d_{y\_min})\,]\}$$

Homogeneous Coordinate

- Motivation
  - $Translate: \vec{Q} = \vec{P} + \vec{T}(T_x, T_y)$
  - $Scale: \vec{Q} = S(S_x, S_y) * \vec{P}$
  - Translation involves a vector addition instead of a vector-matrix multiplication.
  - Better if all transforms are accomplished using a vector-matrix multiply.

Homogeneous Coordinate



- A two-dimensional point can be represented by one of the points along the ray in 3D space

$$P_{2d} = (x, y), \qquad P_H = (\mathrm{x}, \mathrm{y}, \mathrm{z})$$

- To convert from P_H to P_{2d}, divide each coordinate by the Z coordinate and discard the 3rd coordinate.

Homogeneous Matrices

- Translation

$$T_H(T_x, T_y) = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} P_x + T_x \\ P_y + T_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

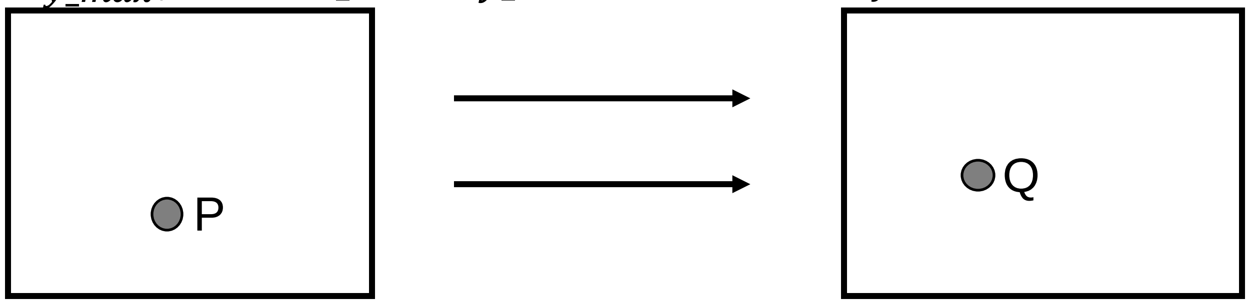
- Scale

$$S_H(S_x, S_y) = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} = \begin{bmatrix} P_x S_x \\ P_y S_y \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

Affine transformation

(d_{x_min}, d_{y_max}) (d_{x_max}, d_{y_max}) (v_{x_min}, v_{y_max}) (v_{x_max}, v_{y_max})



(d_{x_min}, d_{y_min}) (d_{x_max}, d_{y_min}) (v_{x_min}, v_{y_min}) (v_{x_max}, v_{y_min})

$\mathrm{Q} = \mathrm{T}(v_{x\_min}, v_{y\_min}) + \mathrm{S}(S_{vx}, S_{vy})\ \{\ \mathrm{S}(S_{dx}, S_{dy})\ \ [\mathrm{P} + \mathrm{T}(-d_{x\_min}, -d_{y\_min})\ ]\}$

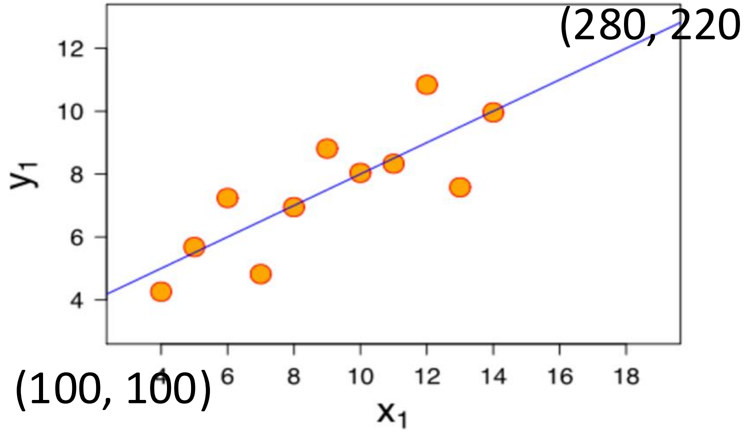
$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix}$$

In-class exercise

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

x: [2, 20], y: [2, 14]



$$Q = T(v_{x\_min}, v_{y\_min}) + S(S_{vx}, S_{vy}) \{ S(S_{dx}, S_{dy}) \ [P + T(-d_{x\_min}, - d_{y\_min})]\}$$

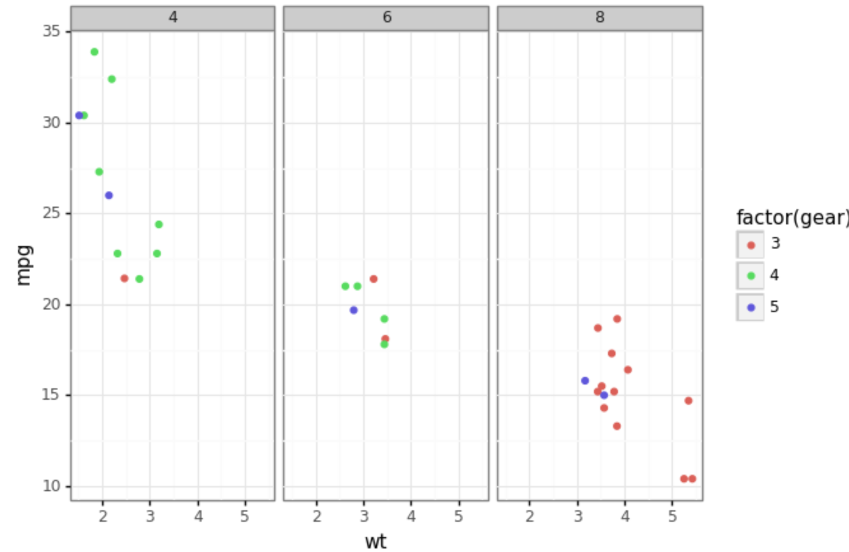
$$\begin{bmatrix} Q_x \\ Q_y \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} P_x \\ P_y \\ 1 \end{bmatrix} =$$

Why it matters

- Fully automatically in *ggplot*

```
(ggplot(mtcars,
        aes('wt', 'mpg',
            color='factor(gear)'))
        + geom_point()
        + facet_wrap('~cyl')
        + theme_bw())
```



	name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

Why it matters

- Need to specify in *d3.js*

```
// Add X axis
var x = d3.scaleLinear().domain([0, 1]).range([0, vp.width]);
```

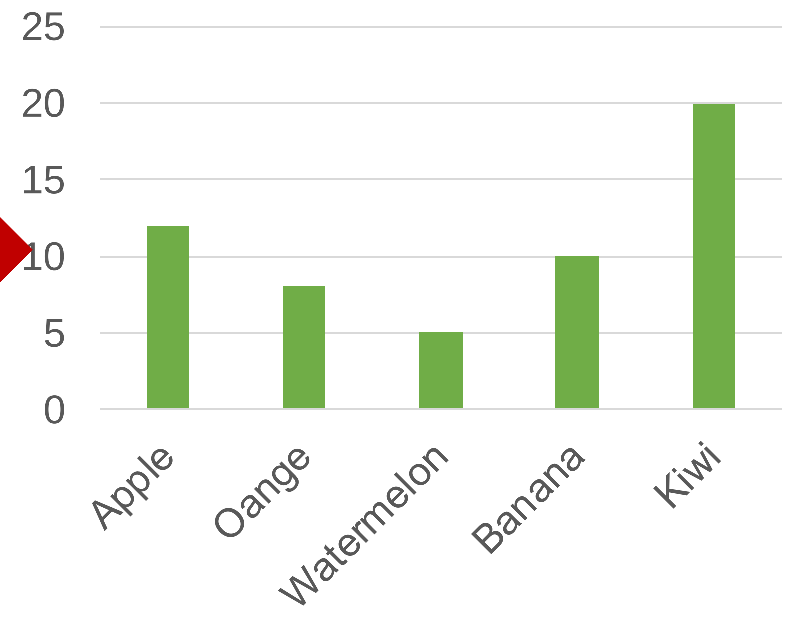
```
// Add Y axis
var y = d3.scaleLinear().domain([0, 1]).range([vp.height, 0]);
```

- How about visualizations on other displays like mobile phone?
  - Responsive data visualization
  - Demo: http://nrabinowitz.github.io/rdv/

In-class exercise

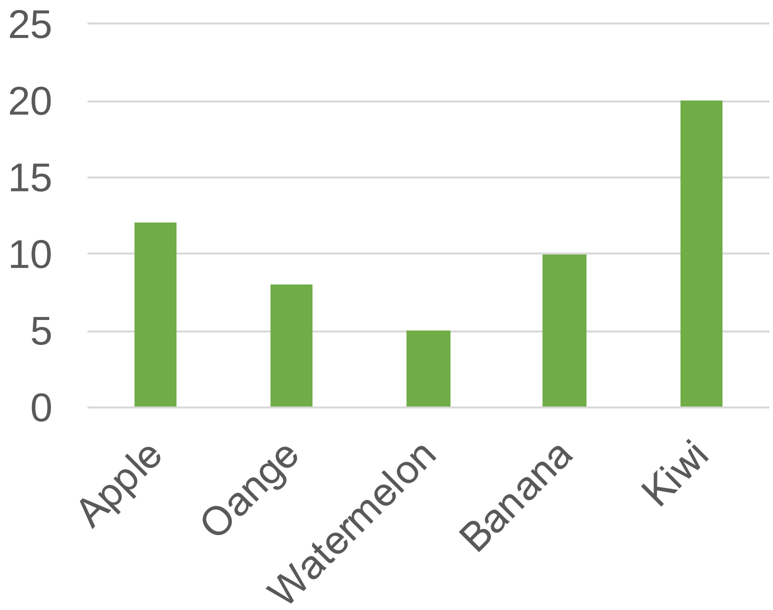
- All data above are quantitative attributes.
- How can we map data of categorical/ordered attributes to positions?

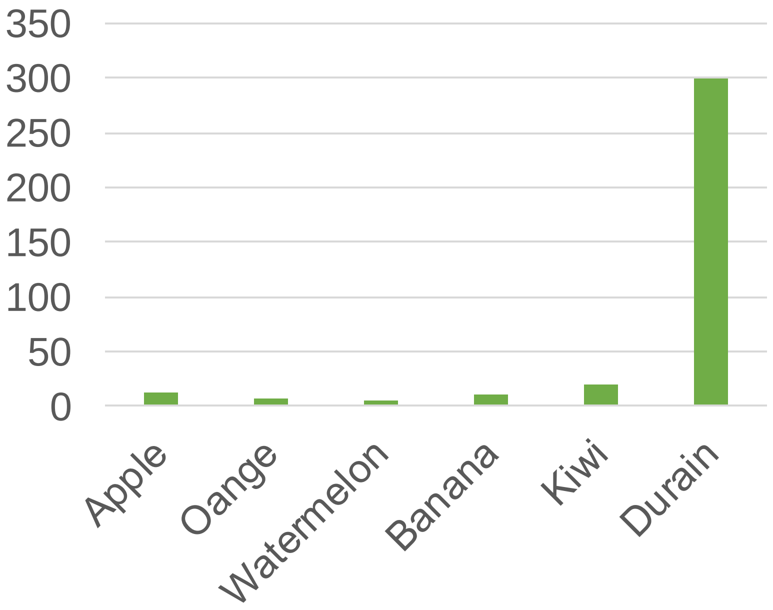
Fruit	Price
Apple	12
Orange	8
Watermelon	5
Banana	10
Kiwi	20



Questions

- How to set the maximum value?
  - Why choose 25?
  - What if we have another item costs 300?





Data Exploration & Visualization

Module 2: Data Transformation & Mapping

- Data transformation
  - Aggregation, sampling, discretization, attribute transformation, arrangement, selection, feature creation
- Mapping data to visuals
  - Numbers to positions
- D3 implementation

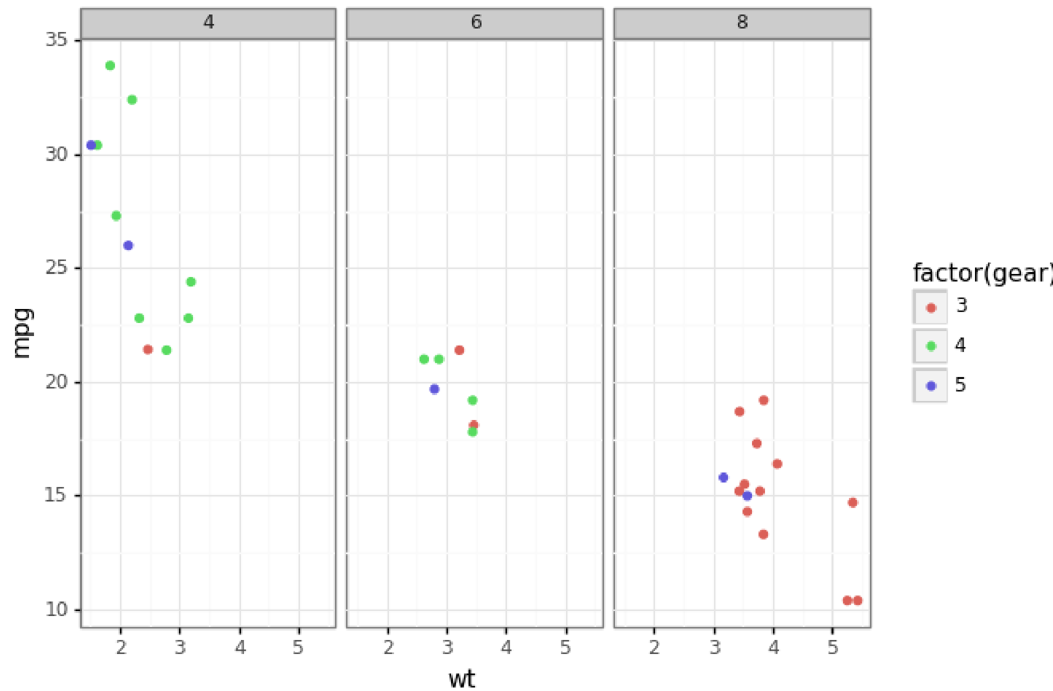
Grammar and languages

- “*Grammar of Graphics*,” Wilkinson, 1999
  - First proposed a grammar for constructing layered visualizations
  - Concepts include:
    - Data, Scale, Geometry, Coordinates, Facets, and Aesthetics, etc.

- “*ggplot2*,” Wickham, 2005
  - A visualization library in R
  - Implemented the *Grammar of Graphics*
    - Made modifications to focus more on the layers
    - “*A Layered Grammar of Graphics*,” Wickham, 2010

ggplot2

```
(ggplot(mtcars,
        aes('wt', 'mpg',
            color='factor(gear)'))
 + geom_point()
 + facet_wrap('~cyl')
 + theme_bw())
```

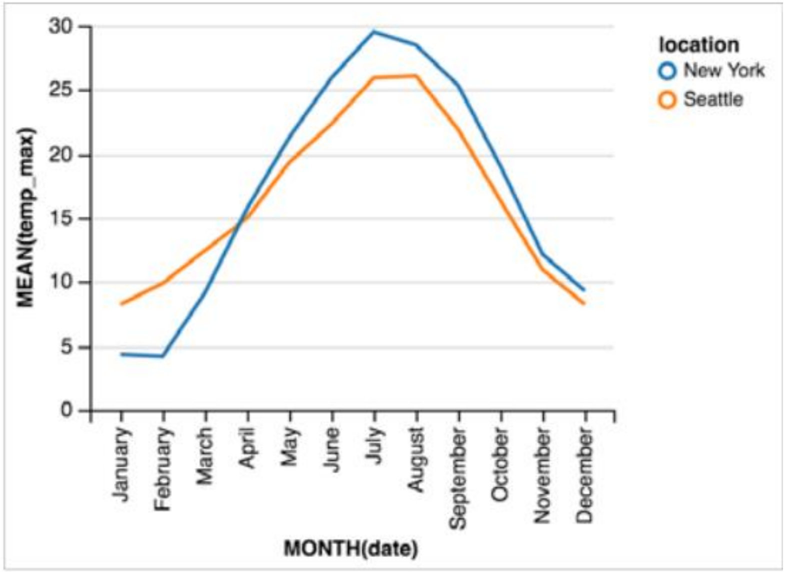


	name	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

D3, Vega, Vega-lite

- Outside of the *R*, Jeff Heer (Univ. of Washington) has been working on a similar effort but for web-based development
  - Flare (2005), Heer. (Note: Written in Java)
  - Protovis (2009), Bostock and Heer
- D3.js (2011), Bostock and Heer
  - Key feature: maps data to SVG elements
- Vega (2015), Satyanarayan et al.
  - Key feature: a specification based language
- Vega-Lite (2016), Satyanarayan et al.
  - Key feature: makes interactivity a first-class citizen

```
{
  "data": {
    "url": "data/weather.csv",
    "formatType": "csv" },
  "mark": "line",
  "encoding": {
    "x": {
      "field": "date",
      "type": "temporal",
      "timeUnit": "month" },
    "y": {
      "field": "temp_max",
      "type": "quantitative",
      "aggregate": "mean" },
    "color": {
      "field": "location",
      "type": "nominal" }
  }
}
```



Example Vega-lite language and visualization