Data Visualization (CIS 490/680)

Data

Dr. David Koop

JavaScript in one slide

- Interpreted and Dynamically-typed Programming Language
- Statements end with semi-colons, normal blocking with brackets
- Variables: `var a = 0; let b = 2;`
- Operators: `+, -, *, /, [ ]`
- Control Statements: `if (<expr>){…} else {…}, switch`
- Loops: `for, while, do-while`
- Arrays: `var a = [1,2,3]; a[99] = 100; console.log(a.length);`
- Functions: `function myFunction(a,b) { return a + b; }`
- Objects: `var obj; obj.x = 3; obj.y = 5;`
  - Prototypes for instance functions
- Comments are `/* Comment */` or `// Single-line Comment`

Including JavaScript in HTML

- Use the script tag
- Can either inline JavaScript or load it from an external file

```
- <script type="text/javascript">
      a = 5, b = 8;
      c = a * b + b - a;
  </script>
  <script type="text/javascript" src="script.js"/>
```

- Script tag can reference local or **remote** external javascript files
- The order the javascript is in is the order it is executed
- Example: in the above, `script.js` can access the variables `a`, `b`, and `c`

JavaScript Features

- Any object can serve as an associative array

```
states = {"AZ": "Arizona", "MA": "Massachusetts"};
```

- Array functions: `map`, `filter`, `reduce`, `forEach`
  - `Object.keys(states).filter(d => d.startsWith("A"));`
- Function chaining is common (sometimes the original object is returned, others another object is returned)
  - `$("#myElt").css("color", "blue").height(200).width(320)`
- Closures are functions that "remember their environments" [MDN]
  - ```
    function makeAdder(x) {
      return function(y) {
        return x + y;
      };
    }
    var add5 = makeAdder(5);
    ```
    ```

JavaScript Objects

- `var student = {name: "John Smith", id: "000012345", class: "Senior", hometown: "Peoria, IL, USA"};`
- Objects contain multiple values: key-value pairs called **properties**
- Accessing properties via dot-notation: `student.name`
- Always works via bracket-notation: `student["name"]`
- May also contain functions:
  - ```
    var student = {firstName: "John",
                   lastName: "Smith",
                   fullName: function() { return this.firstName +
                                          " " + this.lastName; }};
    ```
  - `student.fullName()`
    ```

Functional Programming in JavaScript

- Functions are first-class objects in JavaScript
- You can pass a function to a method just like you can pass an integer, string, or object
- Instead of writing loops to process data, we can instead use a `map/filter/reduce/forEach` function on the data that runs our logic for each data item
- `map`: transform each element of an array
- `filter`: check each element of an array and keep only ones that pass
- `forEach`: run the function for each element of the array
- `reduce`: collapse an array to a single object

Using Array Functions

- `var a = [2, 4, 7, 11, 22, 84];`
- Named function:
  - ```
    function isEven(d) {
        return (d % 2 == 0);
    }
    a.filter(isEven);
    ```
- Anonymous function
  - `a.filter(function(d) { return (d % 2 == 0); });`
- Arrow function
  - `a.filter(d => (d % 2 == 0));`
    ```

Manipulating the DOM with JavaScript

- Key global variables:
  - `window`: Global namespace
  - `document:` Current document
  - `document.getElementById(…)`: Get an element via its id
- HTML is parsed into an in-memory document (DOM)
- Can access and **modify** information stored in the DOM
- Can add information to the DOM

Example: JavaScript and the DOM

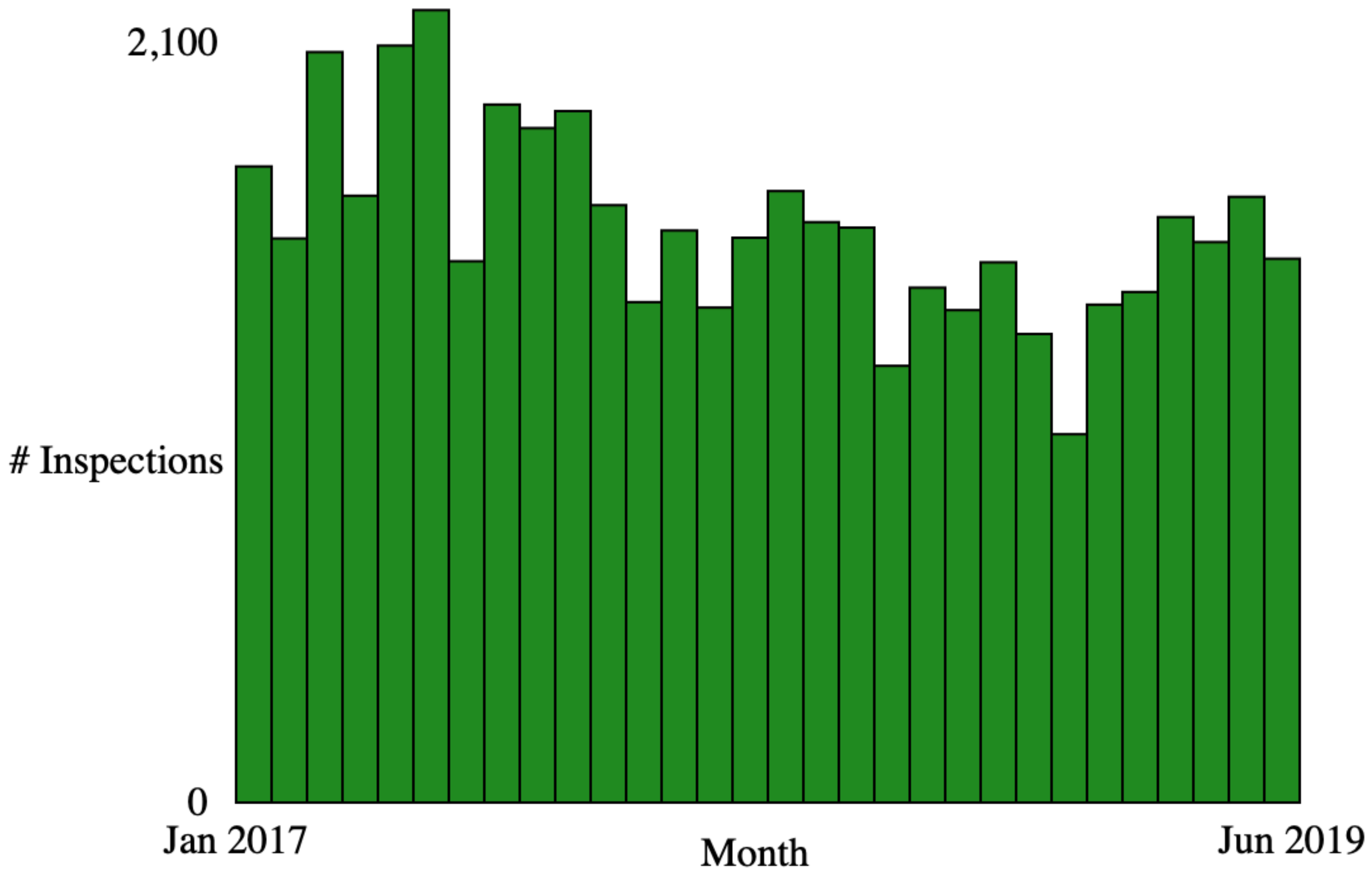
- Start with no real content, just divs:

```
<div id="firstSection"></div>
<div id="secondSection"></div>
<div id="finalSection"></div>
```

- Get existing elements:
  - `document.querySelector`
  - `document.getElementById`
- Programmatically add elements:
  - `document.createElement`
  - `document.createTextNode`
  - `Element.appendChild`
  - `Element.setAttribute`
- Link

Assignment 2

- Link
- Three parts: table, horizontal bar chart, vertical bar chart
  - data processing
  - highlighting (CS 680)
- Vertical chart can be tricky
- Start early!
- Questions?



Creating SVG figures via JavaScript

- SVG elements can be accessed and modified just like HTML elements
- Create a new SVG programmatically and add it into a page:
  - ```
    var divElt = document.getElementById("chart");
    var svg = document.createElementNS(
                   "http://www.w3.org/2000/svg", "svg");
    divElt.appendChild(svg);
    ```
- You can assign attributes:
  - ```
    svg.setAttribute("height", 400);
    svg.setAttribute("width", 600);
    svgCircle.setAttribute("r", 50);
    ```

Manipulating SVG via JavaScript

- SVG can be navigated just like the DOM
- Example:

```
function addEltToSVG(svg, name, attrs) {
    var element = document.createElementNS(
                 "http://www.w3.org/2000/svg", name);
    if (attrs === undefined) attrs = {};
    for (var key in attrs) {
       element.setAttribute(key, attrs[key]);
    }
    svg.appendChild(element);
}
mysvg = document.getElementById("mysvg");
addEltToSVG(mysvg, "rect", {"x": 50, "y": 50,
                            "width": 40,"height": 40,
                            "fill": "blue"});
```

- Notebook

SVG Manipulation Example

- Draw a horizontal bar chart
  - `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`
- Steps?

SVG Manipulation Example

- Draw a horizontal bar chart
  - `var a = [6, 2, 6, 10, 7, 18, 0, 17, 20, 6];`
- Steps:
  - Programmatically create SVG
  - Create individual rectangle for each item
- Link:
  - https://codepen.io/dakoop/pen/mdbxQKe

"Computer-based visualization systems provide visual representations of **datasets** designed to help people carry out tasks more effectively."

— T. Munzner

Data

- What is this data?

R011	42ND STREET & 8TH AVENUE	00228985	00008471	00000441	00001455	00000134	00033341	00071255
R170	14TH STREET-UNION SQUARE	00224603	00011051	00000827	00003026	00000660	00089367	00199841
R046	42ND STREET & GRAND CENTRAL	00207758	00007908	00000323	00001183	00003001	00040759	00096613

- **Semantics**: real-world meaning of the data
- **Type**: structural or mathematical interpretation
- Both often require **metadata**
  - Sometimes we can infer some of this information
  - Line between data and metadata isn't always clear

Semantics

- The meaning of the data
- Example: 94023, 90210, 52790, 02747

Semantics

- The meaning of the data
- Example: 94023, 90210, 52790, 02747
  - Attendance at college football games?

Semantics

- The meaning of the data
- Example: 94023, 90210, 52790, 02747
  - Attendance at college football games?
  - Salaries?

Semantics

- The meaning of the data
- Example: 94023, 90210, 52790, 02747
  - Attendance at college football games?
  - Salaries?
  - Zip codes?
- Cannot always infer based on what the data looks like
- Often require semantics to better understand data
- Column names help with semantics
- May also include rules about data: a zip code is part of an address that uniquely identifies a residence
- Useful for asking good questions about the data

Data

	REMOTE	STATION	FF ▼	SEN/DIS	7-D AFAS UNL	D AFAS/RMF I	JOINT RR TKT	7-D UNL	30-D UNL
1	R011	42ND STREET & 8TH AVENUE	00228985	00008471	00000441	00001455	00000134	00033341	00071255
2	R170	14TH STREET-UNION SQUARE	00224603	00011051	00000827	00003026	00000660	00089367	00199841
3	R046	42ND STREET & GRAND CENTRAL	00207758	00007908	00000323	00001183	00003001	00040759	00096613
4	R012	34TH STREET & 8TH AVENUE	00188311	00006490	00000498	00001279	00003622	00035527	00067483
5	R293	34TH STREET - PENN STATION	00168768	00006155	00000523	00001065	00005031	00030645	00054376
6	R033	42ND STREET/TIMES SQUARE	00159382	00005945	00000378	00001205	00000690	00058931	00078644
7	R022	34TH STREET & 6TH AVENUE	00156008	00006276	00000487	00001543	00000712	00058910	00110466
8	R084	59TH STREET/COLUMBUS CIRCLE	00155262	00009484	00000589	00002071	00000542	00053397	00113966
9	R020	47-50 STREETS/ROCKEFELLER	00143500	00006402	00000384	00001159	00000723	00037978	00090745
10	R179	86TH STREET-LEXINGTON AVE	00142169	00010367	00000470	00001839	00000271	00050328	00125250
11	R023	34TH STREET & 6TH AVENUE	00134052	00005005	00000348	00001112	00000649	00031531	00075040
12	R029	PARK PLACE	00121614	00004311	00000287	00000931	00000792	00025404	00065362
13	R047	42ND STREET & GRAND CENTRAL	00100742	00004273	00000185	00000704	00001241	00022808	00068216

Data Terminology

- Items
  - An **item** is an individual discrete entity
  - e.g. row in a table, node in a network
- Attributes
  - An **attribute** is some specific property that can be measured, observed, or logged
  - a.k.a. variable, (data) dimension
  - e.g. a column in a table

Items & Attributes

A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box	[illegible]	7/17/07
32	7/16/07	2-High	Medium Box	[illegible]	7/18/07
32	7/16/07	2-High	Medium Box	[illegible]	7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	[illegible]	5-Low	Wrap Bag	0.56	1/20/05
69	[illegible]	4-Not Specified	Small Pack	0.44	6/6/05
69	[illegible]	4-Not Specified	Wrap Bag	0.6	6/6/05
70	12/18/06	5-Low	Small Box	0.59	12/23/06
70	12/18/06	5-Low	Wrap Bag	0.82	12/23/06
96	4/17/05	2-High	Small Box	0.55	4/19/05
97	1/29/06	3-Medium	Small Box	0.38	1/30/06
129	11/19/08	5-Low	Small Box	0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

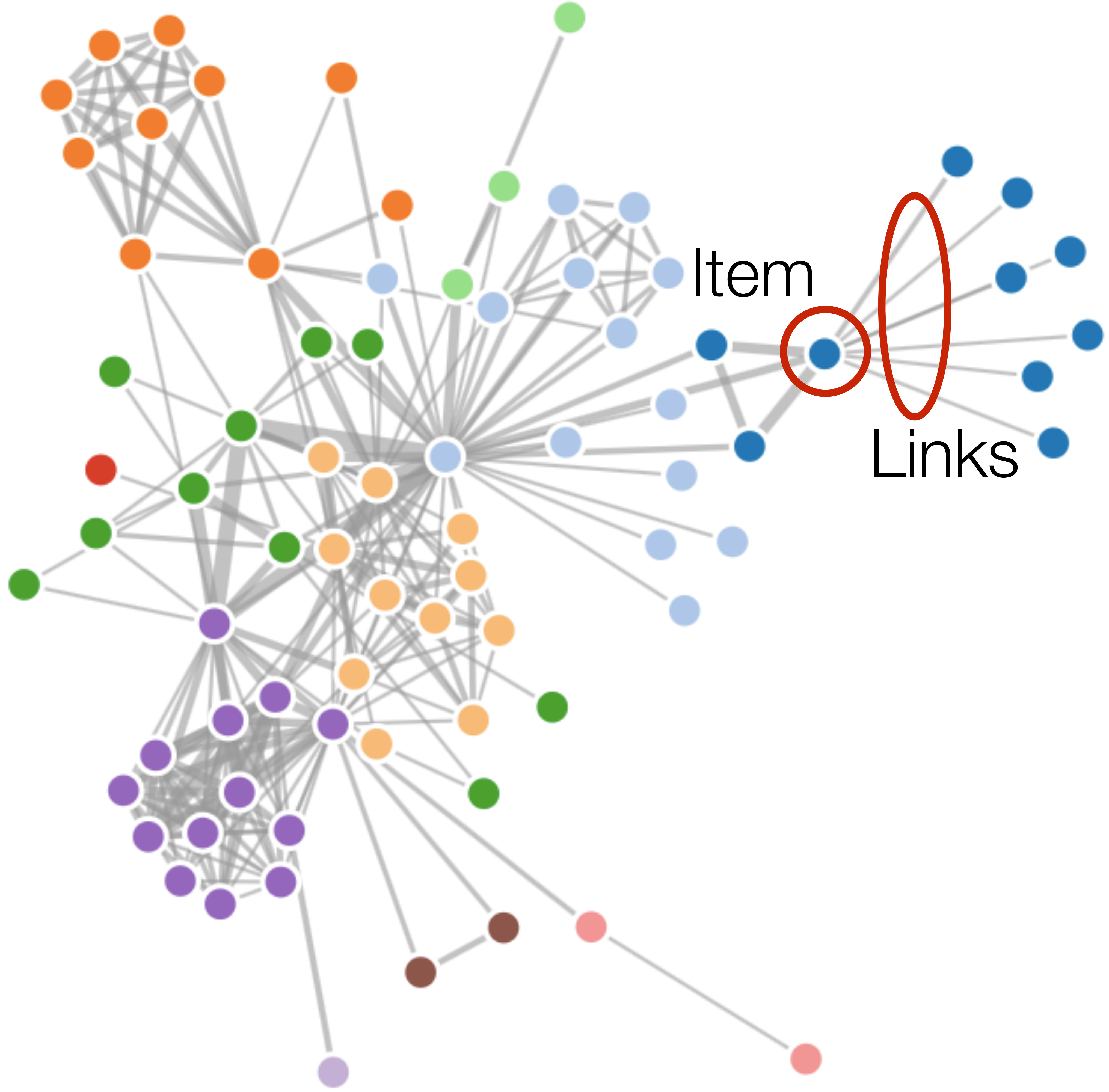
attribute

item

Data Types

- Nodes
  - Synonym for item but in the context of networks (graphs)
- Links
  - A **link** is a relation between two items
  - e.g. social network friends, computer network links

Items & Links

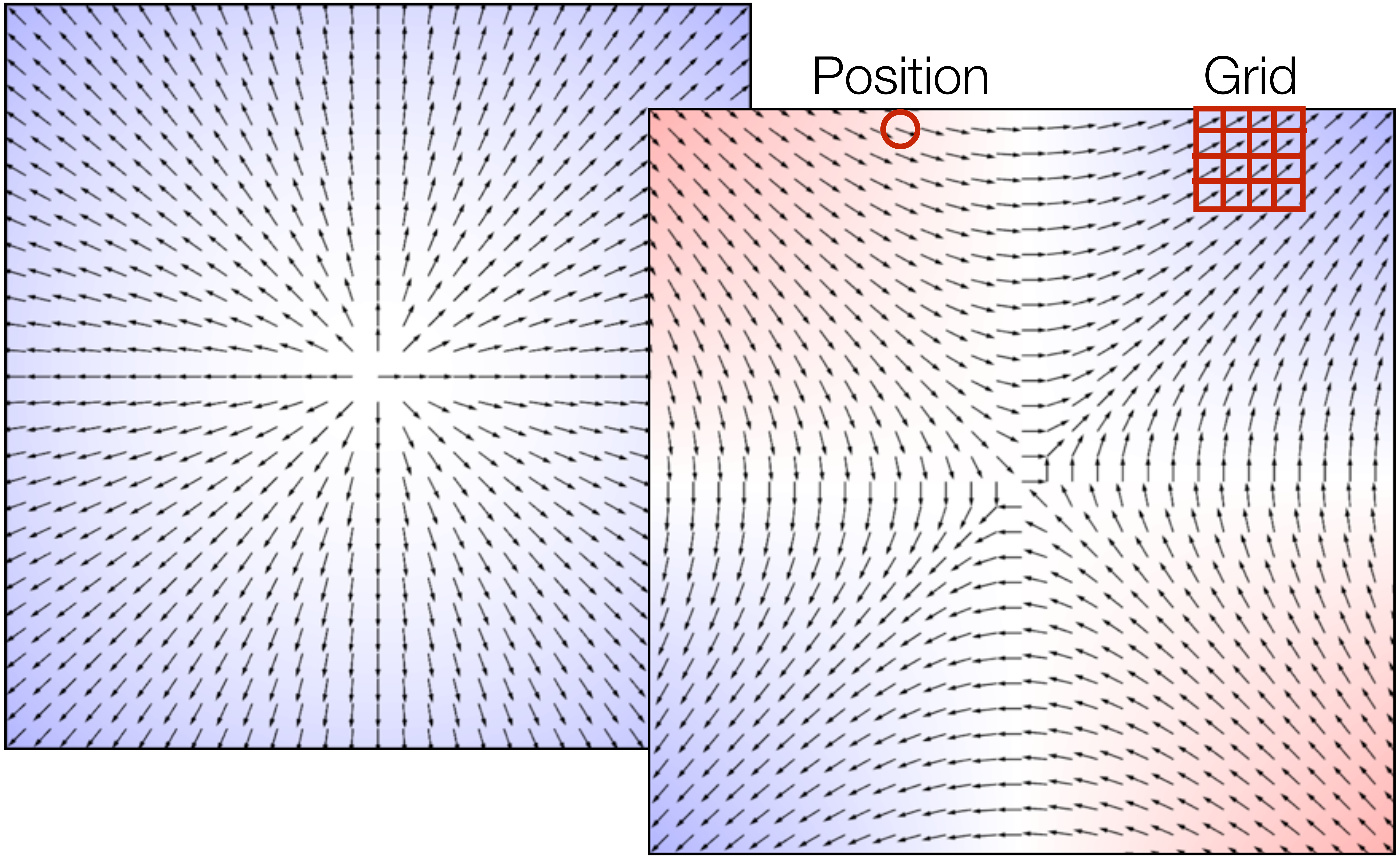


[Bostock, 2011]

Data Types

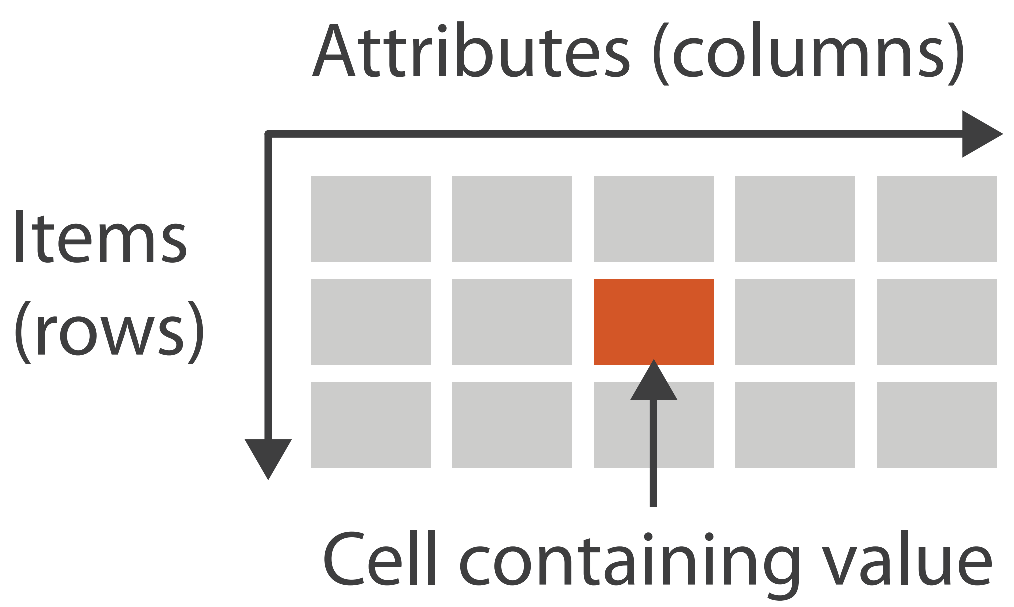
- Positions:
  - A **position** is a location in space (usually 2D or 3D)
  - May be subject to projections
  - e.g. cities on a map, a sampled region in an CT scan
- Grids:
  - A **grid** specifies how data is sampled both geometrically and topologically
  - e.g. how CT scan data is stored

Positions and Grids

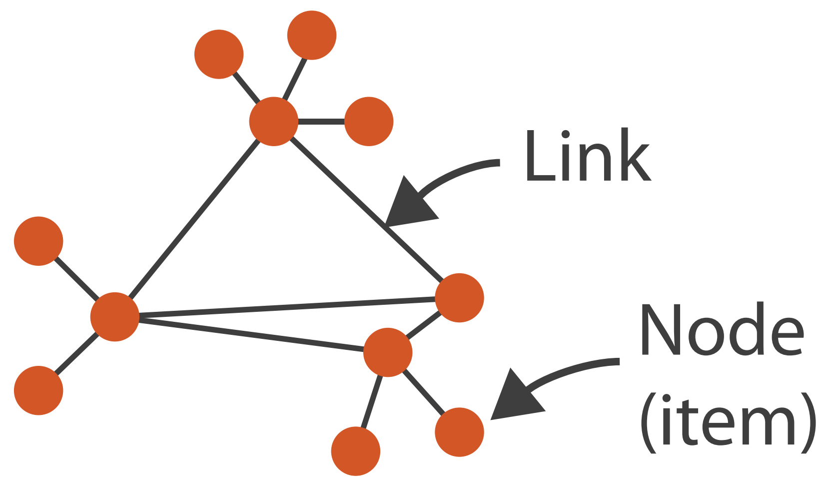


Dataset Types

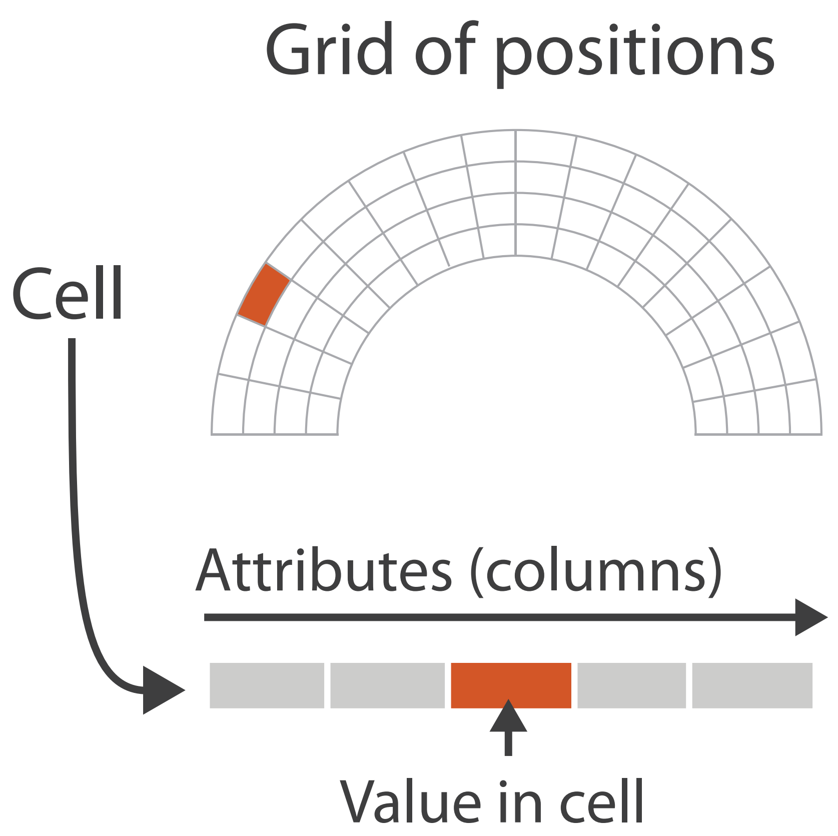
- Tables



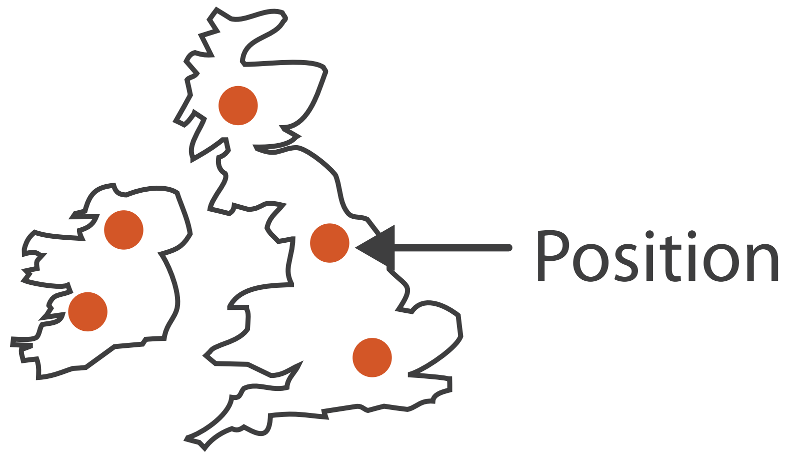
- Networks



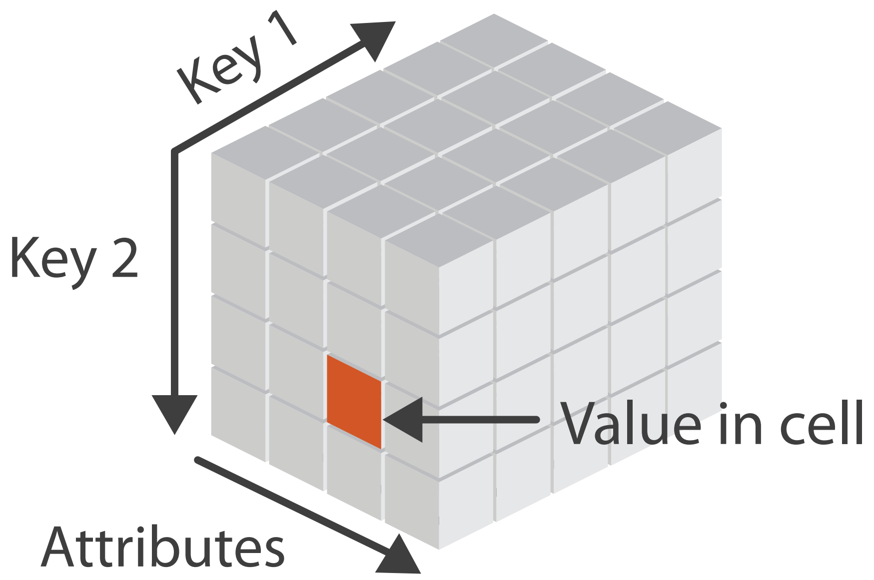
- Fields (Continuous)



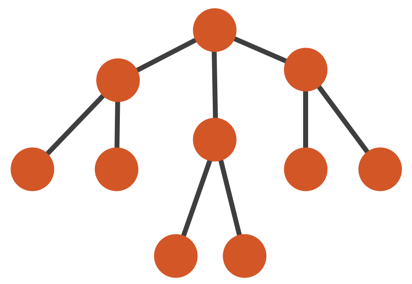
- Geometry (Spatial)



- *Multidimensional Table*



- *Trees*



[Munzner (ill. Maguire), 2014]

Tables

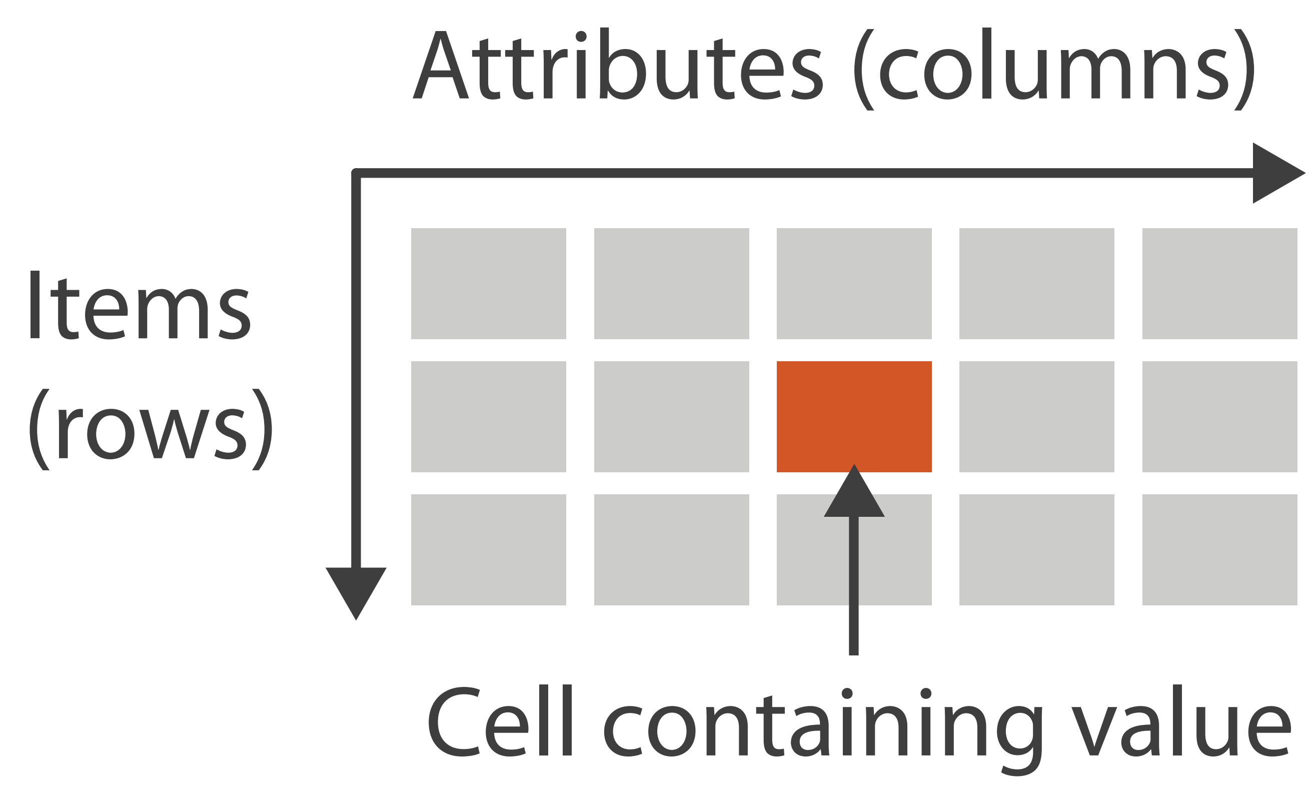
A	B	C	S	T	U
Order ID	Order Date	Order Priority	Product Container	Product Base Margin	Ship Date
3	10/14/06	5-Low	Large Box	0.8	10/21/06
6	2/21/08	4-Not Specified	Small Pack	0.55	2/22/08
32	7/16/07	2-High	Small Pack	0.79	7/17/07
32	7/16/07	2-High	Jumbo Box	[illegible]	7/17/07
32	7/16/07	2-High	Medium Box	[illegible]	7/18/07
32	7/16/07	2-High	Medium Box	[illegible]	7/18/07
35	10/23/07	4-Not Specified	Wrap Bag	0.52	10/24/07
35	10/23/07	4-Not Specified	Small Box	0.58	10/25/07
36	11/3/07	1-Urgent	Small Box	0.55	11/3/07
65	3/18/07	1-Urgent	Small Pack	0.49	3/19/07
66	[illegible]	5-Low	Wrap Bag	0.56	1/20/05
69	[illegible]	4-Not Specified	Small Pack	0.44	6/6/05
69	[illegible]	4-Not Specified	Wrap Bag	0.6	6/6/05
70	12/18/06	5-Low	Small Box	0.59	12/23/06
70	12/18/06	5-Low	Wrap Bag	0.82	12/23/06
96	4/17/05	2-High	Small Box	0.55	4/19/05
97	1/29/06	3-Medium	Small Box	0.38	1/30/06
129	11/19/08	5-Low	Small Box	0.37	11/28/08
130	5/8/08	2-High	Small Box	0.37	5/9/08
130	5/8/08	2-High	Medium Box	0.38	5/10/08
130	5/8/08	2-High	Small Box	0.6	5/11/08
132	6/11/06	3-Medium	Medium Box	0.6	6/12/06
132	6/11/06	3-Medium	Jumbo Box	0.69	6/14/06
134	5/1/08	4-Not Specified	Large Box	0.82	5/3/08
135	10/21/07	4-Not Specified	Small Pack	0.64	10/23/07
166	9/12/07	2-High	Small Box	0.55	9/14/07
193	8/8/06	1-Urgent	Medium Box	0.57	8/10/06
194	4/5/08	3-Medium	Wrap Bag	0.42	4/7/08

attribute

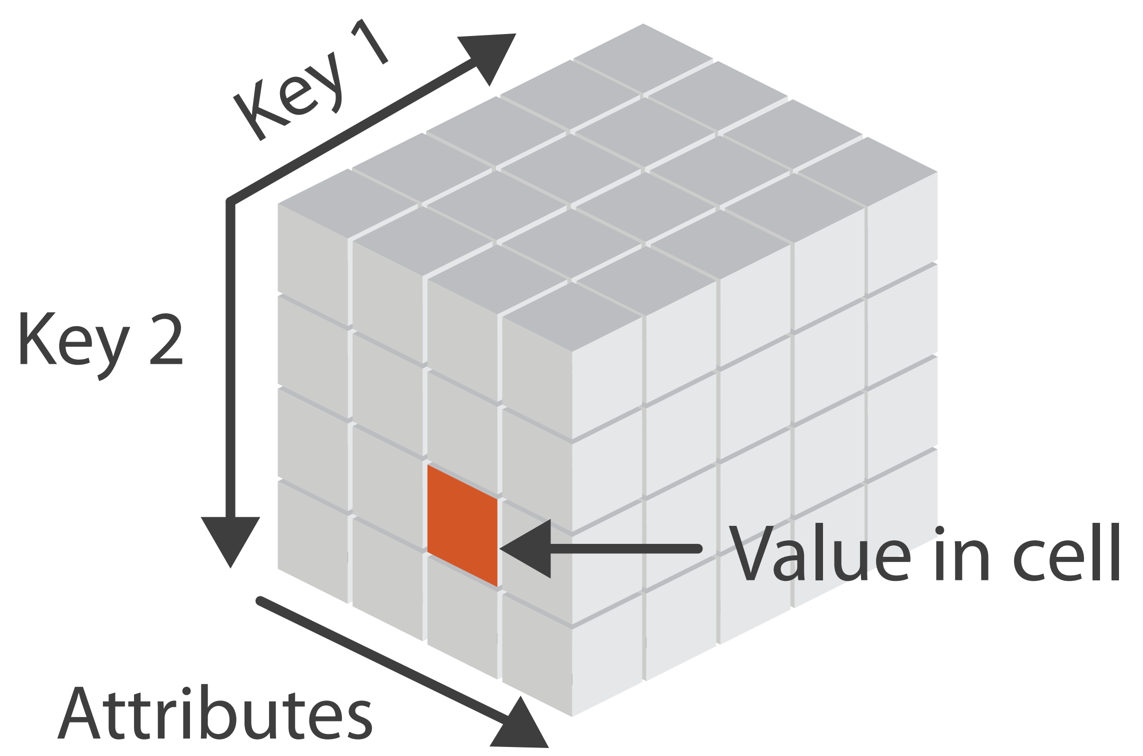
cell

item

Tables



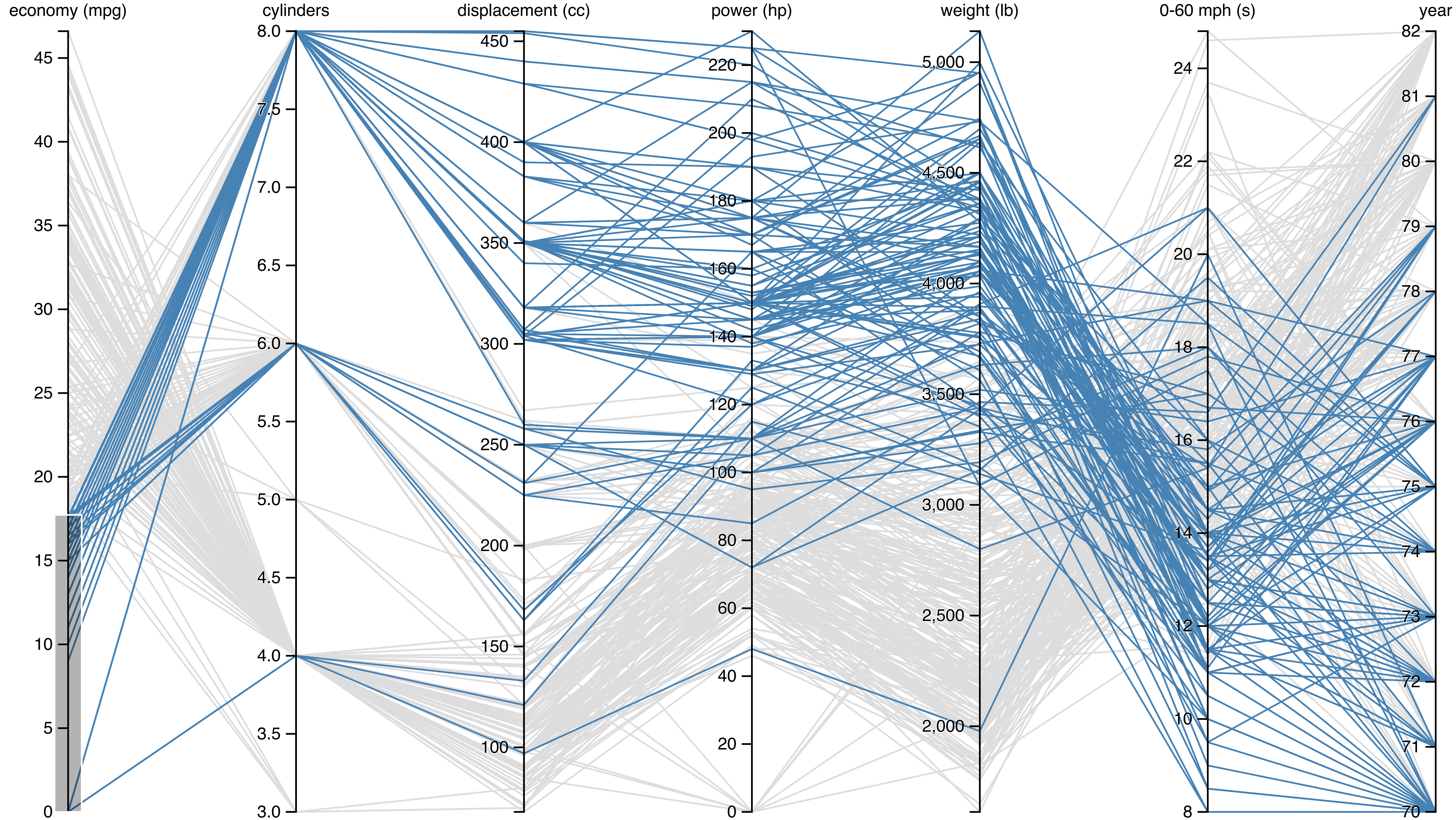
➔ *Multidimensional Table*



- Data organized by rows & columns
  - row ~ item (usually)
  - column ~ attribute
  - label ~ attribute name
- Key: identifies each item (row)
  - Usually **unique**
  - Allows **join** of data from 2+ tables
  - Compound key: key split among multiple columns, e.g. (state, year) for population
- Multidimensional:
  - Split compound key: data cube with (state, year)

[Munzner (ill. Maguire), 2014]

Table Visualizations

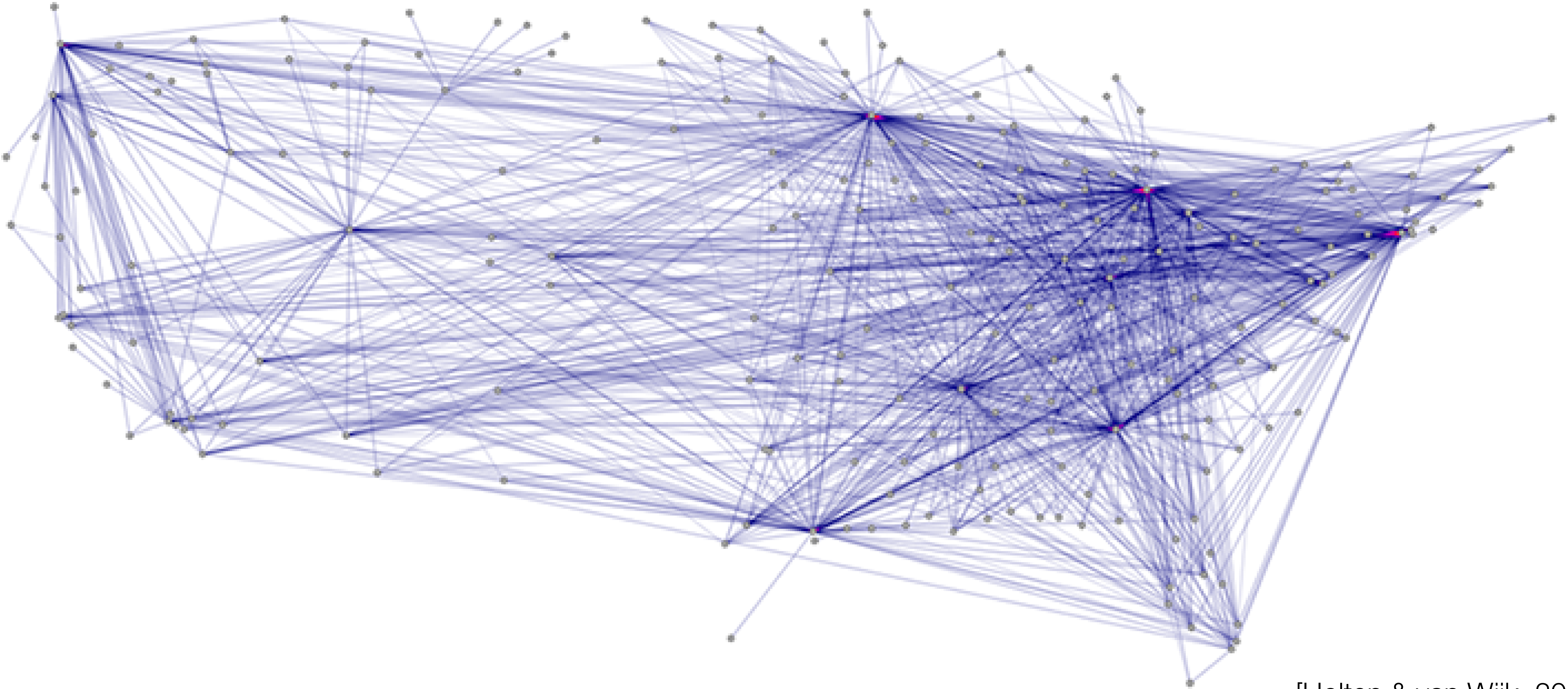


[M. Bostock, 2011]

Networks

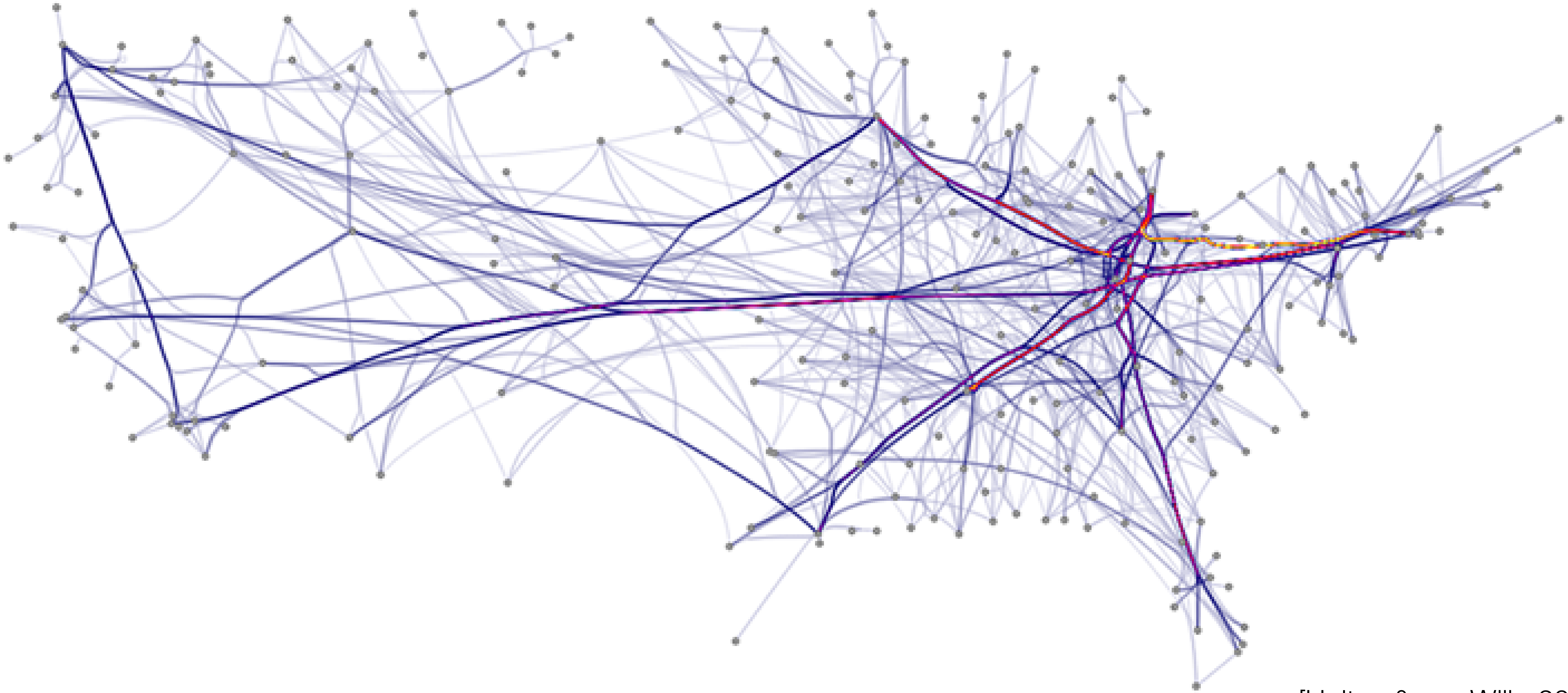
- Why networks instead of graphs?
- Tables can represent networks
  - Many-many relationships
  - Also can be stored as specific graph databases or files

Networks



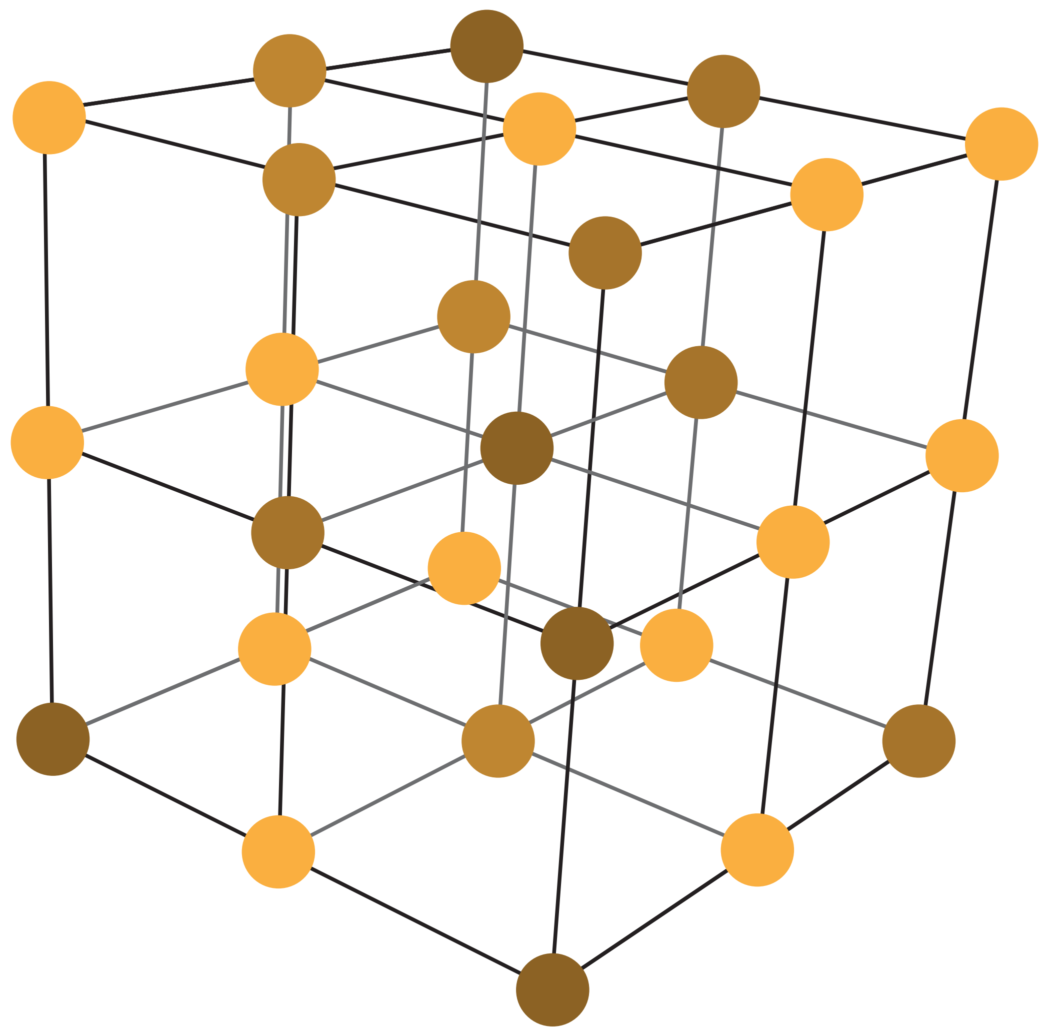
[Holten & van Wijk, 2009]

Networks



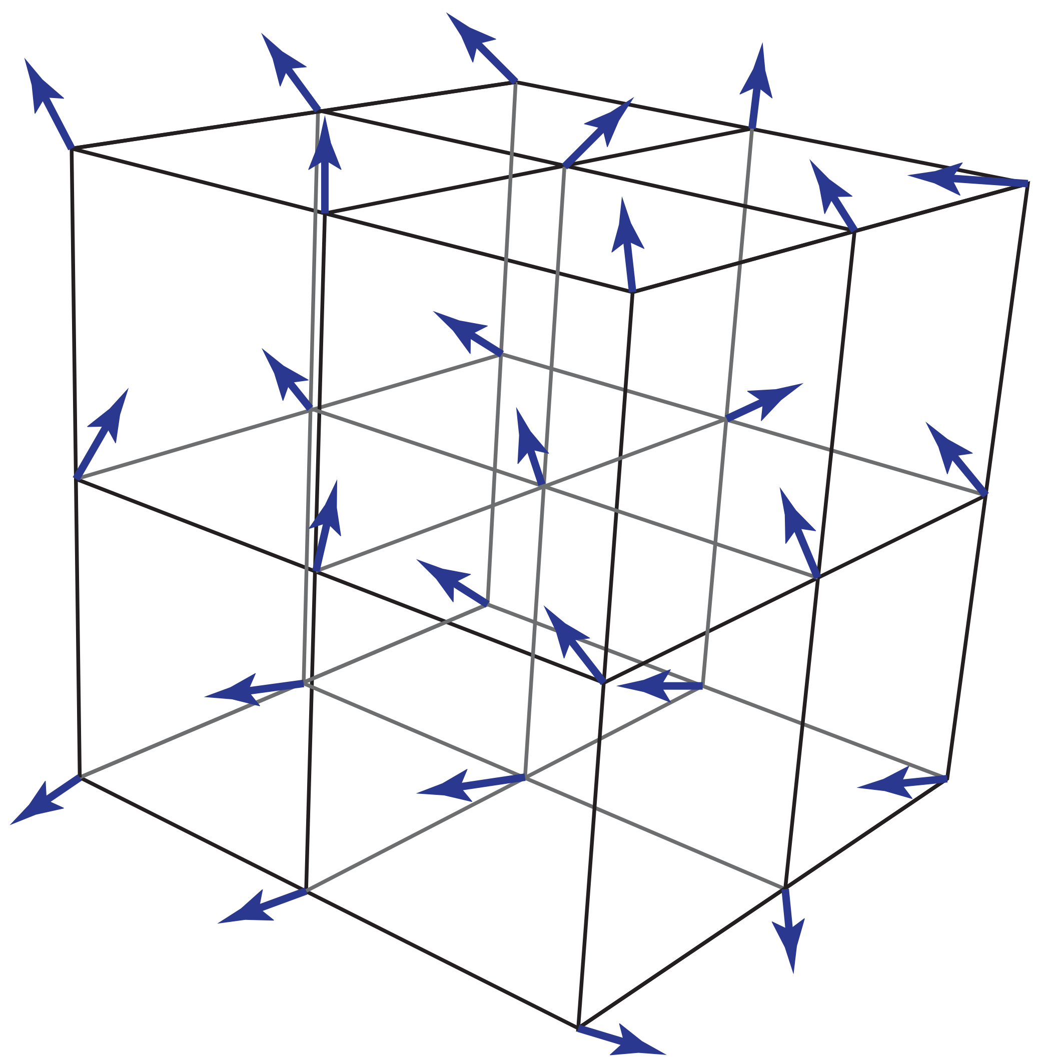
[Holten & van Wijk, 2009]

Fields



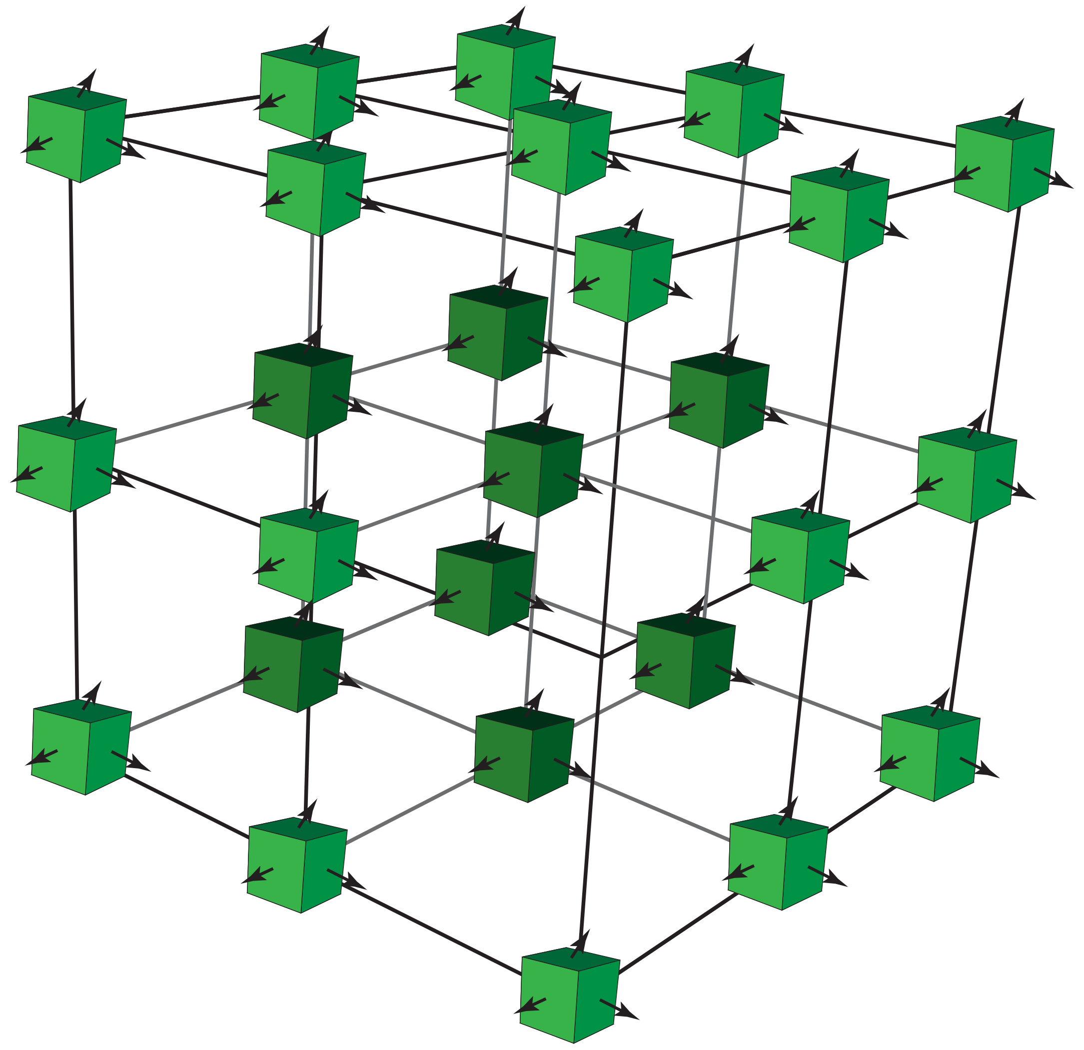
Scalar Fields

(Order-0 Tensor Fields)



Vector Fields

(Order-1 Tensor Fields)



Tensor Fields

(Order-2+)

Each point in space has an associated...

$$s_0$$

Scalar

$$\begin{bmatrix} v_0 \\ v_1 \\ v_2 \end{bmatrix}$$

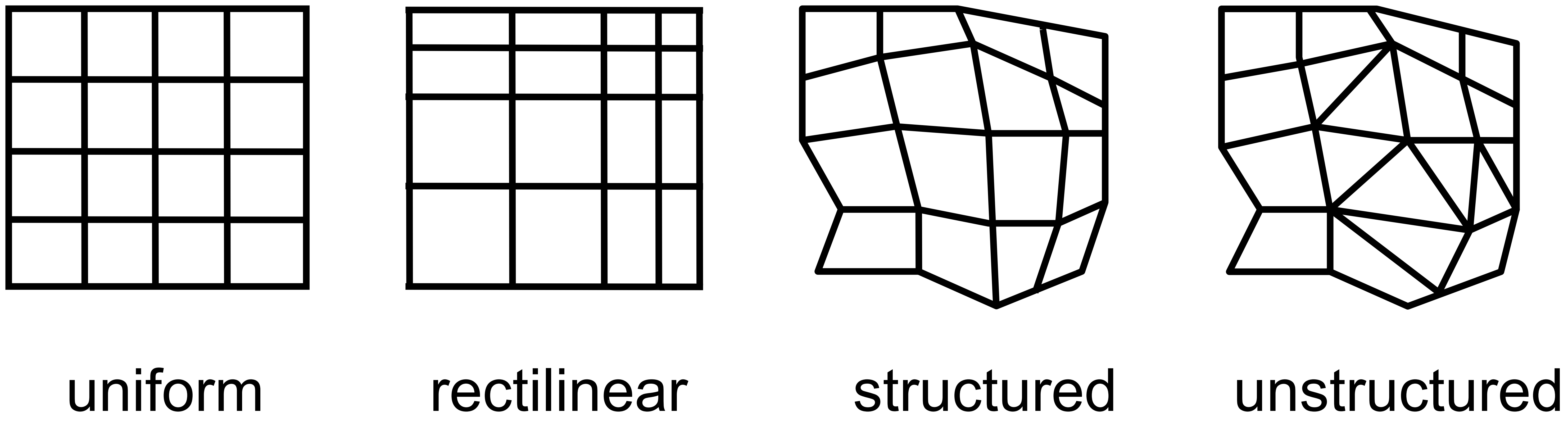
Vector

$$\begin{bmatrix} \sigma_{00} & \sigma_{01} & \sigma_{02} \\ \sigma_{10} & \sigma_{11} & \sigma_{12} \\ \sigma_{20} & \sigma_{21} & \sigma_{22} \end{bmatrix}$$

Tensor

Fields

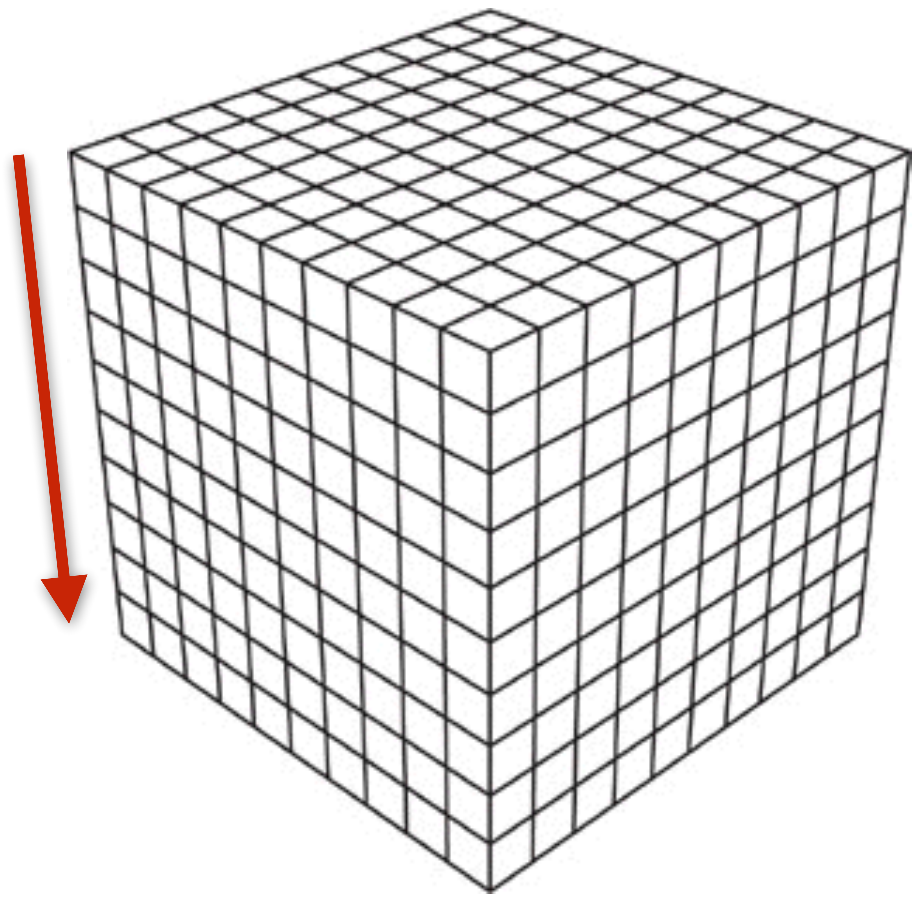
- Difference between **continuous** and **discrete** values
- Examples: temperature, pressure, density
- **Grids** necessary to sample continuous data:

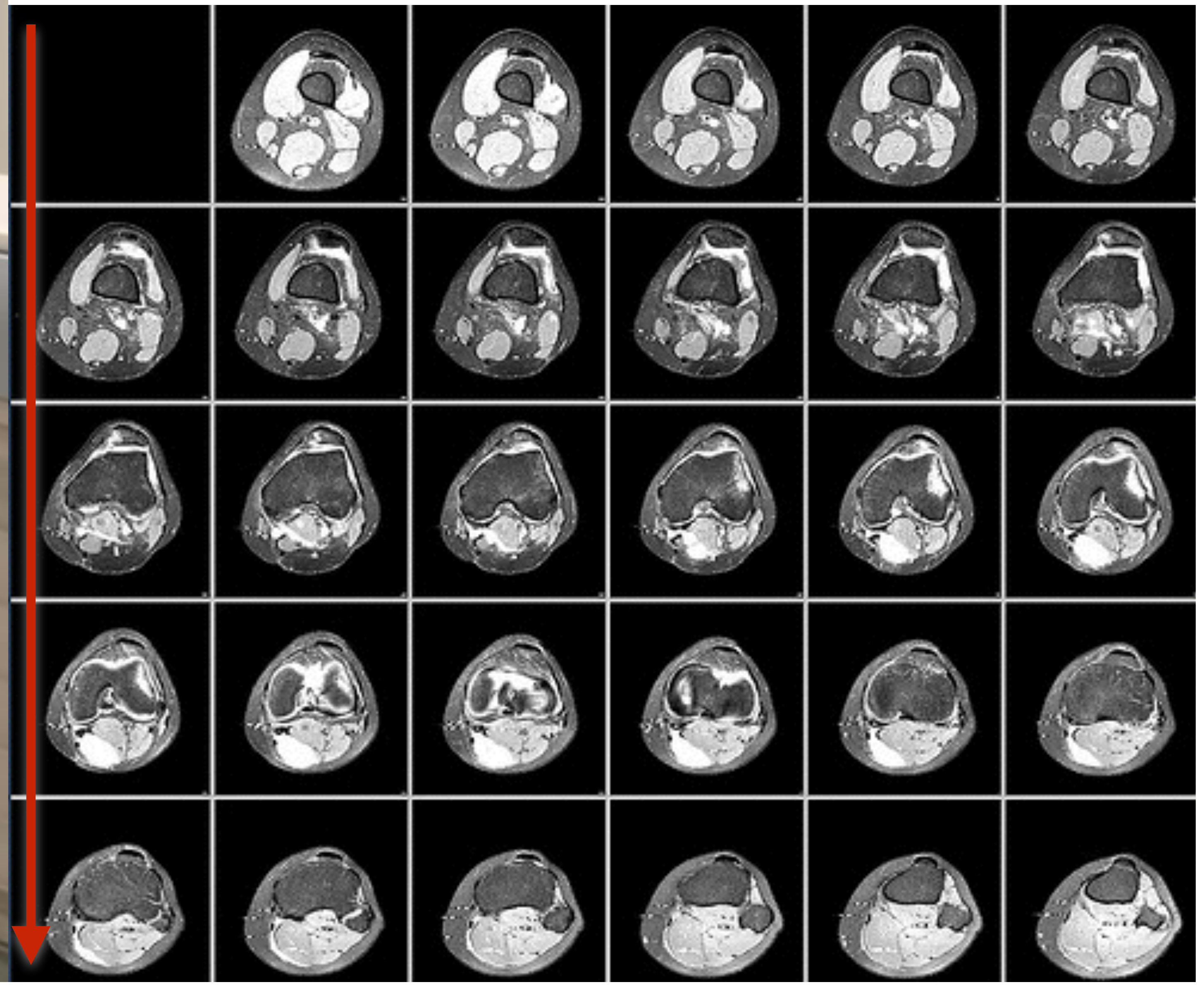


[Weiskopf, Machiraju, Möller]

- **Interpolation**: "how to show values between the sampled points in ways that do not mislead"

Spatial Data Example: MRI



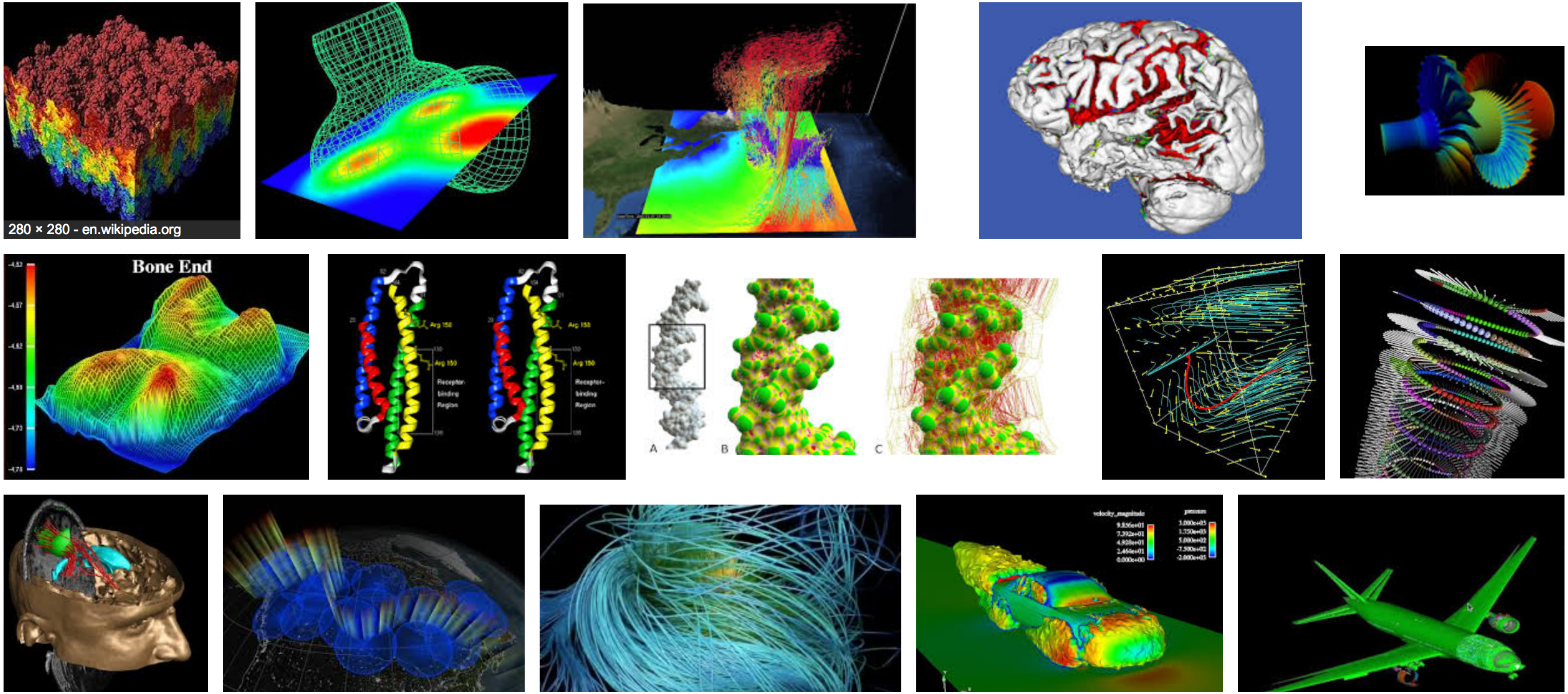


[via Levine, 2014]

Scivis and Infovis

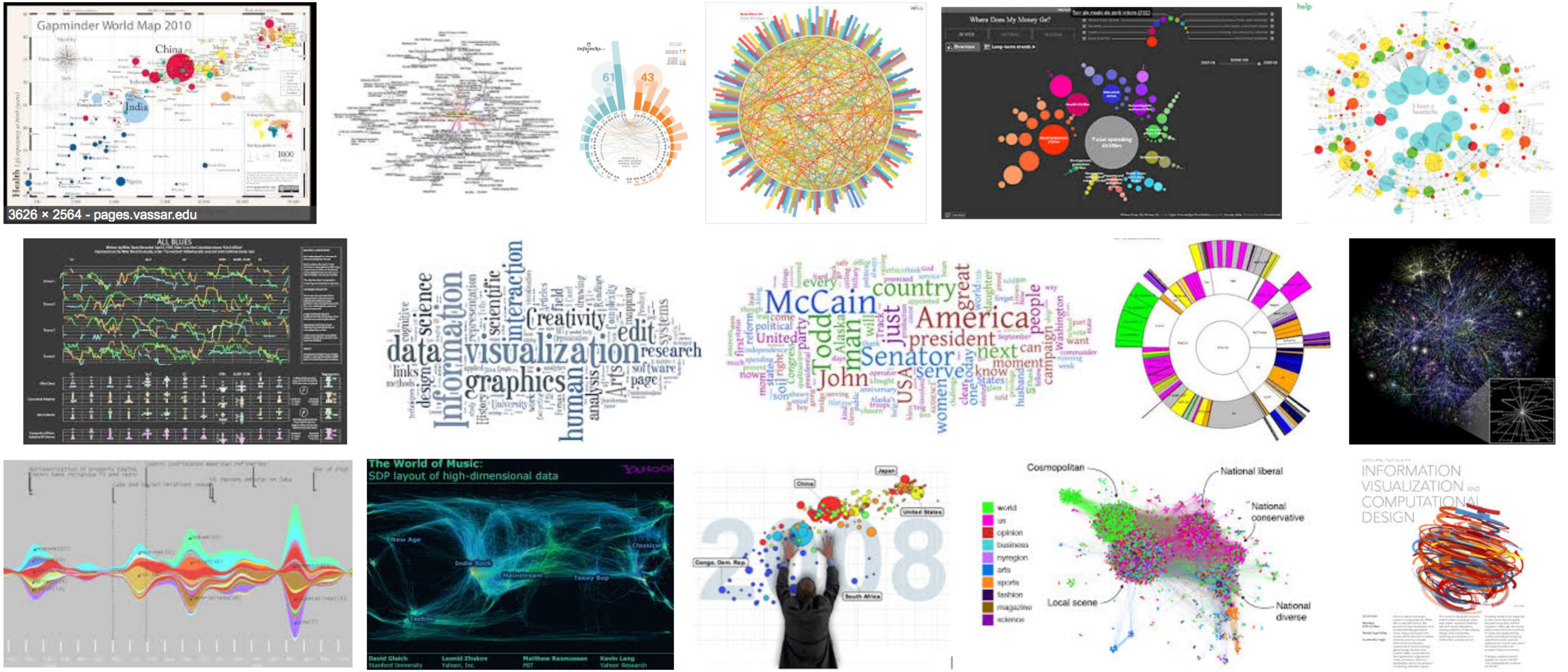
- Two subfields of visualization
- **Scivis** deals with data where the spatial position is given with data
  - Usually continuous data
  - Often displaying physical phenomena
  - Techniques like isosurfacing, volume rendering, vector field vis
- In **Infovis**, the data has no set spatial representation, designer chooses how to visually represent data

SciVis



[Google Image Search for "scientific visualization", 2017]

InfoVis



[Google Image Search for "information visualization", 2017]