Data Visualization (CSCI 627/490)

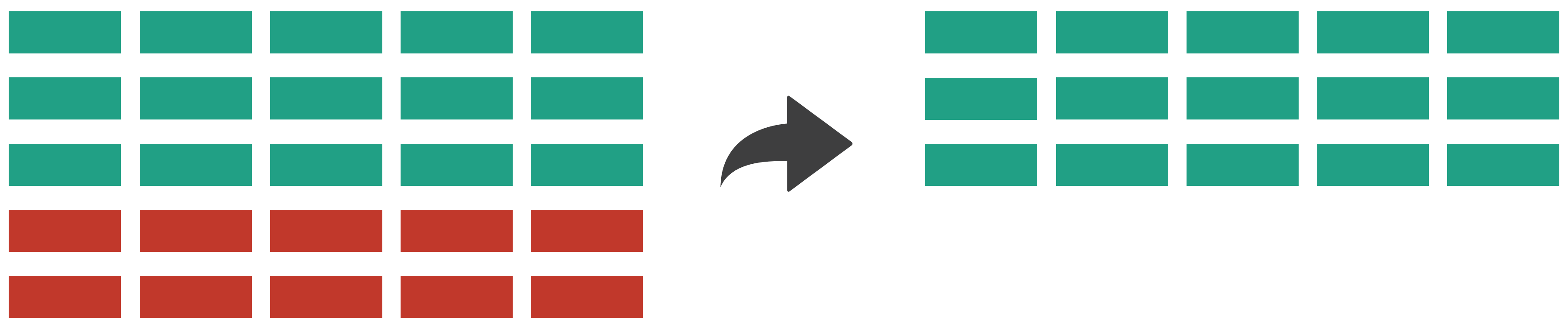
Data Manipulation & Isosurfacing

Dr. David Koop

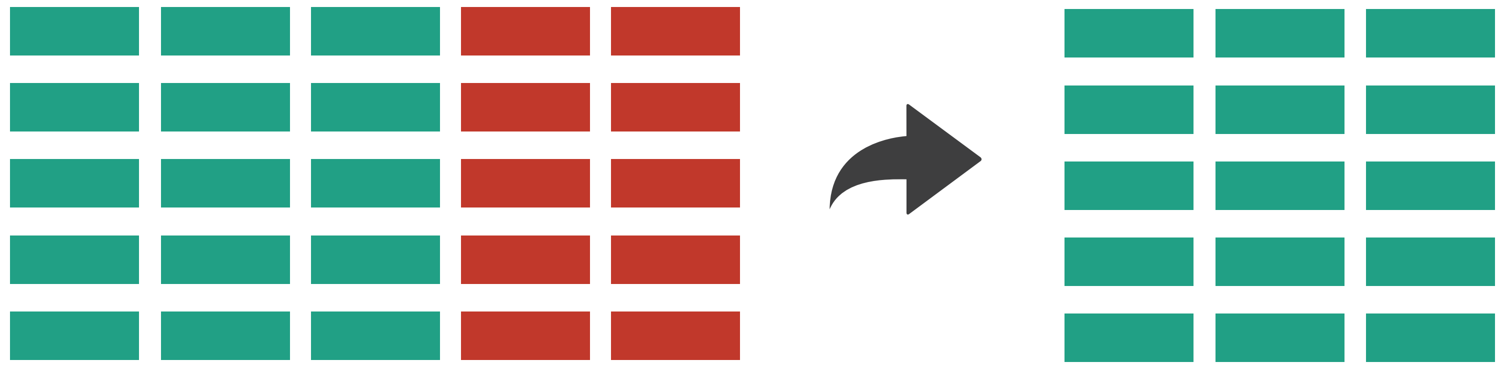
Overview: Reducing Items & Attributes

Filter

→ Items

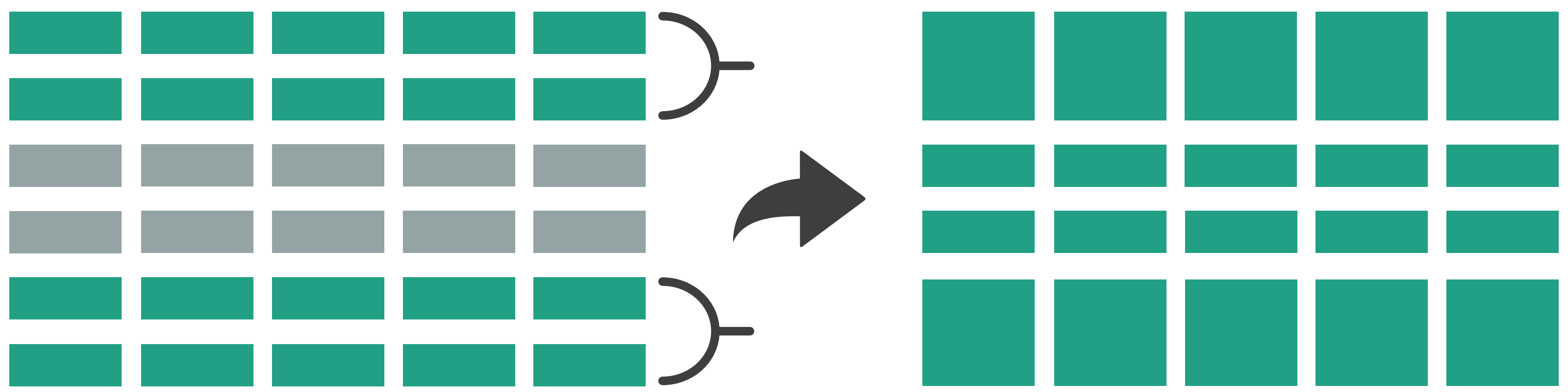


→ Attributes

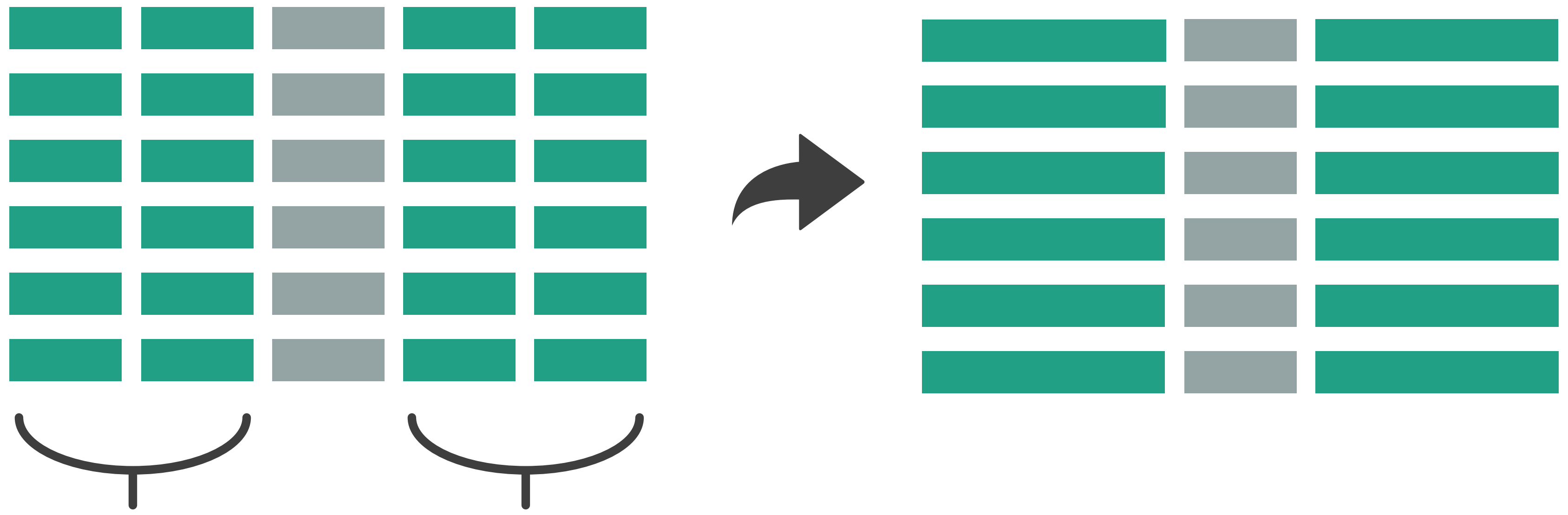


Aggregate

→ Items

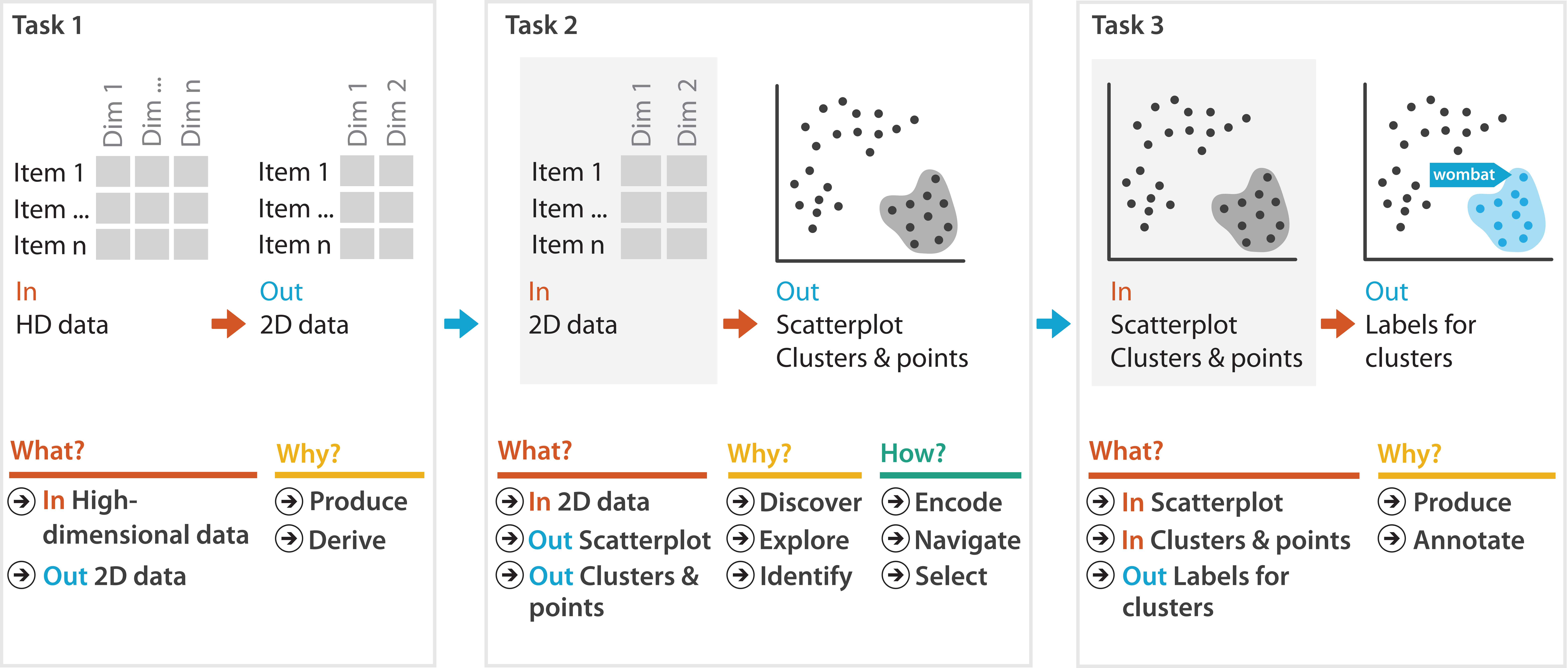


→ Attributes



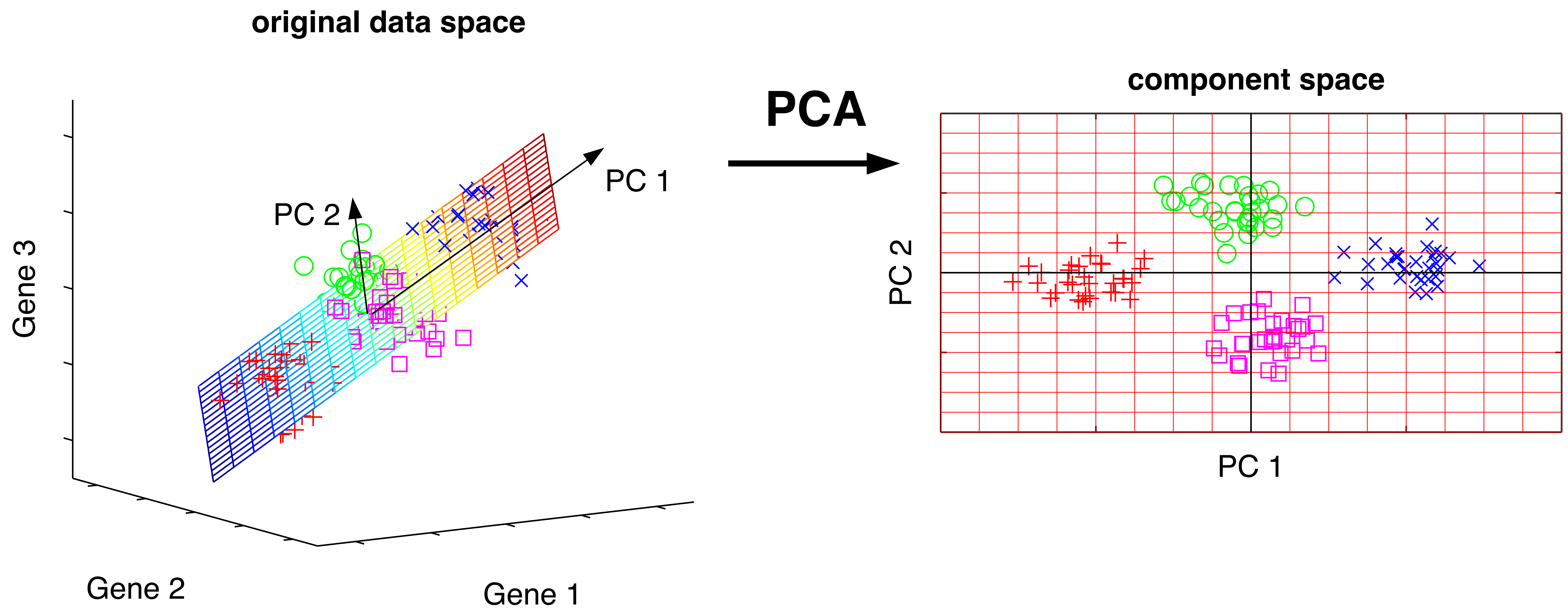
[Munzner (ill. Maguire), 2014]

Tasks in Understanding High-Dim. Data

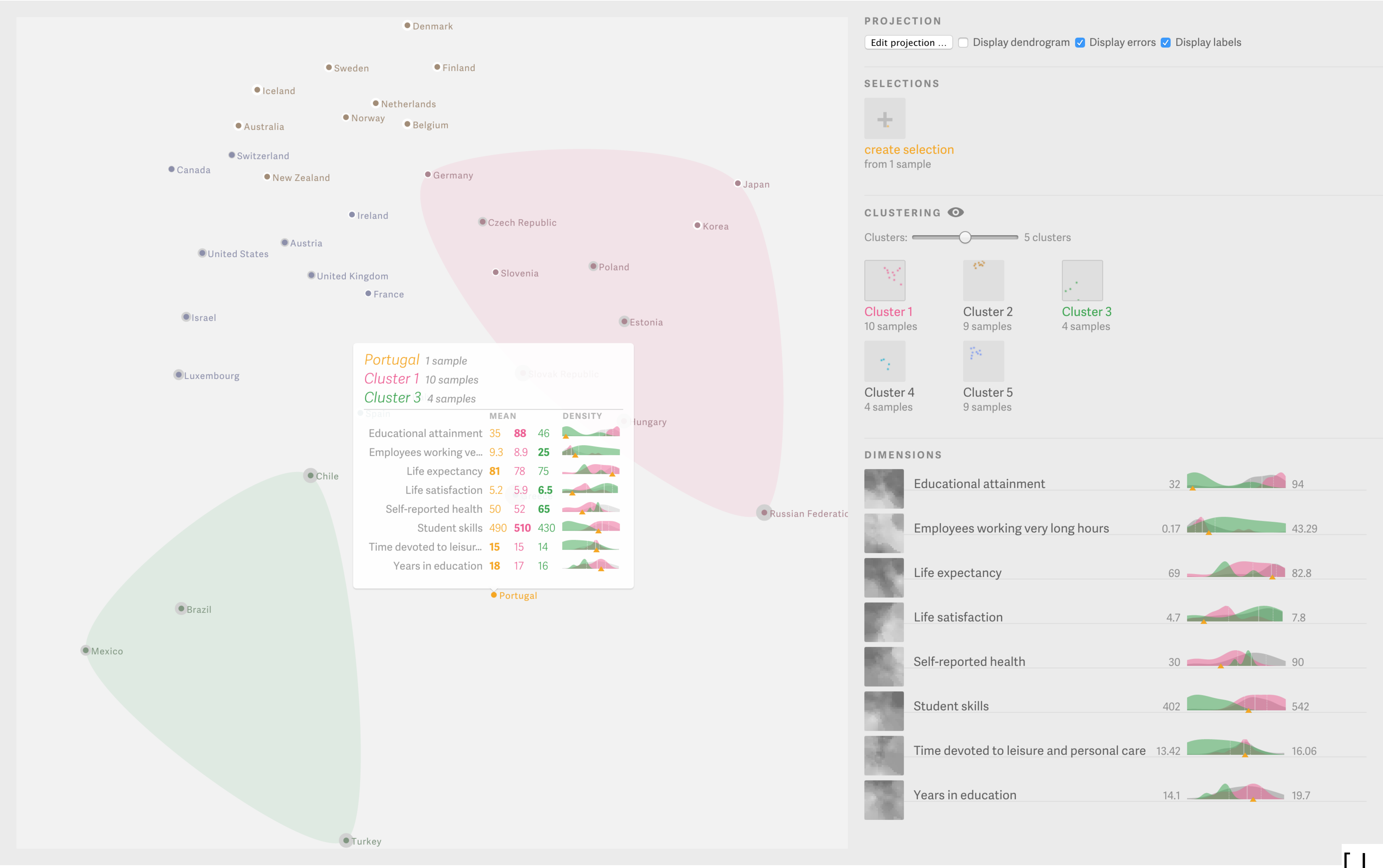


[Munzner (ill. Maguire), 2014]

Principle Component Analysis (PCA)

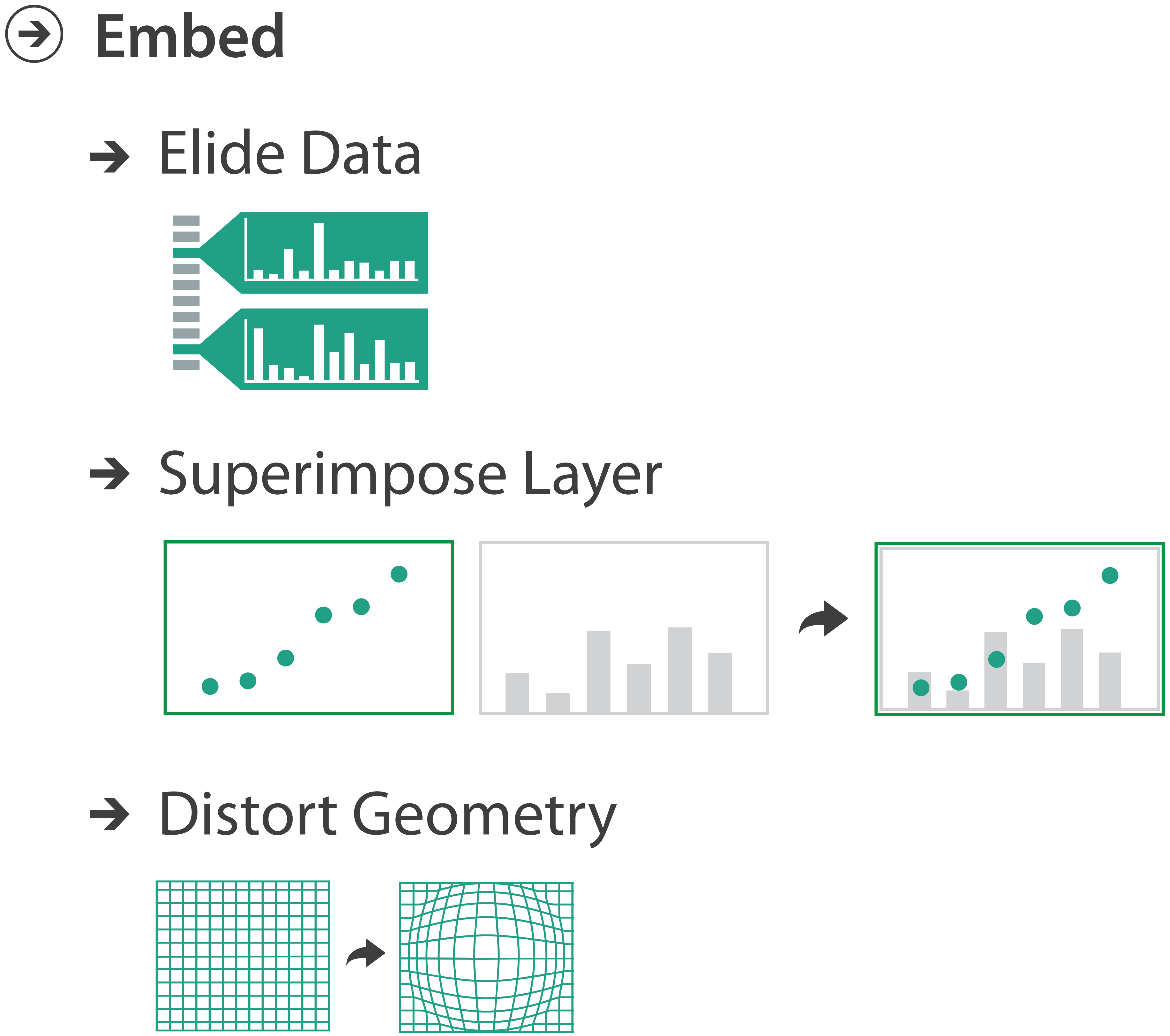


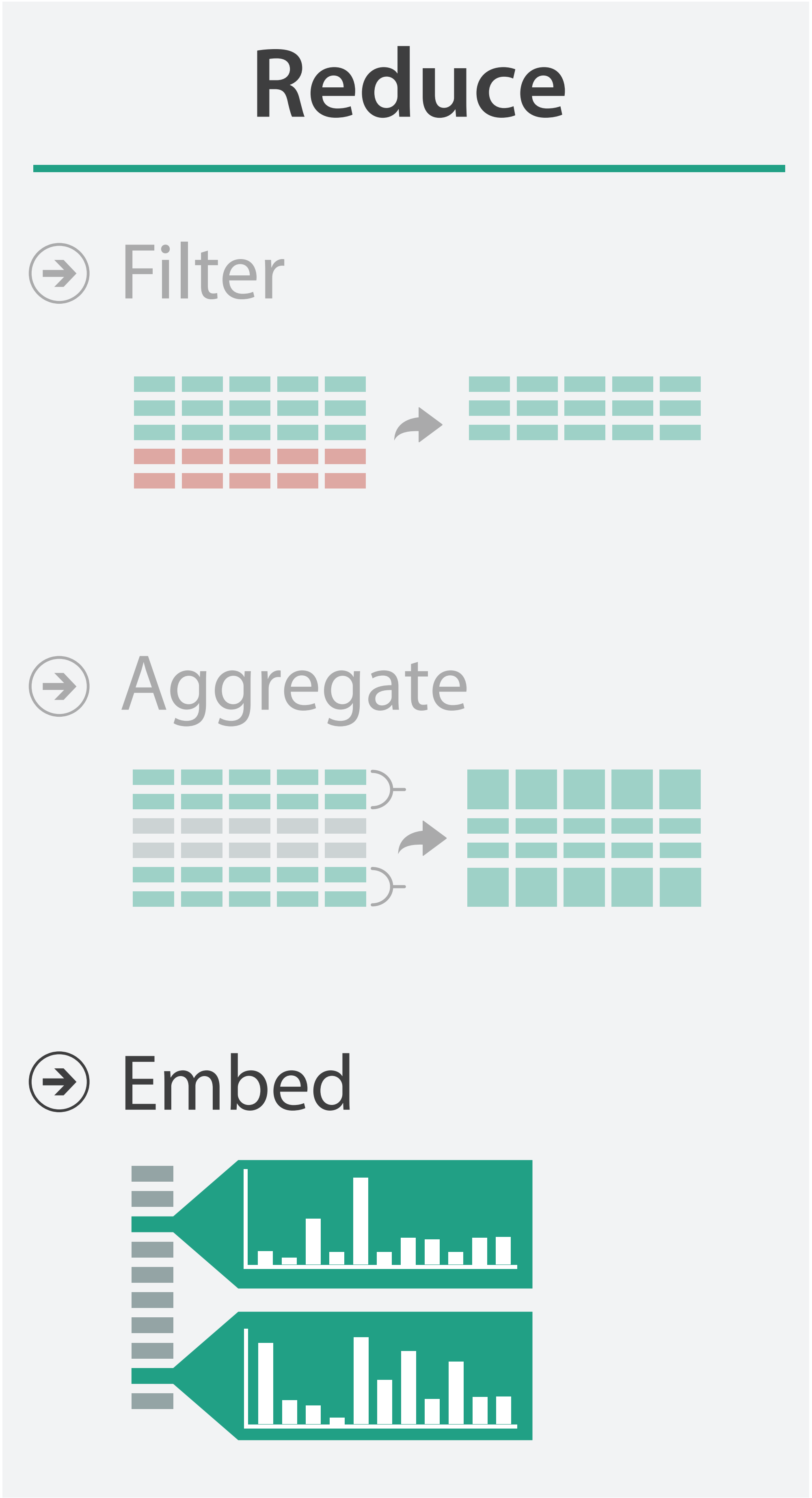
Probing Projections



[J. Stahnke et al., 2015]

Focus+Context Overview

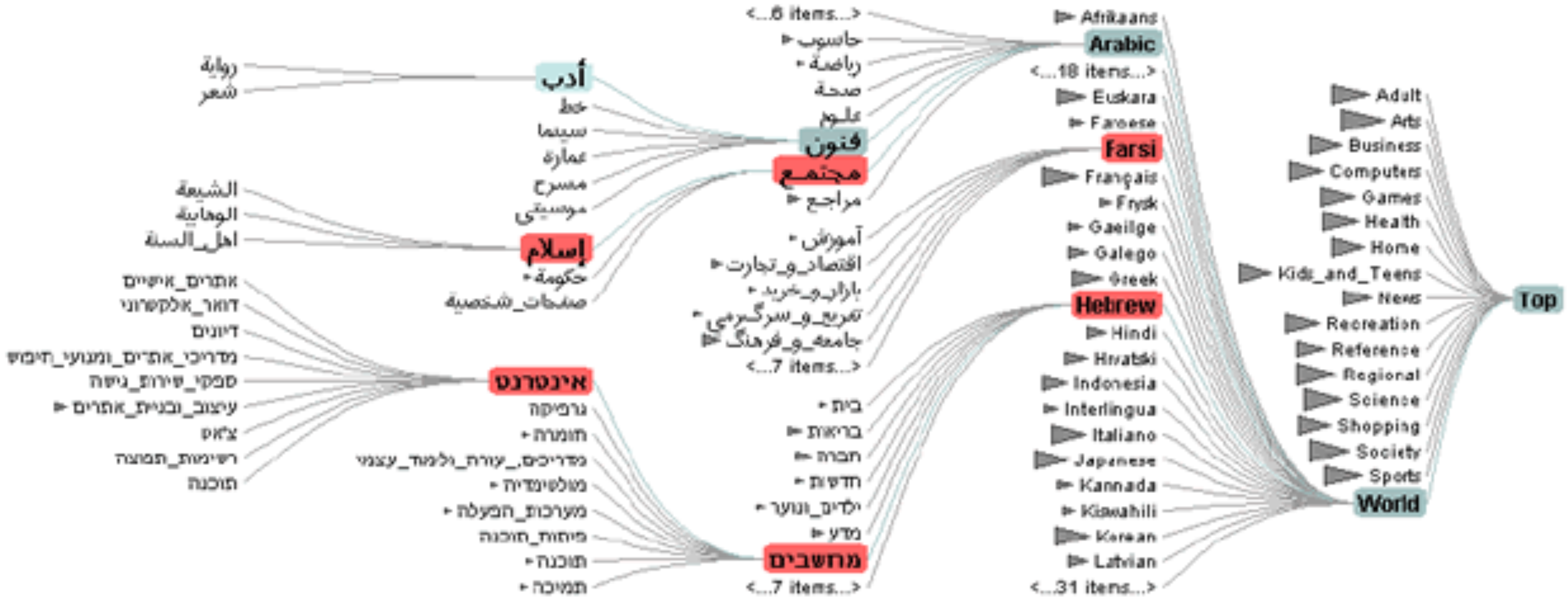




[Munzner (ill. Maguire), 2014]

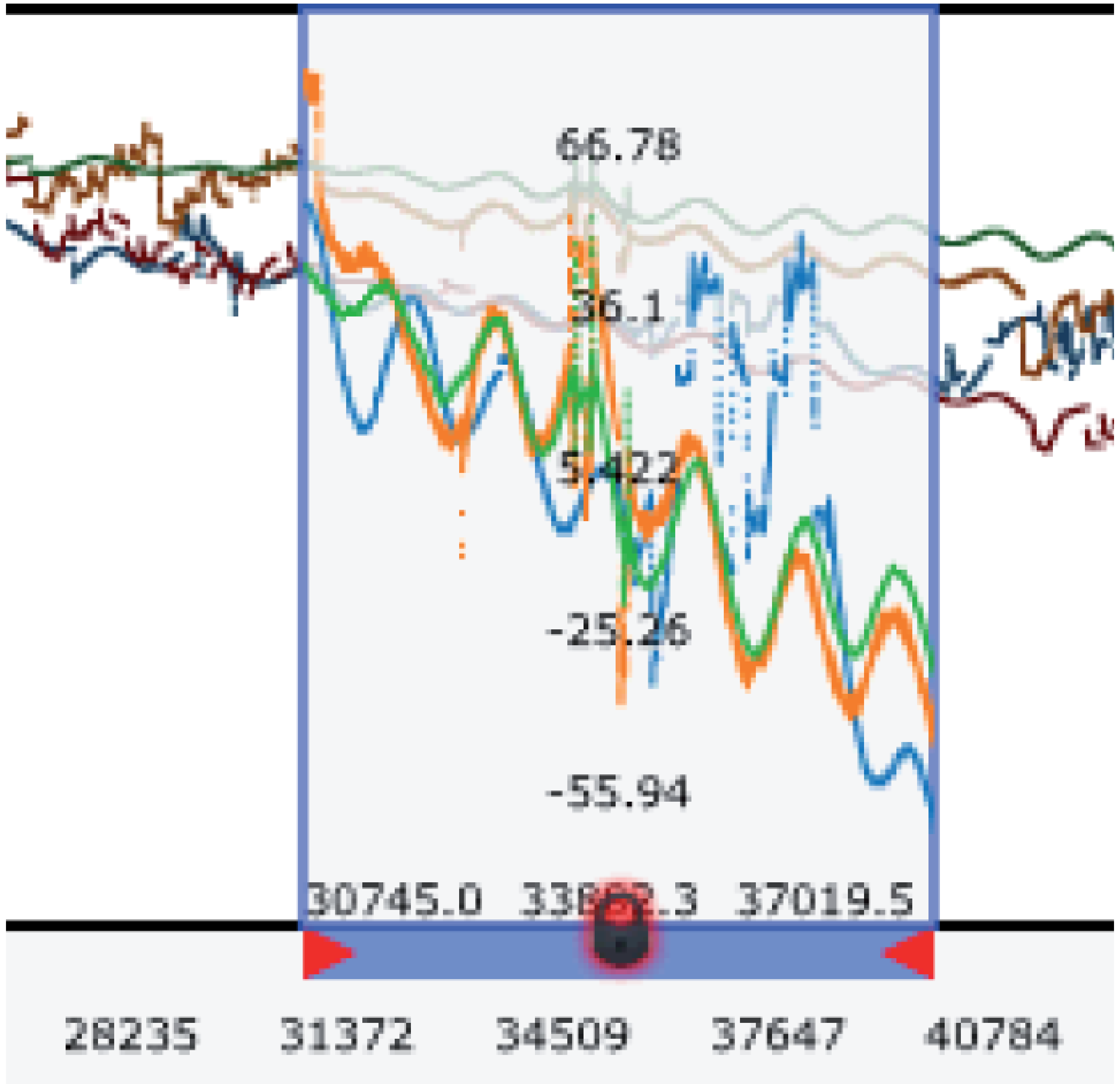
Elision & Degree of Interest Function

- DOI = I(x) - D(x,y)
  - I: interest function
  - D: distance (semantic or spatial)
  - x: location of item
  - y: current focus point
  - Interactive: y changes

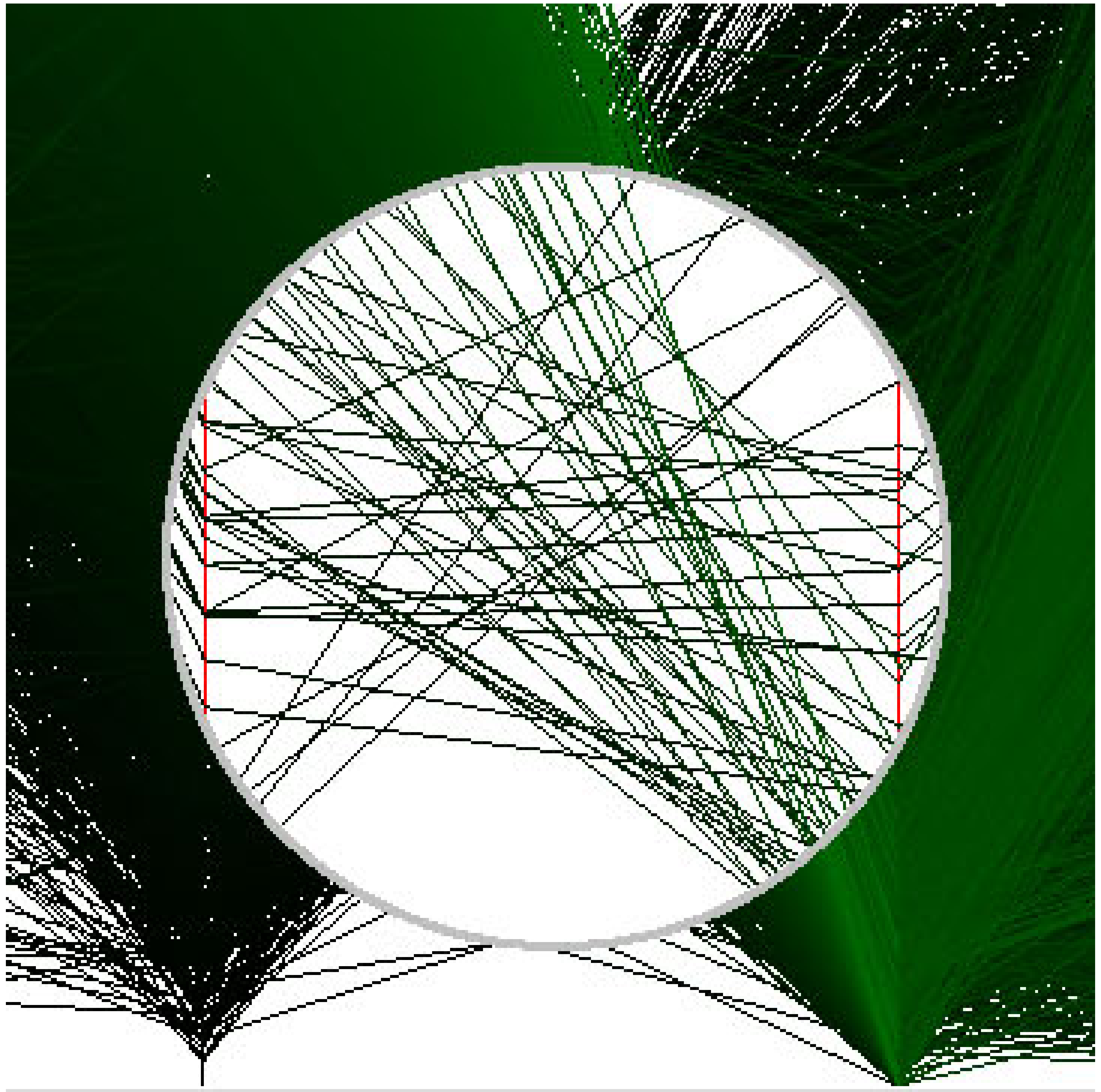


[Heer and Card, 2004]

Superimposition with Interactive Lenses



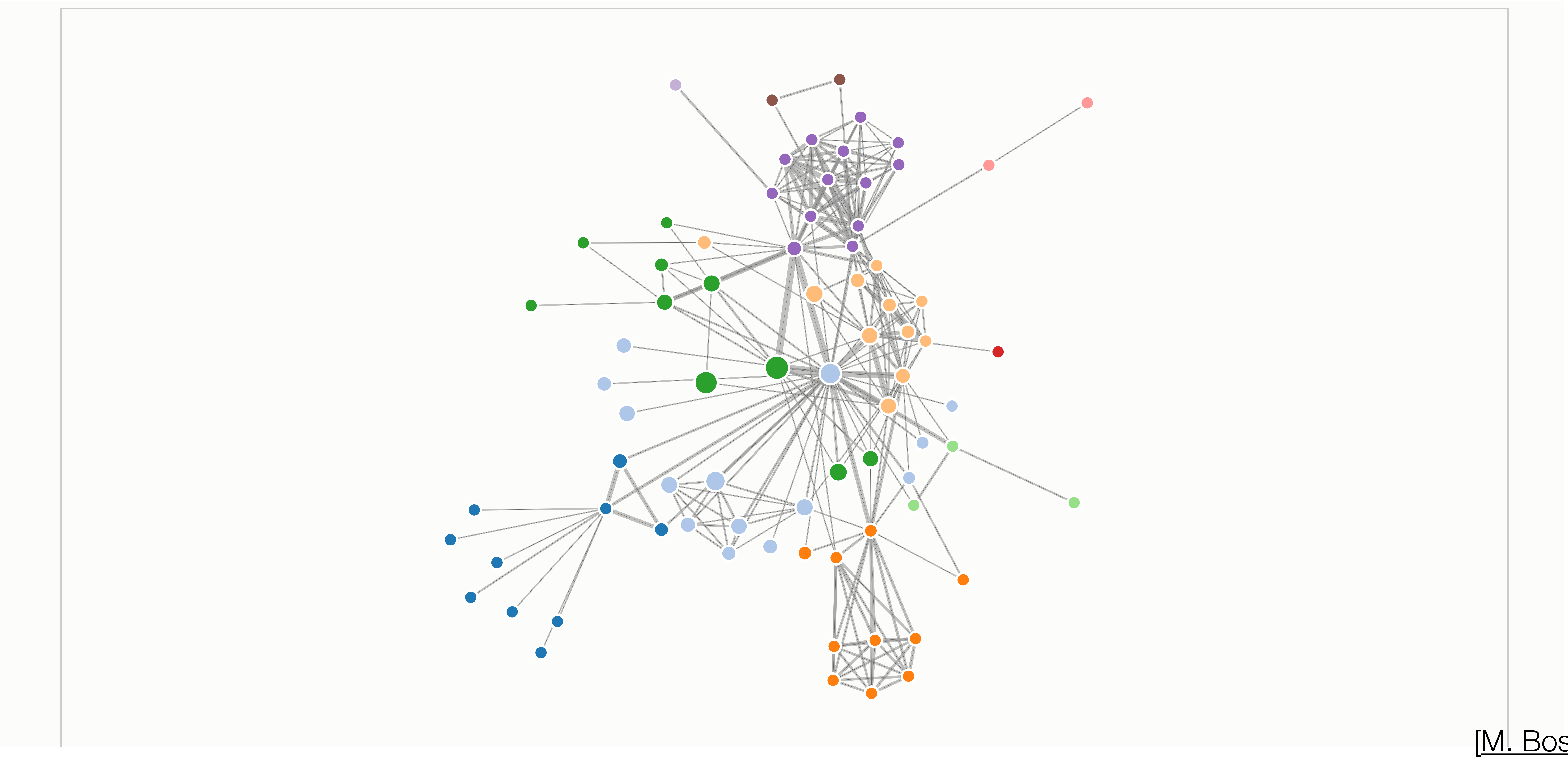
(a) Alteration



(b) Suppression

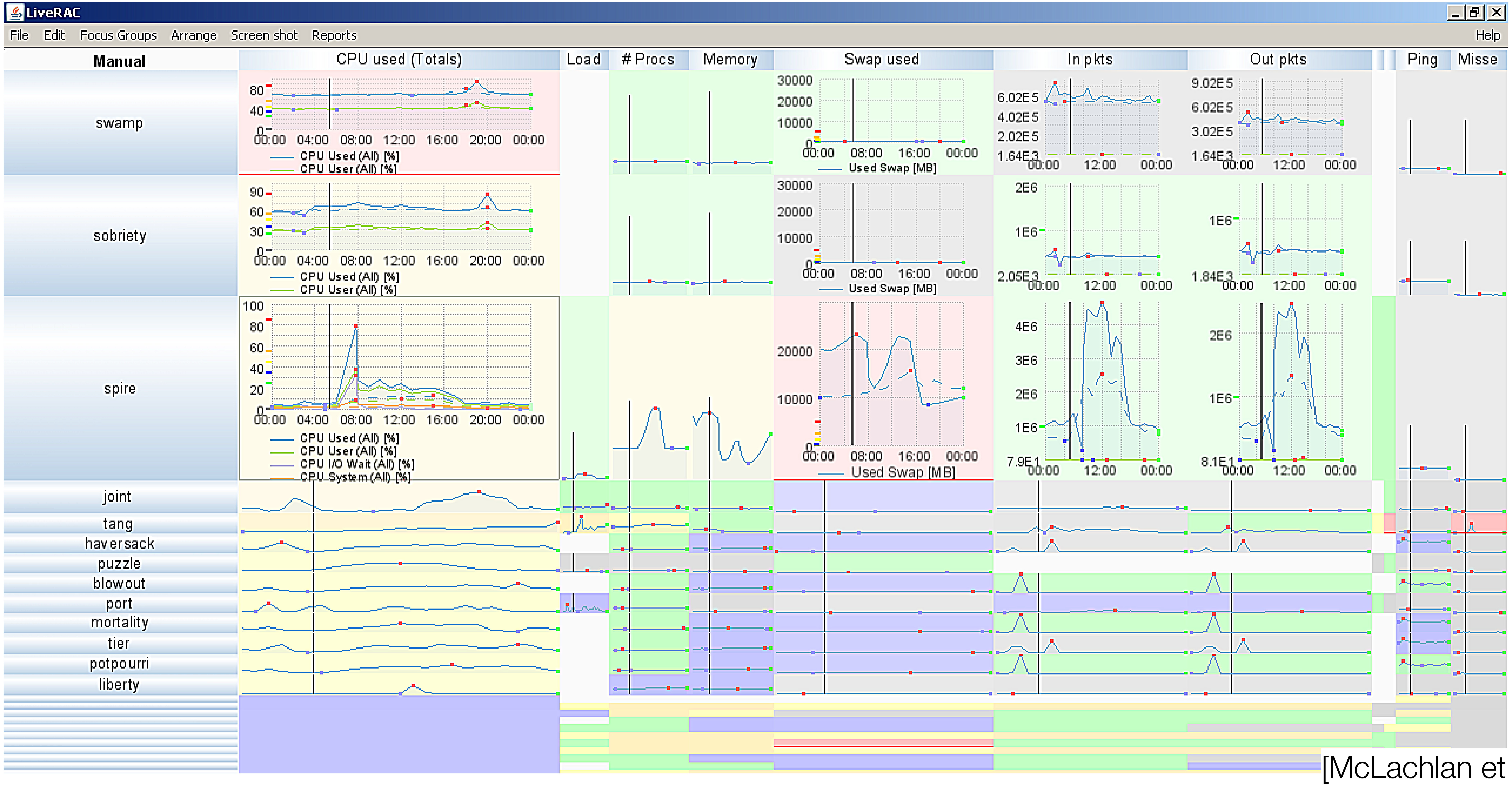
[ChronoLenses and Sampling Lens in Tominski et al., 2014]

Distortion



[M. Bostock]

Distortion: Stretch and Squish Navigation



[McLachlan et al., 2008]

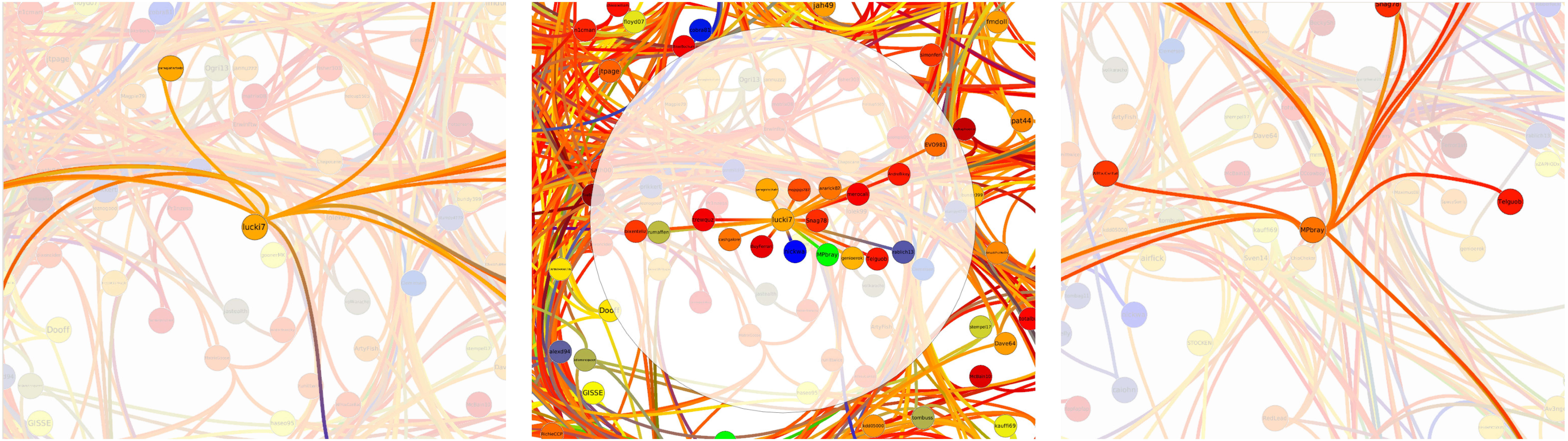
H3 Layout

[[T. Munzner](), 1998]

H3 Layout

[T. Munzner, 1998]

Focus+Context in Network Exploration



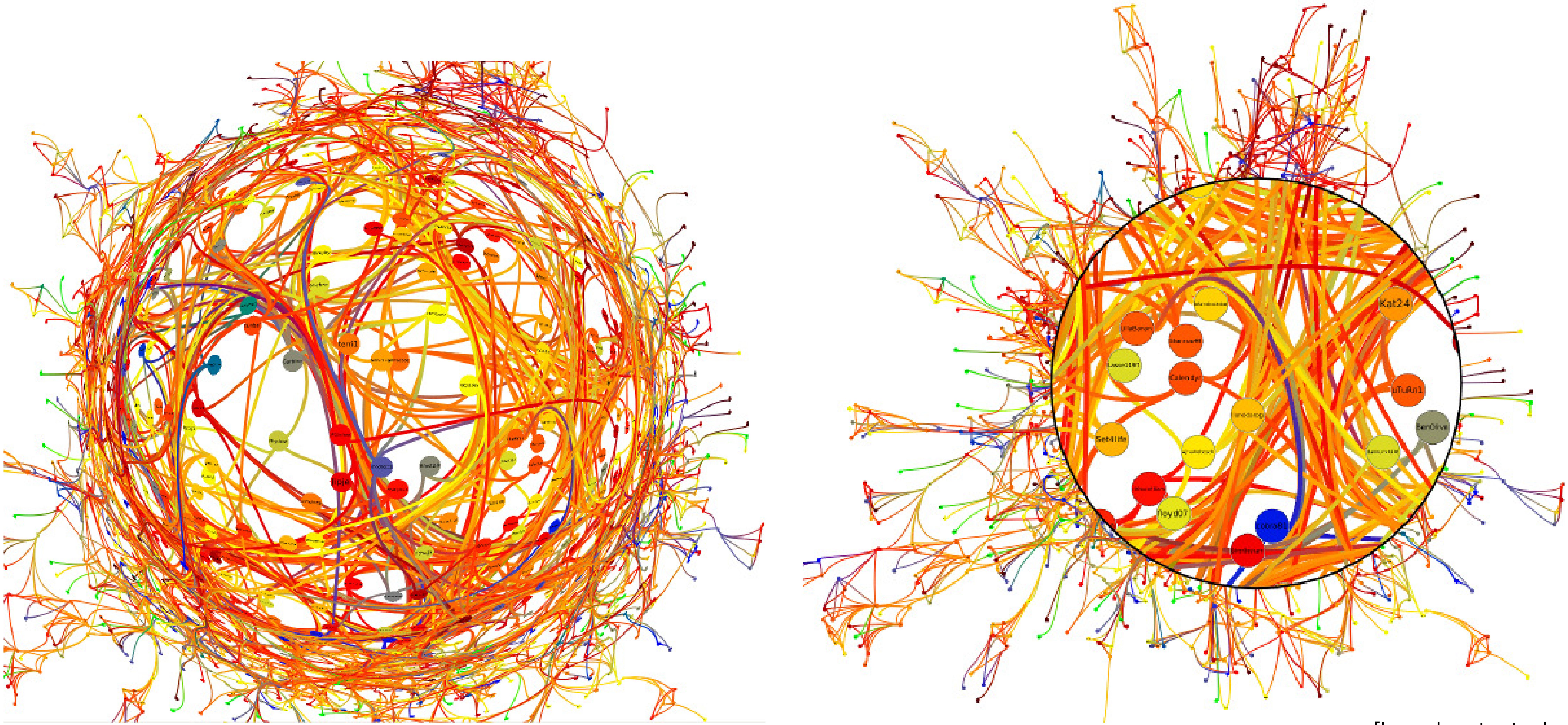
(a) Bring (step 1) – Selecting a node fades out all graph elements but the node neighborhood.

(b) Bring (step 2) – Neighbor nodes are pulled close to the selected node.

(c) Go – After selecting a neighbor (the green node in Fig. 4(b)), a short animation brings the focus towards a new neighborhood.

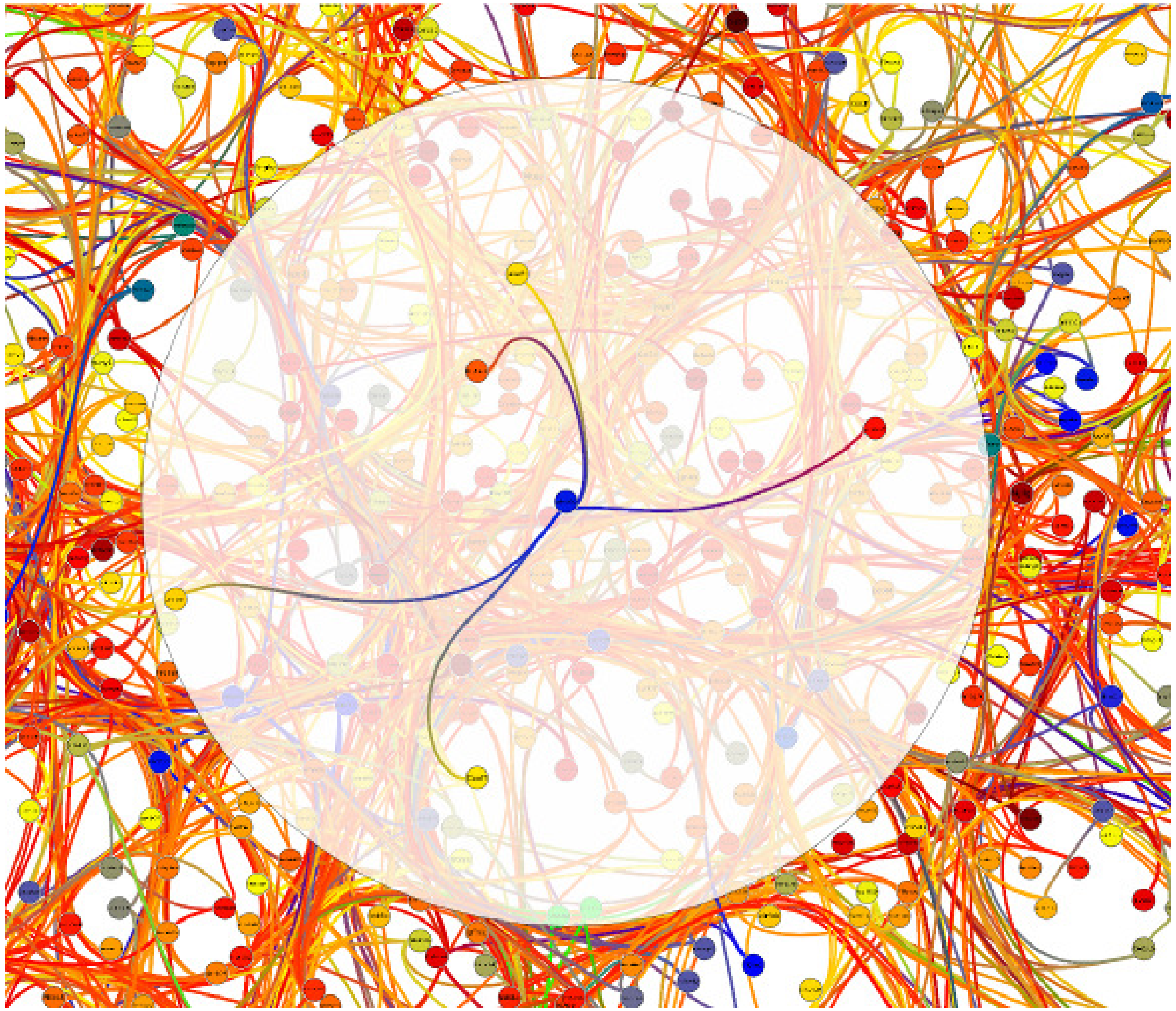
[Lambert et al., 2010]

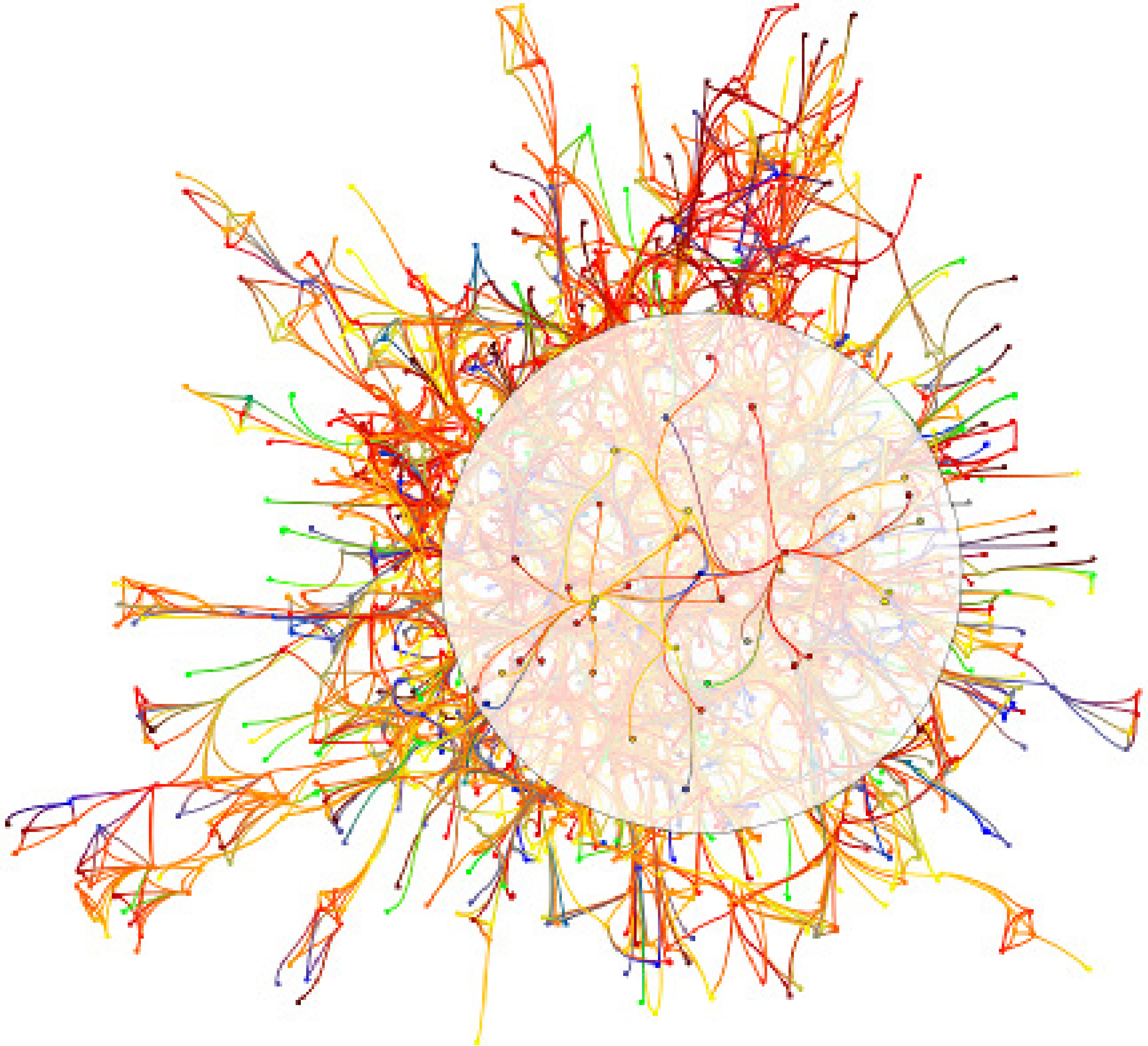
Focus+Context in Network Exploration



[Lambert et al., 2010]

Focus+Context in Network Exploration





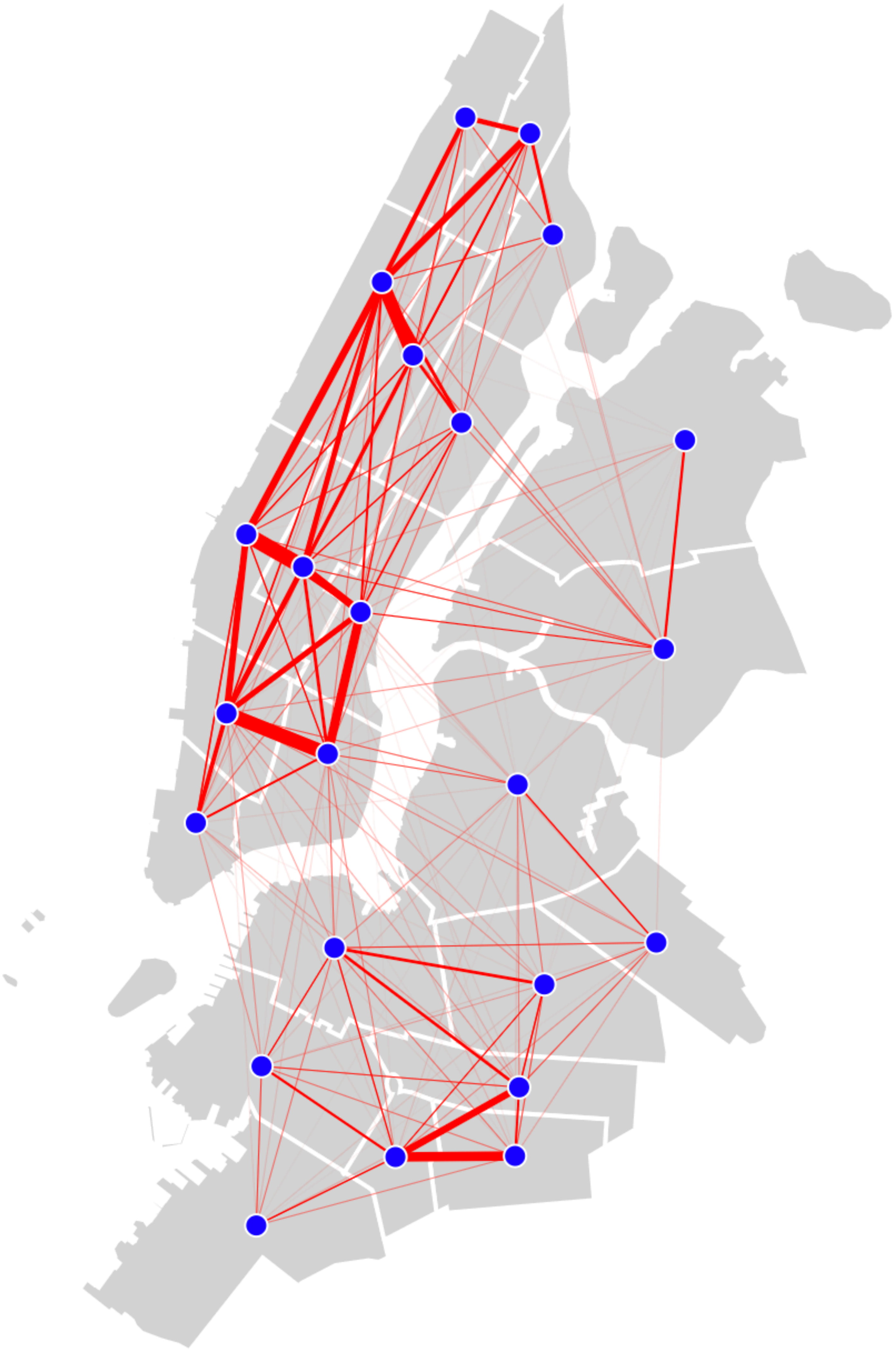
[Lambert et al., 2010]

Project Design

- Feedback:
  - Data Manipulation?
  - Questions lead, not technique!
  - Be creative! (interaction too) https://xeno.graphics
- Work on turning your visualization ideas into designs
- Turn in:
  - Two Design Sketches (like sheets 2-4 from 5 Sheet Design)
  - One Bad Design Sketch (like sheets 2-4: here, justify why bad)
  - Progress on Implementation
- Due Friday

Assignment 5

- Map of Citi Bike trips
  - Multiple Views
  - Linked Highlighting
  - Filtering
  - Aggregation
- Due Monday, Nov. 23



Data Wrangling

- Problem 1: Visualizations need data
- Solution: The Web!
- Problem 2: Data has extra information I don't need
- Solution: Filter it
- Problem 3: Data is dirty
- Solution: Clean it up
- Problem 4: Data isn't in the same place
- Solution: Combine data from different sources
- Problem 5: Data isn't structured correctly
- Solution: Reorder, map, and nest it

Hosting data

- github.com
- gist.github.com
- figshare.com
- myjson.com
- Other services

Cross-origin resource sharing (CORS)

- Restricts where data can be loaded from
- If developing locally, can
  - Run a web server locally (`python -m http.server` or npm's `http-server`)
  - Put the data on a website (like github), make sure to use **raw** URLs
- If loading JavaScript, this sometimes requires more help
  - https://gitcdn.xyz

Filtering Data

- Often useful to filter data before loading into D3

Why JavaScript?

- Python and R have great support for this sort of processing
- Data comes from the Web, want to put visualizations on the Web
- Sometimes unnecessary to download, process, and upload!
- More tools are helping JavaScript become a better language

JavaScript Data Wrangling Resources

- Latest version: https://observablehq.com/@berkeleyvis/learn-js-data
- My old version: https://observablehq.com/@dakoop/learn-js-data
- Based on http://learnjsdata.com/
- Good coverage of data wrangling using JavaScript

Comma Separated Values (CSV)

- File structure:

```
cities.csv:

city,state,population,land area
seattle,WA,652405,83.9
new york,NY,8405837,302.6
boston,MA,645966,48.3
kansas city,MO,467007,315.0
```

- Loading using D3:

```
d3.csv("/data/cities.csv").then(function(data) {
  console.log(data[0]);
});
```

- Result:

```
=> {city: "seattle", state: "WA", population: 652405, land area: 83.9}
```

- Values are strings! Convert to numbers via the unary `+` operator:
  - `d.population => "652405"`
  - `+d.population => 652405`

[http://learnjsdata.com]

Tab Separated Values (TSV)

- File structure:

```
animals.tsv:

name	type	avg_weight
tiger	mammal	260
hippo	mammal	3400
komodo dragon	reptile	150
```

- Loading using D3:

```
d3.tsv("/data/animals.tsv").then(function(data) {
  console.log(data[0]);
});
```

- Result:

```
=> {name: "tiger", type: "mammal", avg_weight: "260"}
```

- Can also have other delimiters (e.g. '|', ';')

[http://learnjsdata.com]

JavaScript Object Notation (JSON)

- File Structure:

```
employees.json:
[
 {"name":"Andy Hunt",
  "title":"Big Boss",
  "age": 68,
  "bonus": true
 },
 {"name":"Charles Mack",
  "title":"Jr Dev",
  "age":24,
  "bonus": false
 }
]
```

- Loading using D3:

```
d3.json("/data/employees.json".then(function(data) {
  console.log(data[0]);
});
```

- Result:

```
=> {name: "Andy Hunt", title: "Big Boss", age: 68, bonus: true}
```

[http://learnjsdata.com]

Loading Multiple Files

- Use Promise.all to load multiple files and then process them all

```
Promise.all([d3.csv("/data/cities.csv"),
             d3.tsv("/data/animals.tsv")])
  .then(analyze);

function analyze(data) {
  cities = data[0]; animals = data[1];

  console.log(cities[0]);
  console.log(animals[0]);
}
=> {city: "seattle", state: "WA", population: "652405", land area: "83.9"}
{name: "tiger", type: "mammal", avg_weight: "260"}
```

[http://learnjsdata.com]

Combining Data

- Suppose given products and brands
- Brands have an `id` and products have a `brand_id` that matches a brand
- Want to join these two datasets together
  - `Product.brand_id` => `Brand.id`
- Use a nested `forEach`/`filter`
- Use a native join command

[http://learnjsdata.com]

Summarizing Data

- d3 has min, max, and extent functions of the form
  - 1st argument: dataset
  - 2nd argument: accessor function
- Example:

```
var landExtent = d3.extent(data, function(d) { return d.land_area; });
console.log(landExtent);
=> [48.3, 315]
```

- Summary statistics, e.g. mean, median, deviation → same format
- Median Example:

```
var landMed = d3.median(data, function(d) { return d.land_area; });
console.log(landMed);
=> 193.25
```

[http://learnjsdata.com]

Grouping Data

- Take a flat structure and turn it into a (potentially nested) map
- Similar to a groupby in databases
- Data

```
var expenses = [{"name":"jim","amount":34,"date":"11/12/2015"},
  {"name":"carl","amount":120.11,"date":"11/12/2015"},
  {"name":"jim","amount":45,"date":"12/01/2015"},
  {"name":"stacy","amount":12.00,"date":"01/04/2016"},
  {"name":"stacy","amount":34.10,"date":"01/04/2016"},
  {"name":"stacy","amount":44.80,"date":"01/05/2016"}
];
```

- Grouping:

```
expensesByName = d3.group(expenses, d => d.name)
```

- Results:

```
Map(3) { "jim" => Array(2) [Object, Object]
         "carl" => Array(1) [Object]
         "stacy" => Array(3) [Object, Object, Object] }
```

Rollup Data

- Data

```
var expenses = [{"name":"jim","amount":34,"date":"11/12/2015"},
  {"name":"carl","amount":120.11,"date":"11/12/2015"},
  {"name":"jim","amount":45,"date":"12/01/2015"},
  {"name":"stacy","amount":12.00,"date":"01/04/2016"},
  {"name":"stacy","amount":34.10,"date":"01/04/2016"},
  {"name":"stacy","amount":44.80,"date":"01/05/2016"}
];
```

- Using d3.rollup:

```
expensesAvgAmount = d3.rollup(
  expenses,
  v => d3.mean(v, d => d.amount), // aggregate by the mean of amount
  d => d.name // group by name
)
```

the aggregation function (difference from group)

- Result:

```
Map(3) {
  "jim" => 39.5
  "carl" => 120.11
  "stacy" => 30.3
}
```

[http://learnjsdata.com]

groups and rollups

- Both group and rollup return Map objects
- groups and rollups are the same functions but return nested arrays
- More examples: https://observablehq.com/@d3/d3-group

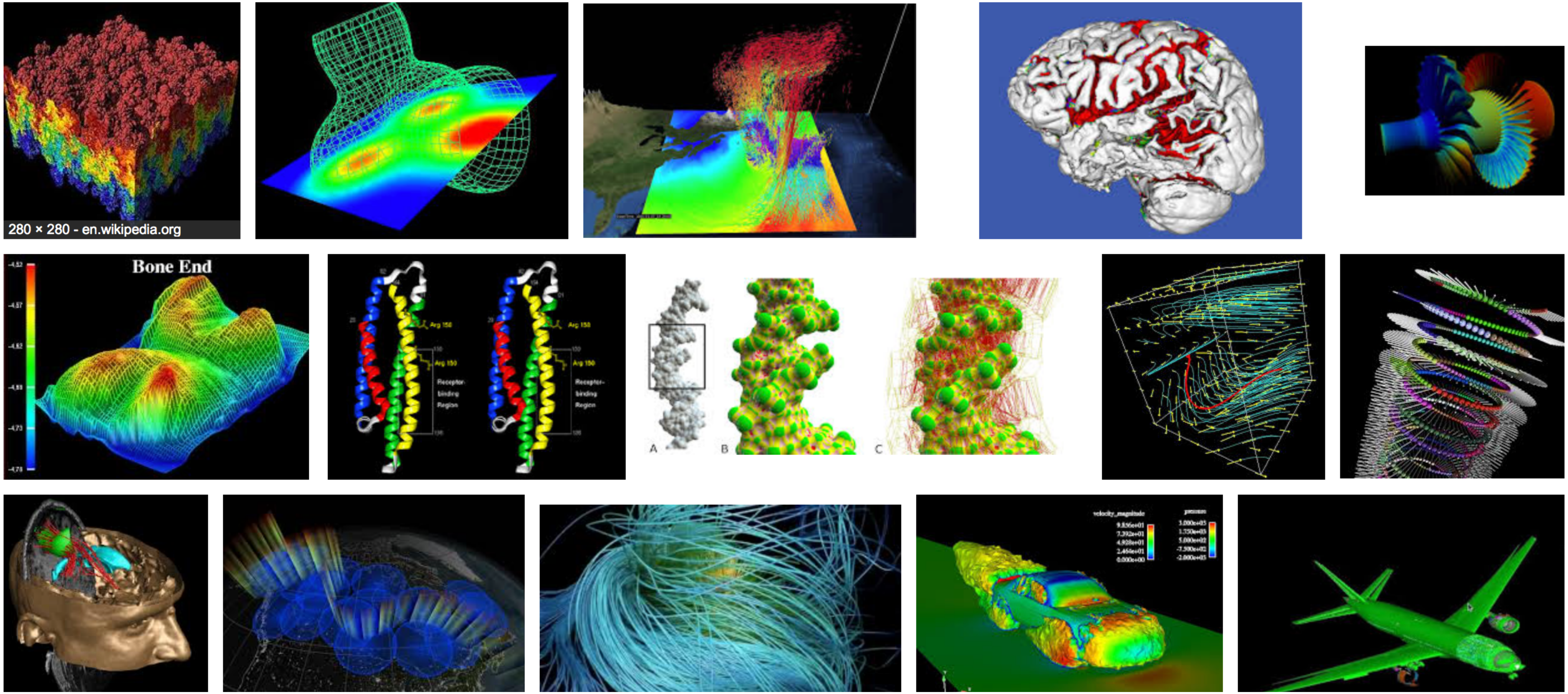
arquero

- New library for query processing and transformation of array-backed data tables:
- https://observablehq.com/@uwdata/arquero?collection=@uwdata/arquero

Scivis and Infovis

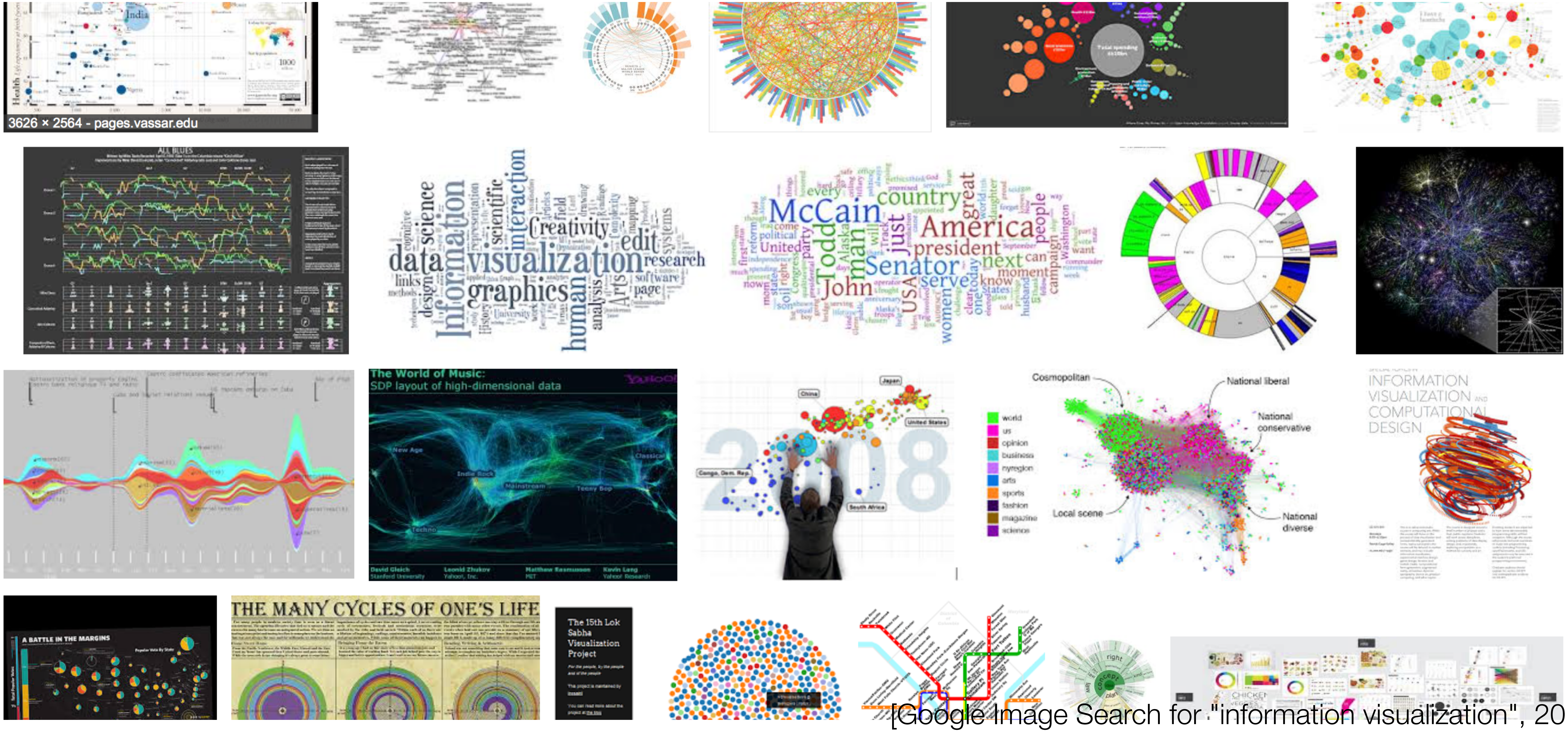
- Two subfields of visualization
- **Scivis** deals with data where the spatial position is given with data
  - Usually continuous data
  - Often displaying physical phenomena
  - Techniques like isosurfacing, volume rendering, vector field vis
- In **Infovis**, the data has no set spatial representation,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

SciVis



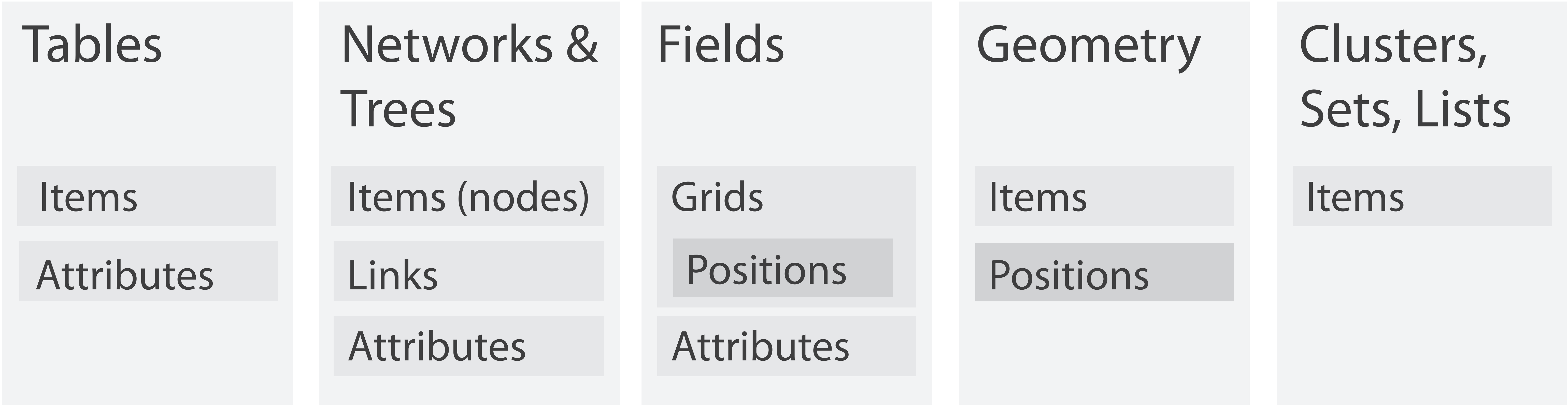
[Google Image Search for "scientific visualization", 2017]

InfoVis



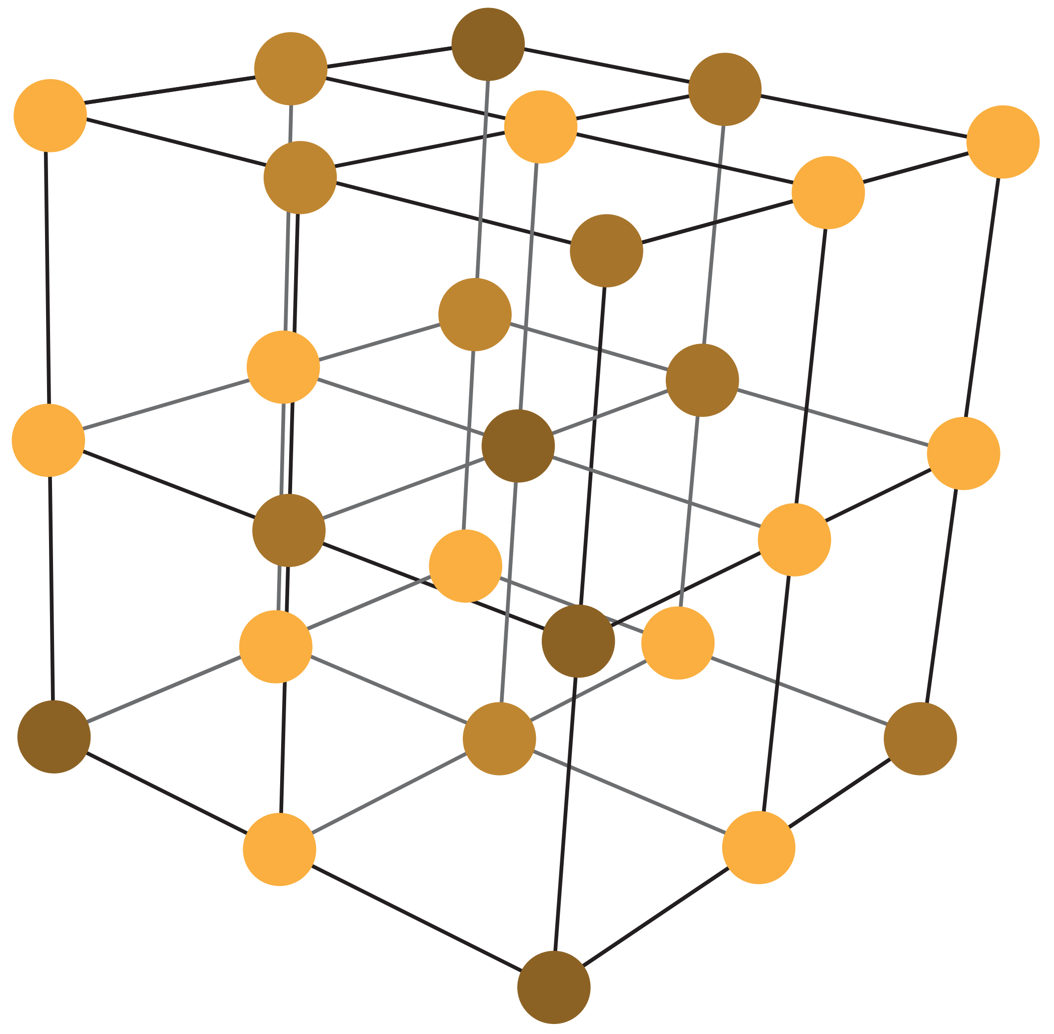
[Google Image Search for "information visualization", 2017]

Fields



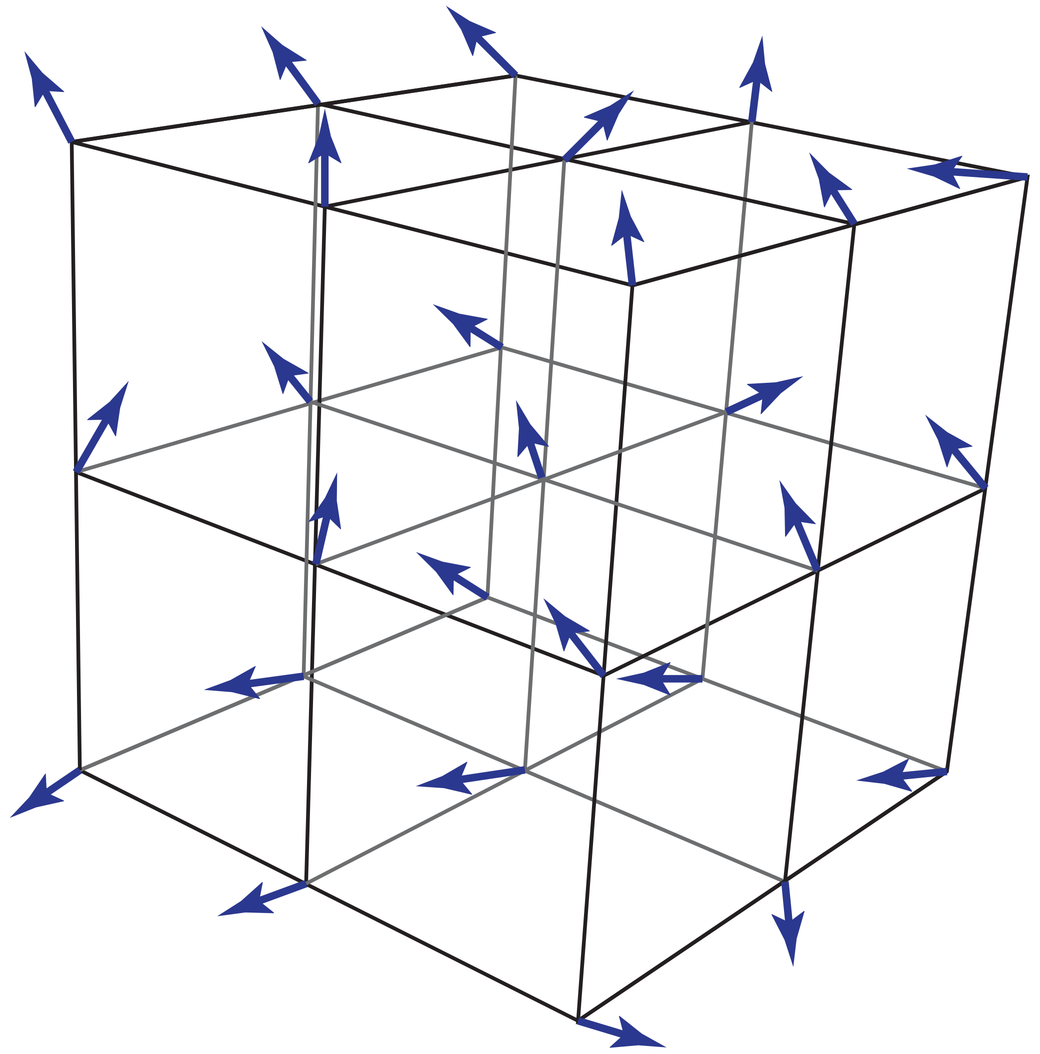
- Values come from a **continuous** domain, infinitely many values
- **Sampled** at certain positions to approximate the entire domain
- Positions are often aligned in **grids**
- Often measurements of natural or simulated phenomena
- Examples: temperature, wind speed, tissue density, pressure, speed, electrical conductance

Fields in Visualization



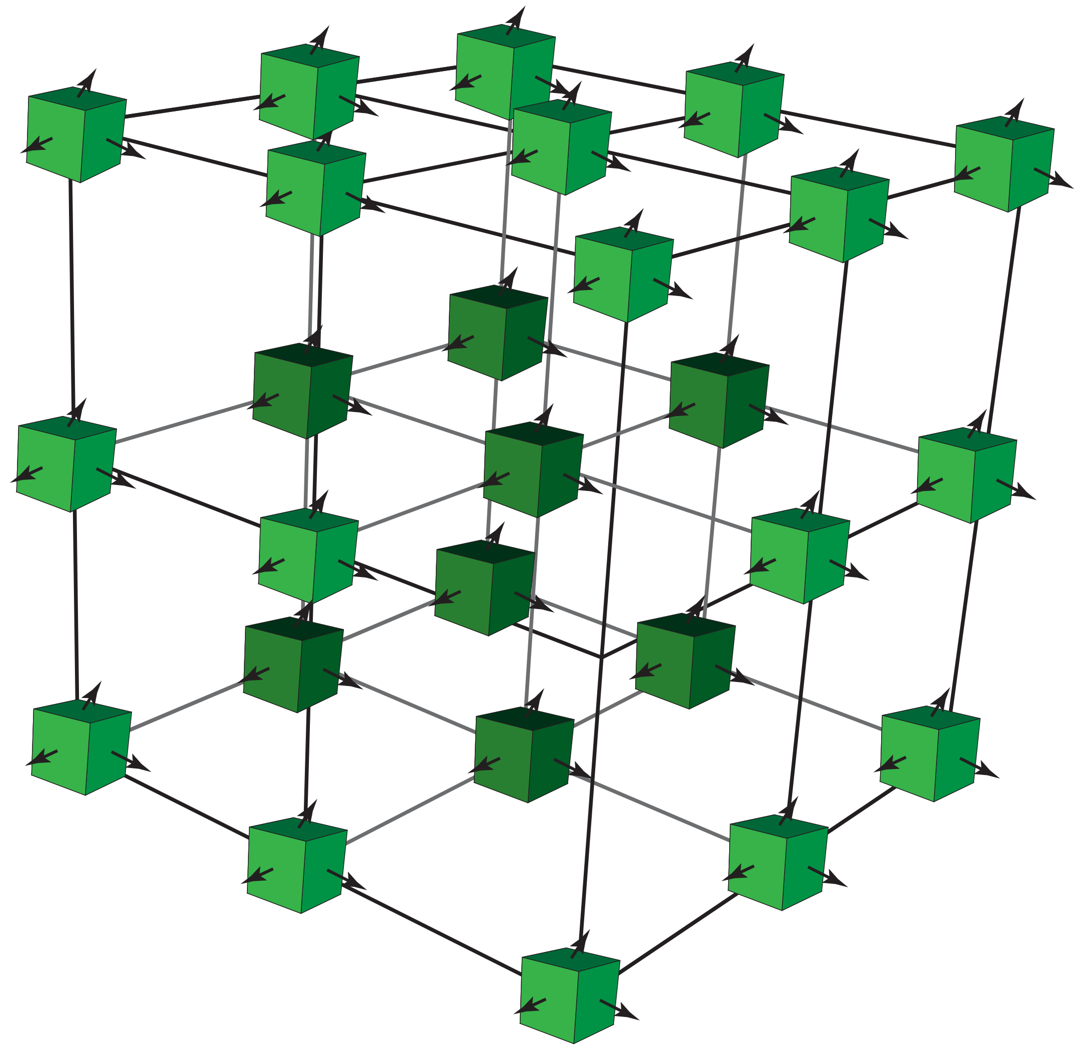
Scalar Fields

(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