Information Visualization

SWE 432, Fall 2018

Design and Implementation of Software for the Web

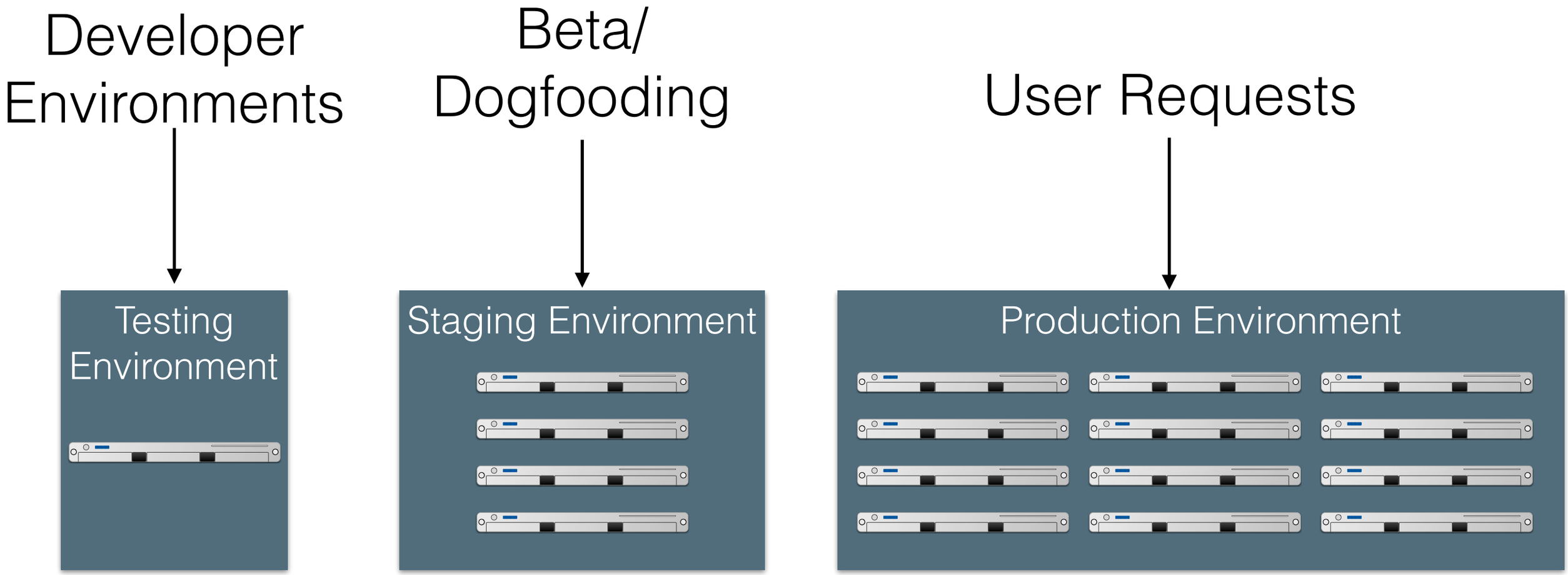
Today

- What is an information visualization?
- Lots of examples of visualizations
- Patterns and anti-patterns for visualization
- How can you build an information visualization d3.js?

Review: What is a software process?

- A structured set of activities required to develop a software product
  - Specification
  - Design and implementation
  - Validation
  - Evolution (operation and maintenance)
- Goal: Minimize Risk
  - Falling behind schedule
  - Changes to requirements
  - Bugs/unintended effects of changes

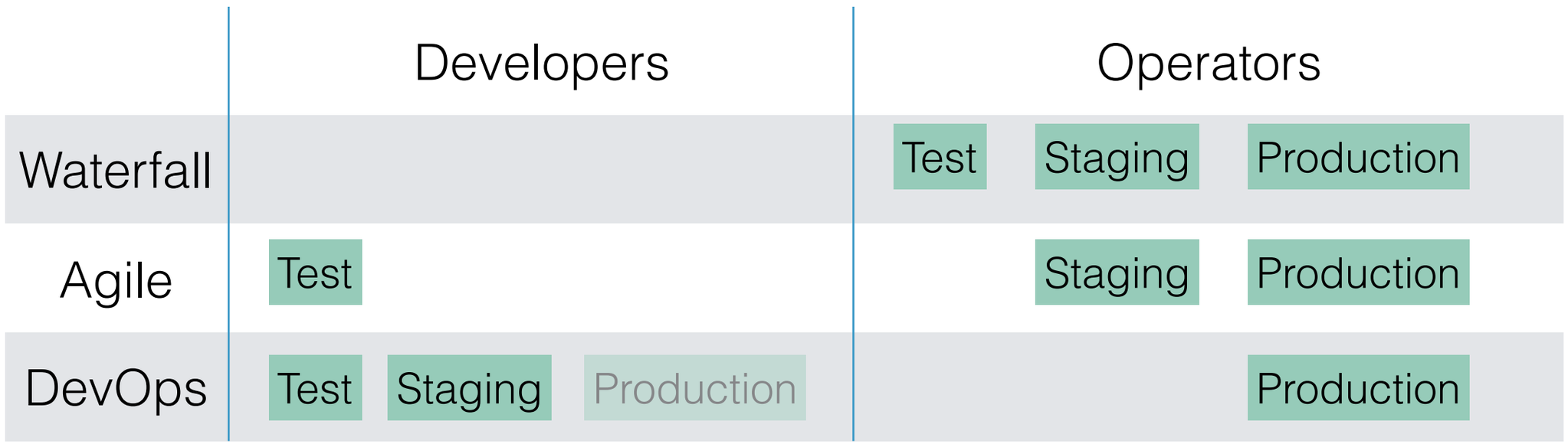
Review: Test-Stage-Production



Revisions are “promoted” towards production

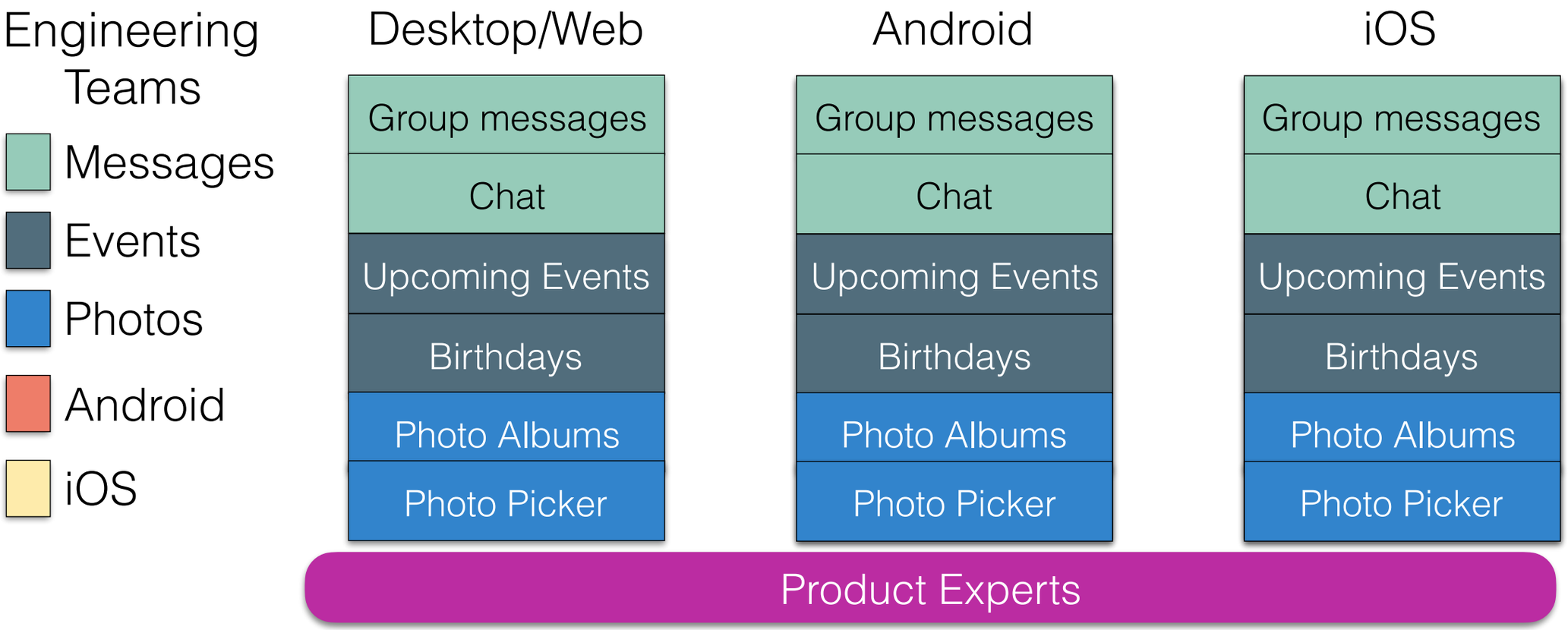
Review: Operations Responsibility

- Once we **deploy**, someone has to monitor software, make sure it's running OK, no bugs, etc
- Assume 3 environments:
  - Test, Staging, Production
- Whose job is it?

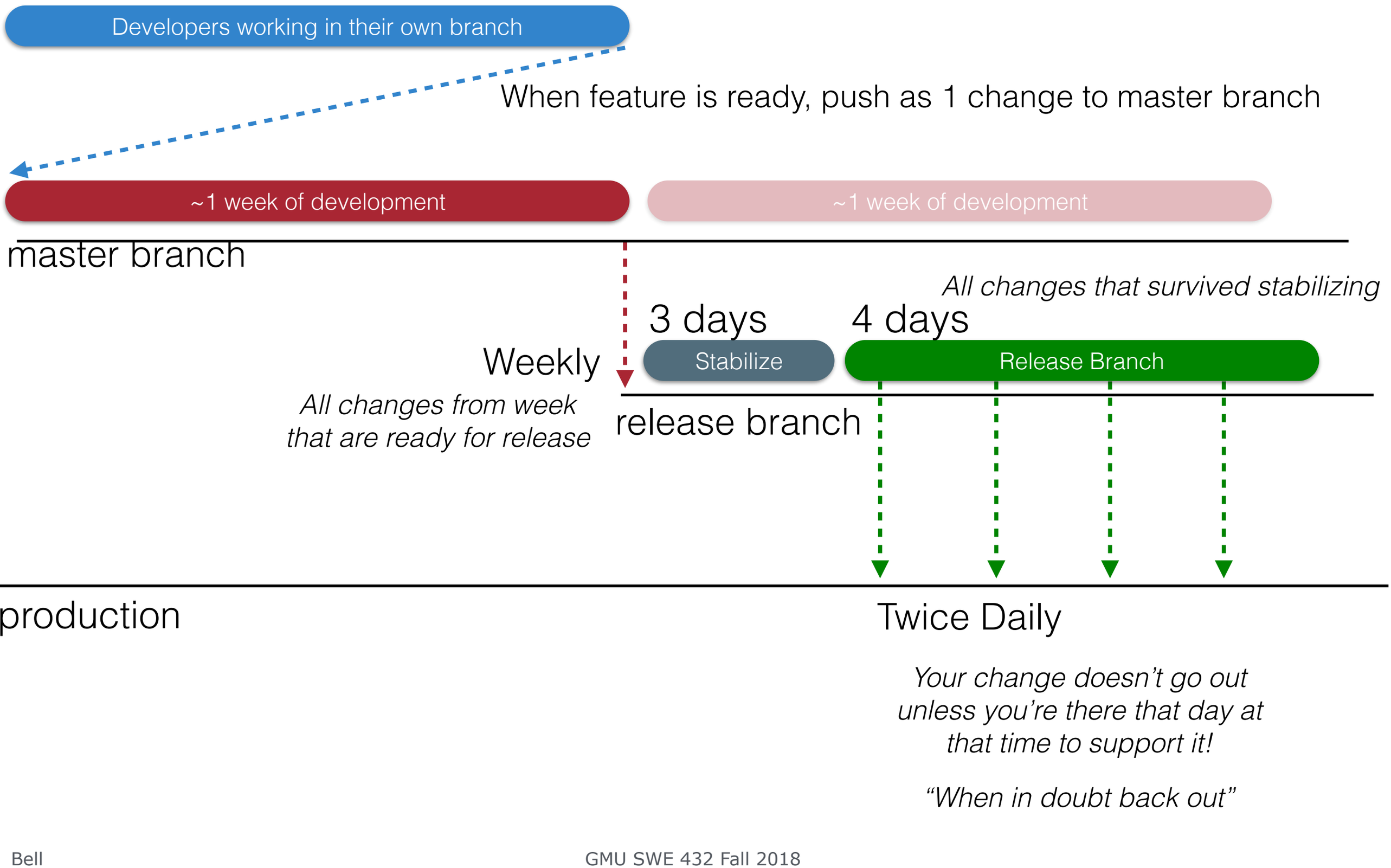


Review: DevOps Values

- No silos, no walls, no responsibility "pipelines"
- One team owns changes "from cradle to grave"
- *You* are the support person for your changes, regardless of platform
- Example: Facebook mobile teams



Review: Deployment Example: Facebook.com



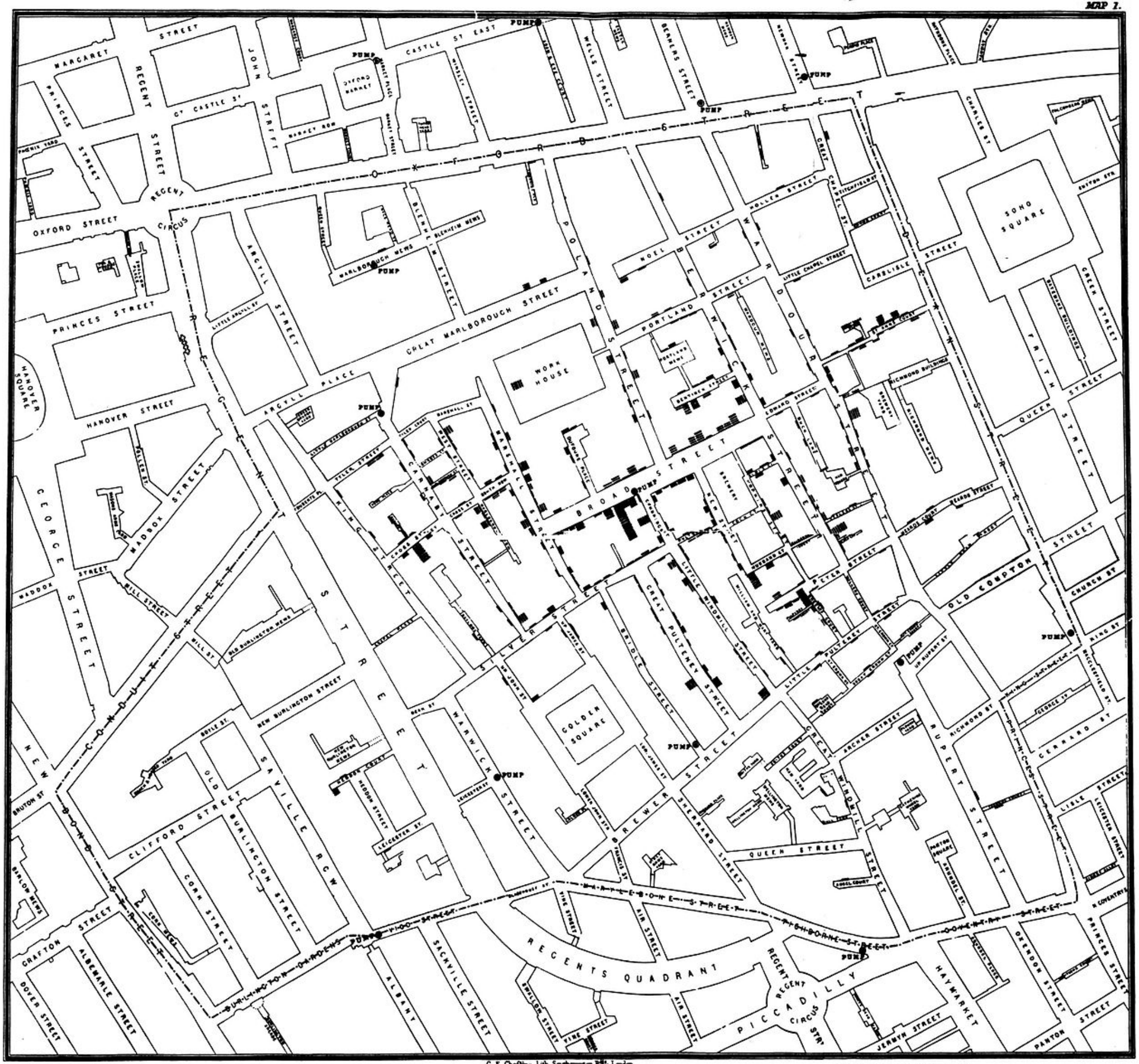
Information visualization

- Technology has made data pervasive
  - health, finance, commerce, customer, travel, demographics, communications, …
  - some of it “big”
- Information visualization: the use of interactive visual representations to amplify cognition
  - e.g., discover insights, answer questions

Cholera Epidemic in London, 1854

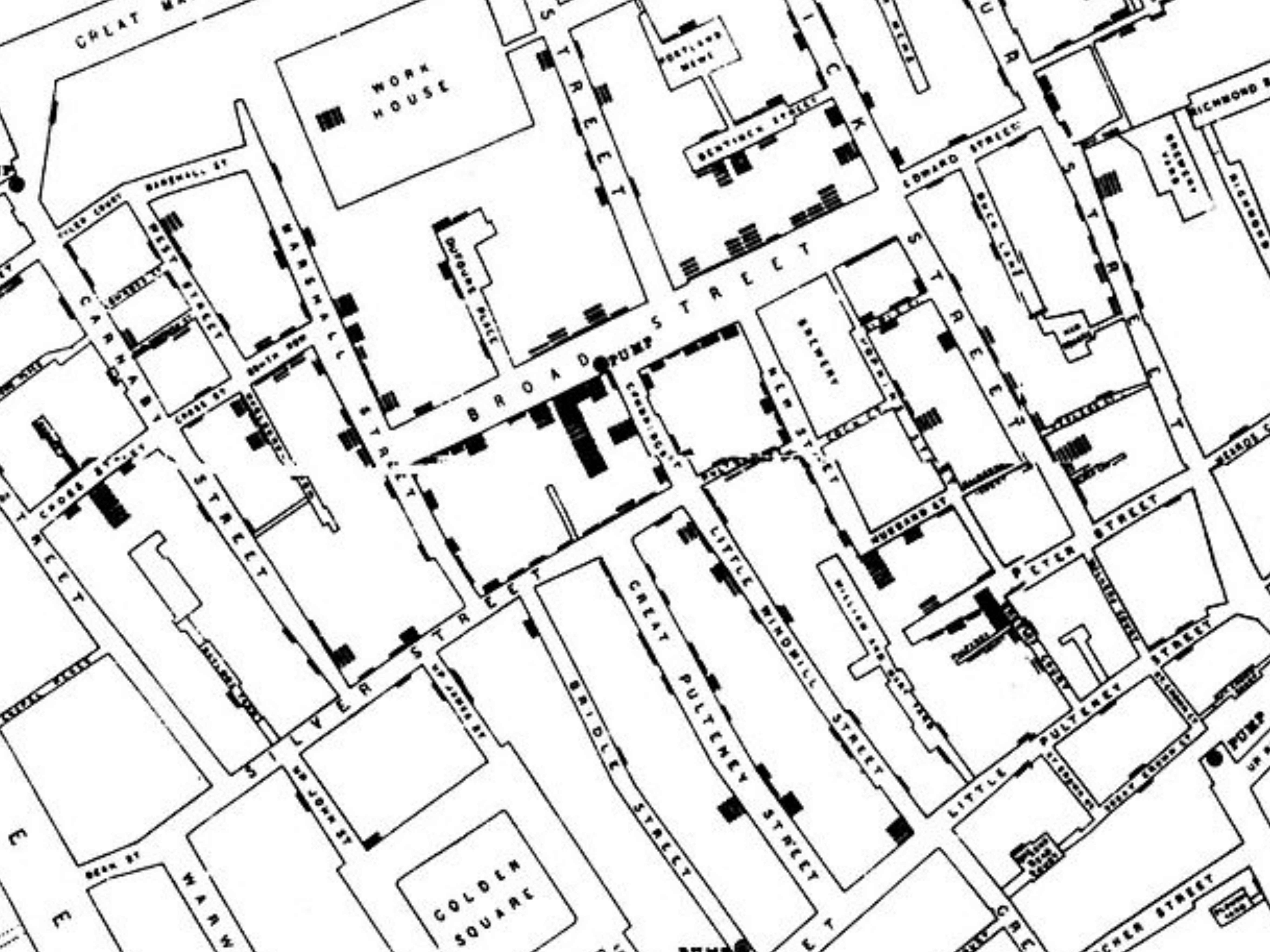
- >500 fatal attacks of cholera in 10 days
  - Concentrated in Broad Street area of London
  - Many died in a few hours
- Dominant theory of disease: caused by noxious odors
- Afflicted streets deserted by >75% inhabitants

MAP 1.



C. F. Cheffins, Lith. Southampton Bldgs. London.

SCALE 30 INCHES TO A MILE.



WORK HOUSE
PORTLAND MEWS
BENTINCK STREET
EDWARD STREET
RICHMOND
BROAD STREET
PUMP
BREWERY
MARSHALL ST
WEST STREET
CARNABY
CROSS ST
HUBBARD ST
PETER STREET
SILVER STREET
GREAT PULTENEY STREET
LITTLE WINDMILL STREET
BRIDLE STREET
LITTLE PULTENEY
GOLDEN SQUARE
PUMP

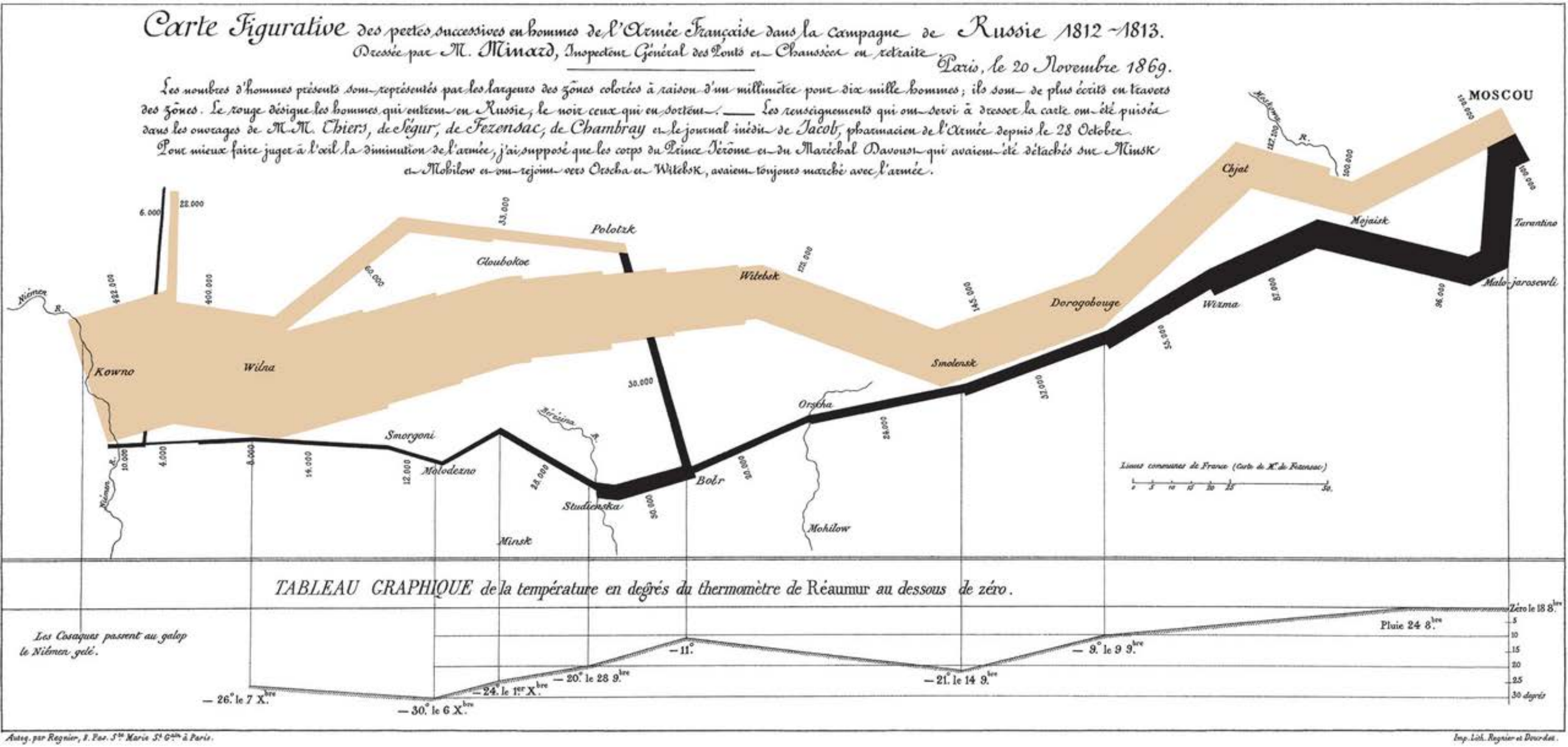
Investigation and aftermath

- Based on **visualization**, did case by case investigation
- Found that **61 / 83** positive identified as using well water from Broad Street pump
- Board ordered pump-handle to be removed from well
- Epidemic soon **ended**
- Solved centuries old question of how cholera spread

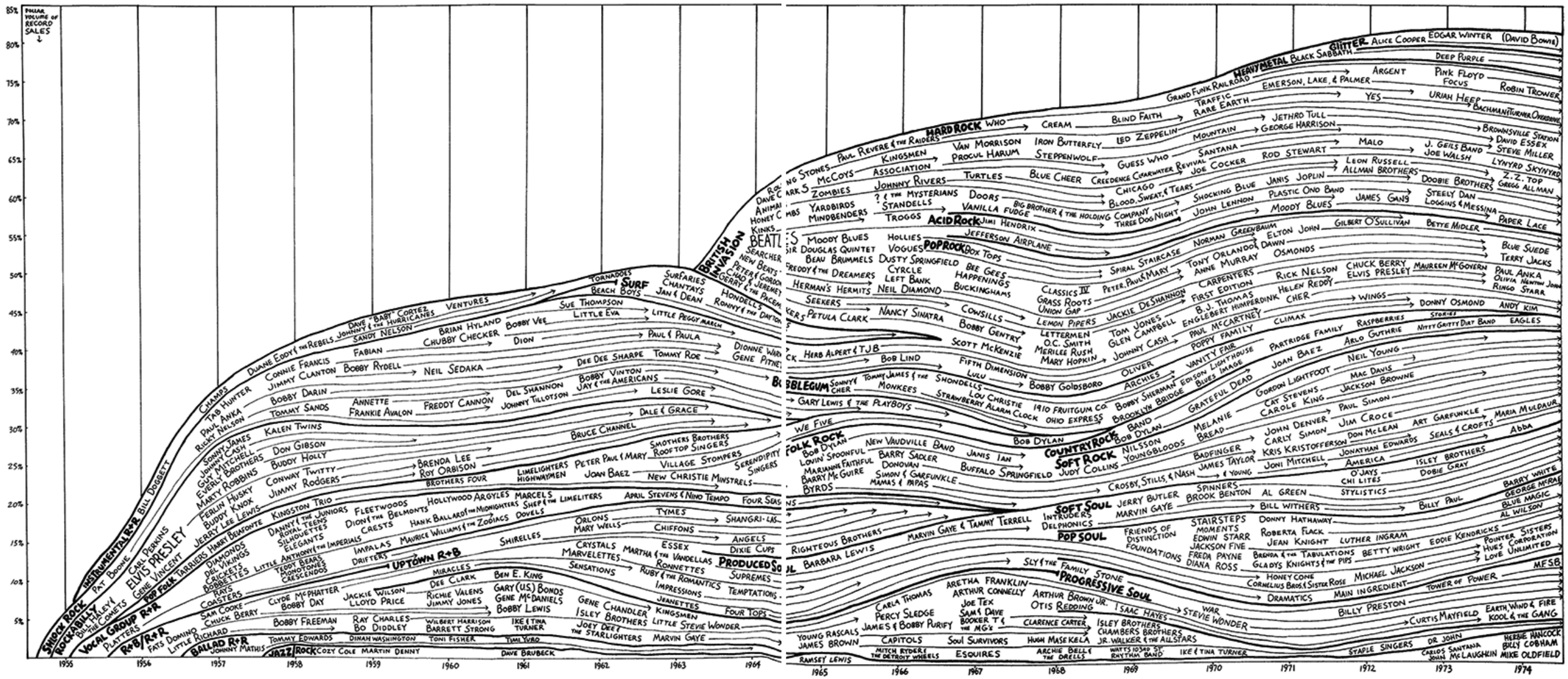
Methods used by Snow

- Placed data in appropriate **context** for assessing cause & effect
  - Plotted on map, included well location
  - Reveals proximity as cause
- Made quantitative **comparisons**
  - Fewer deaths closer to brewery, could investigate cause
- Considered **alternative** explanations & contrary cases
  - Investigated cases not close to pump, often found connection to pump
- Assessment of possible **errors** in numbers

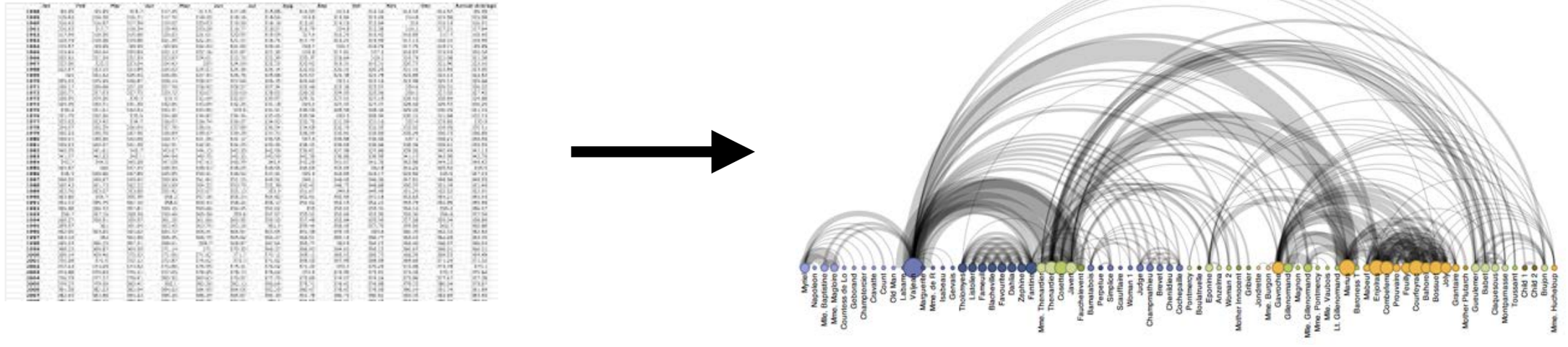
Charles Minard's Map of Napoleon's Russian Campaign of 1812



Chapel & Garofalo, Rock ’N Roll is Here to Pay: The History and Politics of the Music Industry



What is an information visualization?



- Data —> Visual representation
  - Rows in data table —> elements in data visualization
    - e.g., historical person —> circle in visualization
  - Columns of data —> visual variables
    - e.g., relationship to another person —> edge in network visualization

Data transformations

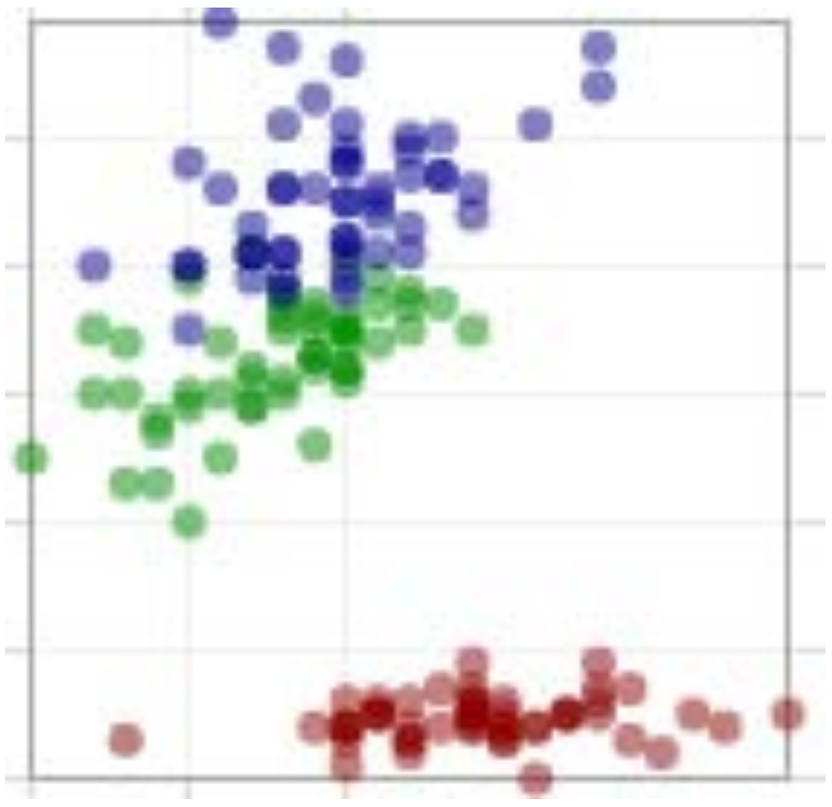
- Classing / binning: Quantitative —> ordinal
  - Maps ranges onto **classes** of variables
  - Can also count # of items in each class w/ histogram
- Sorting: Nominal —> ordinal
  - Add order between items in sets
- Descriptive statistics: mean, average, median, max, min, …

Visual structures

- 3 components
  - spatial substrate
  - marks
  - marks' graphical properties

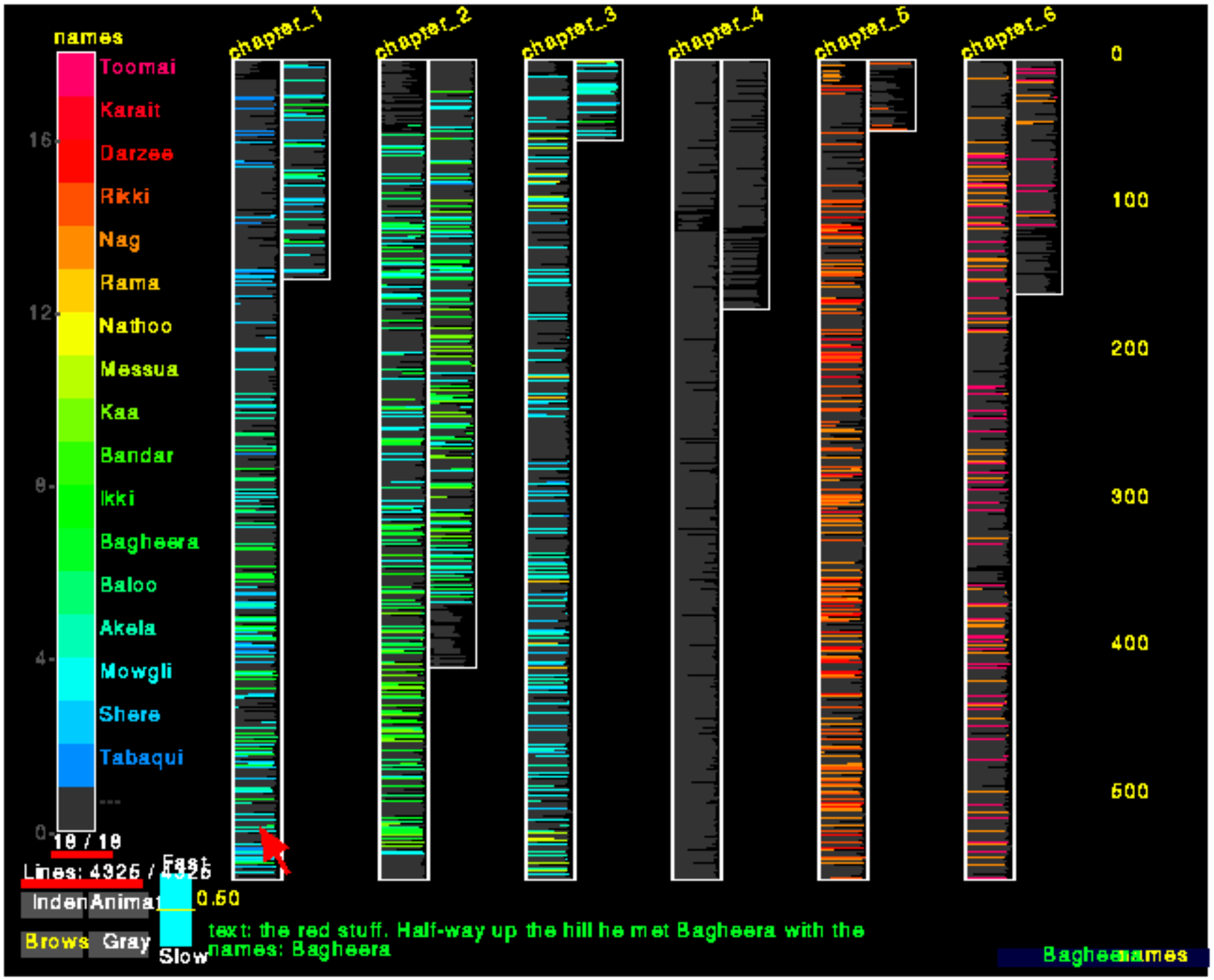
Spatial substrate

- Axes that divide space
- Types of axes - unstructured, nominal, ordinal, quantitative
- Composition - use of multiple orthogonal axes (e.g., 2D scatterplot, 3D)



Folding

- continuing an axis by continuing in different space

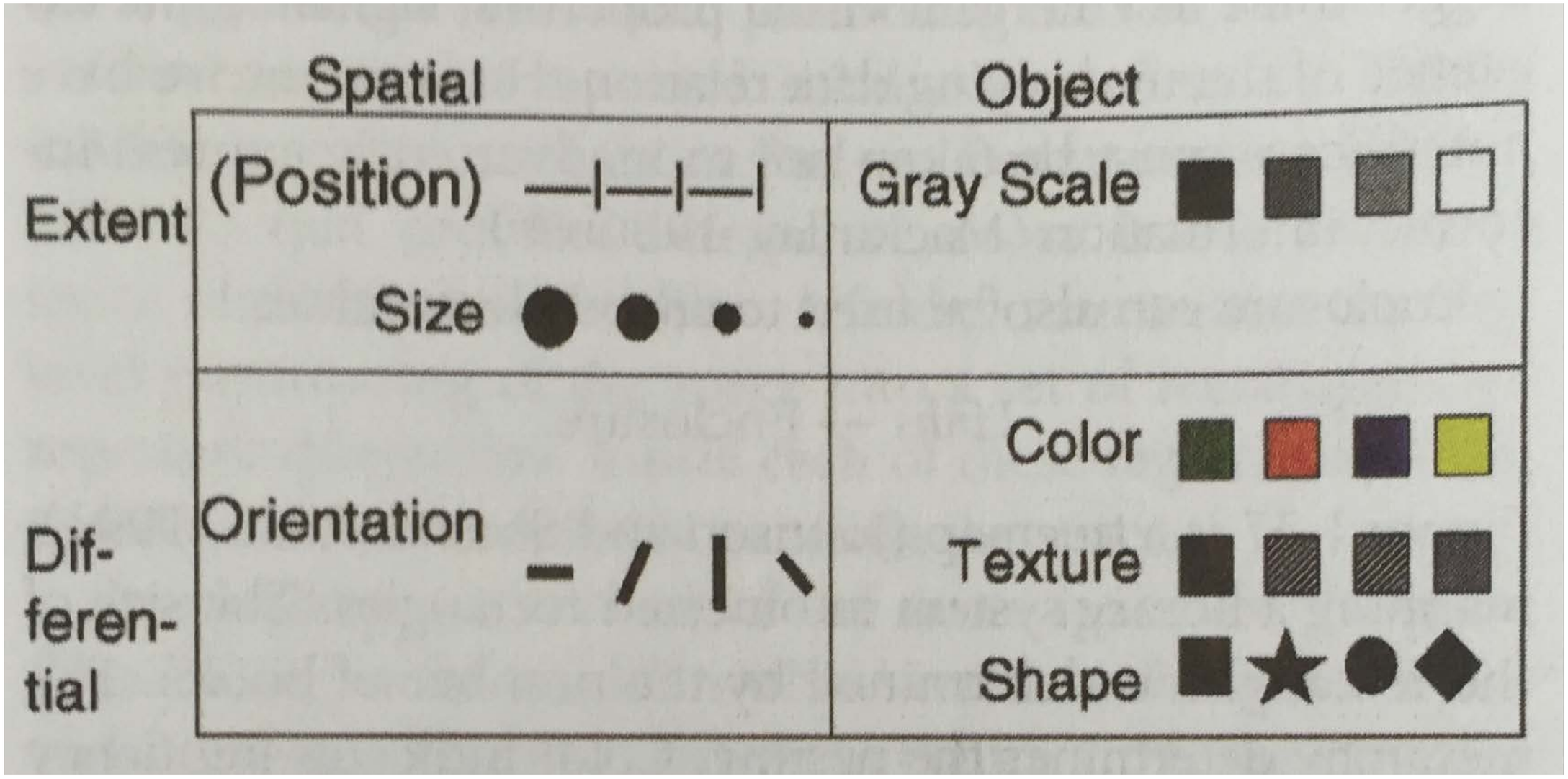


Marks

- Points (0D)
- Lines (1D)
- Areas (2D)
- Volumes (3D)

Marks' graphical properties

- Quantitative (Q), Ordinal (O), Nominal (N)
- Filled circle - good; open circle - bad



Effectiveness of graphical properties

- Quantitative (Q), Ordinal (O), Nominal (N)
- Filled circle - good; open circle - bad

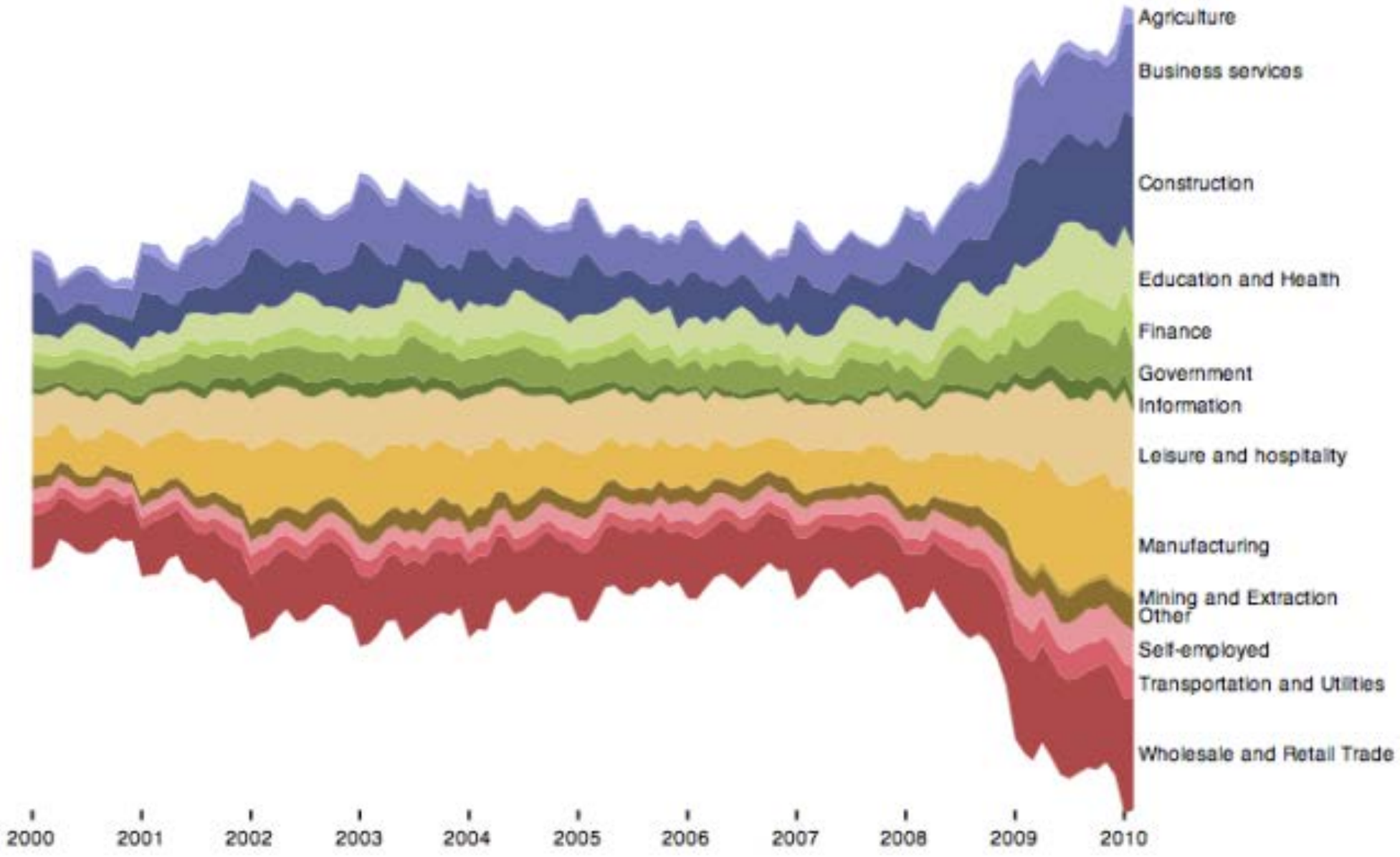
	Spatial	Q	O	N	Object	Q	O	N
Extent	(Position)	●	●	●	Grayscale	◑	●	○
	Size	●	●	●				
Differential	Orientation	◑	◑	●	Color	◑	◑	●
					Texture	◑	◑	●
					Shape	○	○	●

Examples of visualizations

Time-series data

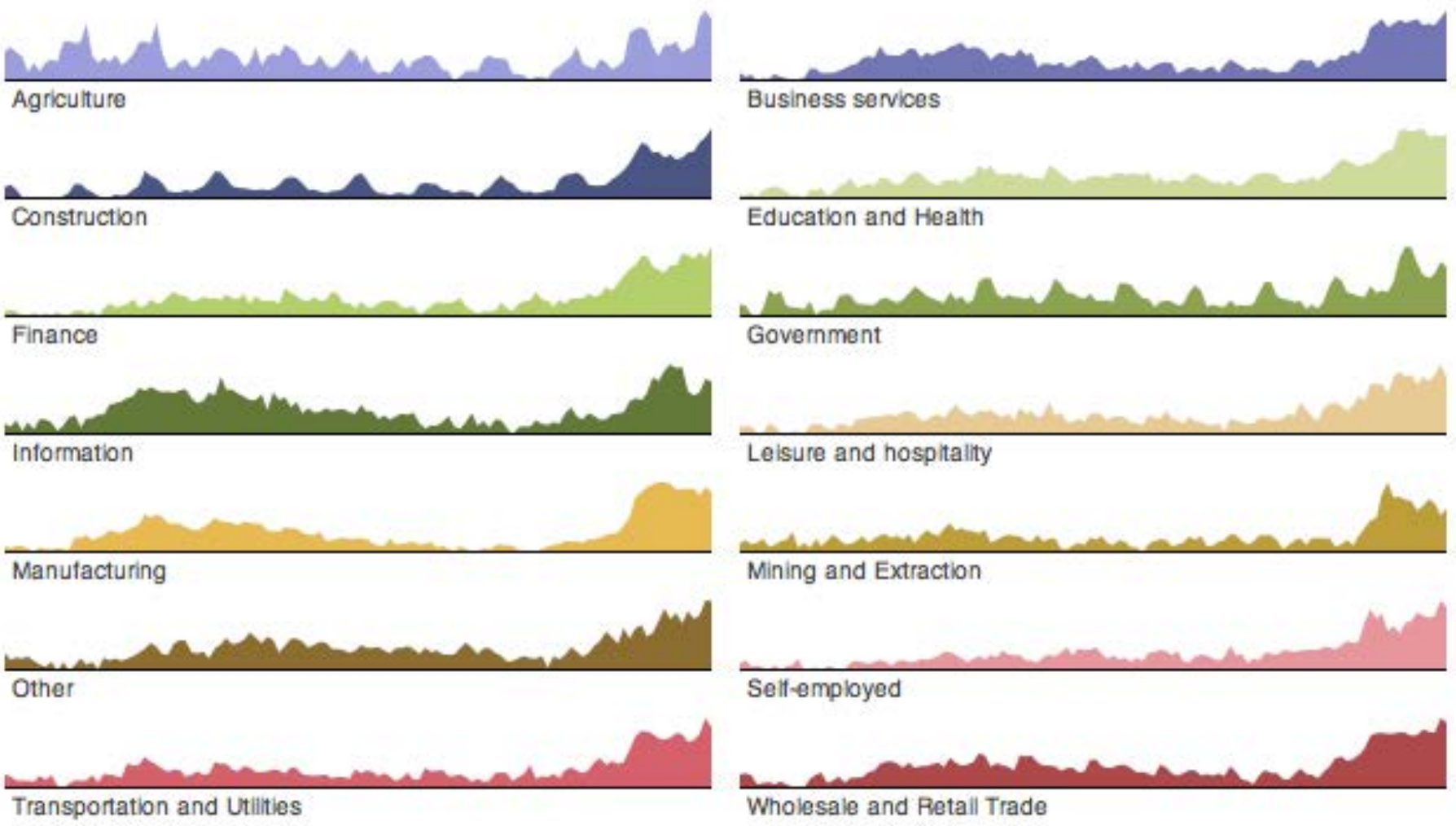
Stacked graph

- Supports visual summation of multiple components



Small multiples

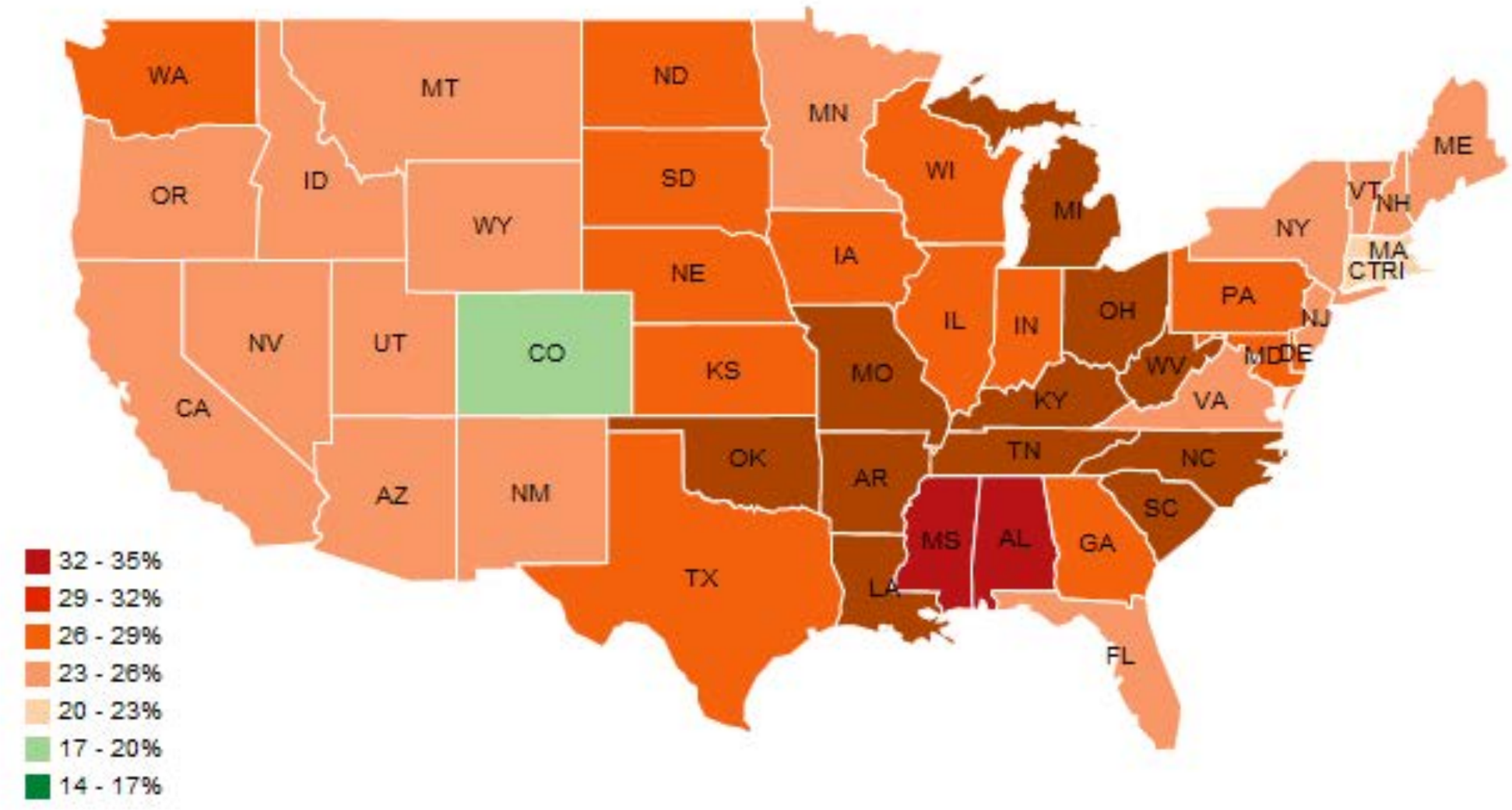
- supports separate comparison of data series
- may have better legibility than placing all in single plot



Maps

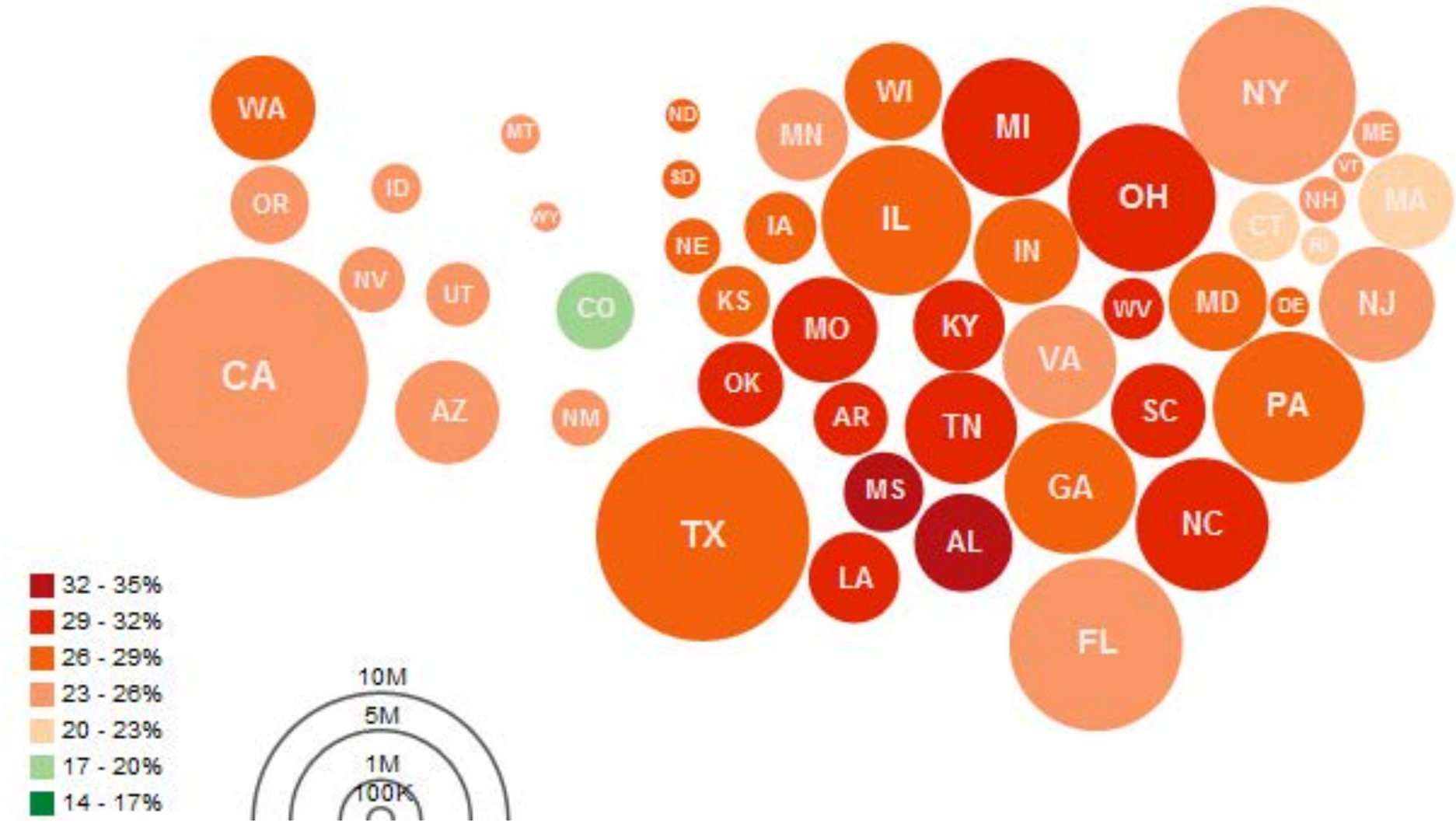
Choropleth map

- Groups data by area, maps to color



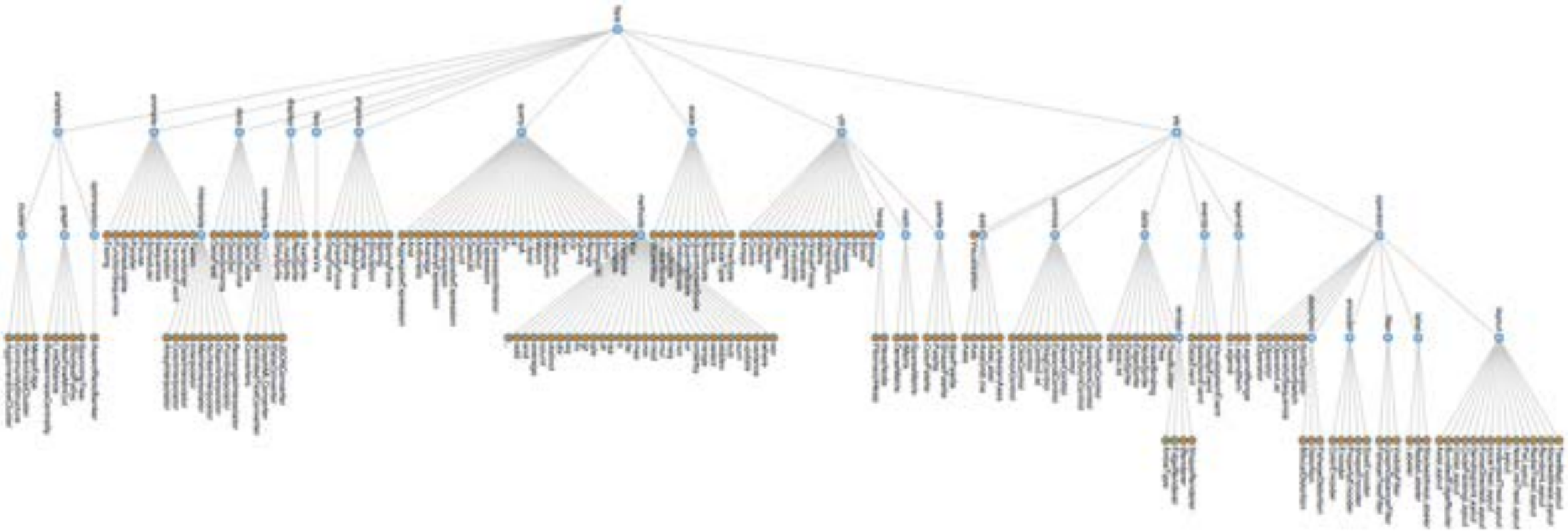
Cartograms

- Encodes two variables w/ size & color



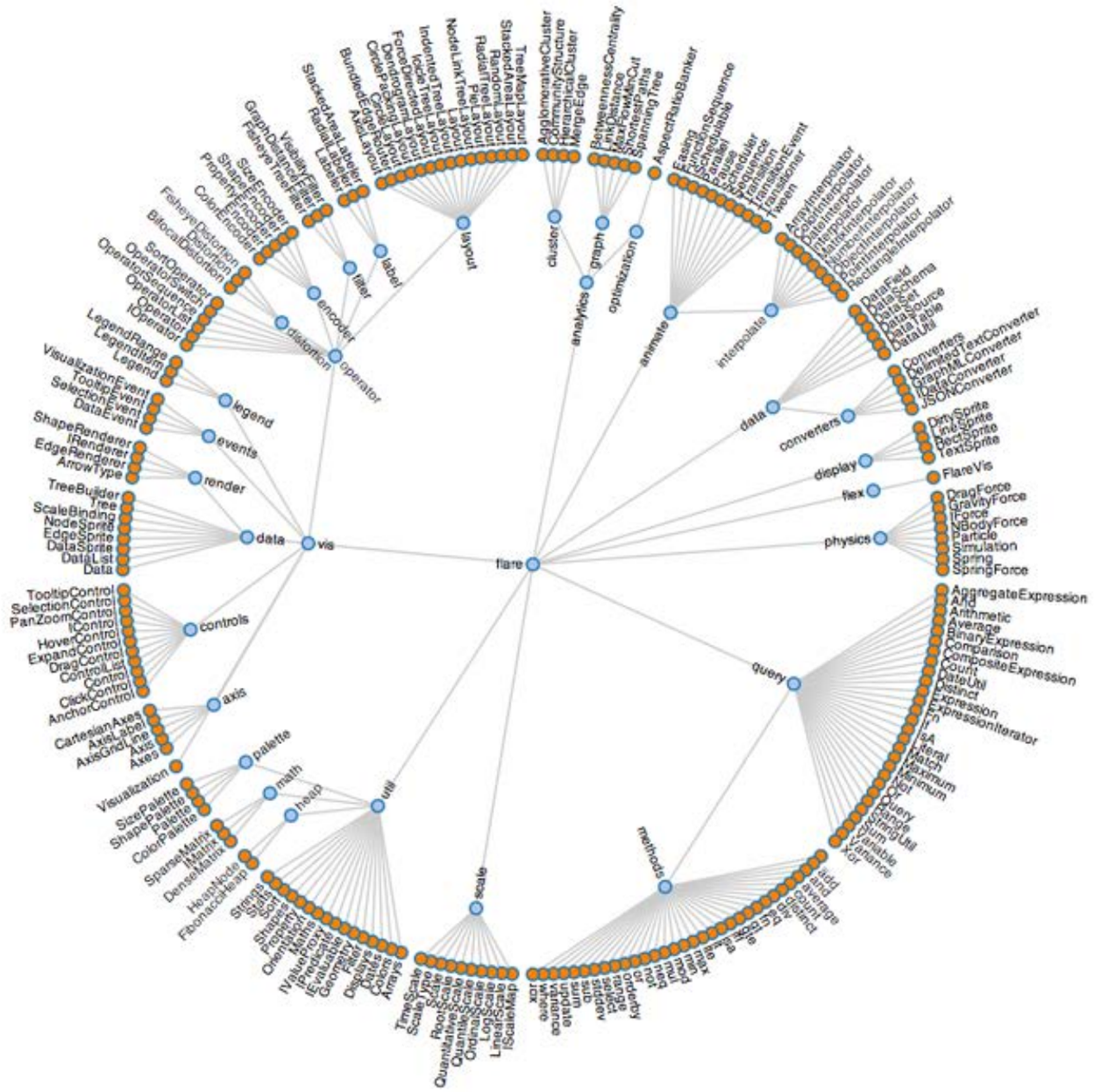
Hierarchies

Node link diagram



Dendrogram

- leaf nodes of hierarchy on edges of circle



Treemaps


Labeler
Strings
Shapes
Maths
Interpolator
Transitioner
Displays
Geometry
Arrays
Easing
Transition
Data
DataList
CircleLayout
StackedAreaLayout
Layout
AxisLayout
PieLayout
Distortion
OperatorList
ColorPalette
Palette
FibonacciHeap
Dates
Stats
Property
Colors
Sort
Tween
Scheduler
Parallel
Sequence
Pause
NodeSprite
ScaleBinding
DataSprite
TreeBuilder
Tree
Legend
LegendRange
Axis
CartesianAxes
Visualization
Query
Expression
Comparison
DateUtil
StringUtil
Arithmetic
Match
Scale
TextSprite
DirtySprite
Simulation
Spring
Particle
FlareVis

Some challenges in information visualizations

- Data binding
  - You have data. How do you create corresponding visual elements?
  - How do you update the visual elements if the data changes?
    - Or the user updates what they want to see…
- Scales
  - How do data values correspond to position, size, color, etc. of visual elements?
- Transitions
  - How do you smoothly animate changes between visual states?

Design considerations

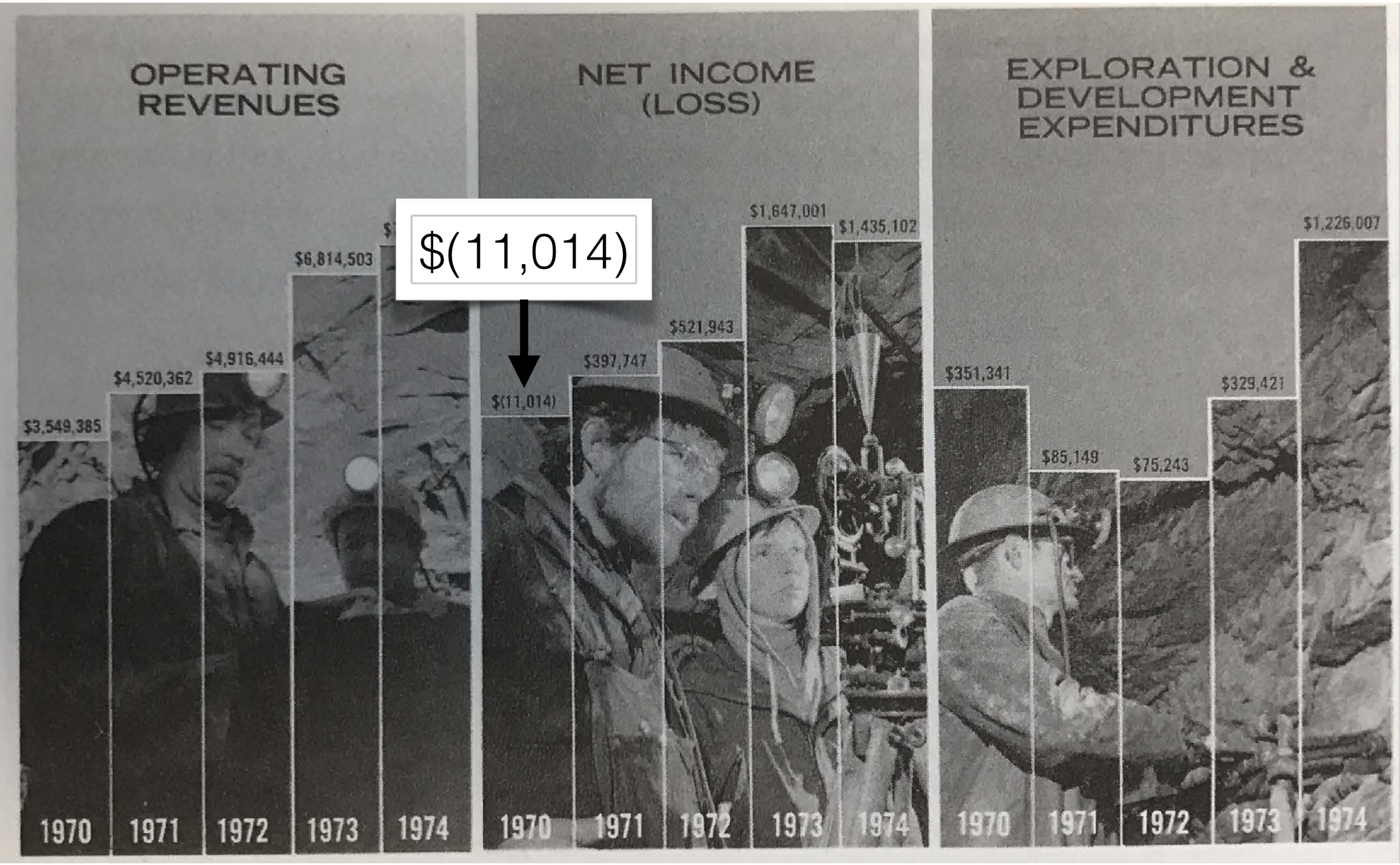
Tufte's principles of graphical excellence

- show the **data**
- induce the viewer to think about the substance rather than the methodology
- avoid distorting what the data have to say
- present **many** numbers in a small space
- make large data sets **coherent**
- encourage the eye to **compare** different pieces of data
- reveal data at several levels of detail, from overview to fine structure
- serve reasonable clear **purpose**: description, exploration, tabulation, decoration

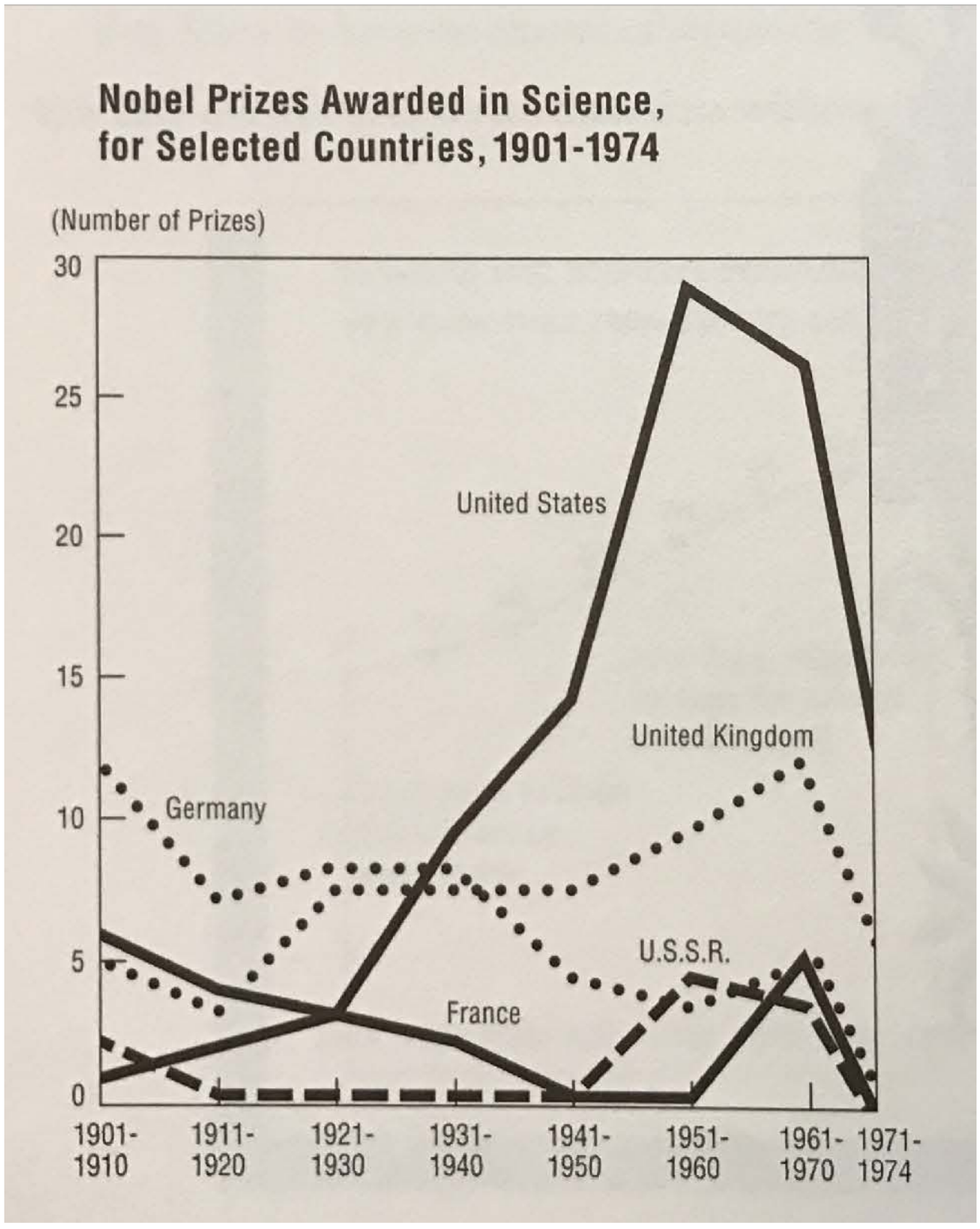
Distortions in visualizations

- Visualizations may distort the underlying data, making it harder for reader to understand truth
- Use of **design** variation to try to falsely communicate **data** variation

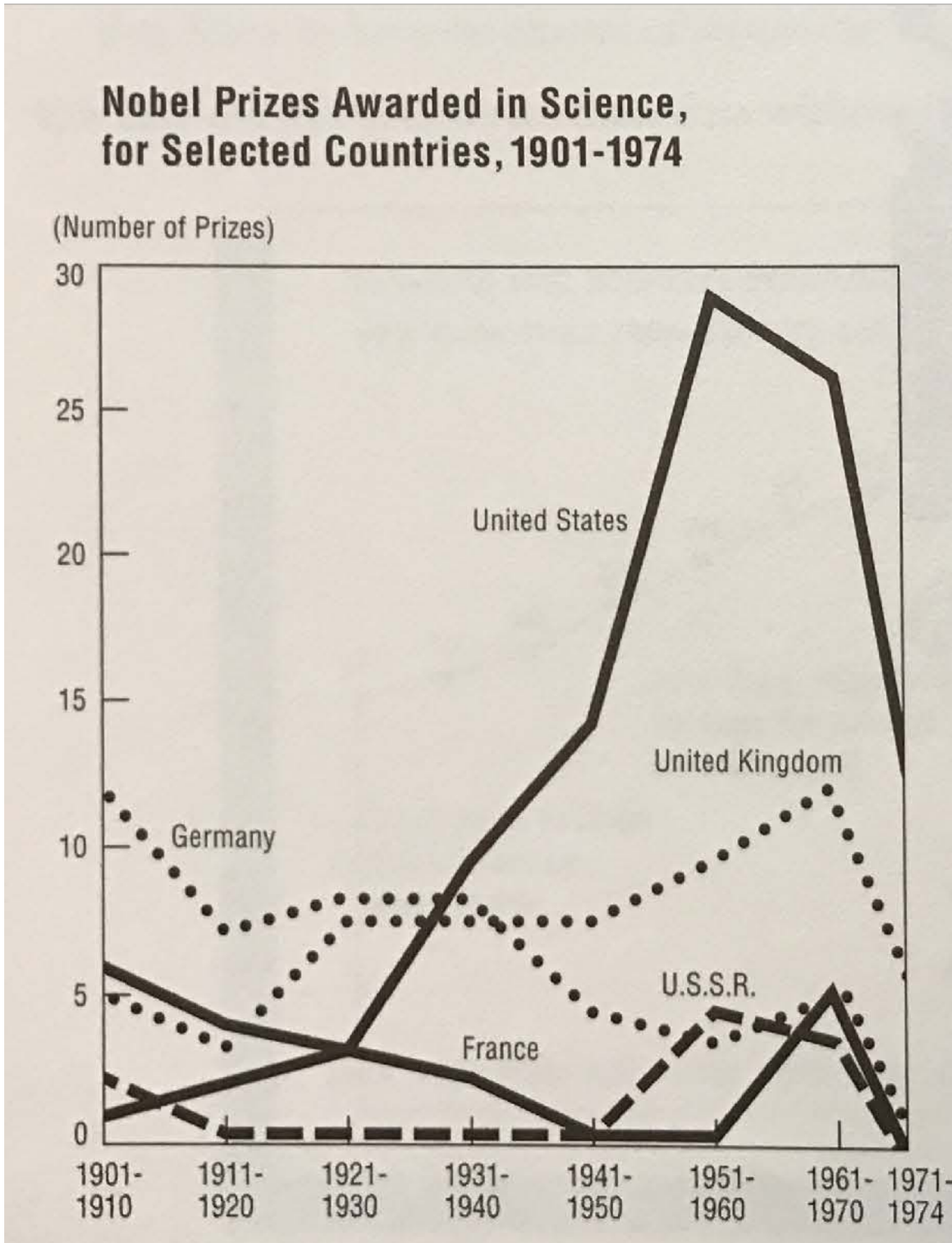
Example

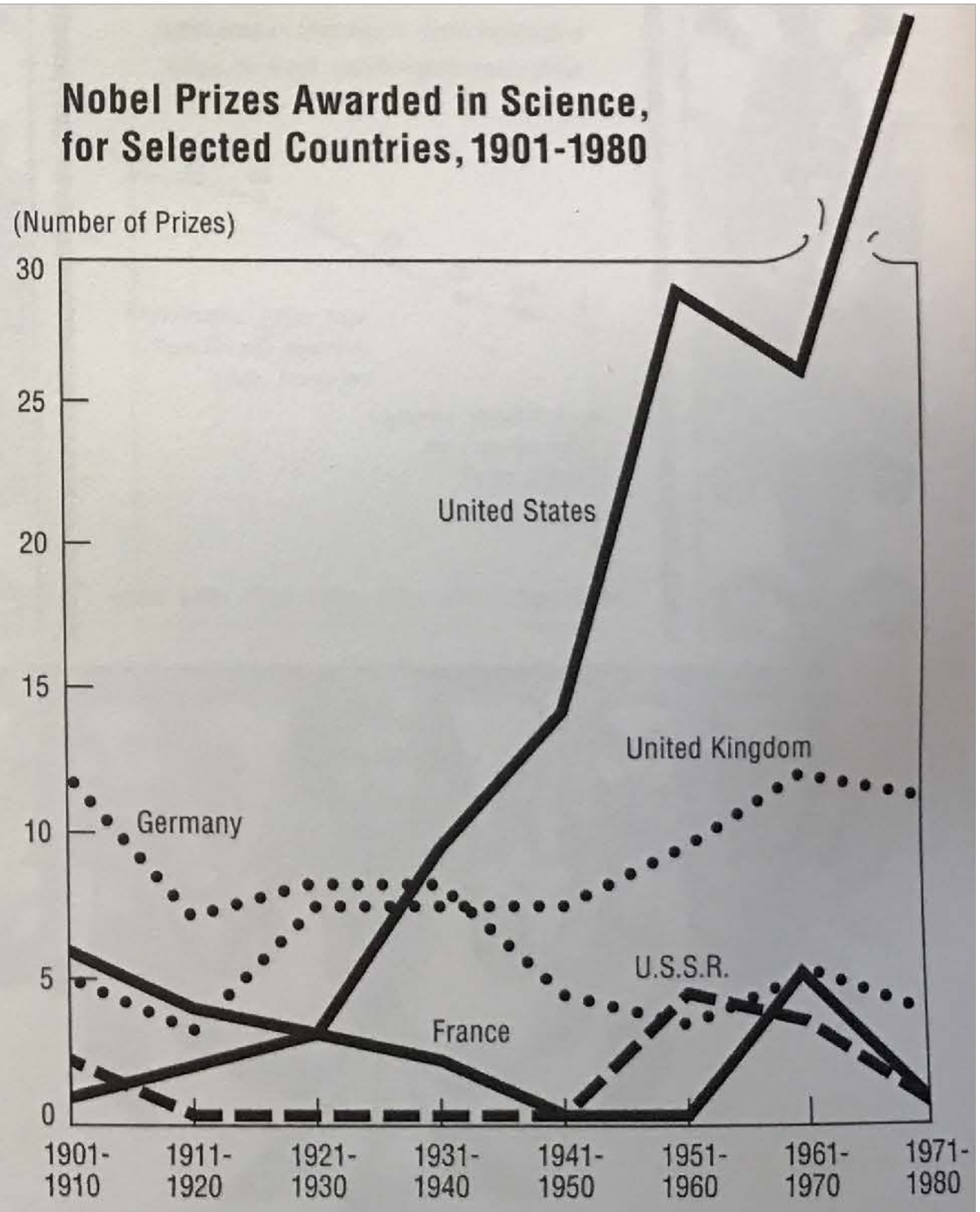

OPERATING REVENUES
$3,549,385
$4,520,362
$4,916,444
$6,814,503
1970
1971
1972
1973
1974
NET INCOME (LOSS)
$(11,014)
$(11,014)
$397,747
$521,943
$1,647,001
$1,435,102
1970
1971
1972
1973
1974
EXPLORATION & DEVELOPMENT EXPENDITURES
$351,341
$85,149
$75,243
$329,421
$1,226,007
1970
1971
1972
1973
1974

Example

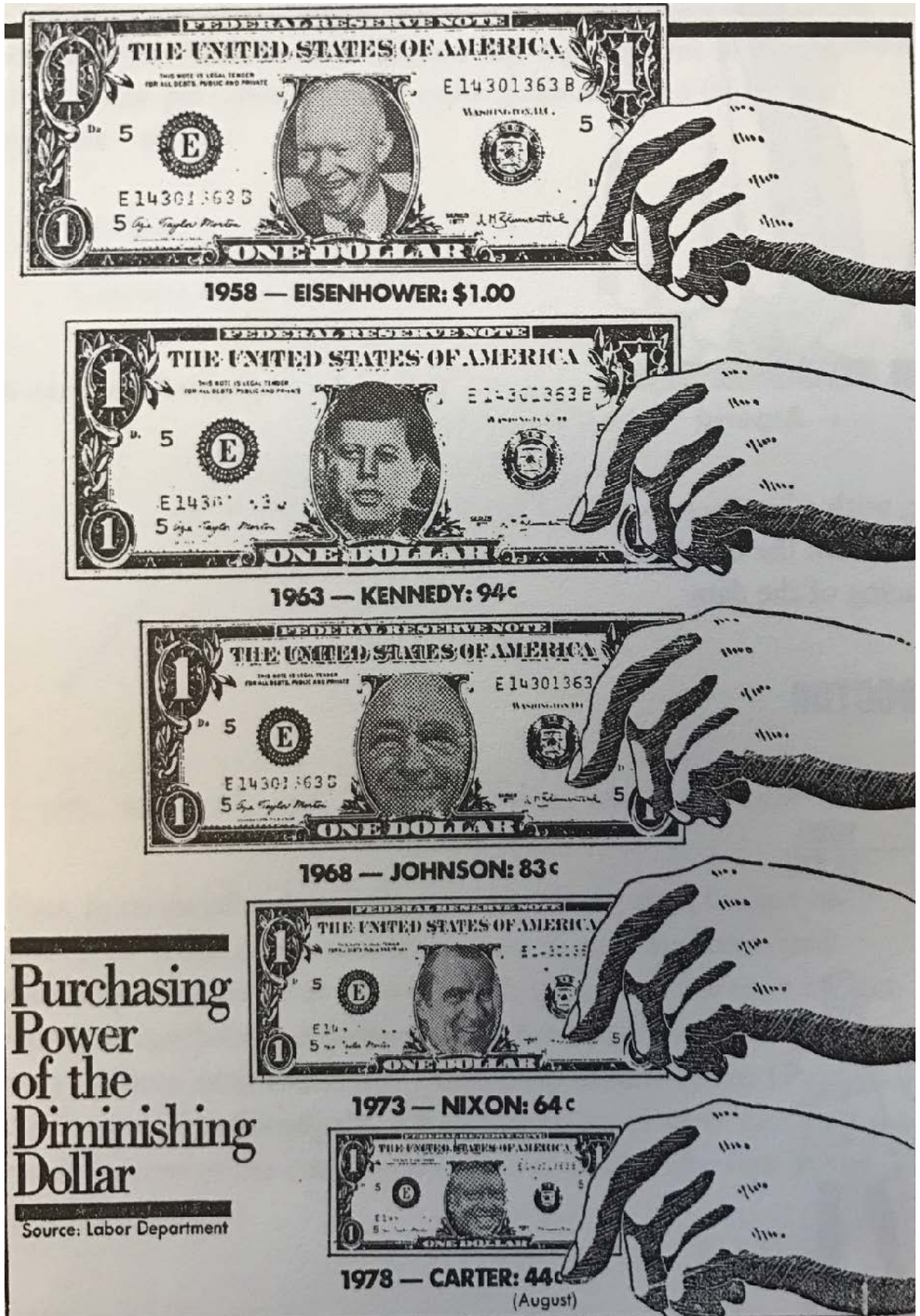

Nobel Prizes Awarded in Science,
for Selected Countries, 1901-1974
(Number of Prizes)
30
25
20
15
10
5
0
United States
United Kingdom
Germany
U.S.S.R.
France
1901-1910
1911-1920
1921-1930
1931-1940
1941-1950
1951-1960
1961-1970
1971-1974

Example (corrected)

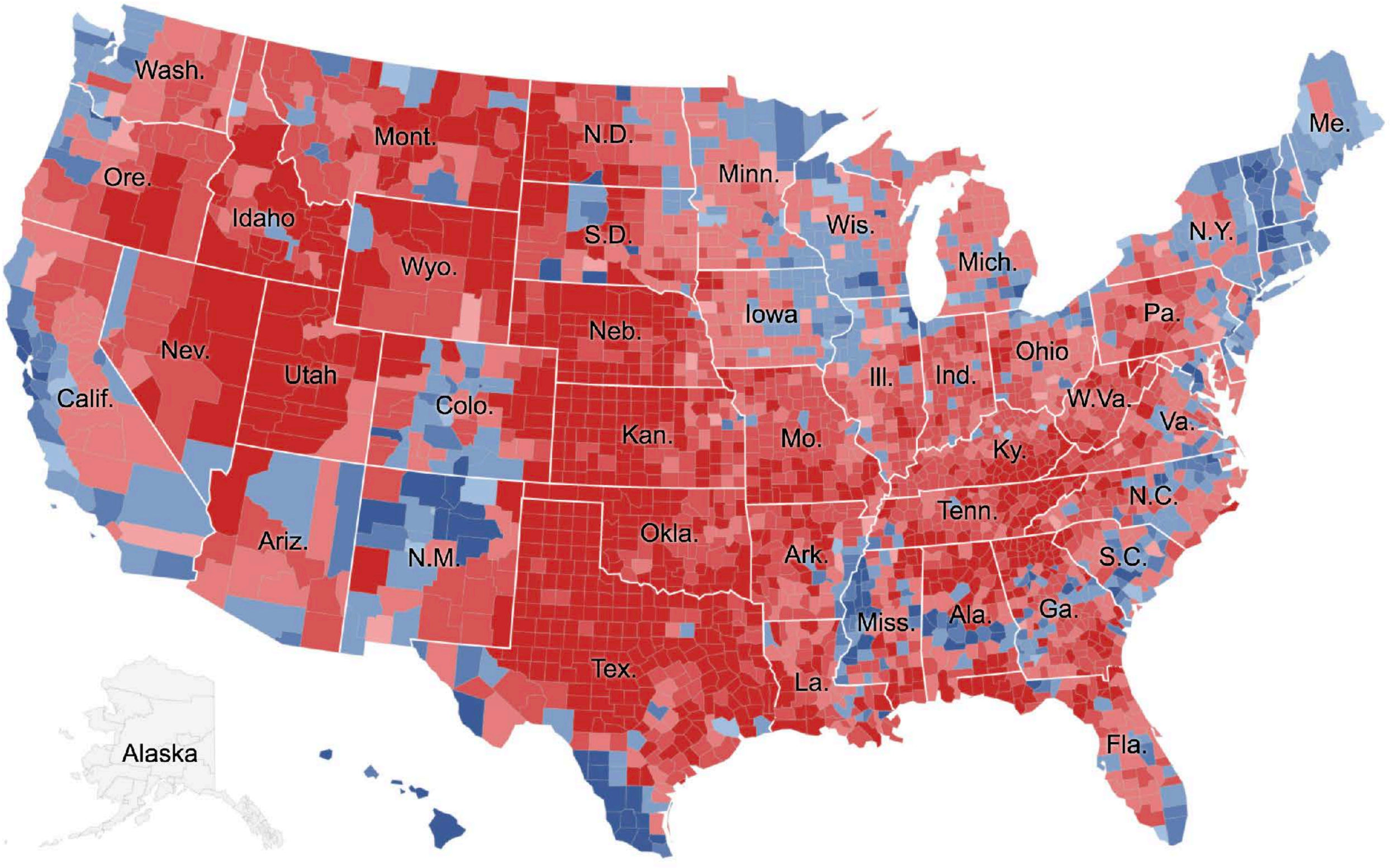




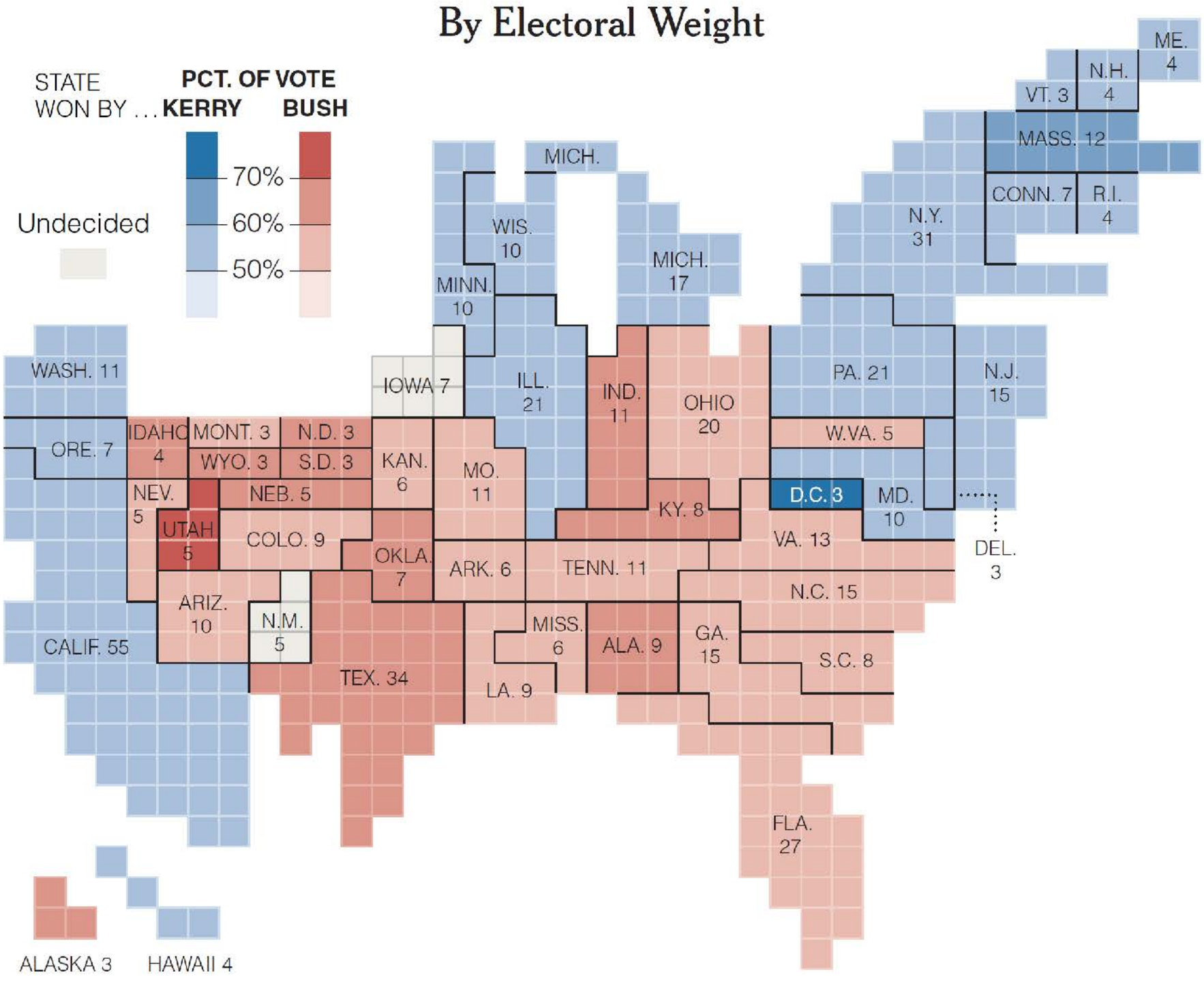
Example


1958 — EISENHOWER: $1.00
1963 — KENNEDY: 94¢
1968 — JOHNSON: 83¢
1973 — NIXON: 64¢
1978 — CARTER: 44¢
(August)
Purchasing Power of the Diminishing Dollar
Source: Labor Department

Traditional Electoral Map



Weighted Electoral Map



Data-ink

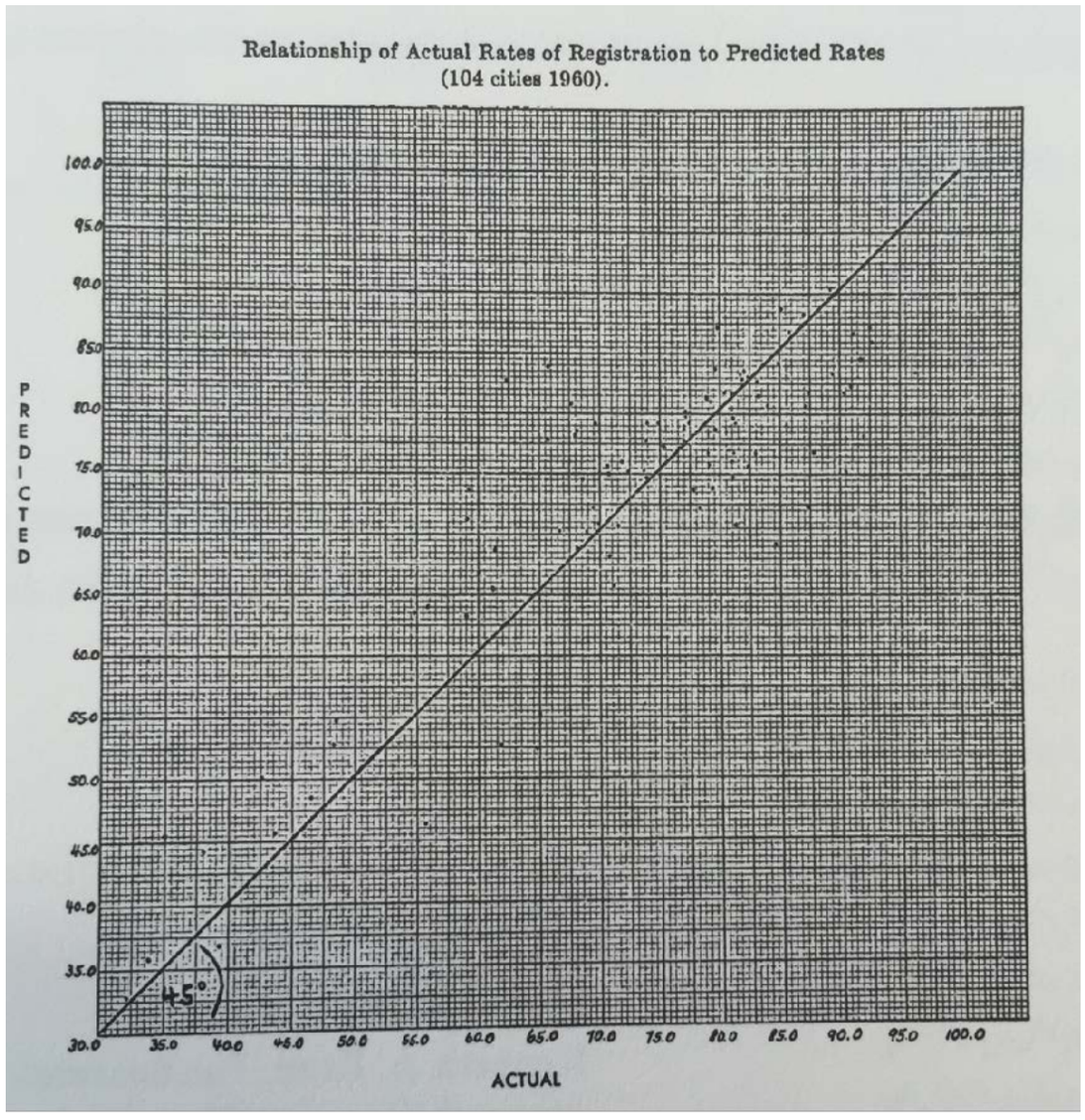
- Data-ink - non-redundant ink encoding data information

$$\text{Data-ink ratio} = \frac{\text{Data-ink}}{\text{Total ink used to print the graphic}}$$

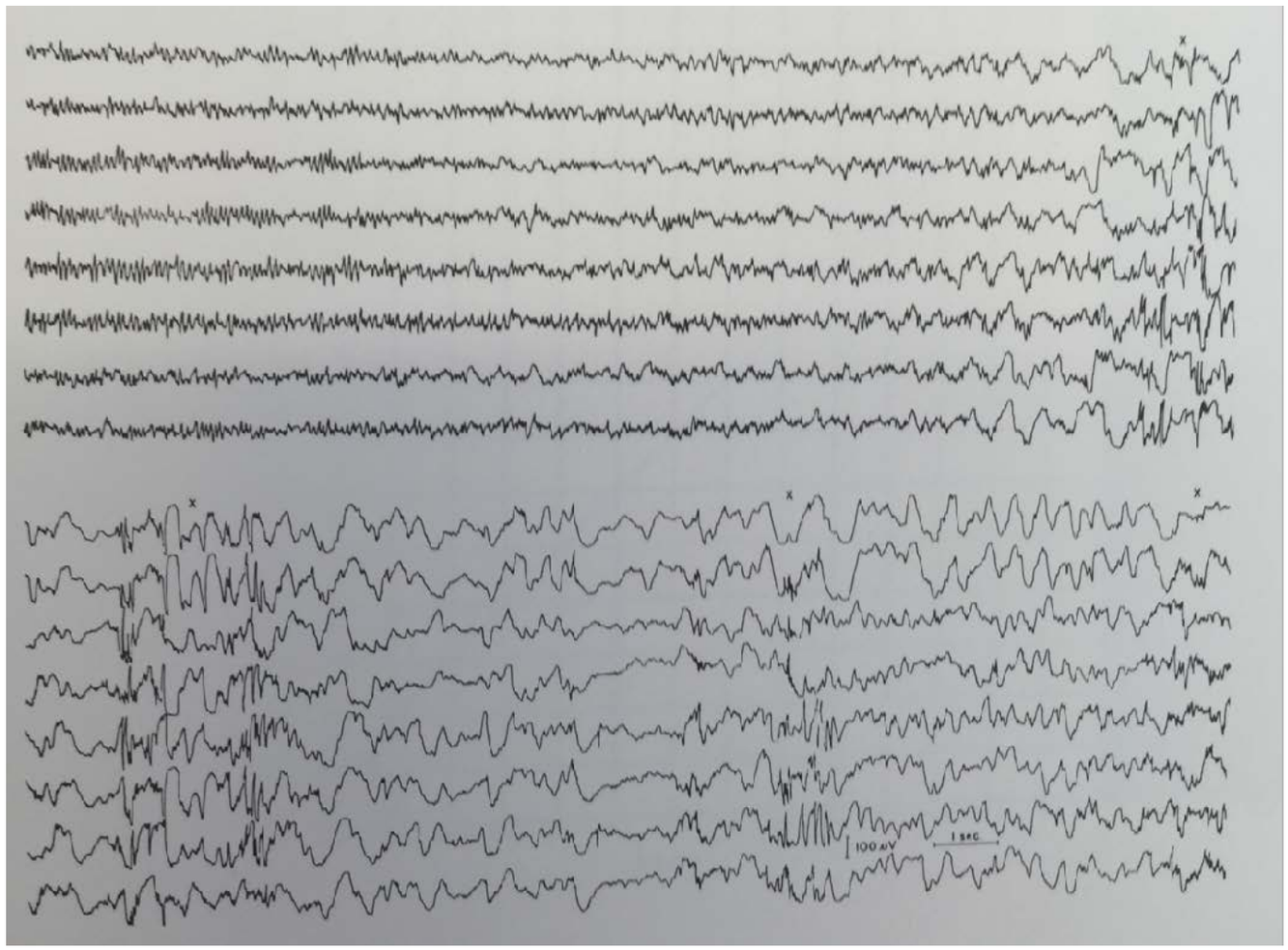
= proportion of a graphic's ink devoted to the non-redundant display of data-information

= 1.0 – proportion of a graphic that can be erased

Examples of data-ink ratio



~0

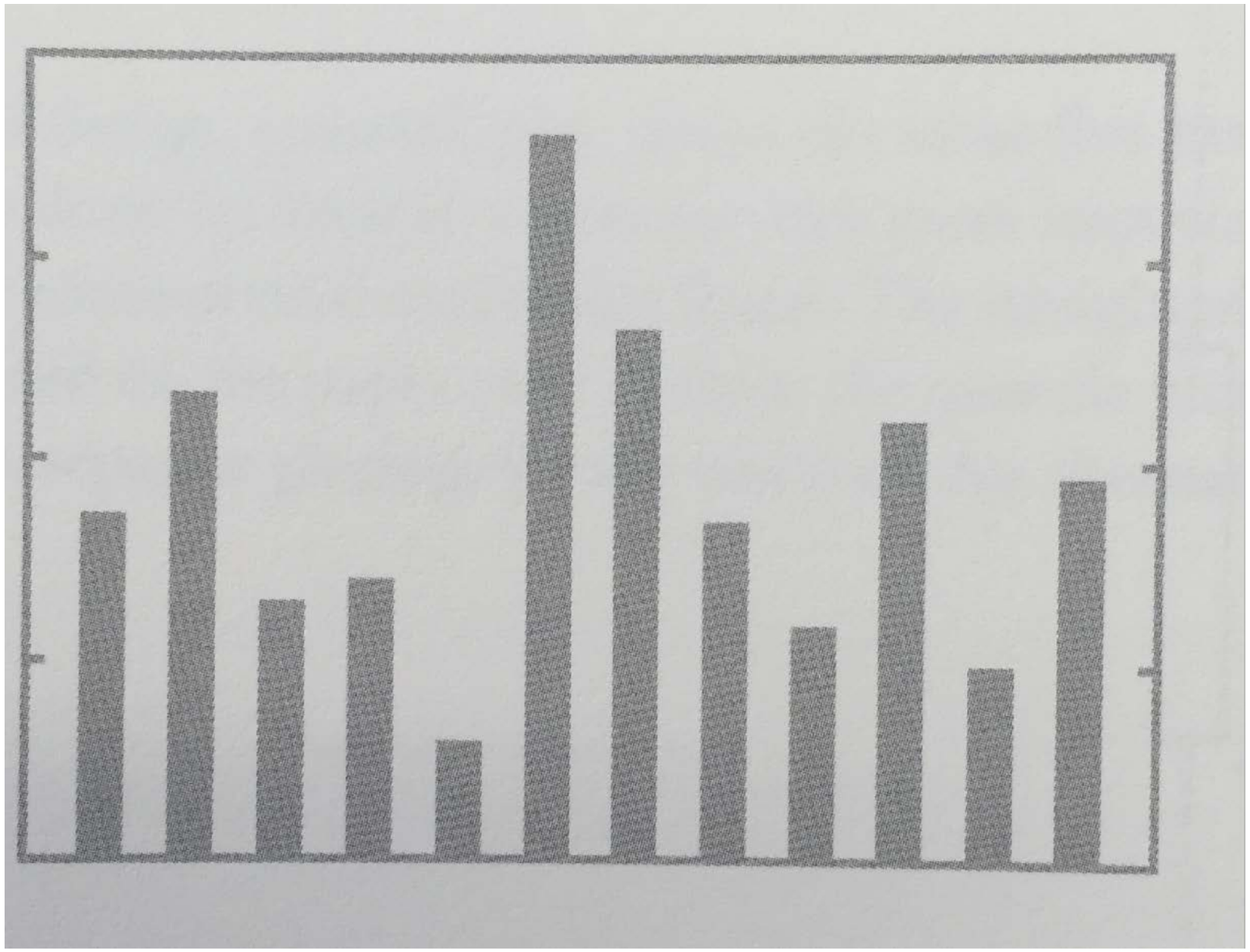


1.0

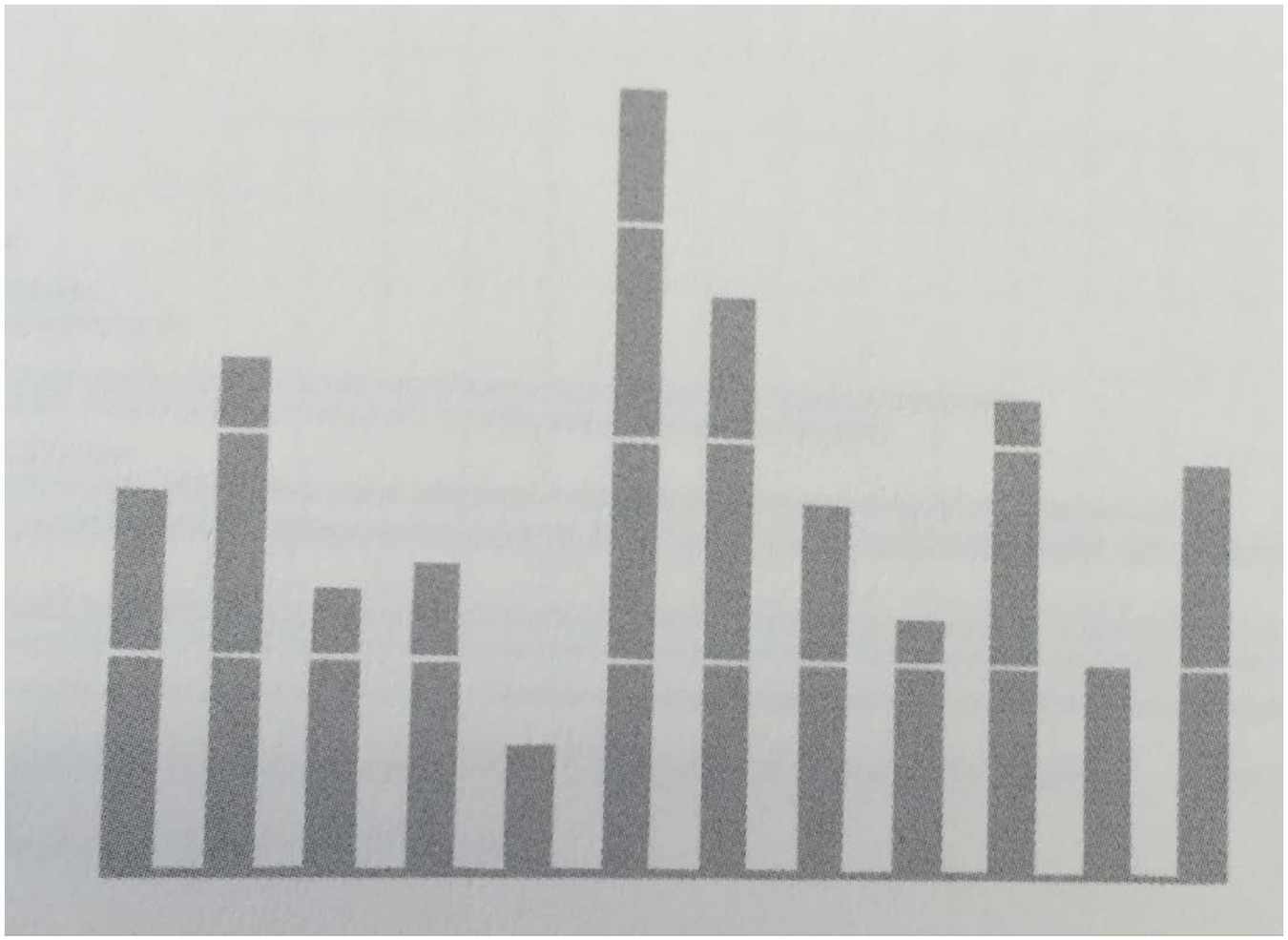
Design principles for data-ink

- (a.k.a. aesthetics & minimalism / elegance & simplicity)
- **Above all else show the data**
  - Erase non-data-ink, within reason
    - Often not valuable and distracting
    - Redundancy not usually useful

Example



Example (revised)

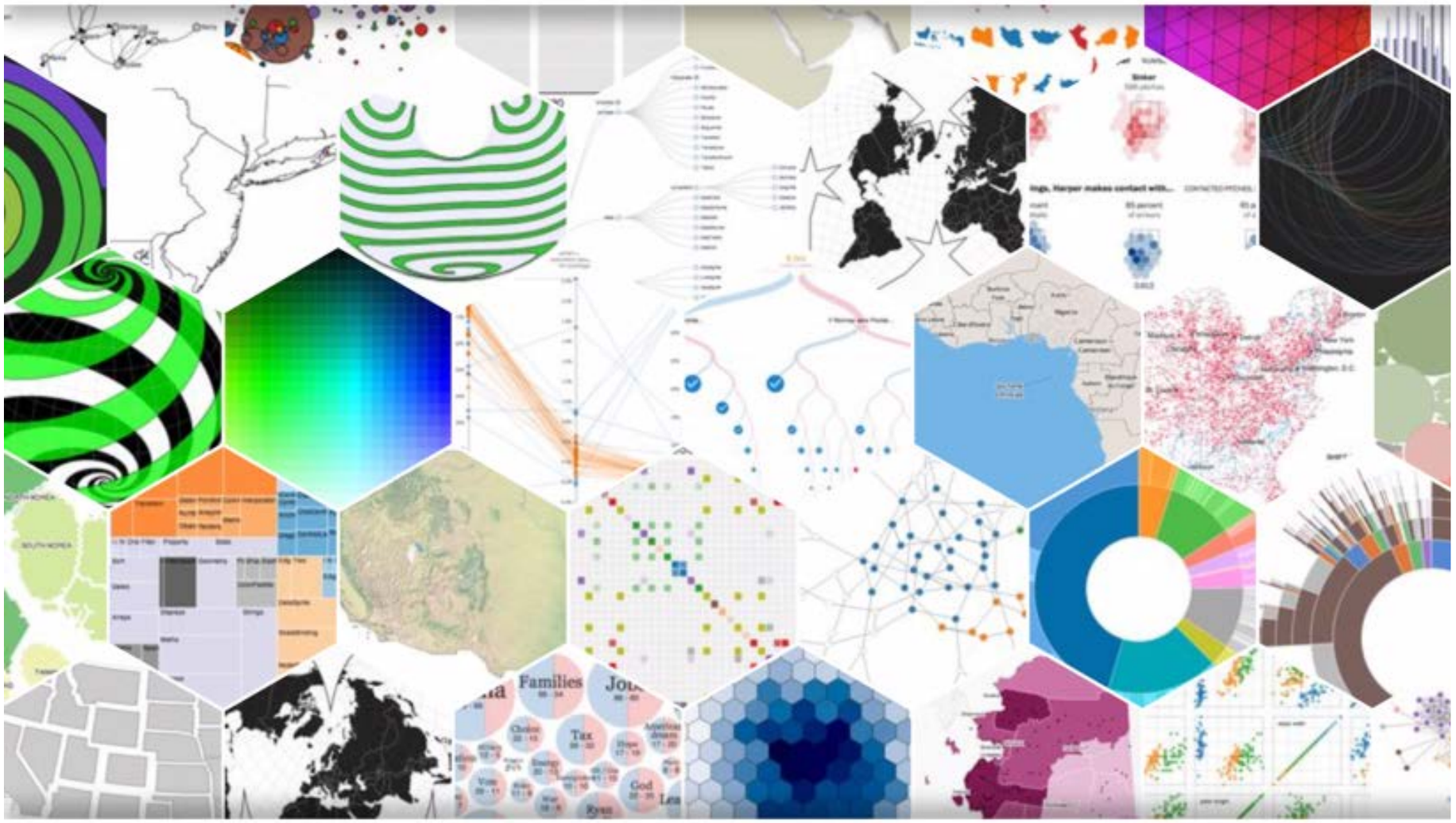


https://d3js.org

Overview Examples Documentation Source

Fork me on GitHub

Data-Driven Documents



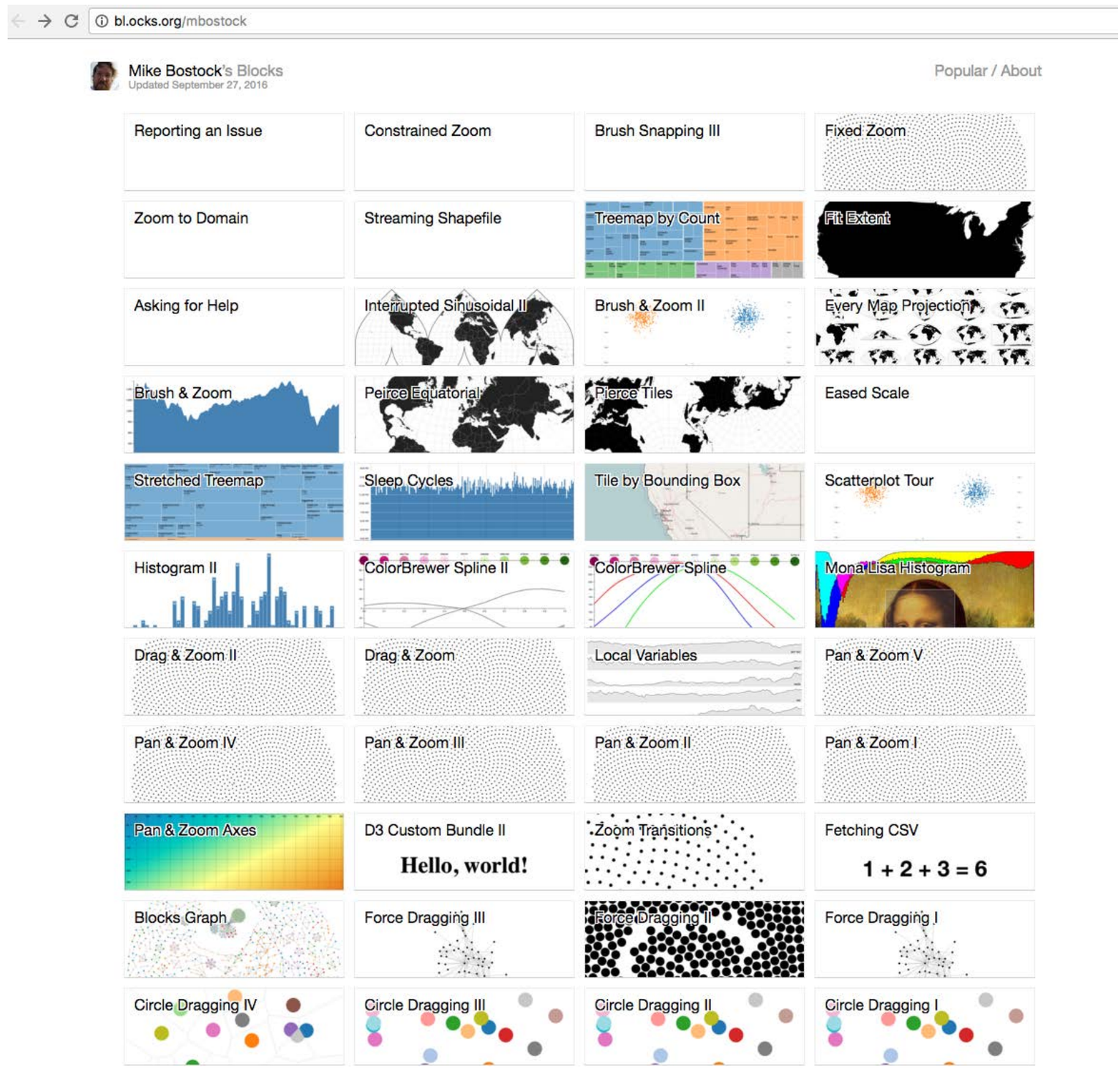
D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

D3.js

- Most popular information visualization framework for the web
  - Designed by Mike Bostock as part of his PhD
- Transform data into a visual representation
  - e.g., build HTML elements for elements in an array
- Based on web standards, including HTML, CSS, SVG

Learning D3

- Many tutorials
- Many, **many** examples
  - Frequent pattern: copy similar visualization, customize for your needs
- But… be careful you use d3 v4
  - Current version



Key concepts we'll cover today

- Selections
- Dynamic properties
- Data joins (a.k.a. data binding)
- Scales
- SVG
- Loading data

Selections

- D3 handles the mapping between data and DOM elements using “selectors”
- Example: Select all <p> tags, set their color to be white

```
d3.selectAll("p").style("color", "white");
```

- Example: Select all <p> tags, set their color based on some callback function:

```
d3.selectAll("p").style("color", function(data, index) {
  return index % 2 ? "black" : "gray";
});
```

Data binding

- We can style elements dynamically based on data.
- But…
  - usually we have a dataset (e.g., time-series data of temperature readings)
  - and we want to directly associate it with some visual elements
  - and it'd be great if we could automatically create elements based on the data.
  - and delete or update the visual elements when the data changes.

Data binding

```
<p>P1</p>
<p>P2</p>
<p>P3</p>
<p>P4</p>
```

```
d3.selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
    .style("font-size", function(d) { return d + "px"; });
```

P1

P2

P3

P4

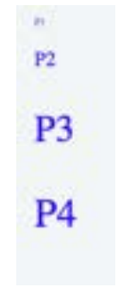
- Bind *data* with visual element.

Data binding is persistent

```
<p>P1</p>
<p>P2</p>
<p>P3</p>
<p>P4</p>
```

```
var p = d3.selectAll("p")
    .data([4, 8, 15, 16, 23, 42])
    .style("font-size", function(d) { return d + "px"; });

p.style("color", "blue");
```



- D3 uses cascade pattern, returning element set.
- By default, visual elements persist once created.
  - Can update style without binding to data again

How do we deal with changing data?

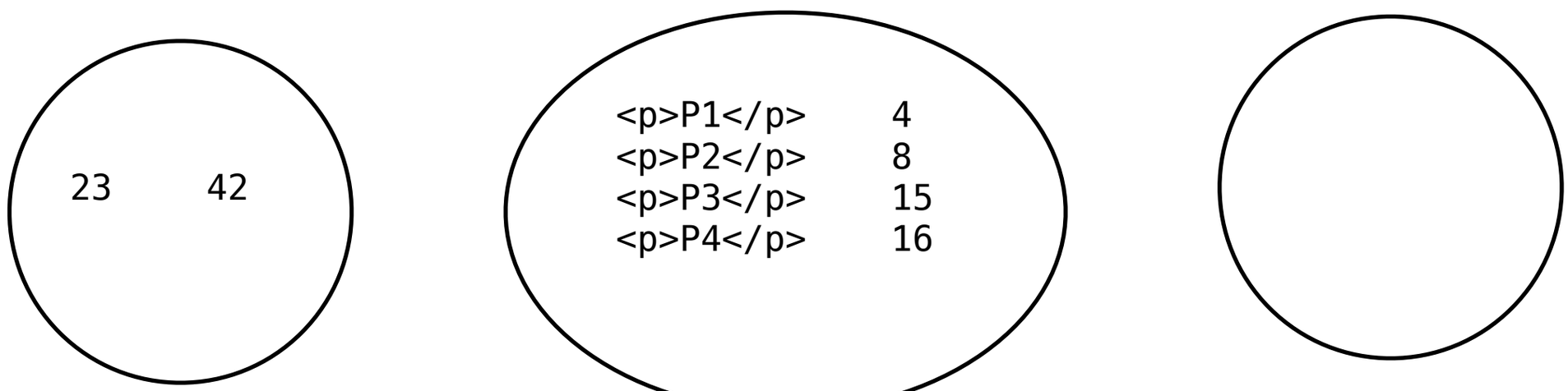
Handling Changing Data

- React:
  - Make components, bind components to state, update state
- D3:
  - Need to provide more control to rendering
    - E.g.: What if I want to highlight data that is new?

Thinking in Joins

```
<p>P1</p>
<p>P2</p>
<p>P3</p>
<p>P4</p>
```

```
var p = d3.selectAll("p")
    .data([4, 8, 15, 16, 23, 42])
```



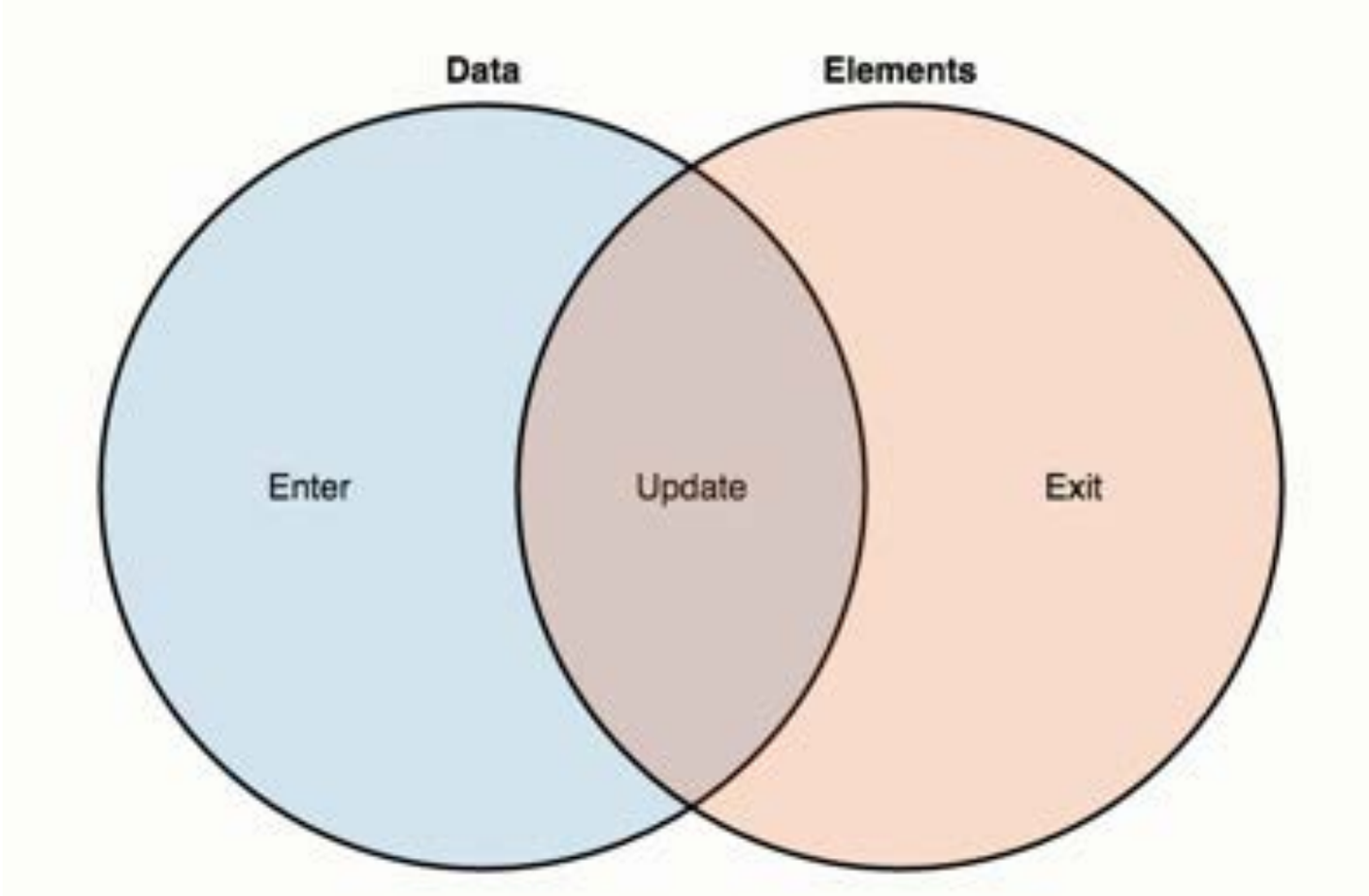

```
.enter(…)        // update (default)        .exit(…)
```

Stuff not on left Stuff in both Stuff not on right

- Elements in selection set undergo data join with elements in data

https://bost.ocks.org/mike/join/

Thinking in joins

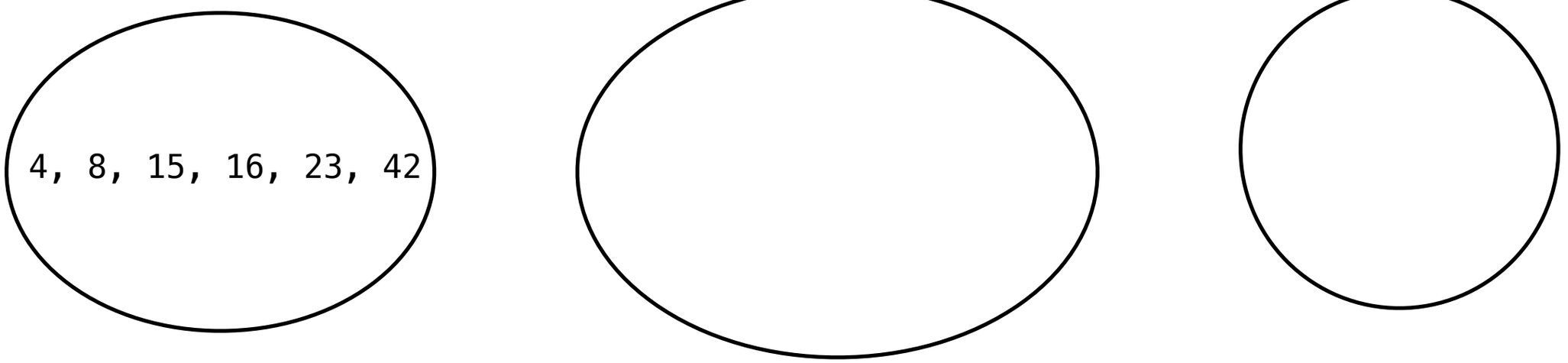


- Extra data —> enter set
- Matched data with elements —> update set
- Extra elements —> exit set

Thinking in Joins

```
<!– no p elements –>
```

```
var p = d3.selectAll("p")
    .data([4, 8, 15, 16, 23, 42])
```




```
.enter(…)
```

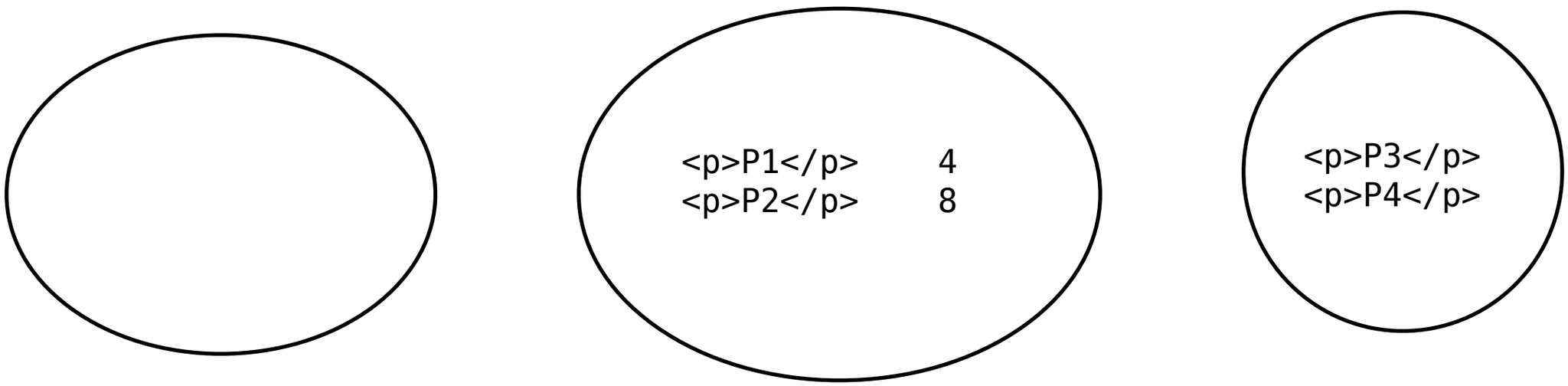
```
// update (default)
```

```
.exit(…)
```

Thinking in Joins

```
<p>P1</p>
<p>P2</p>
<p>P3</p>
<p>P4</p>
```

```
var p = d3.selectAll("p")
    .data([4, 8]);
```




```
.enter(…)        // update (default)        .exit(…)
```

Putting it together

```
// Update…
var p = d3.select("body")
  .selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
    .text(function(d) { return d; });

// Enter…
p.enter().append("p")
    .text(function(d) { return d; });

// Exit…
p.exit().remove();
```

Select all <p> in the <body>

Set the data…

When new data shows up (not in the DOM)

Create it

When data is removed delete it

- Common pattern on data change is to rebind data to elements and separately handle
  - existing elements that should have new visual style (update)
  - new elements that should be created
  - existing elements that should be deleted

Loading data

- What is data?
  - Anything that is an array
    - .data() just cares that it is an array of elements
  - Could be array of numbers, strings, JSON objects
  - If you have a dataset that is an array of JSON objects, pass it to data and you are done

```
.data([{ “a”: 5 }, { “a”: 3}, { “a”: 7 }])
   .text(function(d) { return d.a - 1; });
```

Scaling to fit data

```
.style("width", function(d) { return d * 10 + "px"; });
```

- 10 is a magic number
  - Transforms number in data scale to number in visual representation (“range”) scale
  - Every “1” unit in data should correspond to some unit in output coordinate system
- We’d like to automatically generate reasonable sizes, sizing data to take up all of the space based on range of data, etc.

Scales

```
var x = d3.scale.linear()
    .domain([0, d3.max(data)])
    .range([0, 420]);
```

```
x(4) = 40;
x(2) = 20;
```

- Different types of scales that map domain values (data space) to range values (display space)
- Linear scale uses linear function (e.g., ax + b) to create range value from domain value
- Use:
  - Specify min and max of data
  - Specify min and max of range (output)
  - Generates a function (e.g., x) that will compute range value for a domain value

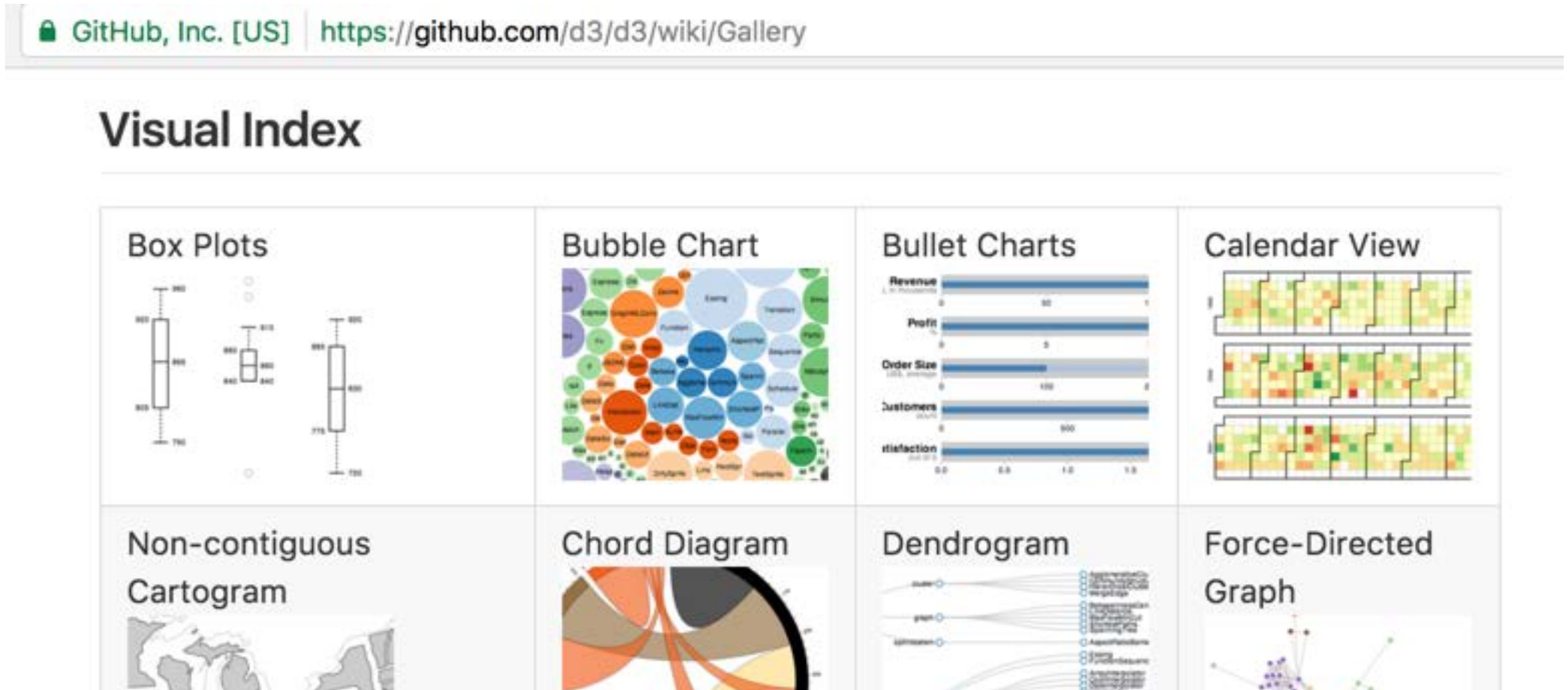
Demo: Generated SVG Bar Chart

http://jsbin.com/baqeyovaho/edit?html,js,output

Using D3

- Best place to start
  - Example code of similar visualization
  - Don’t need to understand *everything*, just enough to make it work

https://github.com/d3/d3/wiki/Gallery



React + D3

- It's a mess
- React and D3 both want to manipulate the DOM
- Options:
  - Let React handle the DOM (D3 can't update it though)
  - Use React's virtual DOM, let D3 update the virtual DOM (not very fast)
  - Ditch D3 or React :/