

Research Topic: Mike Bostock/D3

DATA VISUALIZATION & INFORMATION AESTHETICS

ANTONIE DREYER

09-11-2019

Who is Mike Bostock?

A screenshot of Mike Bostock's Twitter profile. The header image is a colorful, abstract heatmap. Below it, his profile picture is a circular portrait of him with a beard. His name, "Mike Bostock", is followed by a blue verified checkmark. His handle is "@mbostock". His bio reads: "Building a better computational medium. Founder @observablehq. Creator @d3js_org. Former @nytgraphics. Pronounced BOSS-tock." It also lists his location as "San Francisco, CA" and his website as "observablehq.com/@mbostock". On the right side of the profile, there are summary statistics: 9,938 tweets, 2 following, and 59.7K followers. Below these stats are two tabs: "Tweets" and "Tweets & replies". A pinned tweet by Mike Bostock is visible, starting with "To make JavaScript reactive, we".

Mike Bostock 
@mbostock

Building a better computational medium.
Founder [@observablehq](#). Creator
[@d3js_org](#). Former [@nytgraphics](#).
Pronounced BOSS-tock.

📍 San Francisco, CA
🔗 observablehq.com/@mbostock

Tweets 9,938 Following 2 Followers 59.7K

[Tweets](#) [Tweets & replies](#)

Pinned Tweet

Mike Bostock  @mbostock
To make JavaScript reactive, we

- American computer scientist and data-visualization specialist
 - BSE in Computer Science from Princeton
 - Gerald Loeb Award for Images/Graphics (2013-2015)
 - Joint Emmy nominations for work at NY Times
 - Best known for work D3.js (and related libraries)
- Refer to: <https://twitter.com/mbostock>

Who is Mike Bostock?

His Github page: <https://github.com/mbostock>

The screenshot shows Mike Bostock's GitHub profile. At the top, there is a large photo of him with a beard. Below the photo, his name "Mike Bostock" and handle "mbostock" are displayed. A "PRO" badge is present. His bio states: "Building a better computational medium. Founder @observablehq. Creator @d3. Former @nytgraphics. Pronounced BOSS-tock." Below the bio, social links for "@observablehq" (with a location icon), "San Francisco, CA", "Sign in to view email", and a link to "<https://bostocks.org/mike/>" are shown. The main content area has tabs for "Overview", "Repositories 62", "Projects 0", "Stars 42", "Followers 19.8k", and "Following 13". The "Overview" tab is selected. Under "Pinned", there is a box for the "d3/d3" repository, which is described as "Bring data to life with SVG, Canvas, and HTML". It shows 87.2k stars and 21.3k forks. A red arrow points to the star and fork counts. Below this, a chart titled "1,186 contributions in the last year" shows a heatmap of contributions by month and day of the week. A legend indicates that darker shades of green represent more contributions. At the bottom, a "Contribution activity" section shows a timeline from September 2018 to September 2019, with a blue button for "2019" and a "2018" link.

Who is Mike Bostock?

- According to Chris Wiggins, the NY Times Chief Data Scientist, Mike Bostock is a “Digital Superstar (<https://www.kdnuggets.com/2015/01/exclusive-interview-chris-wiggins-nytimes-chief-data-scientist.html>)
- Edward Tufte said in a 2013 interview with the Financial times that Mike Bostock will be one of the most prominent individuals in the field of Data Visualization (Cookson, Clive (2013-07-26). [*"Edward Tufte". FT Magazine. ISSN 0307-1766*](#) Retrieved 09-09-2019)

What does Mike do? (History)

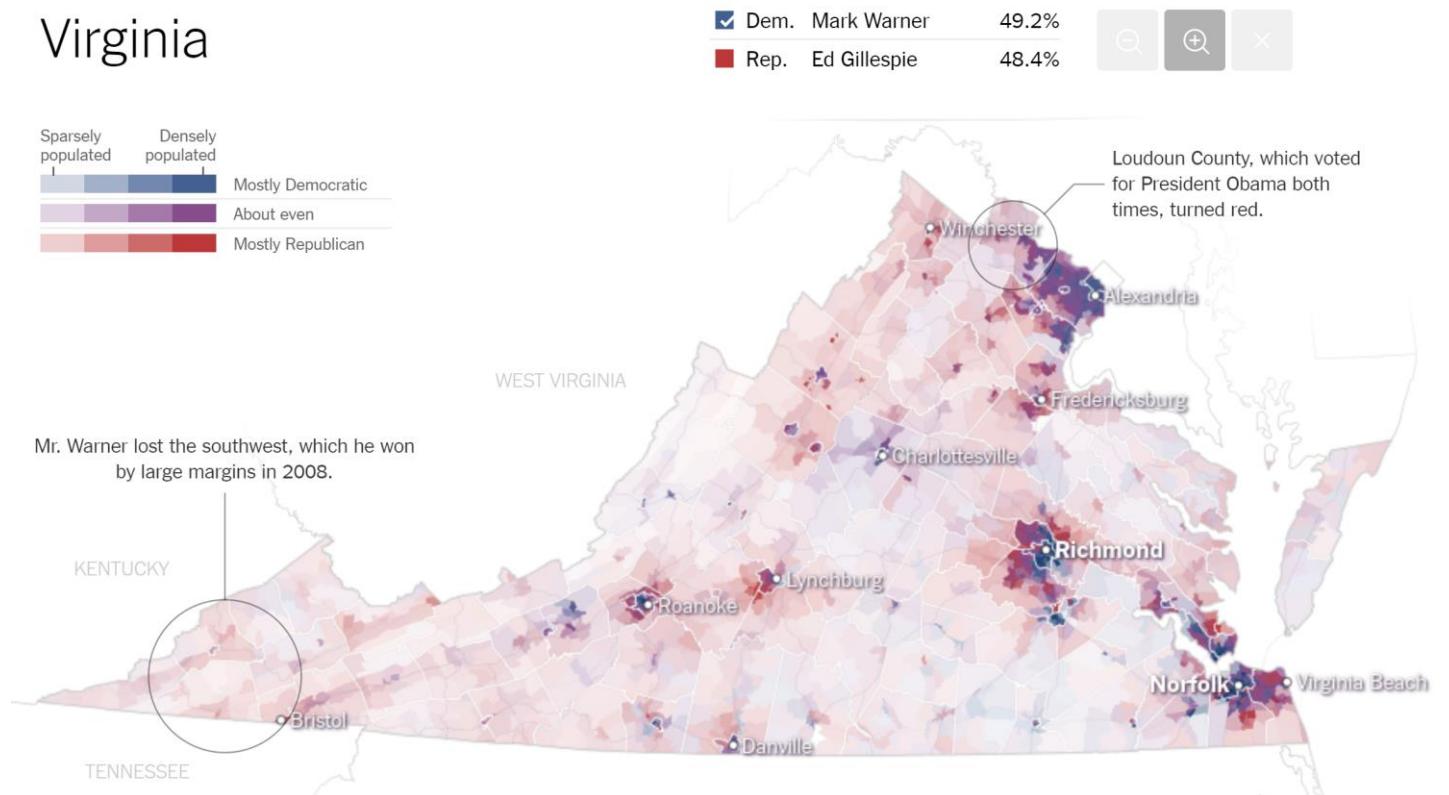
- Previously a PhD student at Stanford University (advised by Jeffrey Heer <http://vis.stanford.edu/>)
- Was a member of the notable Stanford Visualization Group (the group's work included Polaris; the precursor to Tableau Software; <http://idl.cs.washington.edu/about>)
- While at Stanford Mike created the Protovis JavaScript library with Jeff Heer, and Vadim Ogievetsky (<http://mbostock.github.io/protovis/>)
- Worked at the New York Times leading data-visualization projects until 2015
- Adviser to data transformation platform Trifacta (<https://www.trifacta.com/board-directors-advisors/>)
- Currently working toward a more accessible coding platform (<https://observablehq.com/>)

Visualization at the New York Times

<https://www.nytimes.com/interactive/2014/11/04/upshot/senate-maps.html>

The Most Detailed Maps
You'll See From the
Midterm Elections*

By Amanda Cox, Mike Bostock, Derek Watkins, and Shan Carter Nov. 6, 2014 04:03 PM



Visualization at the New York Times

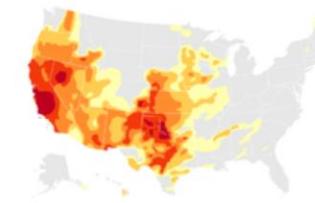
Recent and Archived work (<https://www.nytimes.com/by/mike-Bostock>)

July 24, 2014

Mapping the Spread of Drought Across the U.S.

Maps and charts updated weekly show the latest extent of the drought in the United States.

By MIKE BOSTOCK and KEVIN QUEALY



July 1, 2014

Quiz: Can You Tell What Makes a Good Tweet?

A quiz tests your skill at picking which of two tweets would be shared more.

By MIKE BOSTOCK, JOSH KATZ and NILKANTH PATEL

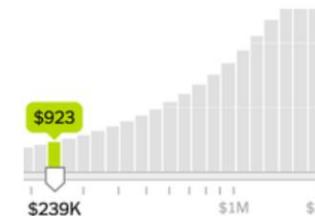


May 22, 2014

Is It Better to Rent or Buy?

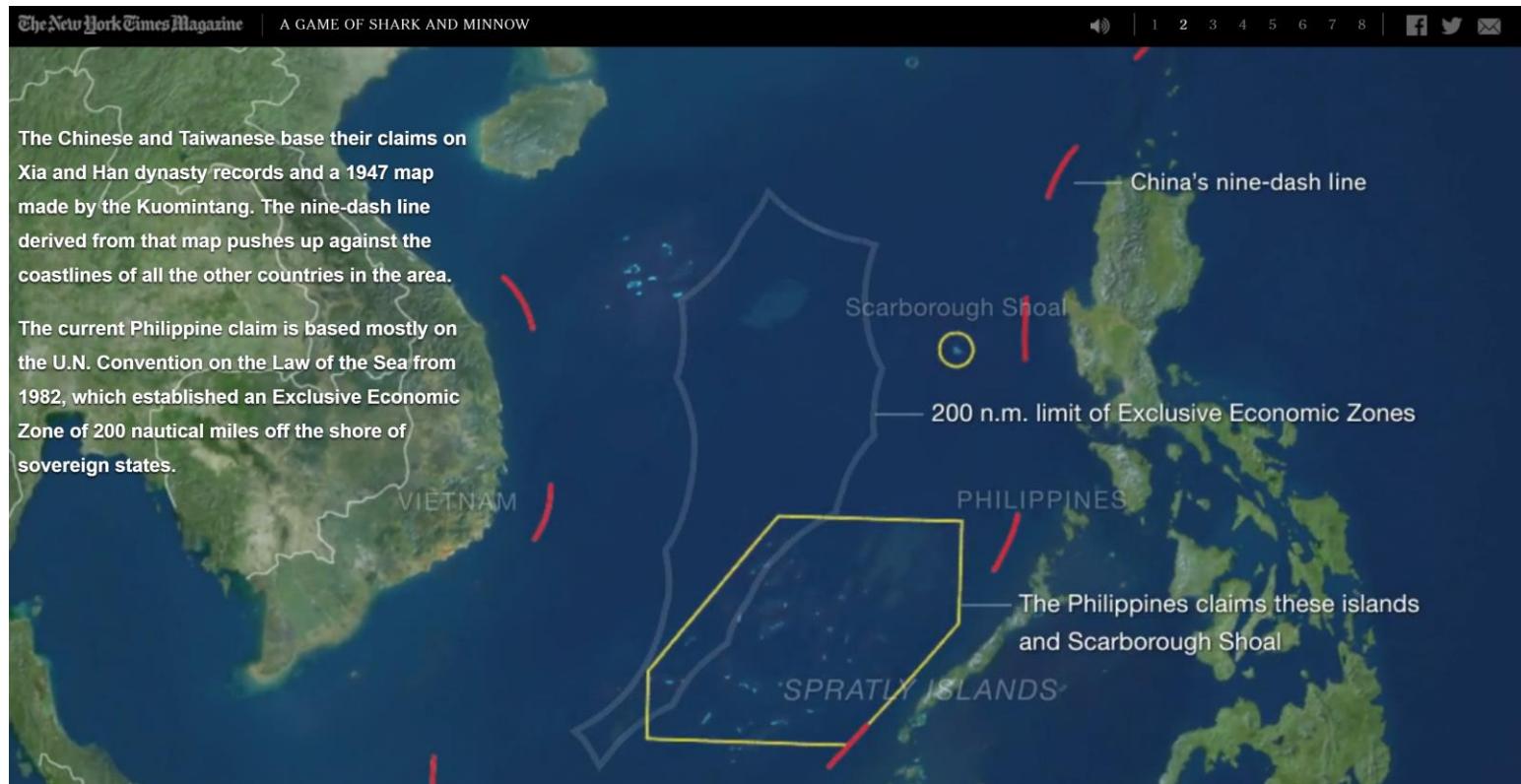
The choice between buying a home and renting one is among the biggest financial decisions that many adults make.

By Mike Bostock, Shan Carter and Archie Tse



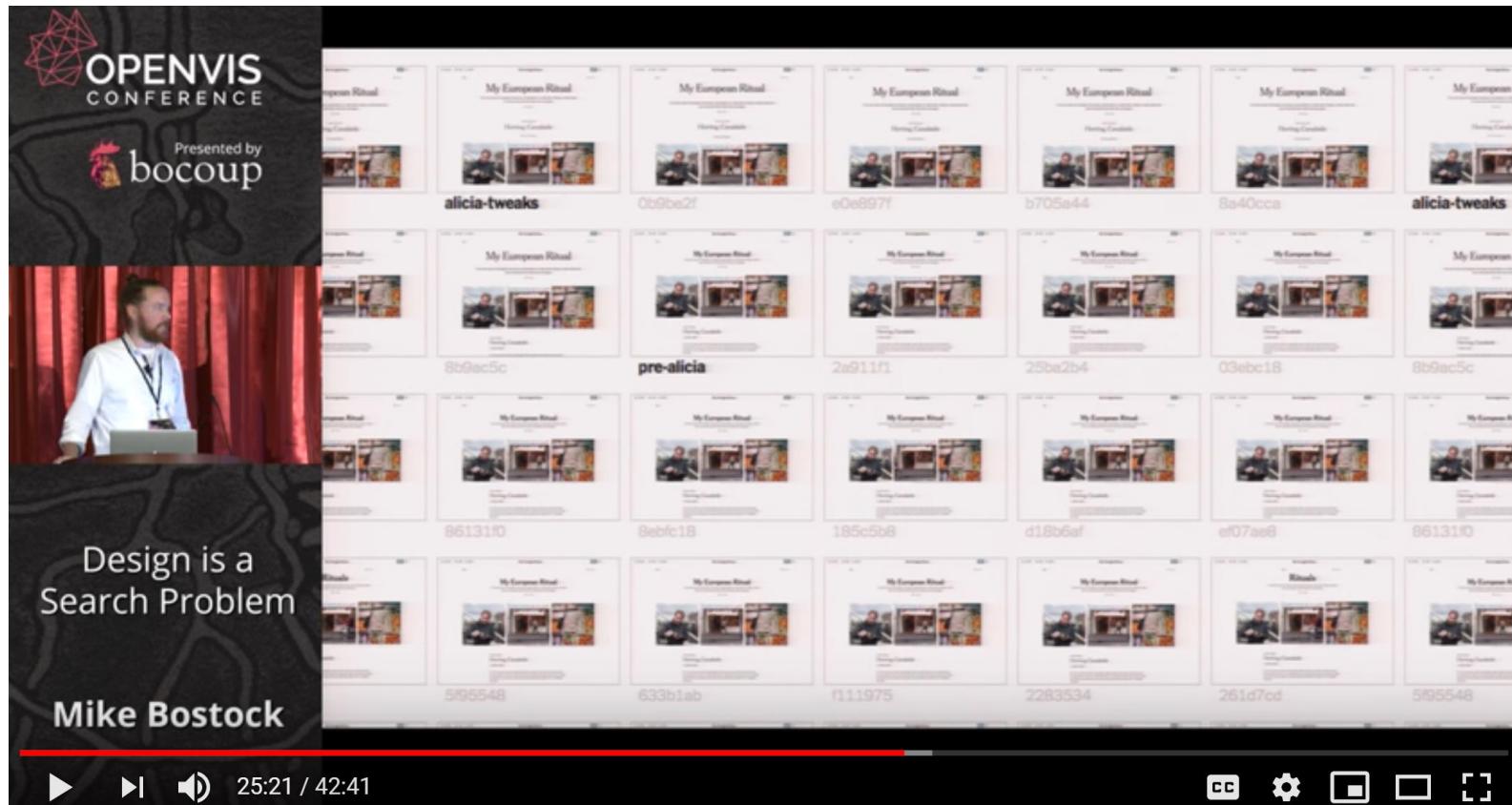
Visualization at the New York Times

Emmy nomination (<http://www.nytimes.com/newsgraphics/2013/10/27/south-china-sea/index.html>)



Visualization at the New York Times

Innovator that worked with Amanda Cox (<https://www.youtube.com/watch?v=fThhbt23SGM>)



Style & purpose of his work

- Protovis; the predecessor of D3.js
(<http://mbostock.github.io/protovis/>)
- “extensible toolkit for constructing visualizations by composing simple graphical primitives”
- “designers specify visualizations as a hierarchy of marks with visual properties defined as functions of data”

M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. IEEE Trans Visualization & Comp Graphics, 15(6):1121–1128, 2009

Protovis: A Graphical Toolkit for Visualization

Michael Bostock and Jeffrey Heer

Abstract—Despite myriad tools for visualizing data, there remains a gap between the notational efficiency of high-level visualization systems and the expressiveness and accessibility of low-level graphical systems. Powerful visualization systems may be inflexible or impose abstractions foreign to visual thinking, while graphical systems such as rendering APIs and vector-based drawing programs are tedious for complex work. We argue that an easy-to-use graphical system tailored for visualization is needed. In response, we contribute Protovis, an extensible toolkit for constructing visualizations by composing simple graphical primitives. In Protovis, designers specify visualizations as a hierarchy of marks with visual properties defined as functions of data. This representation achieves a level of expressiveness comparable to low-level graphics systems, while improving *efficiency*—the effort required to specify a visualization—and *accessibility*—the effort required to learn and modify the representation. We substantiate this claim through a diverse collection of examples and comparative analysis with popular visualization tools.

Index Terms—Information visualization, user interfaces, toolkits, 2D graphics.

1 INTRODUCTION

A popular approach to visualization is to import data into charting software, specify a desired chart type, and then tweak visual parameters as needed to produce the final graphic. Modern charting software may support a dozen or more chart types, such as pie and line, while supporting numerous customizable visual parameters, such as color and font. As noted by Wilkinson [37], this approach is especially popular in user interfaces, where often a chart can be produced in a few clicks.

Chart typologies are successful because they are quick and easy to use, but suffer simultaneously because they are highly restrictive. Many visualizations cannot be made simply because they are not one of the supported types. In addition, despite customization, designers may be unable to control the precise graphical output. A typological grammar limits the space of possible visualizations, as compared to what is possible in more general graphical systems.

As a result, designers may resort to vector-based drawing programs to realize their intent [31]. This is unfortunate; while such programs offer the utmost flexibility, they are not tailored for visualization. Drawing vector graphics by hand is time-consuming and error-prone, and even with the ability to import or generate simple graphics from data, the process often cannot support interaction and live data.

High-level chart types and low-level vector drawing represent two extremes, but in practice designers choose between many different systems, considering *expressiveness* (“Can I build it?”), *efficiency* (“How long will it take?”) and *accessibility* (“Do I know how?”). The choice of tool affects the resulting work, as it biases designers towards visualizations that are easier to produce in the given tool. As Maslow [23] famously quipped, “I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.”

systems are easy to learn but tedious for complex work, while powerful visualization systems can be intimidating to novices or inflexible.

Moreover, the abstractions used by visualization systems may be foreign to designers. While vector graphics editors allow designers to think concretely in terms of graphical marks, most expressive visualization tools make use of abstract specifications of data processing and visual encoding operators. Such systems require that designers translate their intended visual design into toolkit abstractions, often hindering accessibility.

In response, we contribute Protovis, an embedded domain-specific language [19] for constructing visualizations by composing simple graphical marks such as bars, lines and labels. In Protovis, designers specify visualizations as a hierarchy of marks with visual properties defined as functions of data. Inheritance of properties across composed marks—similar to cascading of style sheets used in web design—enables concise visualization definitions with a large expressive range and a minimum of intervening abstractions. Protovis is implemented in JavaScript, with rendering support for HTML 5 canvas, SVG, and Flash.

To evaluate Protovis, we built example applications demonstrating the toolkit’s expressiveness and notational efficiency. We use these examples as points of comparison with Processing and Flare, two popular visualization tools. To assess accessibility, we present a comparative analysis using the Cognitive Dimensions of Notation framework [13] and share feedback from designers using Protovis.

2 RELATED WORK

For the purpose of comparison we divide tools used to visualize data

Examples: Protopvis

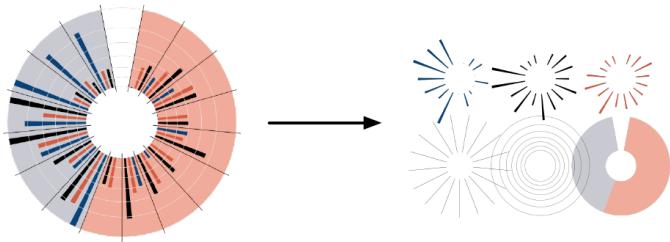


Fig. 1. Decomposing a visualization into marks.

```

new pv.Panel().canvas("fig3a")
  .add(pv.Bar)
  .data([1, 1.2, 1.7, 1.5, .7, .2])
  .bottom(0).width(20)
  .height(function(d) d * 80)
  .left(function() this.index * 25)
  .root.render();

new pv.Panel().canvas("fig3b")
  .data([
    [1, 1.2, 1.7, 1.5, .7],
    [.5, 1, .8, 1.1, 1.3],
    [.2, .5, .8, .9, 1.1]
  ])
  .add(pv.Area)
  .data(function(d) d)
  .fillStyle(pv.Colors.category19.parent)
  .bottom(function() let c = this.cousin()
    c ? (c.bottom + c.height) : 0)
  .height(function(d) d * 40)
  .left(function() this.index * 35)
  .root.render();

```

Fig. 2. Specifying two simple charts. (a) Bar. (b) Stacked area.

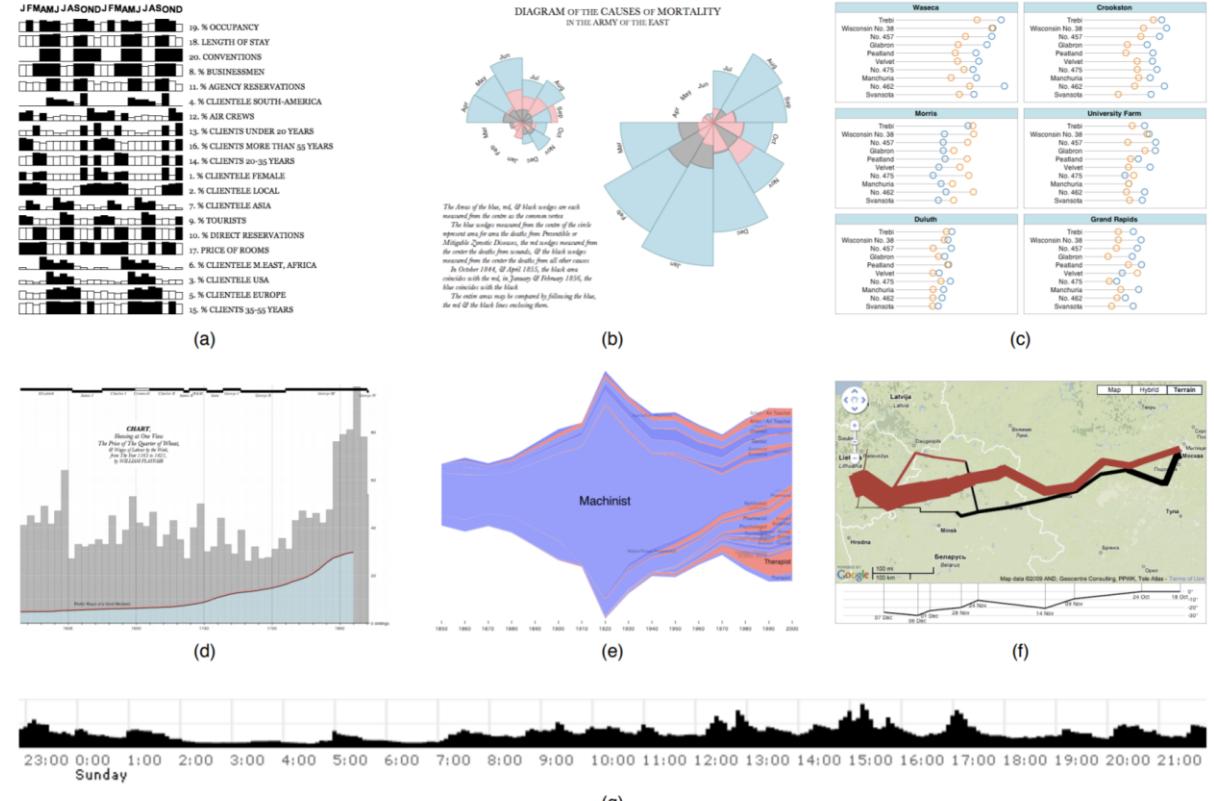


Fig. 6. Example applications. (a) Jacques Bertin's analysis of hotel patterns. (b) Florence Nightingale's chart of deaths in the Crimean War. (c) A trellis display of barley yields. (d) William Playfair's chart comparing the price of wheat and wages. (e) The Job Voyager. (f) Charles Minard's flow map of Napoleon's march to Moscow, as a Google Maps "mash-up". (g) Live Twitter updates containing the word "oakland".

Style & purpose of his work

- Visualization primitives (Bertin's retinals)
- Focuses on browser performance (rendering speed)
- Make his work relatable; e.g. Charles Minard's

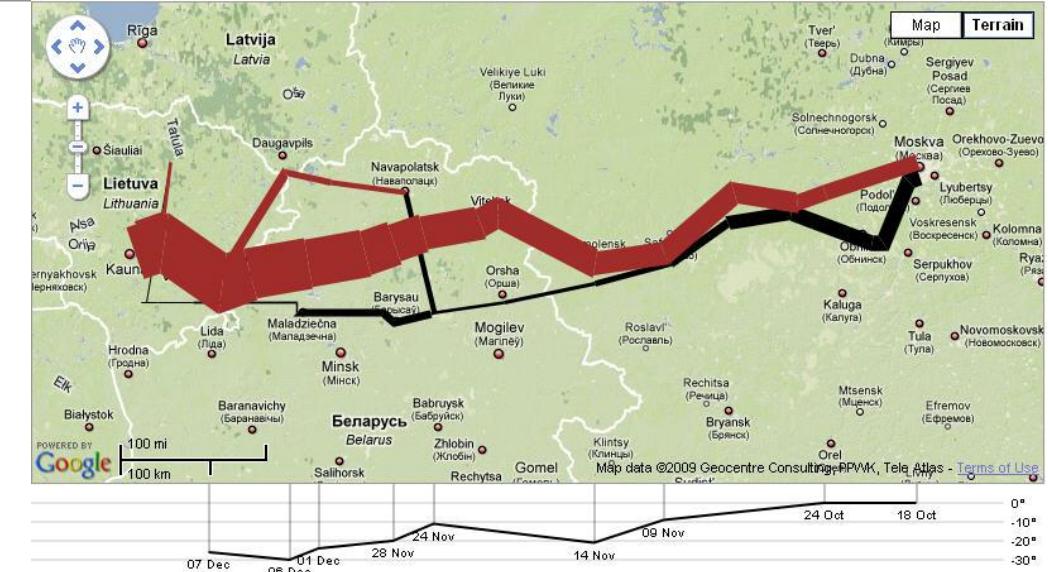
Map of Napoleon's army On Google Maps

(<https://queue.acm.org/detail.cfm?id=1805128>)

- Uses simple examples to explain complex matter

“The power to understand and predict the quantities of the world should not be restricted to those with a freakish knack for manipulating abstract symbols.” - Bret Victor

“Improving the human experience of coding is not just about making your workflow more convenient or efficient. It empowers people to better understand their world.” – Mike Bostock



D3: successor to Protopis



“D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. For example, you can use D3 to generate an HTML table from an array of numbers. Or, use the same data to create an interactive SVG bar chart with smooth transitions and interaction.”

<https://d3js.org/>

Some details of D3



D3: Data-Driven Documents

D3 (or **D3.js**) is a JavaScript library for visualizing data using web standards. D3 helps you bring data to life using SVG, Canvas and HTML. D3 combines powerful visualization and interaction techniques with a data-driven approach to DOM manipulation, giving you the full capabilities of modern browsers and the freedom to design the right visual interface for your data. (<https://github.com/d3/d3>)

- “The goal of D3 is to embrace and build on standard, universal representations of graphics”, rather than “reinventing the wheel each time”. (2011) (<https://chartio.com/blog/mike-bostock-interview/>)
- Focuses on web standards and provides improved performance

Some details of D3

“D3 is not a monolithic framework that seeks to provide every conceivable feature. Instead, D3 solves the crux of the problem: efficient manipulation of documents based on data. ... With minimal overhead, D3 is extremely fast, supporting large datasets and dynamic behaviors for interaction and animation.”

“Modifying documents using the W3C DOM API is tedious: the method names are verbose, and the imperative approach requires manual iteration and bookkeeping of temporary state.”

```
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs.item(i);
    paragraph.style.setProperty("color", "blue", null);
}
```

“D3 employs a declarative approach, operating on arbitrary sets of nodes called selections. For example, you can rewrite the above loop as:

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

<https://d3js.org/>

Some details of D3

“D3 provides numerous methods for mutating nodes: setting attributes or styles; registering event listeners; adding, removing or sorting nodes; and changing HTML or text content. These suffice for the vast majority of needs.”

“D3 lets you transform documents based on data; this includes both creating and destroying elements. D3 allows you to change an existing document in response to user interaction, animation over time, or even asynchronous notification from a third-party.”

“D3’s focus on transformation extends naturally to animated transitions.”

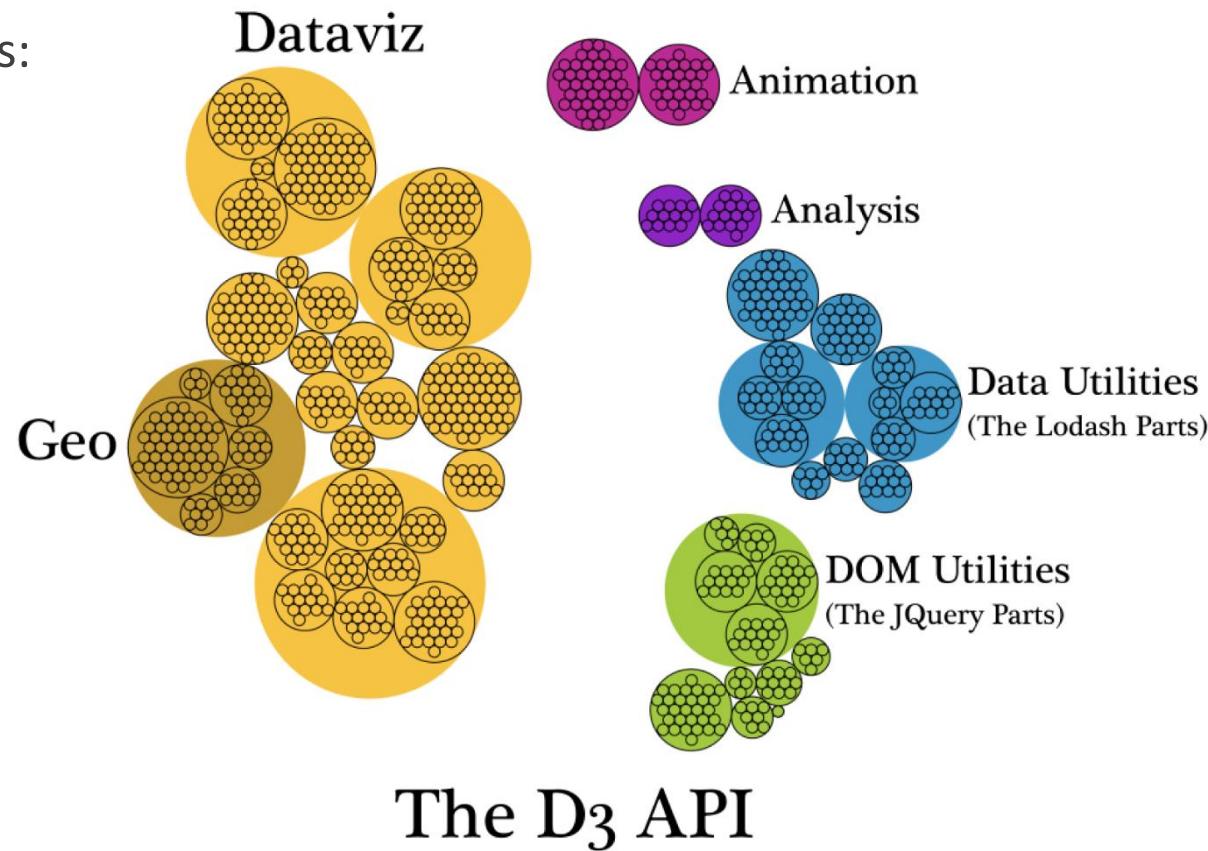
<https://d3js.org/>

The D3 API page:

<https://github.com/d3/d3/blob/master/API.md>

D3 Details

Summary of functions:



from Elijah's [D3 is not a Data Visualization Library](#) article.

https://medium.com/@Elijah_Meeks/d3-is-not-a-data-visualization-library-67ba549e8520

D3 Details

The DOM utilities (JQuery parts):

Selections (d3-selection)

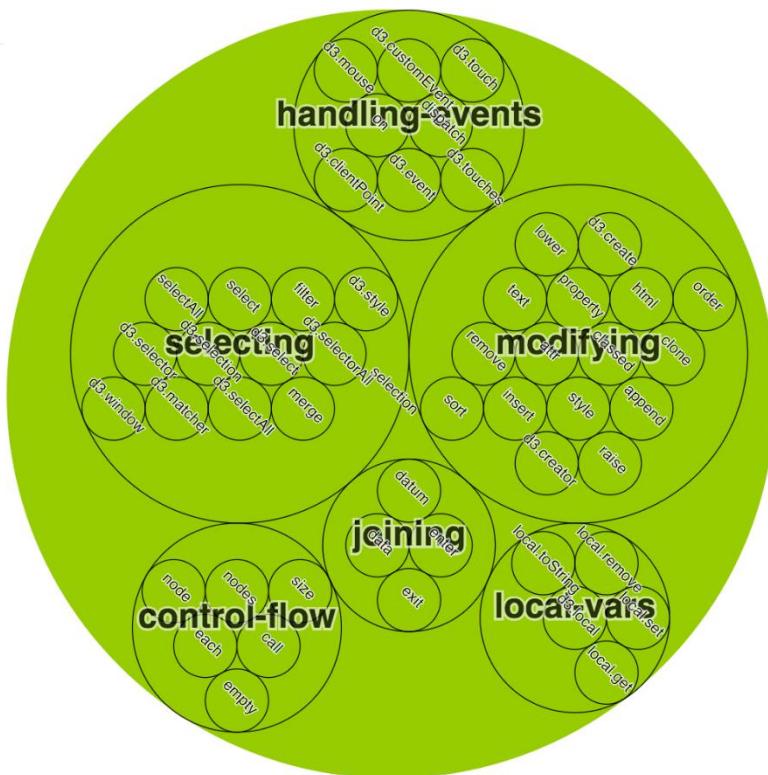
Transform the DOM by selecting elements and joining to data.

Selecting Elements

- `d3.selection` - select the root document element.
 - `d3.select` - select an element from the document.
 - `d3.selectAll` - select multiple elements from the document.
 - `selection.select` - select a descendant element for each selected element.
 - `selection.selectAll` - select multiple descendants for each selected element.
 - `selection.filter` - filter elements based on data.
 - `selection.merge` - merge this selection with another.
 - `d3.matcher` - test whether an element matches a selector.
 - `d3.selector` - select an element.
 - `d3.selectorAll` - select elements.
 - `d3.window` - get a node's owner window.
 - `d3.style` - get a node's current style value.

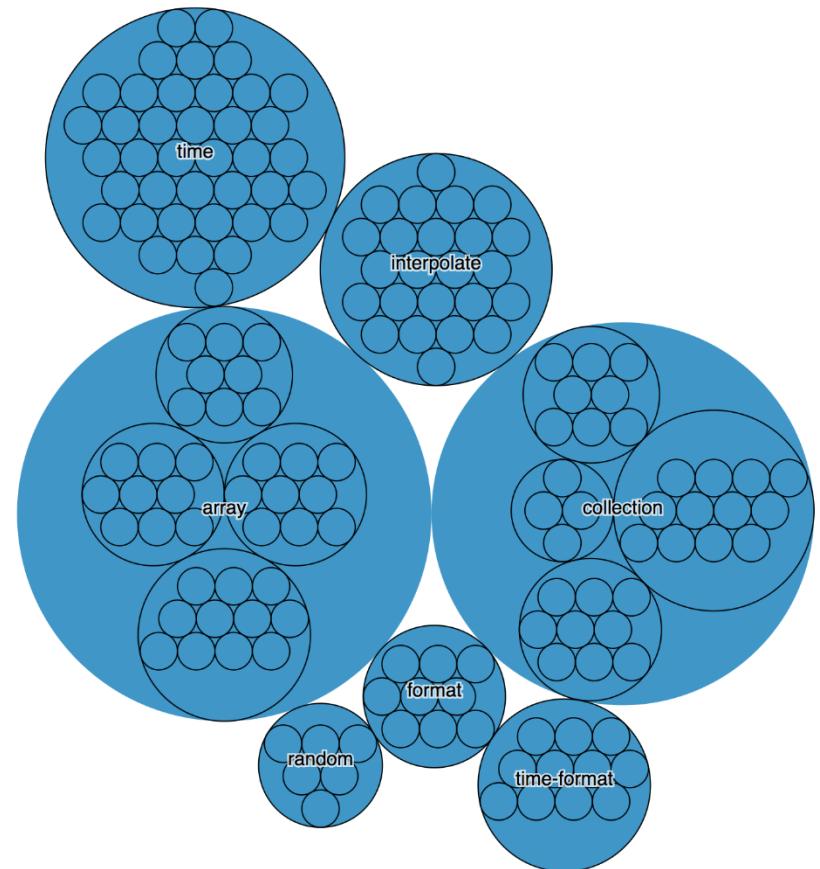
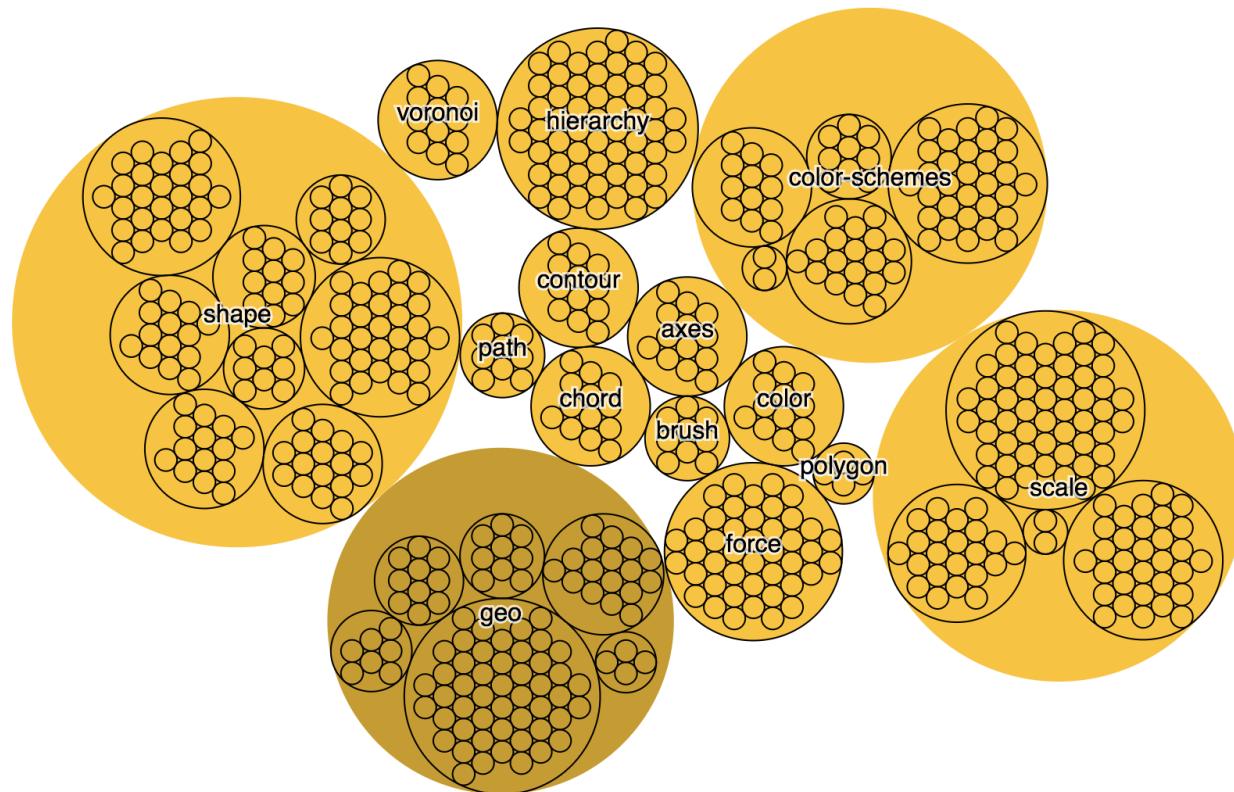
Modifying Elements

- `selection.attr` - get or set an attribute.
 - `selection.classed` - get, add or remove CSS classes.
 - `selection.style` - get or set a style property.
 - `selection.property` - get or set a (raw) property.
 - `selection.text` - get or set the text content.
 - `selection.html` - get or set the inner HTML.
 - `selection.append` - create, append and select new elements.
 - `selection.insert` - create, insert and select new elements.
 - `selection.remove` - remove elements from the document.
 - `selection.clone` - insert clones of selected elements.
 - `selection.sort` - sort elements in the document based on data.
 - `selection.order` - reorders elements in the document to match the selection.
 - `selection.raise` - reorders each element as the last child of its parent.
 - `selection.lower` - reorders each element as the first child of its parent.
 - `d3.create` - create and select a detached element.
 - `d3.creator` - create an element by name.



D3 Details

The Data Visualization and Lodash parts:



D3 Tutorials

Tutorials: (<https://github.com/d3/d3/wiki/Tutorials>)

Project index.js

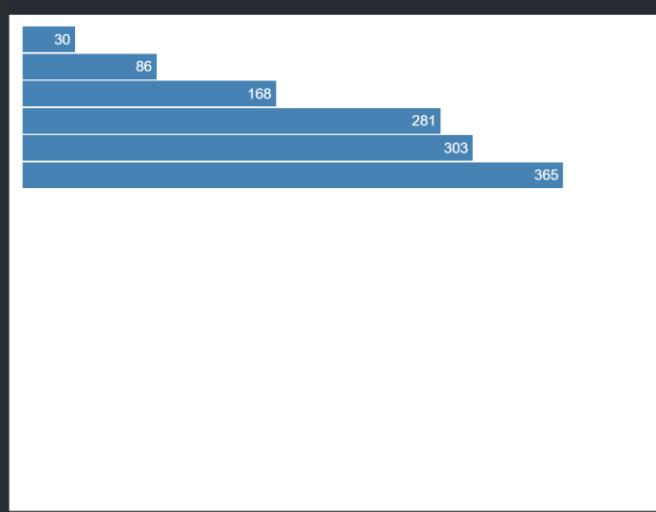
FILES

- # index.css
- index.html
- index.js**

DEPENDENCIES

index.js

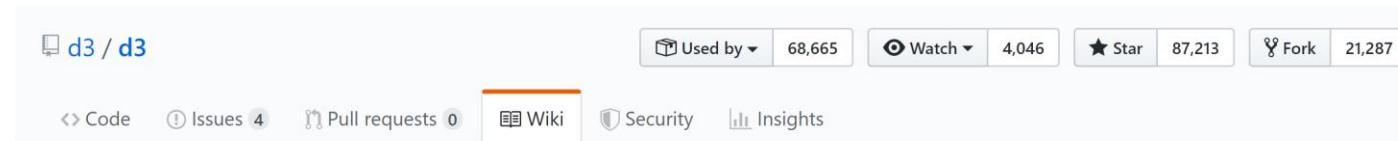
```
1 var data = [30, 86, 168, 281, 303, 365];
2
3 d3.select(".chart")
4   .selectAll("div")
5   .data(data)
6   .enter()
7   .append("div")
8   .style("width", function(d) { return d + "px"; })
9   .text(function(d) { return d; });
```

 448x340 (100%)

RUN CODE / RELOAD SHOW MINI BROWSER

D3 Gallery/Examples

Examples: (<https://github.com/d3/d3/wiki/Gallery>)



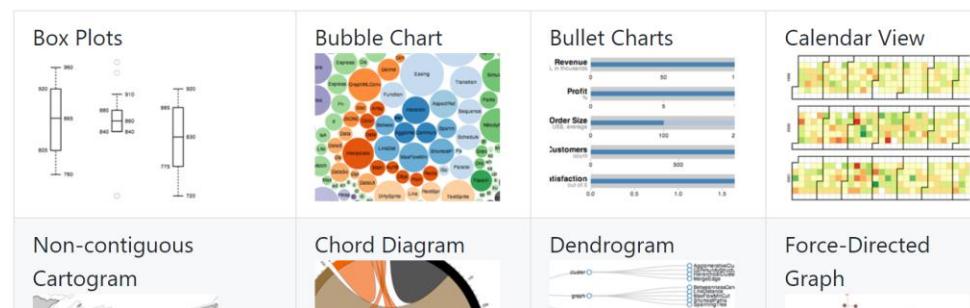
Gallery

Mike Bostock edited this page on Jul 15 · 1292 revisions

Wiki > Gallery

Welcome to the D3 gallery! More examples are available for forking on [Observable](#); see D3's profile and the [visualization collection](#). Please share your work on Observable, or [tweet us a link!](#)

Visual Index



▶ Pages 62



Data-Driven
Documents

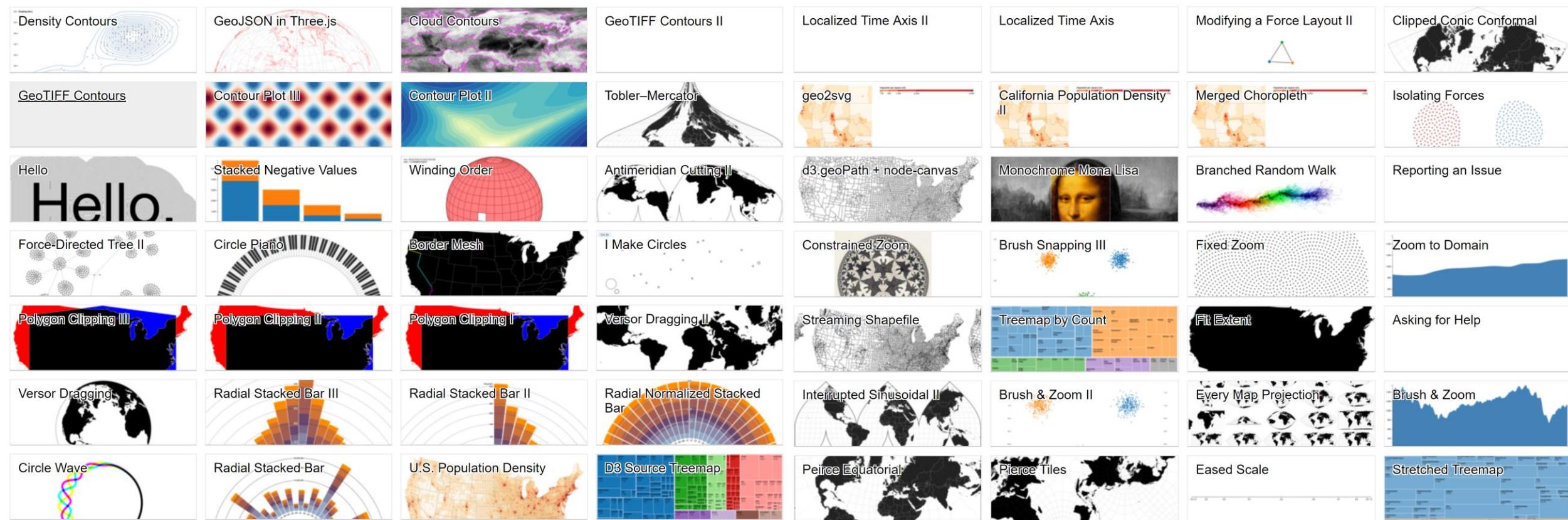
Examples, examples, examples...

“I am a big fan of examples...examples are about demonstrating the potential value of ideas”



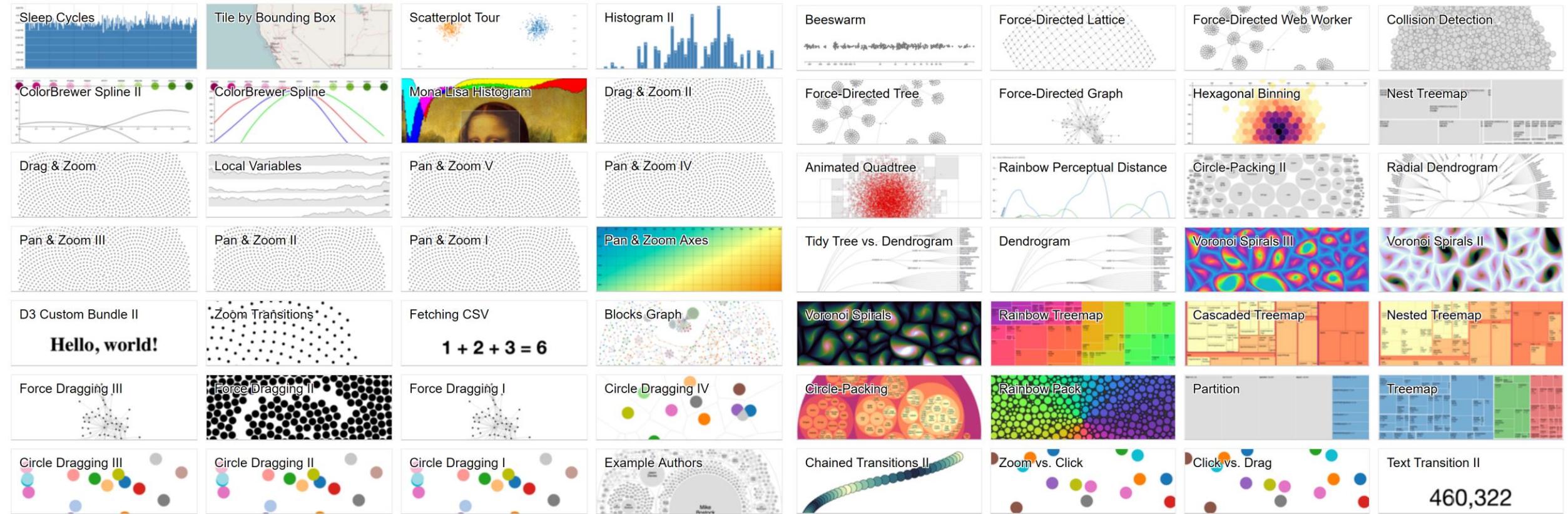
<https://bostocks.org/mike/example/>

Examples, examples, examples...



<https://bl.ocks.org/mbostock>

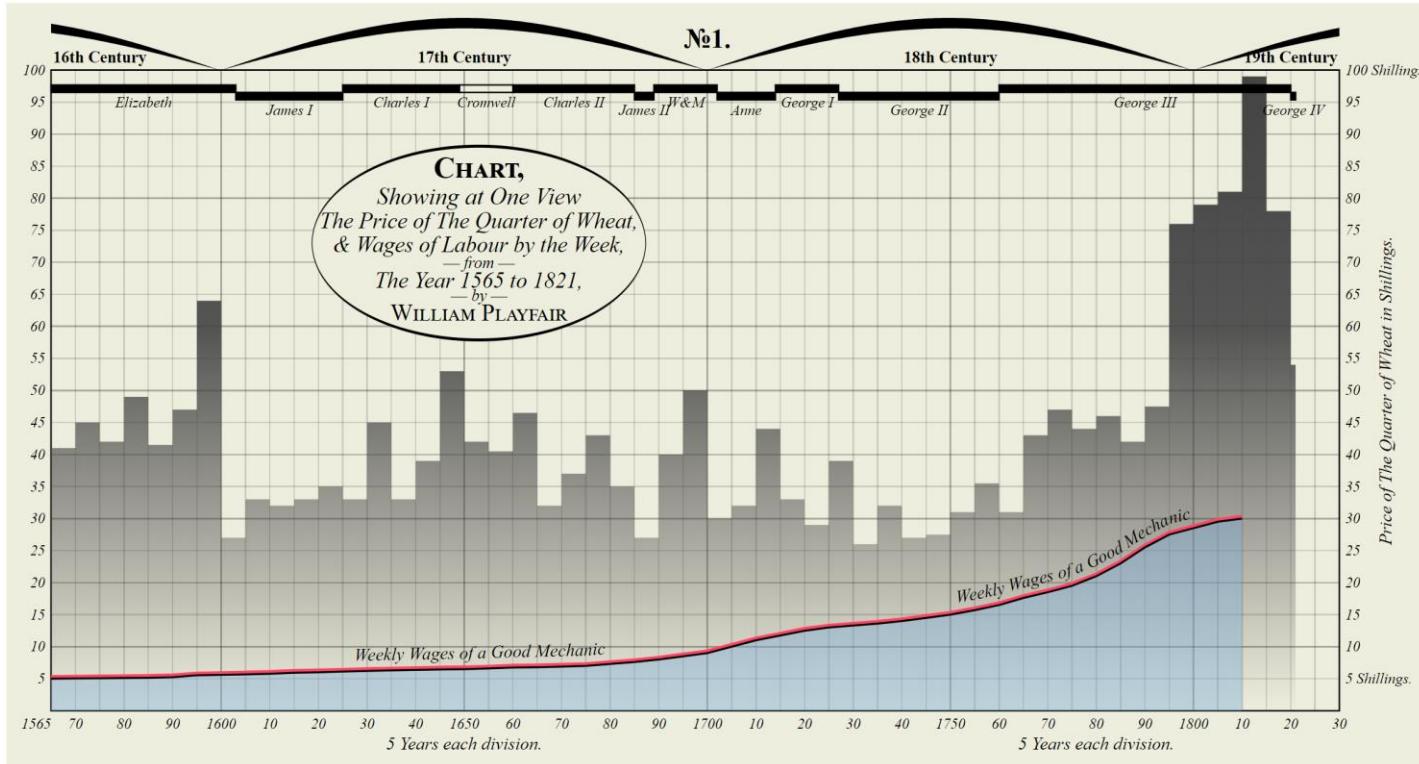
Examples, examples, examples...



<https://bl.ocks.org/mbostock>

Open source user example

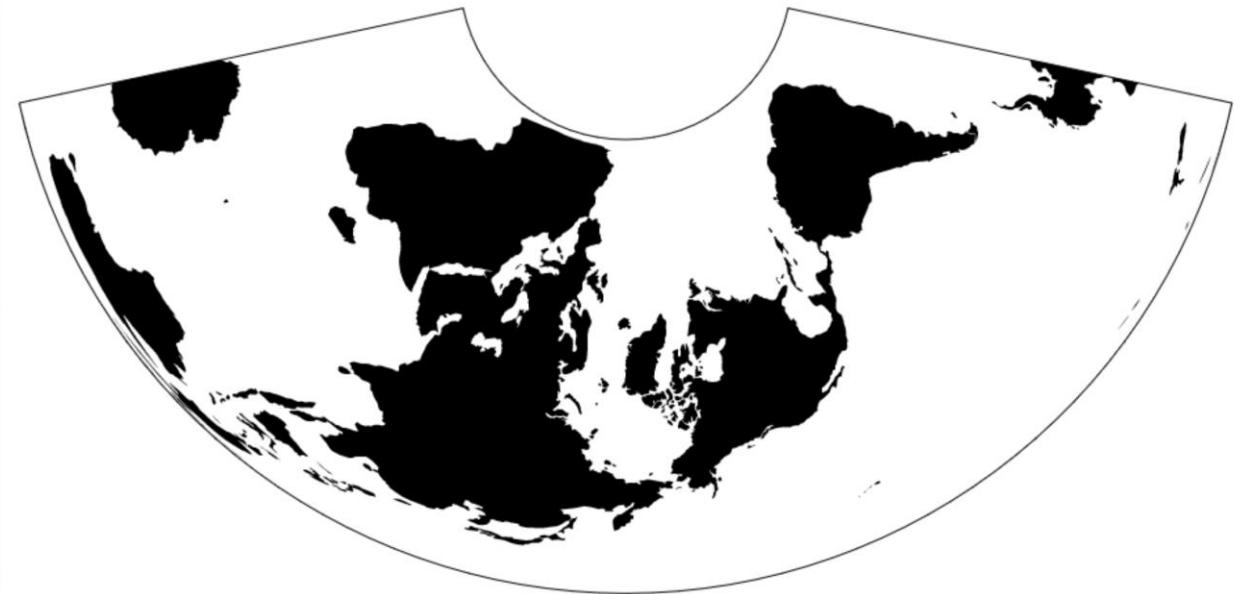
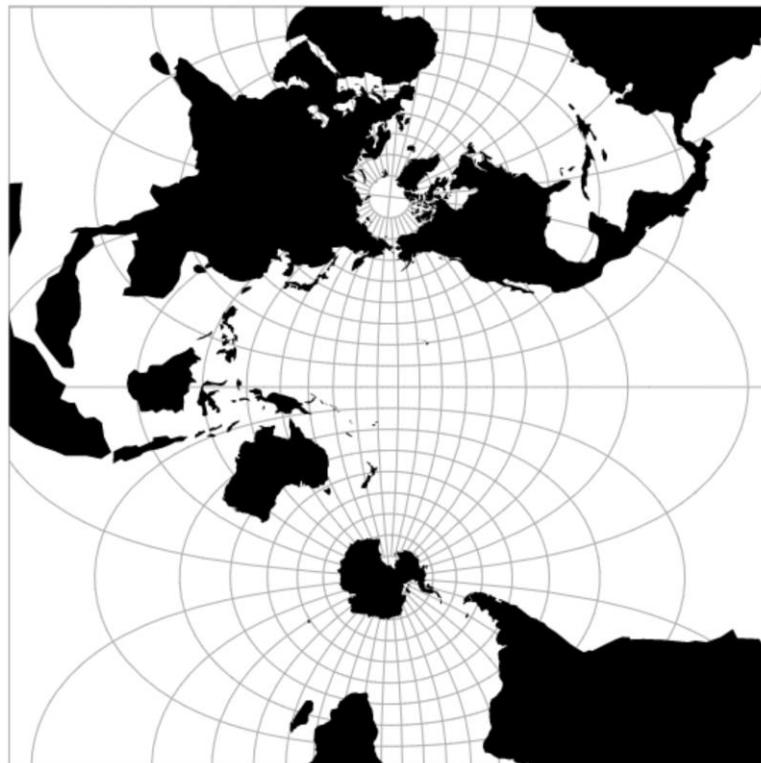
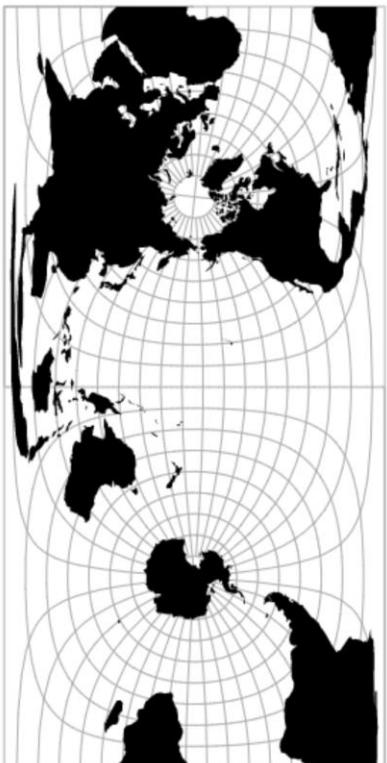
William Playfair's Price of Wheat



<https://blocks.org/borgar/0549b917908d83873c68>

Making the difficult “easy”

- “Make sure the leap from abstract idea to concrete value is a short one.”



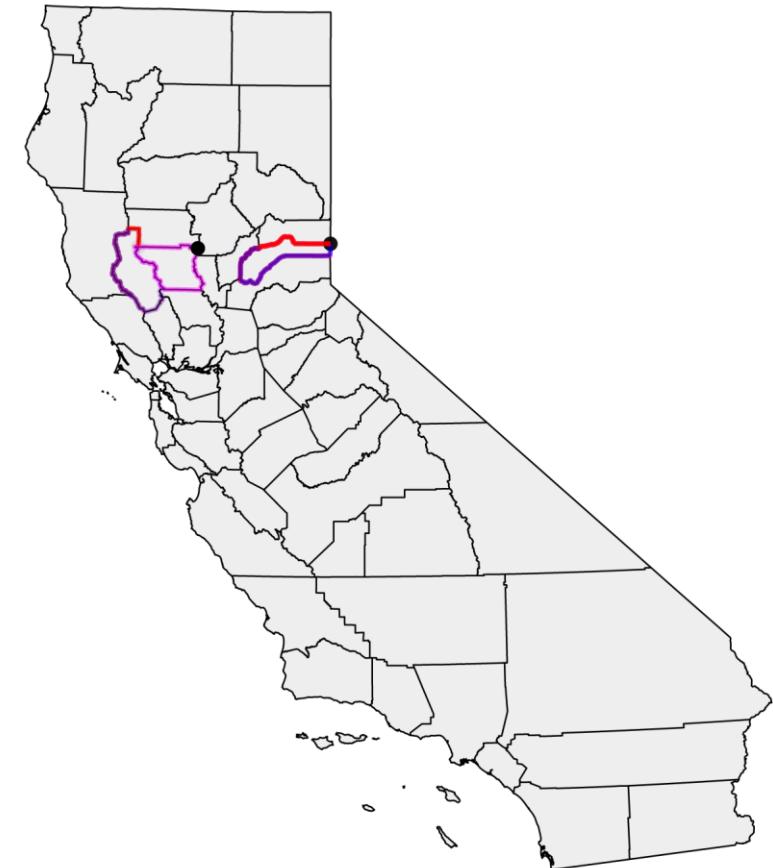
bl.ocks.org/3788999

Making the difficult “easy”

- TopoJSON
(<https://bost.ocks.org/mike/topology/debugger.html>)

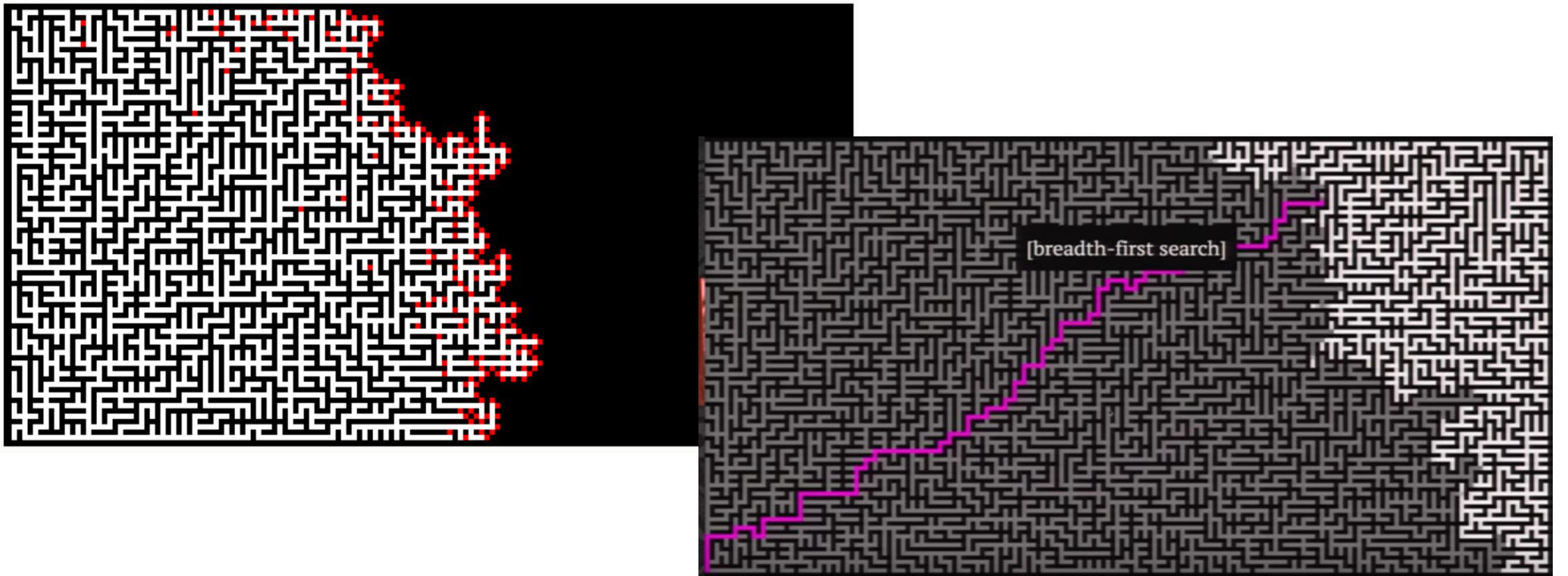
“As its name implies, TopoJSON is a topological geospatial data format.”

“In contrast to a geometry, where each shape is encoded with separate (and often redundant) arrays of coordinates, a topology encodes sequences of coordinates in line fragments called arcs that can be shared.”



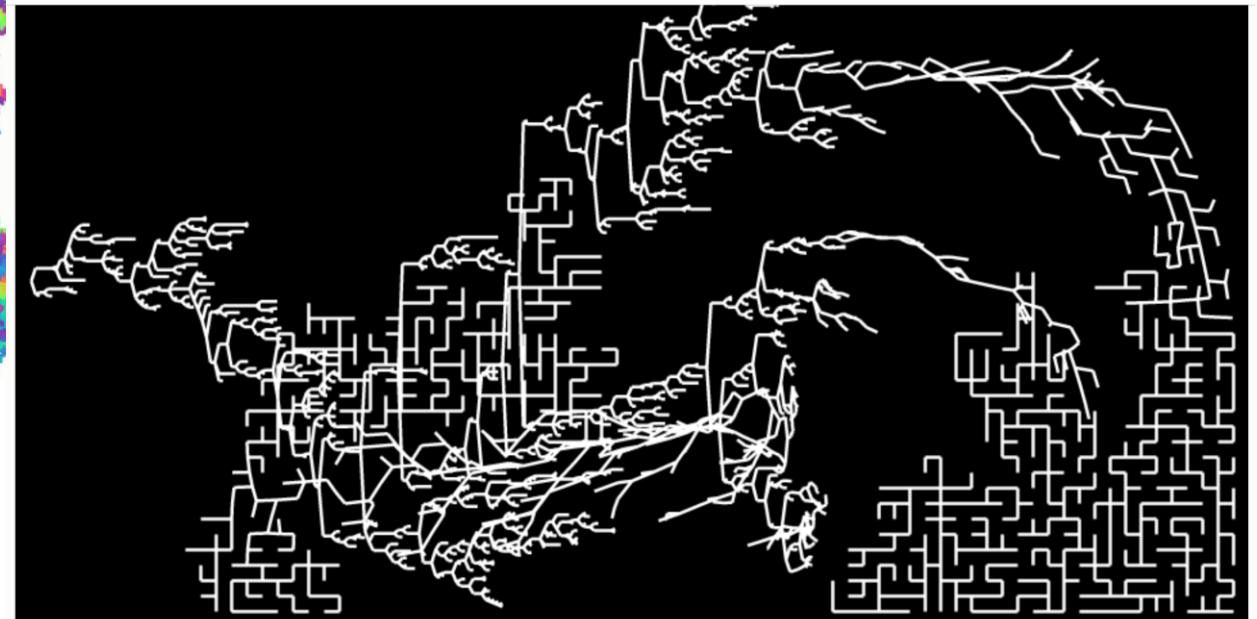
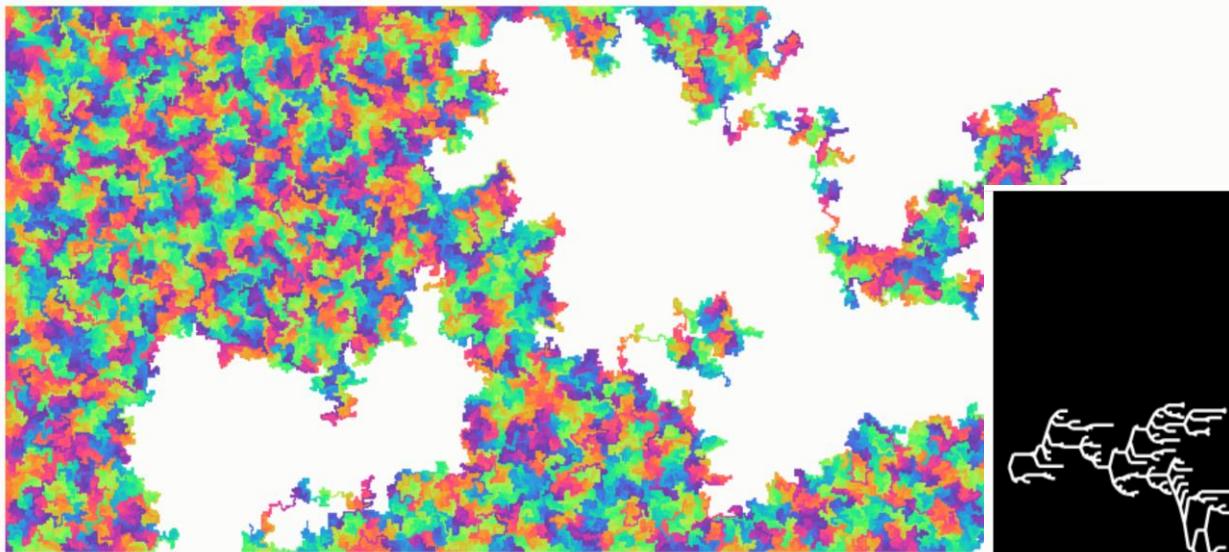
Making the difficult “easy”

Visualizing Algorithms (<https://bost.ocks.org/mike/algorithms/>)



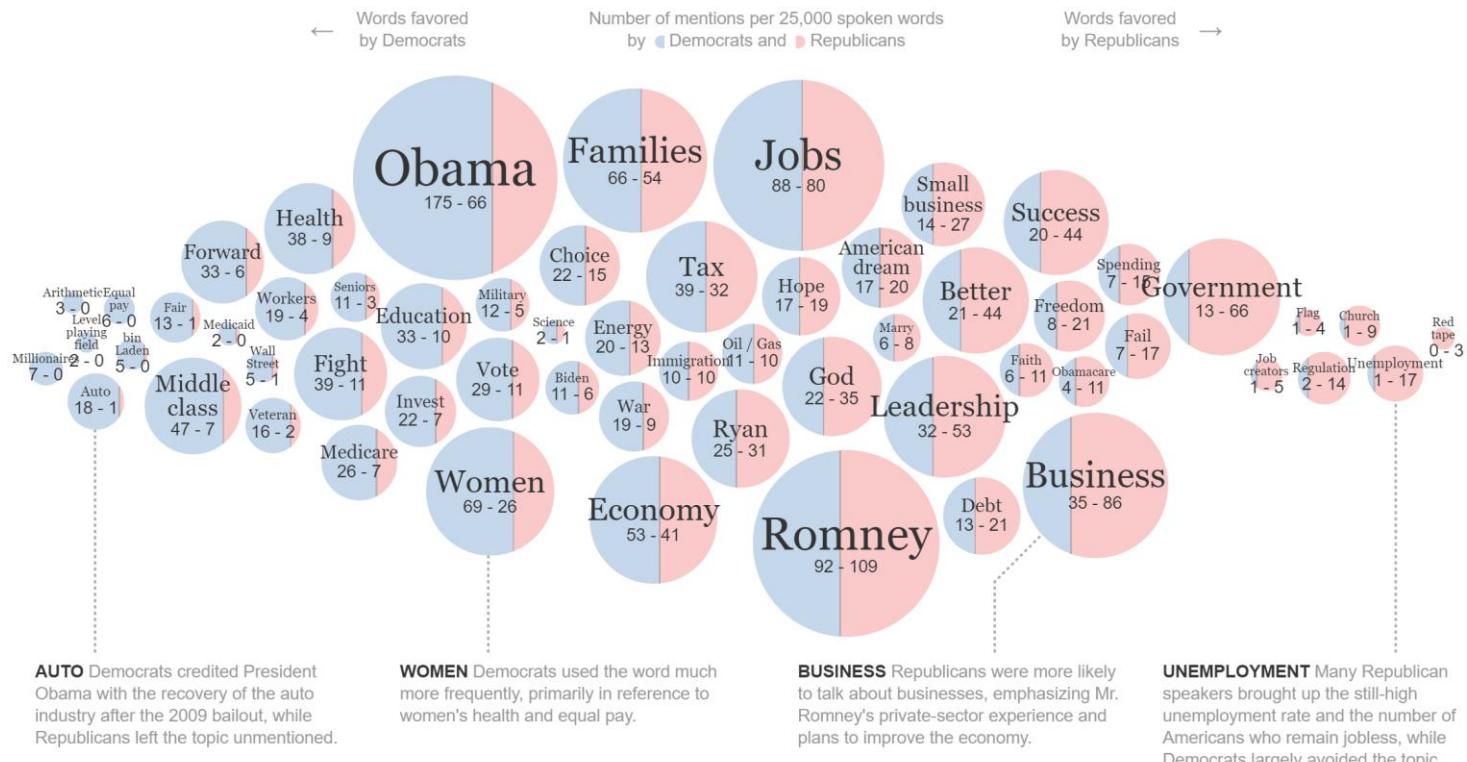
Making the difficult “easy”

Visualizing Algorithms (<https://bost.ocks.org/mike/algorithms/>)



Style & purpose of his work

- “Make sure the leap from abstract idea to concrete value is a short one.”



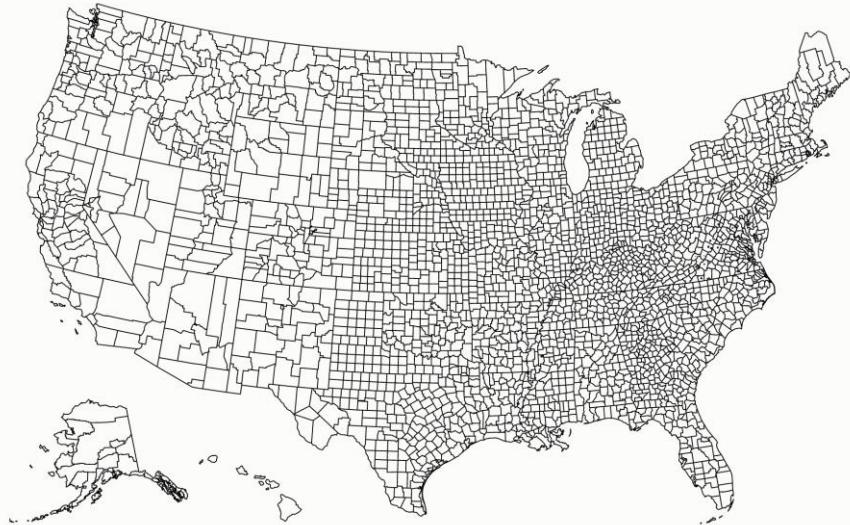
<https://archive.nytimes.com/www.nytimes.com/interactive/2012/09/06/us/politics/convention-word-counts.html>

Style & purpose of his work

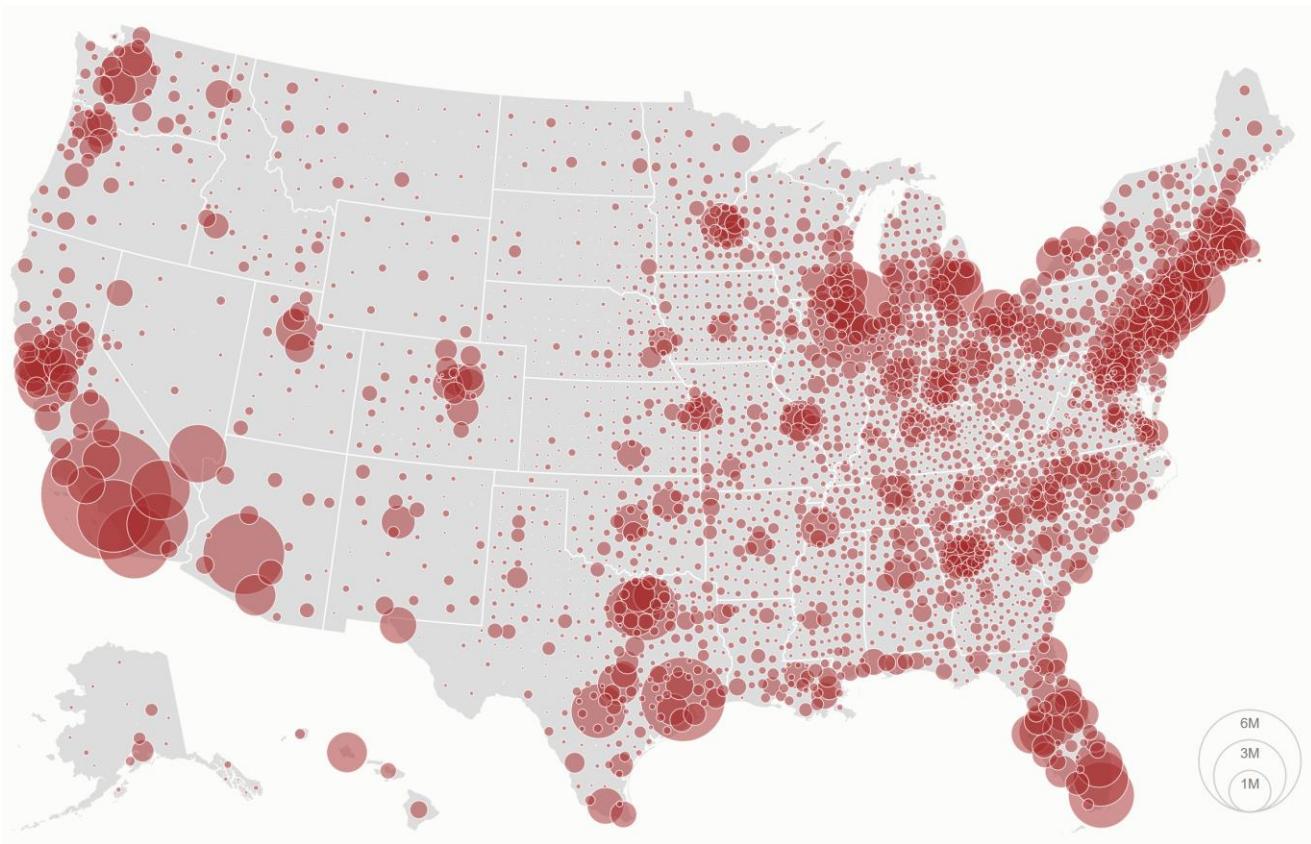
Example: Creating a bubble map

<https://bost.ocks.org/mike/bubble-map/>

Launch your local web server, and then visit your page. It should look something like this:



Two things to note at this stage. First, the `d3.geo.path` instance has a null projection; that's because our TopoJSON is already projected, so we can display it as-is. This greatly improves rendering performance. Second, we're just displaying the county boundaries so far (using `topojson.mesh`). We still have a bit of work to do before we can draw population bubbles.



What does Mike (and D3) do well?

- Simplifies complex problem by breaking it into parts
- Focuses on the essence of the design problem; changing the html DOM by using data
 - “Good design is as little design as possible.” - Mike Bostock quoting Dieter Rams
 - Tufte’s rule of minimizing data/ink ratio
 - “Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.”- Antoine de Saint-Exupery
- Uses tools to make new tools
- Focus on performance & speed. (Crucial for large datasets & interactivity)
- Open-source with a vast library of examples (Makes coding accessible)

What are the limitations/drawbacks?

Noted in Protovis article:

(M. Bostock and J. Heer. Protovis: A graphical toolkit for visualization. IEEE Trans Visualization & Comp Graphics, 15(6):1121–1128, 2009)

- Learning curve for new users.
- Accessibility to non-programmers.
- What impact does the system have on the creative process?

More:

- Applicability of visualization output to media that is not web based.

Personal reflections:

- Learning D3 is one of my main reasons for doing a MS in Data Visualization
- Low-level DOM interaction gives incredible utility to the designer
- The ability to access, change and interact with anything in an html document
- Considering steep learning curves; few things worth doing comes without sacrifice
- Examples, examples, examples!
- Large supporting community (open source culture).
- D3 is a marketable skill on one's CV (industry standard)

Questions to class:

On Mike Bostock's style and approach:

- What is your thoughts on Mike Bostock's approach of making coding more approachable to a wider audience?
- Is the world moving to a trend of easier access coding, or still very much limited to those “with a freakish knack to manipulate symbols”.

On D3:

- Why is accessing the DOM considered the “crux of the problem”?
- Considering the learning curve; when is D3 most optimal to learn and use?
- To what extent would you consider D3 as a marketable skill on your CV (as compared to other visualization tools)?