

VISUALIZATION ON THE WEB

tableau.com



DATA ANALYSIS SOFTWARE

[START YOUR FREE TRIAL](#)

Full-version trial. No credit card required.



Kibana GA



Kibi

Kibi Relational Search and Analytics

Discover Visualize Dashboard Settings

Molecules Molecules Activities (13520737) Assays (1148941) Targets (10776) Papers (59610)

Molecule type

Top 10 molecule_type: Count

molecule_type	Count
Small molecule	1,437,508
Protein	19,405
Unknown	5,379
Antibody	718
Enzyme	88
Oligonucleotide	88
Oligosaccharide	60
Cell	22
Unclassified	6

Indication Class

Top 500 indication_class: Count

indication_class	Count
Antibacterial	319
Antineoplastic	187
Antidepressant	99
Antihypertensive	97
Anti-Inflammatory	89
Analgesic	81
Antipsychotic	60
Radioactive Agent	73

molecules search

pref_name	molecule_type	availability_type	synonyms	chirality
(2S,4S,5R,6R)-6-acetamido-6-((1R,2R)-3-azahexyl)mercury	Small molecule	-1	-	-1
(1R,9S,12S)-12-Methoxy-9-methyl-10,11,11-trimethyl-stannane	Small molecule	-1	-	-1
(1R,9S,12S)-12-Methoxy-9-methyl-10,11,11-triphenyl-stannane	Small molecule	-1	-	-1
(+)-NEOMENTHOL	Small molecule	-1	(+)-Neomenthol	-1
(+)-RR-dichloro-[1,2-bis(4-hydroxyphenyl)]ethanum (B)	Small molecule	-1	-	-1
(+)-(8S)-PARASORBIC ACID	Small molecule	-1	-	-1
(+)-11-DEMETHYL CALANOLIDE A	Small molecule	-1	(+)-11-demethyl calanolide A	-1
(+)-11-DEMETHYL CORDATOLIDE A	Small molecule	-1	(+)-11-demethyl cordatolide A	-1
1-(3,5-DICFTYL THIOL ACTONE	Small molecule	-1	-	-1

Relational Button Activities

.. show related activities (13520737)

Therapeutic vs Non (Chirality)

Legend: -1 (Green), 1 (Blue), 2 (Purple), 0 (White)

A screenshot of the Kibi interface, a relational search and analytics tool. The top navigation bar includes 'Discover', 'Visualize', 'Dashboard', and 'Settings'. Below the navigation is a header with tabs for 'Molecules' (selected), 'Activities (13520737)', 'Assays (1148941)', 'Targets (10776)', and 'Papers (59610)'. On the left, two sidebar panels show 'Top 10 molecule_type' and 'Top 500 indication_class' counts with their respective tables. The main content area is titled 'molecules search' and displays a table of molecules with their structures, names, types, availability, synonyms, and chirality. To the right of the table are two pie charts under the 'Therapeutic vs Non (Chirality)' section, showing the distribution of chirality for therapeutic and non-therapeutic molecules.

<https://siren.solutions/kibi/>

Superset



World's Bank Data

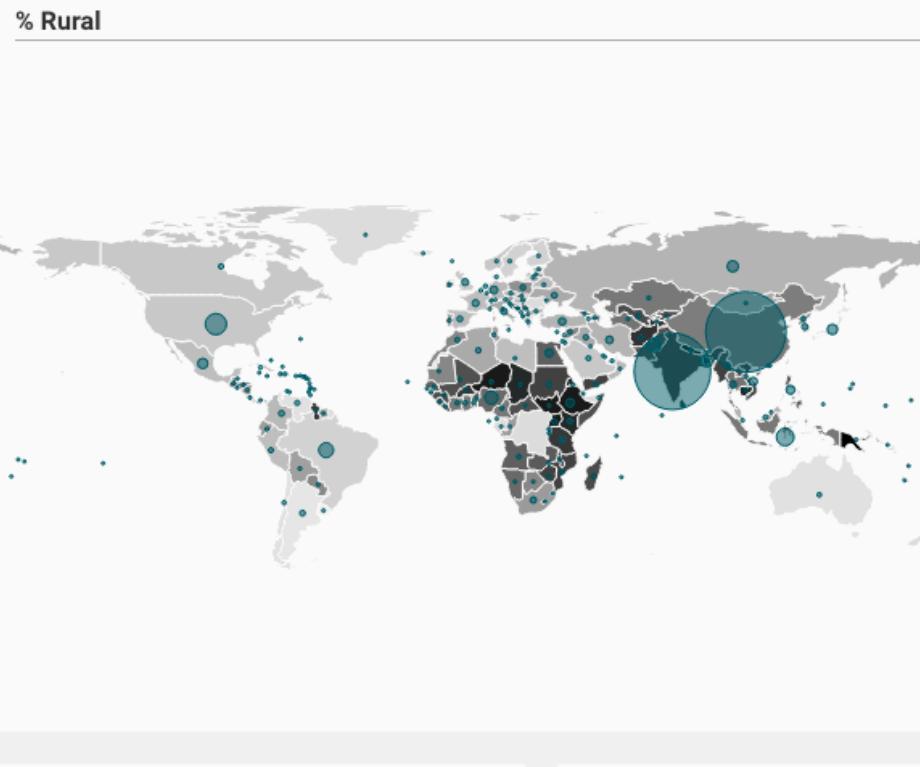


Region Filter

region
Select [region]

World's Population

7.24G
+12.9% over 10Y



Most Populated Countries

country_name	sum_SP_POP_TOTL
China	1.36G
India	1.30G
United States	319M
Indonesia	254M
Brazil	206M
Pakistan	185M
Nigeria	177M
Bangladesh	159M
Russian Federation	144M
Japan	127M
Mexico	125M
Philippines	99.1M
Ethiopia	97.0M
Vietnam	90.7M
Egypt, Arab Rep.	89.6M
Germany	80.9M
Iran, Islamic Rep.	78.1M
Turkey	75.9M
Congo, Dem.	74.9M

NVD3.js

NVD3.js [Home](#) [Examples](#) [Live Code](#) [Source](#) [Blog](#) [Downloads: ZIP TAR.GZ](#)

NVD3 Re-usable charts for d3.js

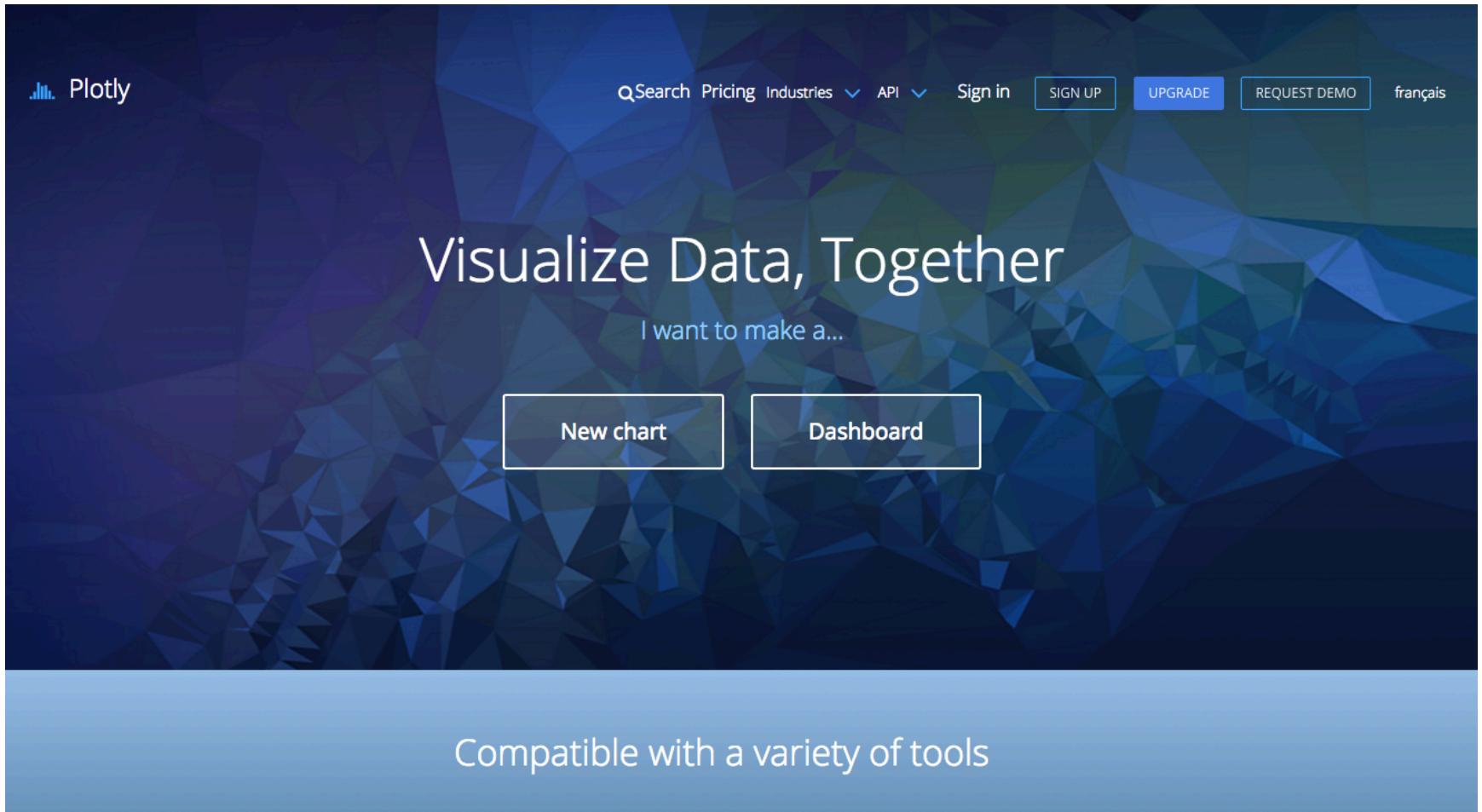
This project is an attempt to build re-usable charts and chart components for d3.js without taking away the power that d3.js gives you. This is a very young collection of components, with the goal of keeping these components very customizable, staying away from your standard cookie cutter solutions.

[View more examples »](#)

[GitHub Repo](#)

The page displays three charts illustrating the capabilities of NVD3.js:

- Bar Chart:** Shows three data streams (Stream0, Stream1, Stream2) as grouped bars. Stream0 is blue, Stream1 is light blue, and Stream2 is orange. The x-axis has ticks at 7, 15, 23, and 31. The y-axis ranges from 0.0 to 3.4. Stream2 has peaks around x=12, 18, and 24.
- Stacked Area Chart:** Shows three stacked areas (Stream0, Stream1, Stream2). Stream0 is blue, Stream1 is light blue, and Stream2 is orange. The x-axis has ticks at 0, 10, 20, 30, and 36. The y-axis ranges from 0.0 to 4.5. The total height of the stack fluctuates between 1.0 and 3.5.
- Grouped Area Chart:** Shows three grouped areas (Stream0, Stream1, Stream2). Stream0 is blue, Stream1 is light blue, and Stream2 is orange. The x-axis has ticks at 0, 10, 20, 30, and 36. The y-axis ranges from 0.0 to 4.5. The areas represent different data series stacked together.



The image shows the homepage of Plotly's website. The background features a dark blue polygonal pattern. At the top left is the Plotly logo. The top navigation bar includes a search bar, links for "Pricing", "Industries", "API", "Sign in", and buttons for "SIGN UP", "UPGRADE", "REQUEST DEMO", and "français". The main headline "Visualize Data, Together" is centered, followed by the subtext "I want to make a...". Below this are two buttons: "New chart" and "Dashboard". A footer banner at the bottom states "Compatible with a variety of tools".

Plotly

Search Pricing Industries API Sign in SIGN UP UPGRADE REQUEST DEMO français

Visualize Data, Together

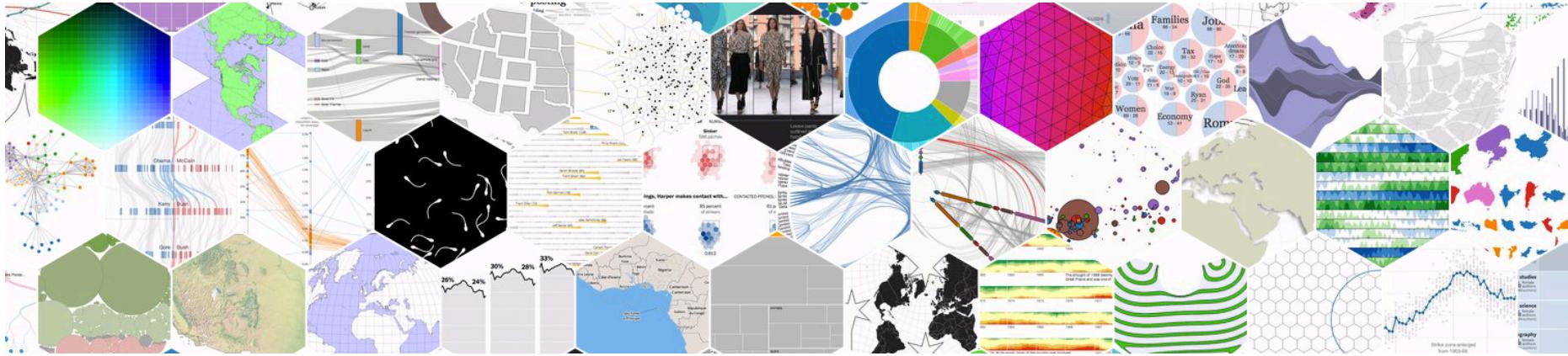
I want to make a...

New chart Dashboard

Compatible with a variety of tools

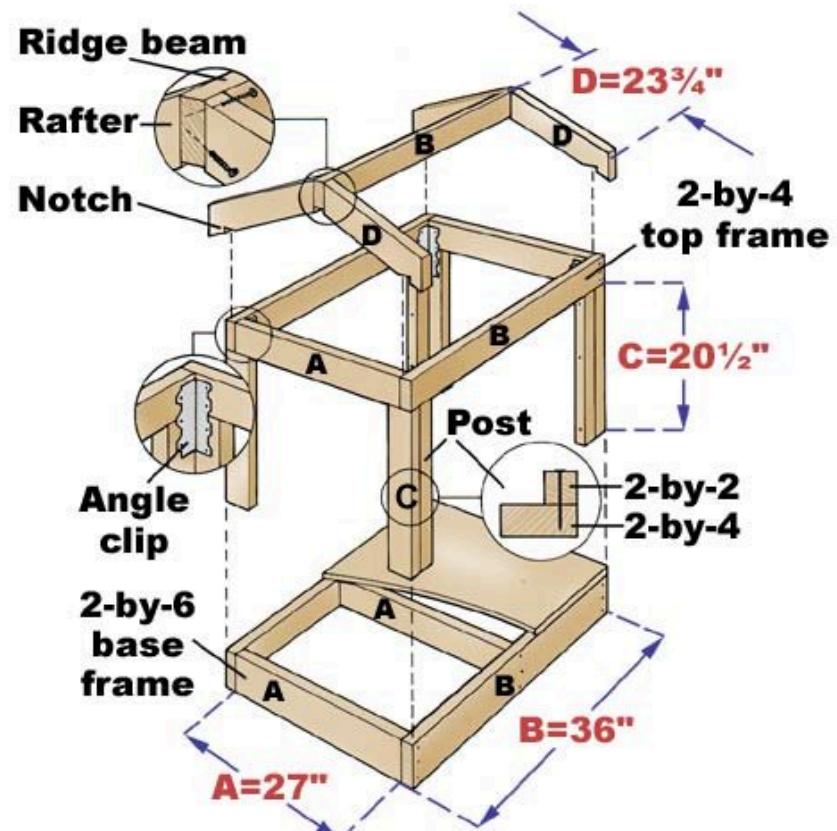


Data-Driven Documents



VISUAL ANALYTICS D3.JS

What is D3?

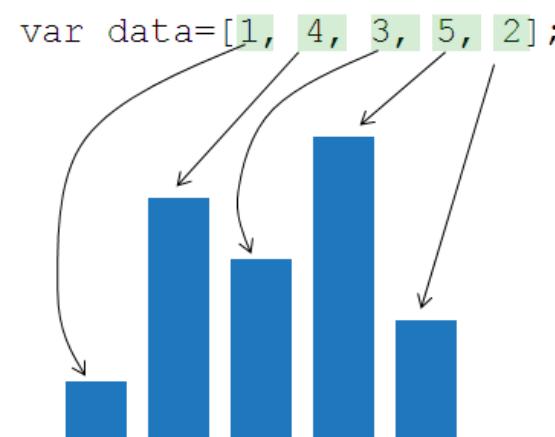


What is D3?



What is D3?

- JavaScript library to make beautiful, interactive, browser-based data visualizations.
- D3 stands for **Data Driven Documents**
- D3.js is a low level visualization library based on Web standards (HTML, CSS, JS, SVG)
- D3.js is Open Source library written by Mike Bostok
- [Mike Bostock Github Profile](#)
- d3js.org

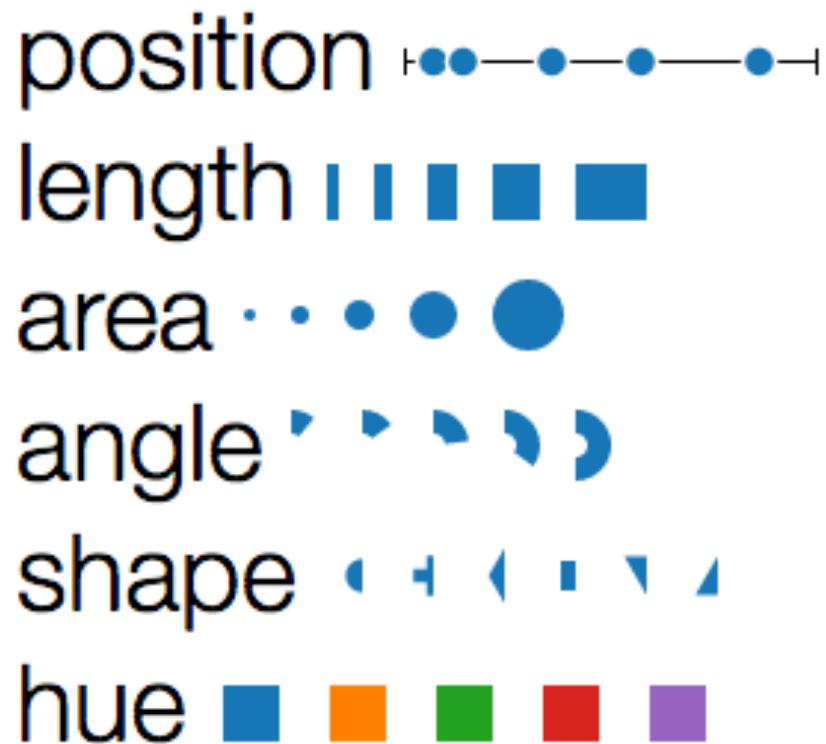


Visualization and Data Graphics

Data Types

- Categorical
- Ordinal
- Quantitative

Visual Variables



Visual Variables -> Documents

- Datum -> Element
 - Associate a graphical mark to each data point
- Data Attribute -> Element Attribute
 - Adjust properties of mark to encode properties of datum

GETTING STARTED

DEVELOPMENT CHECKLIST

Tools

- A modern browser (Chrome, Firefox, etc)
- A modern text editor (TextMate, Sublime, Atom, ...)
- A terminal (Command prompt) to run an http-server [Terminal A]
- A terminal to handle code versioning [Terminal B]
- Node.js and NPM installed

Tools (alternative)

- A modern browser (Chrome, Firefox, etc)
- An integrated IDE, like WebStorm for example
- Node.js and NPM installed

Project init

- Only once
 - `npm install http-server --global`
- In [Terminal A]
 - `cd path/to/project/directory`
 - `npm init`
 - Open current directory in editor
 - `npm install <module>`
 - `http-server`

Web Page Preparation

- Create a file HTML
- Create content for the page
- Include an empty DIV for the visualization
- Install and link D3
- Construct SVG element within the DIV element
- Optionally
 - Create and init git repository

SELECTIONS

CSS Selectors

- CSS provides an efficient way to refer to specific elements in a DOM
- #foo // <any id="foo">
- foo // <foo>...</foo>
- .foo // <any class="foo">
- [foo=bar] // <any foo="bar">
- foo bar //<foo><bar/></foo>

Selector Functions

W3C

- `document.querySelectorAll("h1")`

D3.js / JQuery

- `d3.selectAll("h1")`

Selections are Arrays.

Explore selections with Developer Tools

attr and style methods

```
// select all <h1> elements  
var H1s = d3.selectAll("H1");  
  
H1s.attr("class","newClass");  
H1s.style("fill","yellow");  
H1s.style("font-color","black");
```

Chaining methods

```
d3.selectAll("H1")  
  .attr("class", "newClass")  
  .style("fill", "yellow")  
  .style("font-color", "black");
```

Append new elements

```
var body = d3.select("body");
```

```
var h1 = body.append("h1");
h1.text("Hello!");
```

Modify existing elements

```
var section = d3.selectAll("section");
```

```
var h1 = section.append("h1");
h1.text("Hello!");
```

Exercise #1

- Create the ladder design of the previous lesson, using only D3.js manipulation of DOM



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Stairs example - Multiple implementation</title>
    <style>
        svg{
            background:#fff;
        }

        svg circle{
            fill:#e34a33
        }
    </style>
</head>
<body>
    <!--
        Draw a polyline using the polyline element
    -->
    <svg width="200" height="200">
        <polyline points="0,40 40,40 40,80 80,80
120,120 120,160" fill="white"
stroke="#BBC42A" stroke-width="6" />
    </svg>
</body>
</html>
```

DATA TO ELEMENTS

Selection should correspond to data

```
var numbers =                                     Data           SVG  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
    .data(numbers)  
    .enter().append("line");
```

Selection should correspond to data

```
var numbers =                                     Data          SVG  
[5,10,15,20,25];  
  
var lines =                                         5  
svg.selectAll("line")  
  .data(numbers)                                    10  
  .enter().append("line");  
                                              15  
  
                                              20  
  
                                              25
```

Method `data` joins data with document elements

Selection should correspond to data

```
var numbers =  
[5,10,15,20,25];
```

Data

SVG

```
var lines =  
svg.selectAll("line")
```

5



```
.data(numbers)
```

10



```
.enter().append("line");
```

15



20



25



Method `enter` specifies the action for missing elements

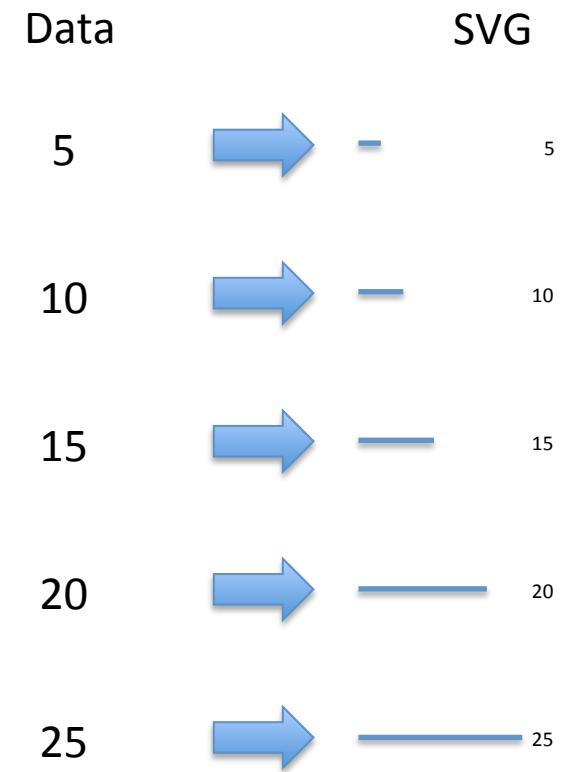
Selection should correspond to data

```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
  .data(numbers)  
  .enter().append("line");
```

Data	SVG
5	 5
10	 10
15	 15
20	 20
25	 25

Selection should correspond to data

```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
  .data(numbers)  
  .enter().append("line");  
  
lines.attr("x1",10)  
  .attr("y1",posy(d,i))  
  .attr("x2",posx(d,i))  
  .attr("y2",posy(d,i))
```



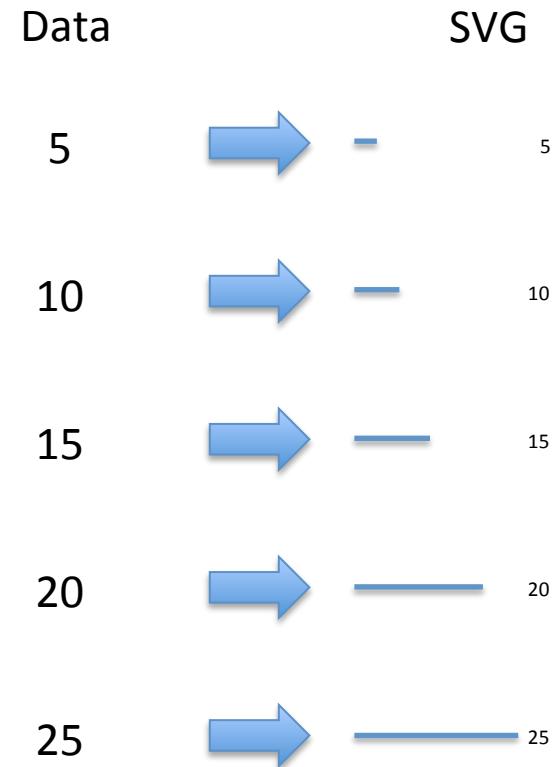
The new elements are bound to data. Data can be used to compute attributes

Selection should correspond to data

```
lines.attr("x1",10)
    .attr("y1",posy(d,i))
    .attr("x2",posx(d,i))
    .attr("y2",posy(d,i));
```

```
var posy = function(d,i){
    return i*10;
}
```

```
var posx = function(d,i){
    return d * 10;
}
```



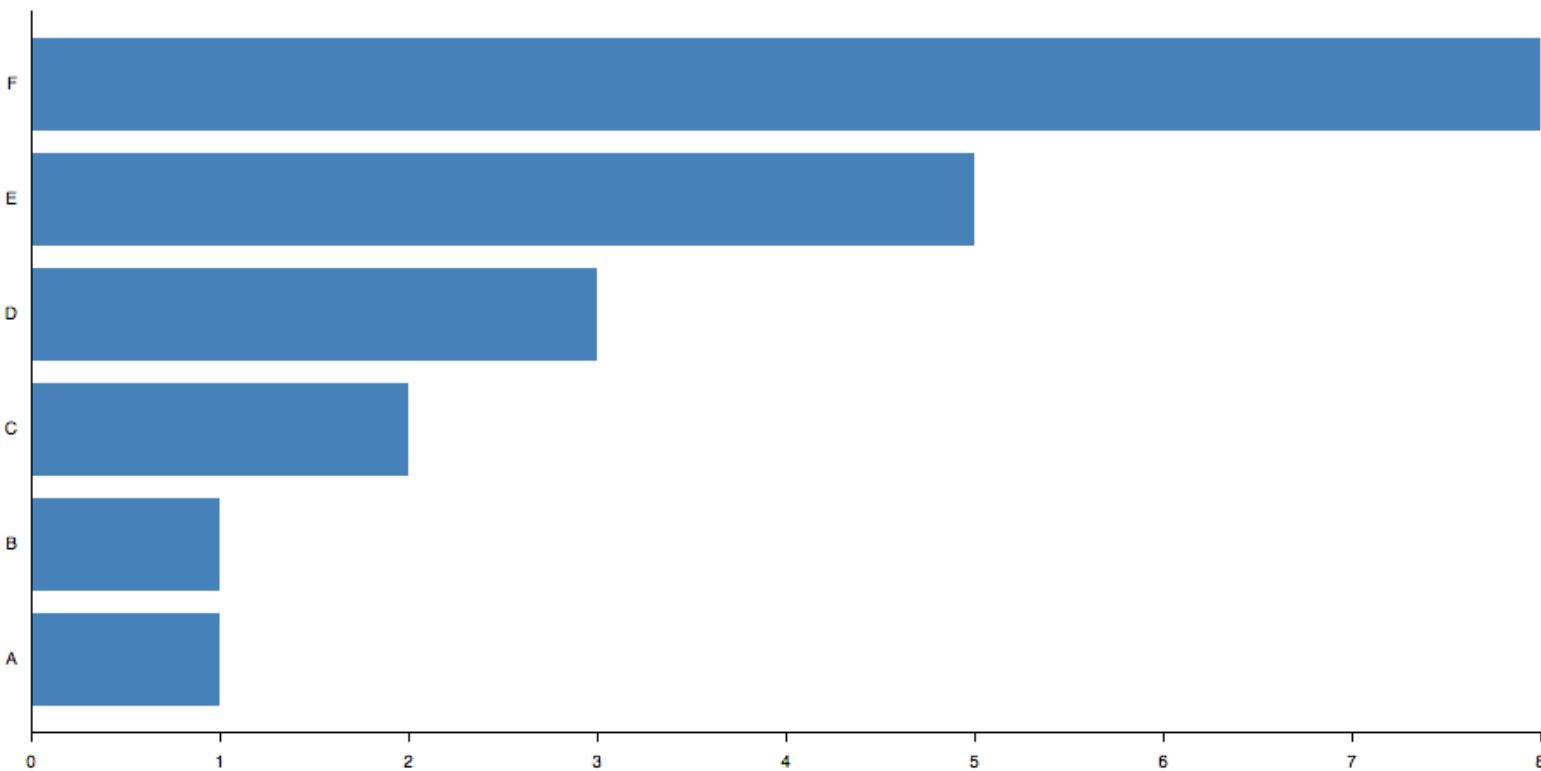
The `attr` functions takes in input a constant value or a function. The function is called automatically by d3, passing the data (`_data_`) bound to the element and a progressive counter

Exercise #2

- Use length visual variable to represent a set of numbers
 - Map numbers to a set of lines
 - Make each line length proportional to the number it represents

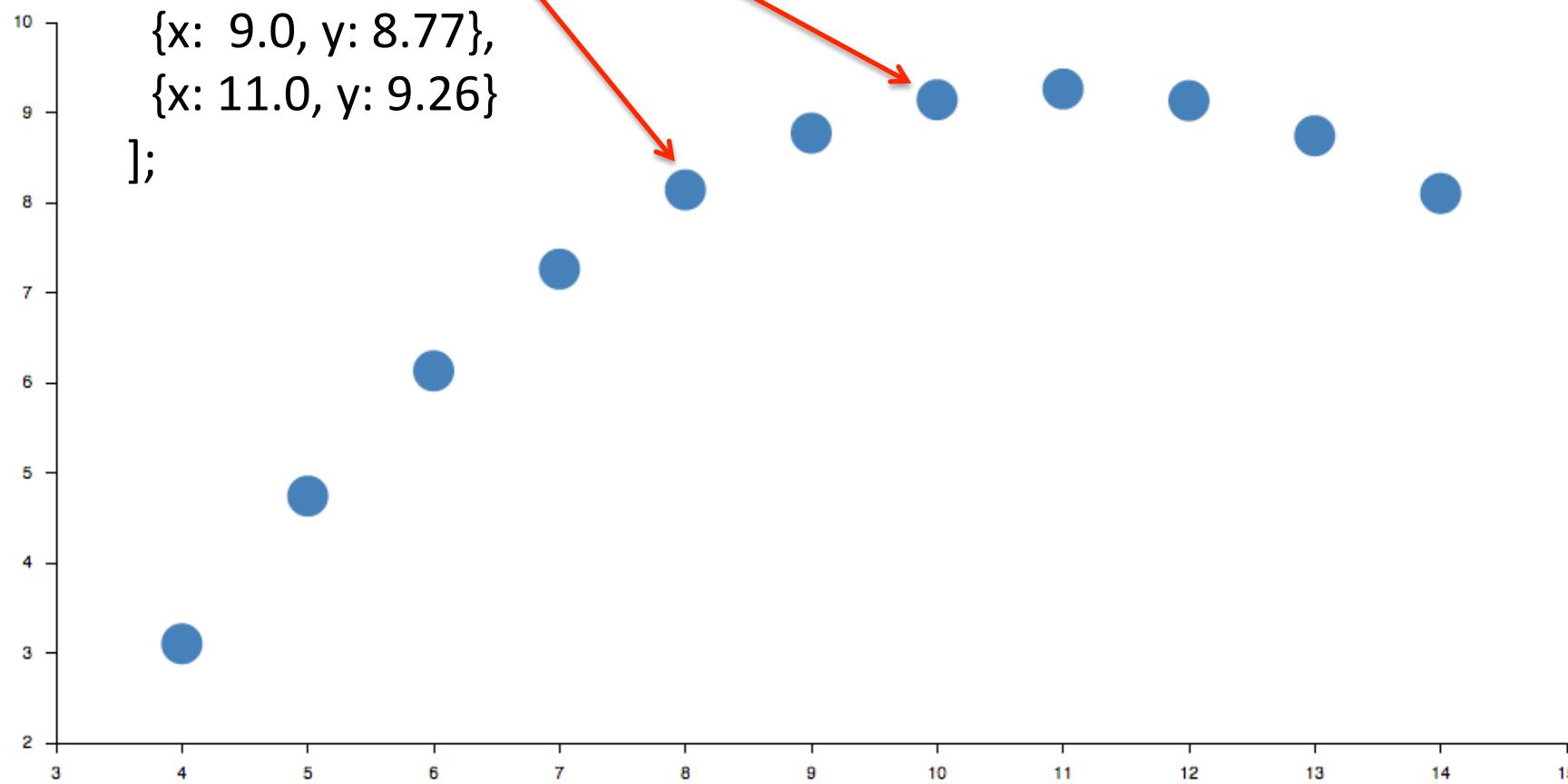
Data can be numbers

```
var numbers= [1, 1, 2, 3, 5, 8];
```



Data can be objects.

```
var data = [  
  {x: 10.0, y: 9.14},  
  {x: 8.0, y: 8.14},  
  {x: 13.0, y: 8.74},  
  {x: 9.0, y: 8.77},  
  {x: 11.0, y: 9.26}  
];
```

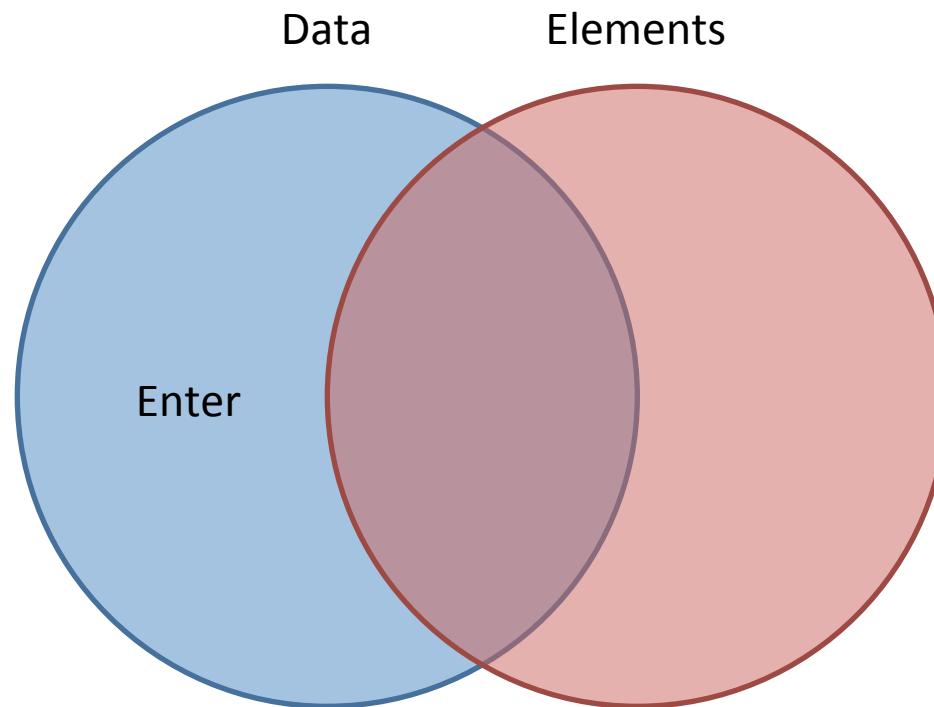


Thinking with Joins

ENTER, EXIT, AND UPDATE

Enter

- New data, for which there were no existing elements.



Entering new elements

```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
    .data(numbers);
```

```
lines  
    .enter().append("line");
```

Data SVG

5 

10 

15 

20 

25 

Entering new elements

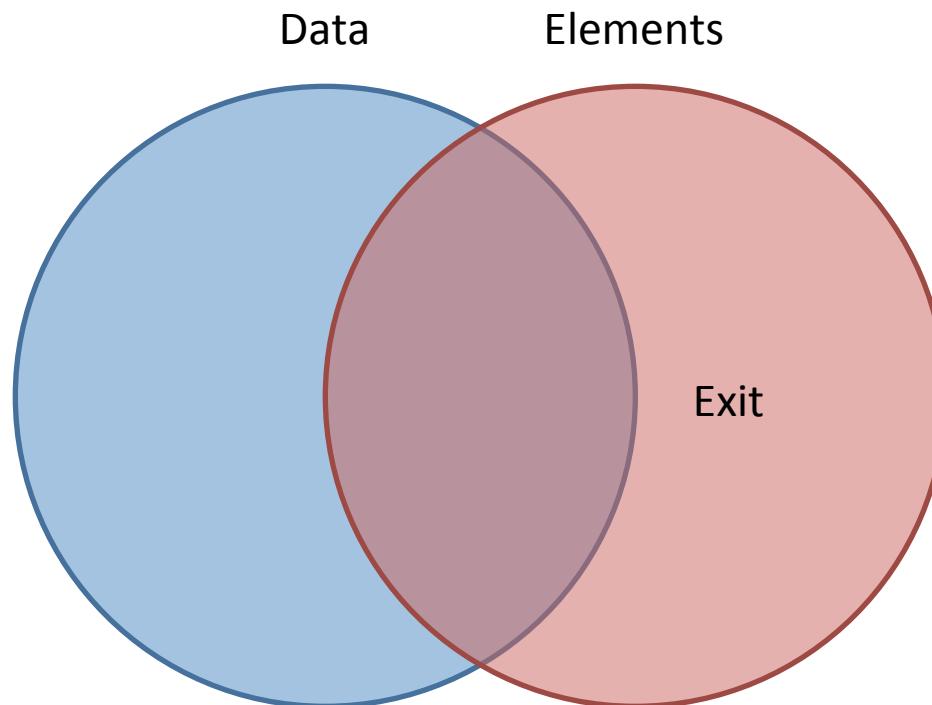
```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
    .data(numbers);
```

```
lines  
    .enter().append("line");
```

Data	SVG
5	
10	
15	
20	
25	

Exit

- Elements that are associated with no data



Exiting unnecessary elements

```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
  .data(numbers);
```

Data

5



SVG

— 5

10



— 10

```
lines  
  .exit().remove();
```

15



— 15



—



—

Entering new elements

```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
    .data(numbers);  
  
lines  
    .exit().remove();
```

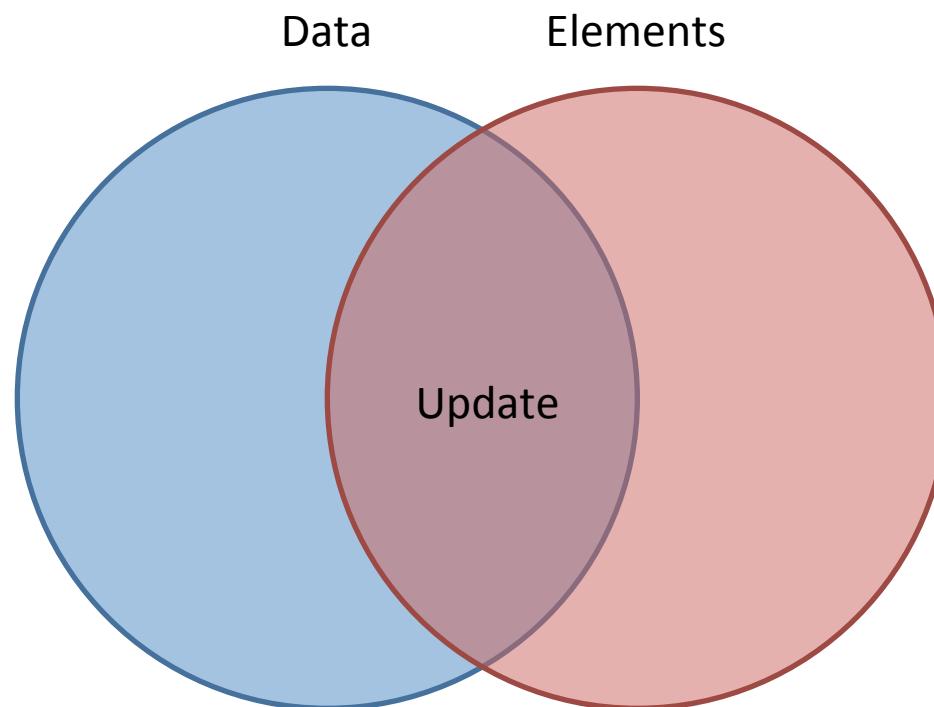
	Data	SVG
5		 5
10		 10
15		 15

Step 2

DATA ATTRIBUTES TO ELEMENTS ATTRIBUTES

Update

- Data already joined with previous elements



Update existing and new elements with new data

```
var numbers =  
[5,10,15,20,25];  
  
var lines =  
svg.selectAll("line")  
  .data(numbers);
```

Data SVG

5 5

10 10

```
lines = lines.enter()  
  .append("line")  
  .merge(lines);
```

15 15

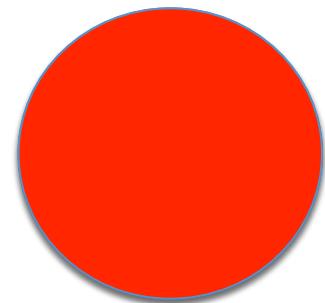
```
lines.attr("x1",10)  
  .attr("y1",posy(d,i))  
  .attr("x2",posx(d,i))  
  .attr("y2",posy(d,i));
```

Joining with key function

```
var data = [
  {name: "Locke", number: 4},
  {name: "Reyes", number: 8},
  {name: "Ford", number: 15},
  {name: "Jarrah", number: 16},
  {name: "Shephard", number: 31},
  {name: "Kwon", number: 34}
];
d3.selectAll("div")
  .data(data, function(d) { return d ? d.name : this.id; })
  .text(function(d) { return d.number; });
```

D3 events

```
svg.append("circle")
    .attr("cx",100)
    .attr("cy",100)
    .attr("r",50)
    .attr("fill", "blue")
    .on("click", function(){
        d3.select(this).attr("fill","red");
    });
}
```



Exercize #1: Alphabet

Useful resources

- <https://d3js.org>
- <https://www.dashingd3js.com/>
- <https://github.com/mbostock/d3/wiki/API-Reference>
- Tutorial by Mike Bostok
- <http://bostocks.org/mike/d3/workshop/>