



VISUALIZATION ON THE WEB

tableau.com



DATA ANALYSIS SOFTWARE

START YOUR FREE TRIAL

Full-version trial. No credit card required.



Kibana GA



Kibi

The screenshot displays the Kibi web application interface. At the top, there is a navigation bar with tabs for Discover, Visualize, Dashboard, and Settings. Below this is a search bar and a set of filters for Molecules (13520737), Assays (1148941), Targets (10776), and Papers (59610). The main content area is a table of search results for molecules, with columns for molregno, pref_name, molecule_type, availability_type, synonyms, and chirality. The table lists several small molecules, including (2S,4S,5R,6R)-5-acetamido-6-((1R,2R)-3-aziridinyl)methylmercury, (6R,9S,12S)-12-Methoxy-9-methyl-10,11,11-trimethylstannane, (6R,9S,12S)-12-Methoxy-9-methyl-10,11,11-triphenylstannane, (-)-NEOMENTHOL, (-)-R,R-dichloro-[1,2-bis(4-hydroxyphenyl)ethane]inane, (-)-(-6S)-PARASORBIC ACID, (+)-11-DEMETHYL CALANOLIDE A, (+)-11-DEMETHYL CORDATOLIDE A, and (+)-3-O-ACETYLMETHYLACTONE.

On the left side, there are two panels: 'Molecule type' showing a top 10 list (Small molecule: 1,437,508; Protein: 19,405; Unknown: 5,379; Antibody: 718; Enzyme: 88; Oligonucleotide: 86; Oligosaccharide: 60; Cell: 22; Unclassified: 6) and 'Indication Class' showing a top 500 list (Antibacterial: 319; Antineoplastic: 187; Antidepressant: 99; Antihypertensive: 97; Anti-inflammatory: 89; Analgesic: 81; Antipsychotic: 80; Radioactive Agent: 73).

On the right side, there are two charts: 'Relational Button Activities' showing a button for 'show related activities (13520737)' and 'Therapeutic vs Non (Chirality)' showing two pie charts. The top pie chart is mostly green (-1), and the bottom pie chart is divided into green (-1), blue (1), purple (2), and dark blue (0).

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

Superset



World's Bank Data



Region Filter

region

Select [region] ▼

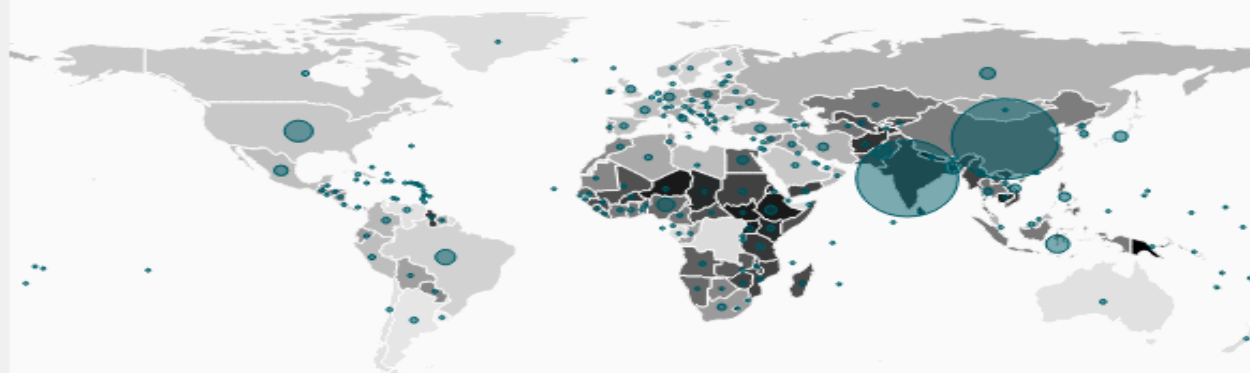
country_name

Select [country_...]

World's Population

7.24G
+12.9% over 10Y

% Rural



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

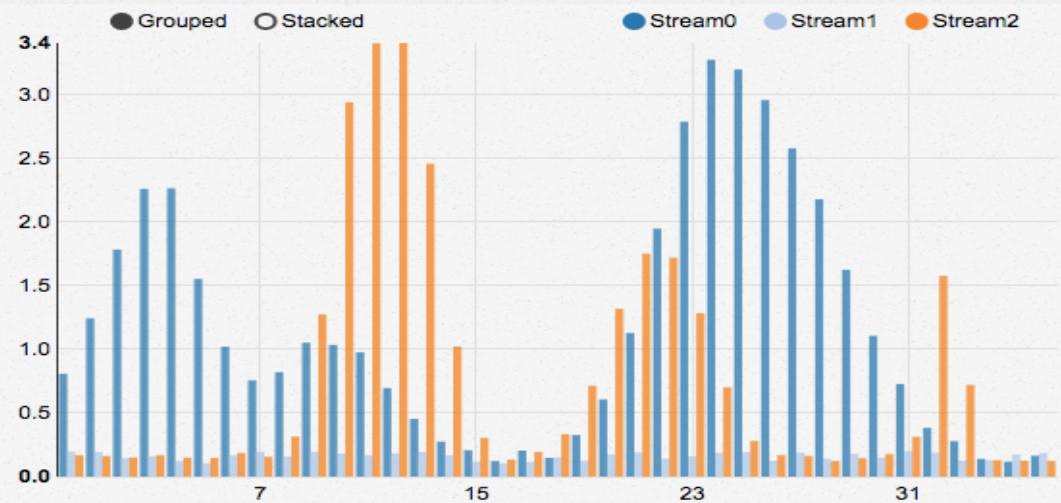
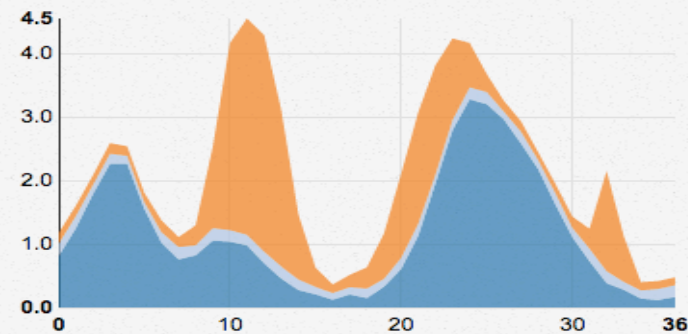
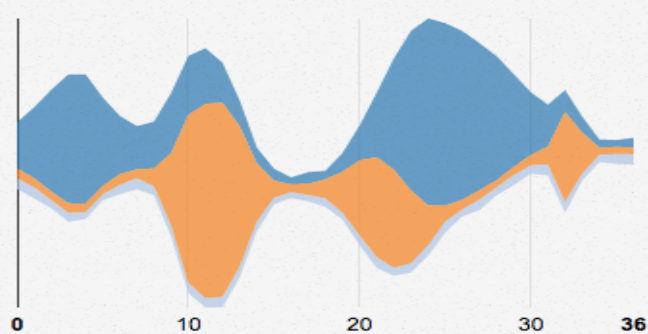
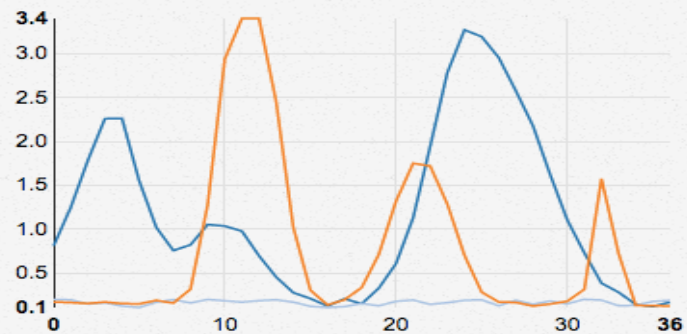
TAR.GZ

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.

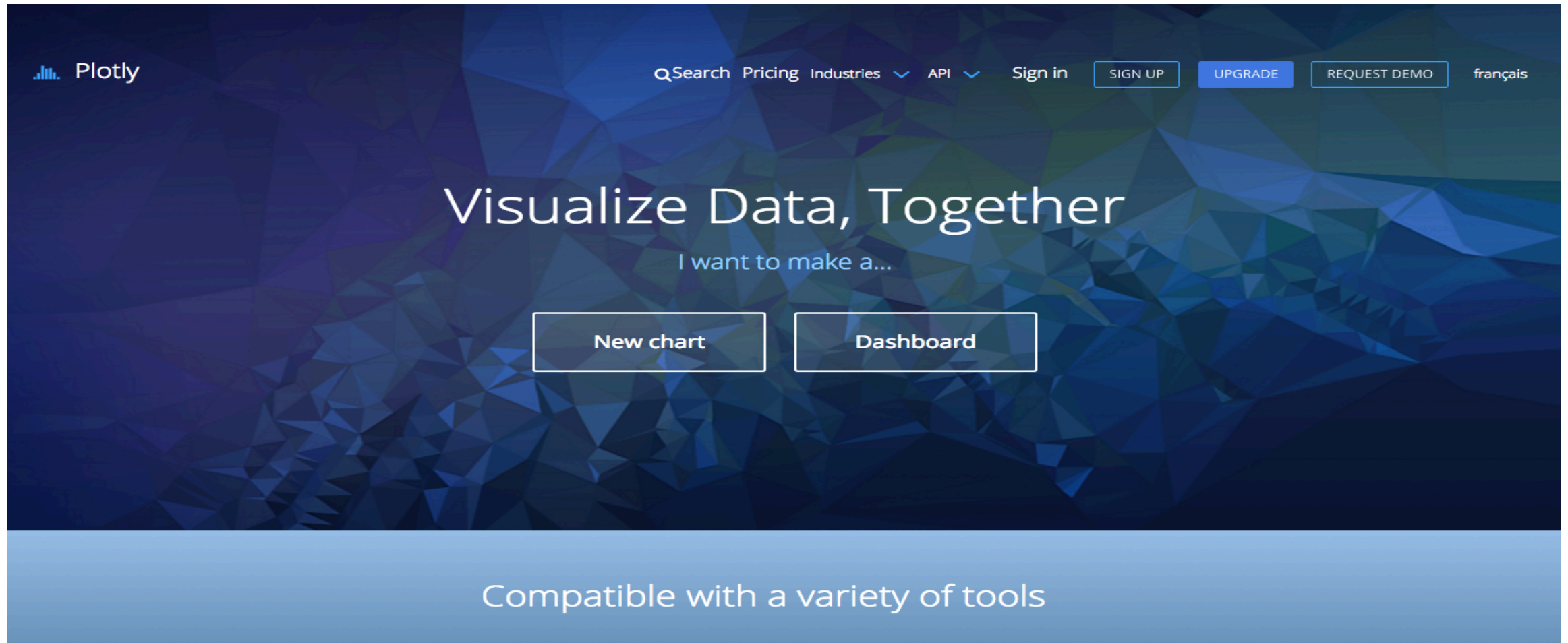
[View more examples »](#)

[GitHub Repo](#)



● Grouped ○ Stacked

● Stream0 ● Stream1 ● Stream2

The image shows the Plot.ly website landing page. The background is a dark blue, low-poly geometric pattern. At the top left is the Plot.ly logo. The top right contains navigation links: 'QSearch', 'Pricing', 'Industries' with a dropdown arrow, 'API' with a dropdown arrow, 'Sign in', a 'SIGN UP' button, a 'UPGRADE' button, a 'REQUEST DEMO' button, and a 'français' link. The main heading is 'Visualize Data, Together' in large white text, with the subtext 'I want to make a...' below it. Two white-outlined buttons, 'New chart' and 'Dashboard', are centered below the subtext. At the bottom, a light blue gradient bar contains the text 'Compatible with a variety of tools'.

Visualize Data, Together

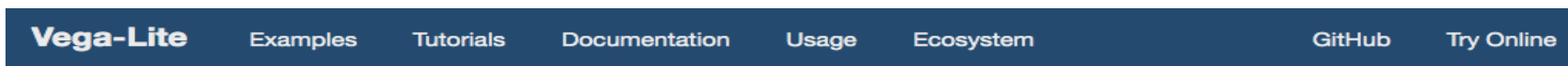
I want to make a...

New chart

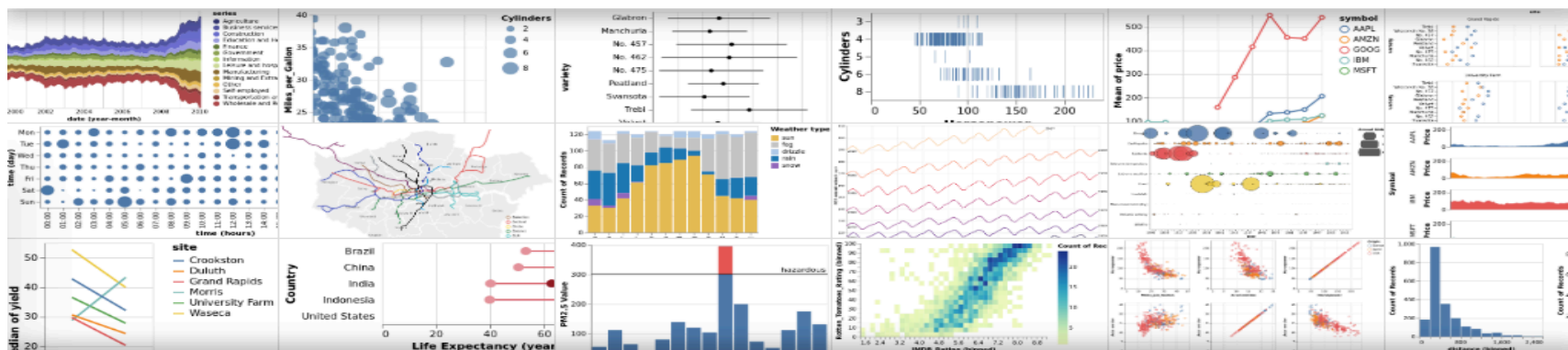
Dashboard

Compatible with a variety of tools

Vega and Vega-lite



Vega-Lite – A Grammar of Interactive Graphics



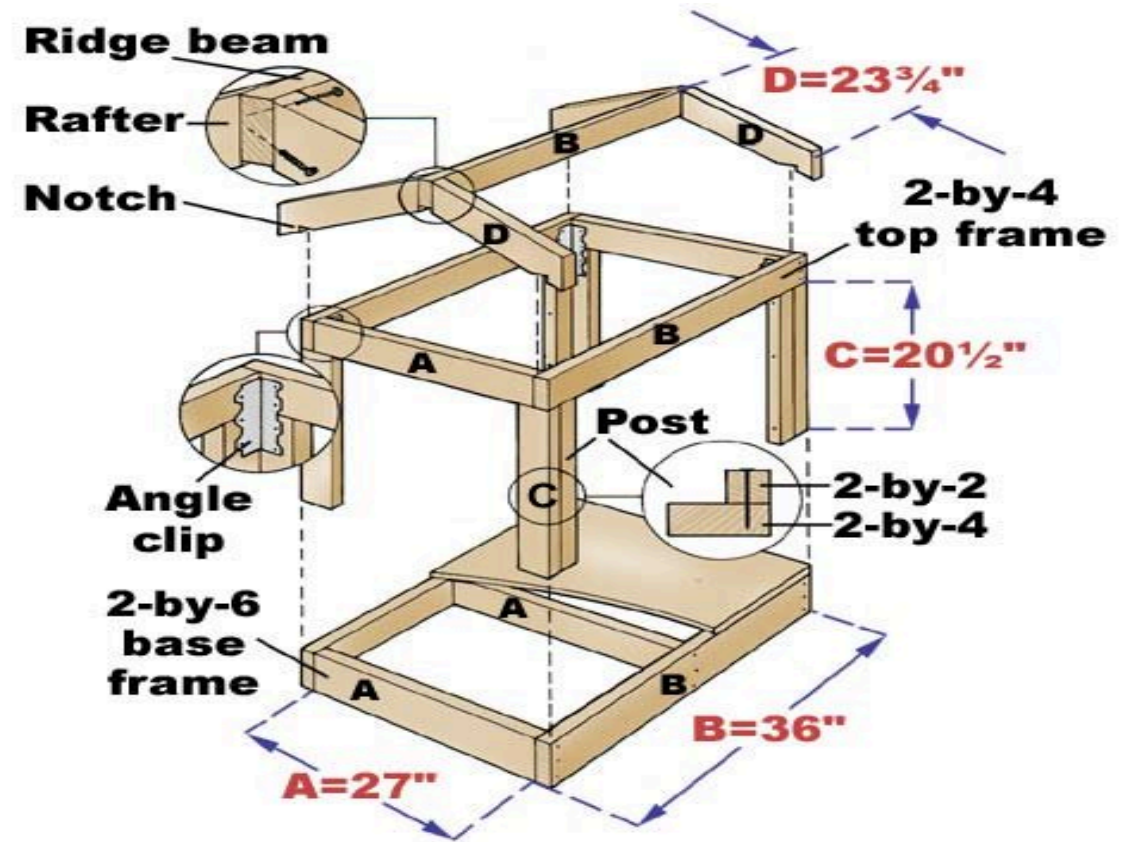
Vega-Lite is a high-level grammar of interactive graphics. It provides a concise JSON syntax for rapidly generating visualizations to support analysis. Vega-Lite specifications can be compiled to [Vega](#) specifications.

Vega-Lite specifications describe visualizations as mappings from data to **properties of graphical marks** (e.g., points or bars). The Vega-Lite compiler **automatically produces visualization components** including axes, legends, and scales. It then determines properties of these components based on a set of **carefully designed rules**. This approach allows specifications to be succinct and expressive, but also provide user control. As Vega-Lite is designed for analysis, it supports **data transformations** such as aggregation, binning, filtering, sorting, and **visual transformations** including stacking and faceting. Moreover, Vega-Lite specifications can be **composed** into layered and multi-view displays, and made **interactive with selections**.

[Get started](#)
Latest Version: 4.7.0

[Try online](#)

What is D3?



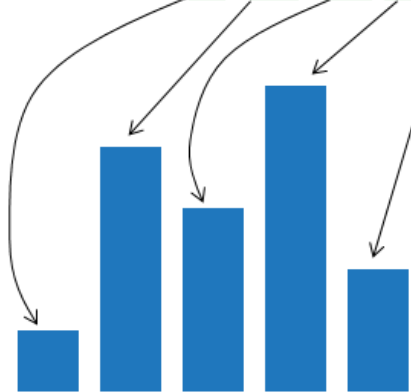
What is D3?



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

```
var data=[1, 4, 3, 5, 2];
```

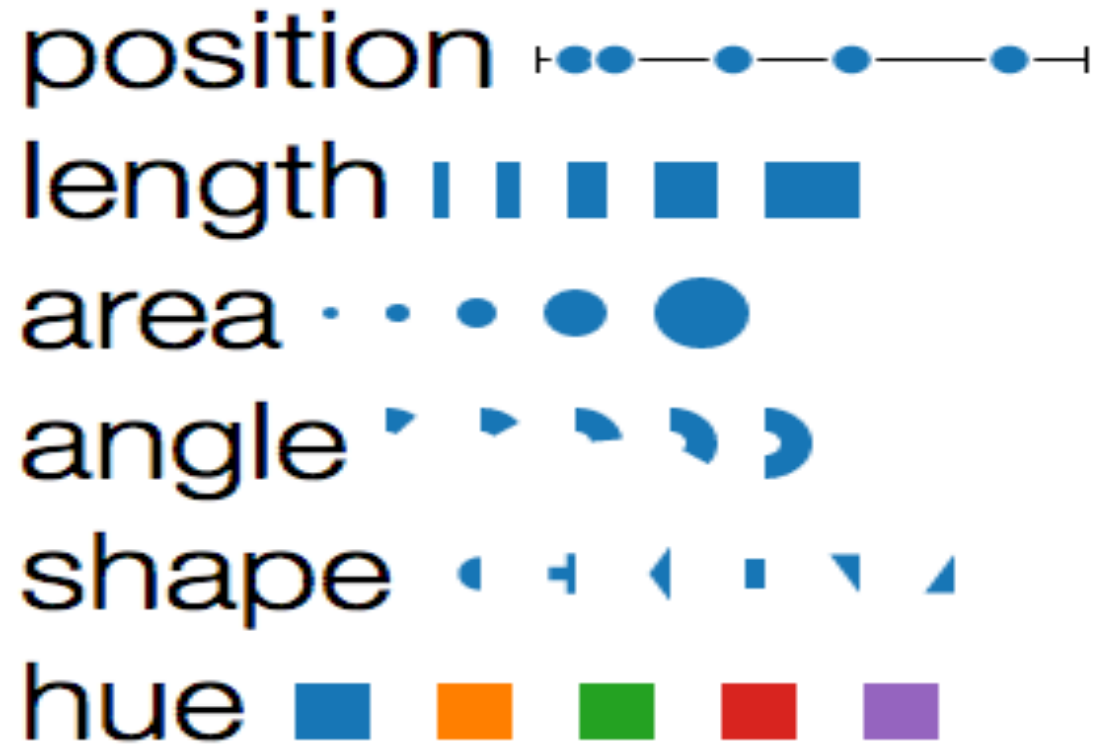


Visualization and Data Graphics

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



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 80,120 120,120 120,160"
    fill="white" stroke="#BBC42A" stroke-width="6" />
  </svg>
</body>
</html>
```



DATA TO 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

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

10

15

20

25

Method [data](#) joins data with document 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



10



15



20



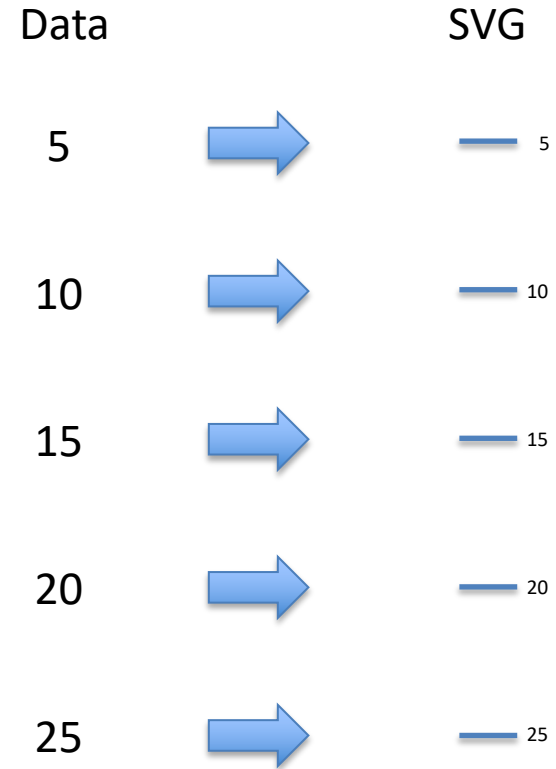
25



Method `enter` specifies the action for missing elements

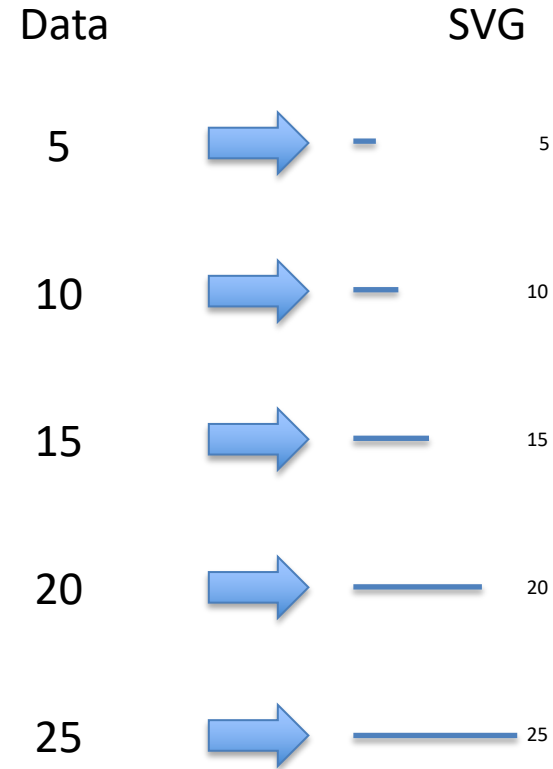
Selection should correspond to data

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



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))
```



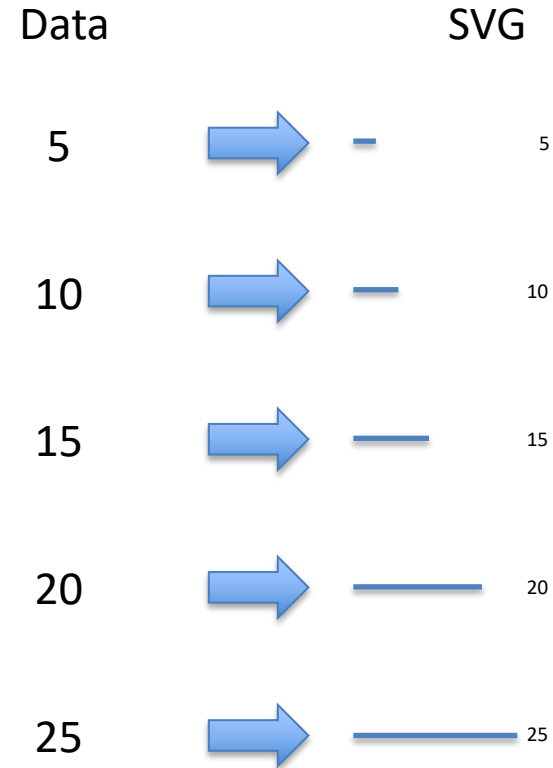
The new elements are bound to data. Data can be used to compute attributes

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;
}
```



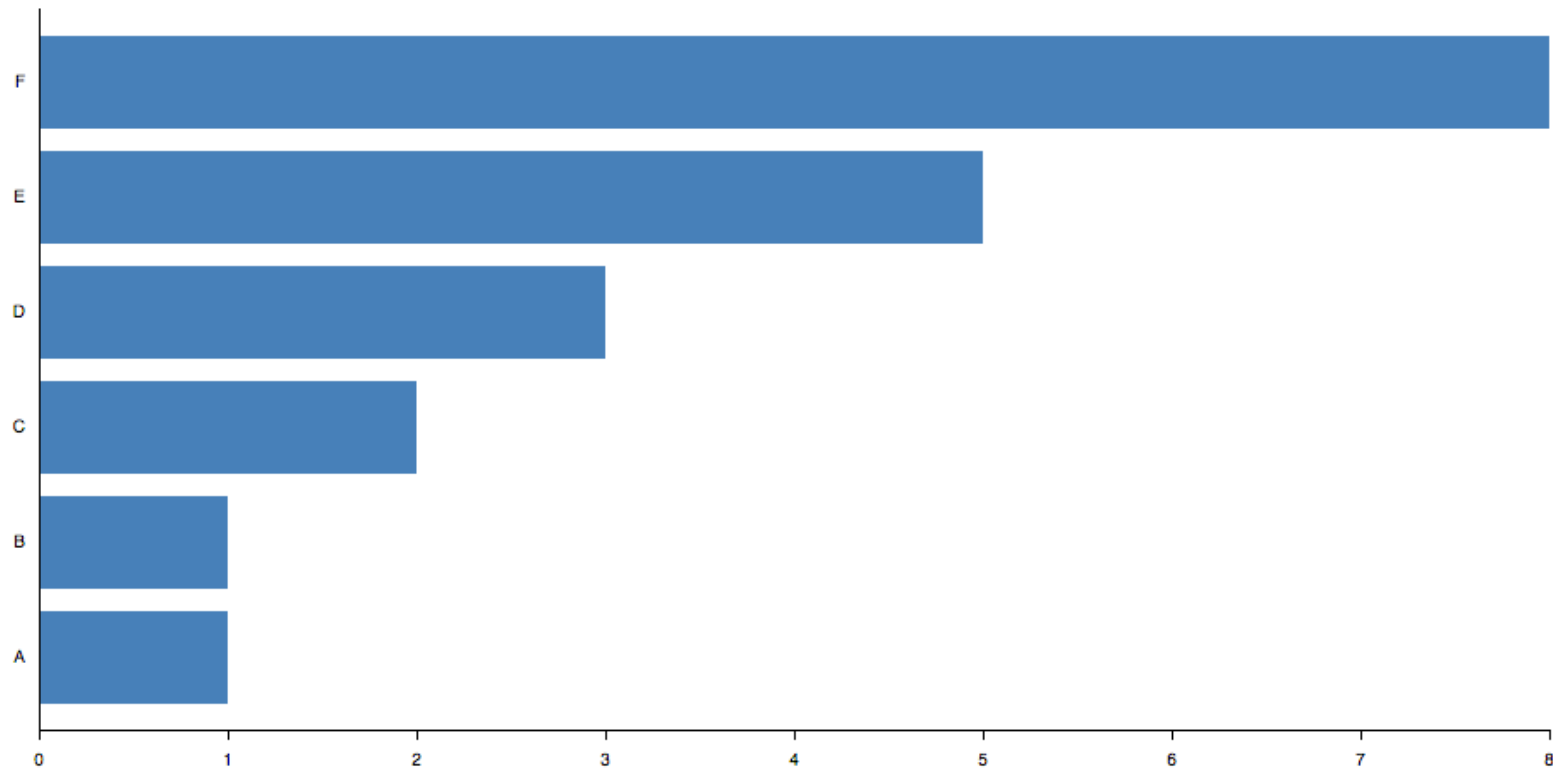
The attr functions takes in input a constant value or a function. The function is called automatically by d3, passing the data (`__data__`) bound to the element and a progressive counter

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

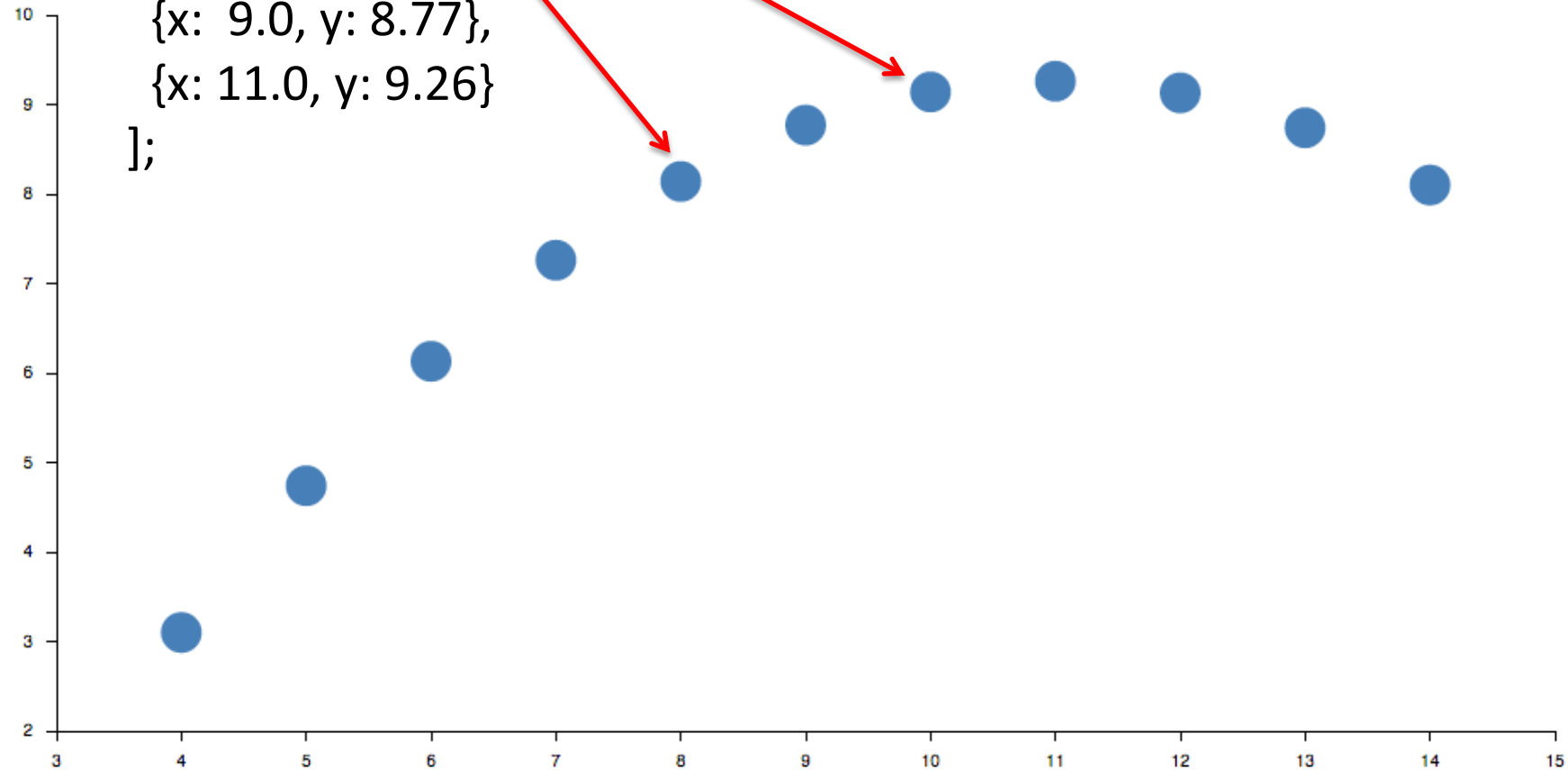
Data can be numbers

```
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}  
];
```

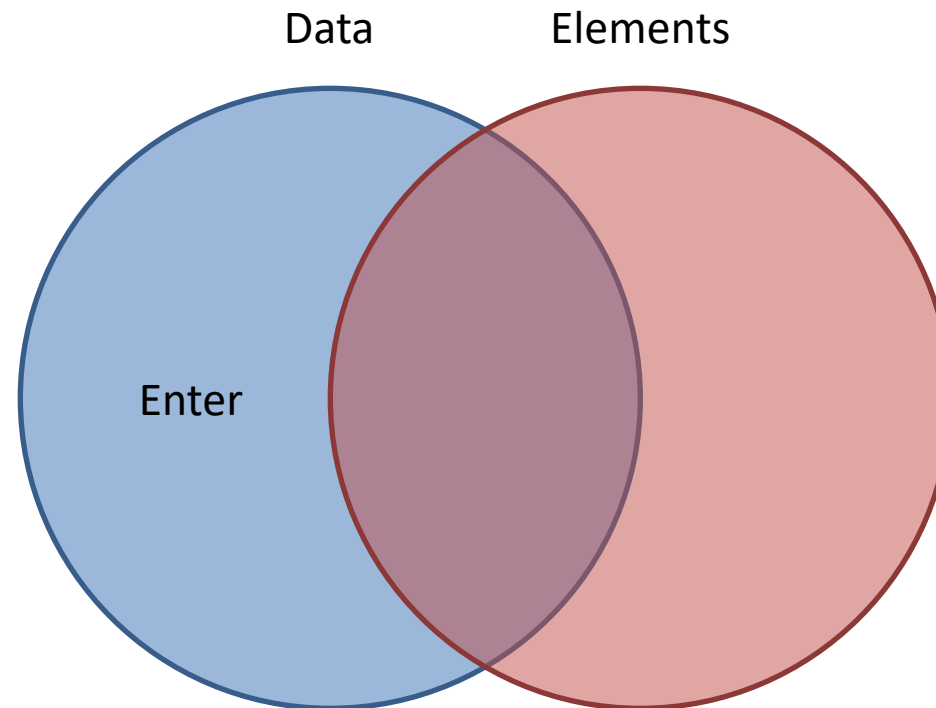


Thinking with Joins

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

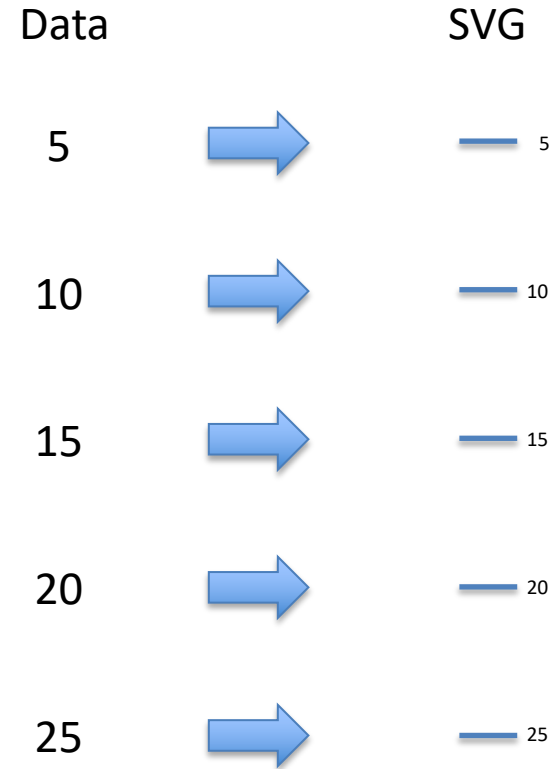


25



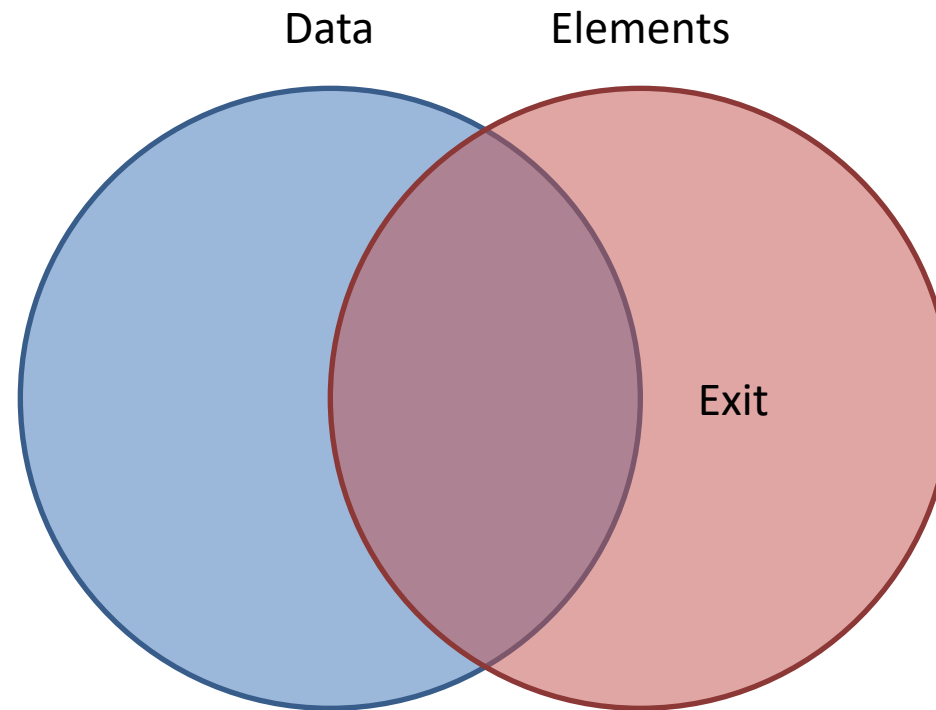
Entering new elements

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



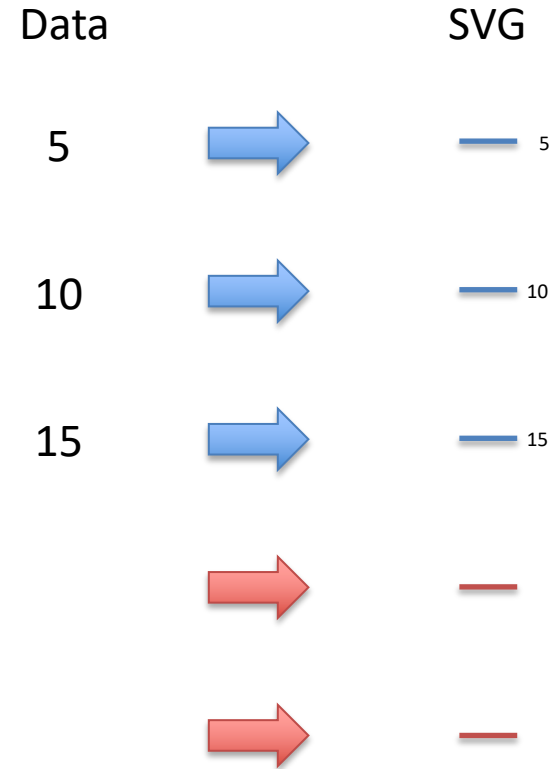
Exit

- Elements that are associated with no data



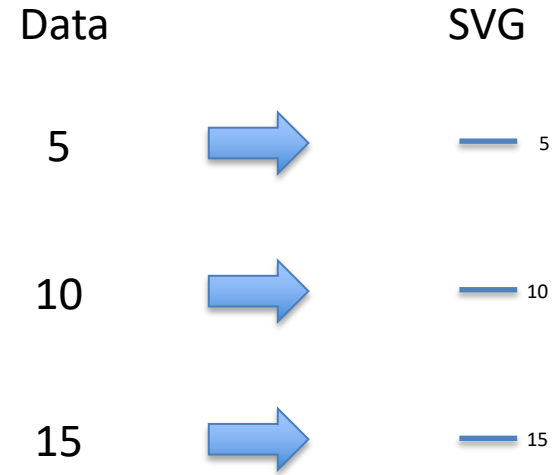
Exiting unnecessary elements

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



Entering new elements

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

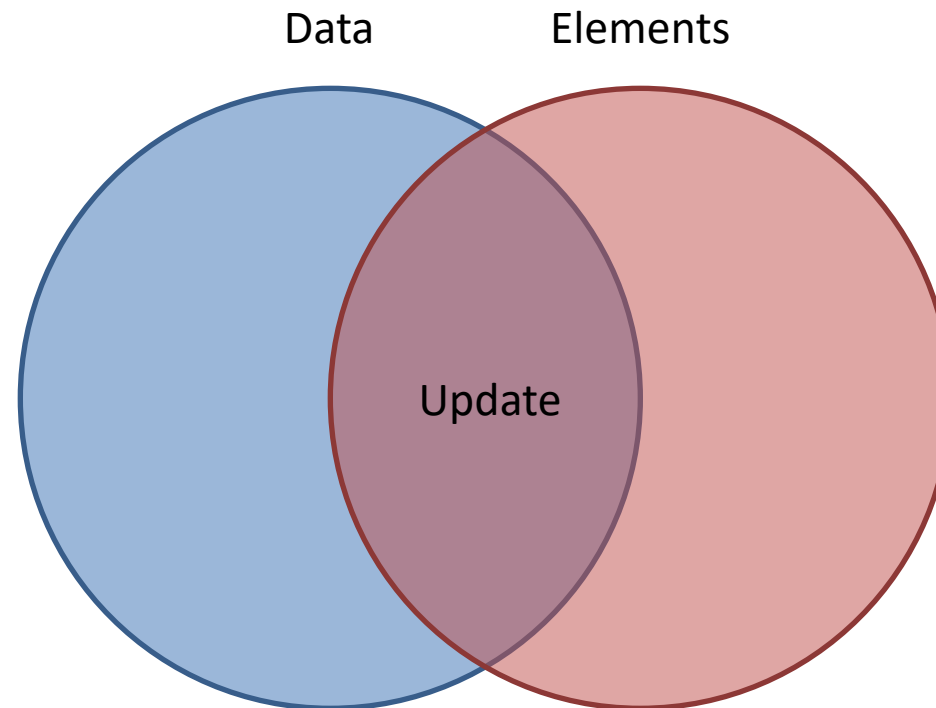


Step 2

DATA ATTRIBUTES TO ELEMENTS ATTRIBUTES

Update

- Data already joined with previous elements

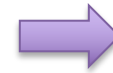


Update existing and new elements with new data

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

Data

5



10



15



SVG

— 5

— 10

— 15

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; });
```

Useful resources

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