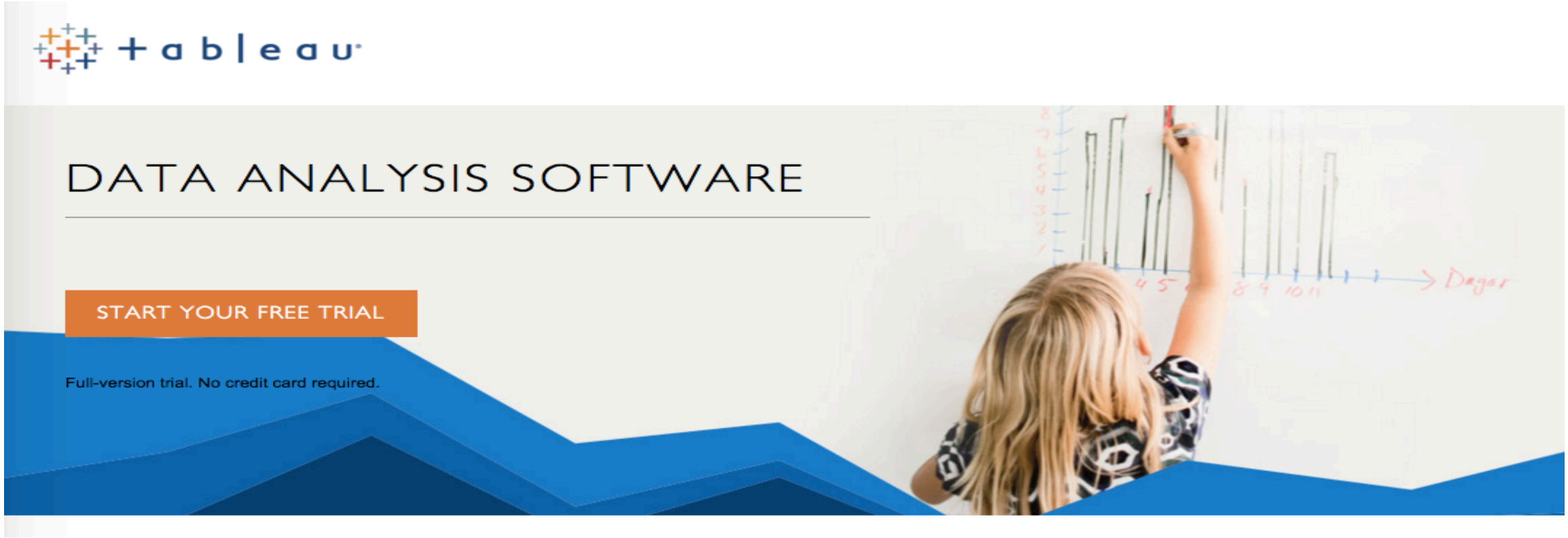
VISUALIZATION ON THE WEB

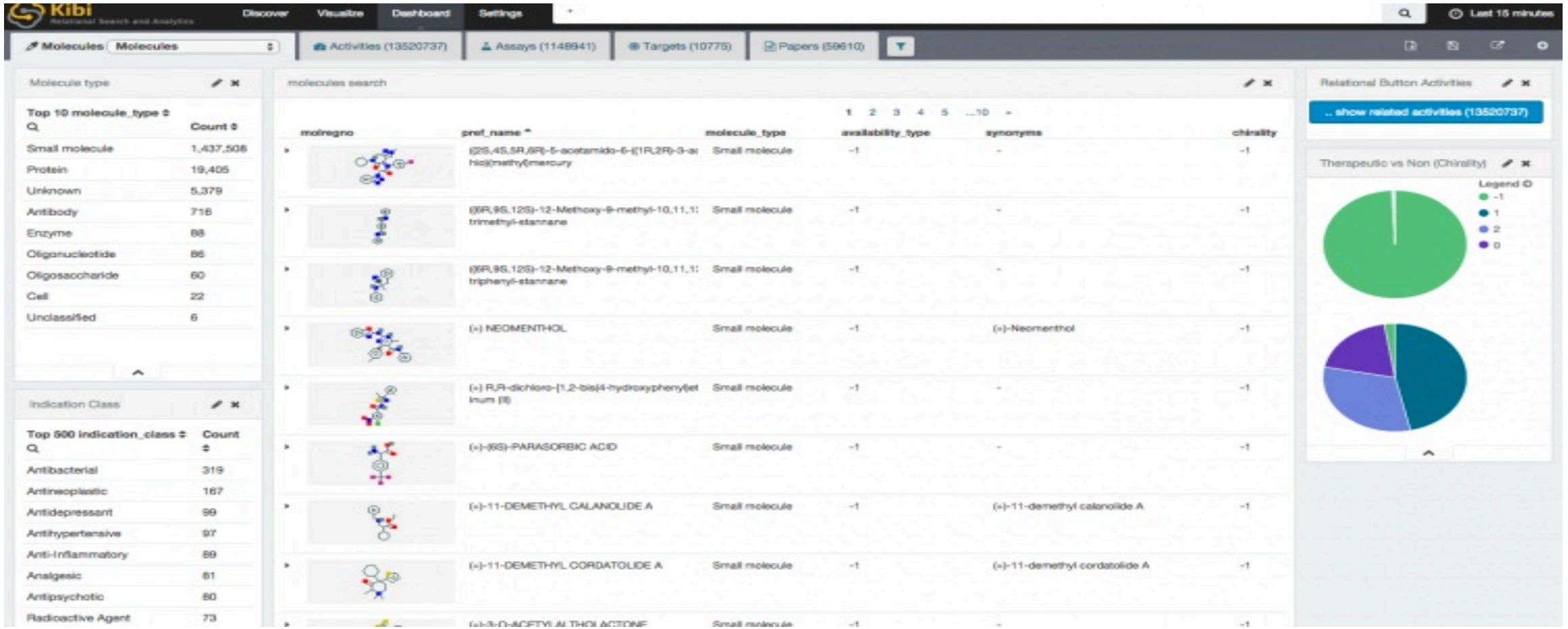
tableau.com


+ a b | e a u
DATA ANALYSIS SOFTWARE
START YOUR FREE TRIAL
Full-version trial. No credit card required.

Kibana GA


Logstash Search
2 days ago to 3 minutes ago
QUERY
bytes:[0 TO 4000000] AND @tags:success
FILTERING
EVENTS OVER TIME
MAP
REVENUE
22%
VOLUME
9%
TOP DESTINATIONS
TOP SOURCES
GEO PAIRS
ALL EVENTS
0 to 100 of 500 available for paging
Fields
Type to filter...
@tags
geo.srcdst
extension
clientip
bytes
id
phpmemory
response
success,security
MY:VN
html
167.12.22.189
8540
1066
200
success,info
IT:MM
png
164.87.170.73
2045
1903
200
success,info
AR:ES
html
222.23.102.238
1801
1133
200
success,info
IN:DZ
html
138.226.66.81
7029
1801
200

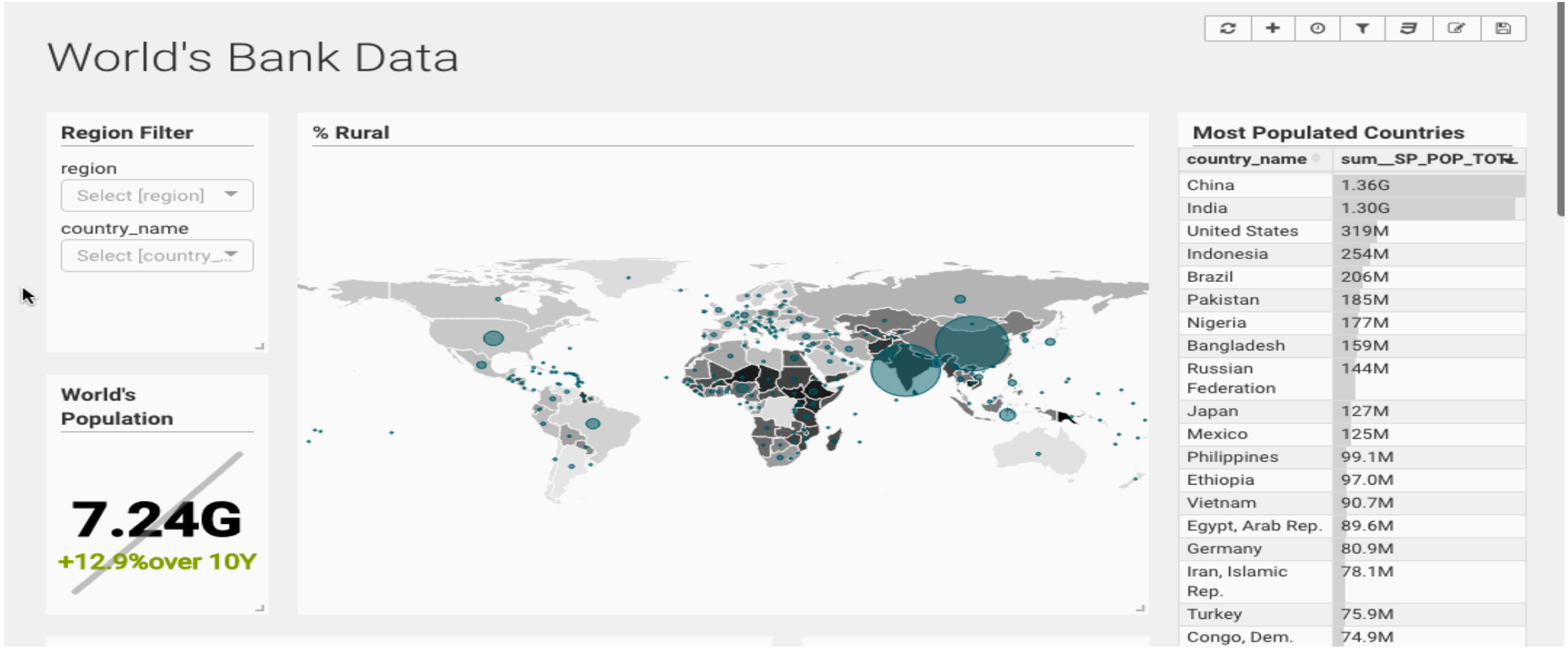
Kibi



https://siren.solutions/kibi/

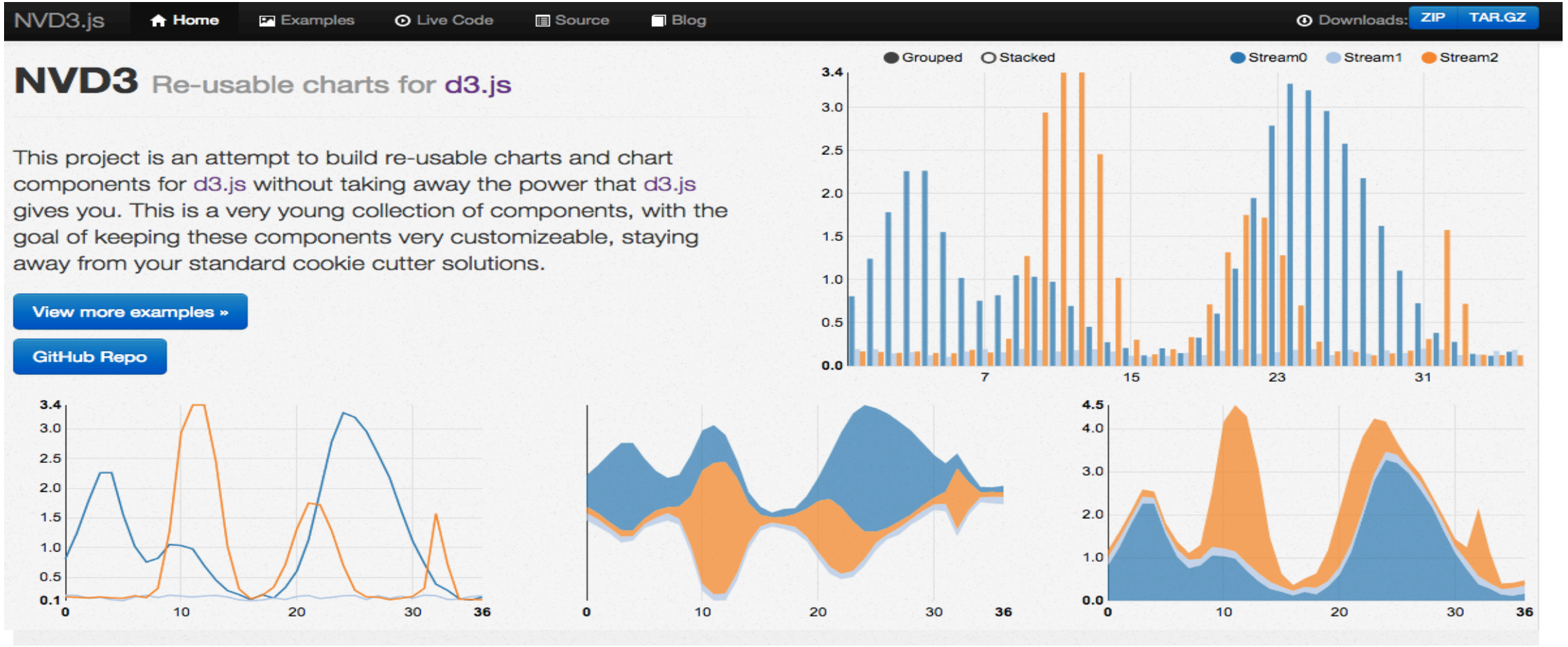
Superset

Superset

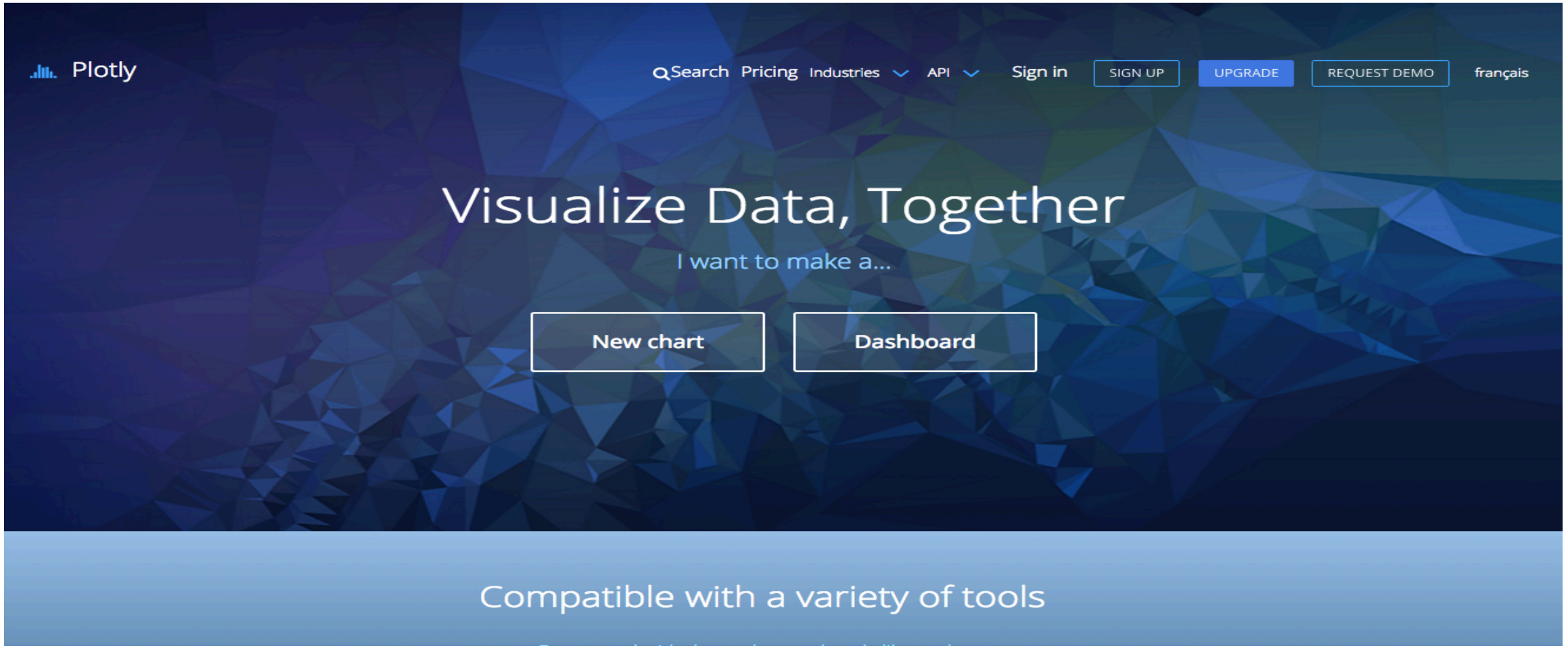


https://github.com/airbnb/superset

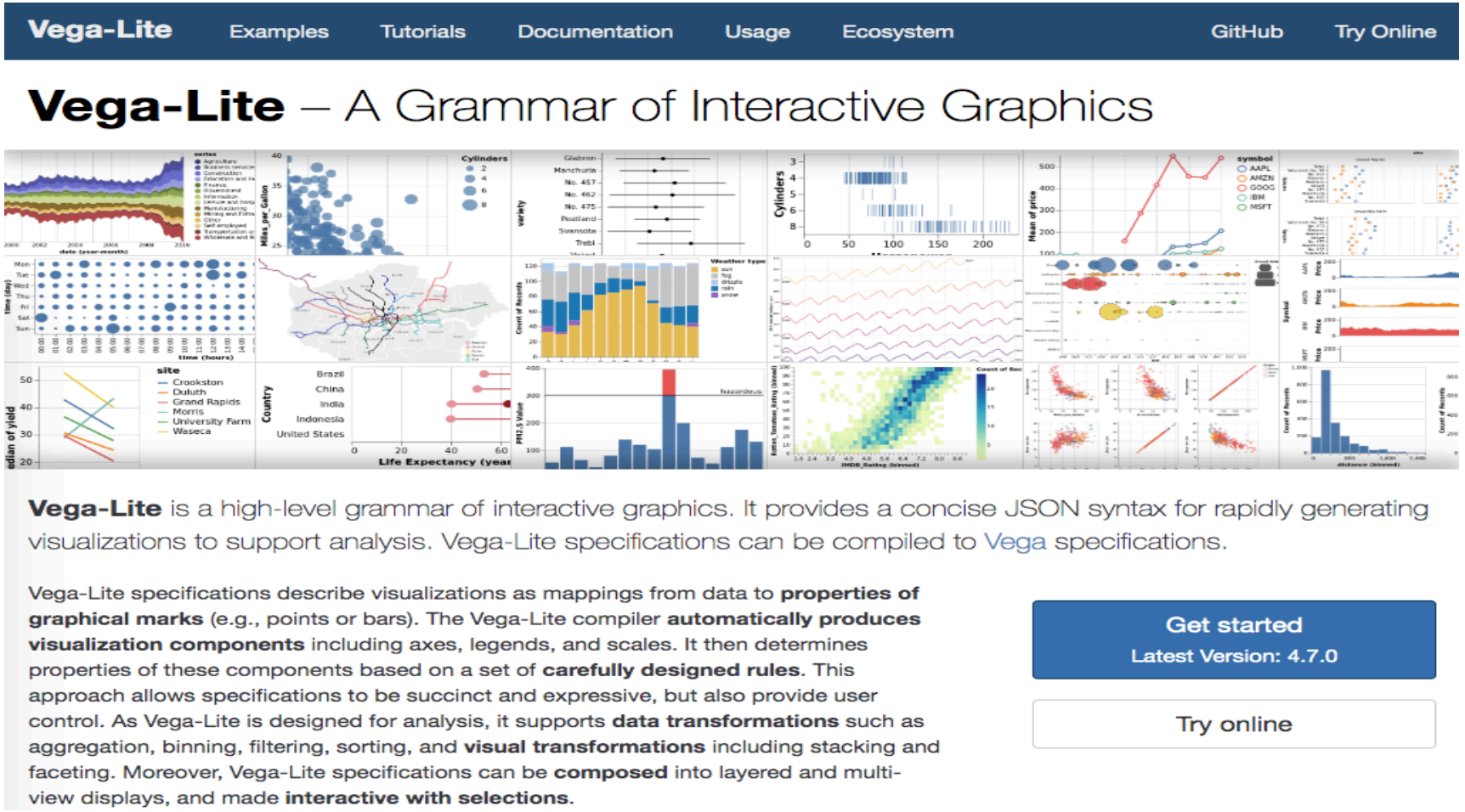
NVD3.js


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 customizeable, staying away from your standard cookie cutter solutions.
View more examples »
GitHub Repo
Grouped
Stacked
Stream0
Stream1
Stream2

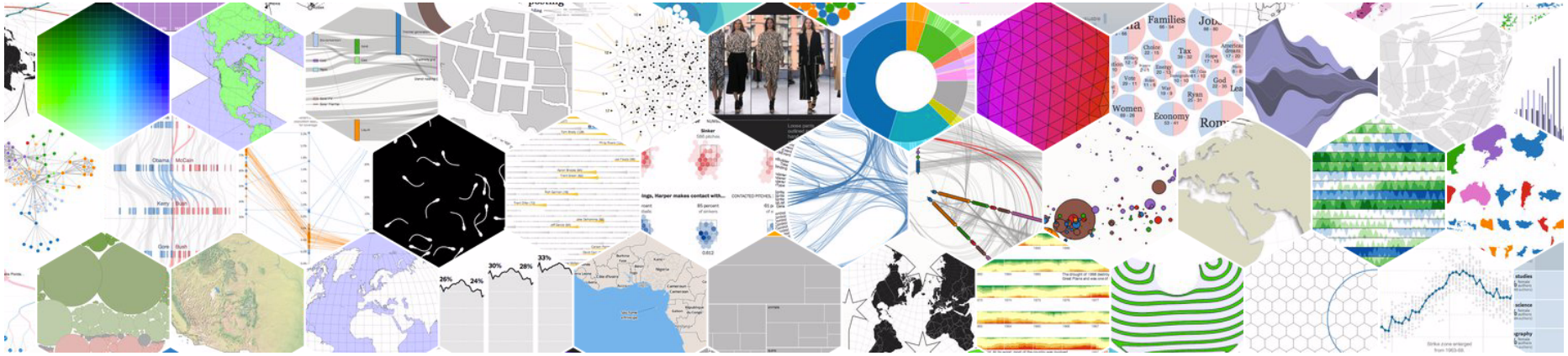
Plot.ly


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

Vega and Vega-lite

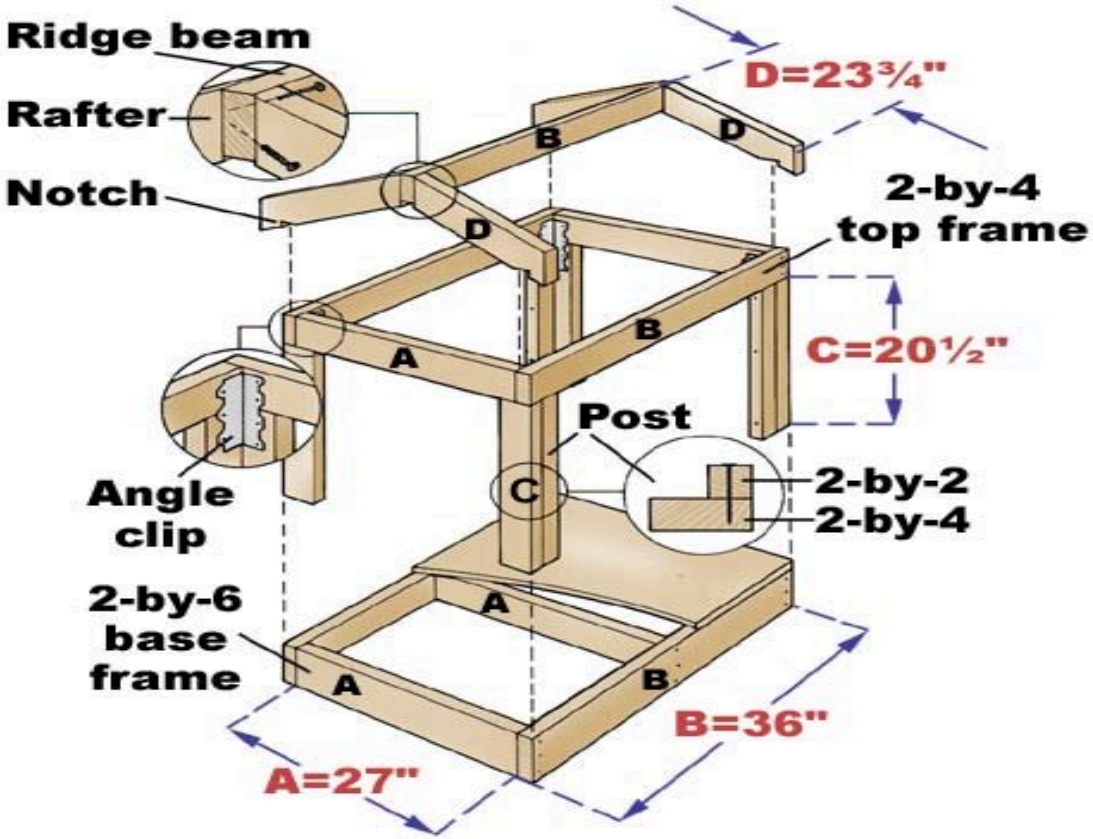


Data-Driven Documents

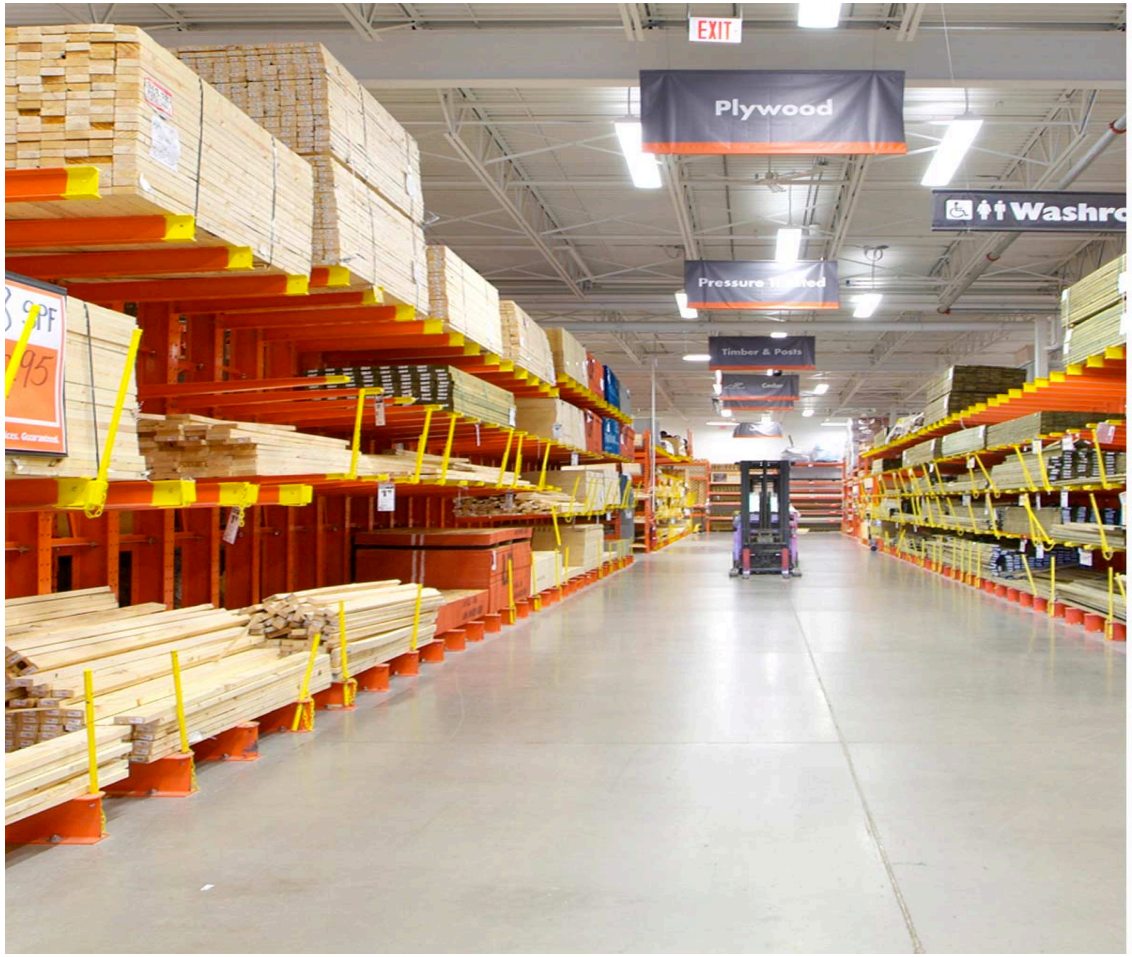


VISUAL ANALYTICS
D3.JS

What is D3?


Ridge beam
Rafter
Notch
D=23¾"
2-by-4
top frame
C=20½"
Post
2-by-2
2-by-4
Angle
clip
2-by-6
base
frame
B=36"
A=27"
A
B
C
D

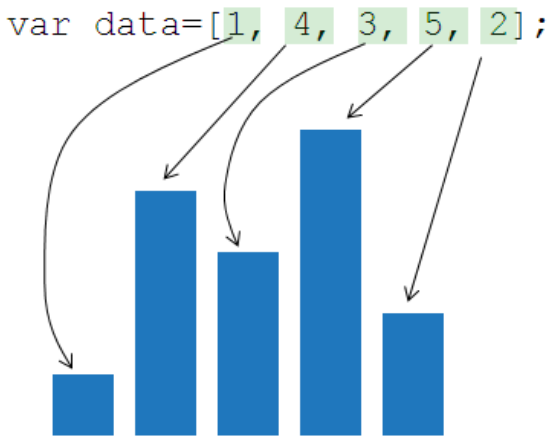
What is D3?


EXIT
Plywood
Timber & Posts


Paint
Colour Solutions
BEHR
RALPH LAUREN
Services
Kitchen
Tools
Storage

What is D3?

- JavaScript library to make beautiful, interactive, browser-based data visualizations.
- D3 stands for **D**ata **D**riven **D**ocuments
- 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

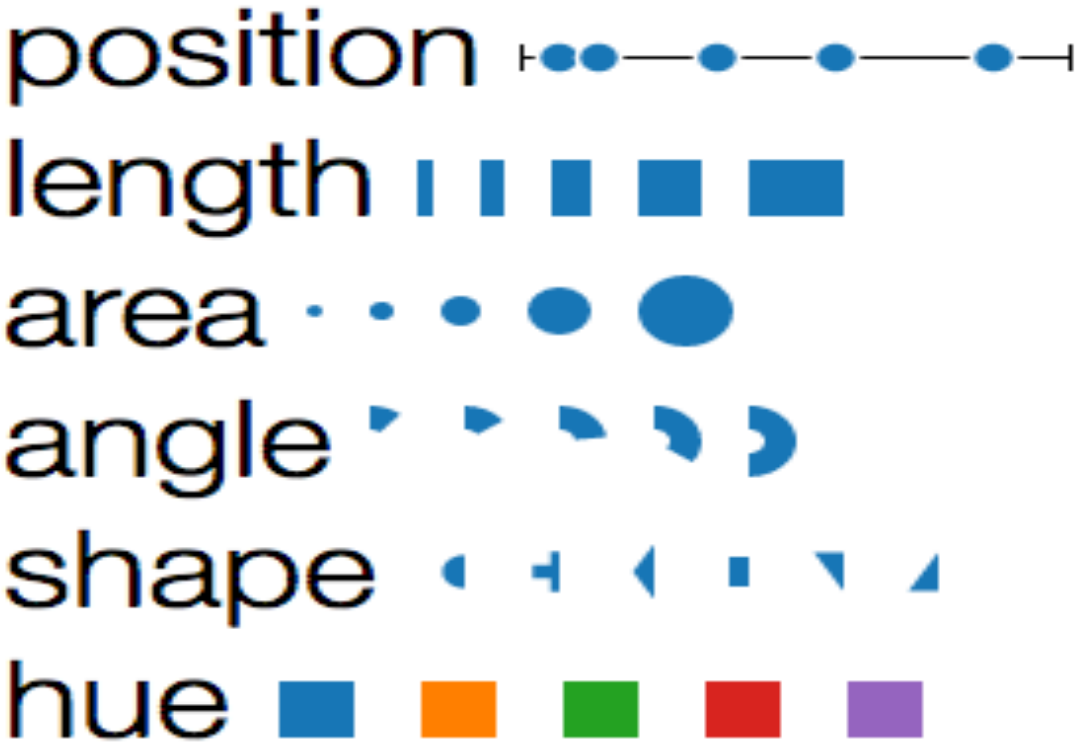


Visualization and Data Graphics

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

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

5

10

15

20

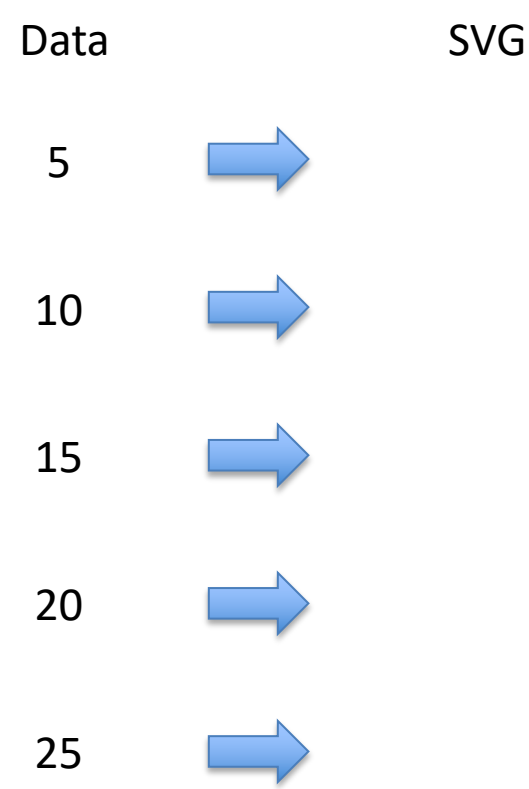
25

SVG

Method data joins data with document elements

Selection should correspond to data

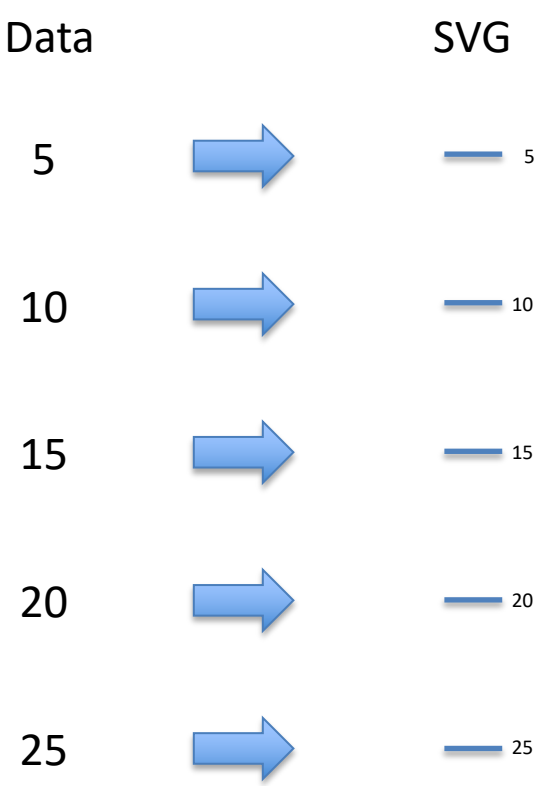
```
var numbers =
[5,10,15,20,25];
var lines =
svg.selectAll(”line")
    .data(numbers)
    .enter().append(”line");
```



Method enter specifies the action for missing elements

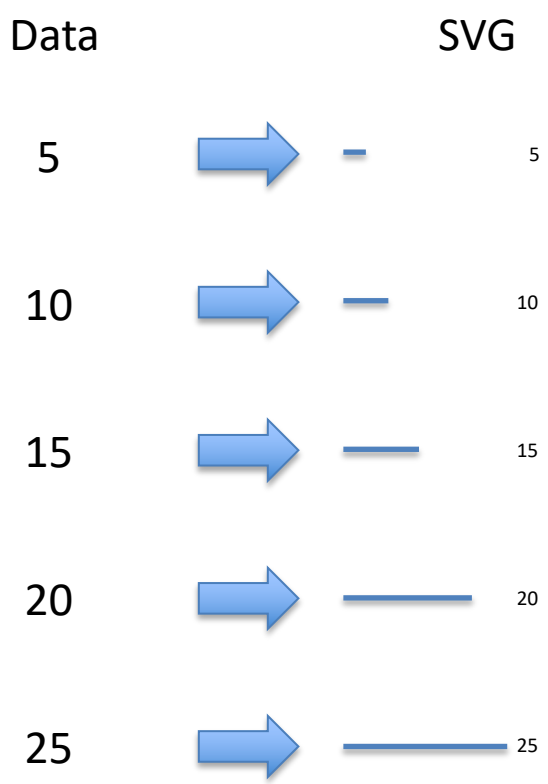
Selection should correspond to data

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



Selection should correspond to data

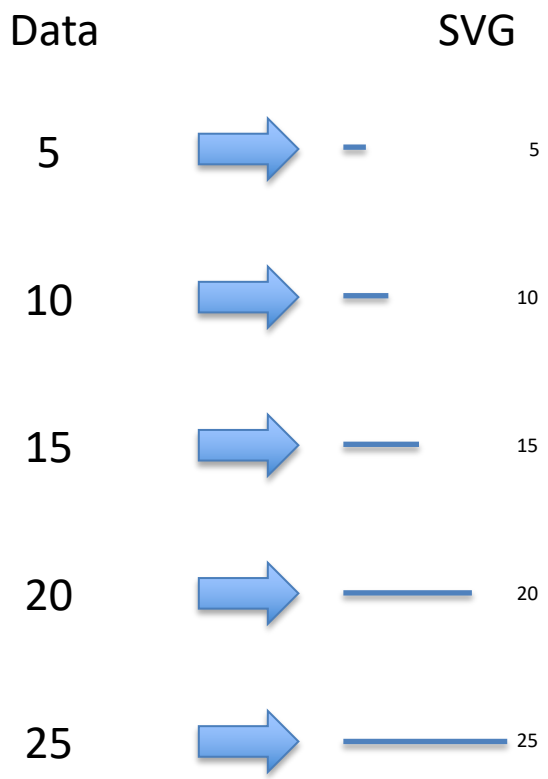
```
var numbers =
[5,10,15,20,25];
var lines =
svg.selectAll(”line")
    .data(numbers)
    .enter().append(”line");

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;
}
```



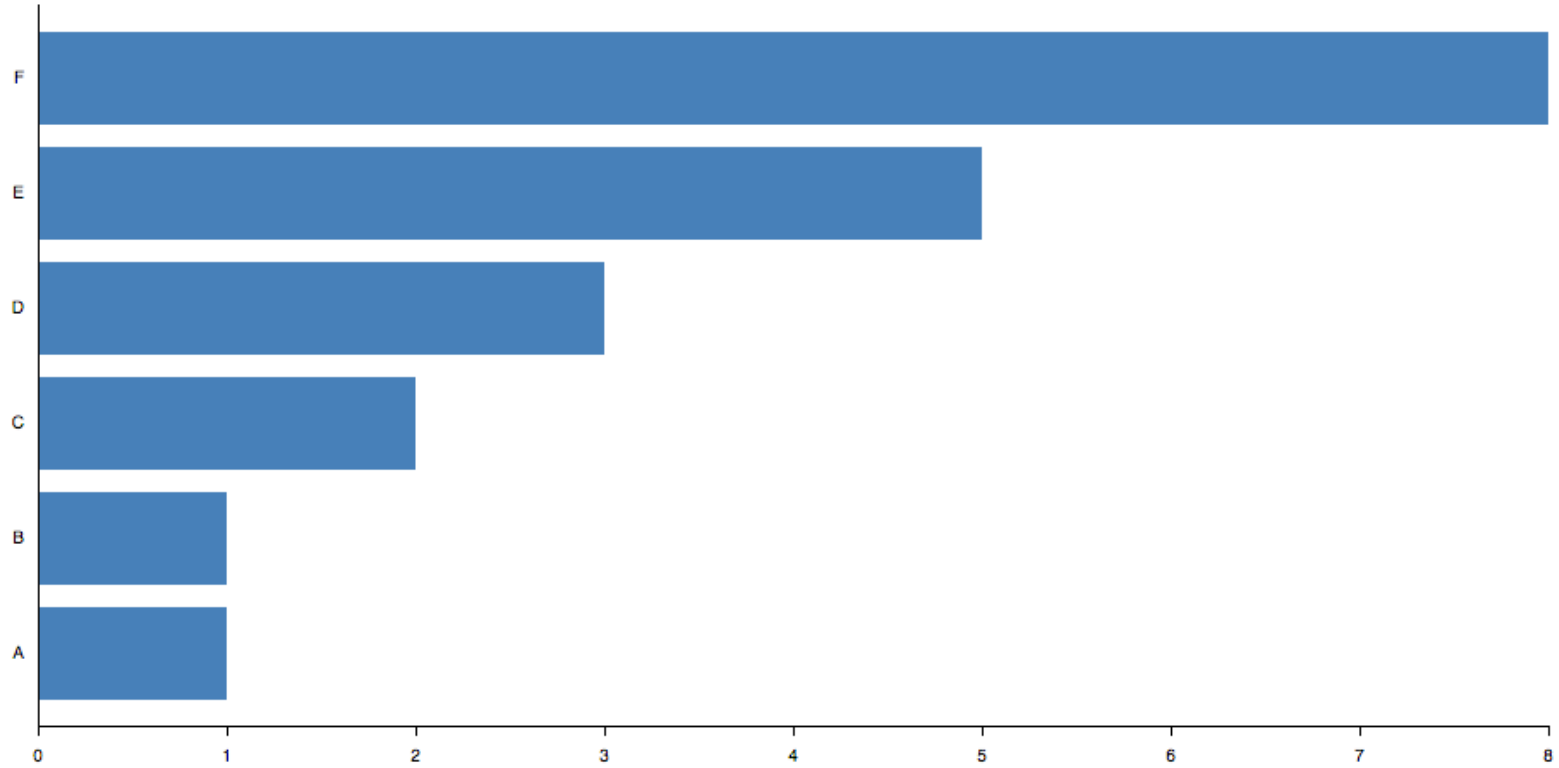
The `attr` functions takes in input a constant value or a function. The function is called automatically by d3, passing the data (`__data__`) bound to the element and a progressive counter

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

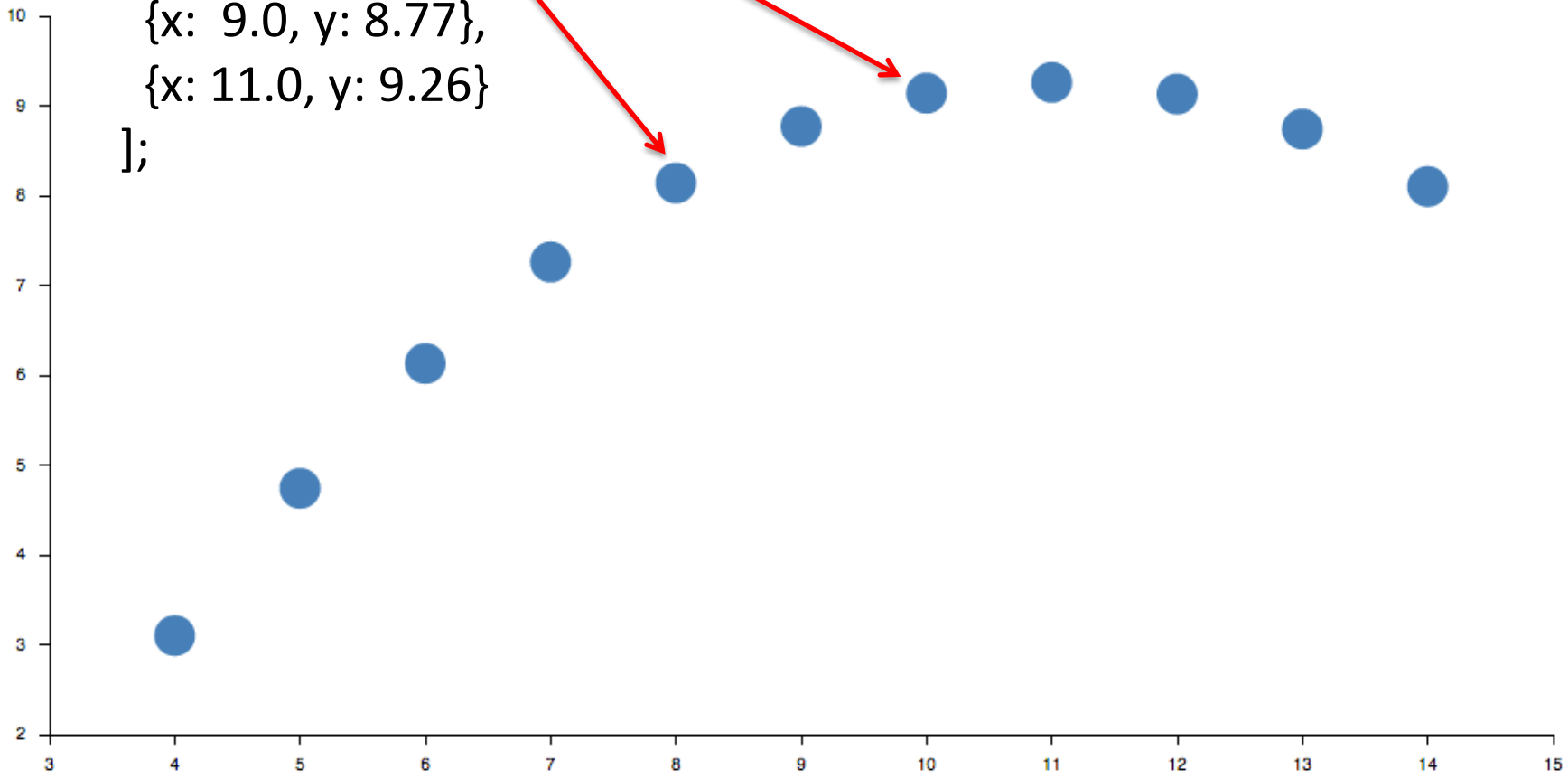
Data can be numbers

```
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}
];
```

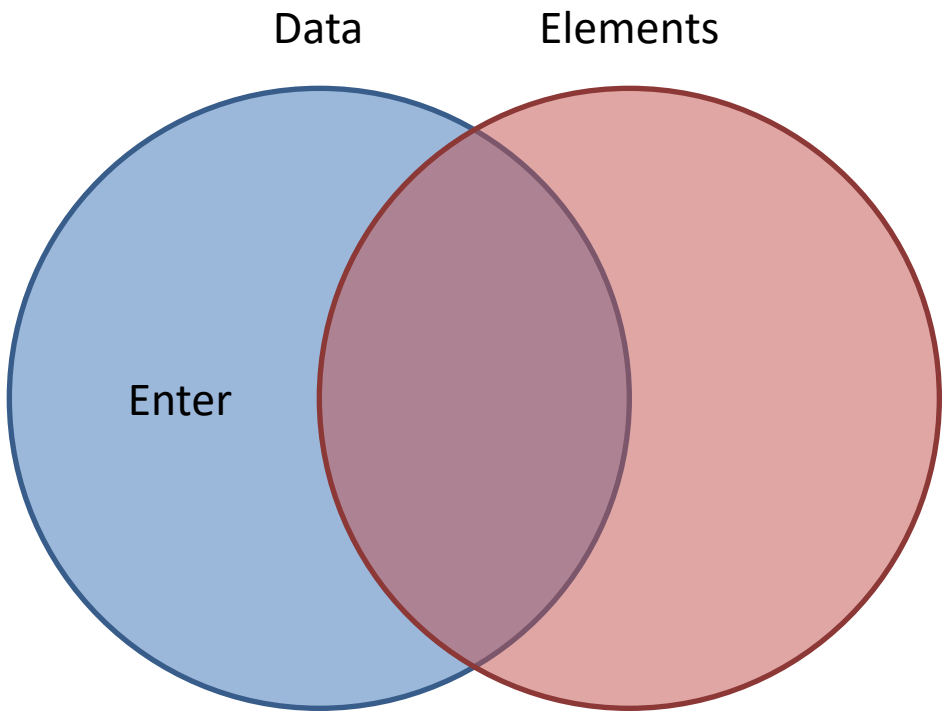


Thinking with Joins

ENTER, EXIT, AND UPDATE

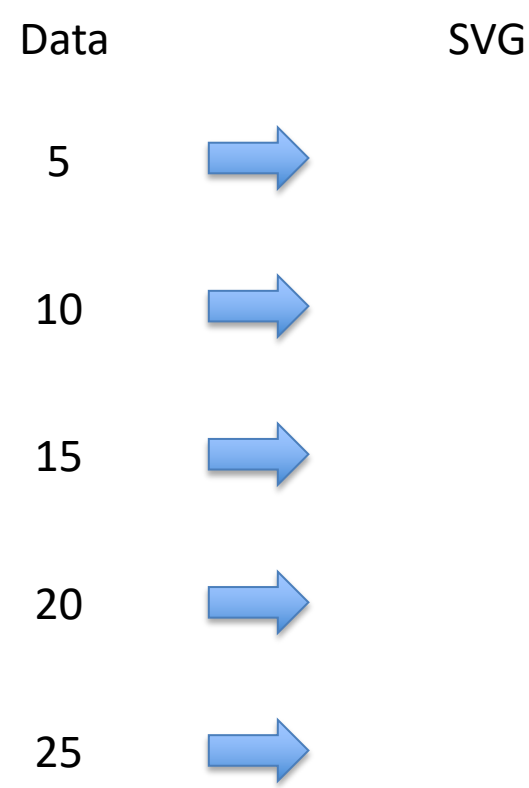
Enter

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



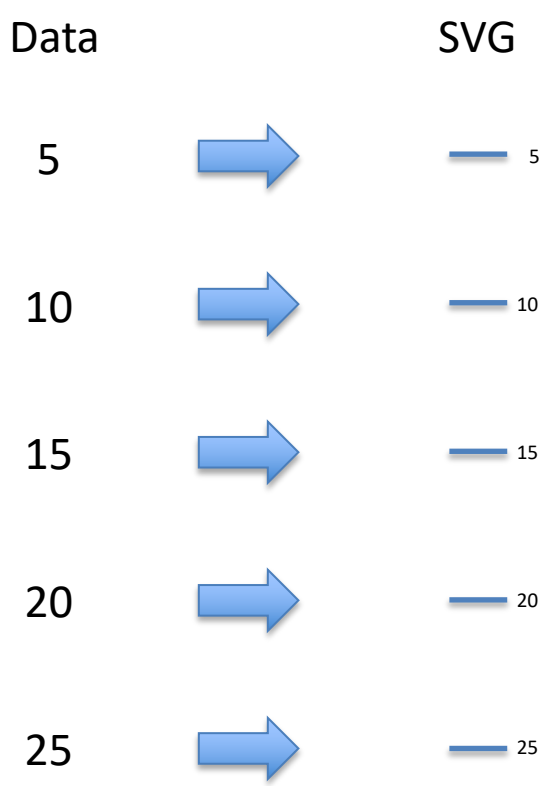
Entering new elements

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

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



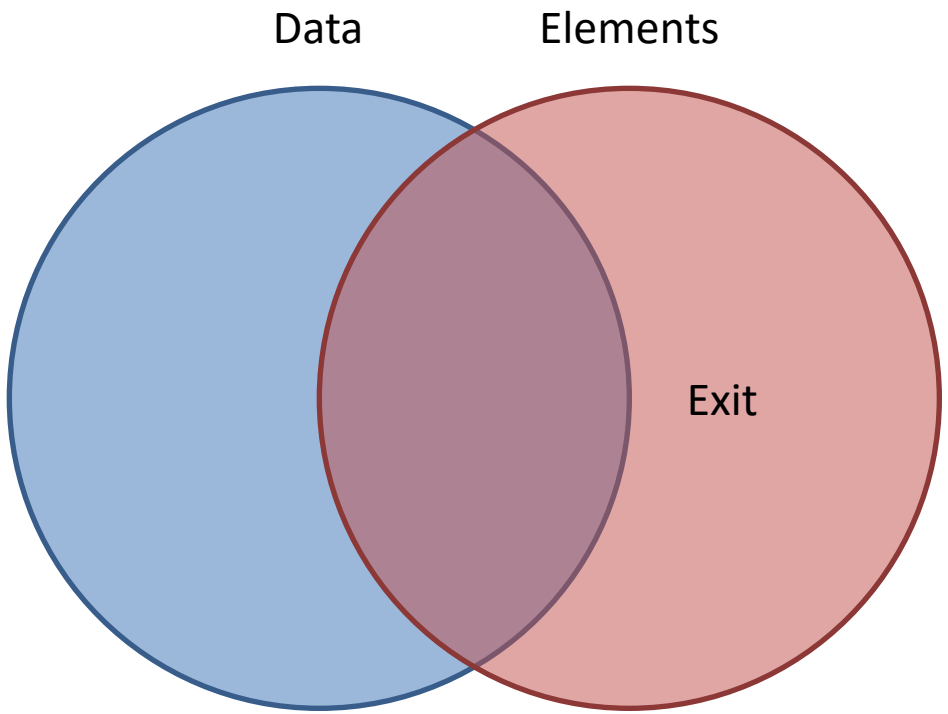
Entering new elements

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

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



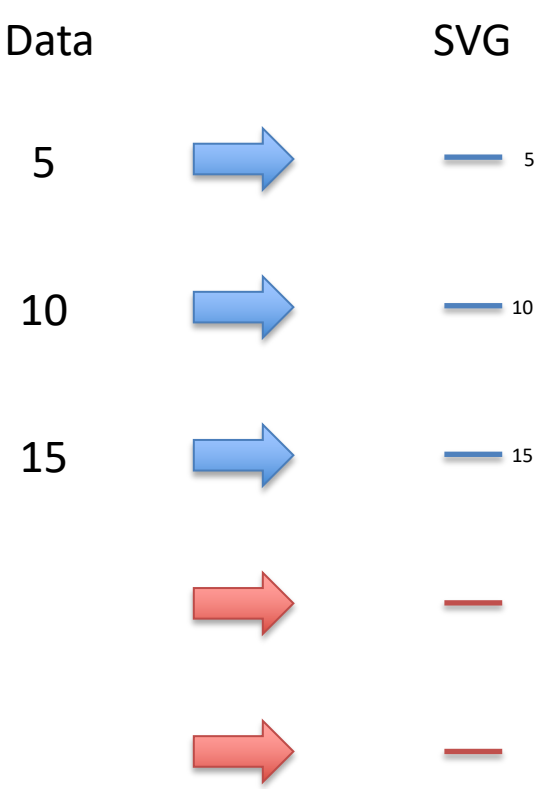
Exit

- Elements that are associated with no data



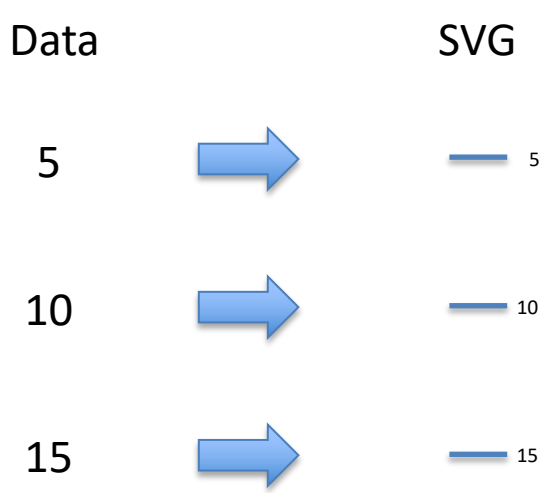
Exiting unnecessary elements

```
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();
```

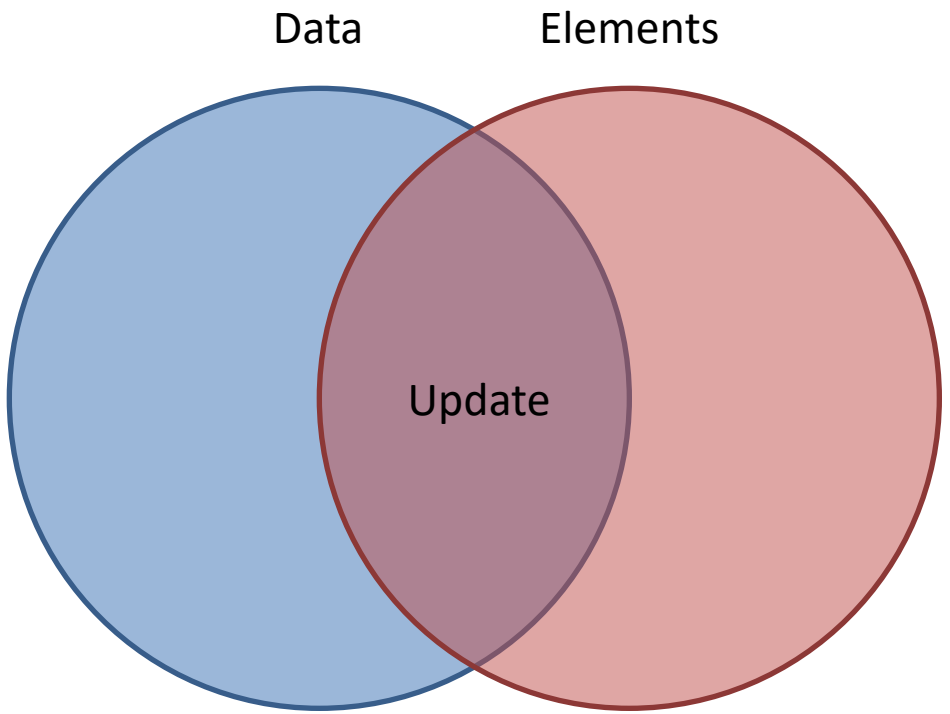


Step 2

DATA ATTRIBUTES TO ELEMENTS ATTRIBUTES

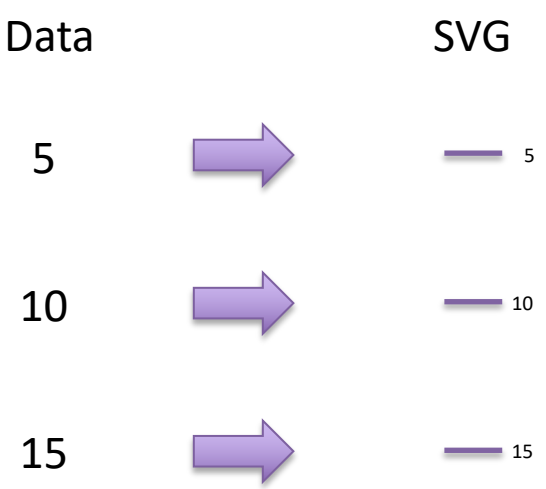
Update

- 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));
```



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/