# VISUALIZATION ON THE WEB

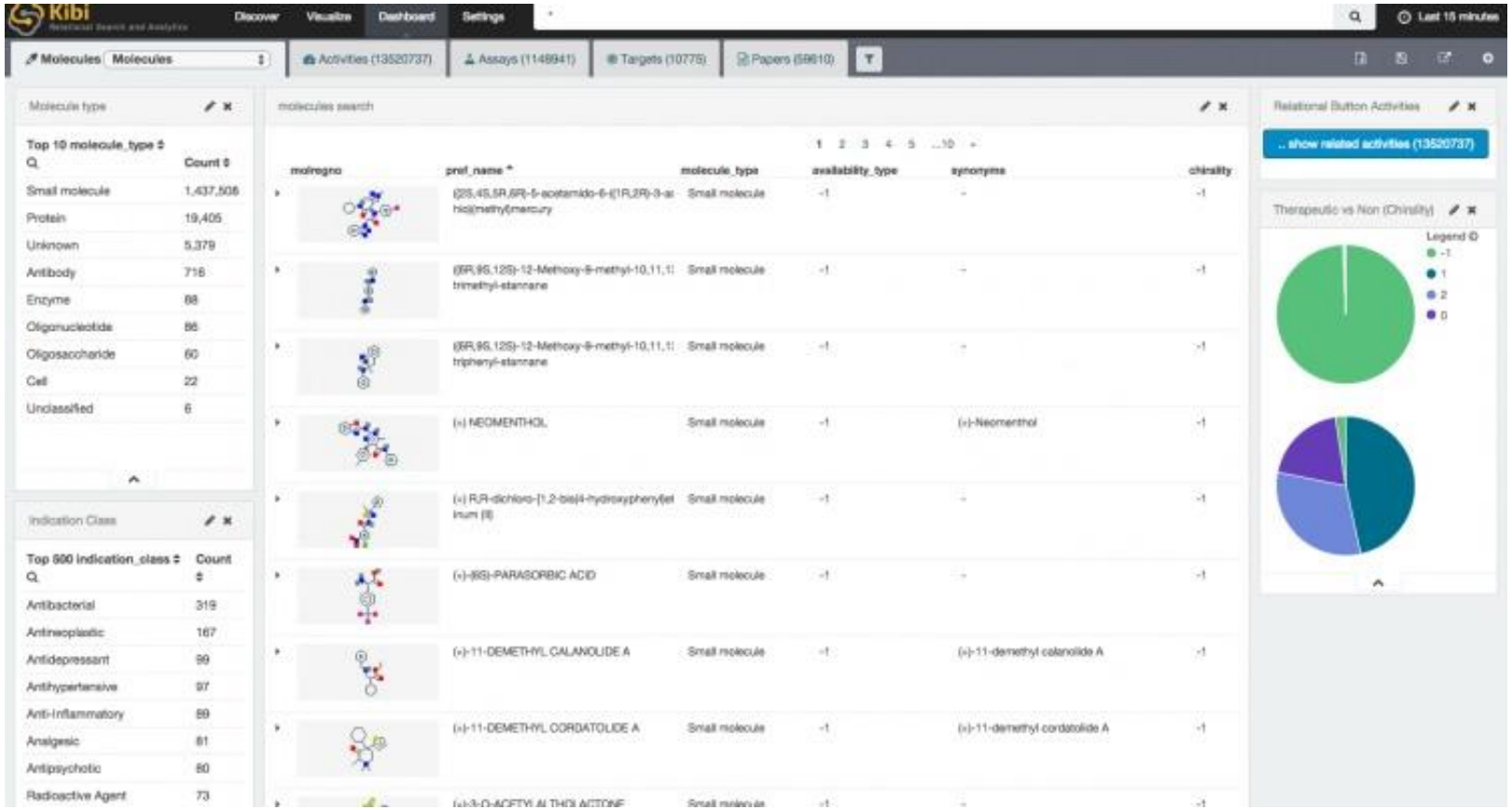
# TABLEAU.COM


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

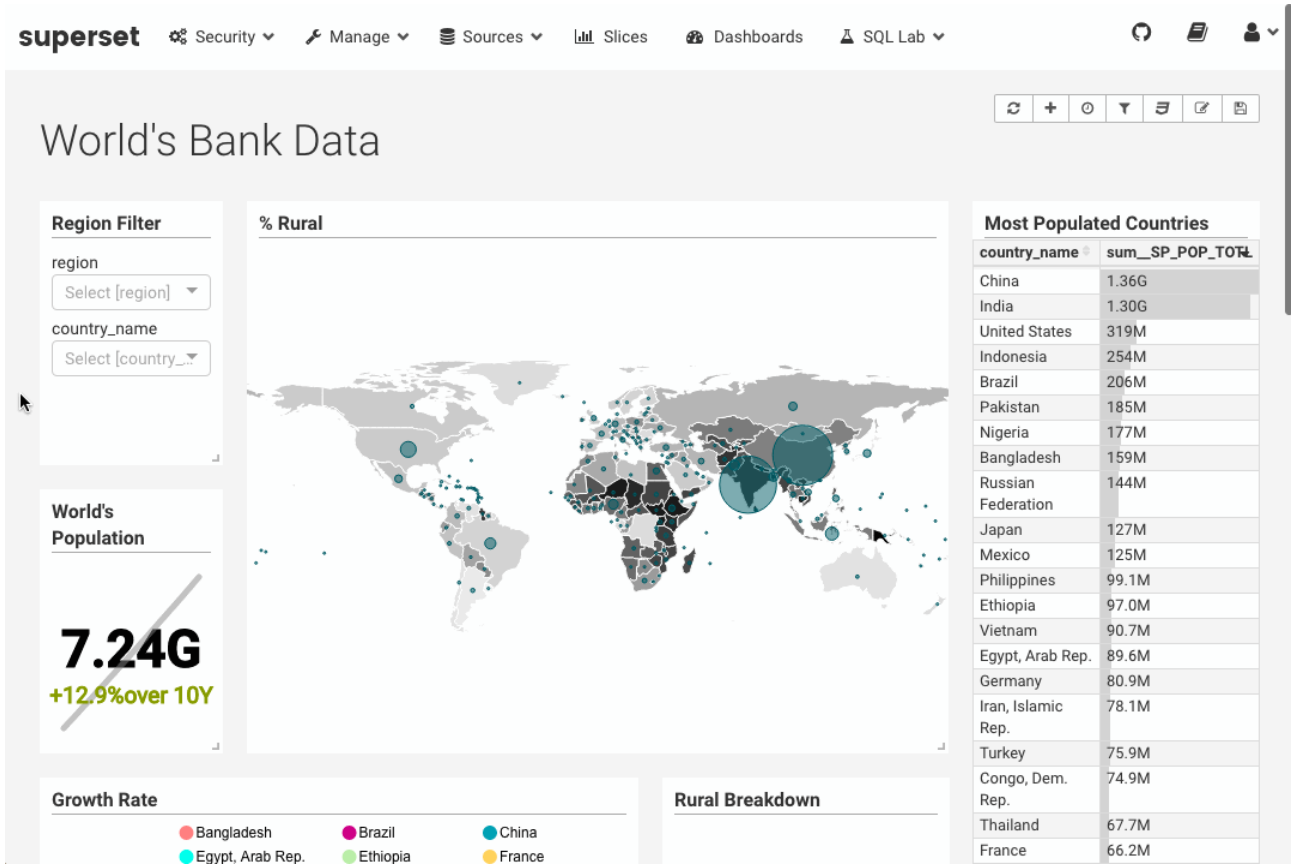
# KIBANA GA


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
Fields

# KIBI

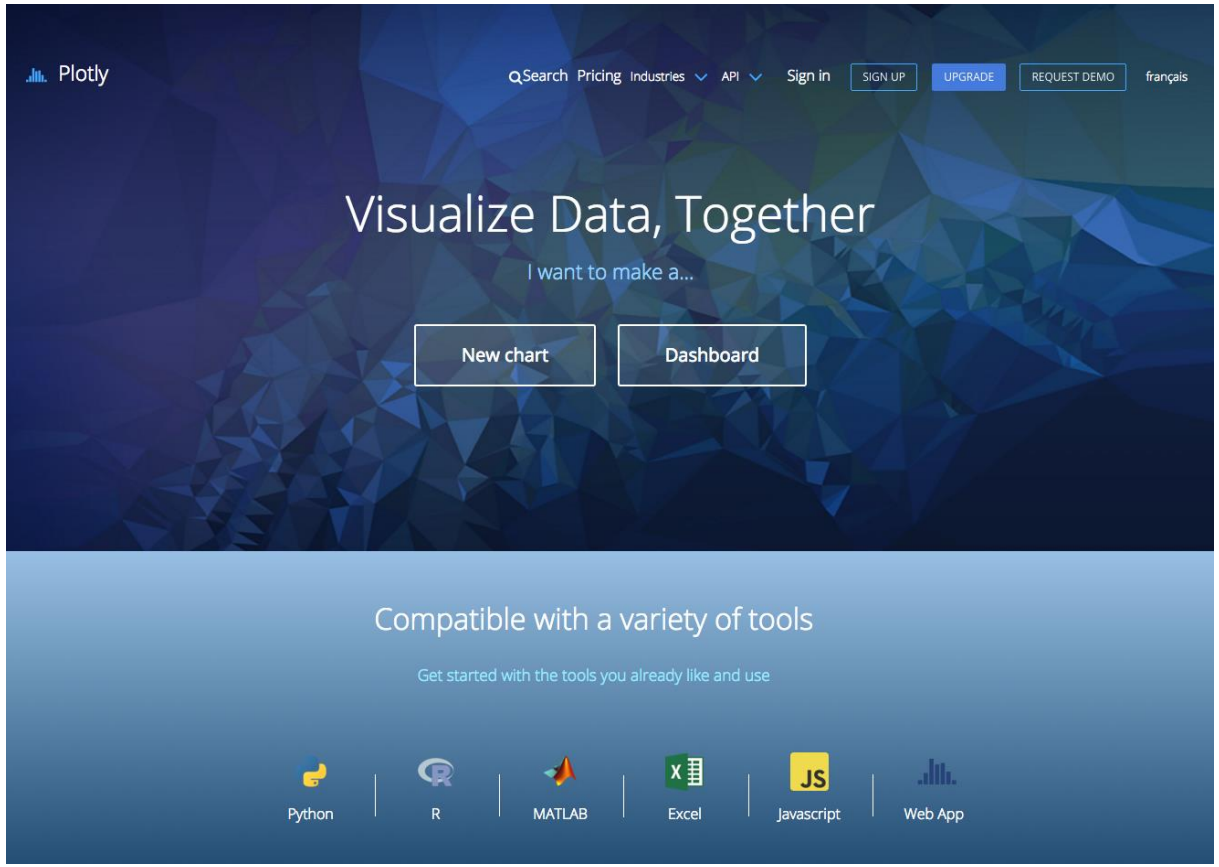

Kibi
Discover
Visualize
Dashboard
Settings
Molecules
Activities (13520737)
Assays (1148941)
Targets (10775)
Papers (59610)
Molecule type
Top 10 molecule_type
Count
Small molecule 1,437,506
Protein 19,405
Unknown 5,379
Antibody 716
Enzyme 88
Oligonucleotide 86
Oligosaccharide 60
Cell 22
Unclassified 6
Indication Class
Top 500 indication_class
Count
Antibacterial 319
Antineoplastic 167
Antidepressant 99
Antihypertensive 97
Anti-Inflammatory 89
Analgesic 81
Antipsychotic 80
Radioactive Agent 73
molecules search
molregno
pref_name
molecule_type
availability_type
synonyms
chirality
Relational Button Activities
.. show related activities (13520737)
Therapeutic vs Non (Chirality)
Legend ID

# Superset

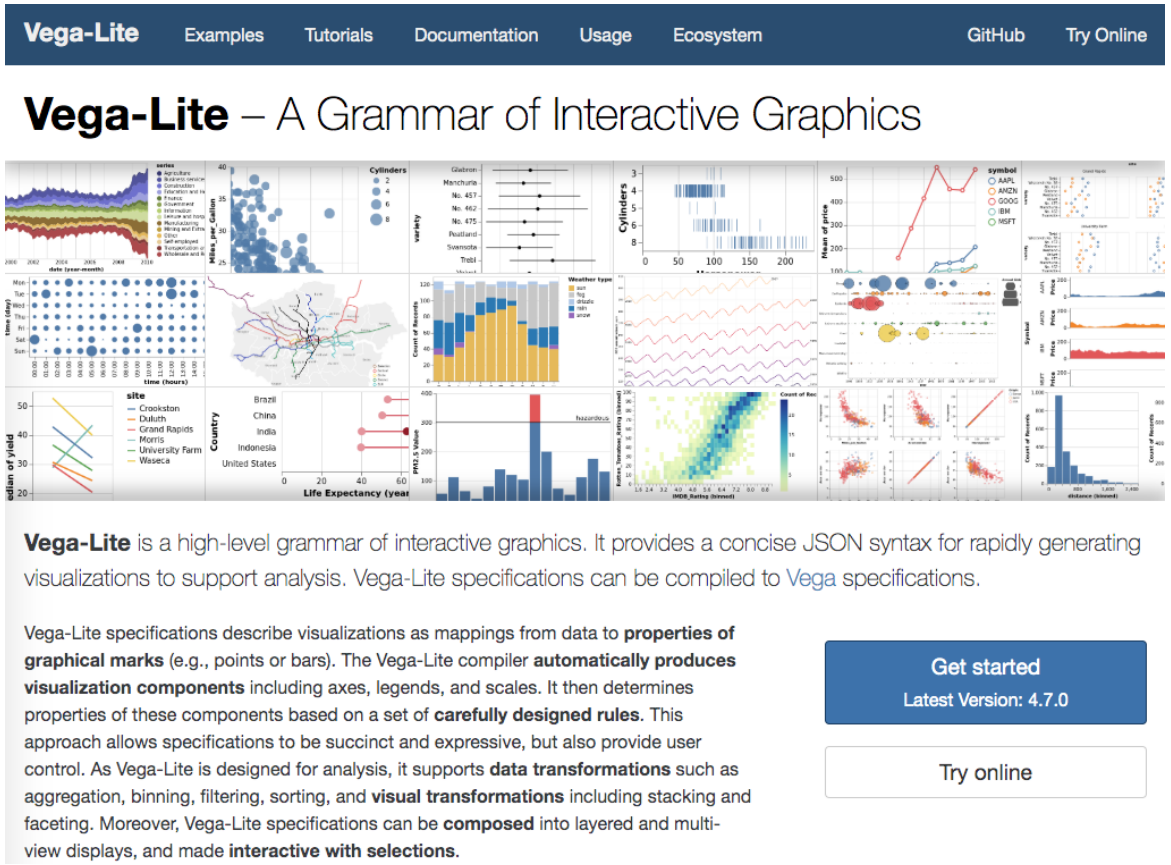



https://github.com/airbnb/superset

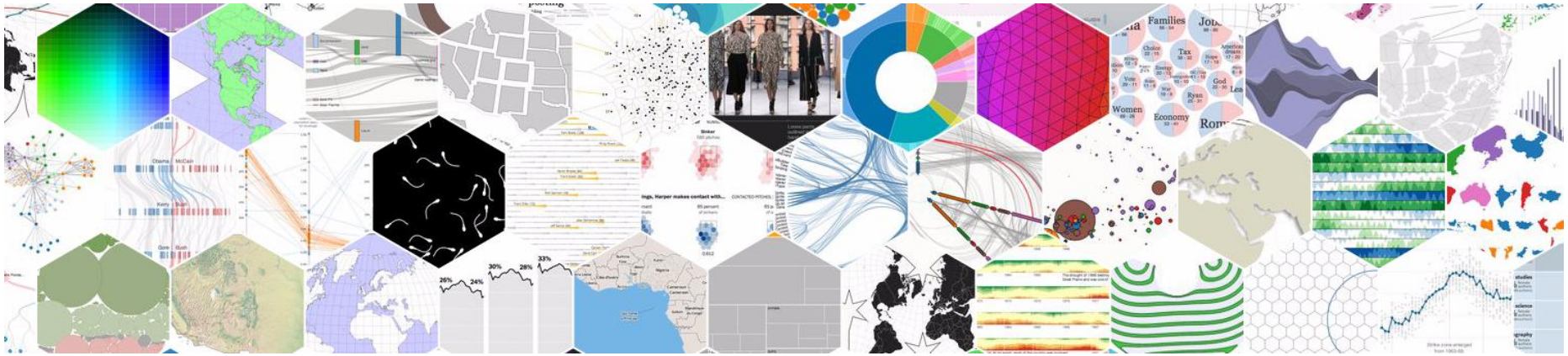
# PLOT.LY


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
Get started with the tools you already like and use
Python
R
MATLAB
Excel
Javascript
Web App

# VEGA AND VEGA-LITE

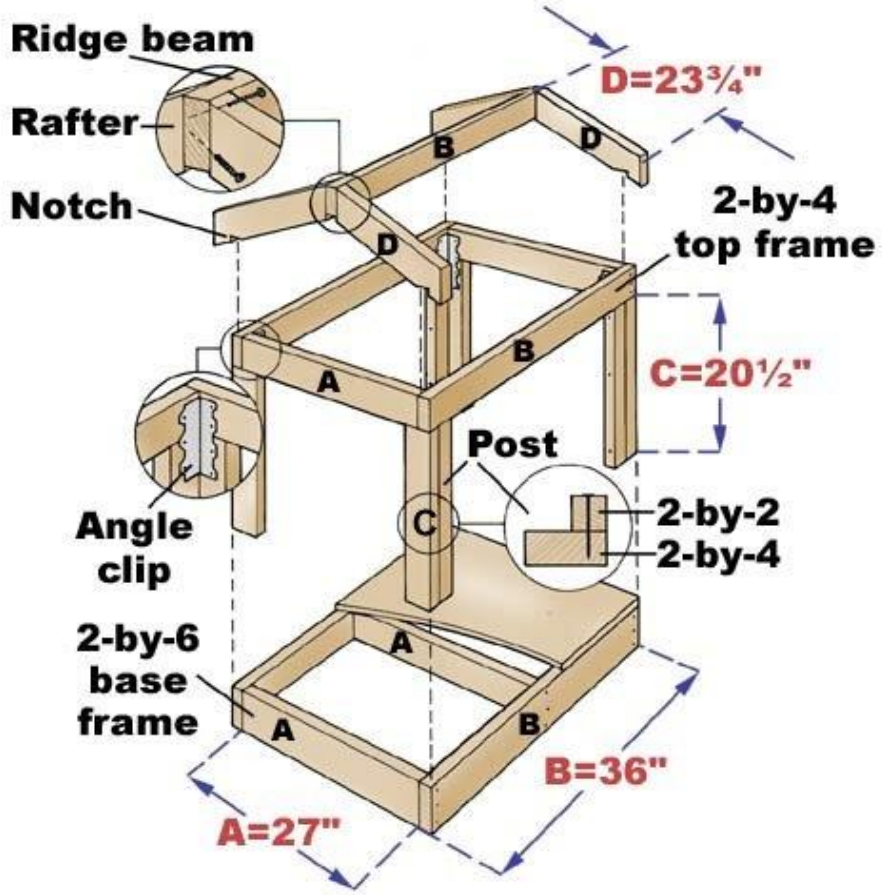


Data-Driven Documents



# VISUAL ANALYTICS
# D3.JS

# WHAT IS D3?

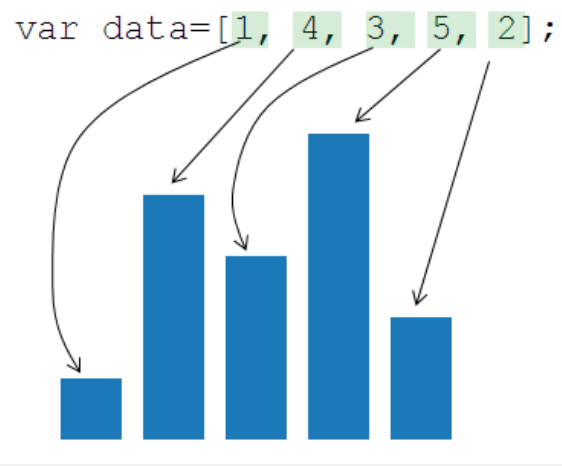

Ridge beam
Rafter
Notch
D=23¾"
2-by-4
top frame
C=20½"
Post
2-by-2
2-by-4
Angle
clip
2-by-6
base
frame
B=36"
A=27"
A
B
C
D

# WHAT IS D3?


EXIT
Plywood
Paint
Colour Solutions
Services
Kitchen
BEHR
RALPH LAUREN

# 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

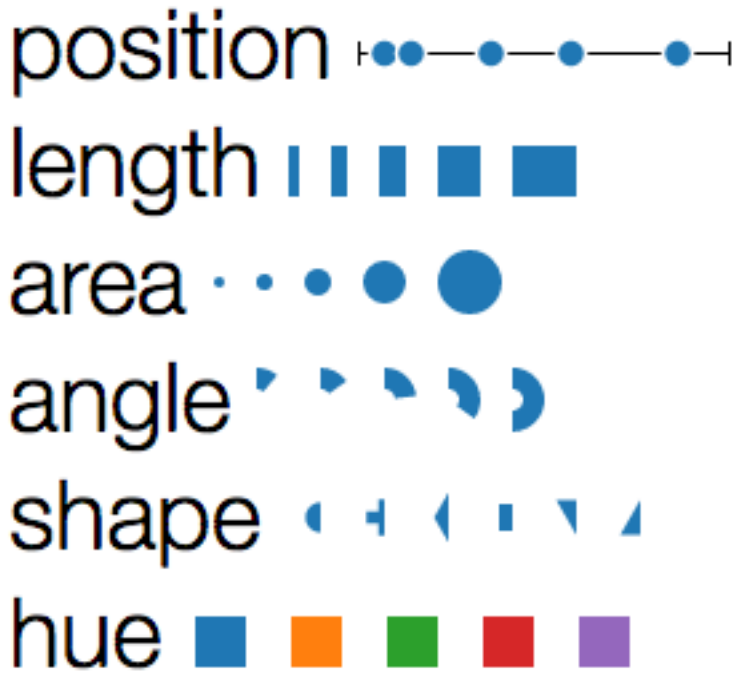


# VISUALIZATION AND DATA GRAPHICS

**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
let 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

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

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

# MODIFY EXISTING ELEMENTS

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

let 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

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

let lines =
svg.selectAll(”line")
        .data(numbers)
        .enter().append(”line
");
```

Data

SVG

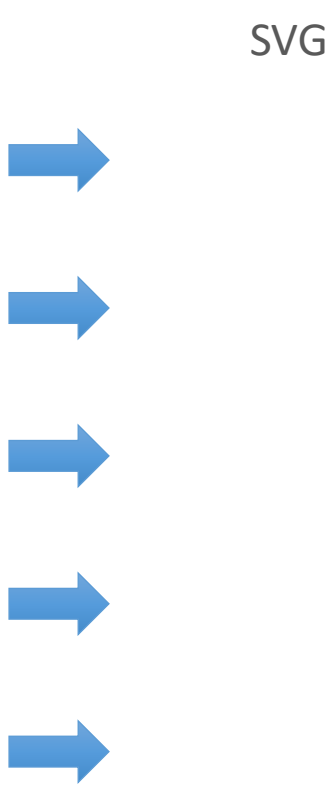
# SELECTION SHOULD CORRESPOND TO DATA

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


Method data joins data with document elements

# SELECTION SHOULD CORRESPOND TO DATA

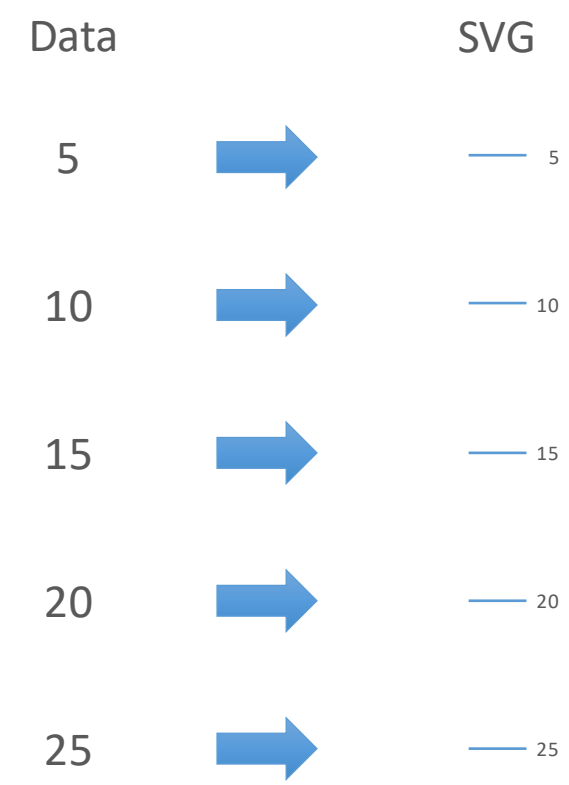
```
let numbers =
[5,10,15,20,25];
let lines =
svg.selectAll(”line")
      .data(numbers)
      .enter().append(”line
");
```




Method enter specifies the action for missing elements

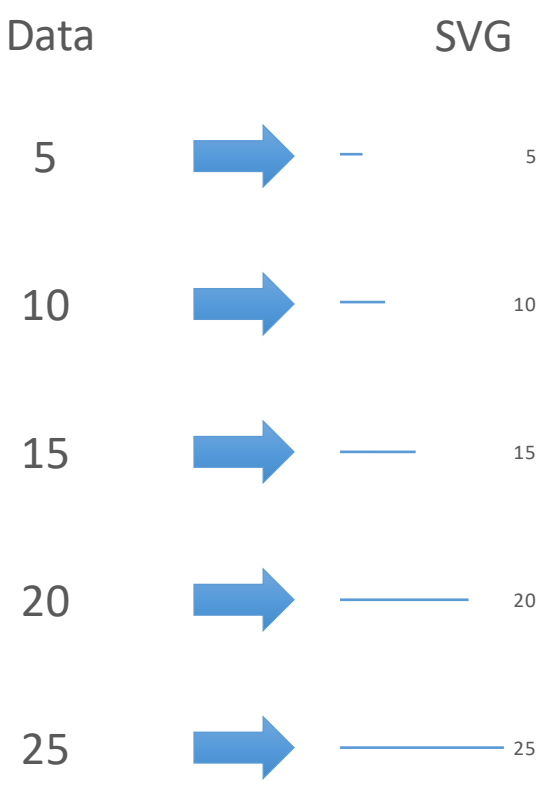
# SELECTION SHOULD CORRESPOND TO DATA

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



# SELECTION SHOULD CORRESPOND TO DATA

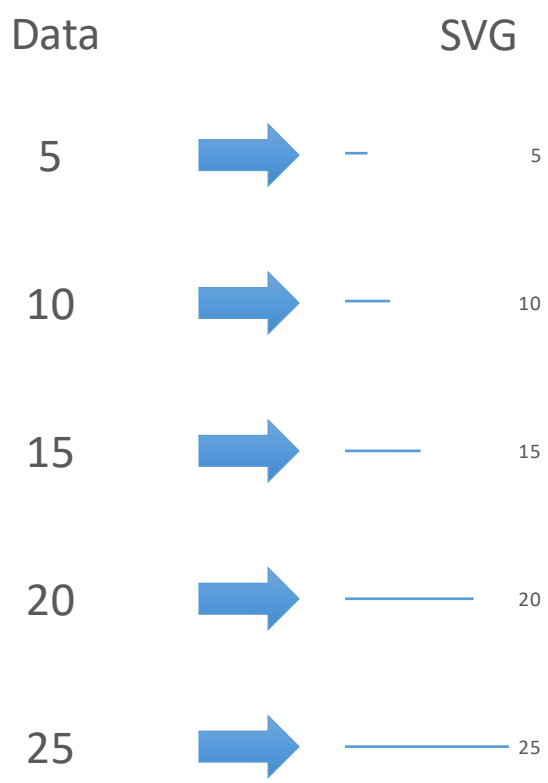
```
let numbers =
[5,10,15,20,25];
let lines =
svg.selectAll(”line")
      .data(numbers)
      .enter().append(”line
");

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));

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

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



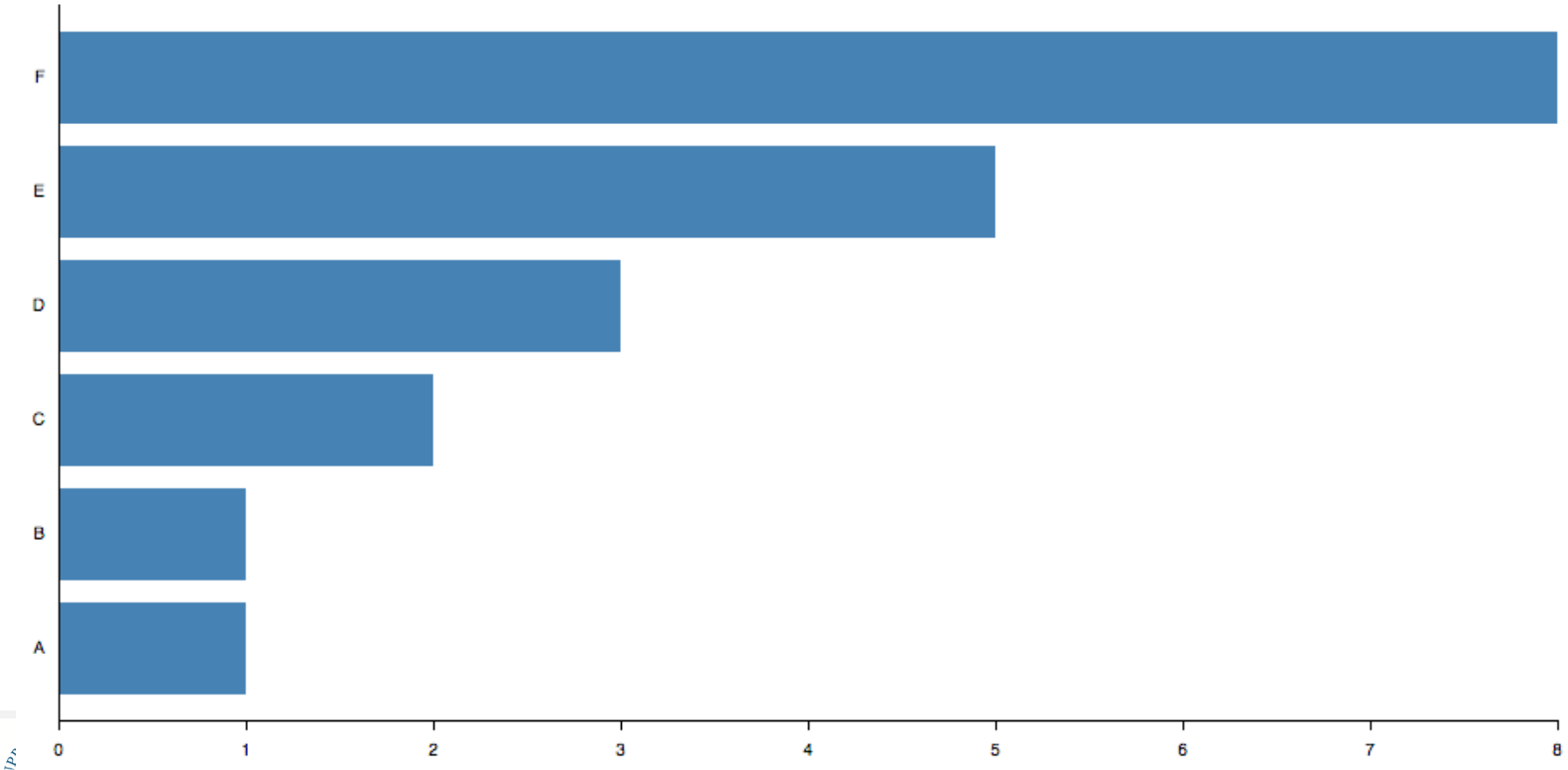

The `attr` functions takes in input a constant value or a function. The function is called automatically by d3, passing the data (`__data__`) bound to the element and a progressive counter

# 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

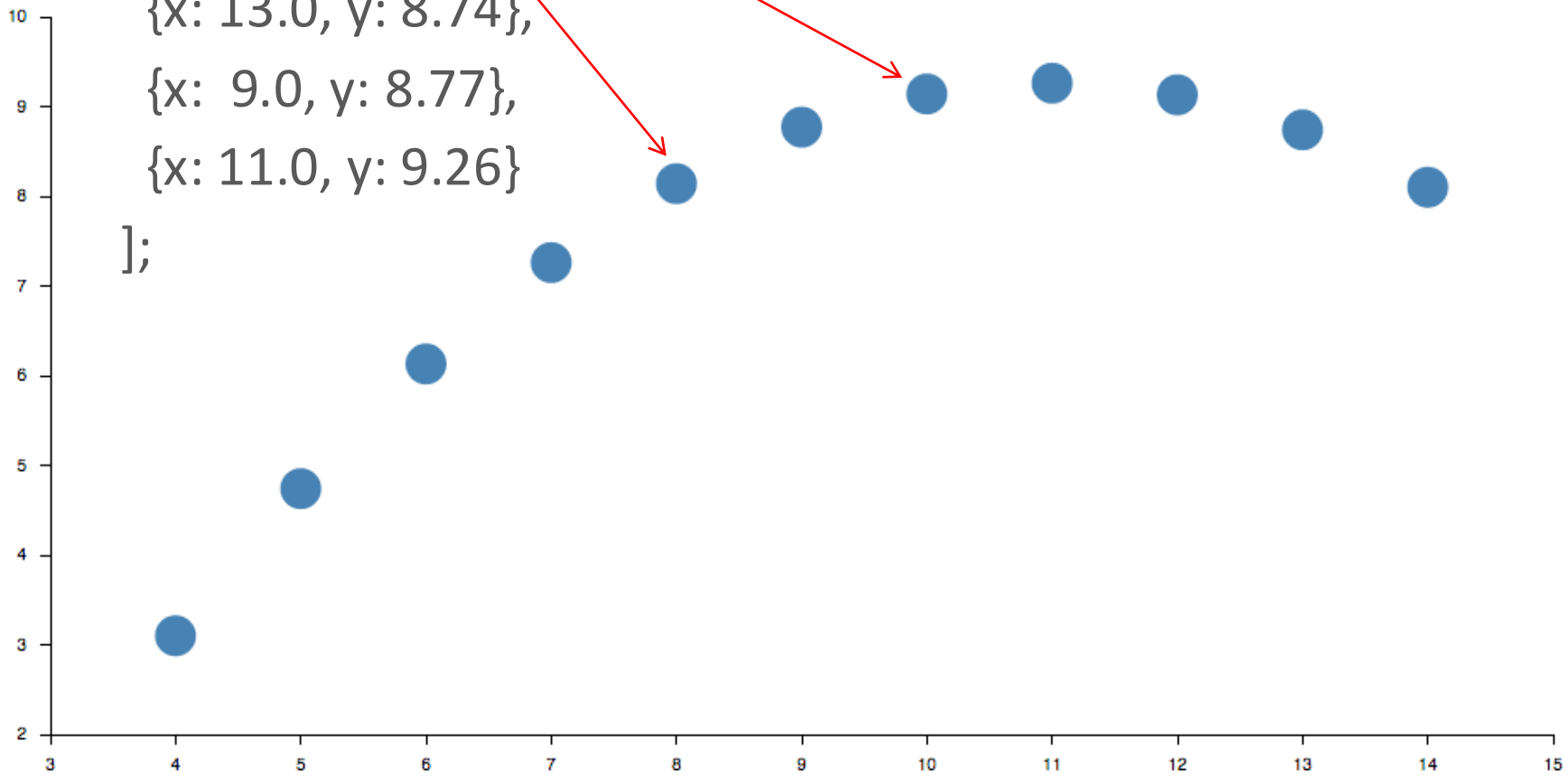
# DATA CAN BE NUMBERS

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



# DATA CAN BE OBJECTS.

```
let 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}
];
```

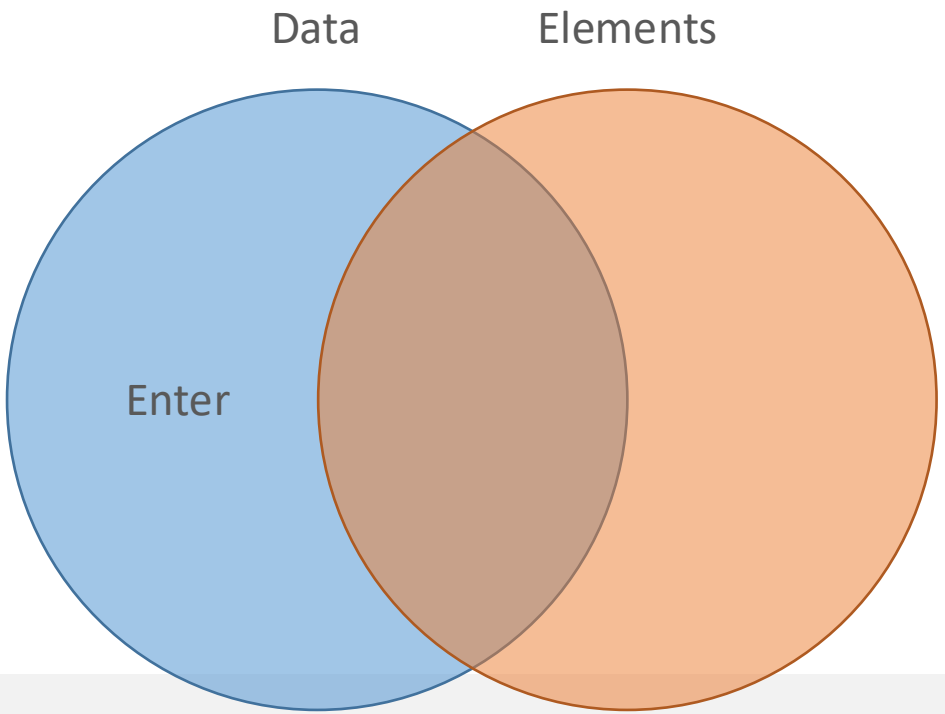


# ENTER, EXIT, AND UPDATE

Thinking with Joins

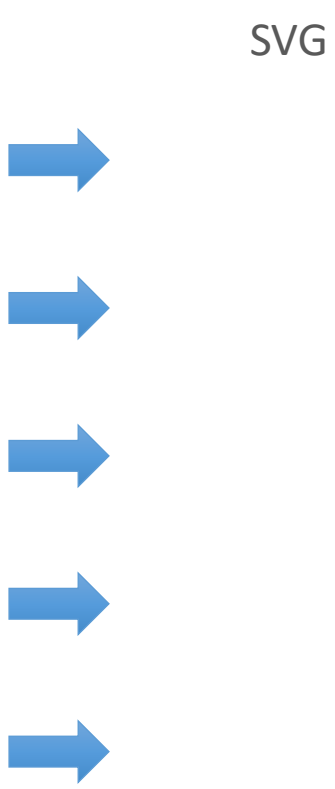
# ENTER

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



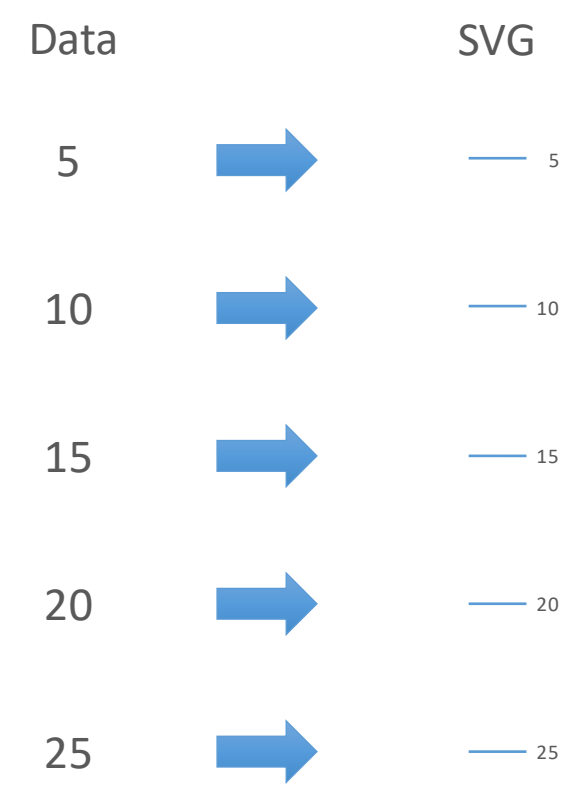
# ENTERING NEW ELEMENTS

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


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



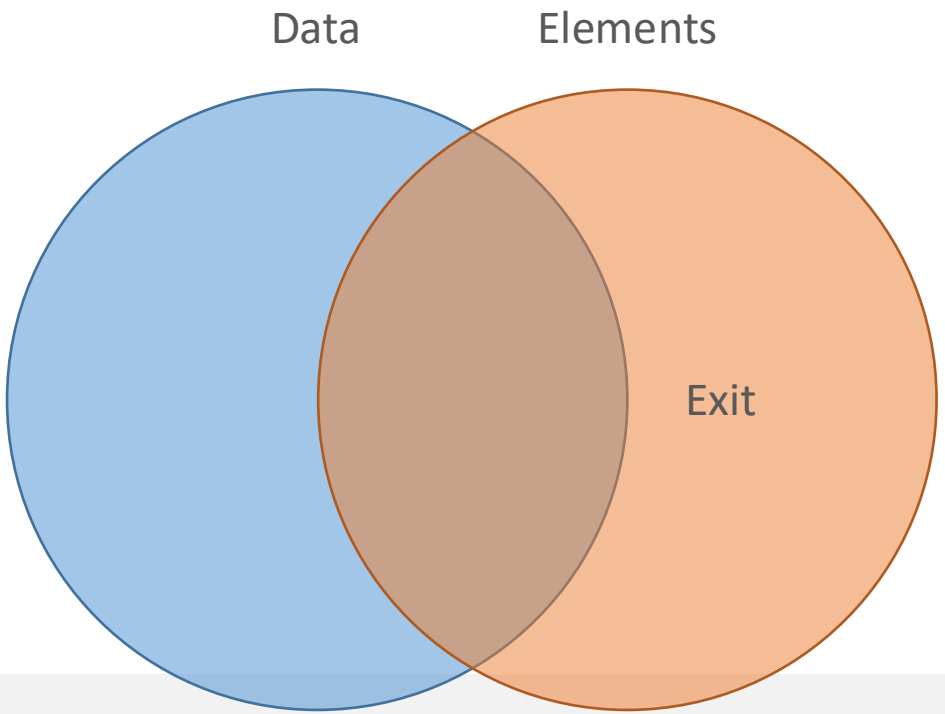
# ENTERING NEW ELEMENTS

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


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



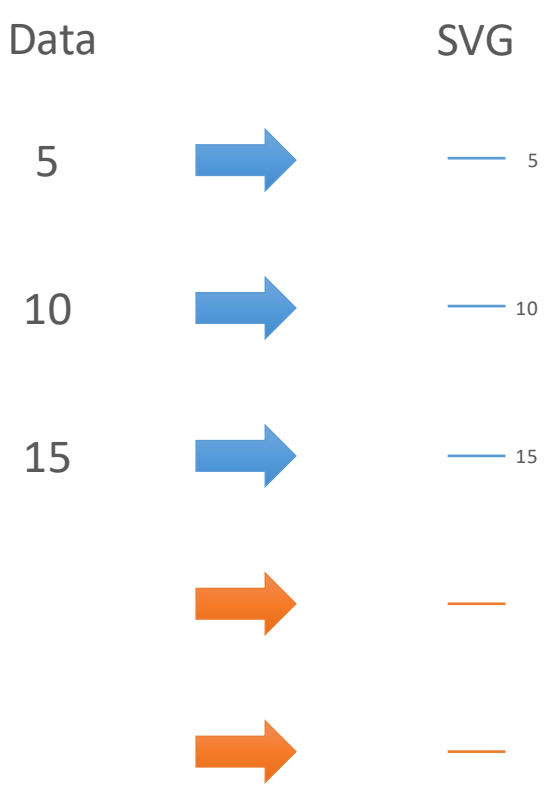
# EXIT

- Elements that are associated with no data



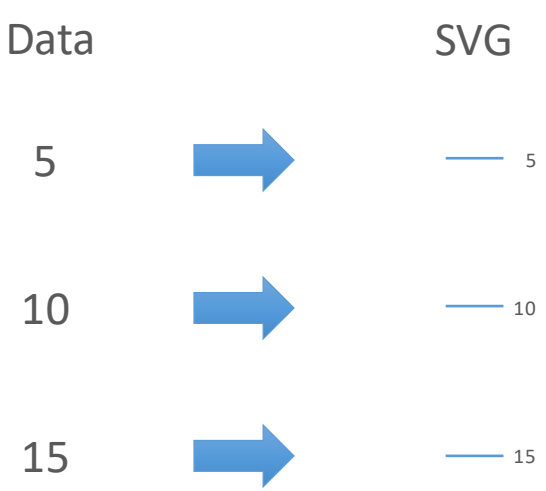
# EXITING UNNECESSARY ELEMENTS

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


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



# ENTERING NEW ELEMENTS

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


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

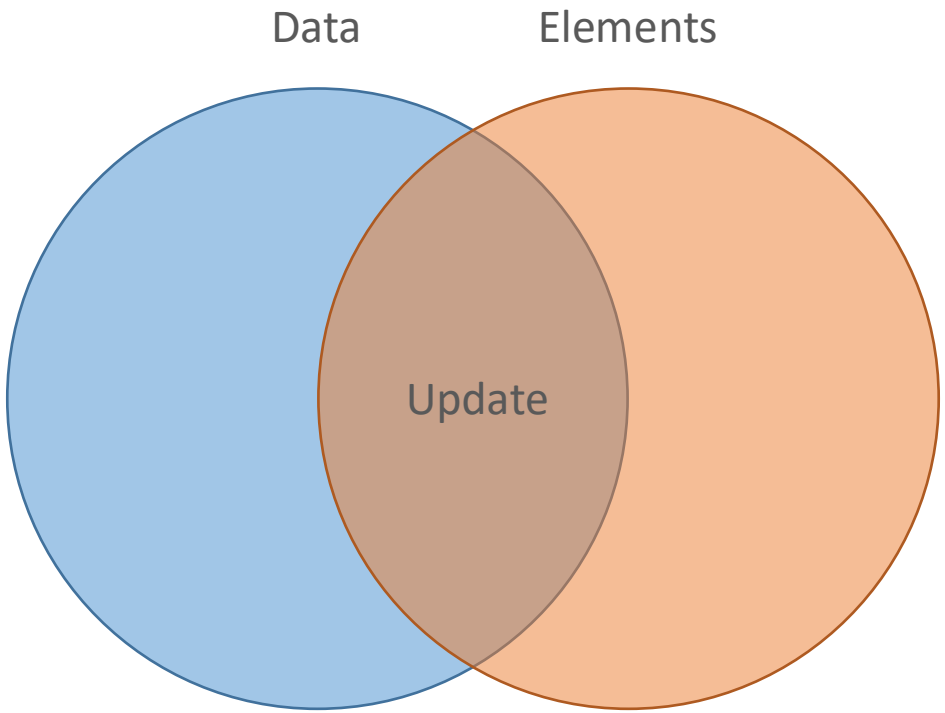


# DATA ATTRIBUTES TO ELEMENTS ATTRIBUTES

Step 2

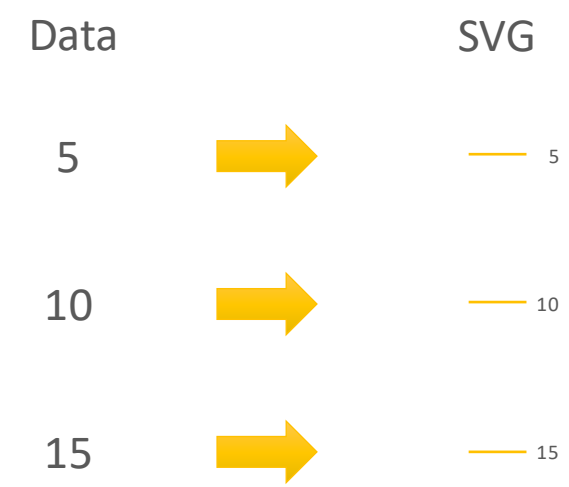
# UPDATE

- Data already joined with previous elements



# UPDATE EXISTING AND NEW ELEMENTS WITH NEW DATA

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

```
let 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

- Tutorials
- http://bost.ocks.org/mike/d3/workshop/
- https://www.oliviavane.co.uk/tutorials/d3/about/tutorial-about