



VISUALIZATION ON THE WEB



TABLEAU.COM



The advertisement features a woman with long blonde hair, seen from behind, drawing a bar chart on a whiteboard. The whiteboard has a vertical axis labeled 1-8 and a horizontal axis labeled 4, 5, 8, 9, 10, 11, with an arrow pointing to the word 'Days'. The Tableau logo is in the top left, and the text 'DATA ANALYSIS SOFTWARE' is centered. Below it is an orange button that says 'START YOUR FREE TRIAL' and a line of text: 'Full-version trial. No credit card required.' The bottom of the ad has a blue geometric pattern.

 **tableau**

DATA ANALYSIS SOFTWARE

[START YOUR FREE TRIAL](#)

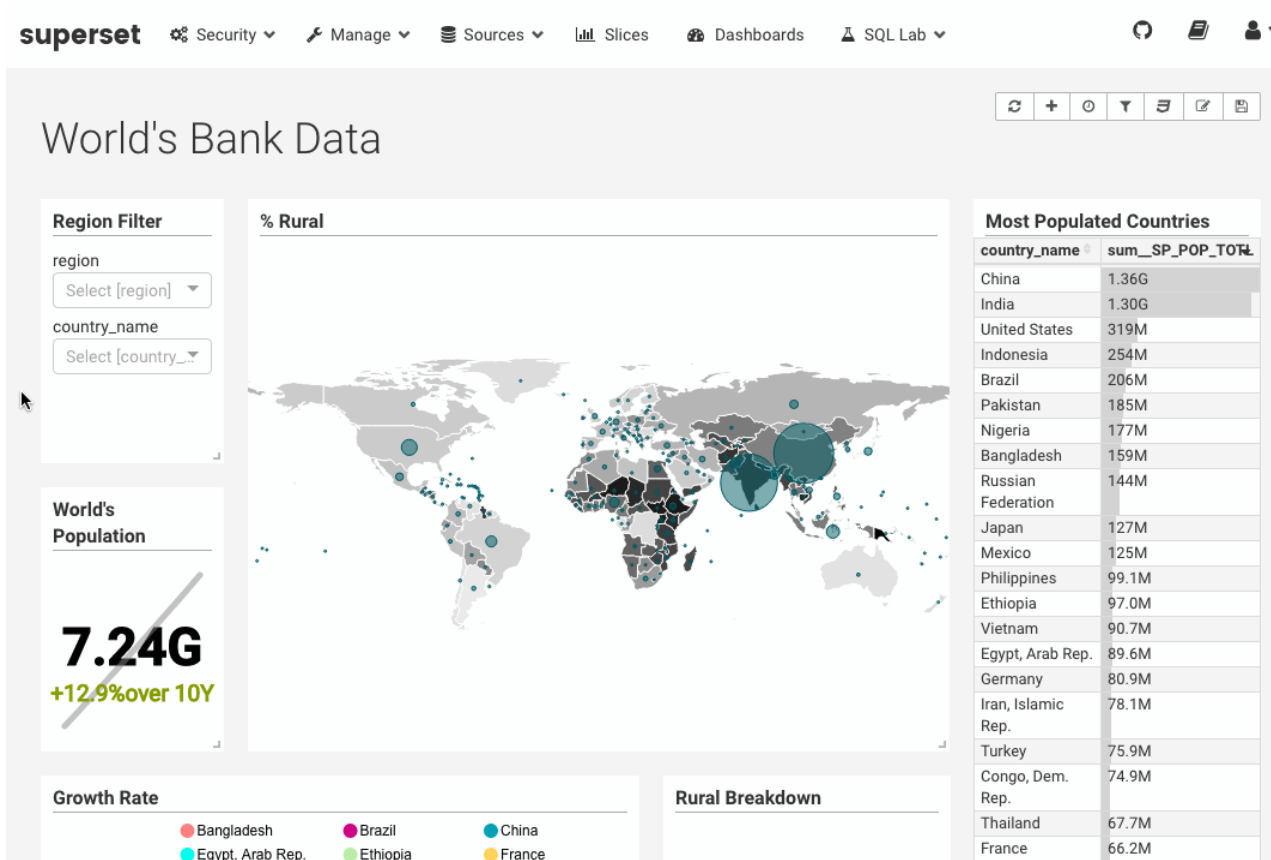
Full-version trial. No credit card required.

KIBANA GA



The screenshot displays the Kibi web application interface. At the top, there are navigation tabs: Discover, Visualize, Dashboard, and Settings. Below these are filters for Molecules (13520737), Assays (1148941), Targets (10775), and Papers (59610). The main content area is a table of search results for molecules, with columns for molecule type, availability type, synonyms, and chirality. The table lists several molecules, including (2S,4S,5R,6R)-5-acetamido-6-((1R,2R)-3-oxo(methyl)mercury), (6R,9S,12S)-12-Methoxy-8-methyl-10,11,11-trimethyl-stannane, (-)-NEOMENTHOL, (-)-1,1-dichloro-[1,2-bis(4-hydroxyphenyl)ethane], (-)-RS-PARASORBIC ACID, (-)-11-DEMETHYL CALANOLIDE A, and (-)-11-DEMETHYL CORDATOLIDE A. To the left of the table is a sidebar with filters for Molecule type and Indication Class. To the right is a panel with 'Relational Button Activities' and 'Therapeutic vs Non (Chirality)' charts.

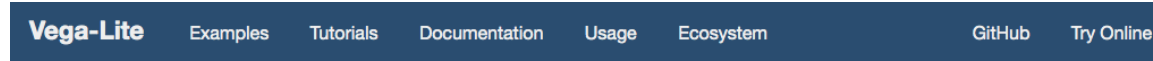
molecule_type	availability_type	synonyms	chirality
Small molecule	-1	-	-1
Small molecule	-1	-	-1
Small molecule	-1	-	-1
Small molecule	-1	(-)-Neomenthol	-1
Small molecule	-1	-	-1
Small molecule	-1	-	-1
Small molecule	-1	(-)-11-demethyl calanolide A	-1
Small molecule	-1	(-)-11-demethyl cordatolide A	-1



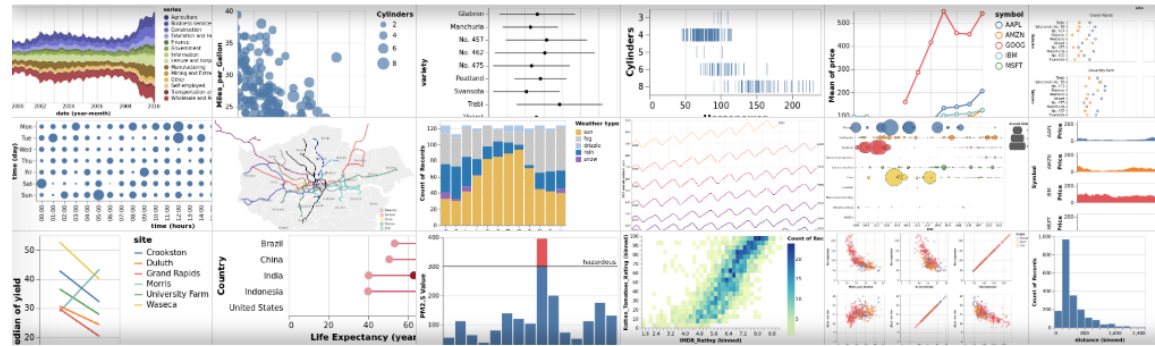
PLOT.LY

The screenshot shows the Plotly website homepage. At the top left is the Plotly logo. The navigation bar includes links for QSearch, Pricing, Industries, API, and Sign in, along with buttons for SIGN UP, UPGRADE, and REQUEST DEMO, and a language selector for français. The main heading is "Visualize Data, Together" with a subtext "I want to make a...". Below this are two buttons: "New chart" and "Dashboard". A section titled "Compatible with a variety of tools" includes the text "Get started with the tools you already like and use" and icons for Python, R, MATLAB, Excel, Javascript, and Web App.

VEGA AND VEGA-LITE



Vega-Lite – A Grammar of Interactive Graphics



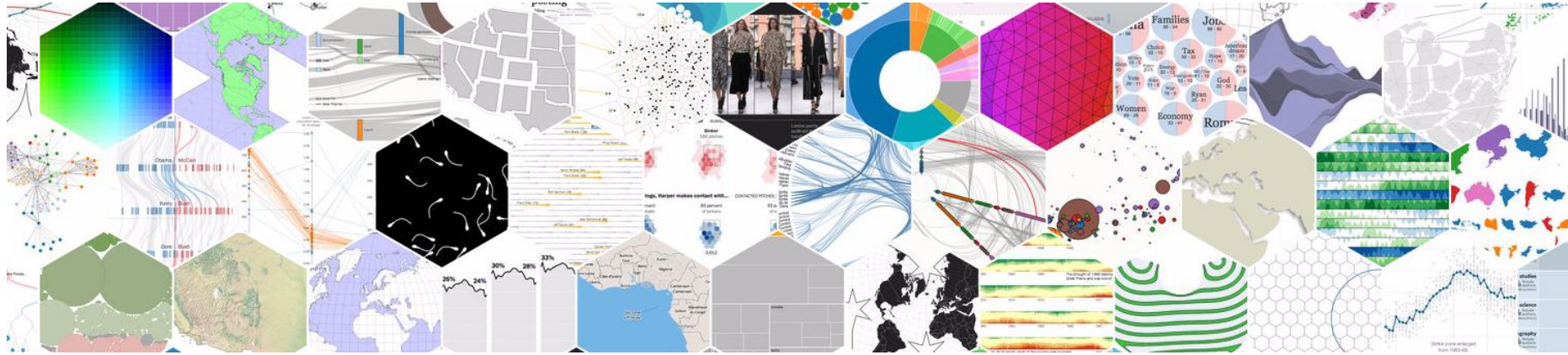
Vega-Lite is a high-level grammar of interactive graphics. It provides a concise JSON syntax for rapidly generating visualizations to support analysis. Vega-Lite specifications can be compiled to Vega specifications.

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

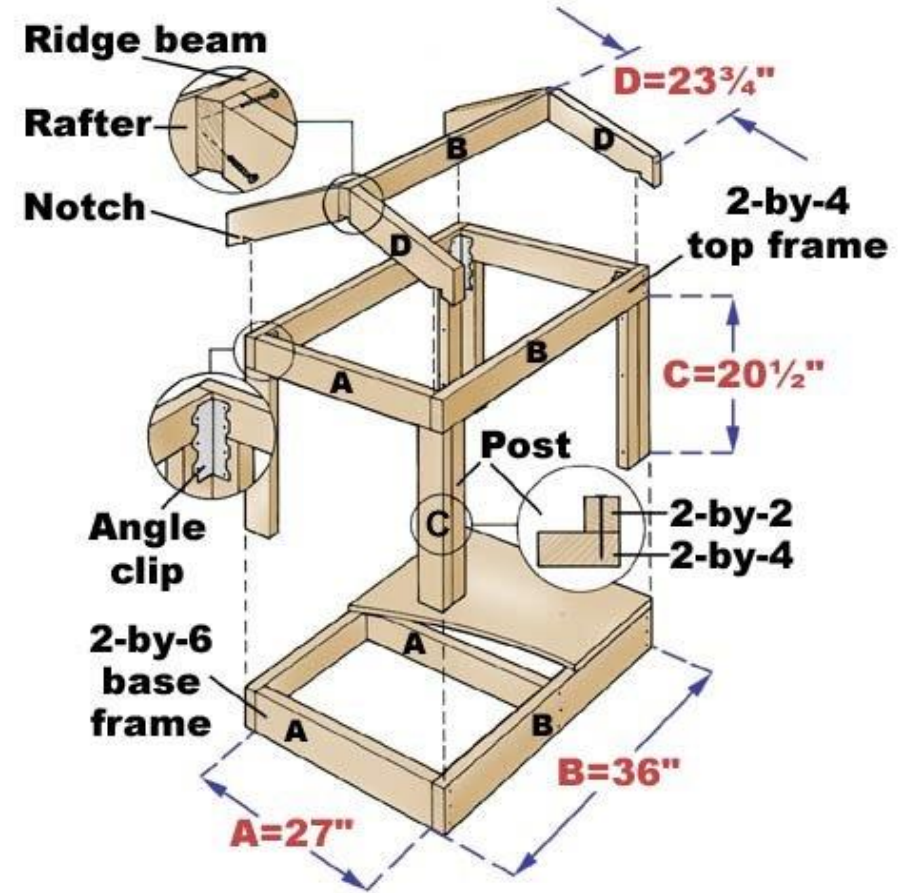
Data-Driven Documents



VISUAL ANALYTICS

D3.JS

WHAT IS D3?



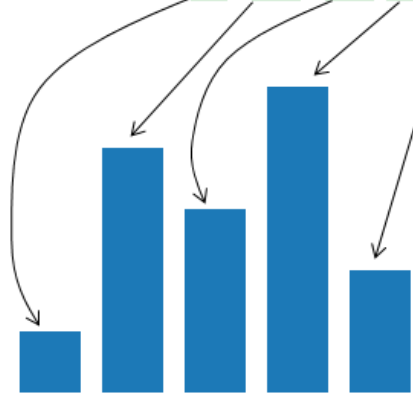
WHAT IS D3?



WHAT IS D3?

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

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

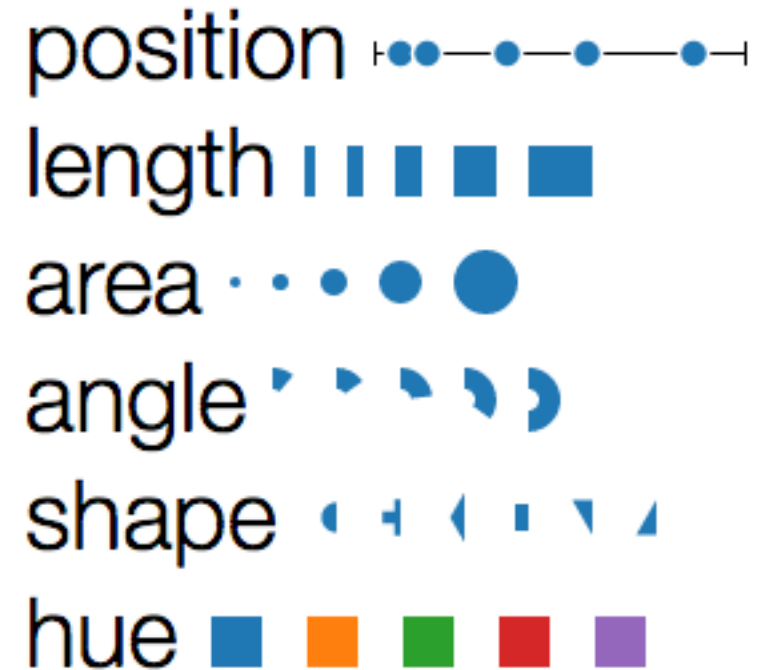


VISUALIZATION AND DATA GRAPHICS

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  
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")  
    .text("");
```

Data

SVG

SELECTION SHOULD CORRESPOND TO DATA

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

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

Data

SVG

5

10

15

20

25

SELECTION SHOULD CORRESPOND TO DATA

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

Data

SVG

5



10



15



20



25



SELECTION SHOULD CORRESPOND TO DATA

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

Data

5



10



15



20



25



SVG

— 5

— 10

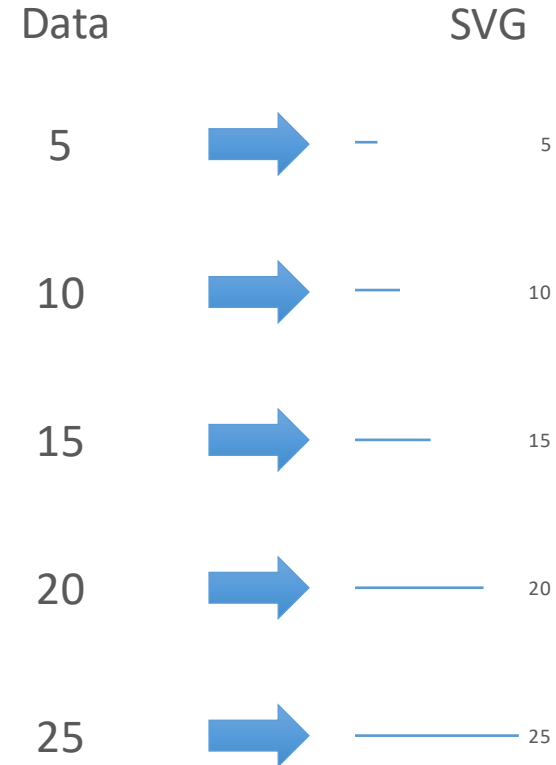
— 15

— 20

— 25

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))
```

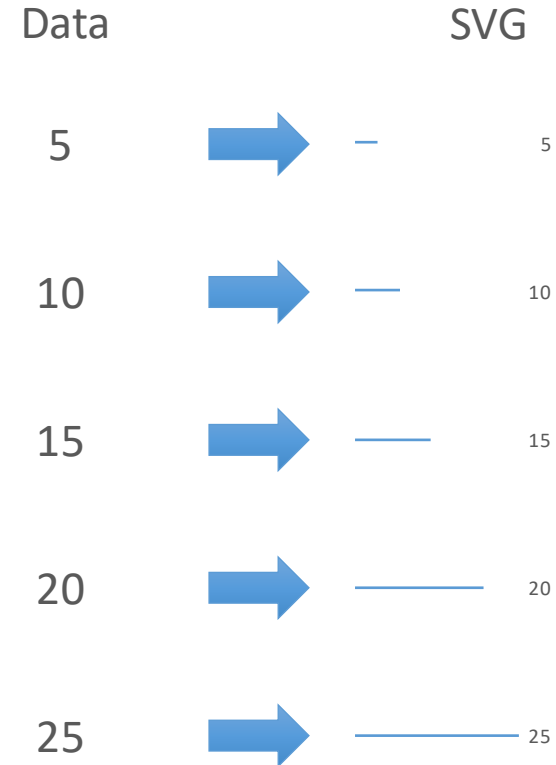


SELECTION SHOULD CORRESPOND TO DATA

```
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;
}
```



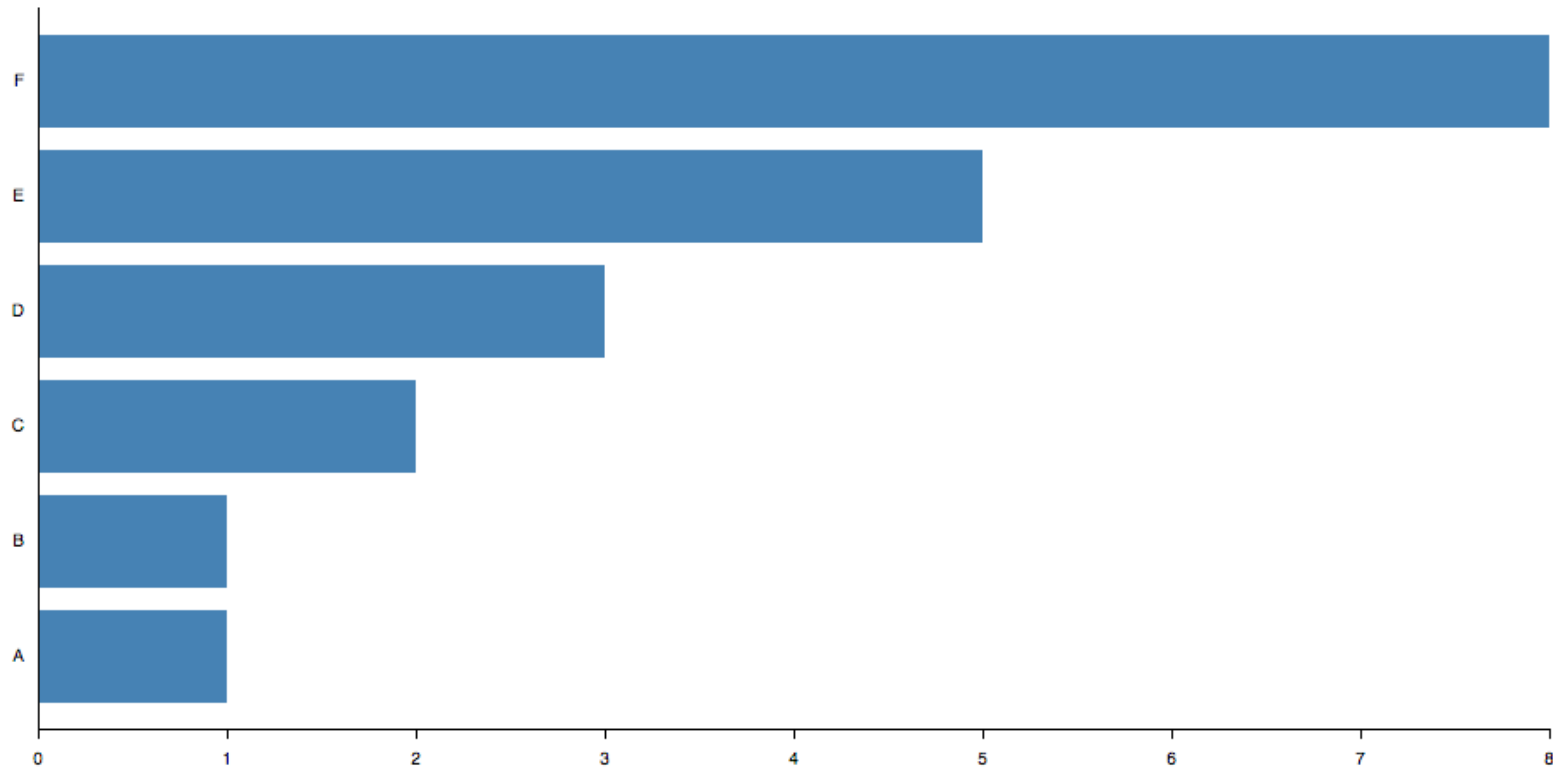
The attr functions takes in input a constant value or a function. The function is called automatically by d3, passing the data (`__data__`) bound to the element and a progressive counter

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

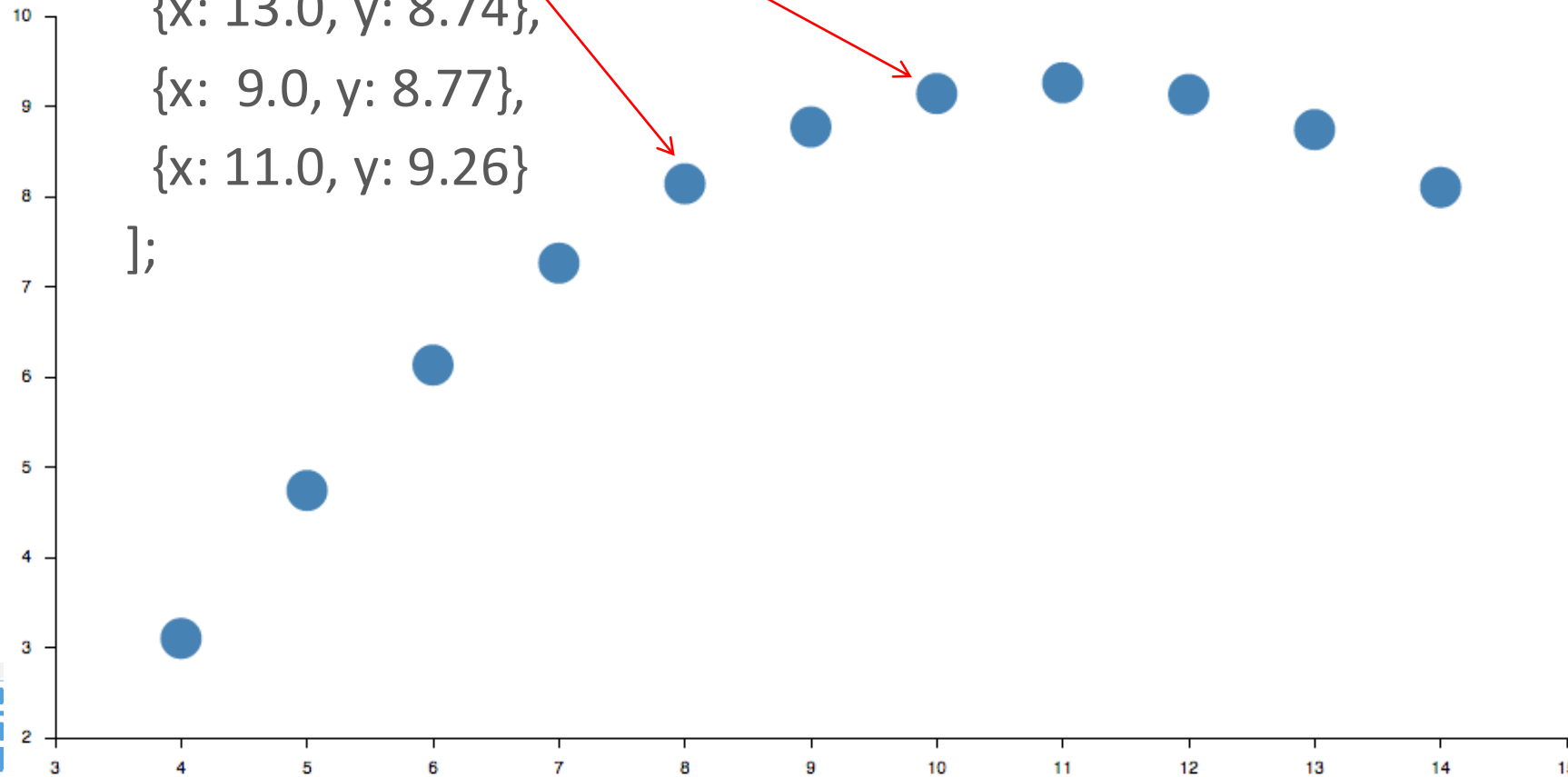
DATA CAN BE NUMBERS

```
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}  
];
```



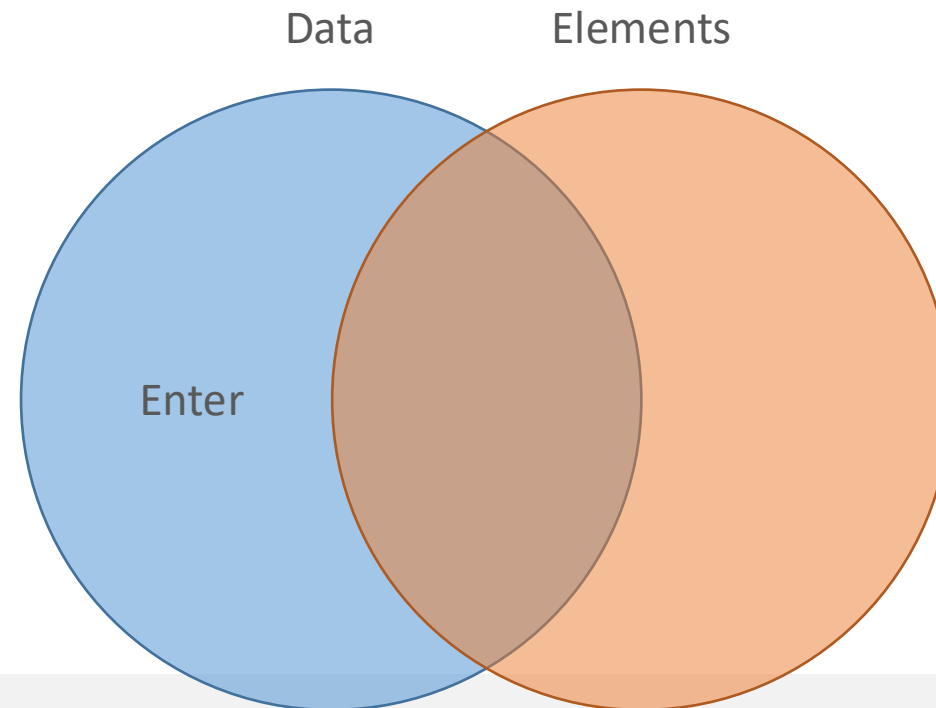
ENTER, EXIT, AND UPDATE

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");
```

Data

5



10



15



20



25



SVG

ENTERING NEW ELEMENTS

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

Data

5



10



15



20



25



SVG

— 5

— 10

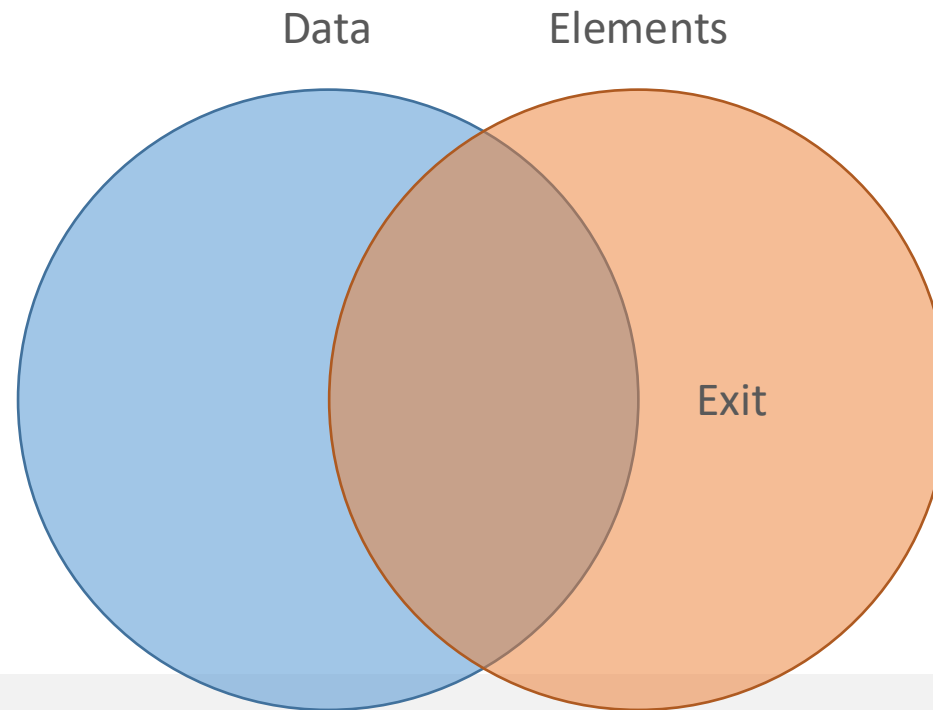
— 15

— 20

— 25

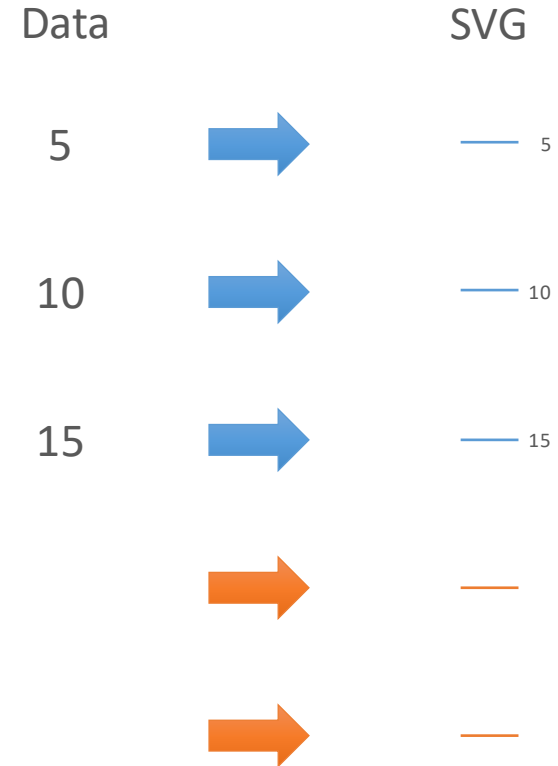
EXIT

- Elements that are associated with no data



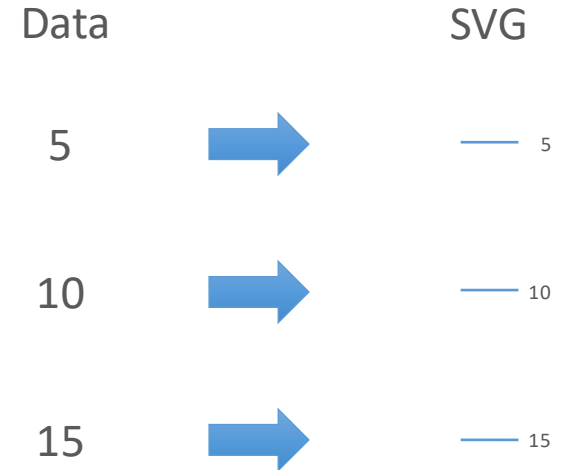
EXITING UNNECESSARY ELEMENTS

```
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();
```

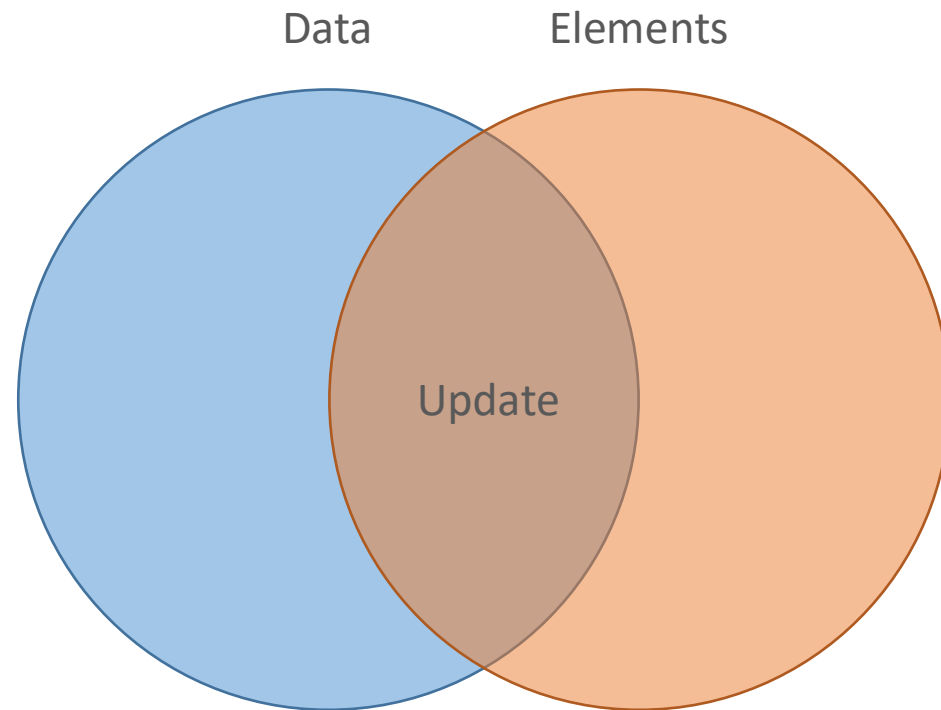


DATA ATTRIBUTES TO ELEMENTS ATTRIBUTES

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));
```

Data

5



10



15



SVG

— 5

— 10

— 15

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>