The D3 Library

Data-**D**riven **D**ocuments

D3 Is a JavaScript Library

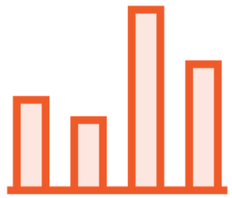
Uses existing language

JavaScript is the most widely used language

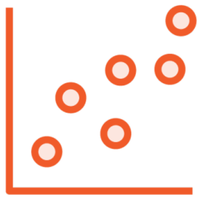
Uses web standards

Best practices, ES6, CSS, browser based technologies

D3 Use Cases



Bar Chart



Scatterplot

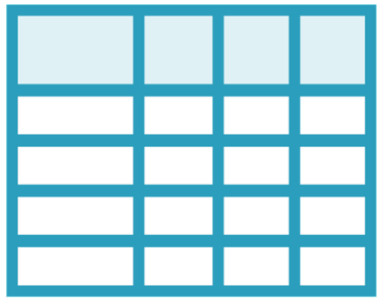
Pie Charts

Network Diagrams

Maps

Creative/ Experimental

Maps Data to DOM Elements in Browser



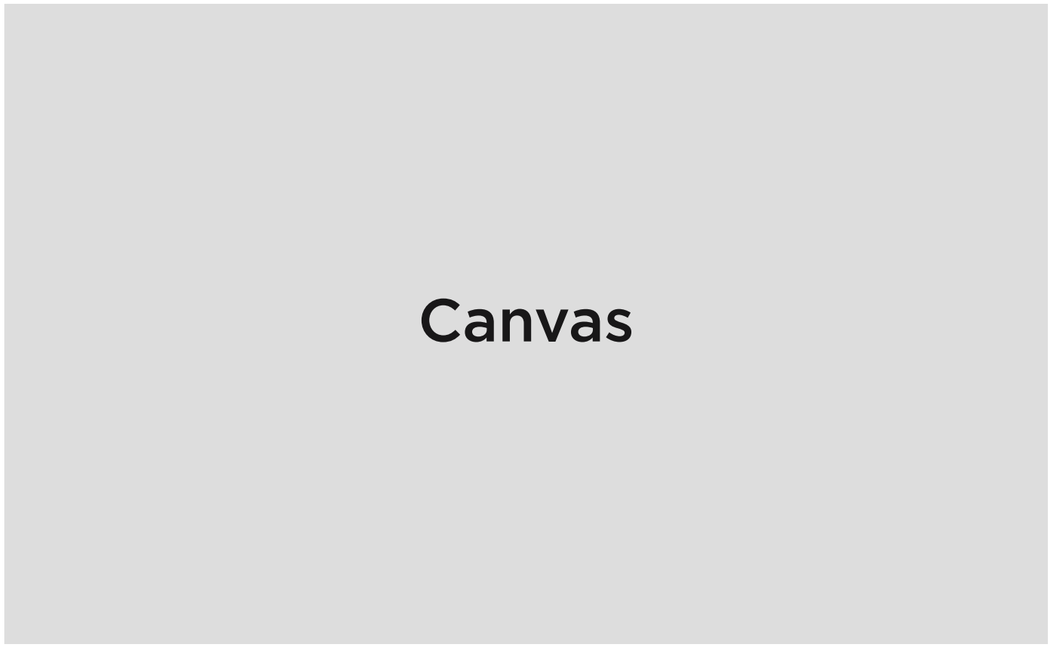
Data

CSV, TSV, JSON, XML, etc

DOM Elements

SVGs, Canvas, DIVs, etc

Common D3 Output



Scalable Vector Graphics (SVG)

Rectangles

```
<svg width="400" height="110">
   <rect width="300" height="100"/>
</svg>
```

Circles

```
<svg width="400" height="110">
   <circle cx="50" cy="50" r="40"/>
</svg>
```

Lines

```
<svg width="400" height="110">
   <line x1="0" y1="0" x2="200" y2="200"/>
</svg>
```

Paths

```
<svg width="400" height="110">
   <path d="M150 0 L75 200 L225 200 Z"/>
</svg>
```

Project Setup

Starter Files

index.html

sales.csv

Local Server Configuration

Node.js

NPM

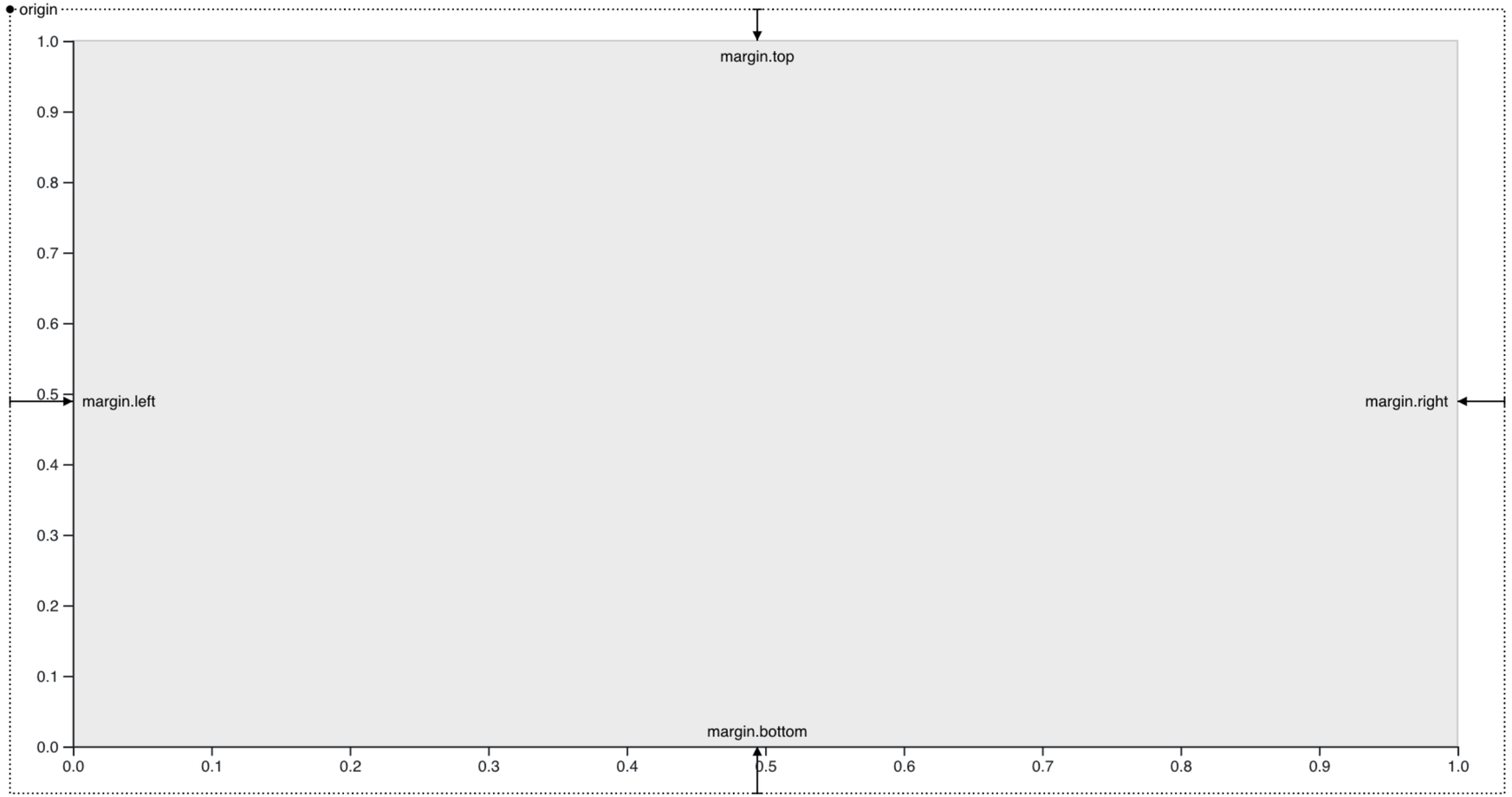
Node-static

Add D3 to the Page

Preparing the SVG

Understanding Margin Conventions in D3

Margin Conventions



Fetching Data

Fetch

Refers to requests, responses, and binding processes of data and other files.

Scales

Scales

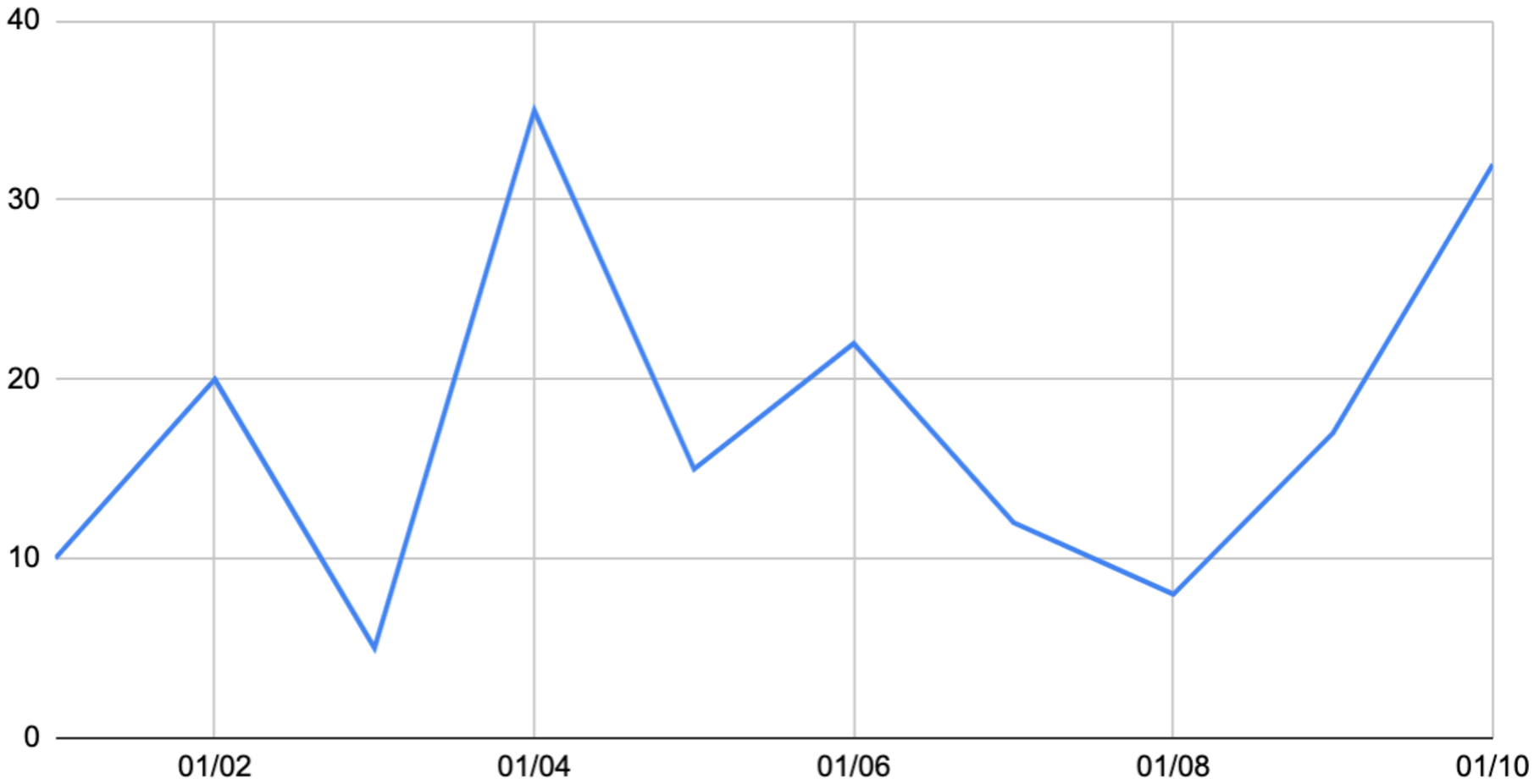
“mapping a dimension of abstract data to a visual representation”

D3 Scales Generators

Linear scale

`.scaleLinear()`

Linear Scale



D3 Scales Generators

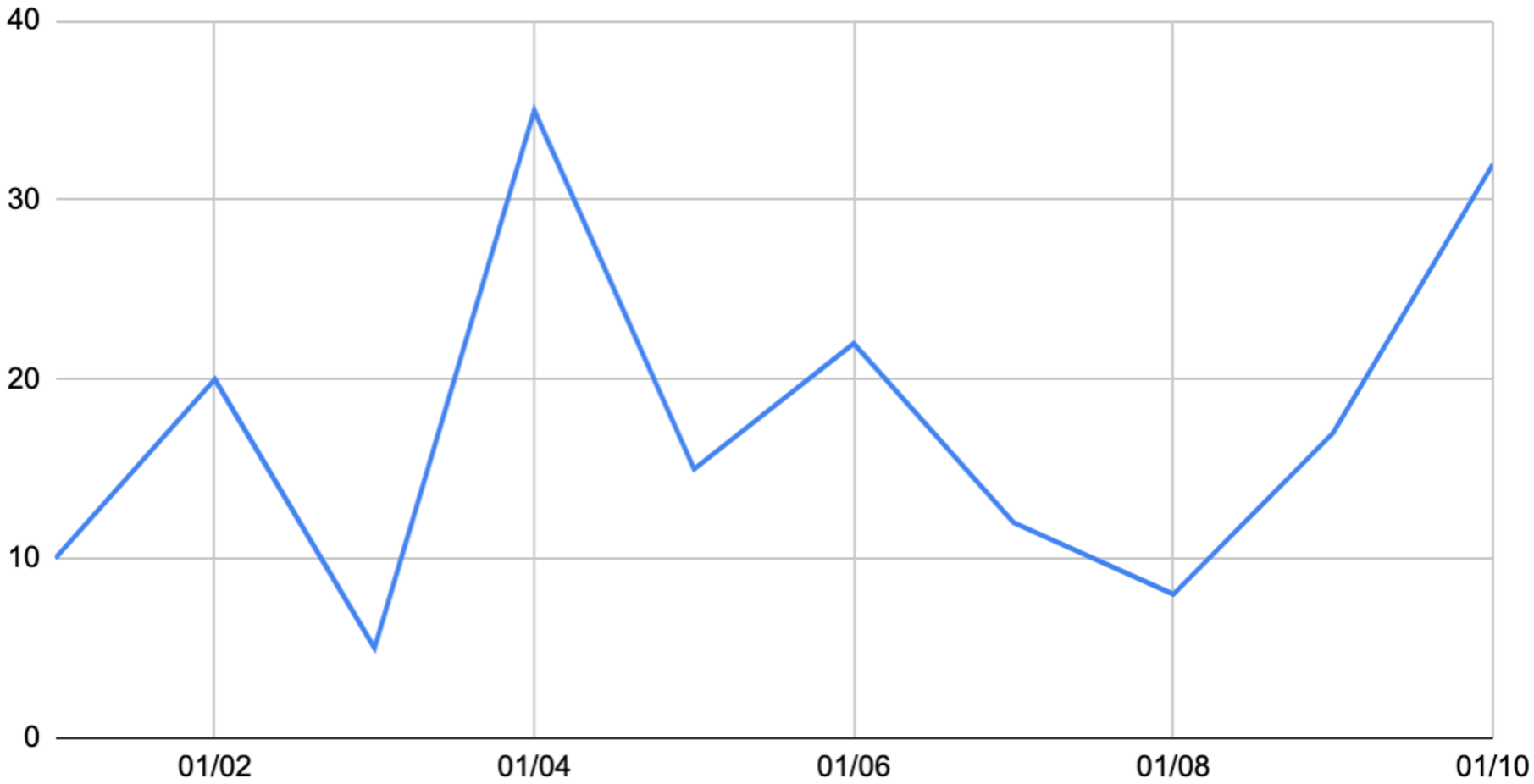
Linear scale

`.scaleLinear()`

Time scale

`.scaleTime()`

Time Scale



D3 Scales Generators

Linear scale

`.scaleLinear()`

Time scale

`.scaleTime()`

Power/Sqrt/Log scale

`.scalePow()`

`.scaleSqrt()`

`.scaleLog()`

.scaleSqrt()

D3 Scales Generators

Linear scale

`.scaleLinear()`

Time scale

`.scaleTime()`

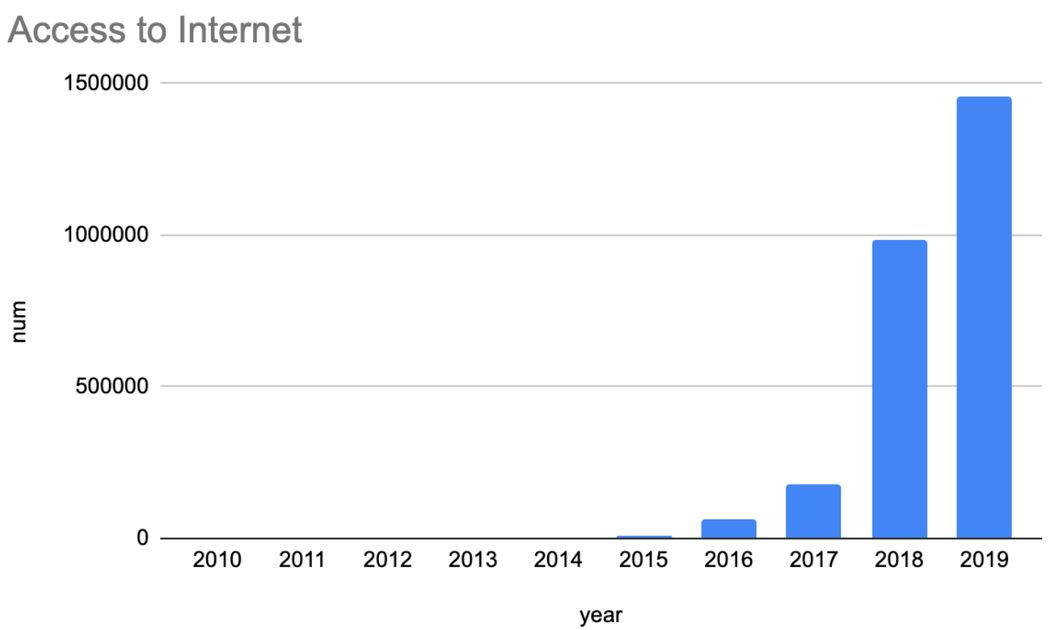
Power/Sqrt/Log scale

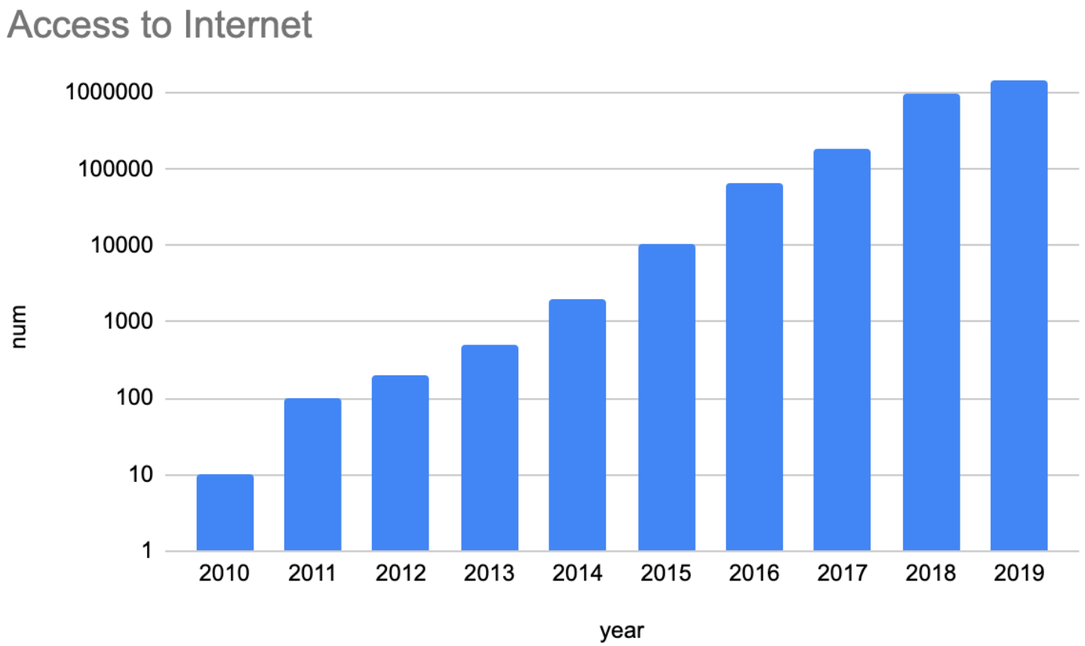
`.scalePow()`

`.scaleSqrt()`

`.scaleLog()`

Linear Scale vs Log Scale





D3 Scales Generators

Linear scale

`.scaleLinear()`

Time scale

`.scaleTime()`

Power/Sqrt/Log scale

`.scalePow()`

`.scaleSqrt()`

`.scaleLog()`

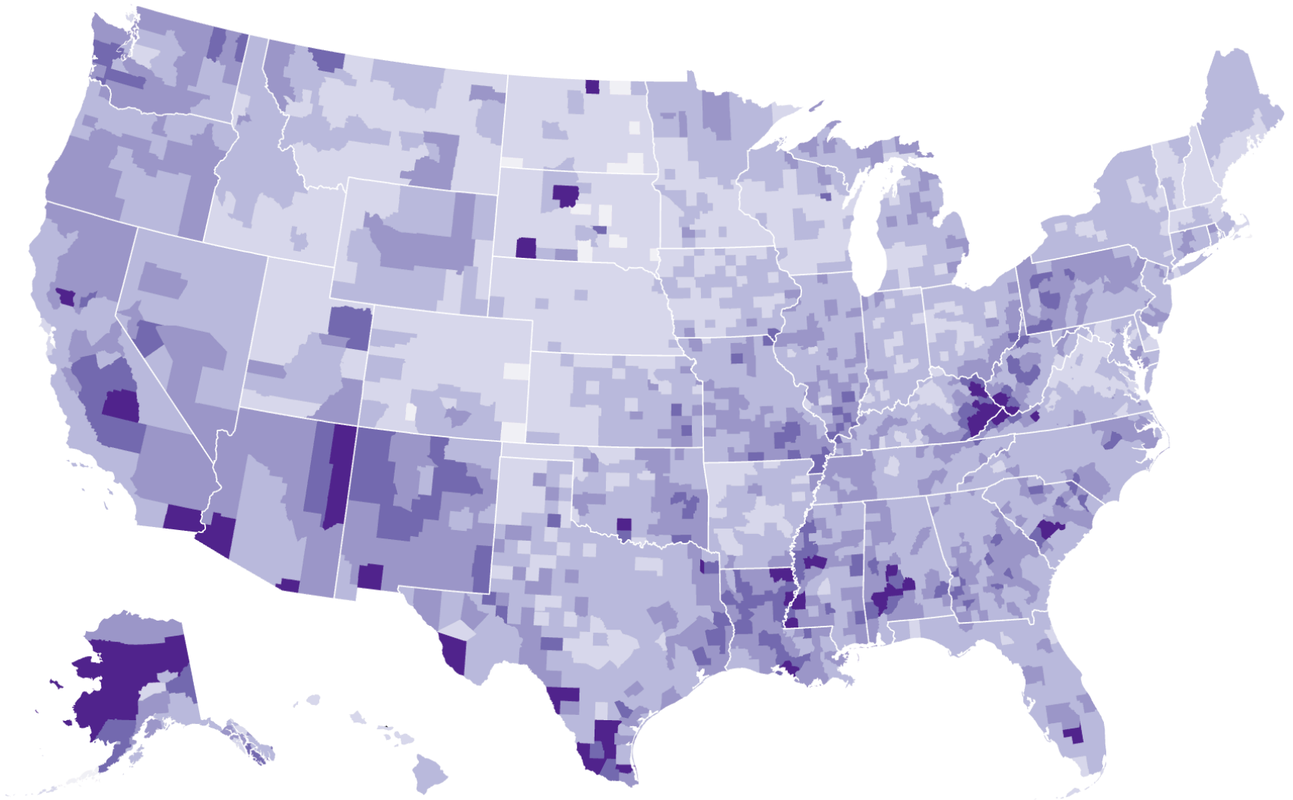
Quantize/Quantile/ Threshold scale

`.scaleQuantize()`

`.scaleQuantile()`

`.scaleThreshold()`

Threshold



D3 Scales Generators

Linear scale

`.scaleLinear()`

Time scale

`.scaleTime()`

Power/Sqrt/Log scale

`.scalePow()`

`.scaleSqrt()`

`.scaleLog()`

Quantize/Quantile/ Threshold scale

`.scaleQuantize()`

`.scaleQuantile()`

`.scaleThreshold()`

Sequential/Diverging scale

`.scaleSequential()`

`.scaleDiverging()`

Sequential

```
d3.scaleSequential(d3.interpolateRainbow)
```

D3 Scales Generators

Linear scale
`.scaleLinear()`

Time scale
`.scaleTime()`

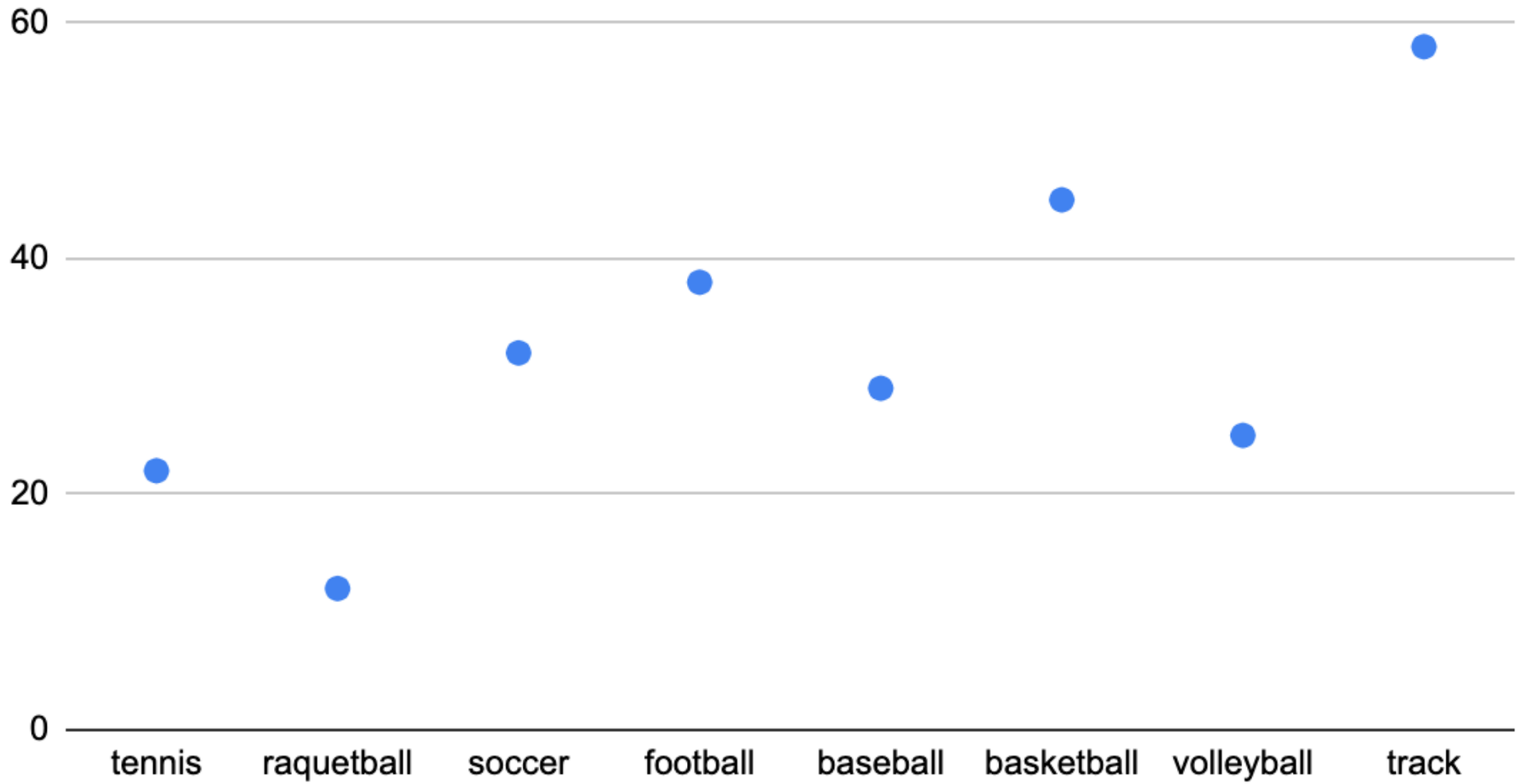
Power/Sqrt/Log scale
`.scalePow()`
`.scaleSqrt()`
`.scaleLog()`

Quantize/Quantile/ Threshold scale
`.scaleQuantize()`
`.scaleQuantile()`
`.scaleThreshold()`

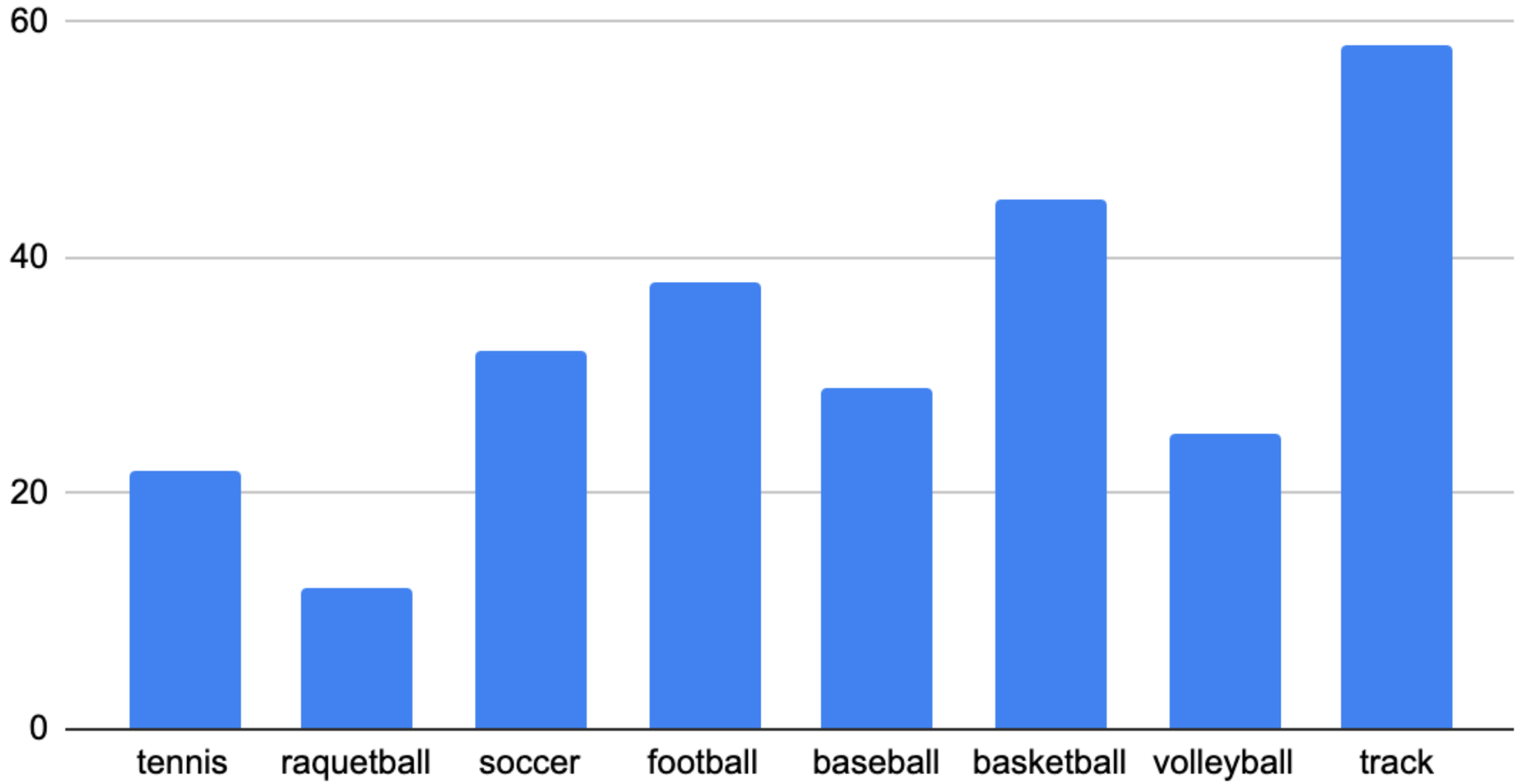
Sequential/Diverging scale
`.scaleSequential()`
`.scaleDiverging()`

Ordinal/Band scale
`.scaleOrdinal()`
`.scaleBand()`

Ordinal

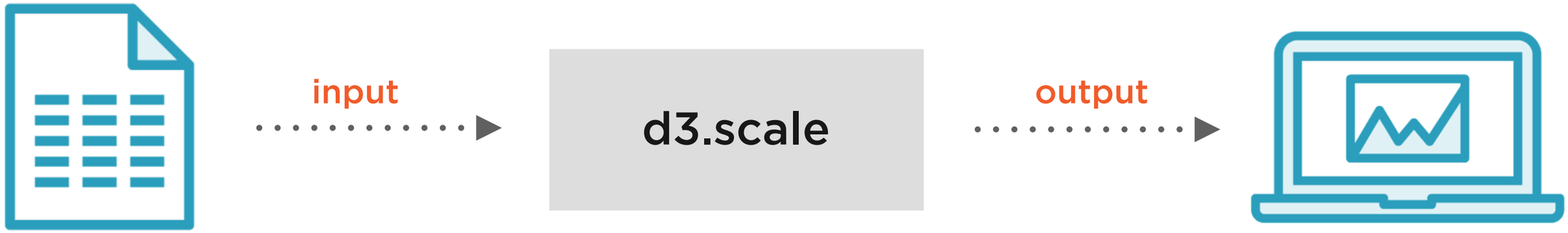

60
40
20
0
tennis
raquetball
soccer
football
baseball
basketball
volleyball
track

Band Scale


60
40
20
0
tennis
raquetball
soccer
football
baseball
basketball
volleyball
track

Domain and Range

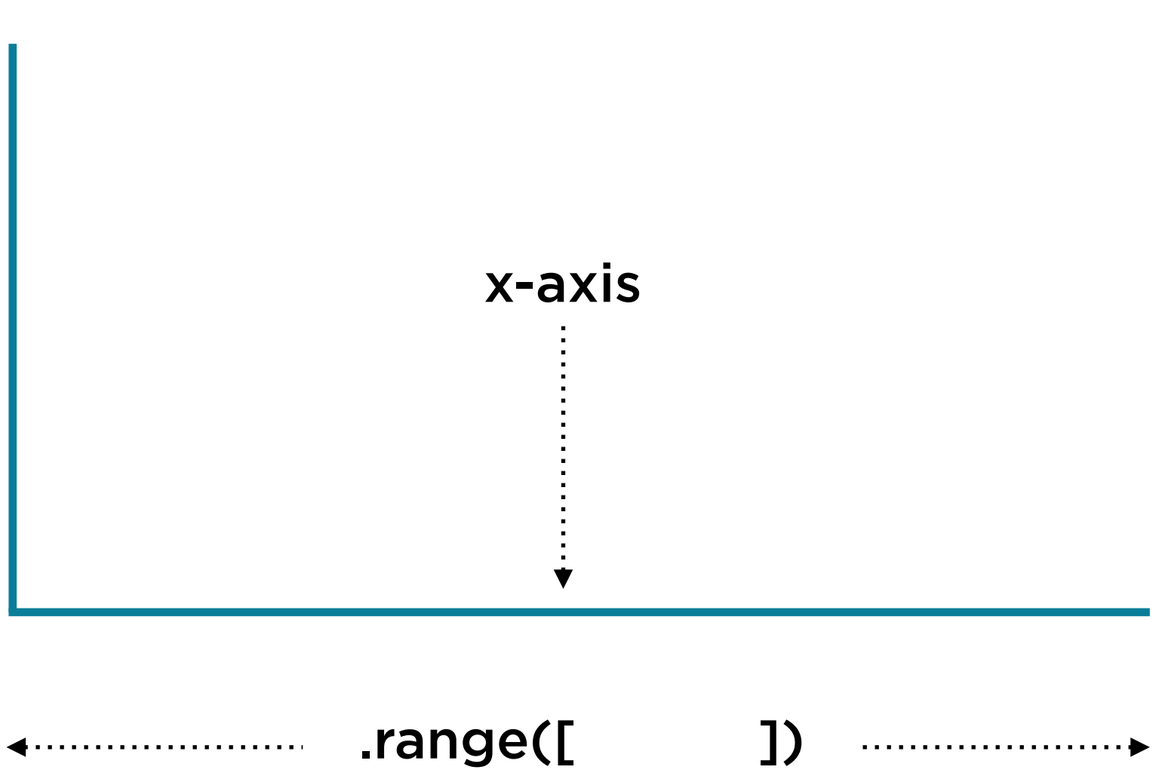
Domain and Range

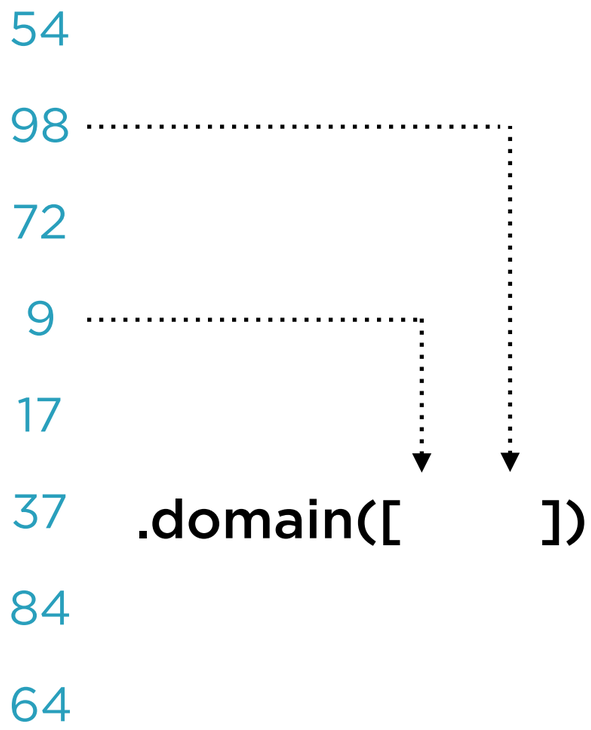


`.domain([])`

`.range([])`

Linear Scales





Domain and Range

Domain (Input)

data values

```
.domain([30, 49922])
```

Range (Output)

desired width of visualization

```
.range([0,width])
```

D3 Scales Generators

Linear scale
Continuous input
Continuous output

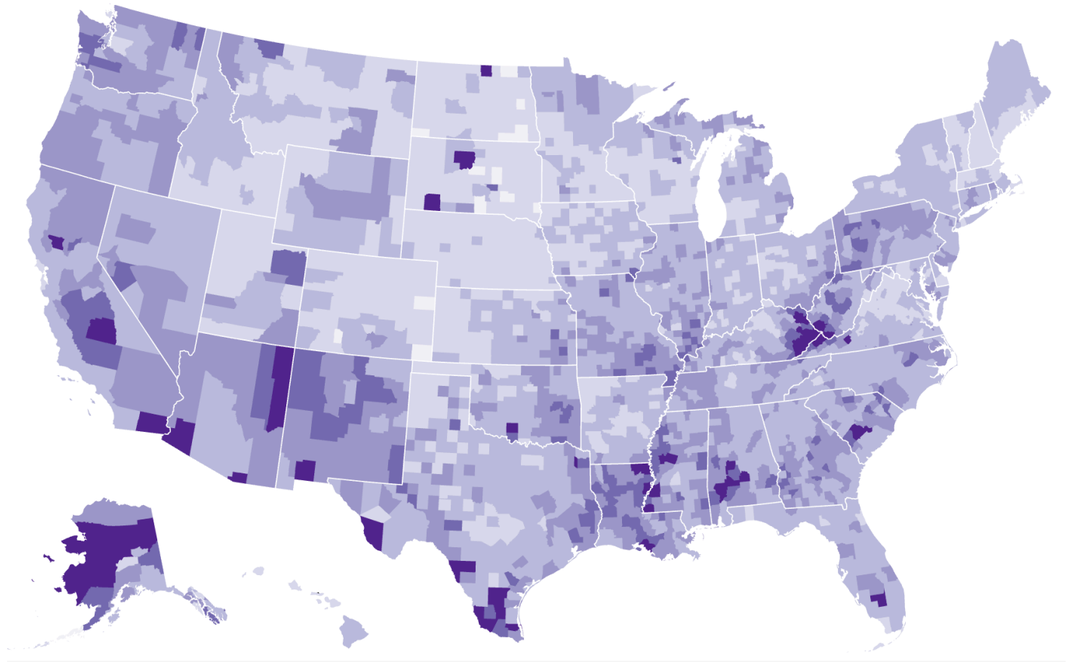
Time scale
Continuous input
Continuous output

Power/Sqrt/Log scale
Continuous input
Continuous output

Quantize/Quantile/ Threshold scale
Continuous input
Discrete output

Sequential/Diverging scale
Continuous input
Continuous output

Threshold Scale



.range([ ])

1.5K

58.7K

19.2K

1.2K

17.3K

7.6K

51.6K

12.4K

.domain([ ])

D3 Scales Generators

Linear scale
Continuous input
Continuous output

Time scale
Continuous input
Continuous output

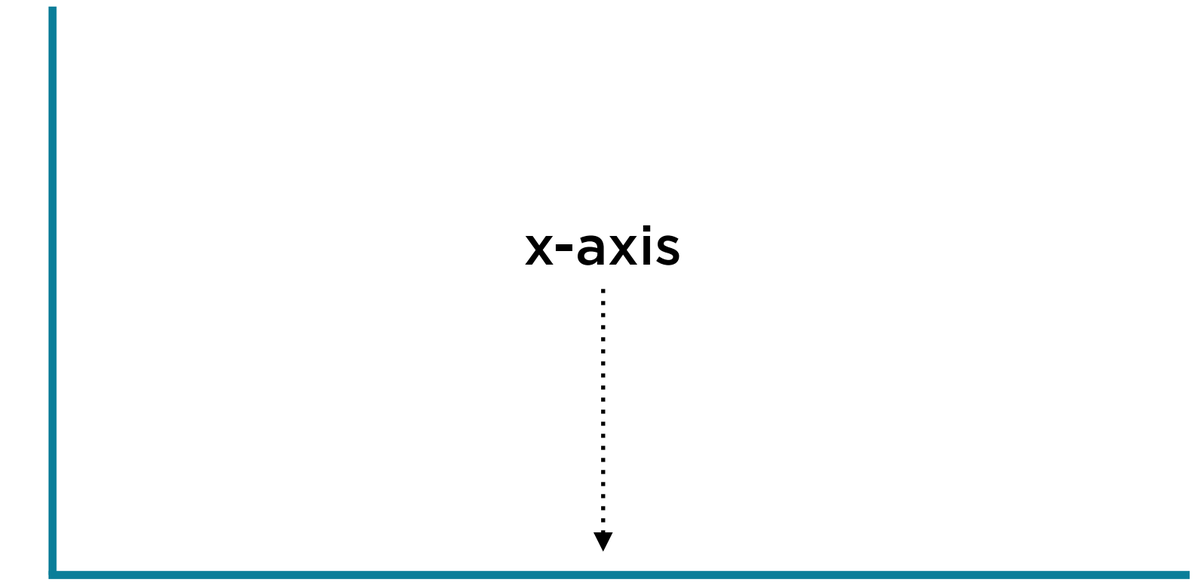
Power/Sqrt/Log scale
Continuous input
Continuous output

Quantize/Quantile/ Threshold scale
Continuous input
Discrete output

Sequential/Diverging scale
Continuous input
Continuous output

Ordinal/Band scale
Discrete input
Discrete output

Band Scale



tennis

soccer

football

track

baseball

.range([ ])

.domain([‘tennis’, ‘soccer’, ...])

Domain and Range

Domain([])

Complete set of values
ex: [0,492]
or
[‘tennis’,‘basketball’,‘raquetball’]

Range([])

Returns interval of data placement from start point
ex: [0,width]
or
[‘red’,’blue’]

Axes

Entering the Data

Data Joins

D3 Code

```
let dataArr =
   [‘Data 1’, ‘Data 2’, ‘Data 3’]

d3.select(‘body’)
   .selectAll(‘div’)
   .data(dataArr)
   .enter()
   .append(‘div’)
```

HTML

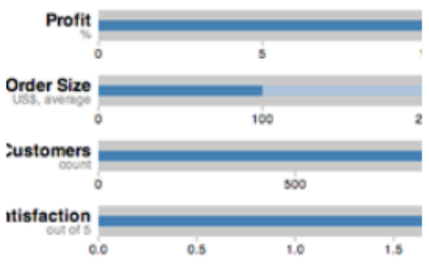
```
<body>
   <div>Data 1</div>
   <div>Data 2</div>
   <div>Data 3</div>
</body>
```


```
<body>
   <div>Data 1</div>
   <div>Data 2</div>
   <div>Data 3</div>
</body>
```

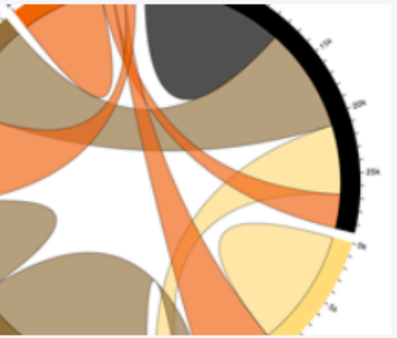
Drawing Rectangles with Data

Adding Labels

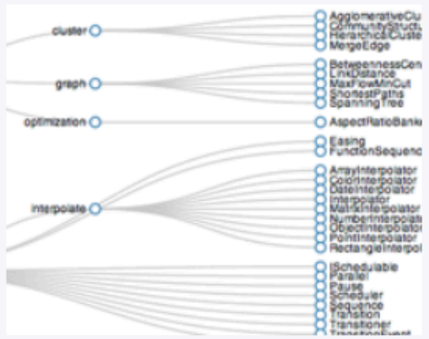
Using D3 Examples



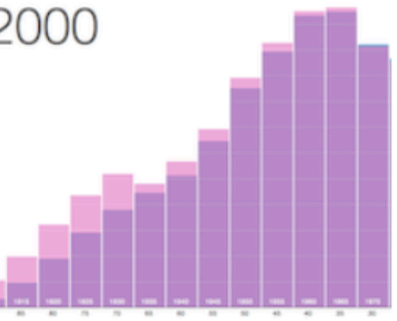
Chord Diagram



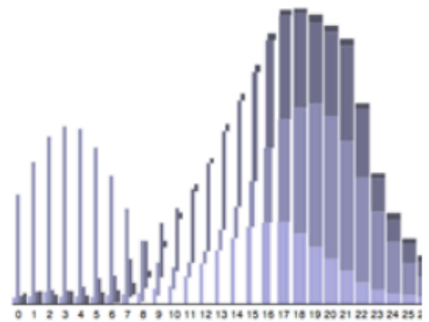
Dendrogram



Population Pyramid

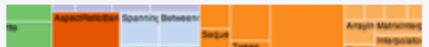


Stacked Bars



Node-Link Tree

Treemap



D3 Gallery

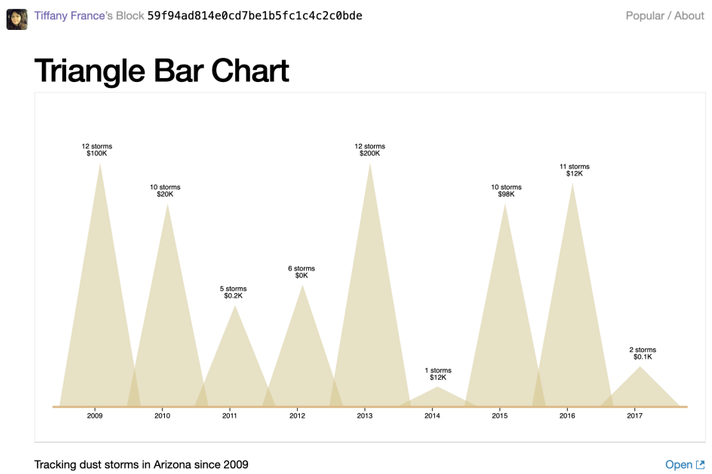
https://github.com/d3/d3/wiki/Gallery

Generally Two Types of Usable Code

Blocks

Observable

D3 Blocks



Blocks are the original way of sharing D3 code

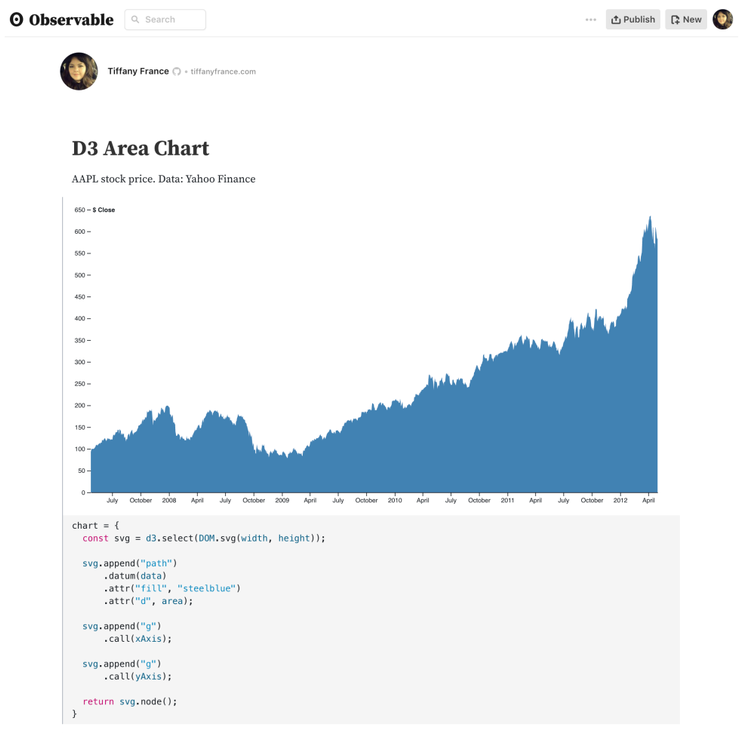
Pros:

- Community sourced, published on GitHub/Gist
- Quick to get up and going

Cons:

- Some blocks are very old
- Version < 4 are not backwards compatible

Observable



Observable is Bostock's latest project

- Similar to Jupyter Notebooks
- Quickly create and view visualizations
- Actively under development
- Can be used for multiple languages/projects

Some things to watch out for:

- Can be complicated to learn/use
- Sometimes difficult to export code for use