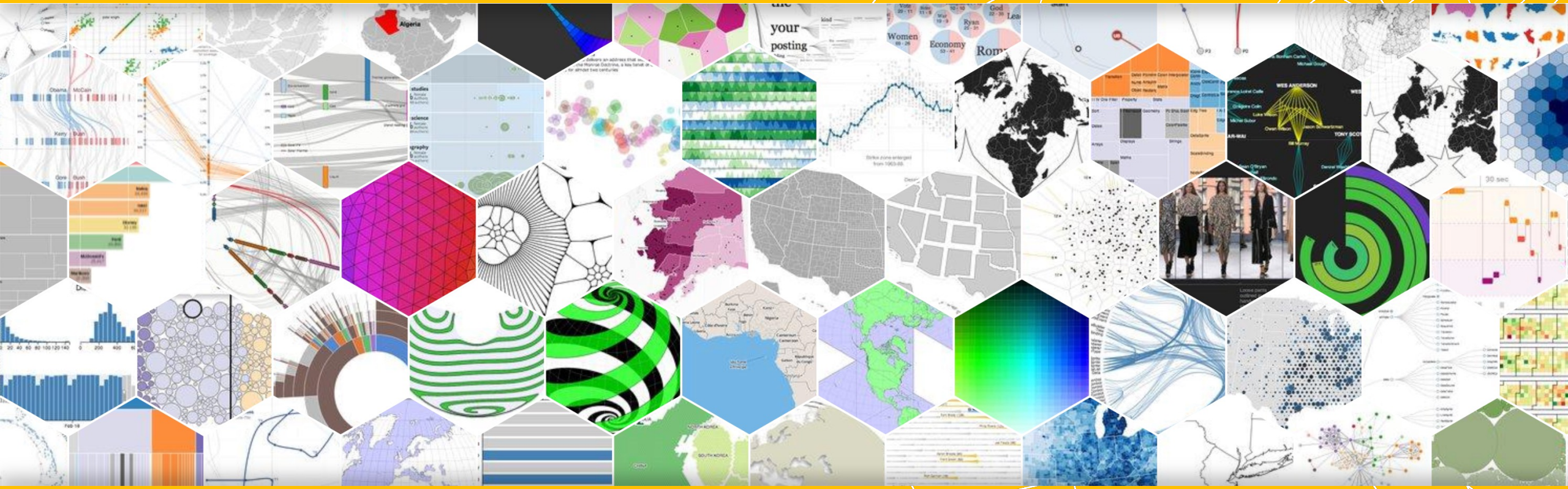
UMA BREVE INTRODUÇÃO AO D3.JS

Diego Mariano

D3.JS - Data-driven documents

D3.js é uma biblioteca **JavaScript** para manipulação de documentos com base em dados.

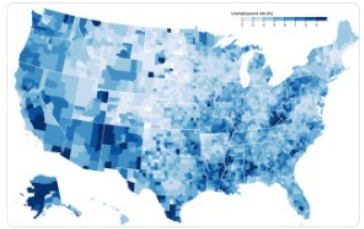
VISUALIZAÇÕES DE DADOS

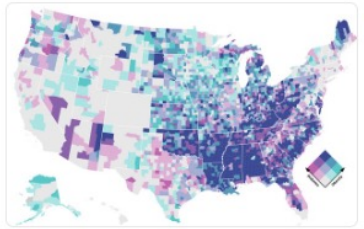


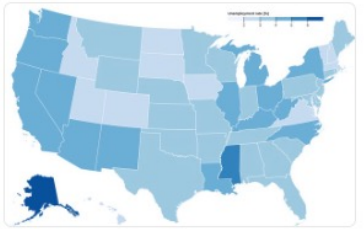
D3js.org

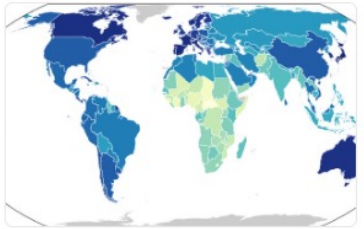
Maps

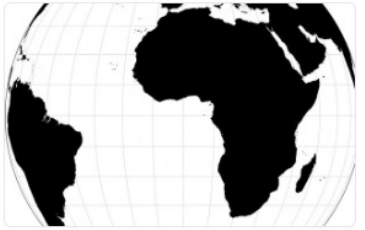
D3 implements a dizzying array of geographic projections. It works great with GeoJSON, TopoJSON, and even shapefiles.

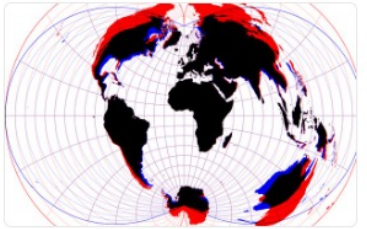

Choropleth

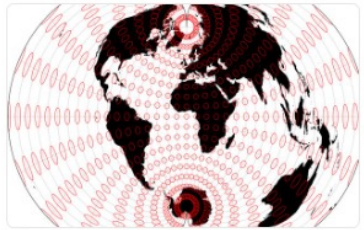

Bivariate choropleth


State choropleth


World choropleth


World map


Projection comparison


Tissot's indicatrix


Web Mercator tiles

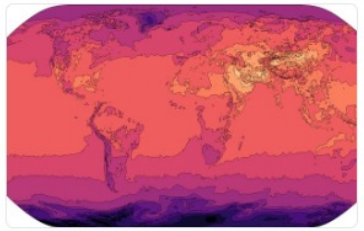

Raster tiles

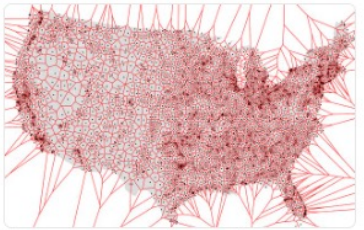

Vector tiles

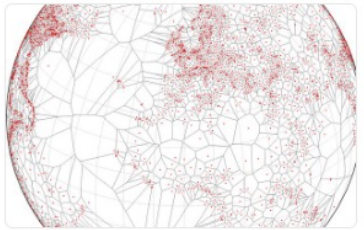

Clipped map tiles

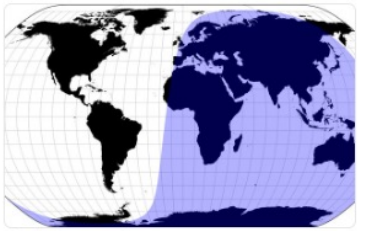

Raster & vector

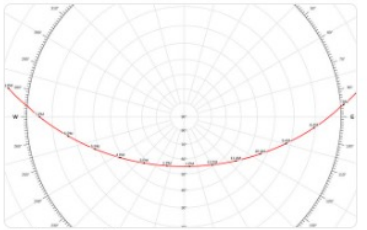

Vector field


GeoTIFF contours

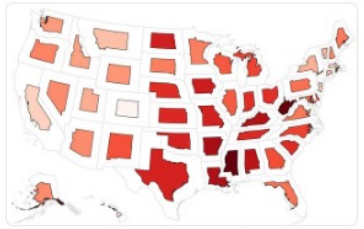

U.S. airports voronoi


World airports voronoi

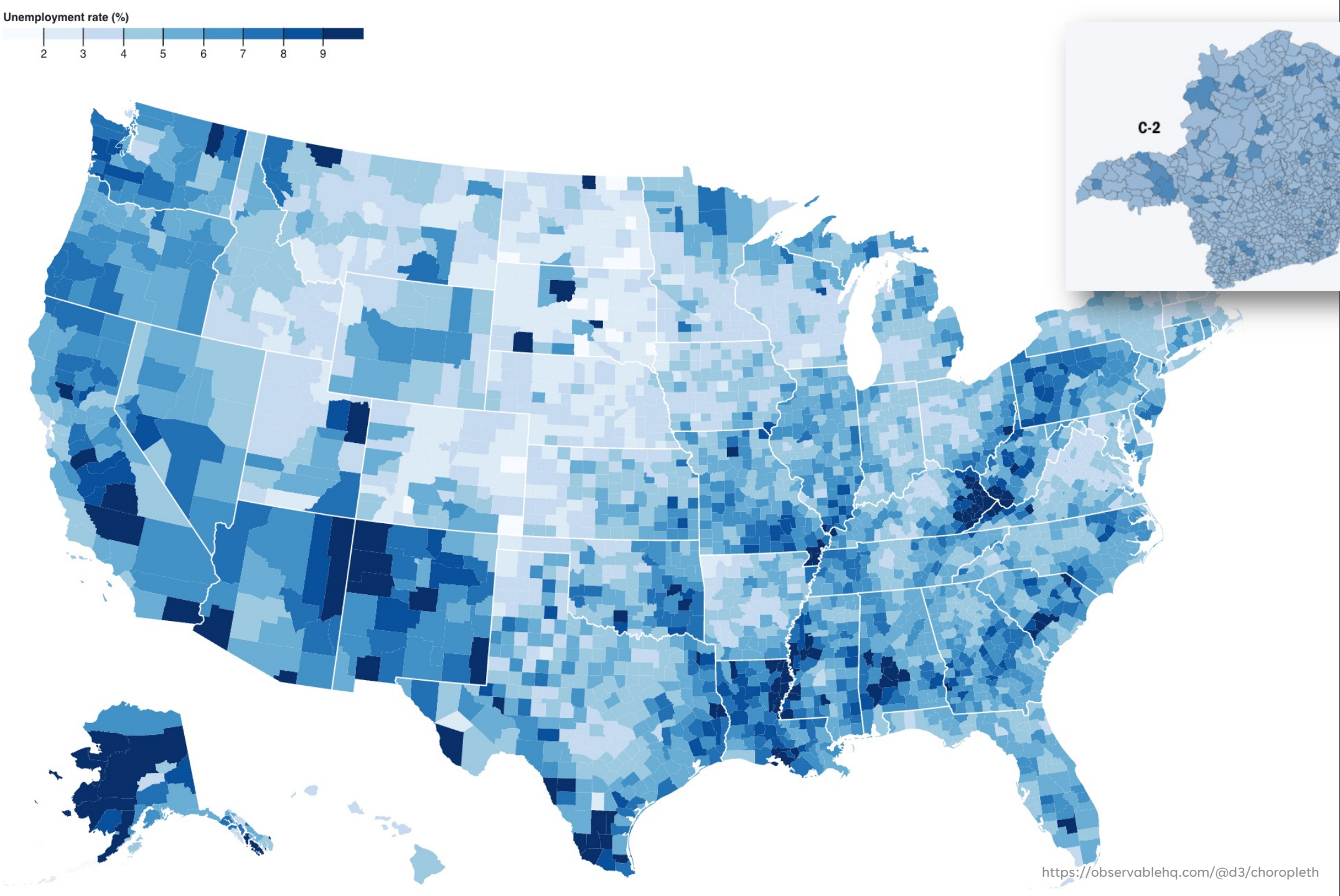

Solar terminator


Solar path


Star map


Non-contiguous cartogram

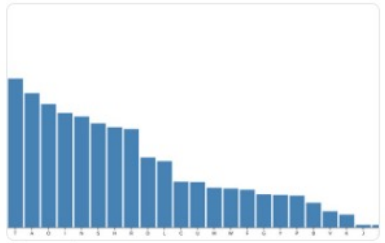
https://observablehq.com/@d3/gallery

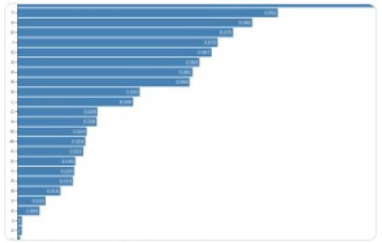


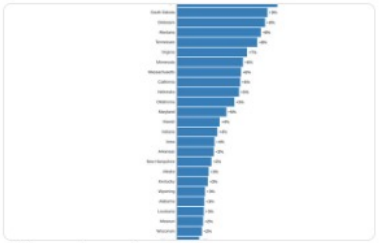
Projeto próprio (dados não publicados)

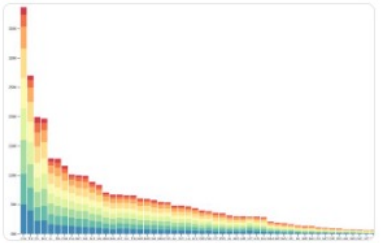
Bars

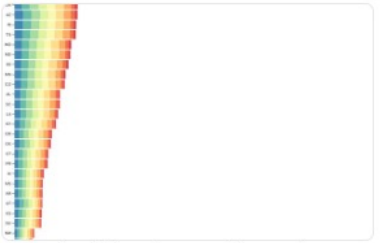
D3 scales and axes support basic charts. Or invent a new form that better serves your needs.

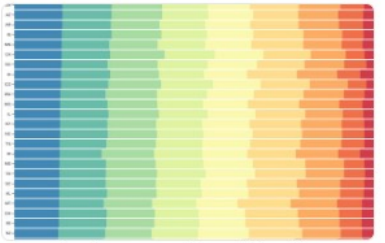

Bar chart

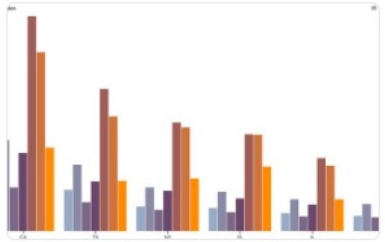

Horizontal bar chart

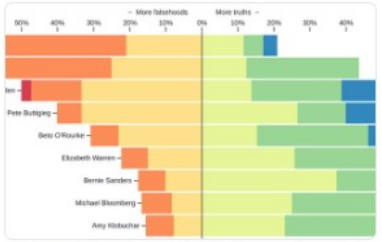

Diverging bar chart


Stacked bar chart

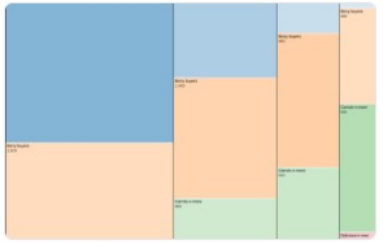

Stacked horizontal bar chart

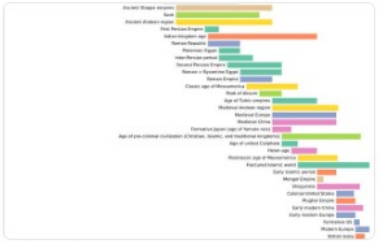

Stacked normalized horizon...

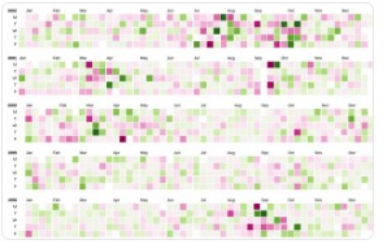

Grouped bar chart

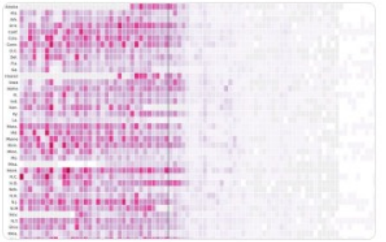


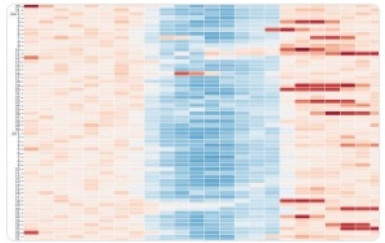
Diverging stacked bar chart

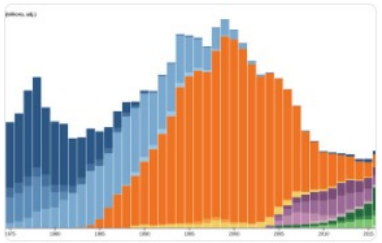

Marimekko chart


World history timeline

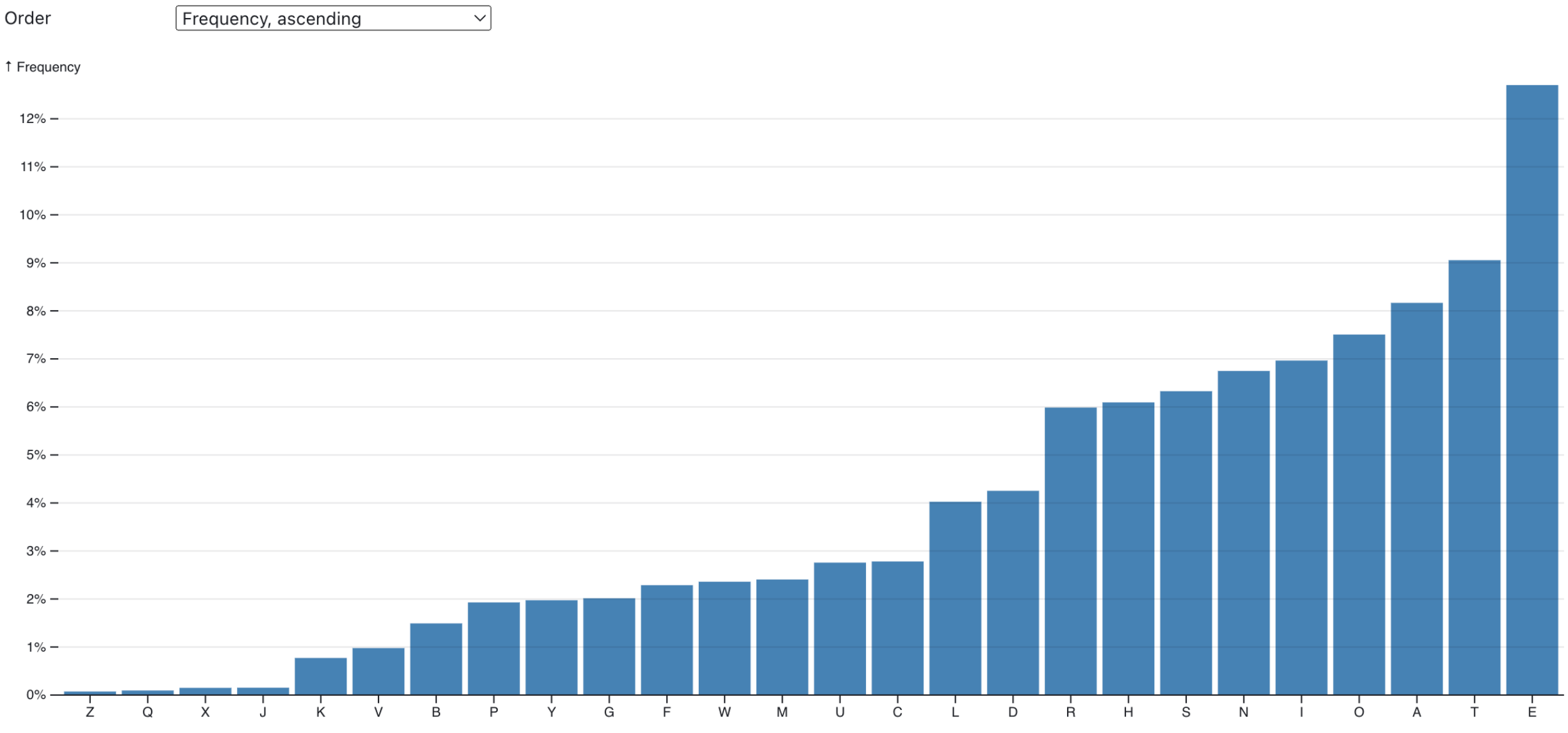

Calendar view


The impact of vaccines


Electricity usage, 2019


Revenue by music format, 1...

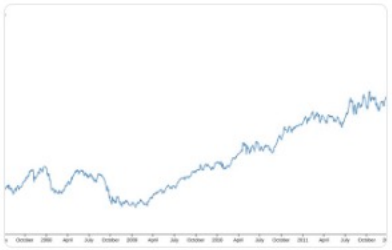
https://observablehq.com/@d3/gallery

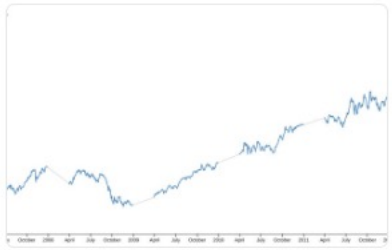


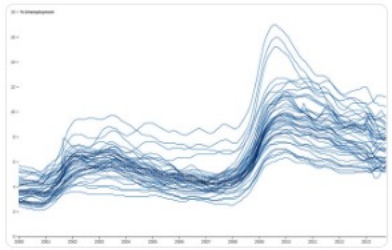
https://observablehq.com/@d3/bar-chart-transitions

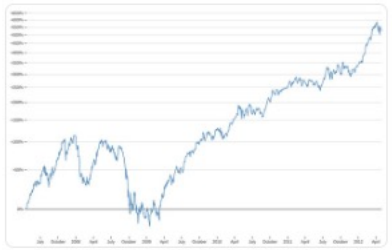
Lines

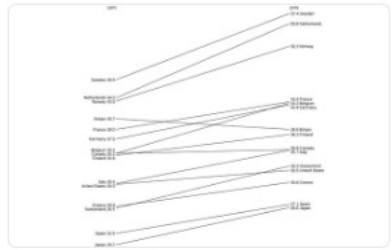
With direct control over graphics, and support for both SVG and Canvas, the possibilities are endless.

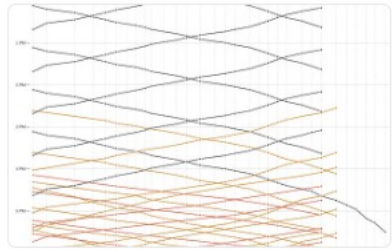

Line chart


Line with missing data

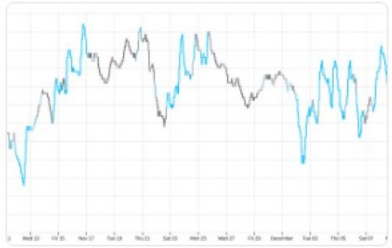

Multi-line chart

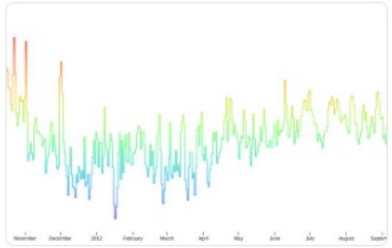

Change line chart

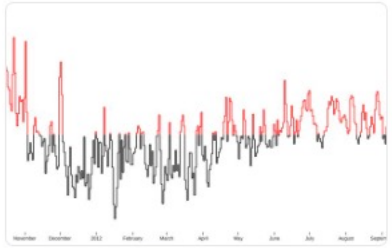

Slope chart

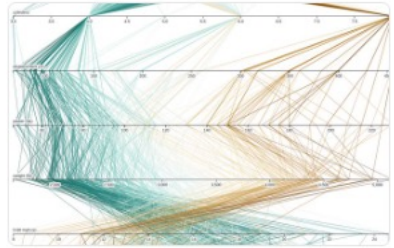

Marey's trains

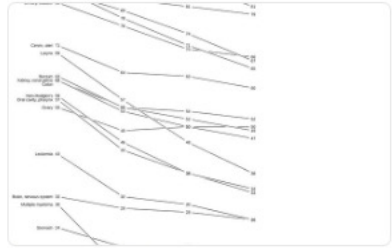

Candlestick chart

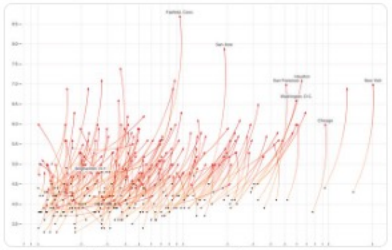

Variable-color line

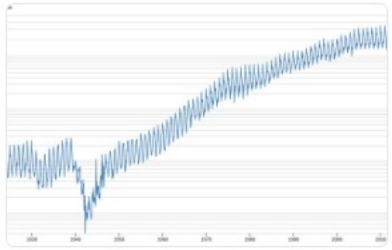

Gradient encoding

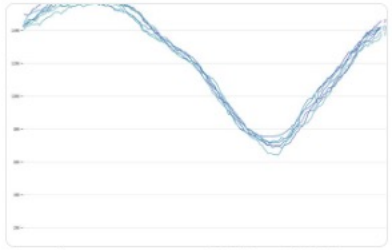

Threshold encoding


Parallel coordinates

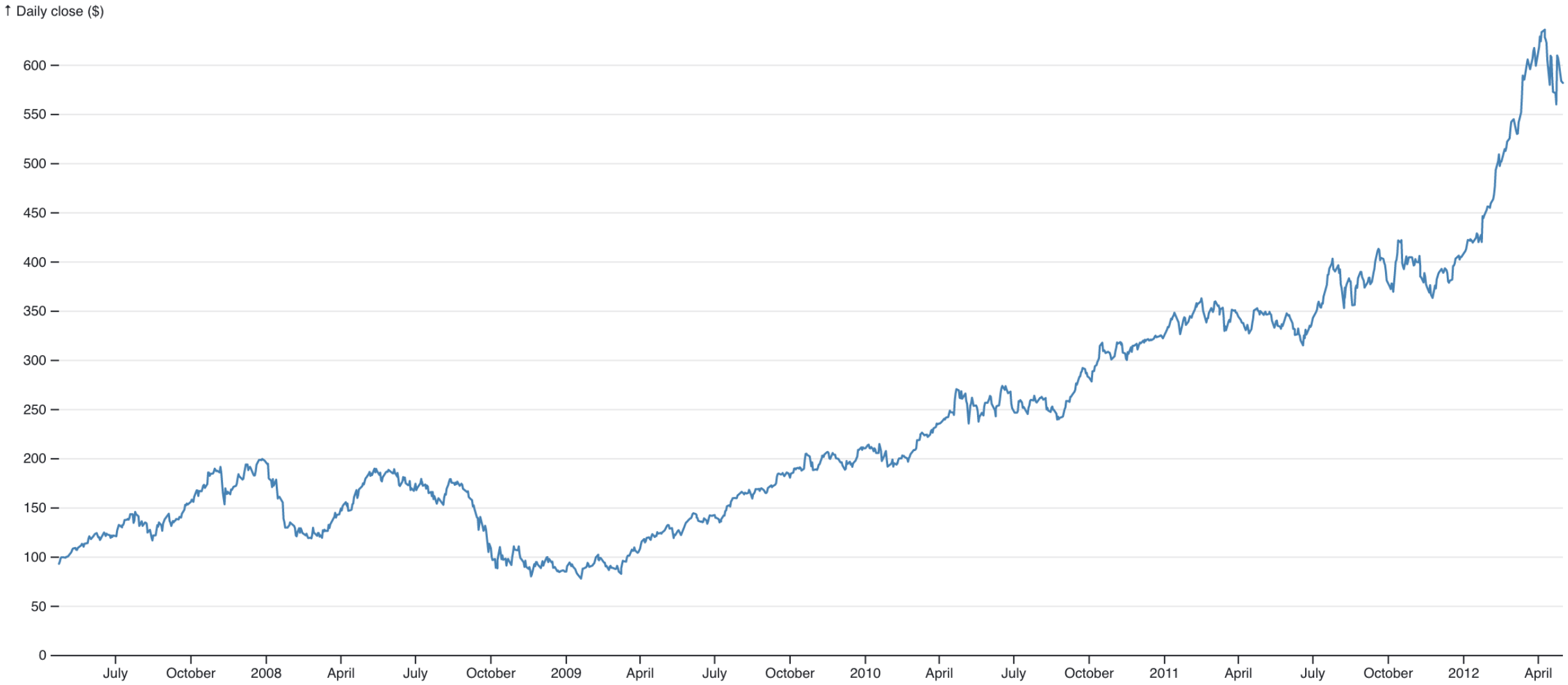

Slope chart


Inequality in American cities


New zealand tourists, 1921–...


Sea ice extent, 1978–2017

https://observablehq.com/@d3/gallery

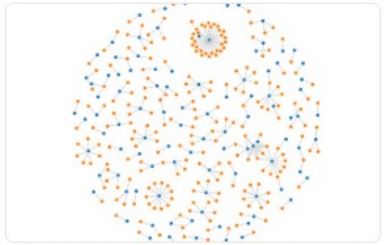


https://observablehq.com/@d3/line-chart

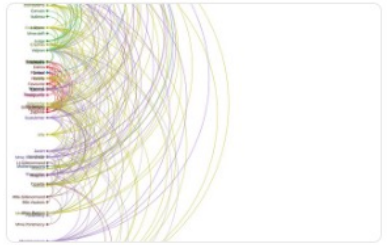
Networks

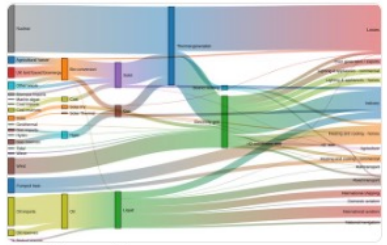
D3 works with networked data (graphs), including simulated forces for resolving competing constraints and an iterative Sankey layout.

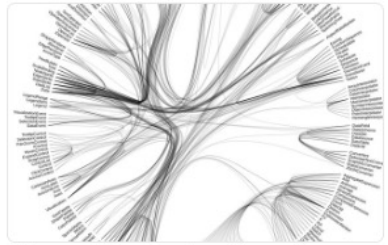

Force-directed graph

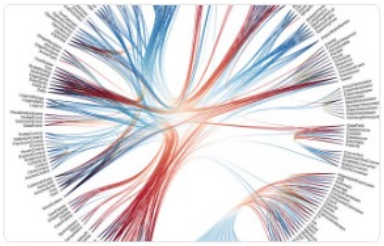

Disjoint force-directed graph

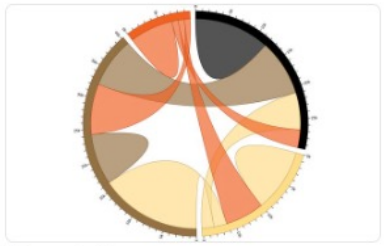

Mobile patent suits

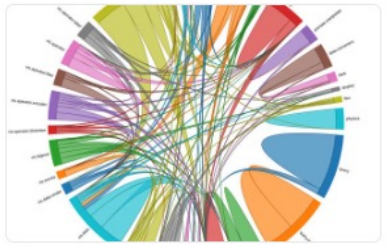

Arc diagram


Sankey diagram


Hierarchical edge bundling

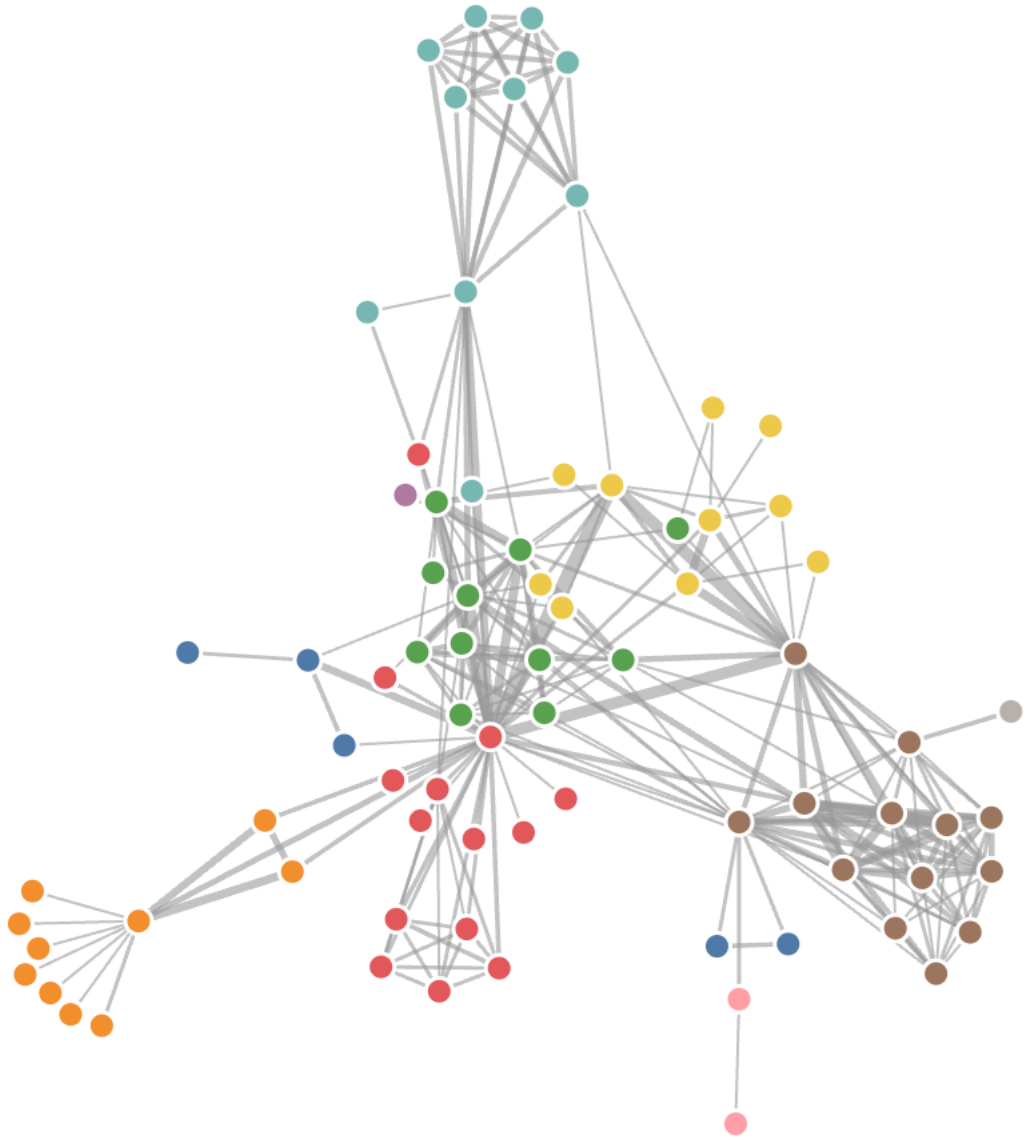

Hierarchical edge bundling


Chord diagram

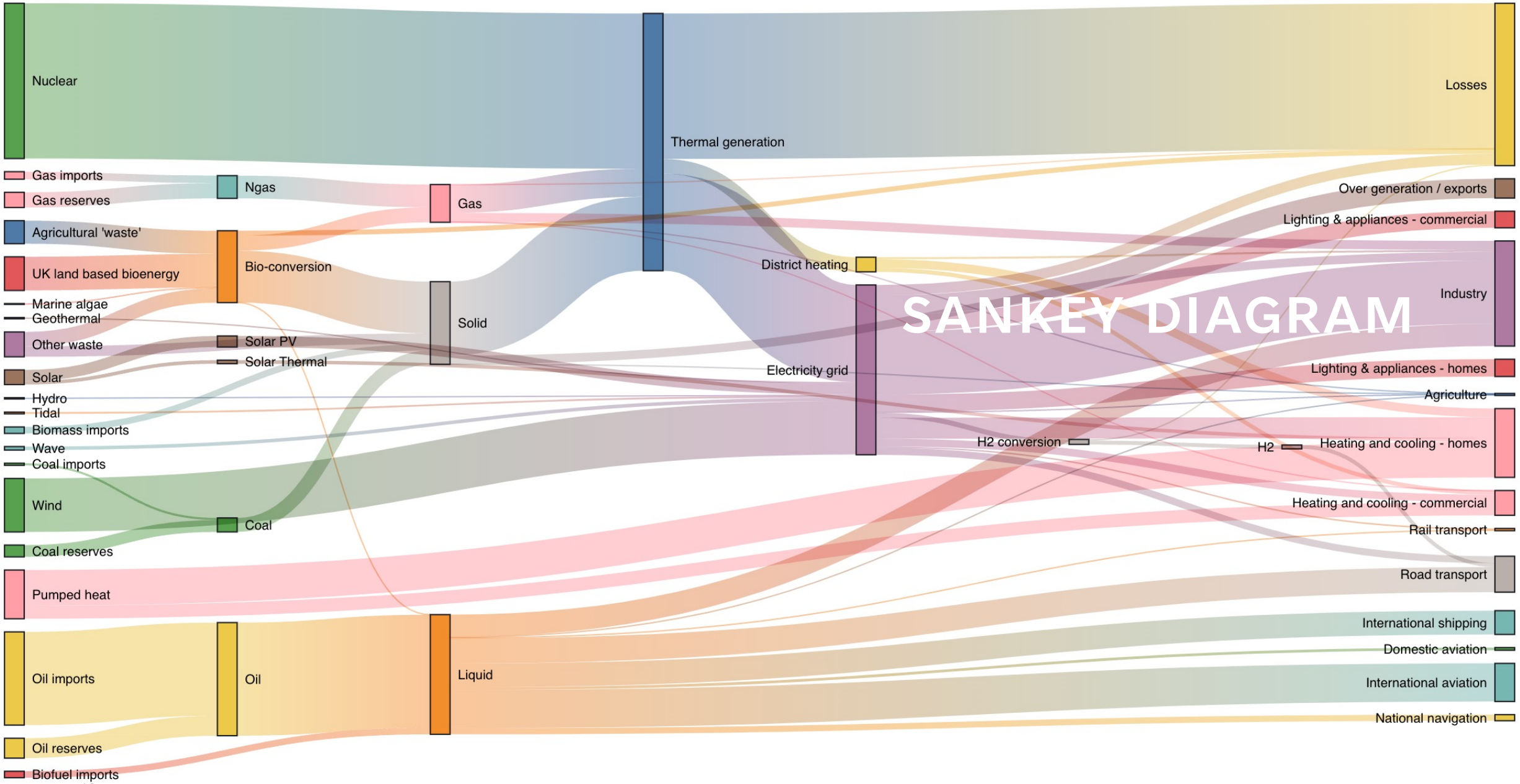

Chord dependency diagram

https://observablehq.com/@d3/gallery

Grafos



https://observablehq.com/@d3/force-directed-graph.com

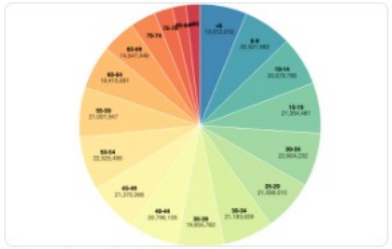


SANKEY DIAGRAM

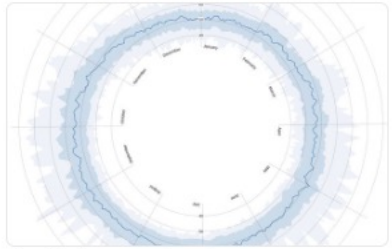
https://observablehq.com/@d3/sankey

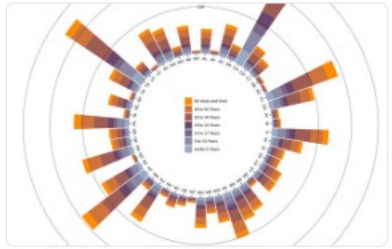
Radial

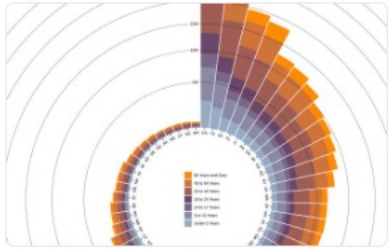
Pies and donuts are good for comparing a part to the whole. And radial layouts can be appropriate for cyclical data.


Pie chart


Donut chart

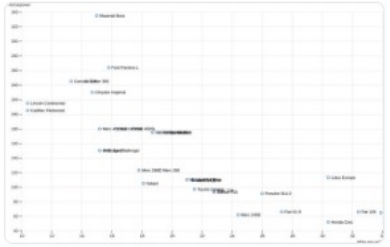

Radial area chart

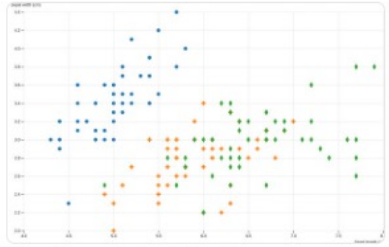

Radial stacked bar chart

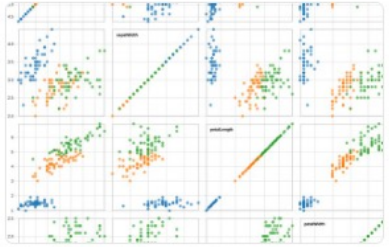

Radial stacked bar chart

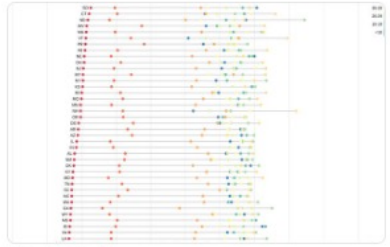
Dots

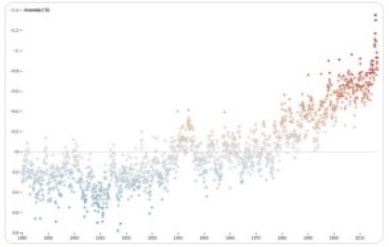
Don’t forget the humble scatterplot. For a single dimension, consider the beeswarm; for finding pairwise dimensional correlations, try a SPLOM.

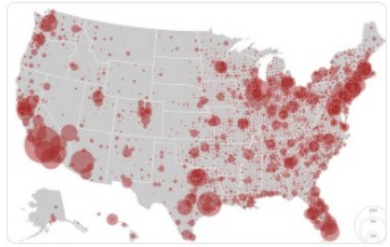

Scatterplot


Scatterplot with shapes


Scatterplot matrix

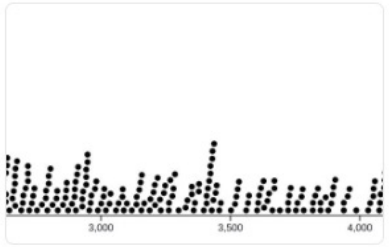

Dot plot


Global temperature trends

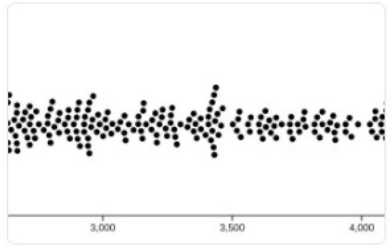

Bubble map


Spike map


Bubble chart



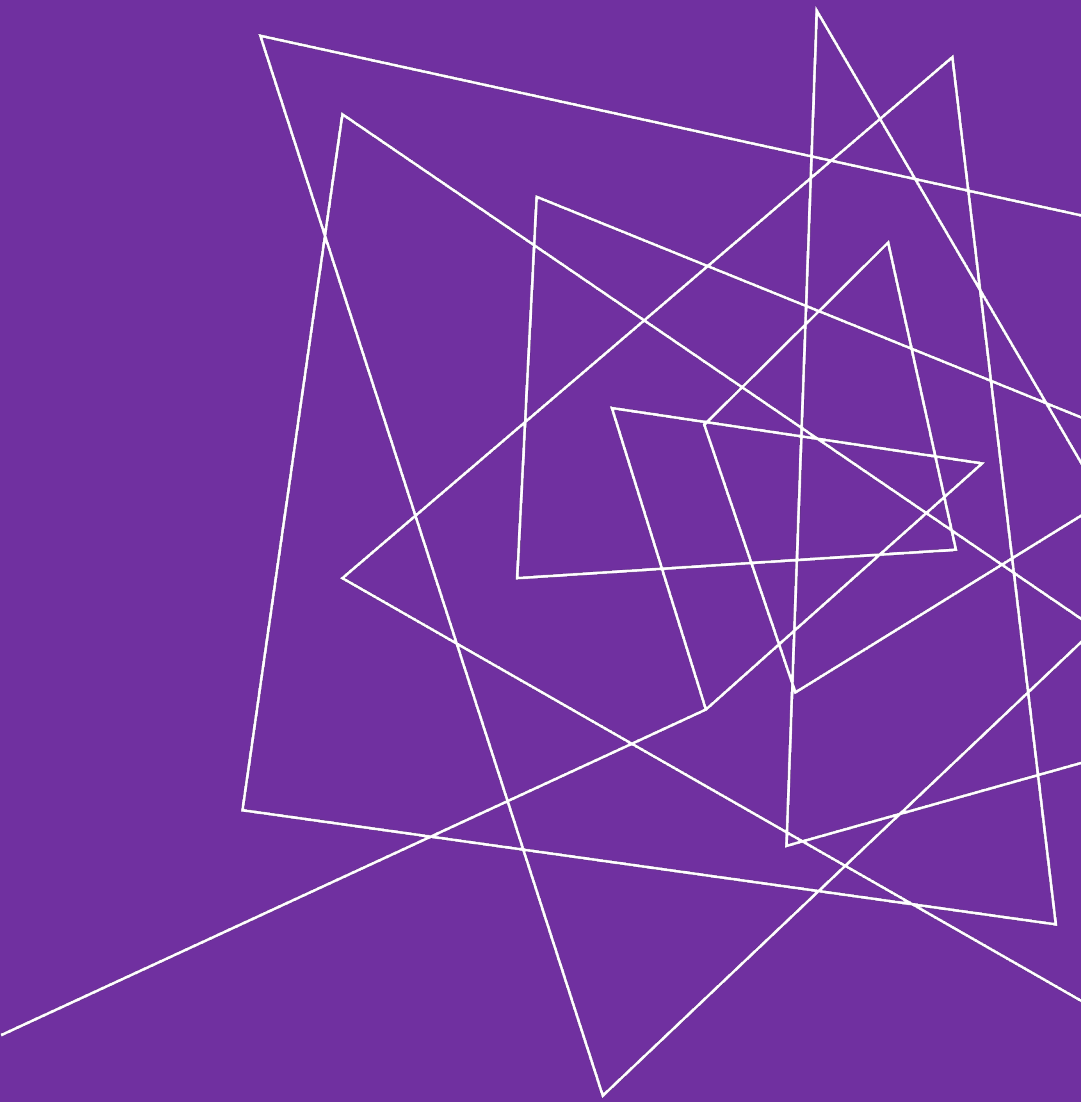
Beeswarm

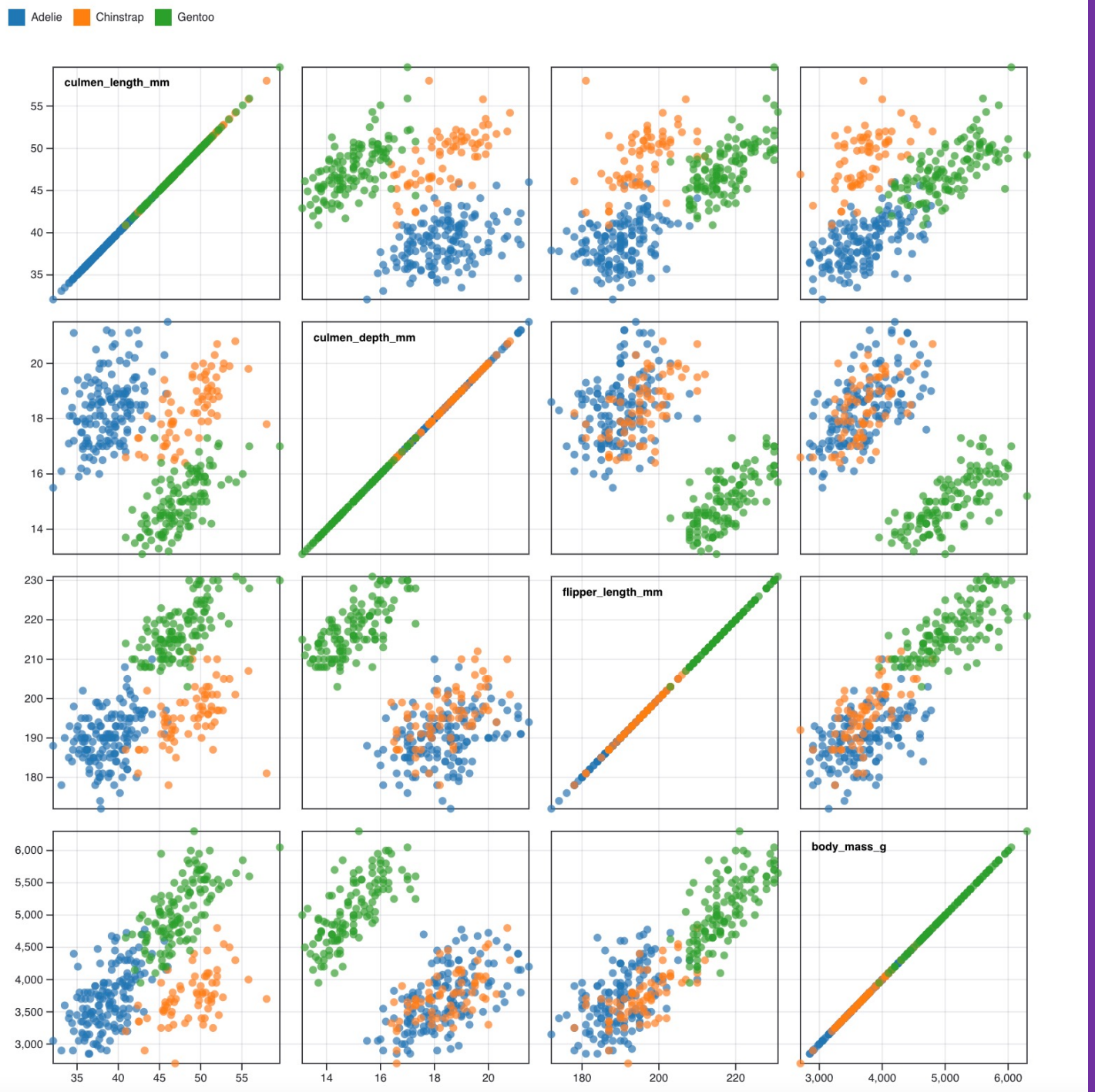


Mirrored beeswarm


Hertzsprung–Russell diagram

https://observablehq.com/@d3/gallery

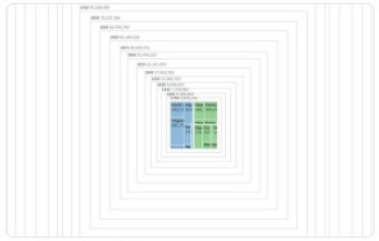




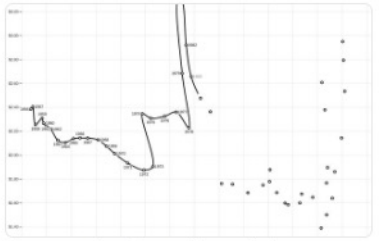
https://observablehq.com/@d3/splom

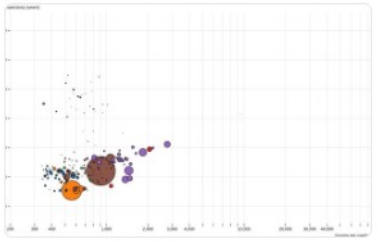
Animation

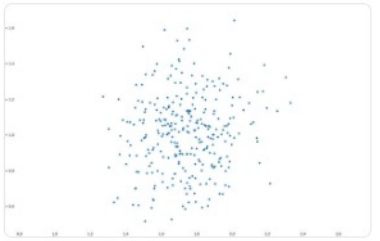
D3's data join, interpolators, and easings enable flexible animated transitions between views while preserving object constancy.


Animated treemap

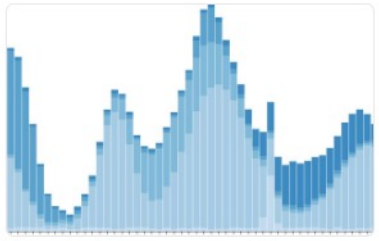

Temporal force-directed gra...

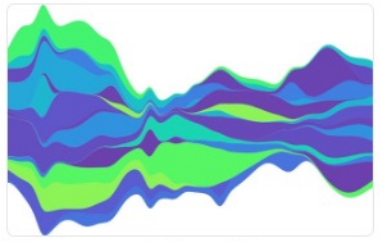

Connected scatterplot


The wealth & health of natio...


Scatterplot tour

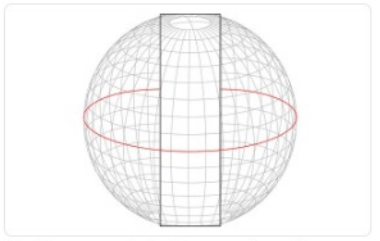
Bar chart race


Stacked-to-grouped bars


Streamgraph transitions

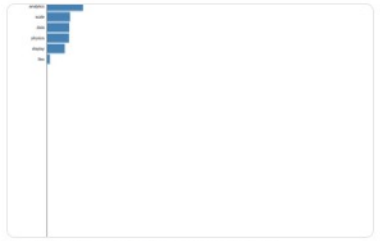

Smooth zooming


Zoom to bounding box

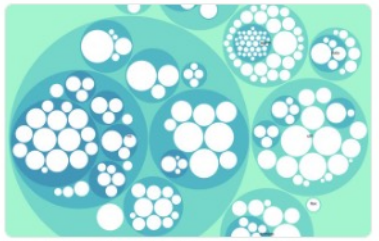

Orthographic to equirectang...

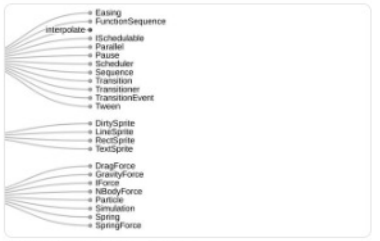

World tour


Walmart's growth


Hierarchical bar chart


Zoomable treemap

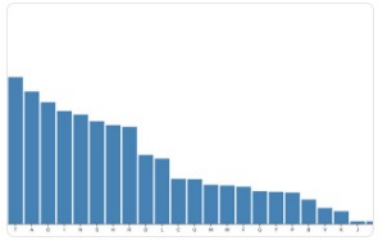

Zoomable circle packing

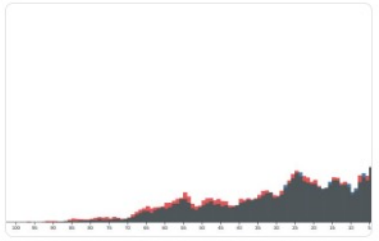


Collapsible tree


Zoomable icicle


Zoomable sunburst


Sortable bar chart


Icelandic population by age,...

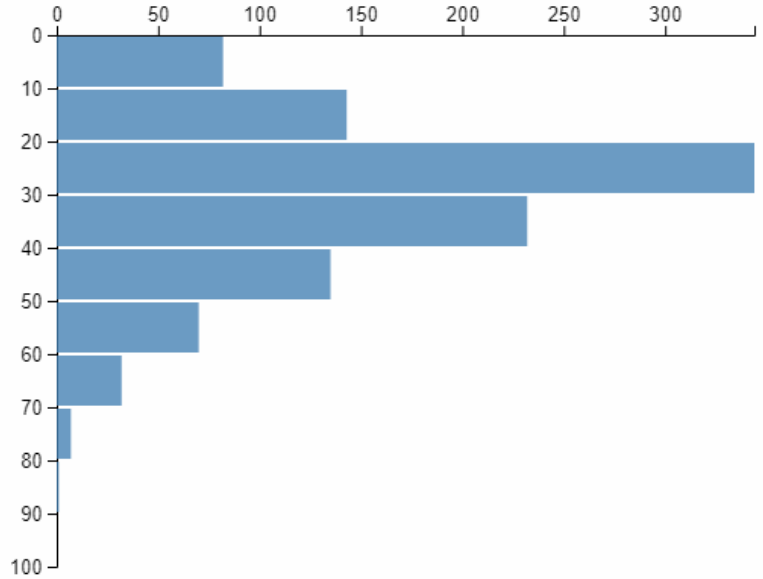
https://observablehq.com/@d3/gallery

Student's First Multiple Coordinated View

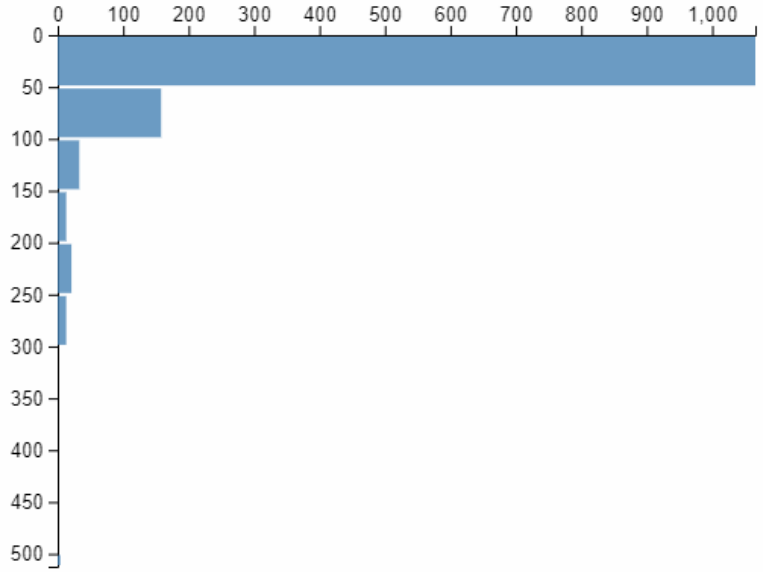
Passenger Class: All Classes **Selected Gender:** None **Selected Survived:** None

Gender Distribution

Age Histogram



Fare Histogram

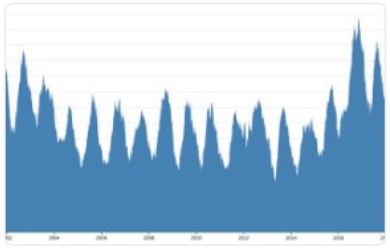


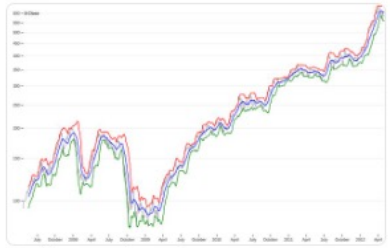
Survived Distribution

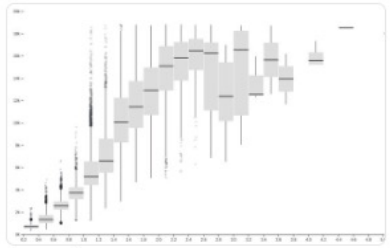
https://github.com/sgratzl/d3tutorial#d3-selections

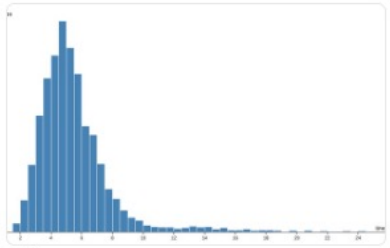
Analysis

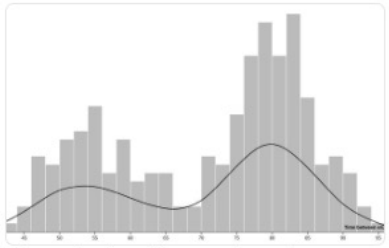
D3 is for more than visualization; it includes tools for quantitative analysis, such as data transformation, random number generation, hexagonal binning, and contours via marching squares.

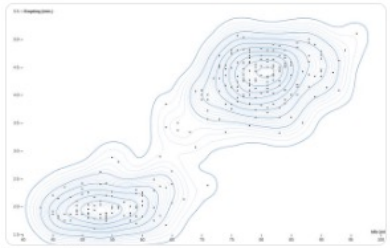

Moving average

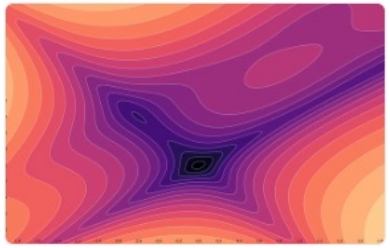

Bollinger bands

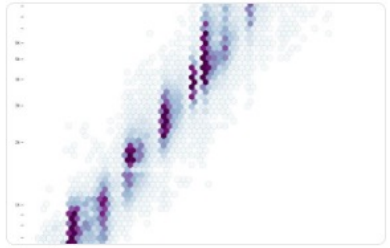

Box plot

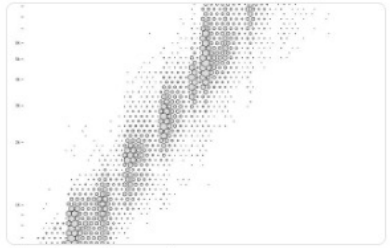

Histogram

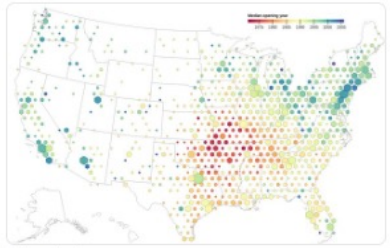

Kernel density estimation

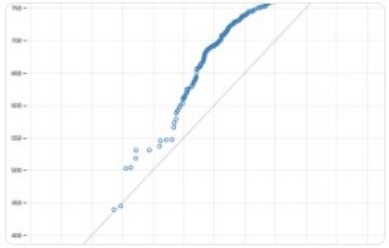

Density contours

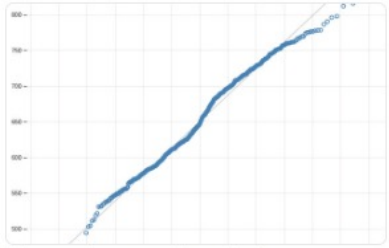

Contours

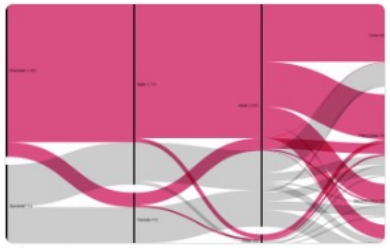

Hexbin


Hexbin (area)


Hexbin map


Q–Q plot


Normal quantile plot


Parallel sets

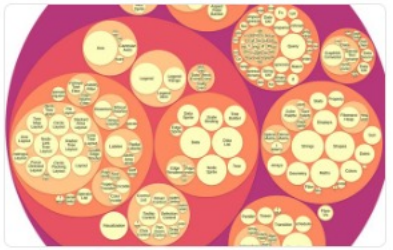
Hierarchies

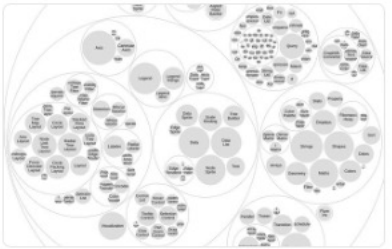
D3 supports hierarchical data, too, with popular layouts such as treemaps, tidy trees, and packed circles. And you retain complete control over how the data is displayed.


Treemap

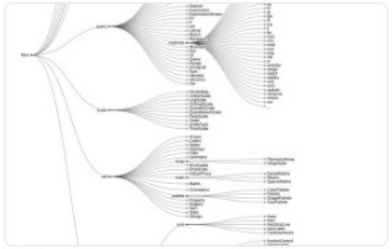

Cascaded treemap

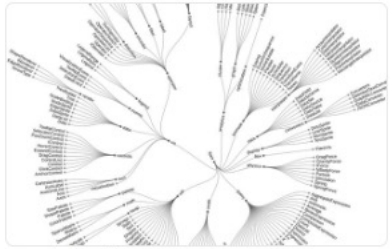

Nested treemap

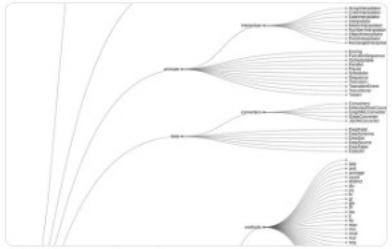

Circle packing

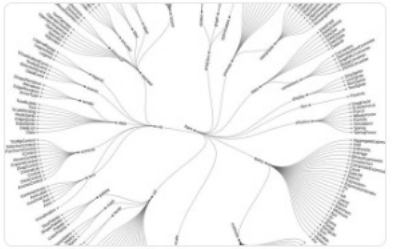

Circle packing (monochrome)


Indented tree


Tidy tree


Radial tidy tree

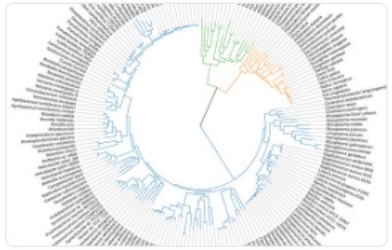

Cluster dendrogram


Radial dendrogram


Sunburst


Icicle

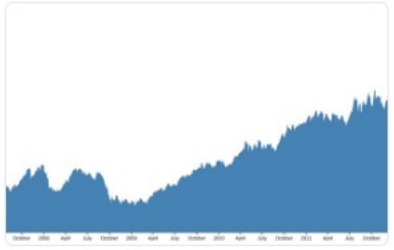

Tangled tree visualization

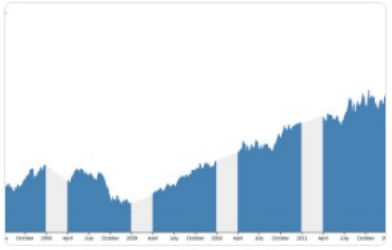

Phylogenetic tree

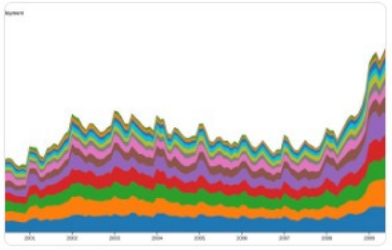

Force-directed tree

Areas

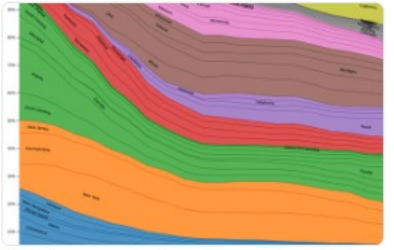
Go beyond basic area charts with difference charts or streamgraphs. Ridgeline plots and horizon charts are great for comparing many simultaneous time series.

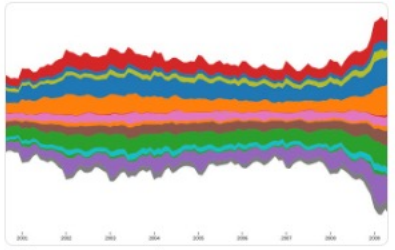

Area chart

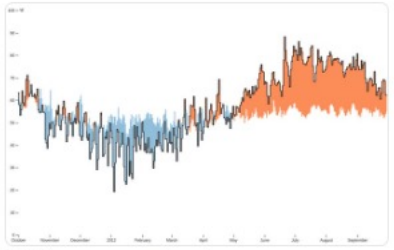

Area with missing data

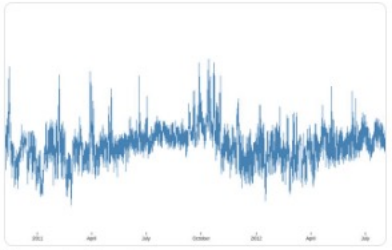

Stacked area chart

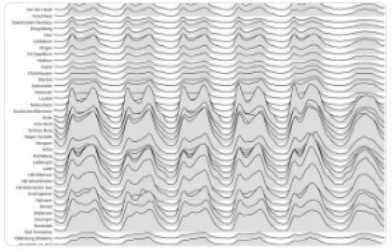

Normalized stacked area ch...

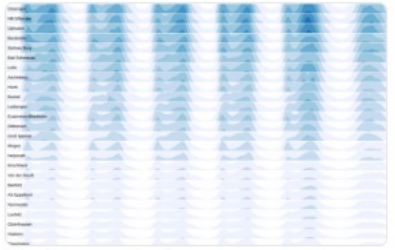

U.S. population by state, 17...

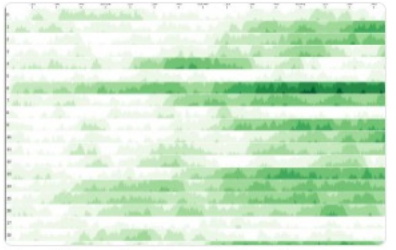

Streamgraph


Difference chart


Band chart


Ridgeline (joy) plot

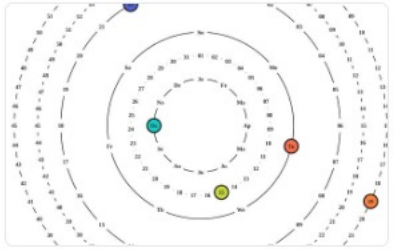

Horizon chart

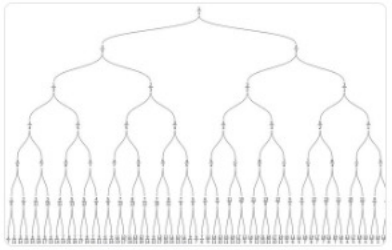

Realtime horizon chart

https://observablehq.com/@d3/gallery

Just for fun

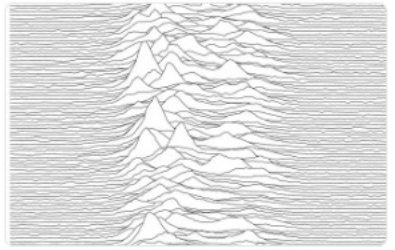
Why not have a little fun? Life's not just about work, you know.

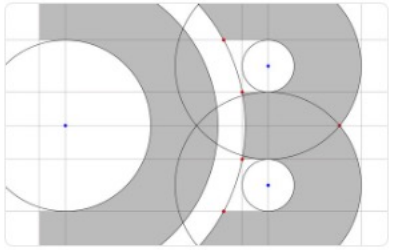

Polar clock

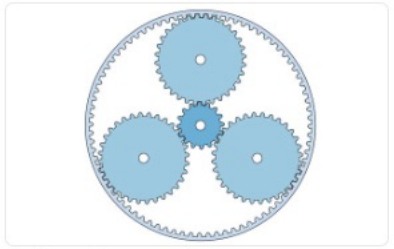

Stern–brocot tree


Voronoi stippling


Watercolor


PSR B1919+21

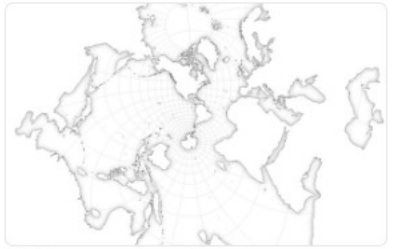

Logo

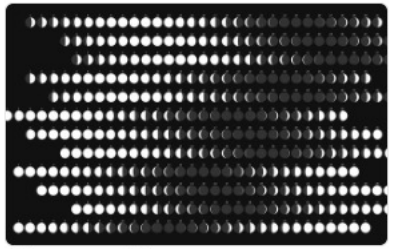

Epicyclic gearing

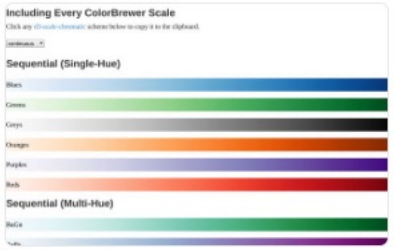

Owls to the max


Tadpoles

Word cloud

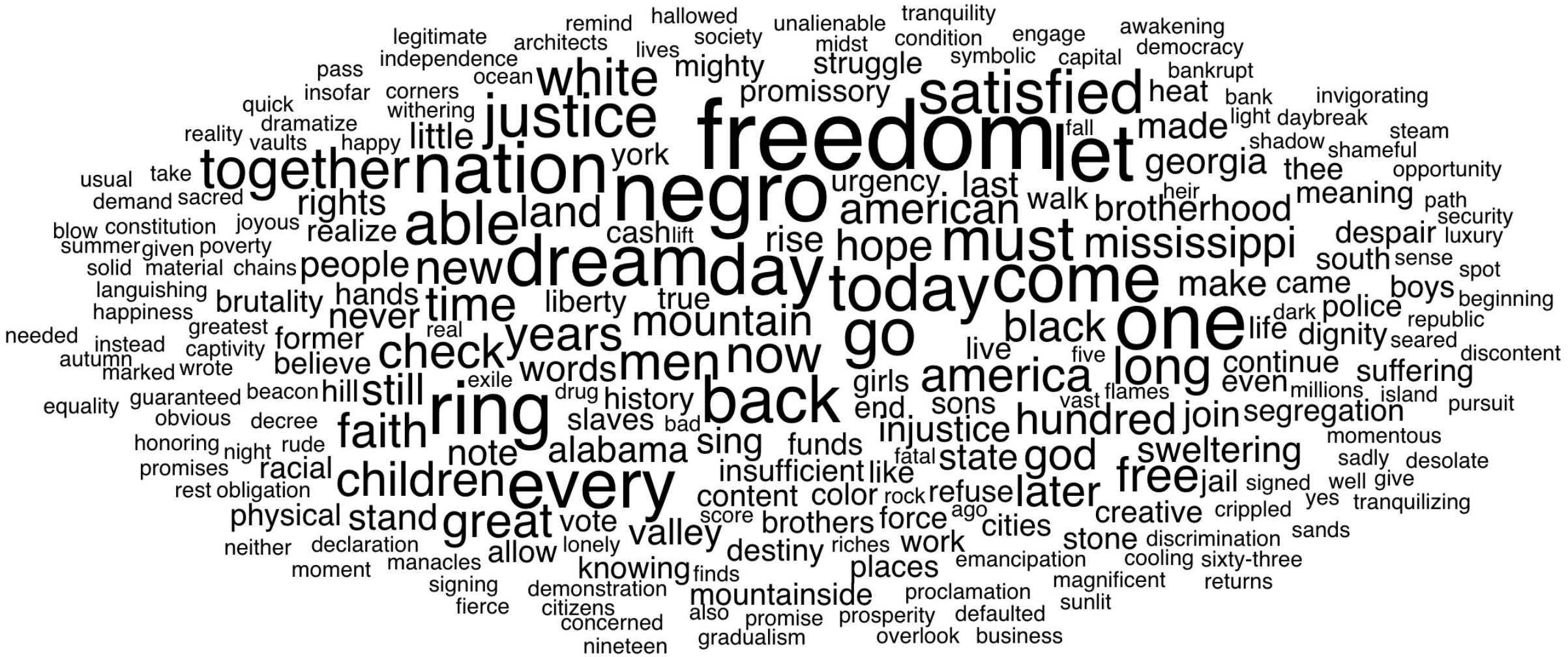

Spilhaus shoreline map


Phases of the moon

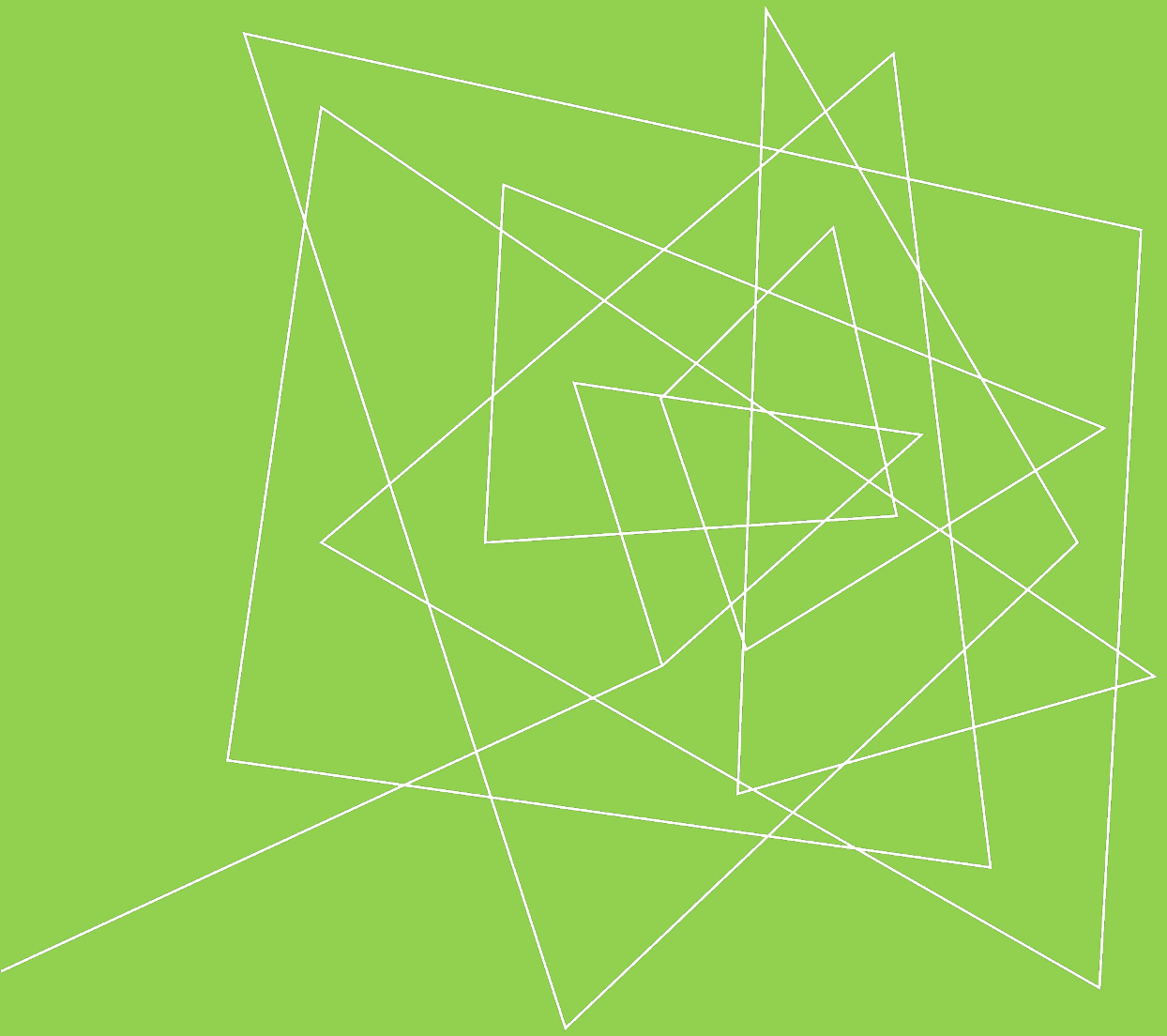


Color schemes

https://observablehq.com/@d3/gallery

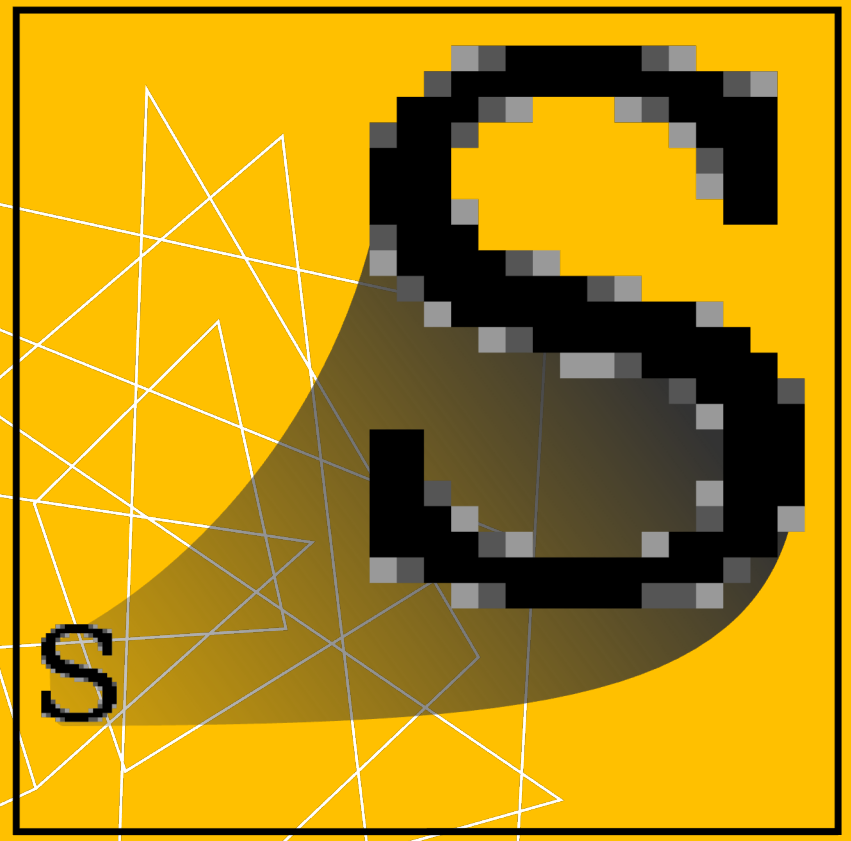


https://observablehq.com/@d3/word-cloud



VANTAGENS DO USO DE D3

Vantagens



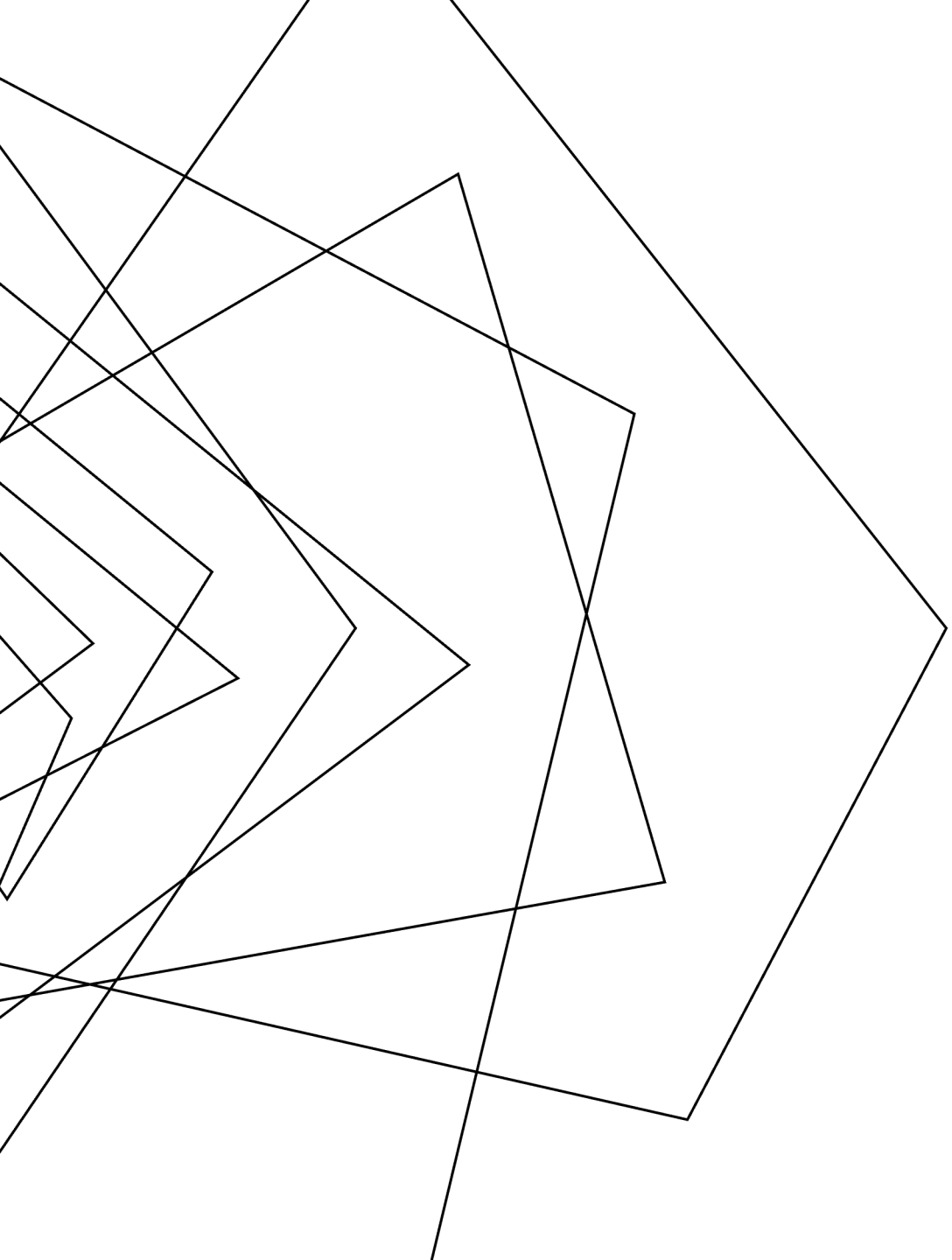
Raster

GIF, JPEG, PNG

Vector

SVG

Fonte: Yug, modifications by Cfaerber et al. (CC BY 2.5). Disponível em: https://commons.wikimedia.org/wiki/File:Bitmap_VS_SVG.svg#metadata



VANTAGENS – COMUNIDADE ATIVA

AMPLA GALERIA DE EXEMPLOS

https://observablehq.com/@d3/gallery

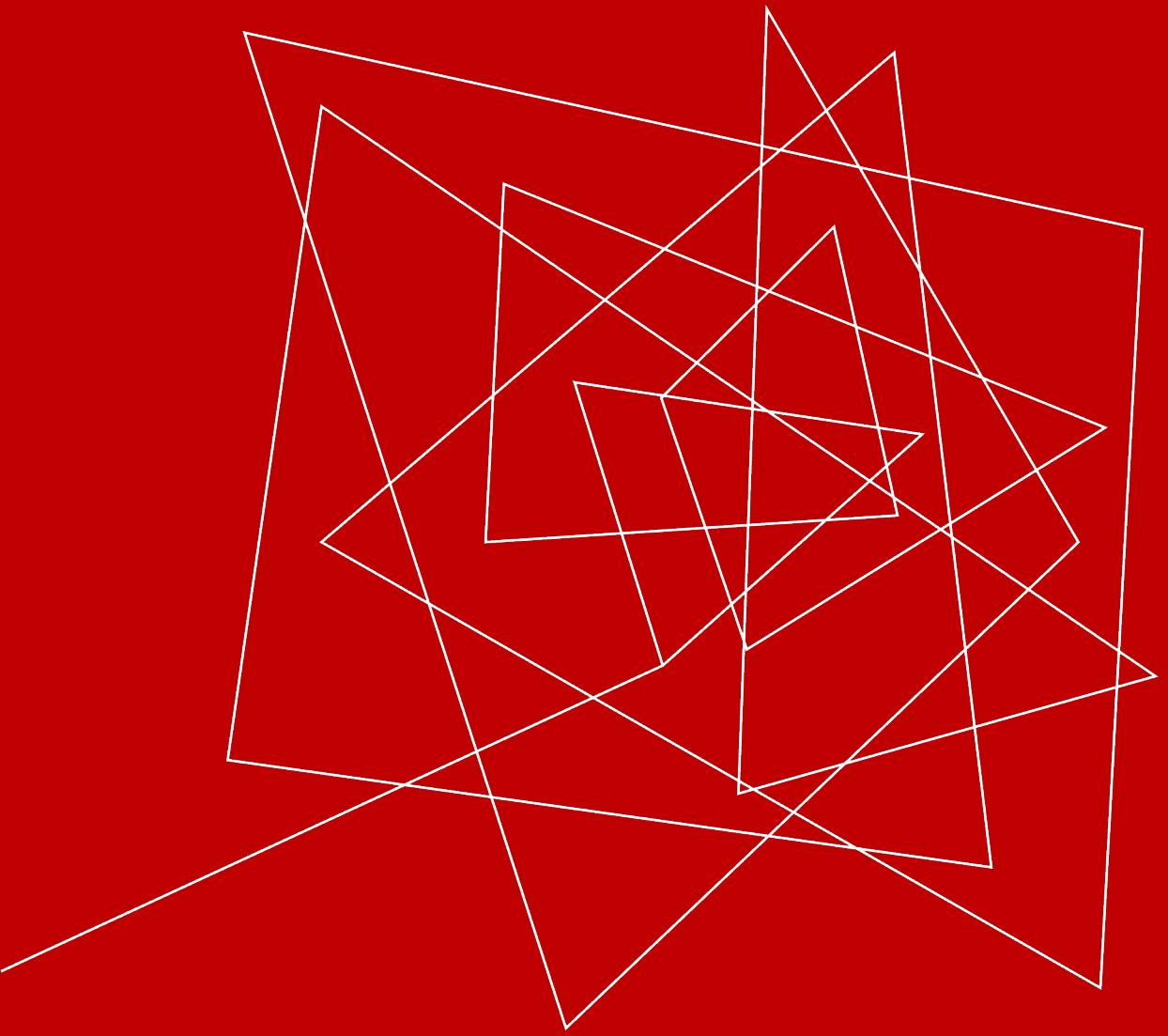
CURSOS DE D3 DISPONÍVEL GRATUITAMENTE

https://www.youtube.com/watch?v=30lR5BlcO48&list=PL9yYRbwpkykuK6LSMLH3bAaPpXaDUXcLV

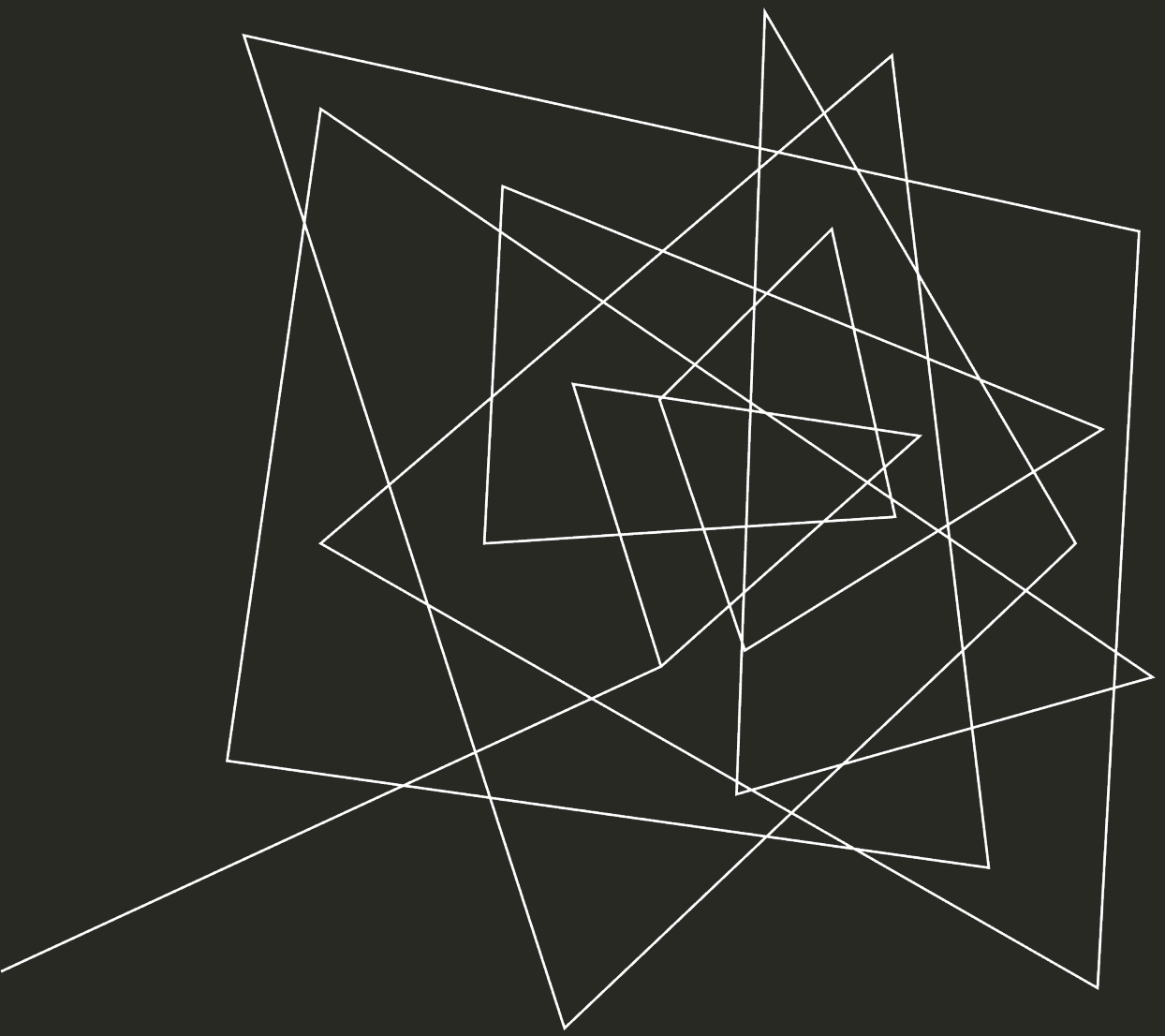
TUTORIAIS

https://d3-graph-gallery.com/index.html

The D3.js Graph Gallery



DESVANTAGENS DO D3



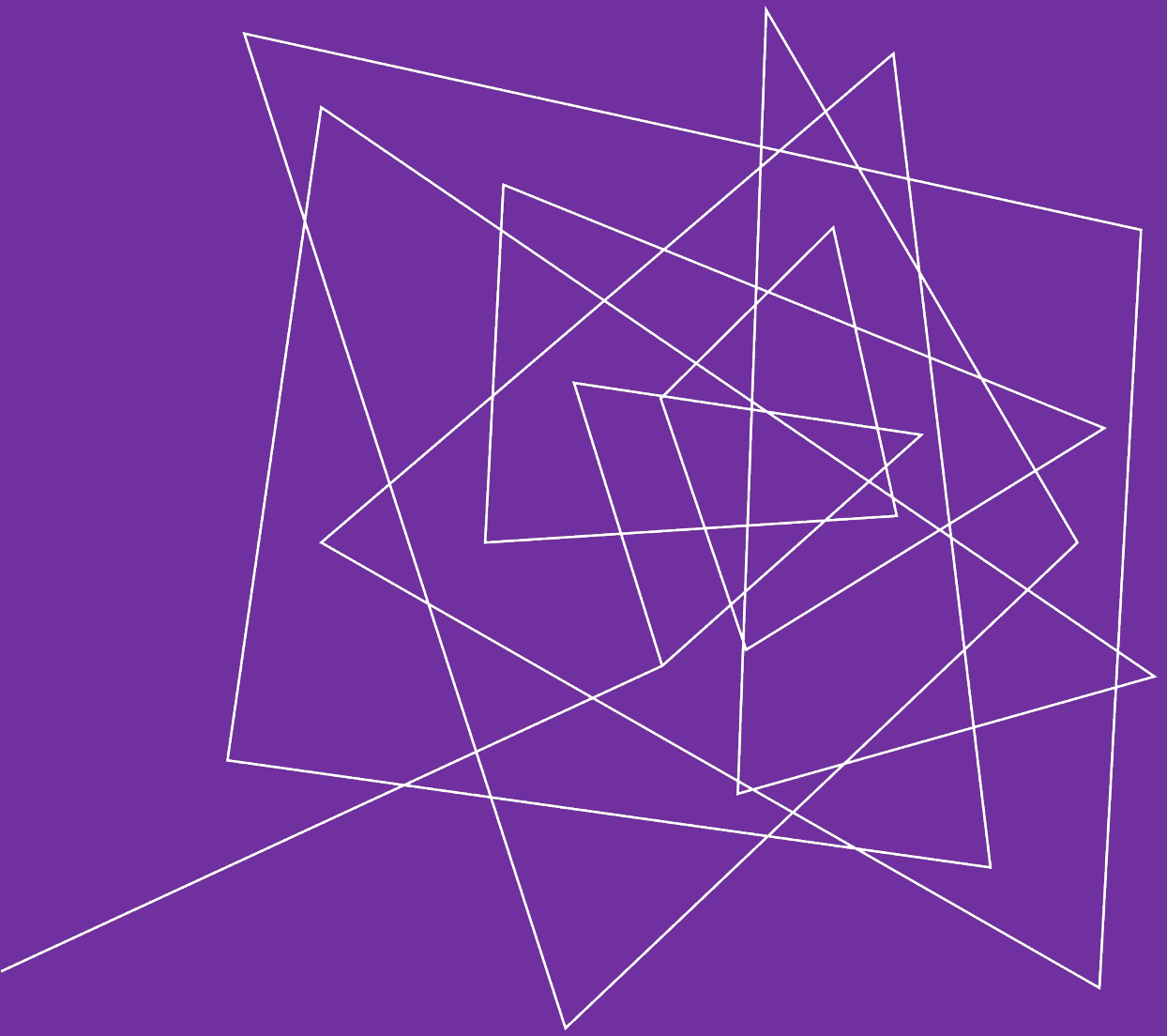
Criar visualizações que fogem aos exemplos é complexo

Desvantagens

Para criar novas visualizações deve-se ter conhecimentos de:

+ Sintaxe da biblioteca

Novos exemplos:
React + Node.js



FERRAMENTAS ALTERNATIVAS

Tableau

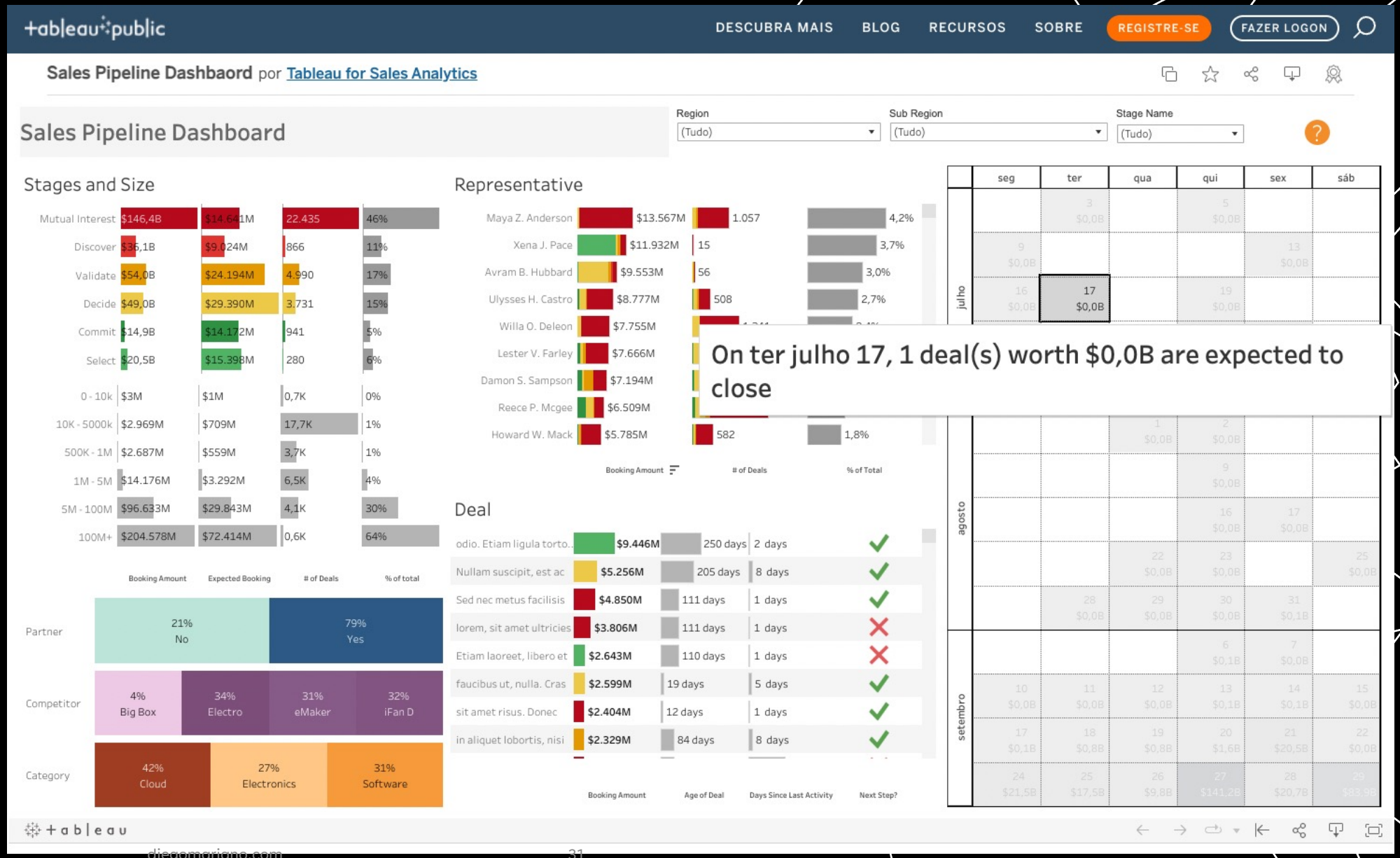
https://www.tableau.com/pt-br

Vantagens:

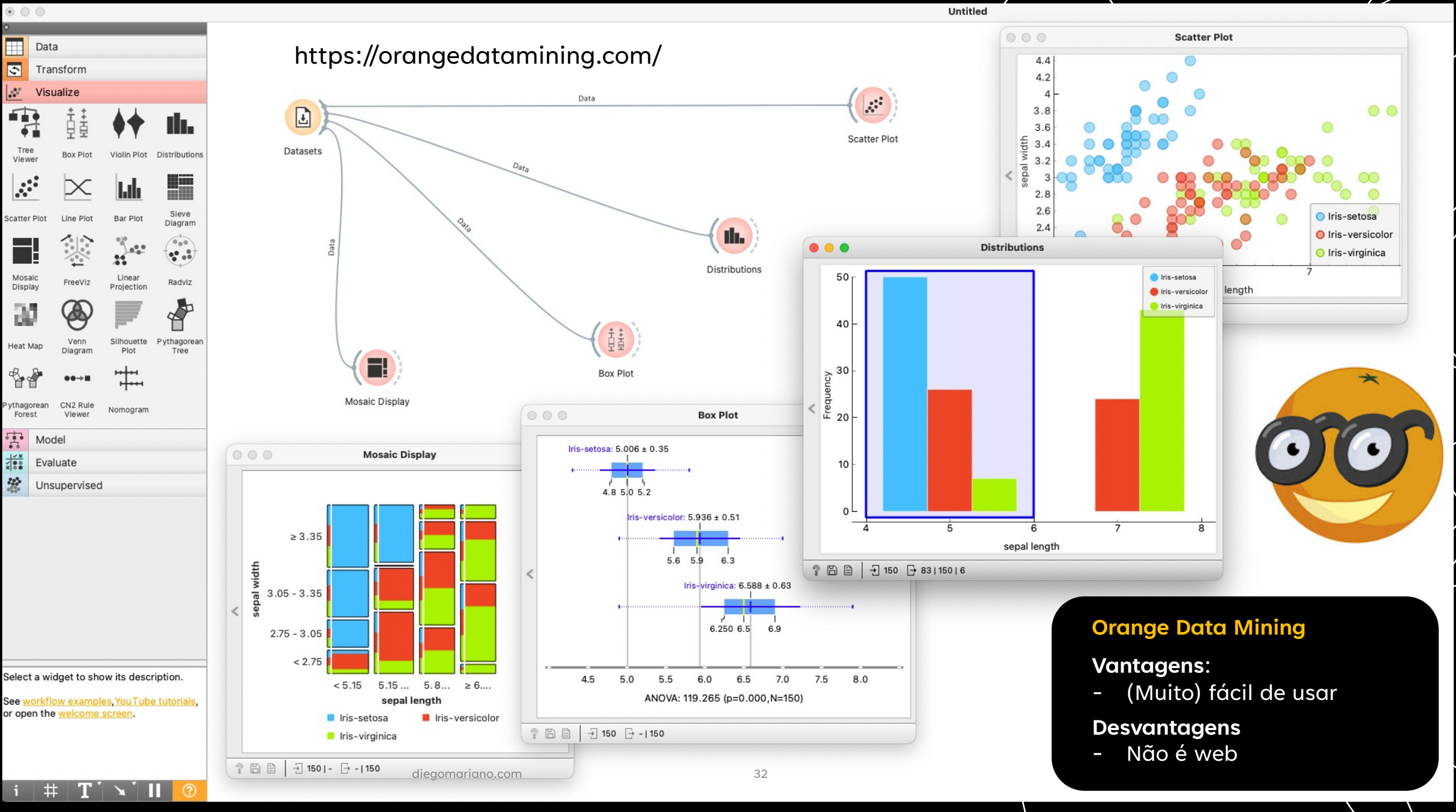
- Fácil de usar
- Desktop/web

Desvantagens

- Lento
- ”Engessado”
- É pago



https://orangedatamining.com/



Orange Data Mining

Vantagens:
- (Muito) fácil de usar

Desvantagens
- Não é web

D3plus

Find components /

- Introduction
- Installation
- Accessibility

CHARTS

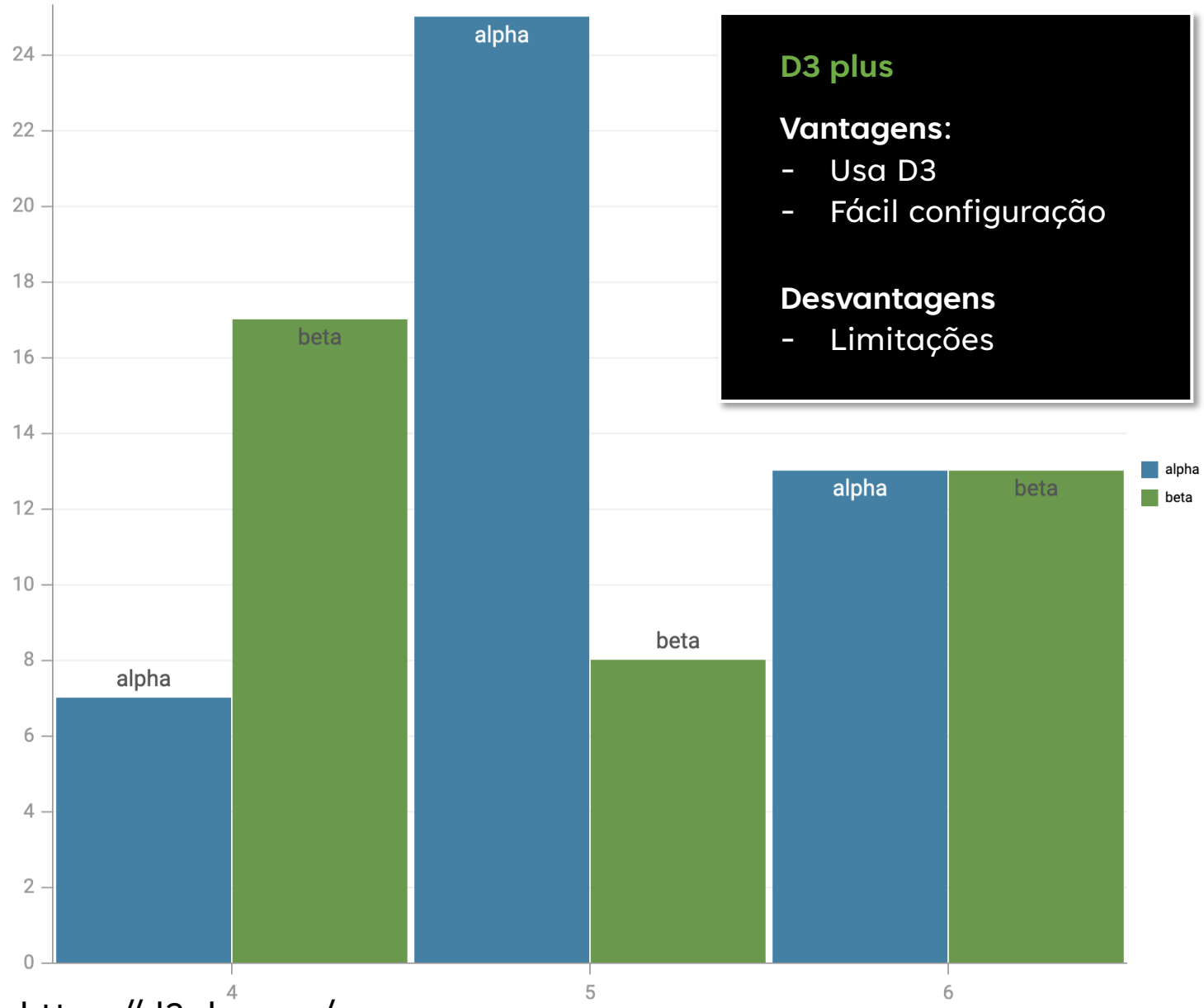
- Area Chart
- Bar Chart
- Box Whisker
- Bubble Pack
- Bump Chart
- Choropleth Map
- Coordinate Point Map
- Donut Chart
- Line Plot
- Matrix
- Network
- Pie Chart
- Priestley
- Radar Plot
- Radial Matrix
- Rings
- Sankey
- Scatter Plot
- Stacked Area
- Treemap

ADVANCED

- Area Chart
  - Changing Area Opacity
- BarChart
- BoxWhisker

Published on Chromatic

Canvas | Docs



D3 plus

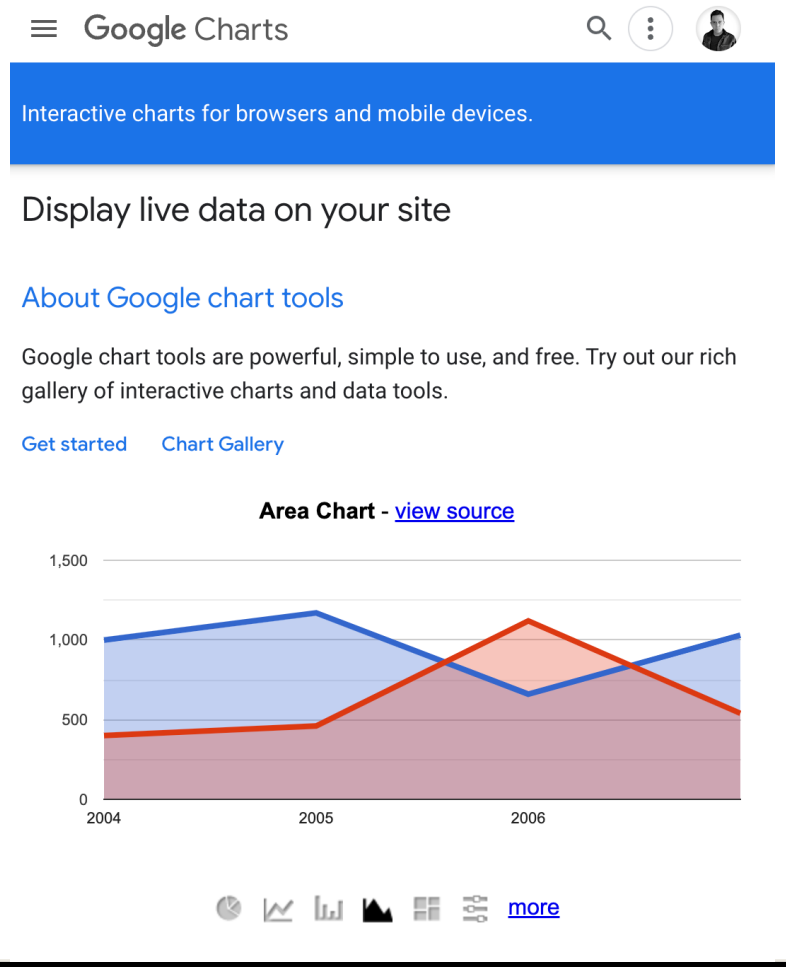
Vantagens:
- Usa D3
- Fácil configuração

Desvantagens
- Limitações

https://d3plus.org/

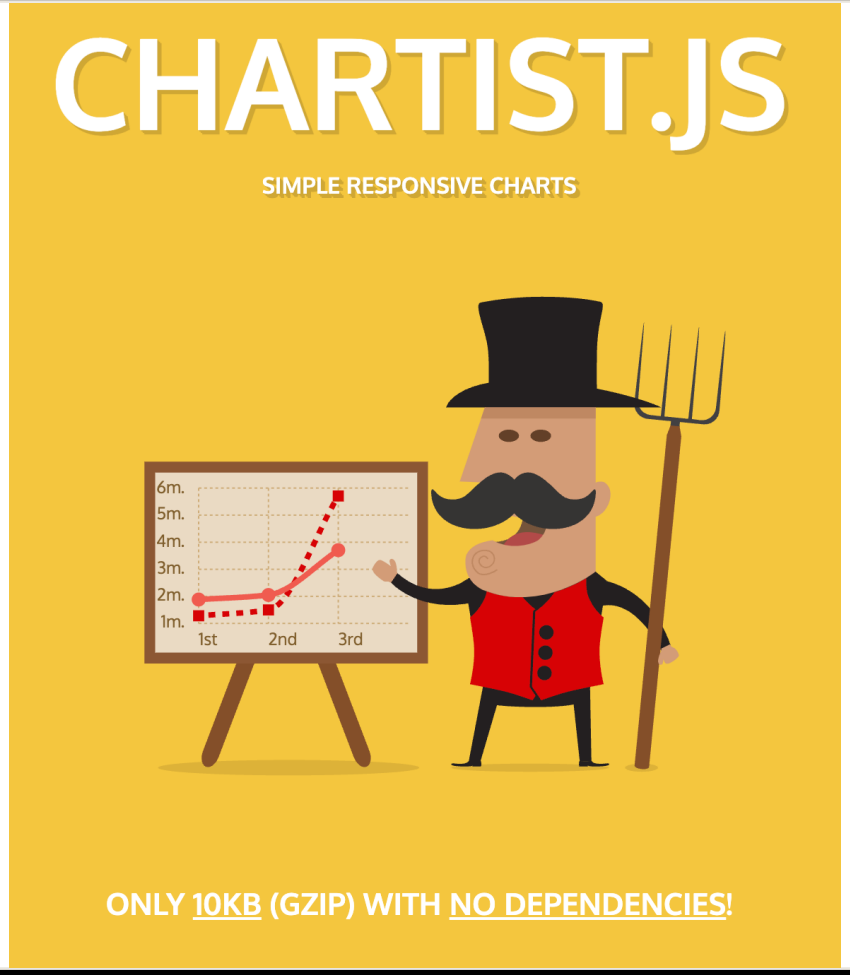
Controls (102)

groupBy*	"id"
annotations	Set object
backgroundConfig	backgroundConfig : { RAW duration : 0 fill : "transparent" }
barPadding	0
baseline	0
confidence	undefined
confidenceConfig	"undefined"
depth	Set number
discrete	x
discreteCutoff	100
groupPadding	5
height	Set number



GOOGLE CHARTS

https://developers.google.com/chart/



CHARTIST.JS

https://gionkunz.github.io/chartist-js/

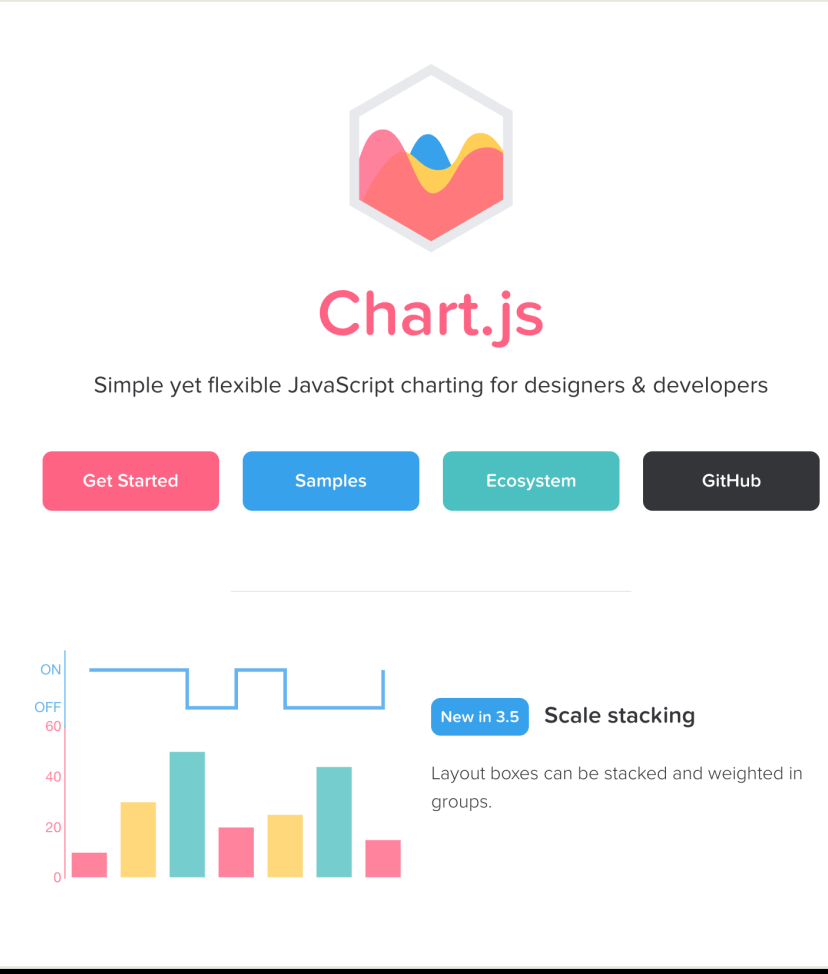


CHART.JS

https://www.chartjs.org/

Revisão

20 horas em 20 min

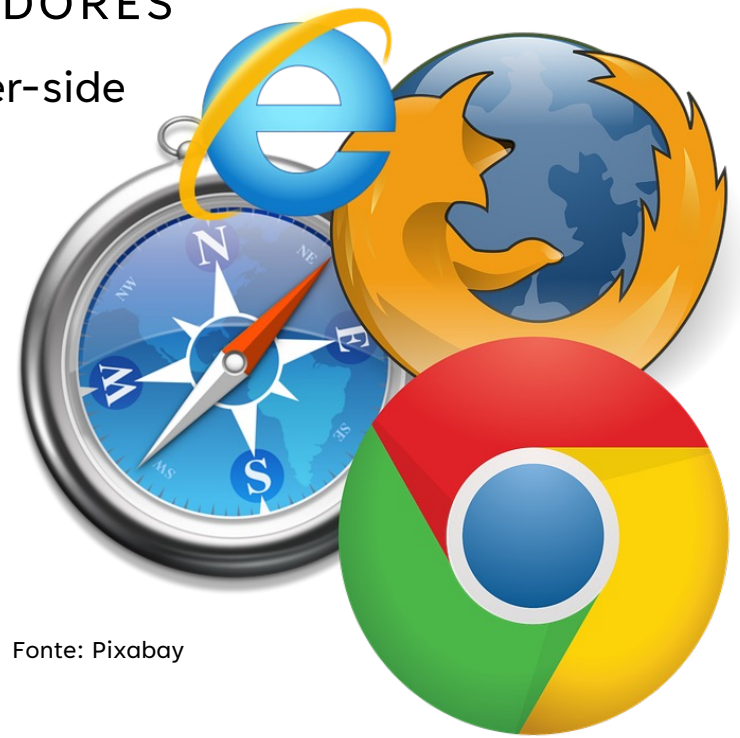
JAVASCRIPT

LINGUAGEM DE PROGRAMAÇÃO

BASEADA EM SCRIPT

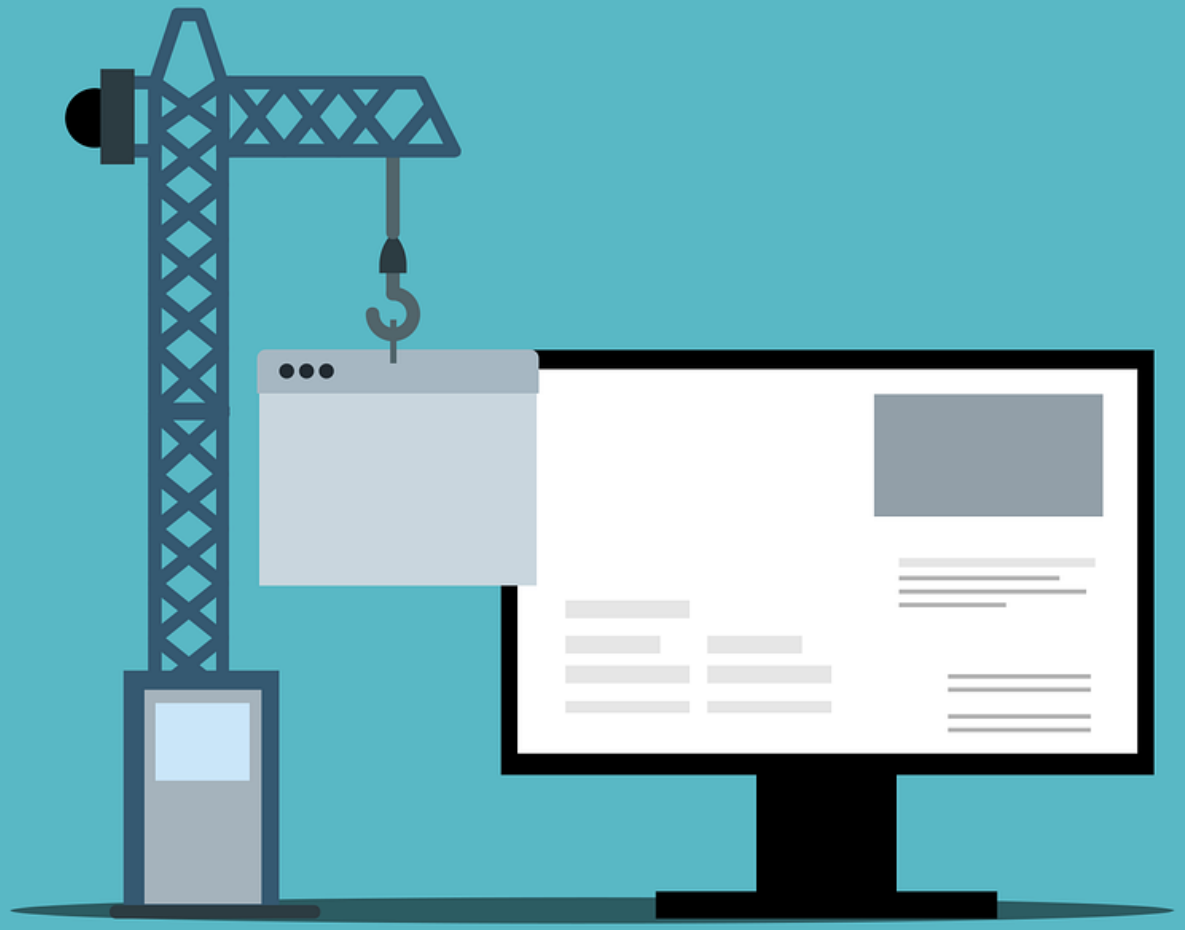
EXECUTADA EM NAVEGADORES

... ou server-side



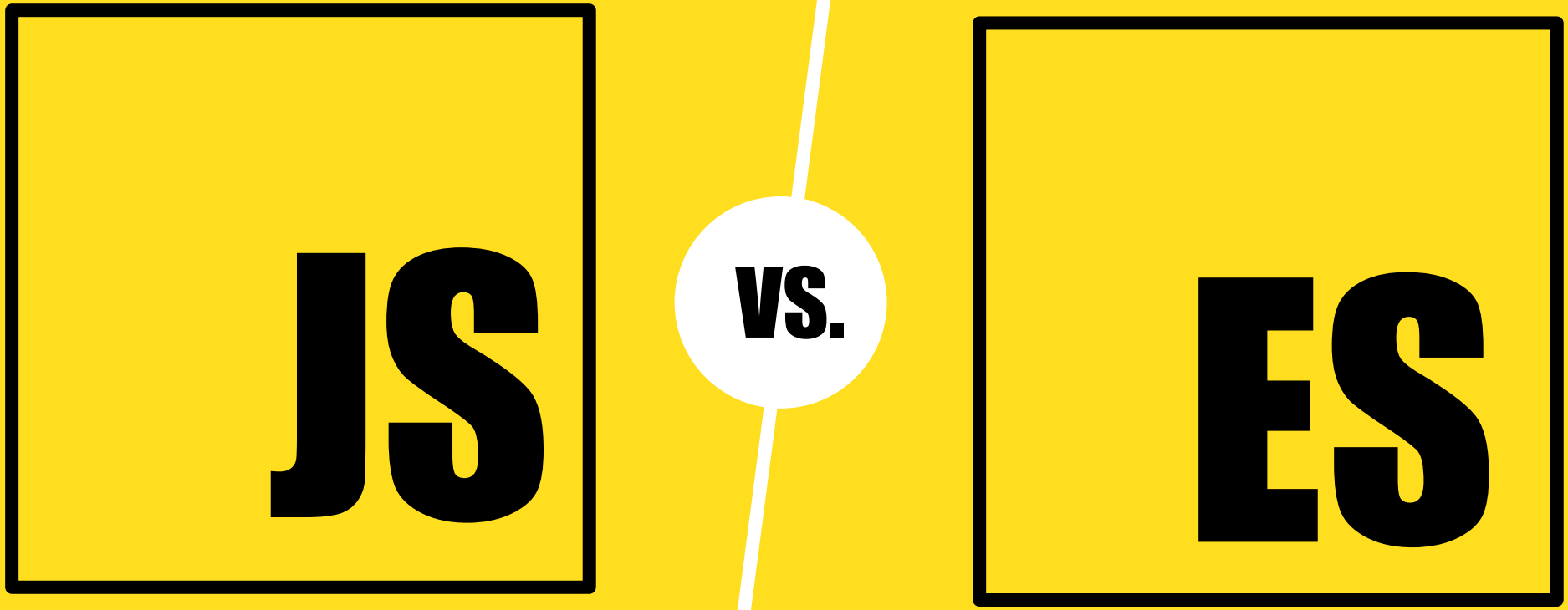
Fonte: Pixabay

SITES MAIS INTERATIVOS

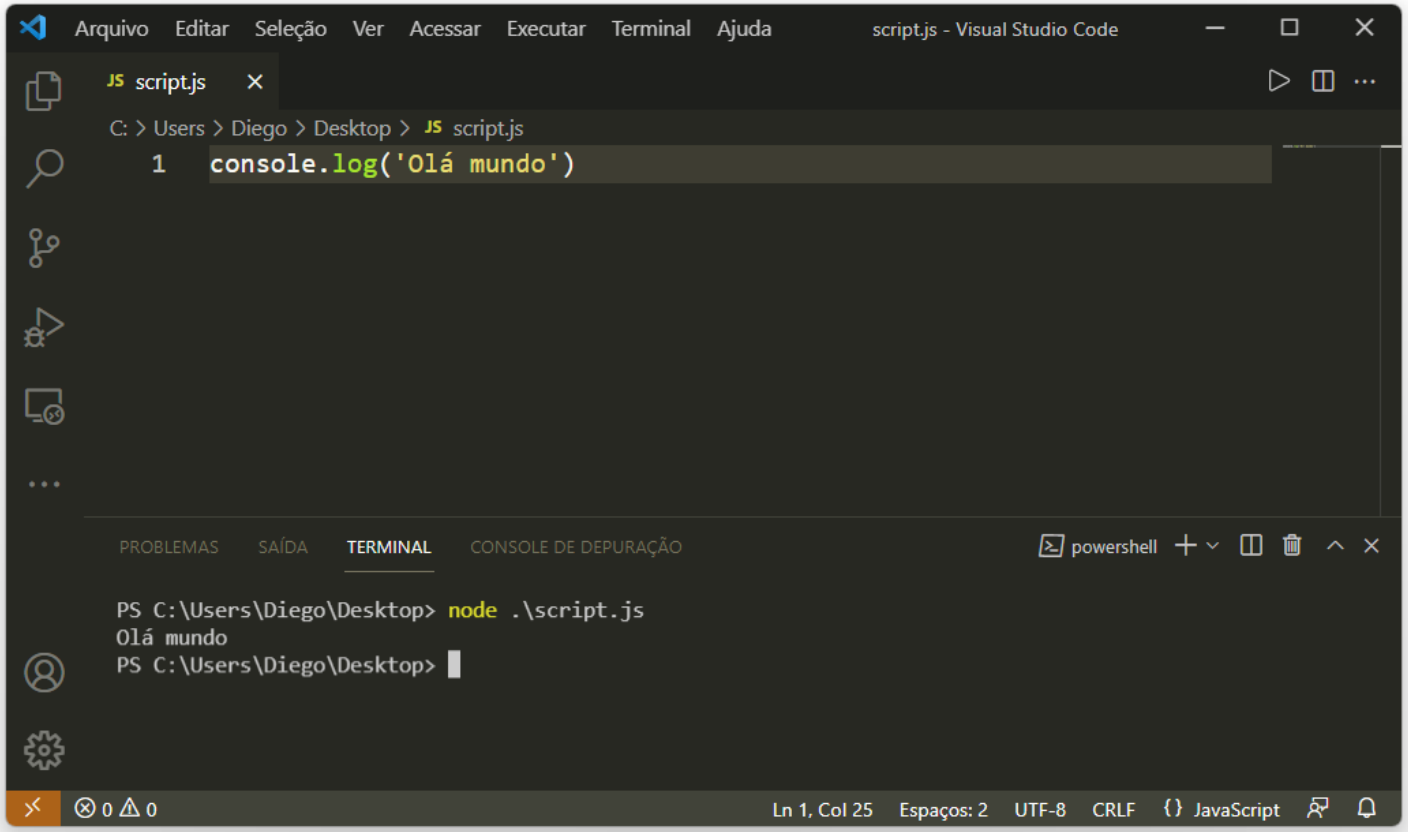


Fonte: Pixabay

ECMASCRIPT



Fonte: próprio autor

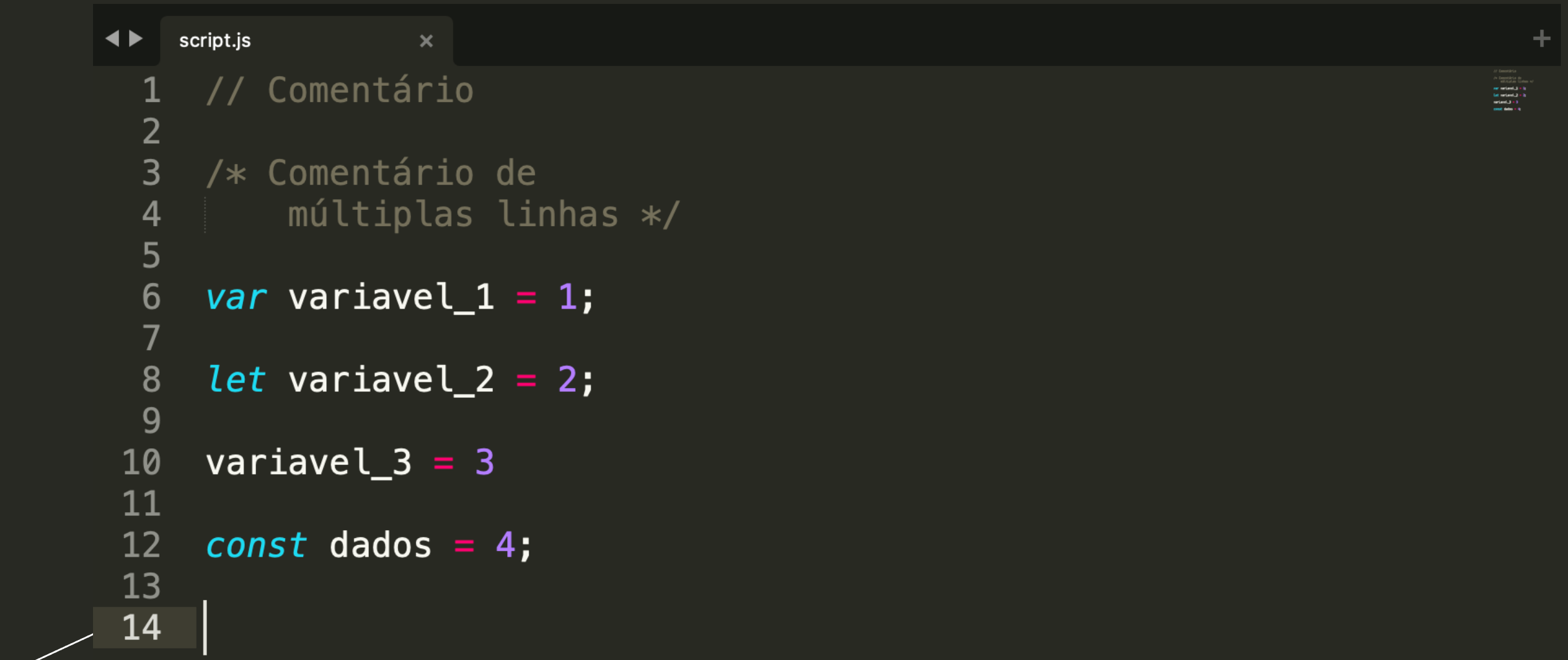


Disponível em https://code.visualstudio.com/docs#vscode

Visual Studio Code

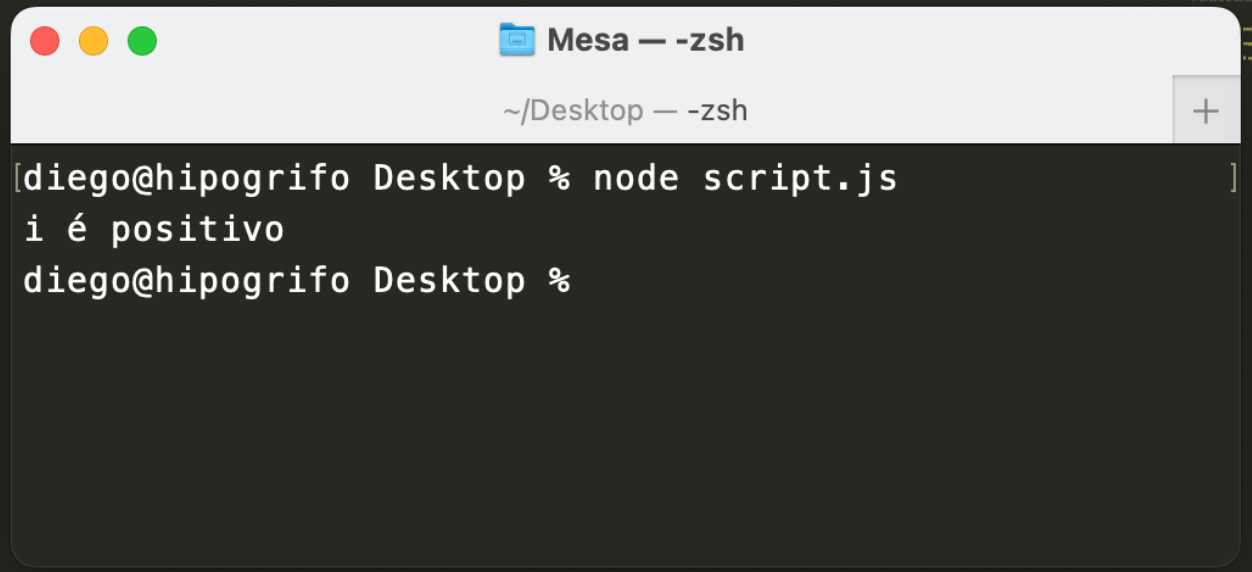
Disponível em https://www.google.com/intl/pt-BR/chrome/

chrome


script.js
1 // Comentário
2
3 /* Comentário de
4 múltiplas linhas */
5
6 var variavel_1 = 1;
7
8 let variavel_2 = 2;
9
10 variavel_3 = 3
11
12 const dados = 4;
13
14

script.js

```
1  // estruturas de controle
2
3  let i = 10;
4
5
6  if(i > 0){
7      console.log("i é positivo");
8  }
9  else if(i < 0){
10     console.log("i é negativo");
11 }
12 else{
13     console.log("i é 0 ou um valor inválido");
14 }
15
16
```



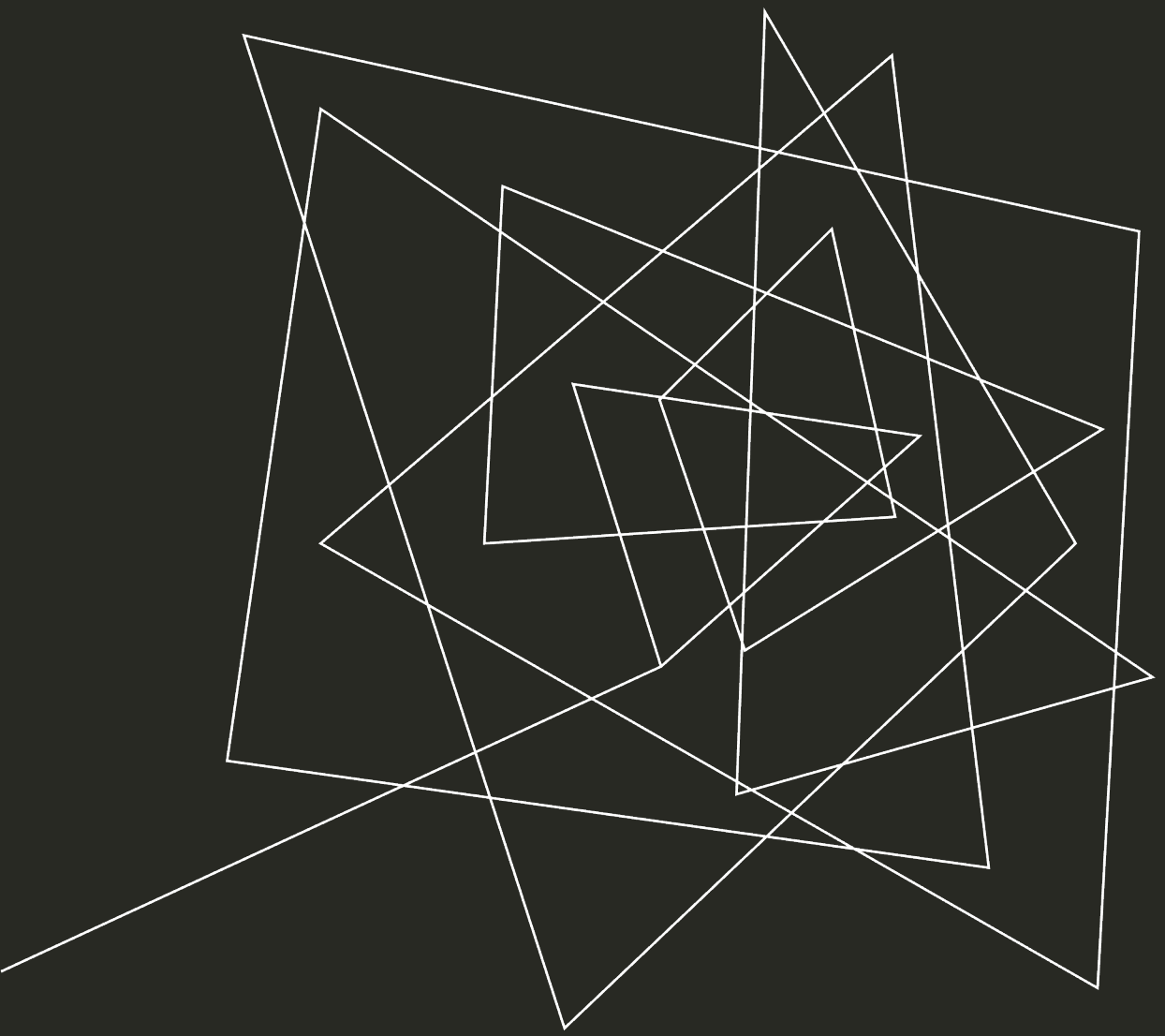
script.js

```
1  // estruturas de repetição
2
3  console.log(" *** WHILE ***");
4
5  let i = 0;
6
7  while(i < 5){
8      console.log(i);
9      i++;
10 }
11
12 console.log(" *** FOR ***");
13
14 let j = [1, 2, 3] // ; são opcionais
15
16 for(item of j){
17     console.log(item);
18 }
19
```

Mesa — -zsh

~/Desktop — -zsh

```
diego@hipogrifo Desktop % node script.js
 *** WHILE ***
0
1
2
3
4
 *** FOR ***
1
2
3
diego@hipogrifo Desktop %
```



Em JavaScript (quase*) tudo é um objeto...

* Exceto tipos primitivos

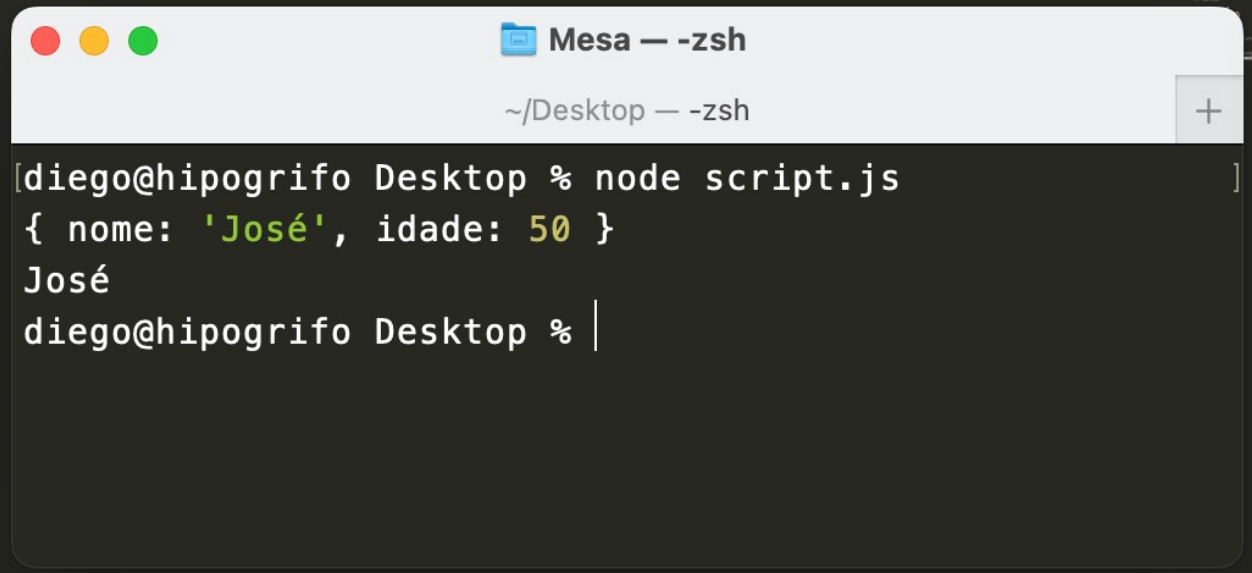
script.js

```
// objetos
let objeto = {}

// propriedades são características
objeto.propriedade

// métodos são ações
objeto.métodos()

// métodos encadeados
objeto.método_1()
      .método_2()
      .método_3()
      .método_4()
```

MÉTODOS SÃO FUNÇÕES

script.js

```
1  // objetos
2
3  let pessoa = {
4      "nome": "José",
5      "idade": 50
6  }
7
8
9  // imprime todo o objeto
10 console.log(pessoa)
11
12 // somente uma propriedade
13 console.log(pessoa.nome)
14
```




script.js
// funções
function soma(a, b){
/* soma o valor a com b */
console.log(a+b);
}
// chamada
soma(2, 2); // 4

FUNÇÕES

JAVASCRIPT FUNCIONAL

Function Declaration

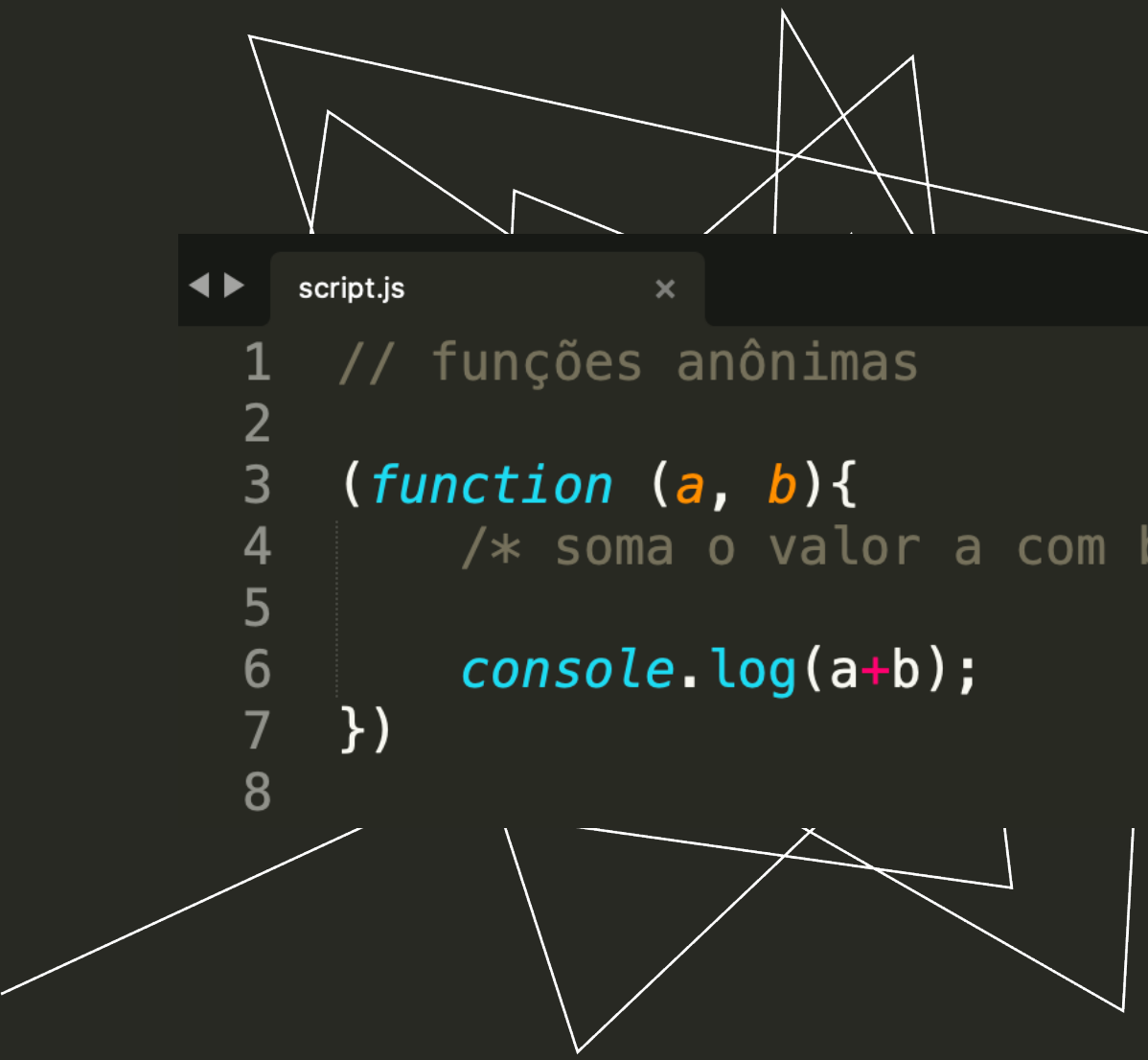
script.js

```
// funções

function soma(a, b){
    /* soma o valor a com b */

    console.log(a+b);
}


// chamada
soma(2, 2);  // 4

```

Function Expression

script.js

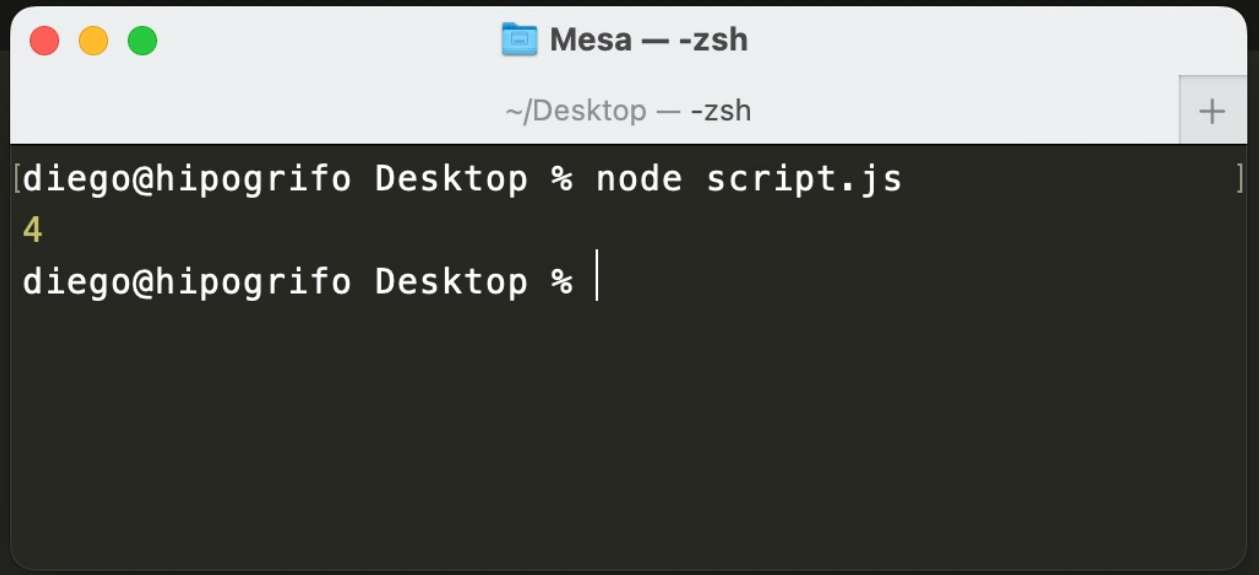
```
// arrow function

const soma = (a, b)=>{
    /* soma o valor a com b */

    console.log(a+b);
}

soma(2, 2);

```



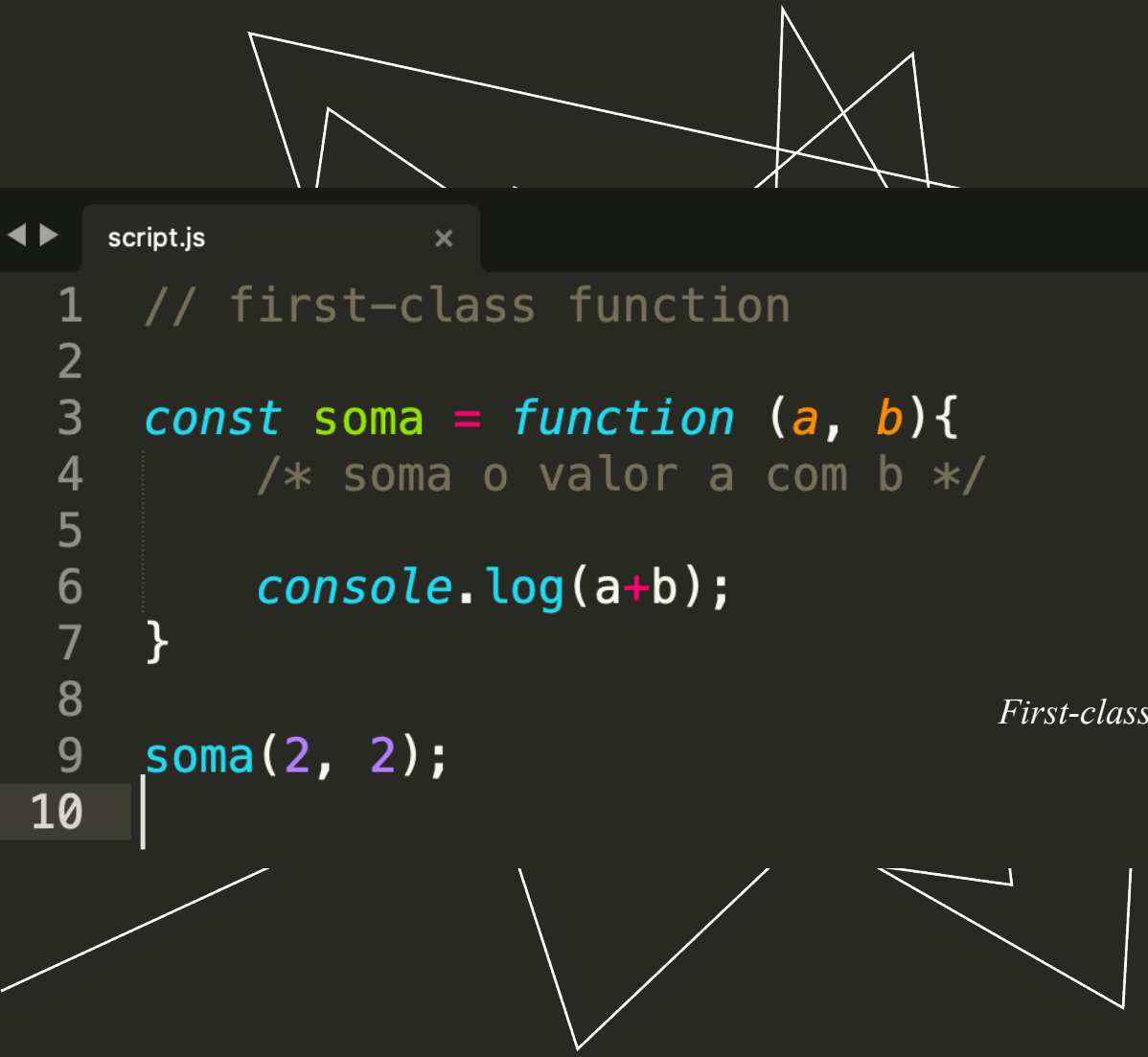
FUNÇÕES ANÔNIMAS

script.js

```
1  // funções anônimas => IIFE
2
3  (function (a, b){
4      /* soma o valor a com b */
5
6      console.log(a+b);
7  })(2, 2);
8
9
10 // IIFE
11 // Immediately Invoked Function Expression
12 // Expressão de Função Imediatamente Invocada
```

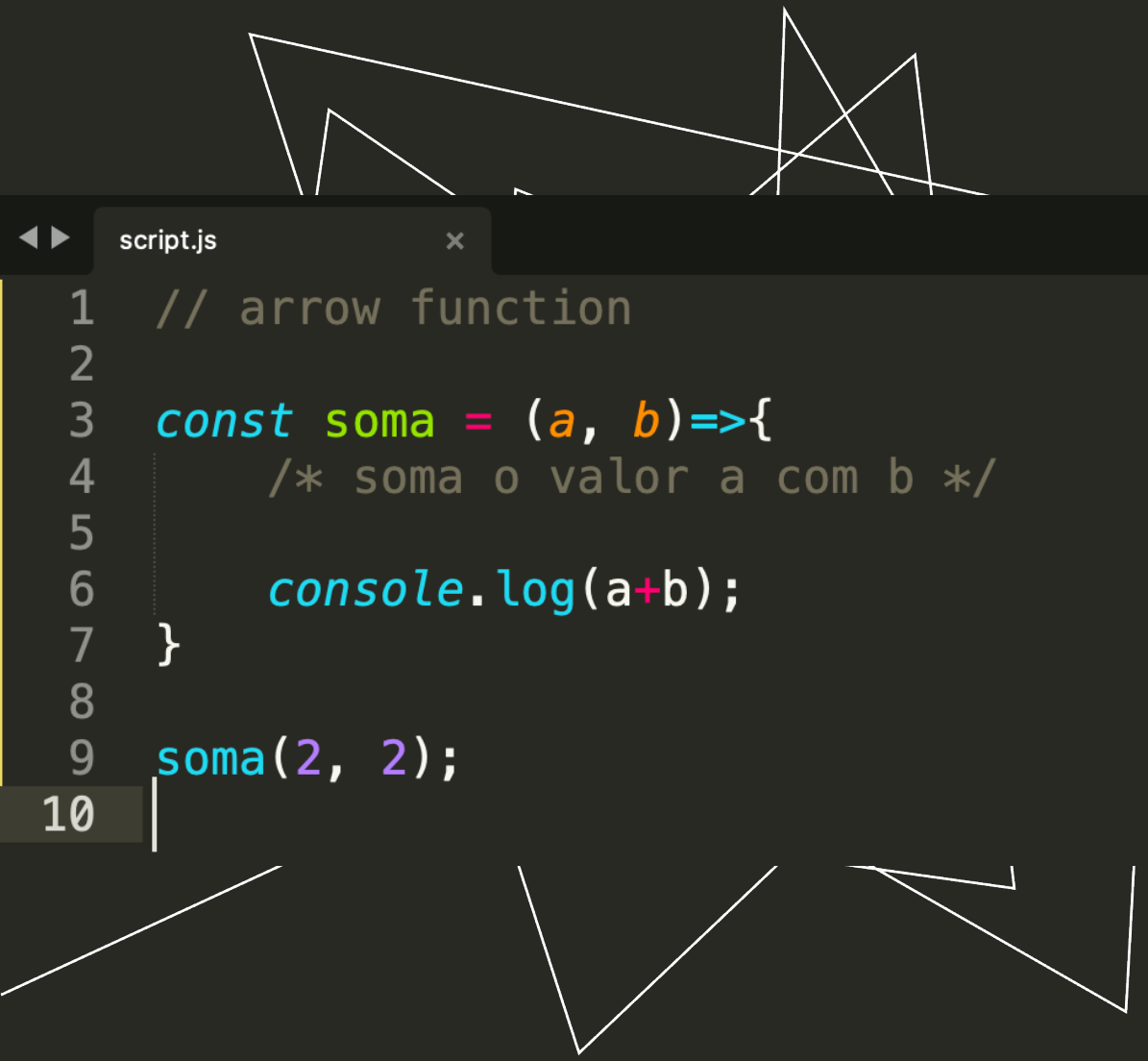


EXECUTANDO



FUNÇÃO DE PRIMEIRA CLASSE

First-class function é uma característica de linguagens de programação que permitem que funções possam ser tratadas como uma variável



ARROW FUNCTION

return

script.js

```
1  // arrow function +simples
2
3  const soma = (a,b) => a+b
4
5  console.log(soma(2, 2))
```

Mesa — -zsh

~/Desktop — -zsh

```
diego@hipogrifo Desktop % node script.js
4
diego@hipogrifo Desktop %
```

ARROW FUNCTION REDUZIDA

script.js

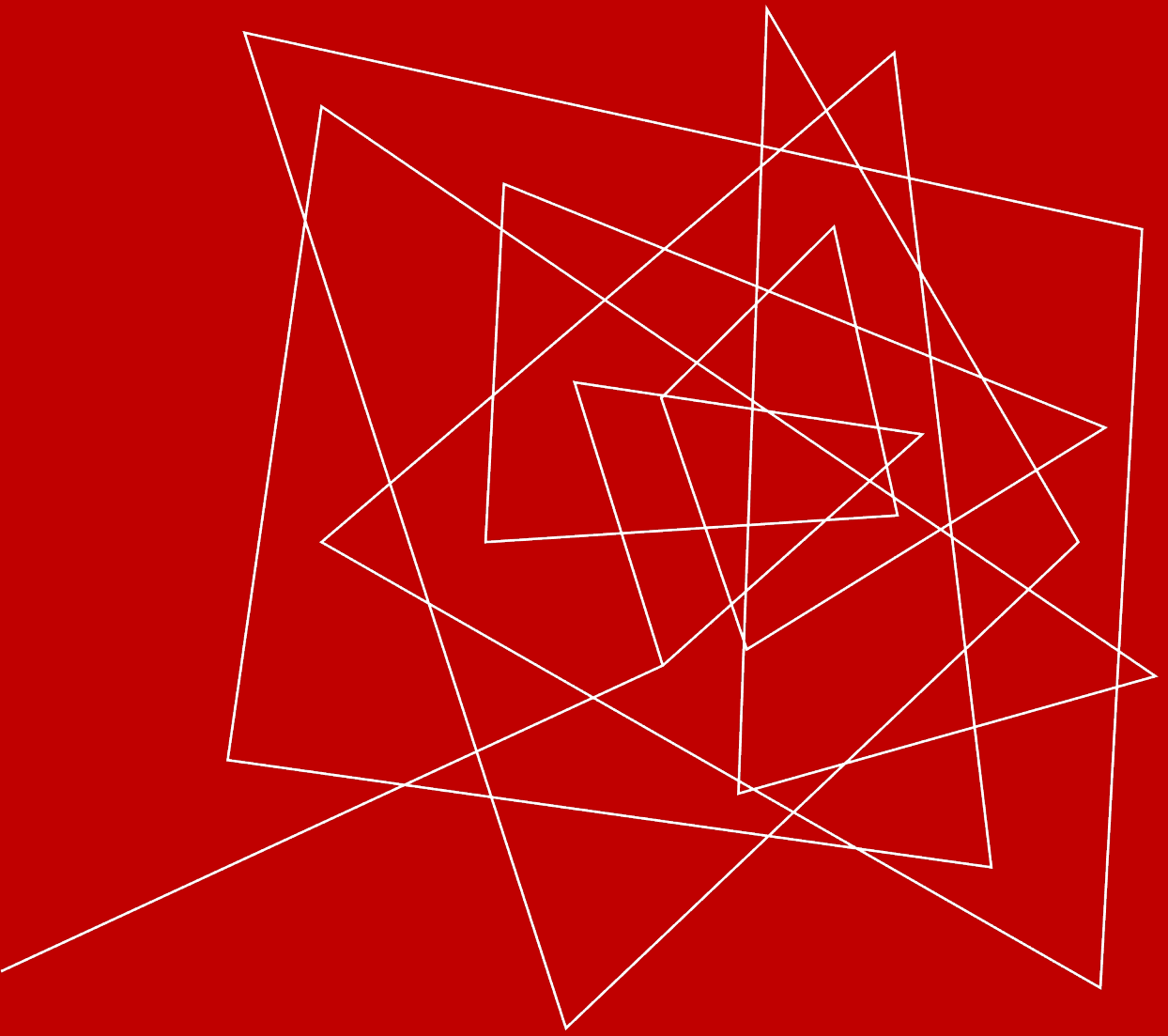
```
1  // função dobro
2  const dobro = i => i*2;
3
4  // array com 5 itens
5  let x = [1, 2, 3, 4, 5];
6
7  // map()
8  x = x.map(dobro);
9
10 console.log(x)
```

Mesa — -zsh

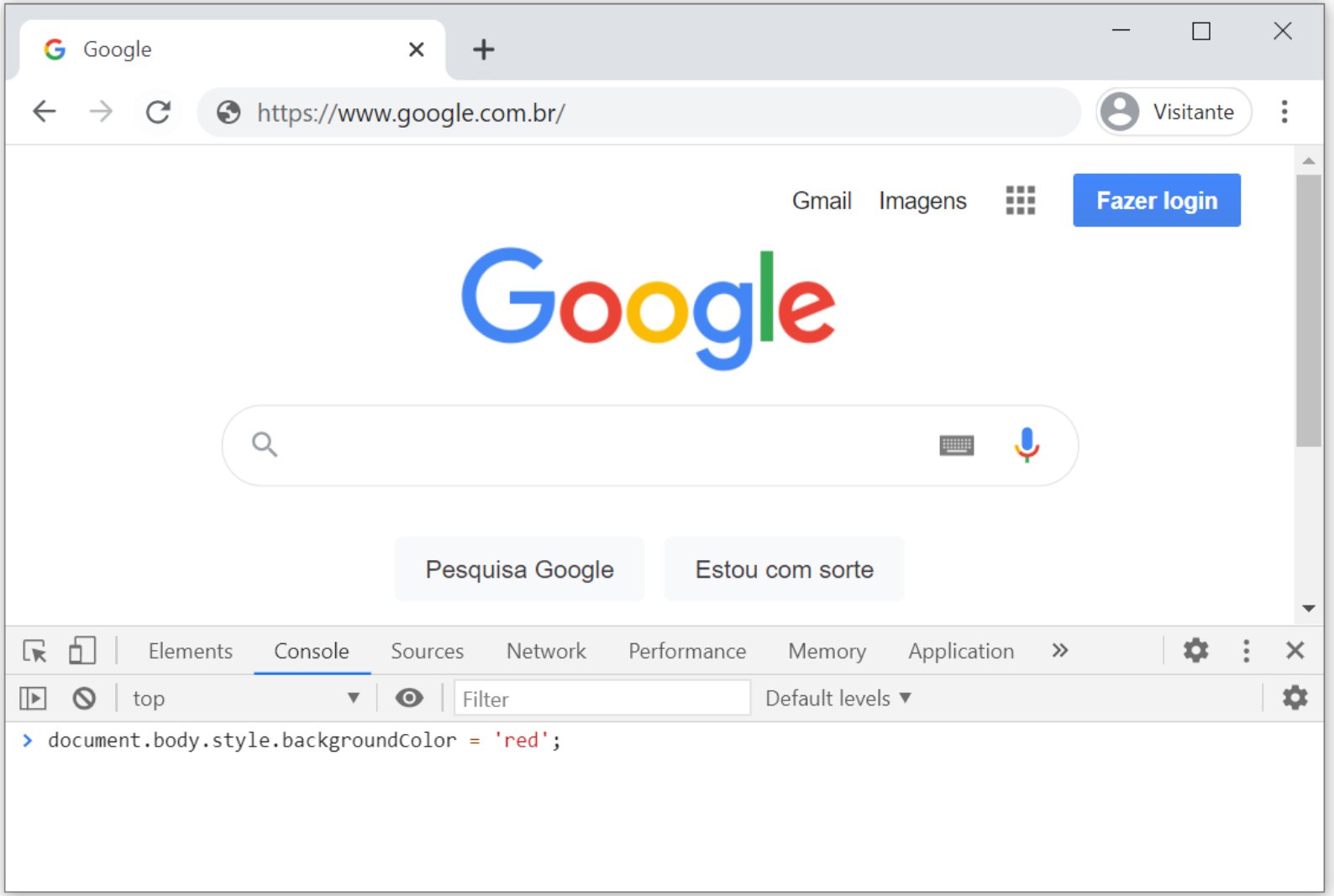
~/Desktop — -zsh

```
diego@hipogrifo Desktop % node script.js
[ 2, 4, 6, 8, 10 ]
diego@hipogrifo Desktop %
```

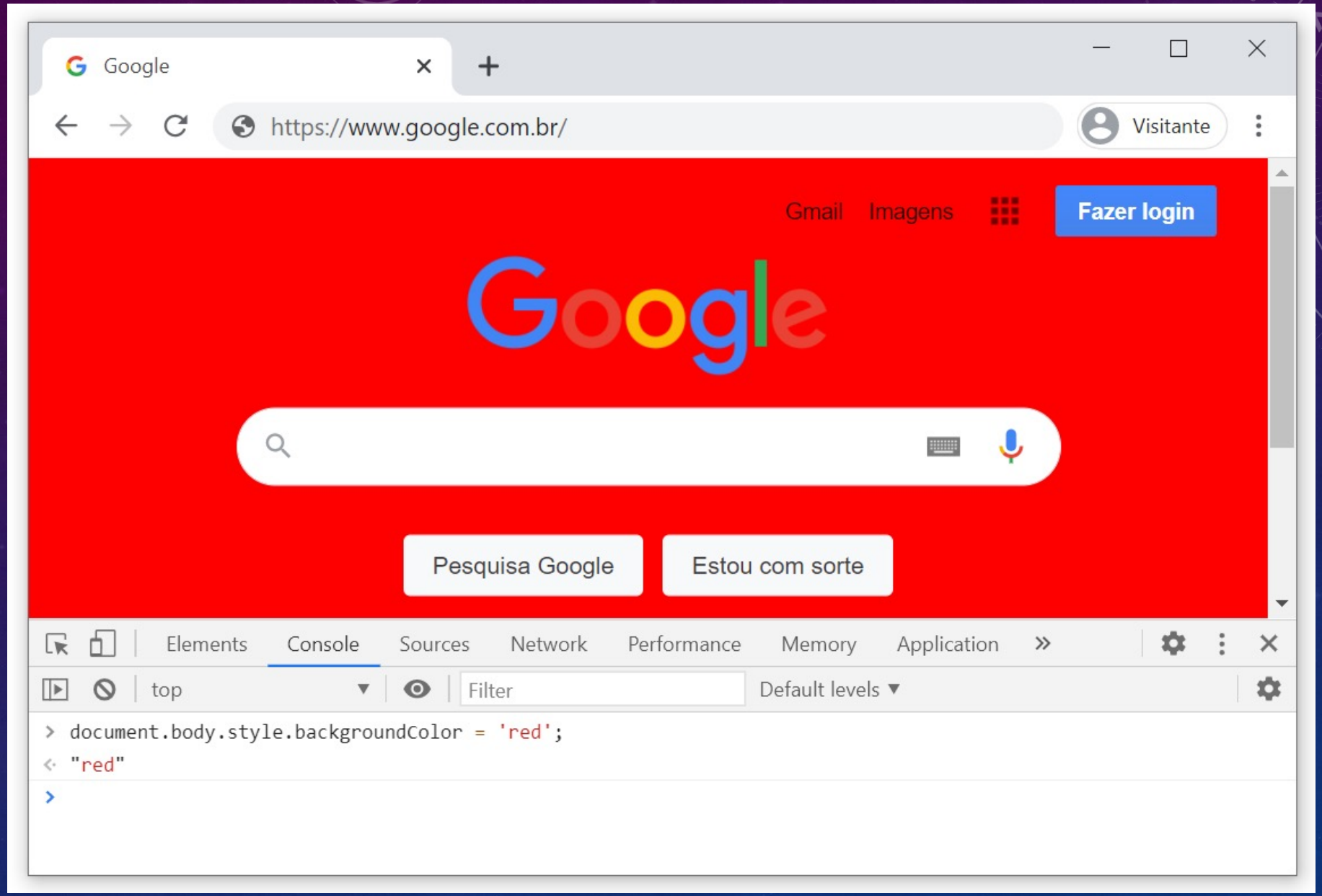
FUNÇÕES DE ARRAYS



MANIPULAÇÃO DO DOM


Google
https://www.google.com.br/
Visitante
Gmail
Imagens
Fazer login
Google
Pesquisa Google
Estou com sorte
Elements
Console
Sources
Network
Performance
Memory
Application
top
Filter
Default levels
> document.body.style.backgroundColor = 'red';

```
document.body.style.backgroundColor = 'red';
```


Google
https://www.google.com.br/
Visitante
Gmail
Imagens
Fazer login
Google
Pesquisa Google
Estou com sorte
Elements
Console
Sources
Network
Performance
Memory
Application
top
Filter
Default levels
> document.body.style.backgroundColor = 'red';
"red"
>

tag <body>

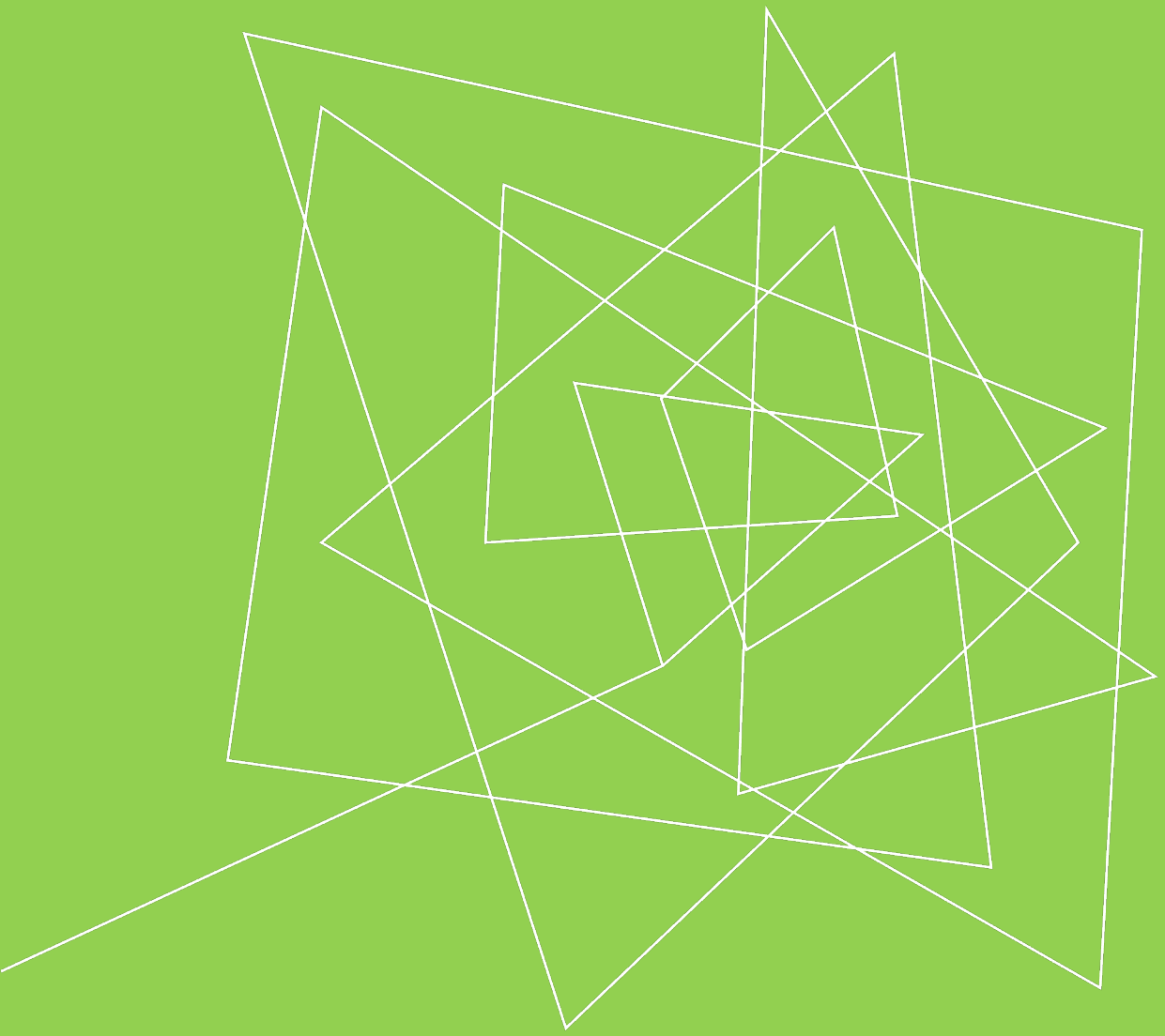
Cor de fundo
(propriedade *background-color*)

```
document.body.style.backgroundColor = 'red';
```

documento

Alterar o estilo

vermelho



D3.JS

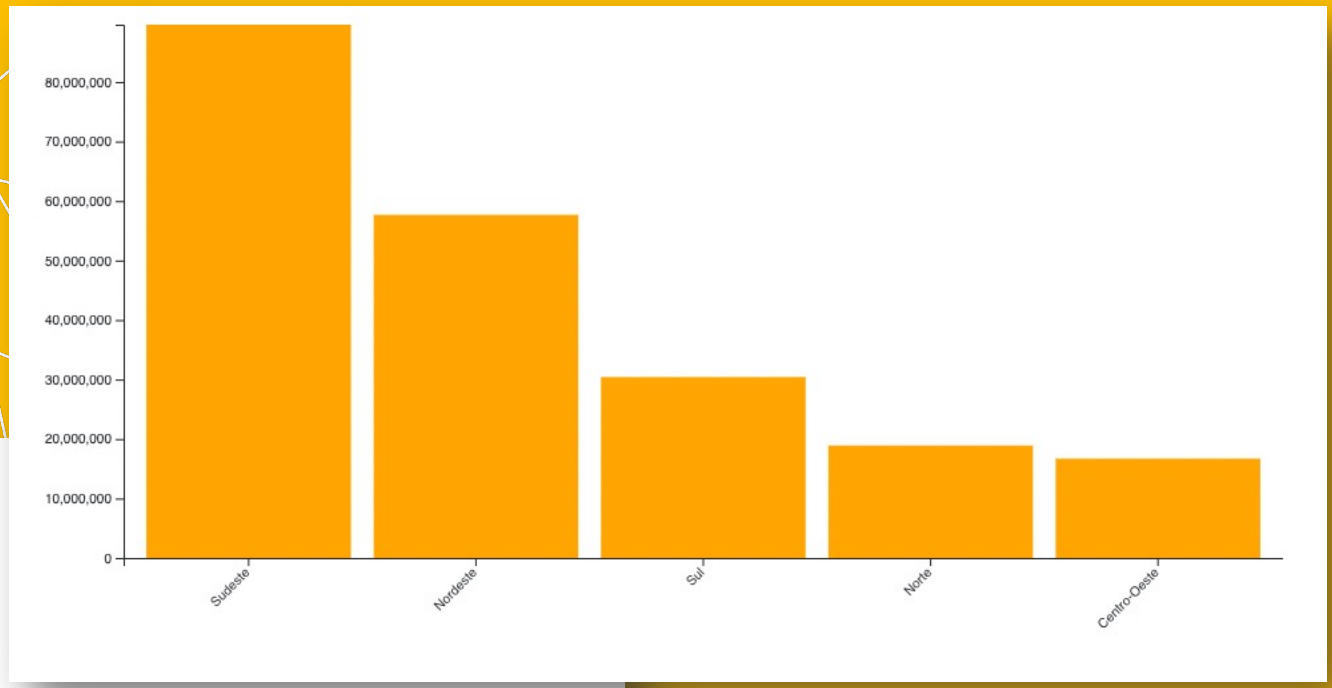
ESTUDO DE CASO

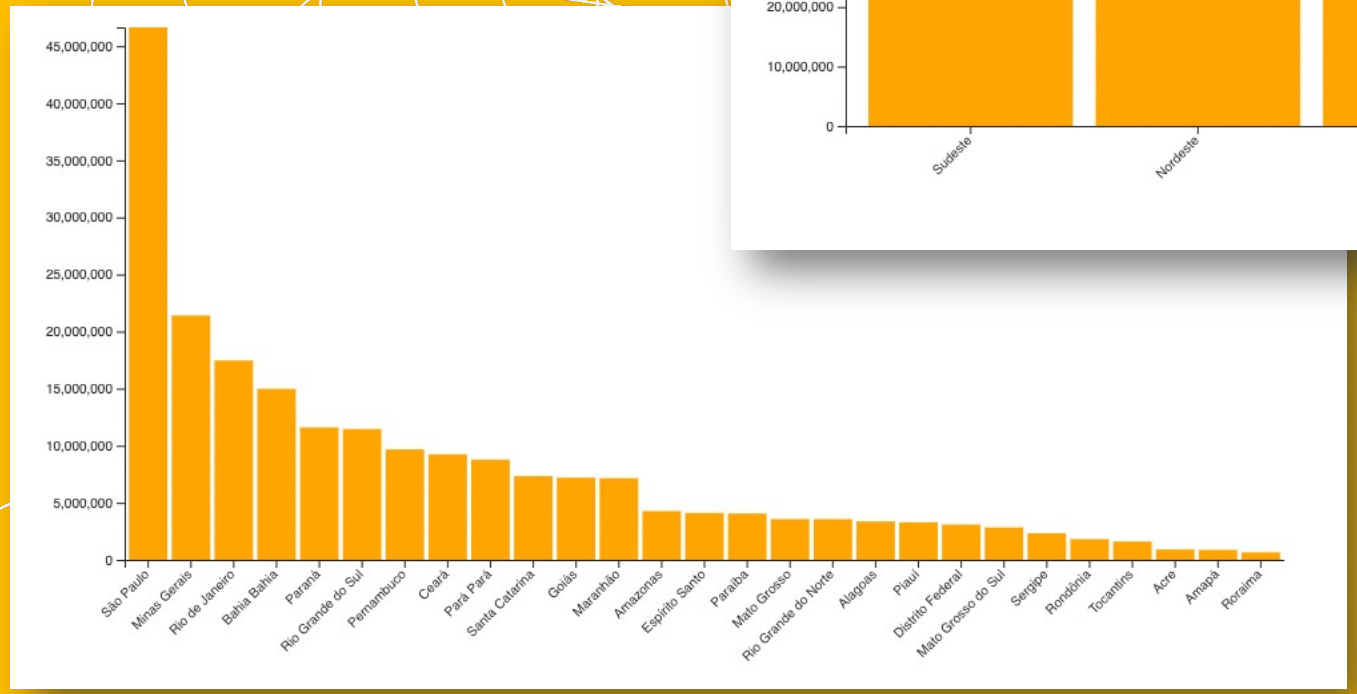
População Brasileira

Código-fonte e slides estão disponíveis em:
https://github.com/dcbmariano/exemplos_d3

index.html

```
<!doctype html>
<html lang="pt-BR">
<head>

  <title>D3js</title>

  <meta charset="utf-8">

  <!-- D3js -->
  <script src="https://d3js.org/d3.v7.js"
                    charset="utf-8"></script>

</head>
<body>

  <div id="grafico"></div>

</body>
</html>
```





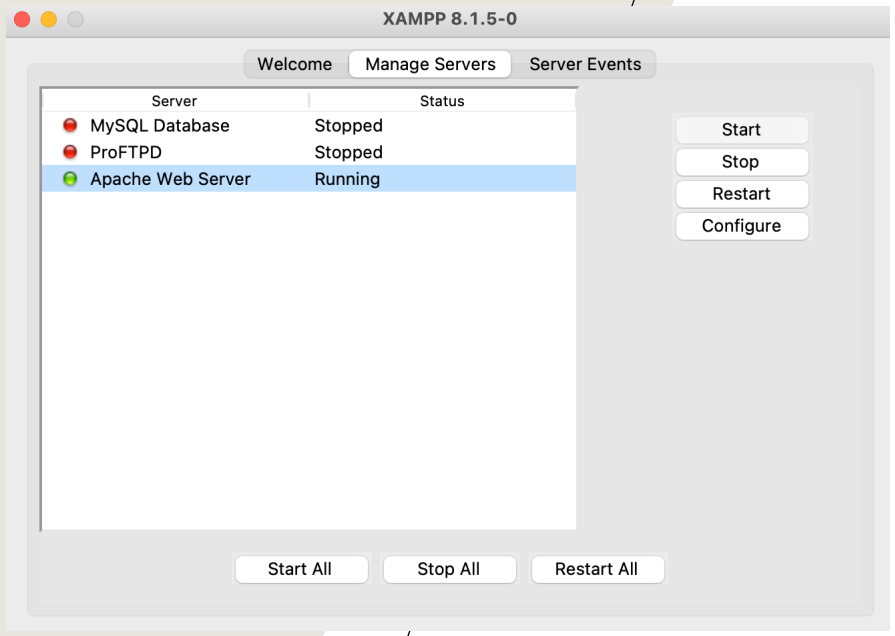
EXEMPLO

População brasileira

https://d3-graph-gallery.com/graph/barplot_basic.html

REQUER SERVIDOR WEB

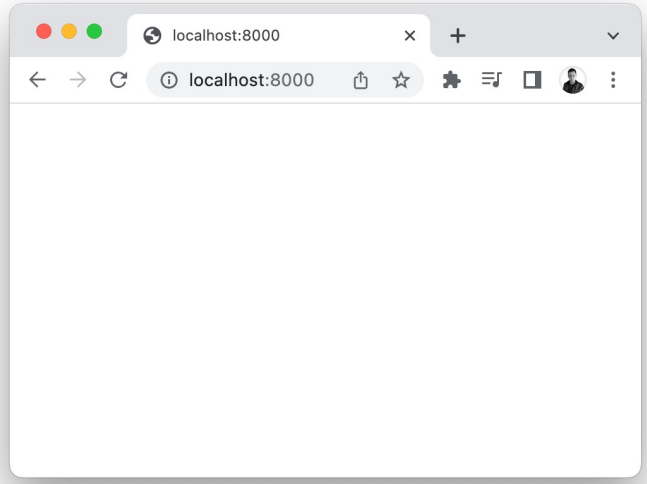
Proibido o acesso direto



SUGESTÕES:

Apache (XAMPP)

- Python
- Node.js
- PHP



php -S localhost:8000

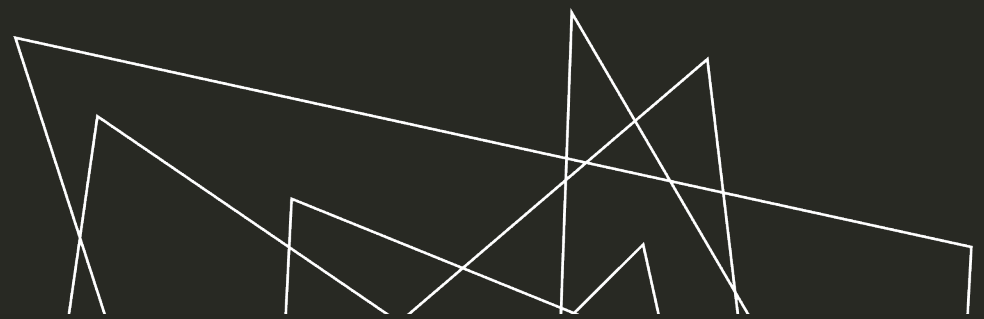
populacao_por_estado.csv | populacao_por_regiao.json

```csv
id,nome,populacao
1,São Paulo,46649132
2,Minas Gerais,21411923
3,Rio de Janeiro,17463349
4,Bahia Bahia,14985284
5,Paraná,11597484
6,Rio Grande do Sul,11466630
7,Pernambuco,9674793
8,Ceará,9240580
9,Pará Pará,8777124
10,Santa Catarina,7338473
11,Goiás,7206589
12,Maranhão,7153262
13,Amazonas,4269995
14,Espírito Santo,4108508
15,Paraíba,4059905
16,Mato Grosso,3567234
17,Rio Grande do Norte,3560903
18,Alagoas,3365351
19,Piauí,3289290
20,Distrito Federal,3094325
21,Mato Grosso do Sul,2839188
22,Sergipe,2338474
23,Rondônia,1815278
24,Tocantins,1607363
25,Acre,906876
26,Amapá,877613
27,Roraima,652713
```

populacao_por_estado.csv | populacao_por_regiao.json

```json
[
    {
        "nome": "Sudeste",
        "populacao":89632912,
        "percentual":42.04
    },
    {
        "nome": "Nordeste",
        "populacao":57667842,
        "percentual":27.09
    },
    {
        "nome": "Sul",
        "populacao":30402587,
        "percentual":14.26
    },
    {
        "nome": "Norte",
        "populacao":18906962,
        "percentual":8.82
    },
    {
        "nome": "Centro-Oeste",
        "populacao":16707336,
        "percentual":7.79
    }
]
```

FONTE: IBGE (população estimada para 2021 / censo 2010)



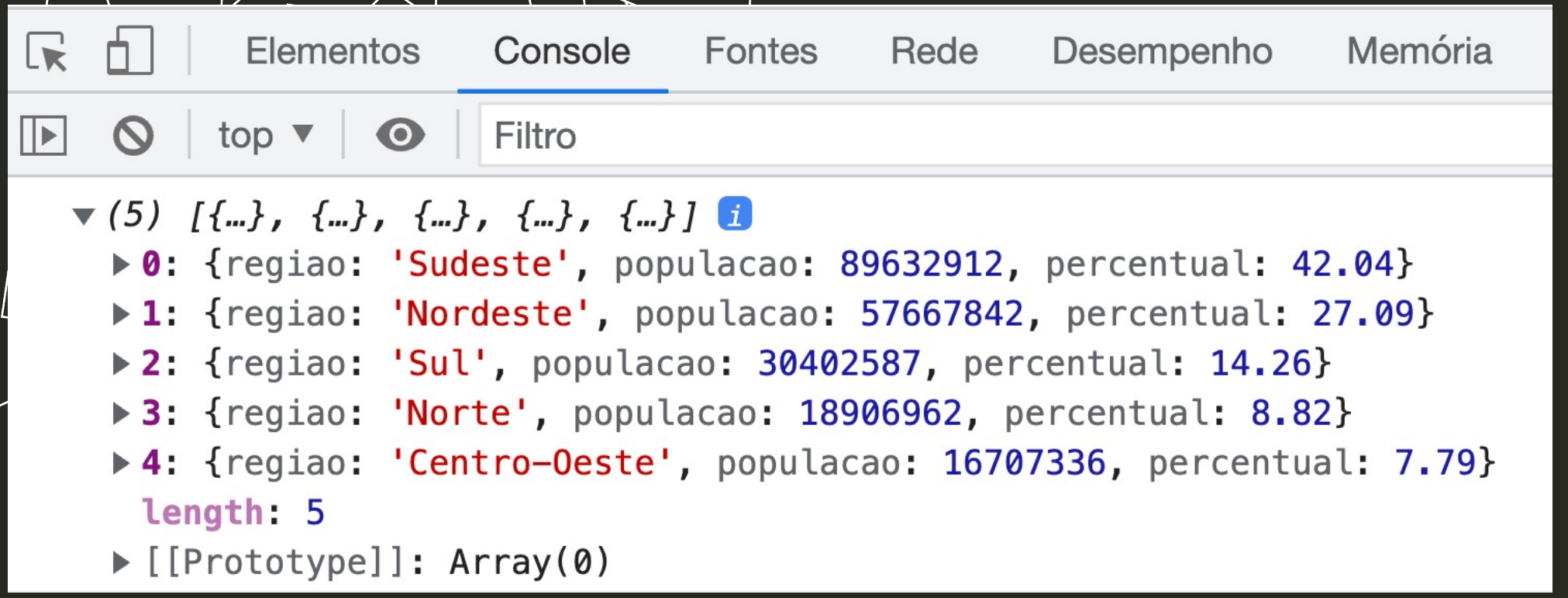
1

Users > diego > Desktop > JS script.js > ...

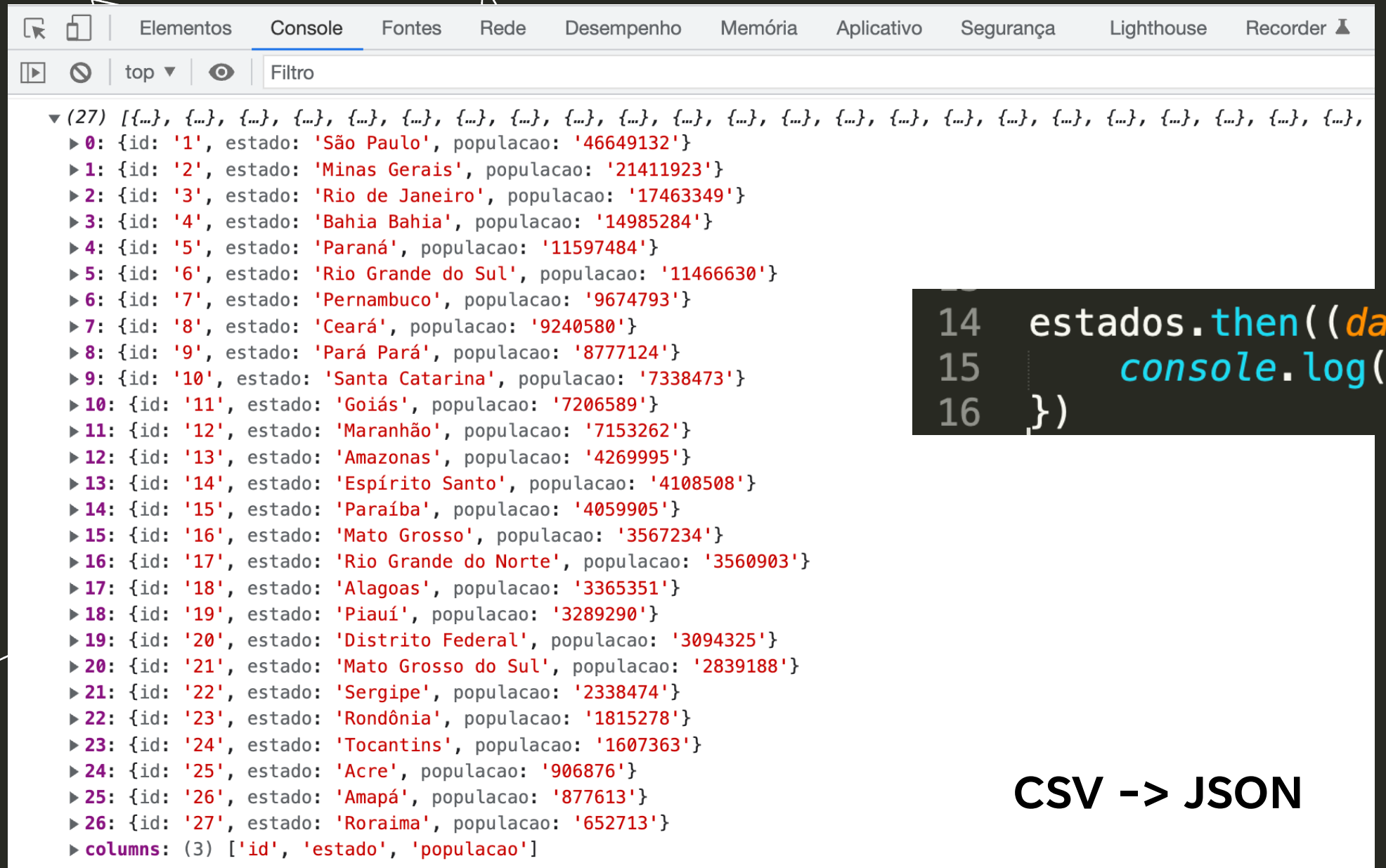
```
1   // 1) Lendo dados ----------------------------------------
2   // JSON
3   const regioes = d3.json("populacao_por_regiao.json");
4
5   // CSV
6   const estados = d3.csv("populacao_por_estado.csv");
7
```

JSON

```
10  regioes.then((dados)=>{
11      console.log(dados);
12  })
```



CSV




```
14  estados.then((dados)=>{
15      console.log(dados);
16  })
```

CSV -> JSON

```
8   // 2) medidas padronizadas ----------------------------------
9   const margem = {
10      superior:10, inferior:90,
11      direita:0, esquerda:90
12  }
13  const altura = 500 - margem.superior - margem.inferior;
14  const largura = 960 - margem.esquerda - margem.direita;
```

2

3

```
16  // 3) adiciona um objeto svg ------------------------------------
17  const svg = d3.select("#grafico")
18    .append("svg")
19      .attr("width", largura + margem.esquerda + margem.direita)
20      .attr("height", altura + margem.superior + margem.inferior)
21    .append("g")
22      .attr("transform", `translate(${margem.esquerda},${margem.superior})`);
```

```
<div id="grafico">
  <svg width="960" height="500">…</svg> == $0
</div>
```

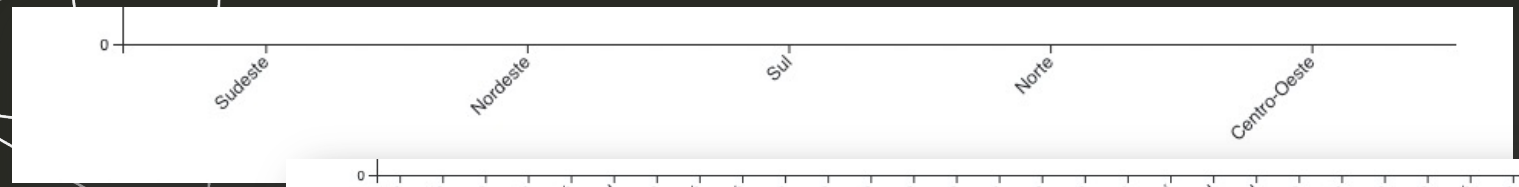
```
24  // 4) para cada valor ----------------------
25  regioes.then((data)=>{
26
27  // 5) Configura Eixo X ---------------------
28  const x = d3.scaleBand()
29    .range([ 0, largura ])
30    .domain(data.map(d => d.nome))
31    .padding(0.1);
32
33  // 6) Plota Eixo X -------------------------
34  svg.append("g")
35    .attr("transform", `translate(0, ${altura})`)
36    .call(d3.axisBottom(x))
37    .selectAll("text")
38      .attr("transform", "translate(-10,0)rotate(-45)")
39      .style("text-anchor", "end");
40
41  // 7) Configura Eixo Y ---------------------
42  const y = d3.scaleLinear()
43    .domain([0, d3.max(data.map(d => +d.populacao))])
44    .range([ altura, 0]);
45
46  // 8) Plota Eixo Y -------------------------
47  svg.append("g")
48    .call(d3.axisLeft(y));
49
50  // 9) cria cada barra ----------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll("barra")
54    .data(data)
55    .join("rect")
56      .attr("x", d => x(d.nome))
57      .attr("y", d => y(d.populacao))
58      .attr("width", x.bandwidth())
59      .attr("height", d => altura - y(d.populacao))
60      .attr("fill", "orange")
61
62  })
```

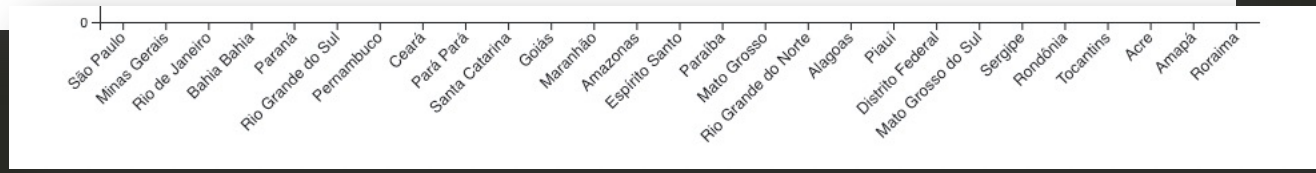
4

5

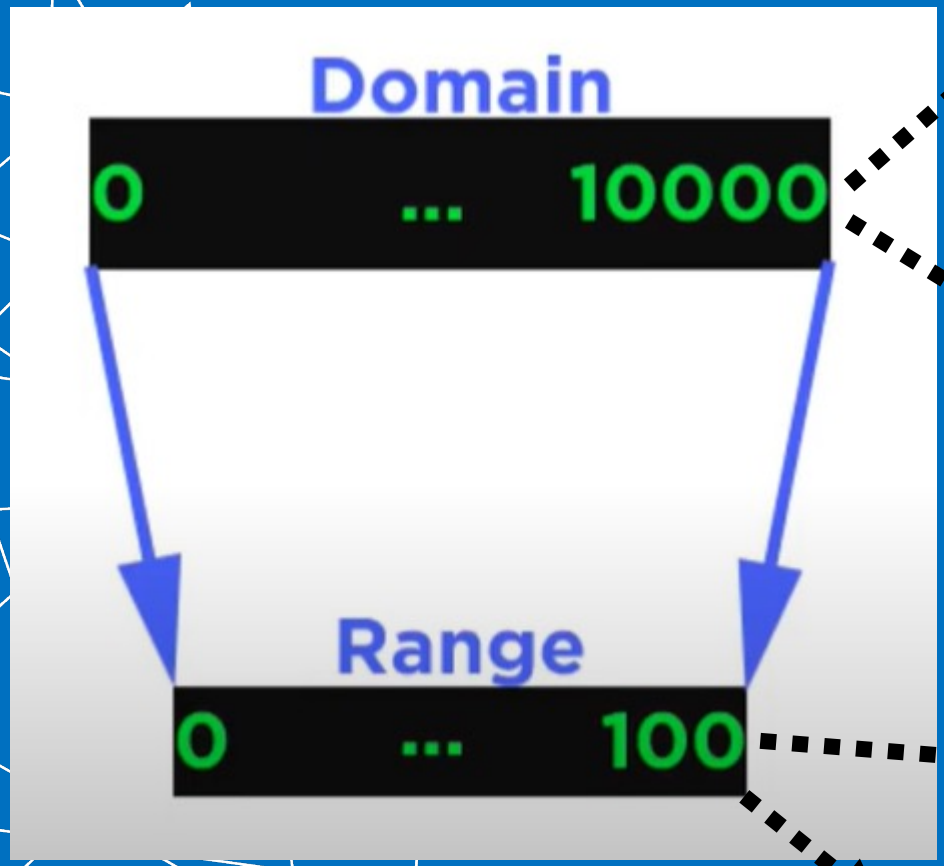
```
27  // 5) Configura Eixo X ----------------------------------
28  const x = d3.scaleBand()
29      .range([ 0, largura ])
30      .domain(data.map(d => d.nome))
31      .padding(0.1);
```

d3.scaleBand() em D3.js é usada para construir uma nova escala de banda com o domínio especificado





https://www.geeksforgeeks.org/d3-js-d3-scaleband-function/

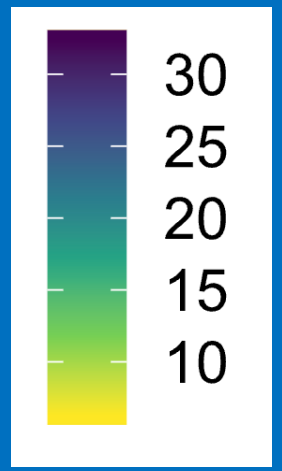


ARRAY DE ITENS

MAIOR VALOR NOS DADOS

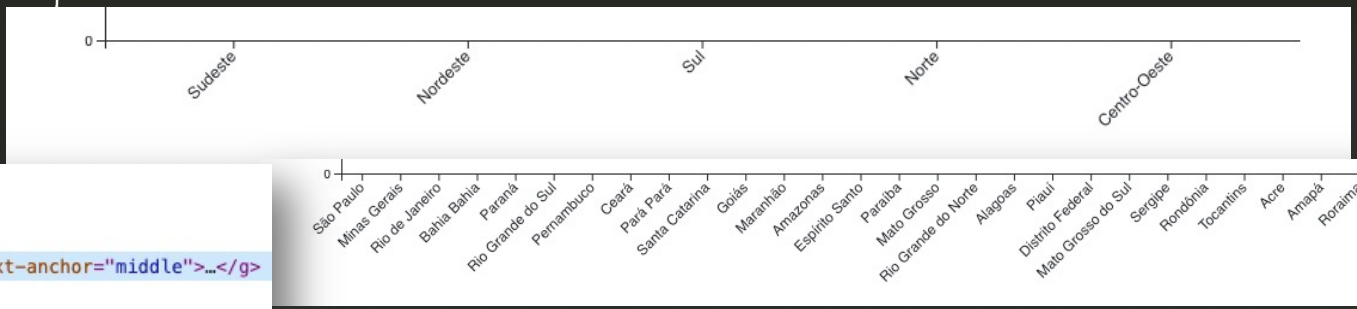
VALOR EM PIXELS

(LARGURA OU ALTURA)



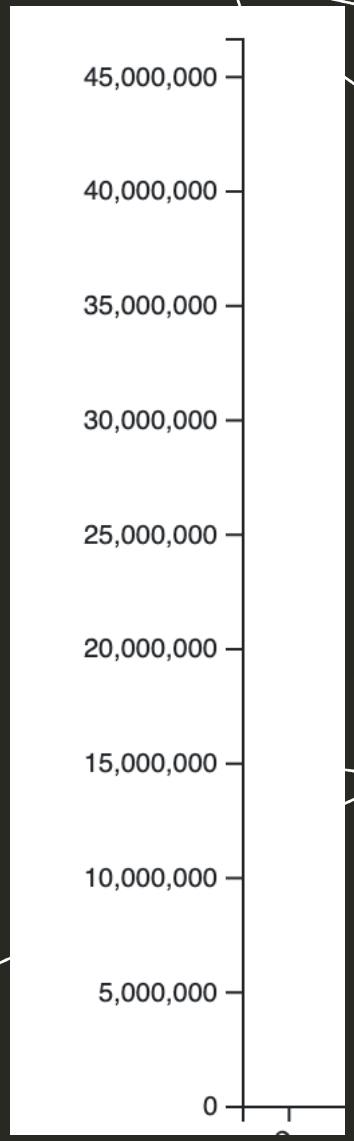
6

```
33  // 6) Plota Eixo X ----------------------------------------
34  svg.append("g")
35    .attr("transform", `translate(0, ${altura})`)
36    .call(d3.axisBottom(x))
37    .selectAll("text").attr('font-size','12px')
38      .attr("transform", "translate(-10,0)rotate(-45)")
39      .style("text-anchor", "end");
```

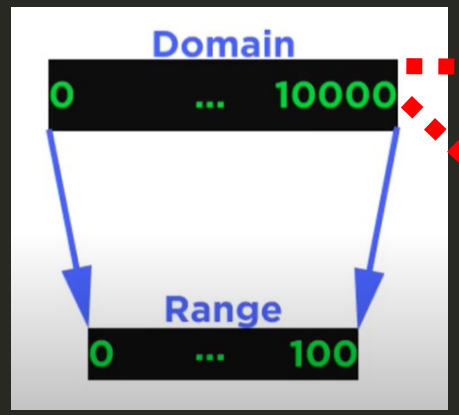



```
▼<div id="grafico">
  ▼<svg width="960" height="500">
    ▼<g transform="translate(90,10)">
      ▶<g transform="translate(0, 400)" fill="none" font-size="10" font-family="sans-serif" text-anchor="middle">…</g>
      ▶<g fill="none" font-size="10" font-family="sans-serif" text-anchor="end">…</g>
      ▶<g class="barras">…</g>
      </g>
    </svg>
  </div>
```

7

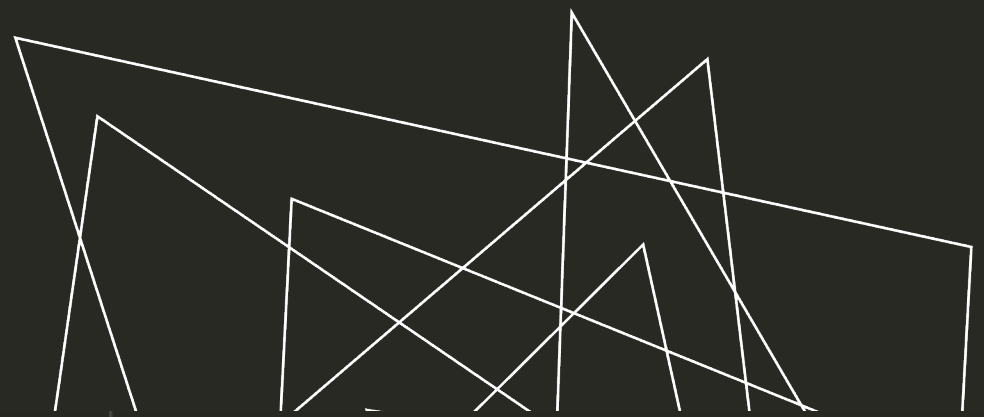



```
41  // 7) Configura Eixo Y ----------------------------------
42  const y = d3.scaleLinear()
43      .domain([0, d3.max(data.map(d => +d.populacao))])
44      .range([ altura, 0]);
```



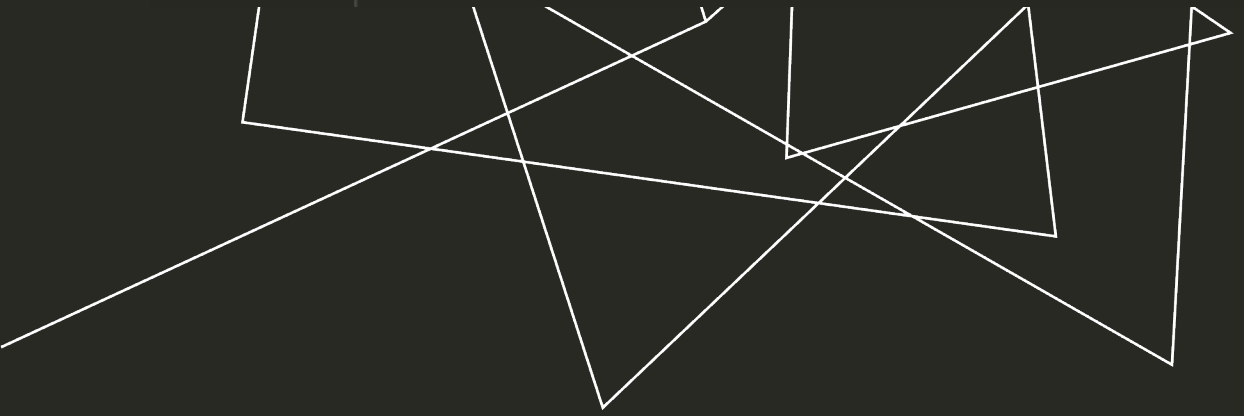
ARRAY DE ITENS

MAIOR VALOR NOS DADOS



8

```
46  // 8) Plota Eixo Y ------------------------------------
47  svg.append("g")
48     .call(d3.axisLeft(y));
```



9

```
50  // 9) cria cada barra ----------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```

9.1

```
50  // 9) cria cada barra ----------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```

```
▼<div id="grafico">
  ▼<svg width="960" height="500">
    ▼<g transform="translate(90,10)">
      ▶<g transform="translate(0, 400)" fill="none" font-size="10"
       font-family="sans-serif" text-anchor="middle">…</g>
      ▶<g fill="none" font-size="10" font-family="sans-serif" text-
       anchor="end">…</g>
      ▶<g class="barras">…</g> == $0
     </g>
   </svg>
 </div>
```

9.2

```
50  // 9) cria cada barra ----------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("
60      .attr("
61      .attr("
```

```
24  // 4) para cada valor -----
25  estados.then((data)=>{
26
```

Quando você vê `selectAll()` seguido por `.data()` estamos selecionando todos os elementos correspondentes, para cada um que existe, associamos um item do array de dados a ele. O uso de .data() retorna o que é chamado de seleção de atualização: contém elementos existentes (se houver) com os dados recém-fornecidos vinculados a esses itens existentes.

https://stackoverflow.com/questions/69819222/what-is-the-difference-between-append-and-join-in-d3-js

9.3

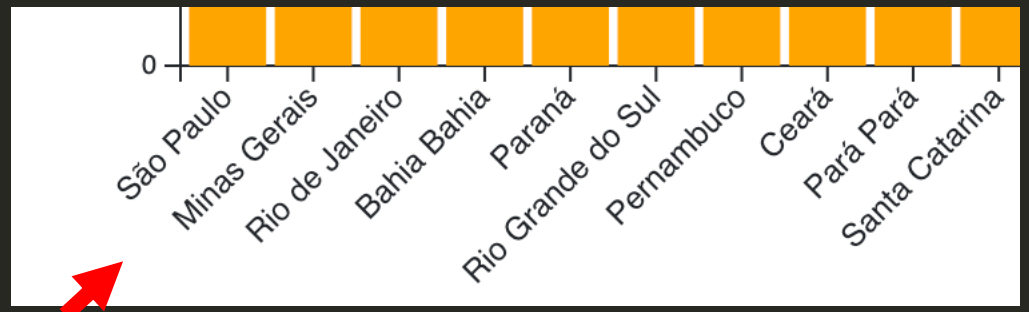
```
50  // 9) cria cada barra ----------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```

```
54    .data(data).enter()
55    .append("rect")
```

Mesmo resultado

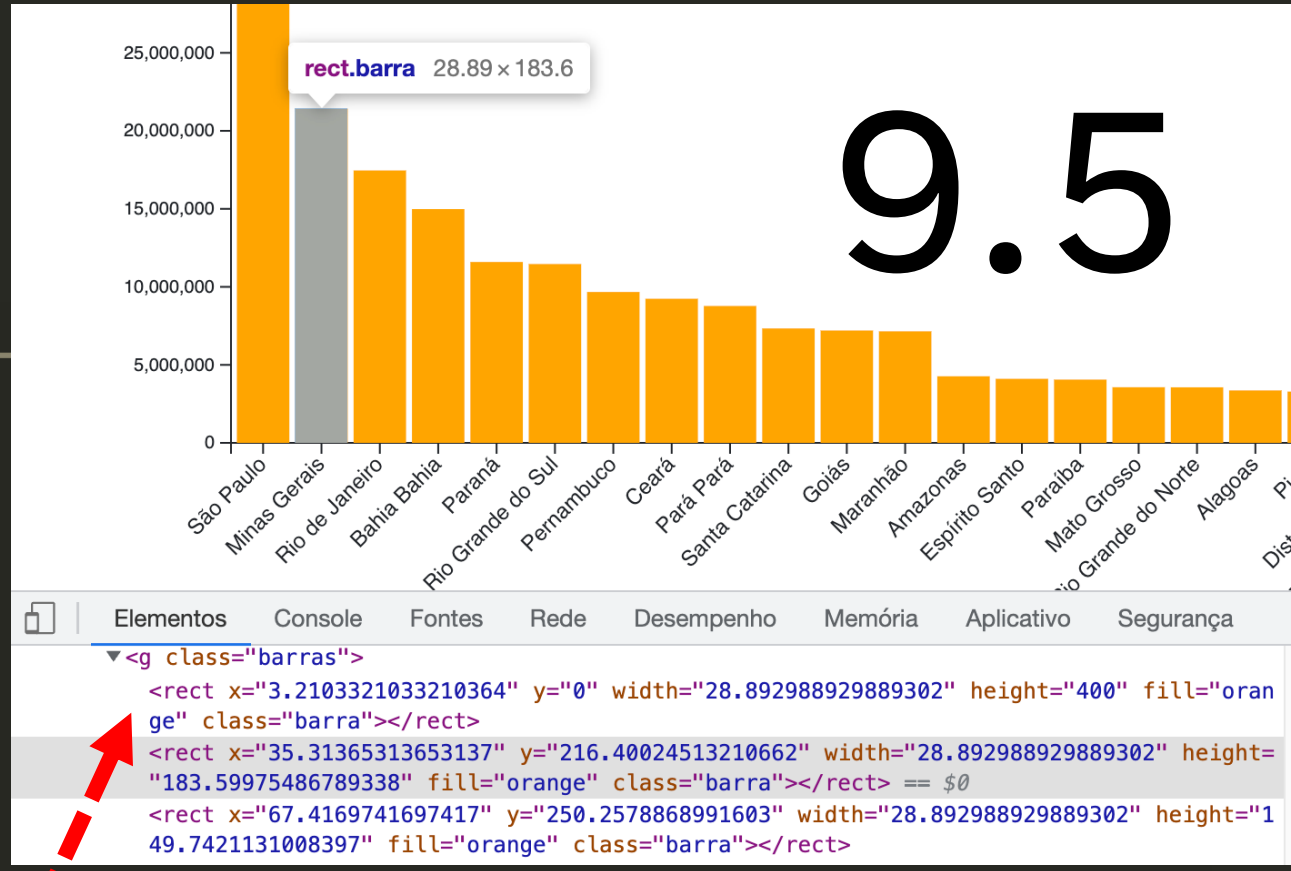
9.4

```
50  // 9) cria cada barra ----------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```



Em qual distância de X a barra será plotada?

```
50  // 9) cria cada barra ---------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```



9.5

Em qual distância de y a barra será plotada?

9.6

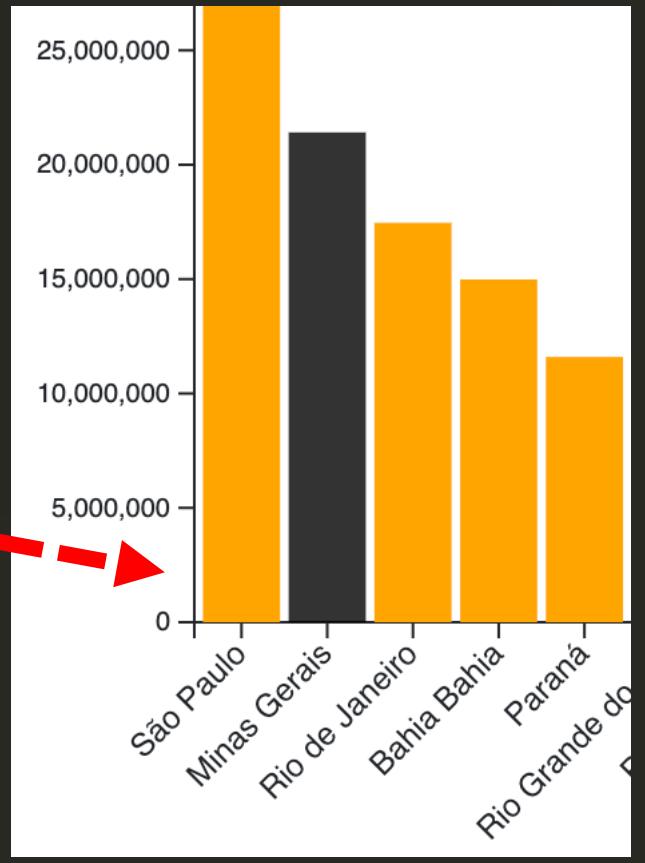
```
50  // 9) cria cada barra ------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```

Largura da barra:

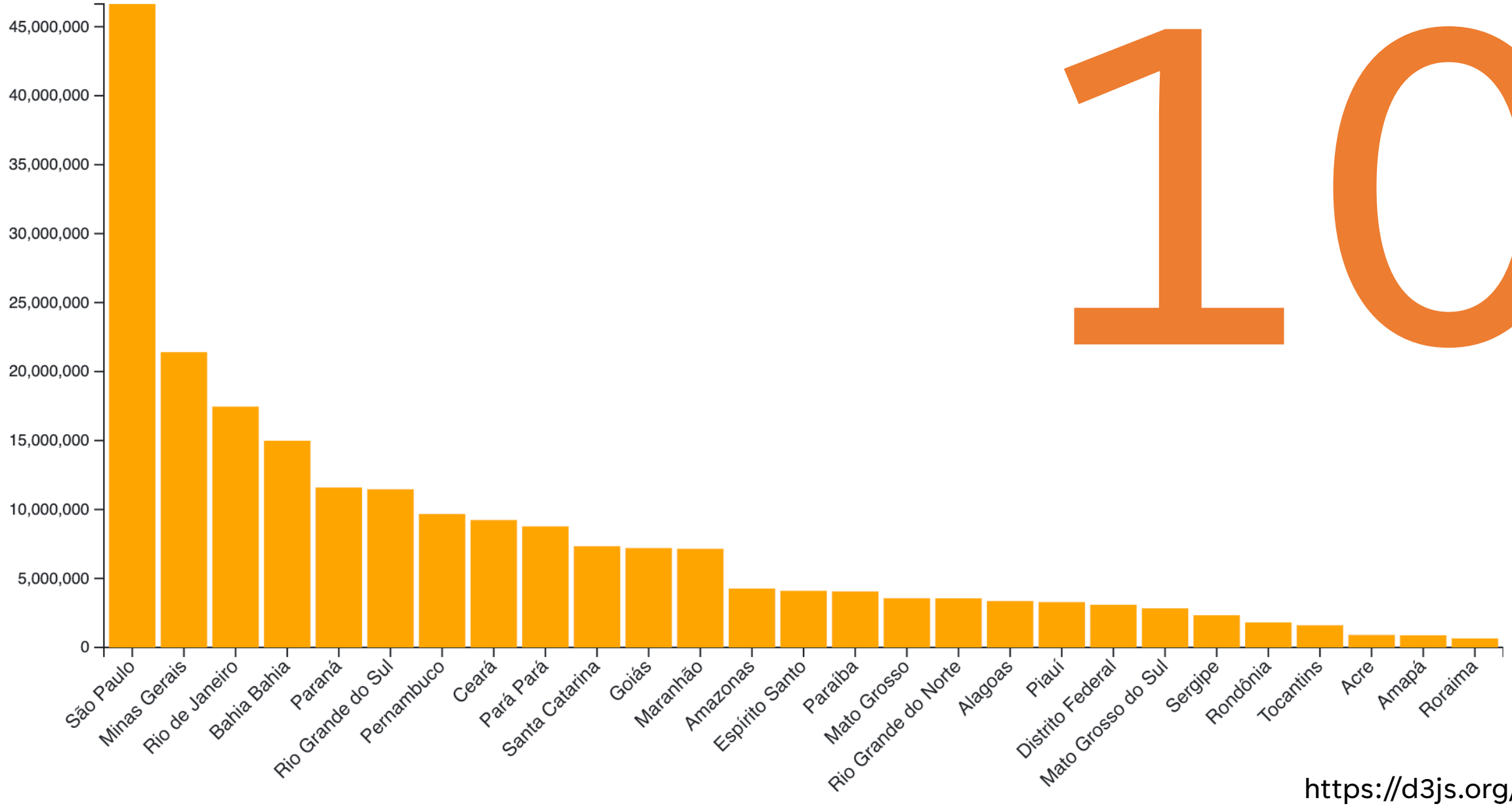
(LARGURA TOTAL – margens) / Quantidade de colunas

9.7

```
50  // 9) cria cada barra ------------------------------------
51  svg.append('g')
52    .attr('class','barras')
53    .selectAll(".barra")
54    .data(data)
55    .join("rect")
56      .attr('class','barra')
57      .attr("x", d => x(d.nome))
58      .attr("y", d => y(d.populacao))
59      .attr("width", x.bandwidth())
60      .attr("height", d => altura - y(d.populacao))
61      .attr("fill", "orange")
```



10



https://d3js.org/

OBRIGADO

Diego Mariano

diego@dcc.ufmg.br

www.diegomariano.com

Código-fonte e slides estão disponíveis em:
https://github.com/dcbmariano/exemplos_d3