

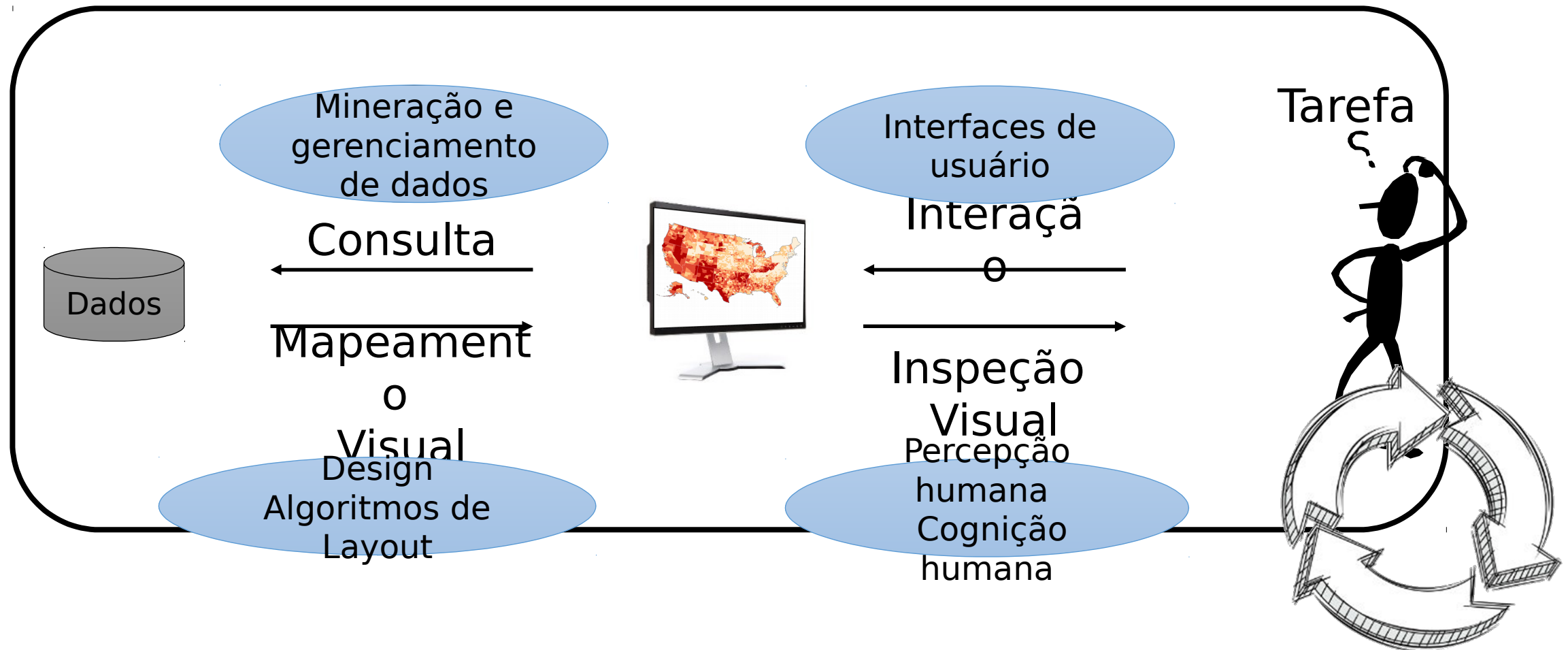
Treinamento Visualização de Dados em D3

[Nivan Ferreira](#)

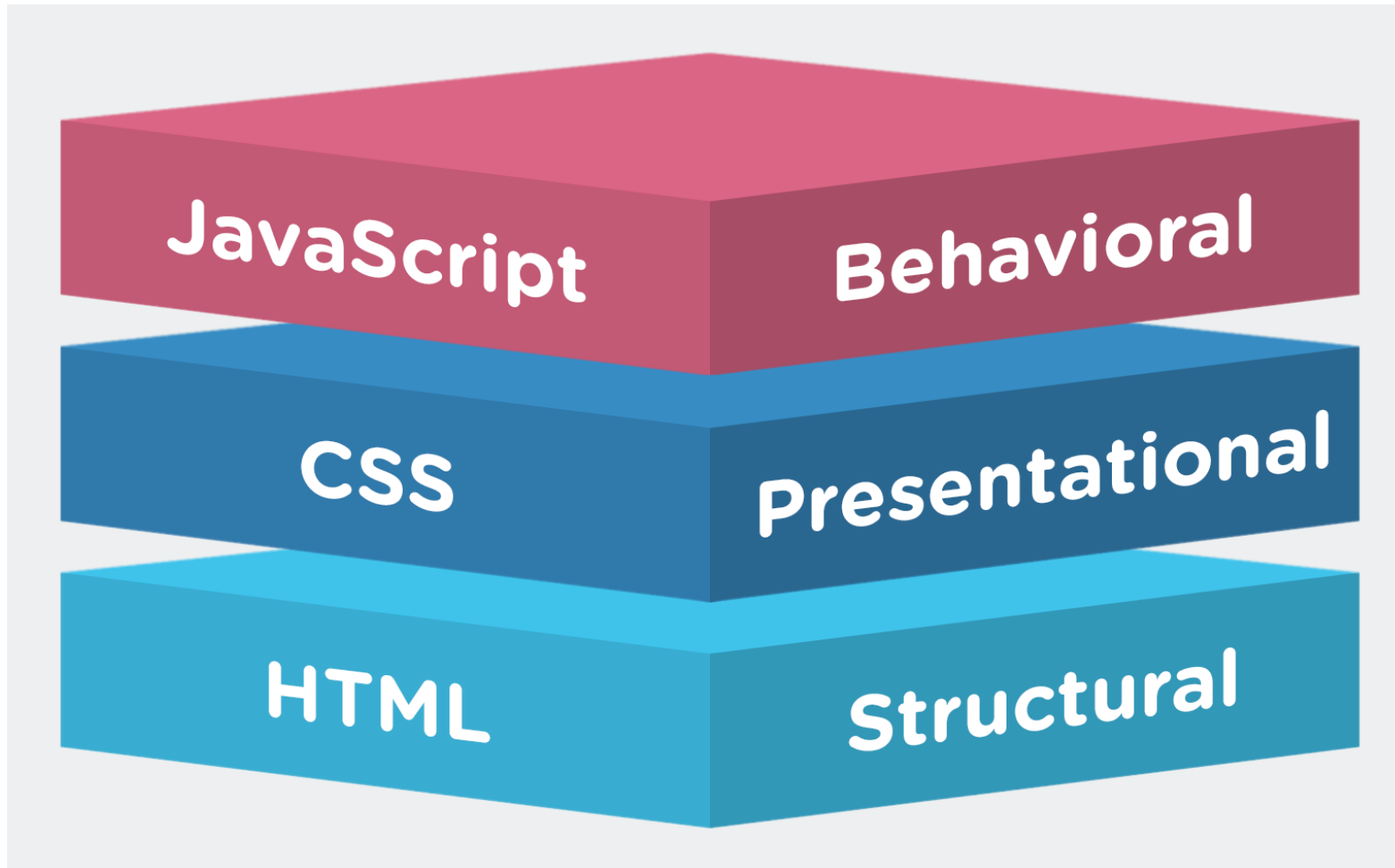
nivan@cin.ufpe.br



Aula Passada



Aula Passada





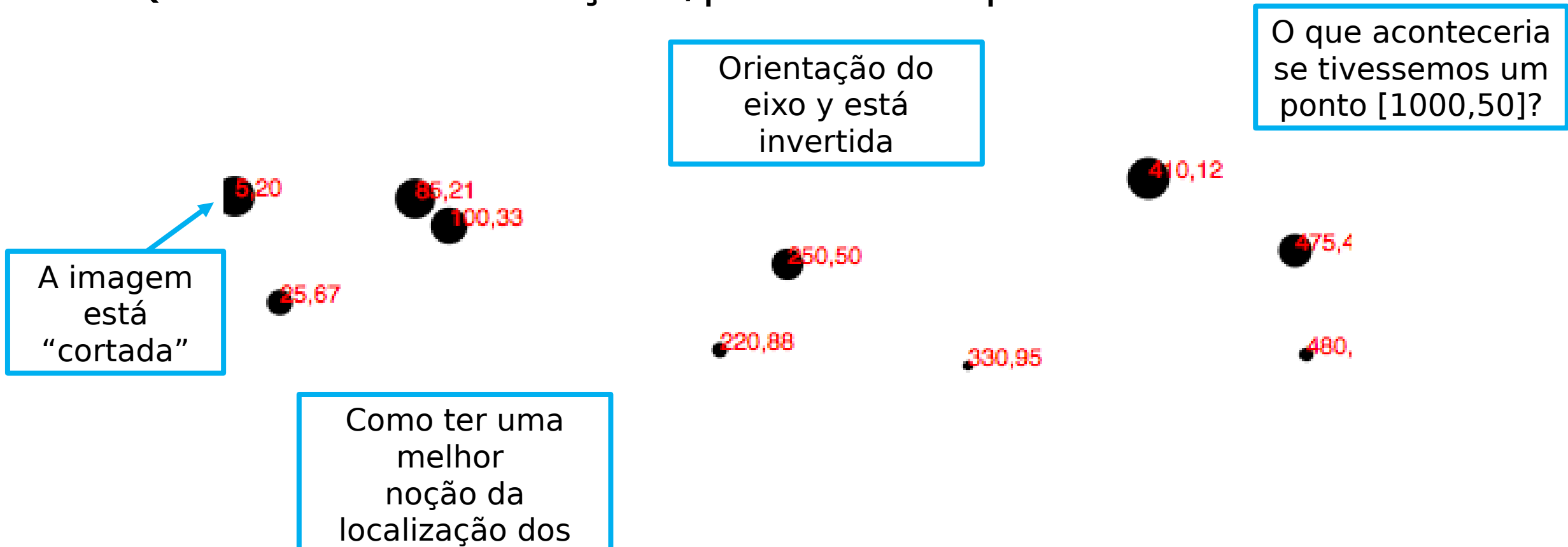
Introdução à Biblioteca D3

Hoje...

Ferramentas do D3.js: Escalas, Eixos,
Seleções, Transições, Interatividade, Layouts

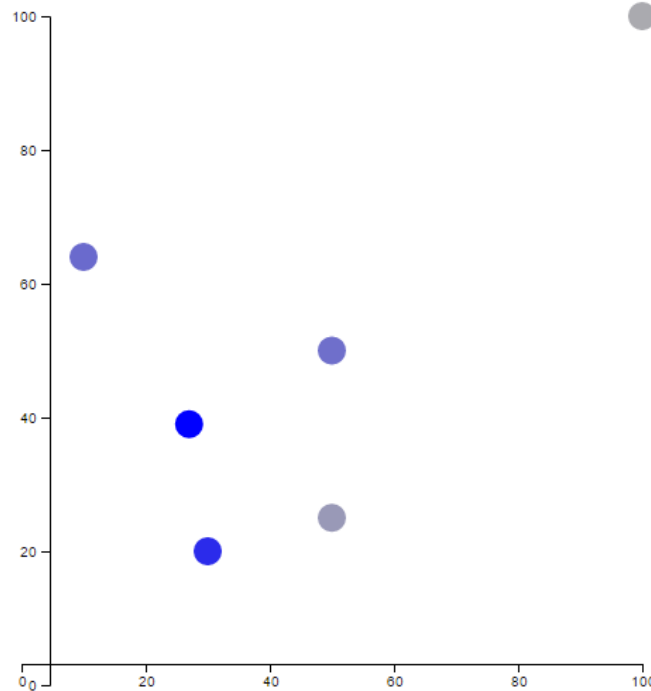
Scatterplot

- Vamos revisar o scatterplot da aula passada
- Quais são as limitações/problemas que temos?



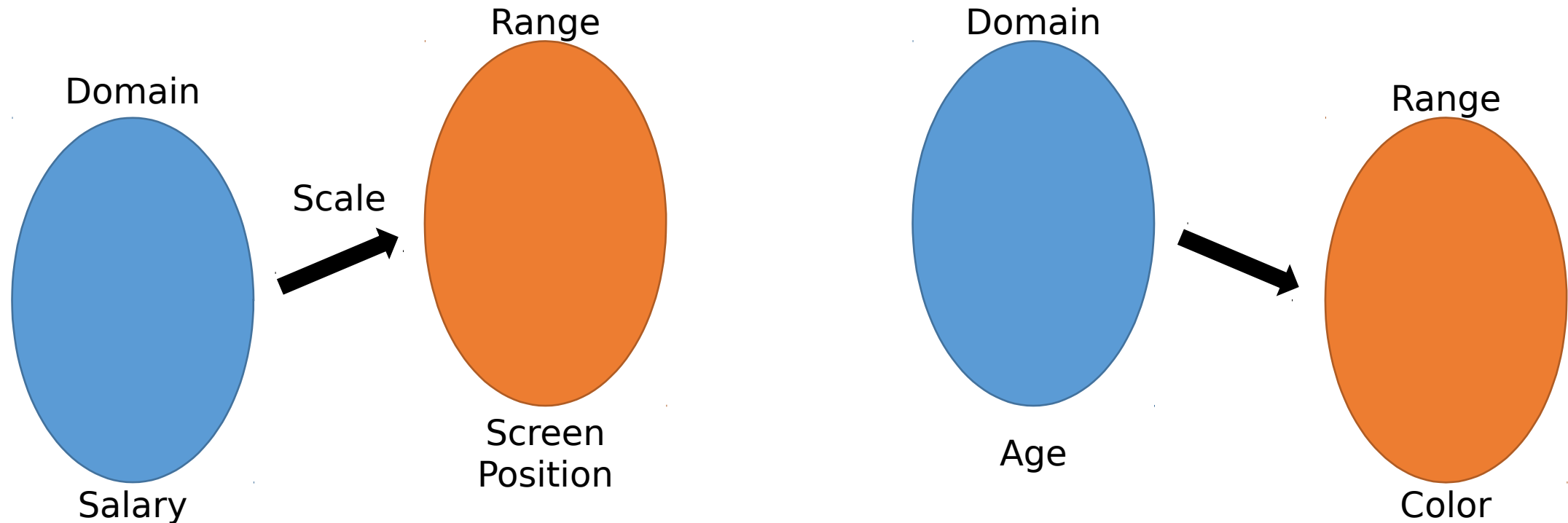
Scatterplot Versão 2

- Vamos revisar o scatterplot da aula passada
- Vamos melhorá-lo e torná-lo dinâmico



Escalas

- “Escalas são funções que mapeiam um domínio de entrada (usualmente dados) para uma domínio de saída (imagem da função) (usualmente propriedades visuais)”



Escalas Lineares

- Funções de Interpolação Linear

- `var scale = d3.scaleLinear().domain([a,b]).range([c,d])`

- Exemplo

- `var scale = d3.scaleLinear().domain([20,50]).range([0,100])`

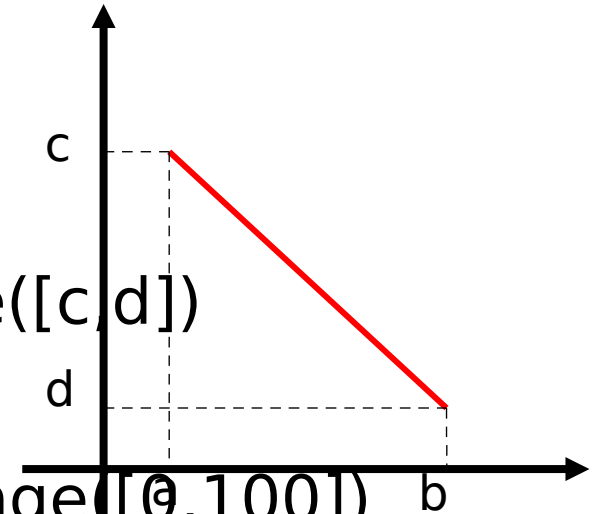
- `scale(20) = ?`

- `scale(50) = ?`

- `scale(35) = ?`

- `scale(0) = ?`

- `scale(100) = ?`



Escalas Lineares

- Usando escalas lineares, podemos mapear dados para coordenadas svg de maneira adequada (evitando que dados sejam plotados fora do svg)
- Processo
 - Encontrar o intervalo dos dados
 - Criar uma escala que mapeia o intervalo de dados em coordenadas de tela apropriadas

Escalas Lineares

- Encontrar o intervalo dos dados
 - d3.extent

- Exemplo

```
var dataset = [5,1,-1,9,100,50];  
console.log(d3.extent(dataset));
```

- Para dados complexos, podemos usar uma função que acessa o elemento com o qual desejamos achar o intervalo

```
var dataset = [[5, 20], [480, 90], [250, 50], [100, 33], [330, 95]];  
console.log(d3.extent(dataset, d=>d[0]));
```

Escalas Lineares

- Agora precisamos criar a escala que mapeia o intervalo em coordenadas de tela
- ```
var dataXInterval = d3.extent(dataset, d => d[0]);
var xScale =
d3.scaleLinear().domain(dataXInterval).range([0, 500]);
```
- ```
var dataYInterval = d3.extent(dataset, d => d[1]);  
var yScale =  
d3.scaleLinear().domain(dataYInterval).range([0, 500]);
```

Escalas Lineares

- Agora usaremos essas escalas

mySVG

.selectAll("circle")

.data(dataset)

.enter()

.append("circle")

.attr("r","10")

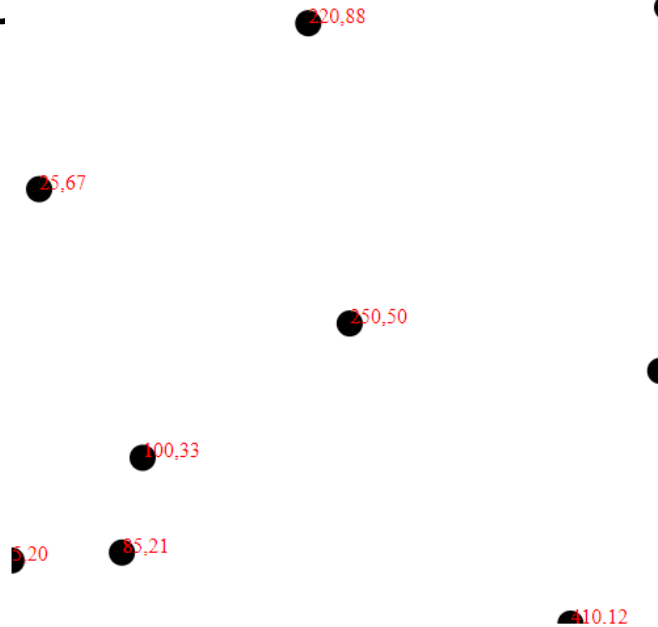
.attr("cx",function(d){return xScale(d[0]);})

.attr("cy",function(d){return yScale(d[1]);});

Escalas Lineares

- Como corrigir o problema da orientação do eixo y?

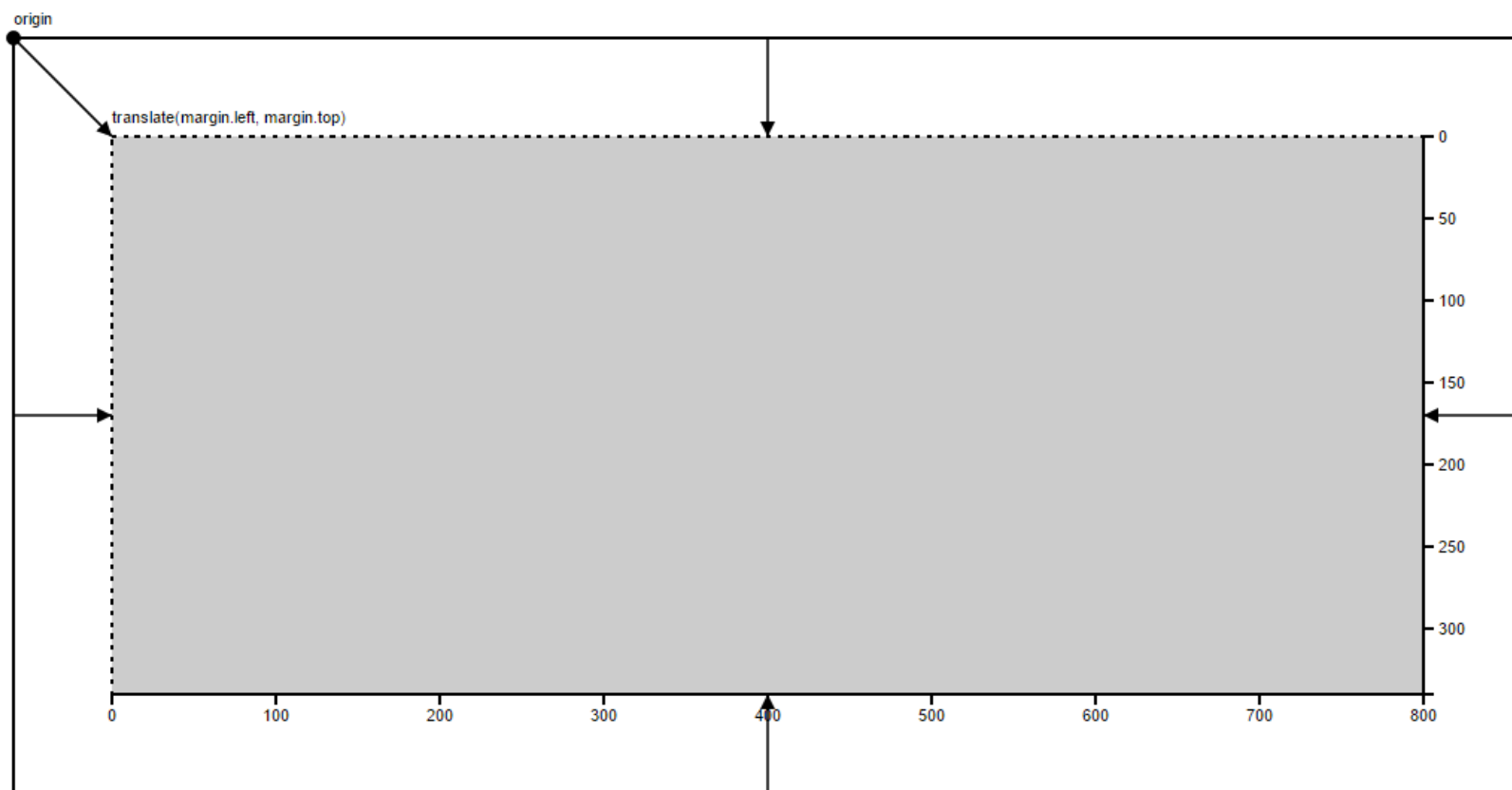
```
var yScale =  
d3.scaleLinear().domain([0, 500]).range([500, 0]);
```



Margins

- Note que os pontos e texto do nosso scatterplot estão fora do SVG

- Para




desenho

Conversão de Margens

- Defina a margem
 - `var margin = {top: 3, right: 2, bottom: 5, left: 5};`
- Defina largura e altura
 - `var width = 500 - margin.left - margin.right;`
 - `var height = 500 - margin.top - margin.bottom;`
- Defina o SVG
 - `var mySVG = d3.select("svg")
 .append("g")
 .attr("transform", "translate(" + margin.left + ","
 + margin.top + ")");`

A partir
daqui,
podemos
ignorar as
margens



Conversão de Margens

- Definimos Escalas para posicionamento

```
var dataXInterval = d3.extent(dataset,d=>d[0]);  
var xScale =  
d3.scaleLinear().domain(dataXInterval).range([0,width]);  
  
var dataYInterval = d3.extent(dataset,d=>d[1]);  
var yScale =  
d3.scaleLinear().domain(dataYInterval).range([height,0]);
```

Escalas de Cores (Colormaps)

- Podemos usar escalas lineares para mapear cores
- D3 automaticamente interpola cores
- Exemplo

- `var cScale =
 d3.scaleLinear().domain([0,100]).range(["black","red"])`
- `cScale(0)?`
- `cScale(100)?`
- `cScale(50)?`



Escalas de Cores (Colormaps)

- Exercício

```
var dataset = [[5, 20, 9], [480, 90, 5], [250, 50, 1],  
[100, 33, 0], [330, 95, 10], [410, 12, 3], [475, 44, 7], [25,  
67, 9], [85, 21, 2], [220, 88, 2]];
```

- Defina a cor dos pontos do scatterplot proporcional à terceira coordenada
- Solução [third_scatterplot.html](#)

Escalas de Tempo

- Escalas lineares com domínio temporal
- `var x = d3.scaleTime()
 .domain([new Date(2000, 0, 1), new Date(2000, 0, 2)])
 .range([0, 960]);`
- Exemplos
 - `x(new Date(2000, 0, 1, 5))?`
 - `x(new Date(2000, 0, 1, 16)) ?`
 - `x.invert(200) ?`
 - `x.invert(640) ?`

Escalas Quantizadas

- Mapeiam um domínio em contínuo em um conjunto discreto

var

```
q=d3.scale.quantize().domain([0,10]).range([0,2,8]);
```

q(0); ?

q(3); ?

q(3.33); ?

q(3.34); ?

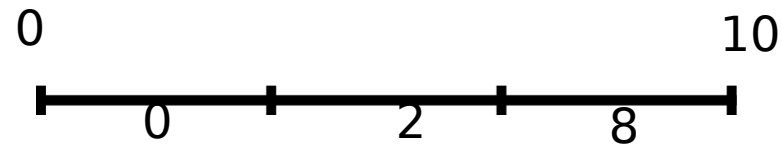
q(5); ?

q(6.66); ?

q(6.67); ?

q(8); ?

q(1000); ?



Escalas Quantizadas

- Mapeiam um domínio em contínuo em um conjunto discreto

var

```
q=d3.scale.quantize().domain([0,10]).range([0,2,8]);
```

```
q(0); // 0
```

```
q(3); // 0
```

```
q(3.33); // 0
```

```
q(3.34); // 2
```

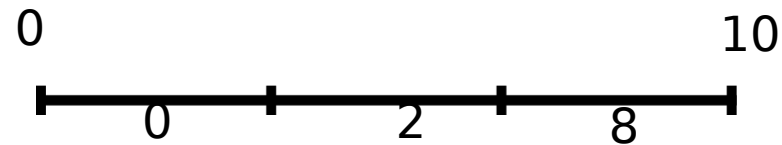
```
q(5); // 2
```

```
q(6.66); // 2
```

```
q(6.67); // 8
```

```
q(8); // 8
```

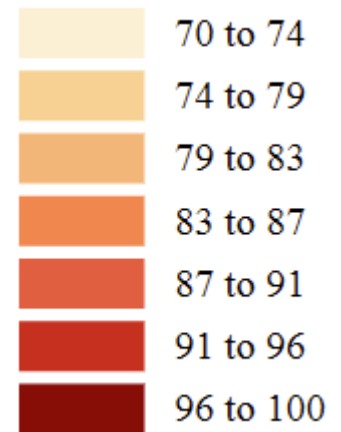
```
q(1000); // 8
```



Escalas Quantizadas

- Muito útil para criar escalas de cores
- ```
var colorset = ['#fef0d9', '#fdd49e',
 '#fdbb84', '#fc8d59',
 '#ef6548', '#d7301f', '#990000'];
```

```
var colorScale = d3.scaleQuantize()
 .domain([70, 100])
 .range(colorset);
```



# Escalas Ordinais

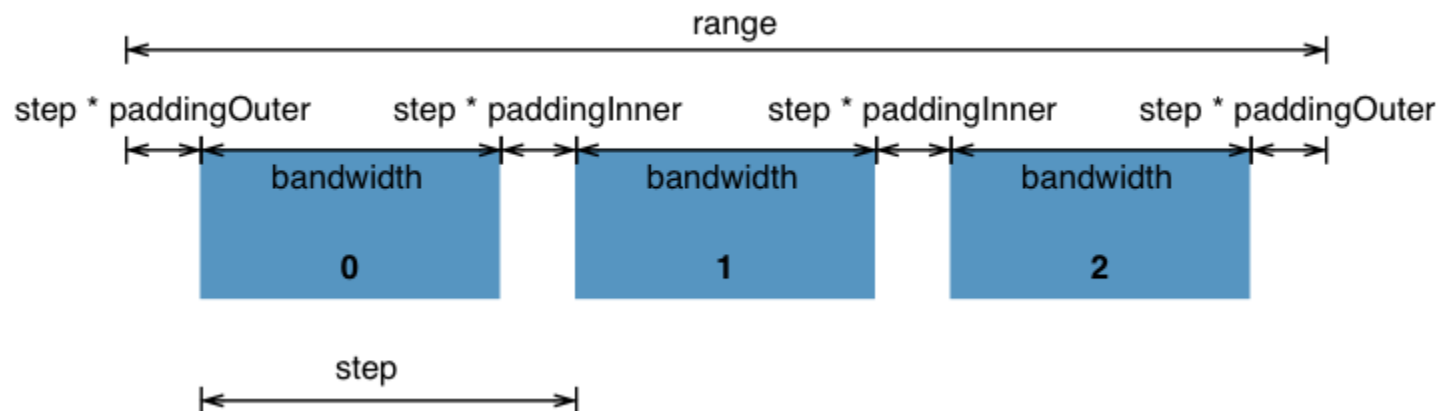
- Domínio e Imagem são discretas
- `var myScale = d3.scaleOrdinal()`
  - `.domain(["Brasil","EUA","Argentina"])`
  - `.range(["yellow","red","blue"])`
- Exemplos
  - `myScale("Brasil")`
  - `myScale("EUA")`
  - `myScale("a")`
  - `myScale("-1")`

**Escalas ordinais  
possuem um valor  
*unknown* para  
entradas fora do  
domínio**



# Escalas de Banda

- Estas escalas têm domínio discreto, mas range contínuo
- O range é dividido em bandas automaticamente
- Muito usadas para plotar histogramas em categorias
- `var myScale = d3.scaleBand ()`



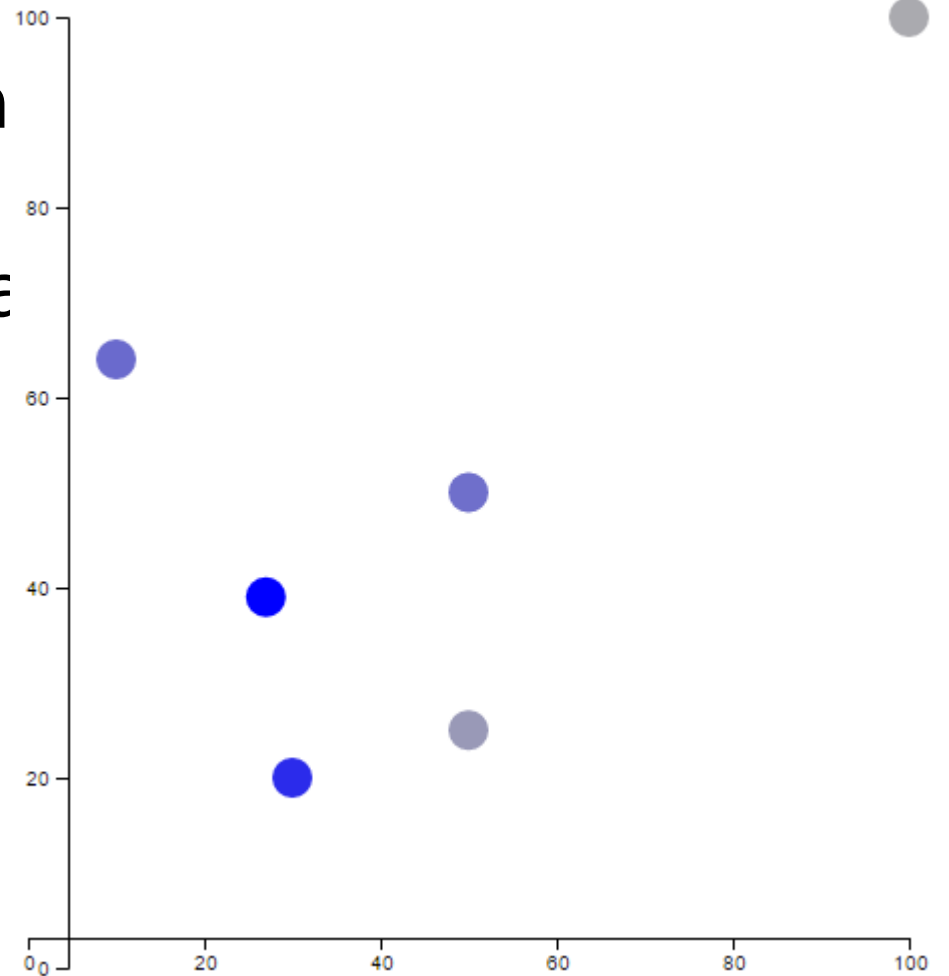
# Exercício

- Modifique o código para plotar histogramas da aula passada para usar uma escala de banda
- Use diferentes valores dos parâmetros de `paddingInner` e `paddingOuter`

Eixos

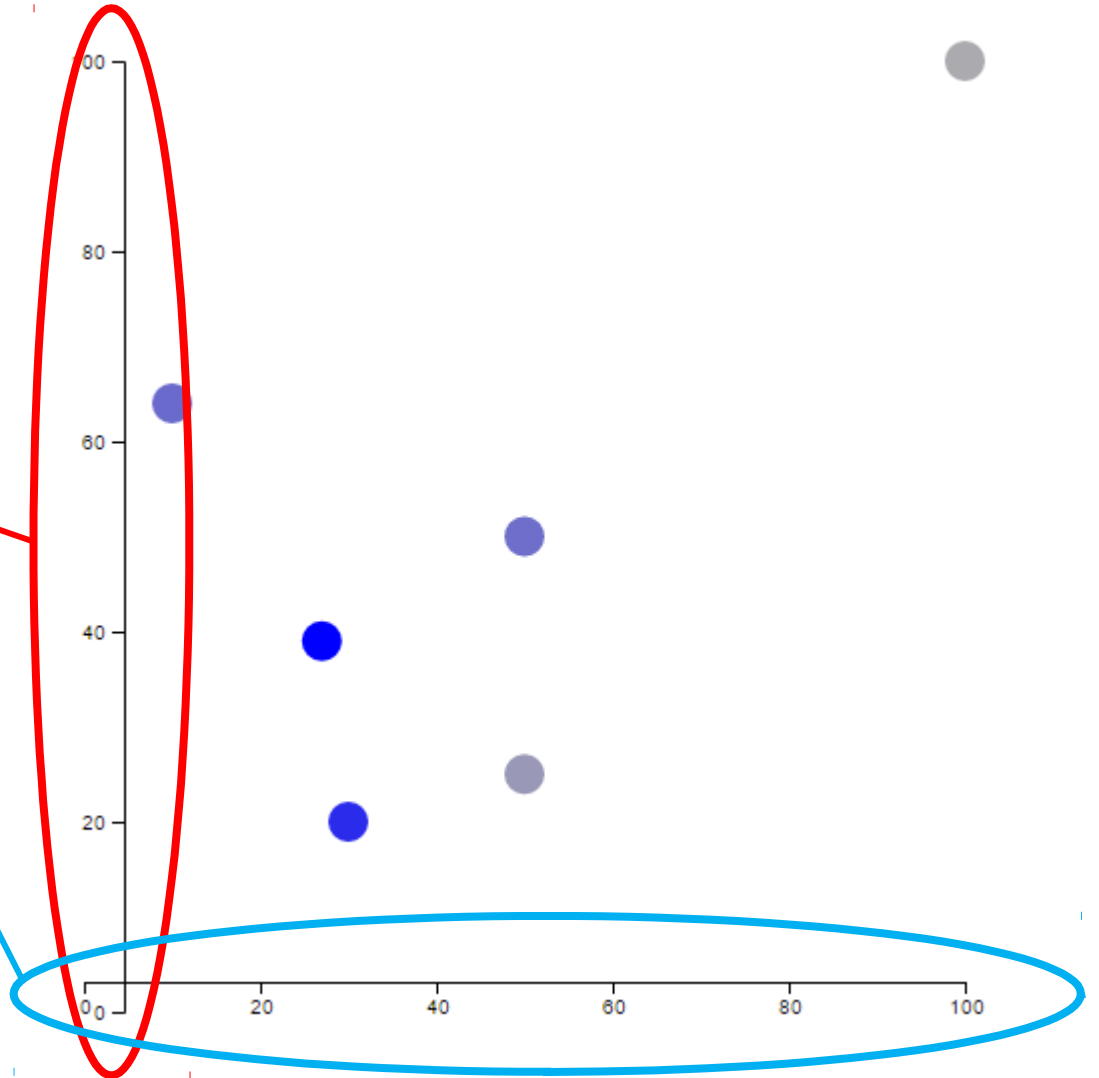
# Eixos

- O d3 possui funcionalidades para criar eixos
- Representações visuais de escala



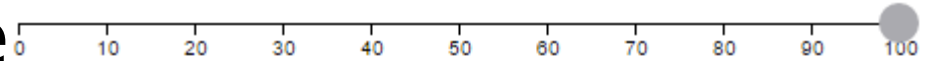
# Eixos

- `d3.axisBottom(xScale);`
- `d3.axisTop(xScale);`
- `d3.axisLeft(xScale);`
- `d3.axisRight(xScale);`

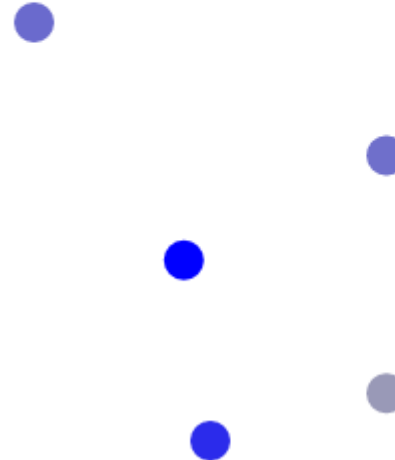


# Eixos

```
var xAxis = d3.axisBottom(xScale,
 mySVGxAxisGroup.call(xAxis);
```



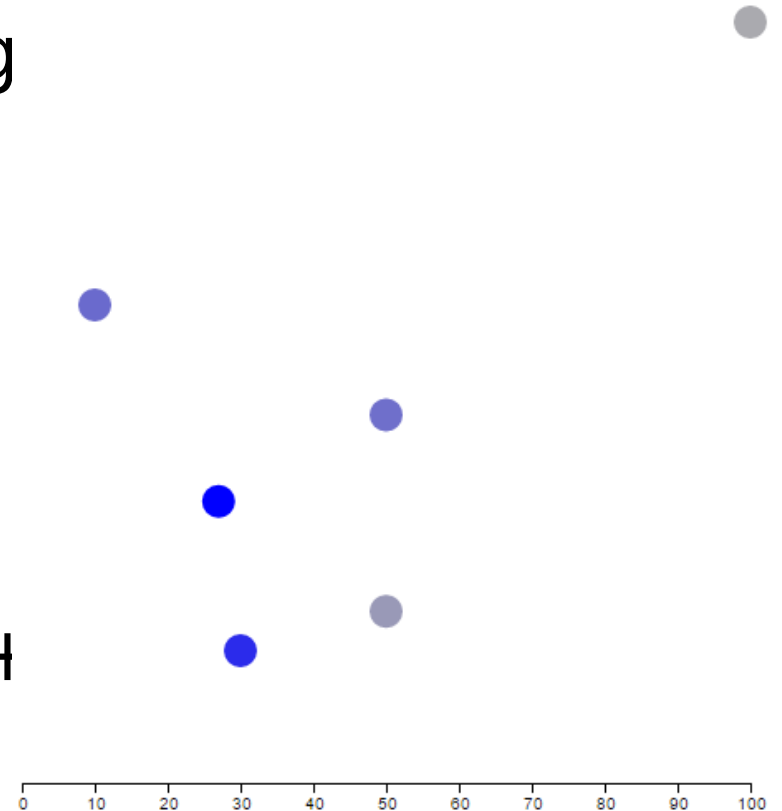
Função que passa a  
selection (chamador)  
para o objecto  
(parâmetro)  
equivalente à  
xAxis(mySVGxAxisGr  
oup)



# Eixos

- Eixos são sempre desenhados na orig
- Para colocar na posição desejada

```
var xAxisGroup = mySVG.append("g")
 .attr("class", "xAxis")
 .attr("transform",
 "translate(0,"+(height-margin.top)+
var xAxis = d3.axisBottom(xScale);
xAxisGroup.call(xAxis);
```



# Eixos

- Olhe na aba de elementos da ferramenta de inspeção do browser para ver a representação do eixo como elementos do DOM.

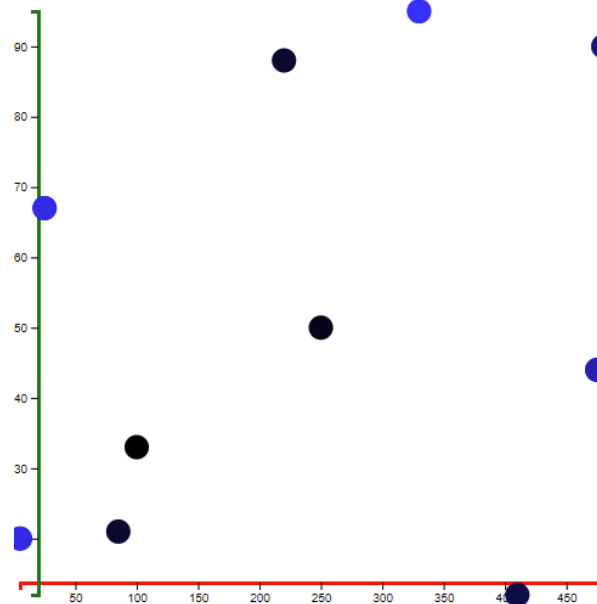
O que você consegue notar?

```
<g class="xAxis" transform="translate(0,470)" fill="none"
font-size="10" font-family="sans-serif" text-anchor="middle">
 <path class="domain" stroke="#000" d=
 "M0.5,6V0.5H480.5V6"></path>
 ▶ <g class="tick" opacity="1" transform=
 "translate(45.973684210526315,0)">...</g>
 ▶ <g class="tick" opacity="1" transform="translate(96.5,0)
 ">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(147.0263157894737,0)">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(197.55263157894737,0)">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(248.07894736842107,0)">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(298.60526315789474,0)">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(349.13157894736844,0)">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(399.65789473684214,0)">...</g>
 ▶ <g class="tick" opacity="1" transform=
 "translate(450.1842105263158,0)">...</g>
</g>
```



# Exercício

- Adicione o eixo y no seu scatterplot
- Use regras CSS para mudar a aparência dos eixos para ter um plot como abaixo
- Faça o mesmo com o D3 select



# Atualizações em Visualizações

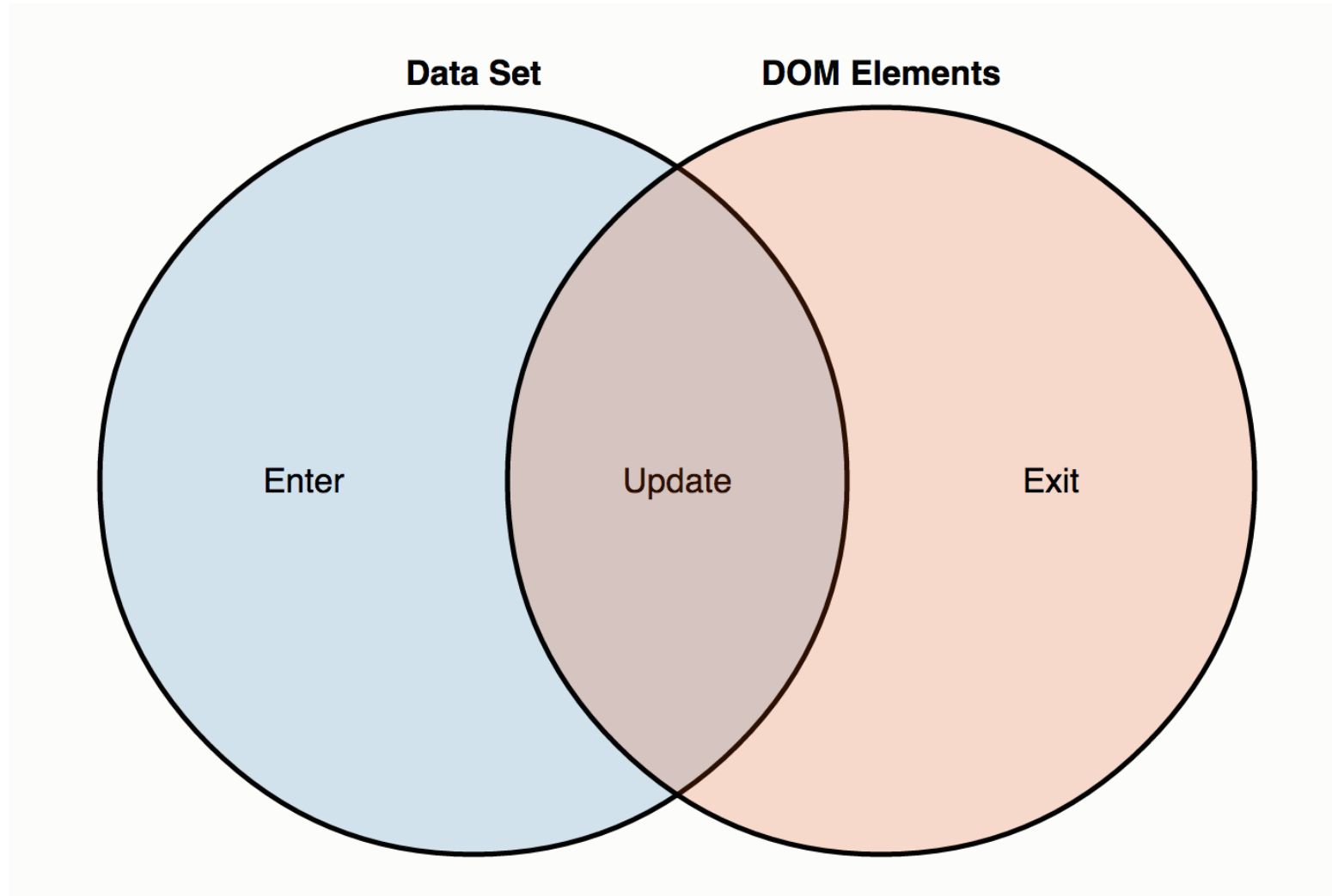
# Updates em Plots

- Bind data
- enter() -> adiciona elementos necessários
- exit() -> remove elementos excedentes
- Update -> set valores dos parâmetros

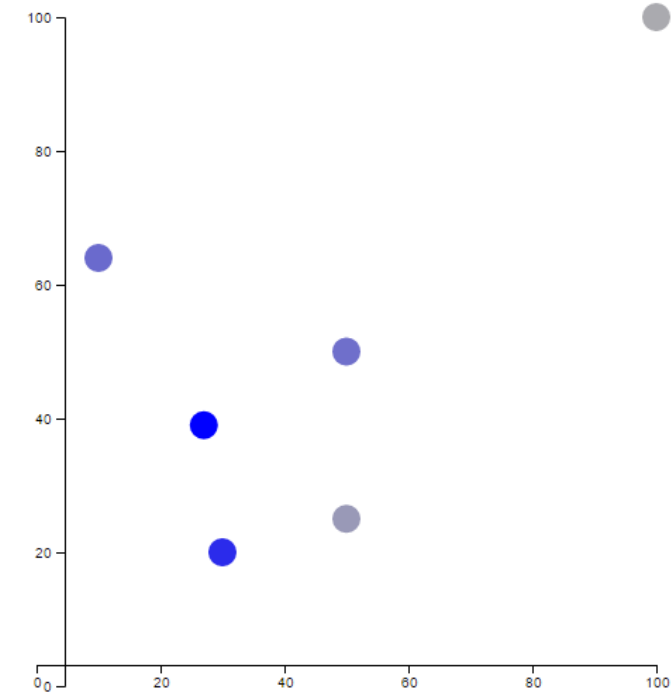
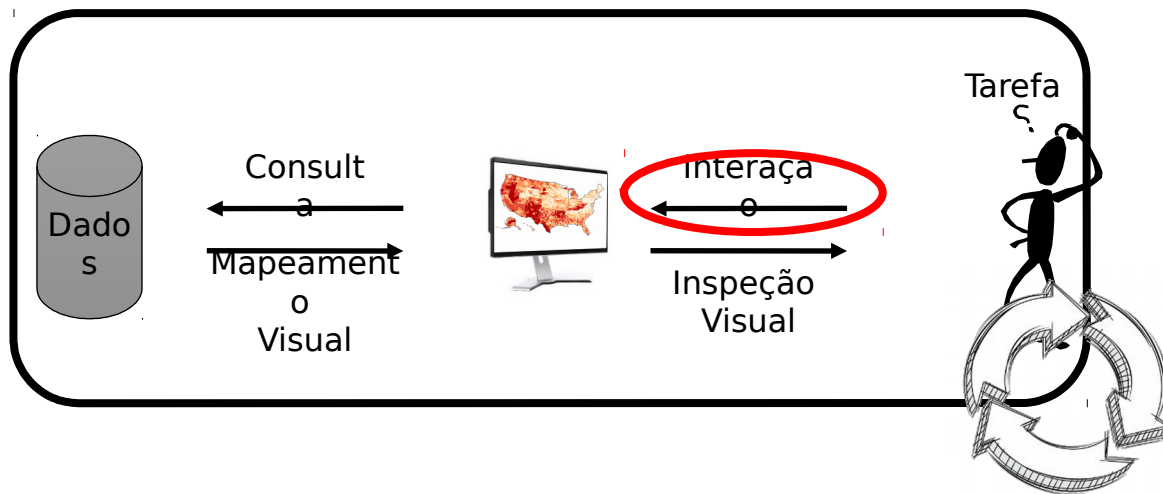
# Updates em Plots

- Bind data
- enter() -> adiciona elementos necessários
- exit() -> remove elementos excedentes
- Update -> set valores dos parâmetros

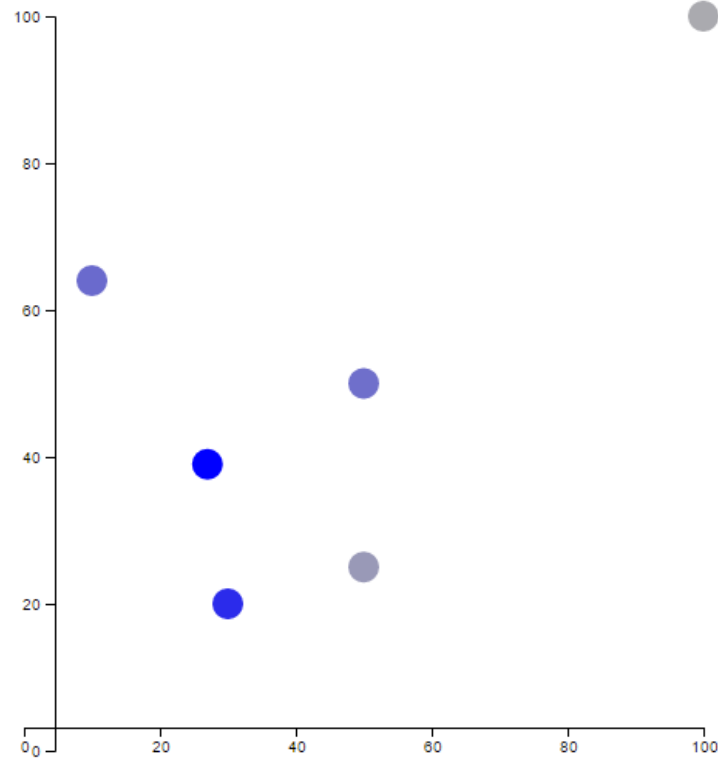
# D3 – Seleções



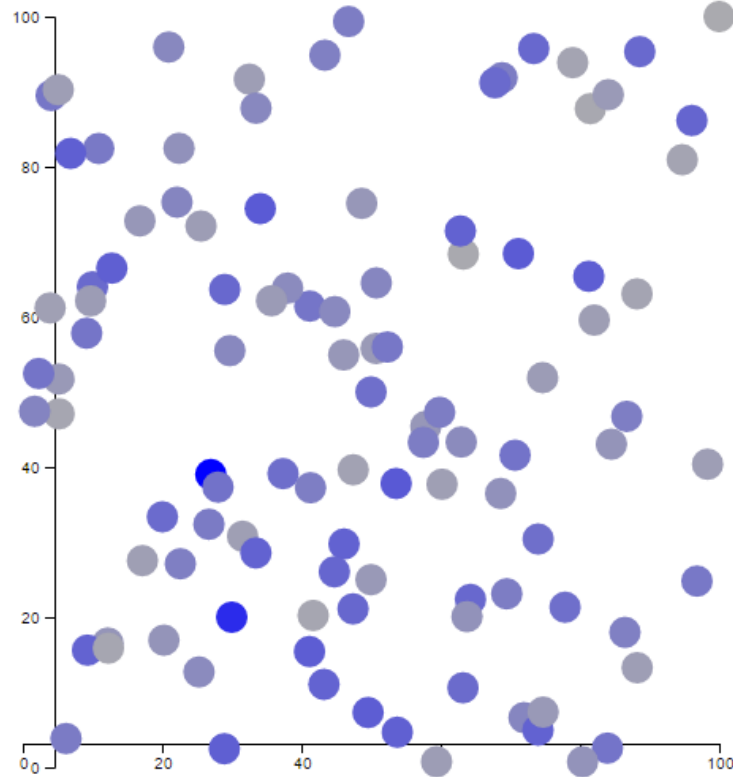
# Adicionar Navegação, Interação e Dinamismo nas Visualizações



# Que Tipo de Interações Gostaríamos de Fazer?



# Que Tipo de Interações Gostaríamos de Fazer?





# Brushes (Seleção com Mouse)

- Representam seleções com mouse

```
var myBrush = d3.brush();
var brushGroup =
mySVG.append("g").attr("class","brush");
myBrush(brushGroup);
```

# Brushes (Seleção com Mouse)

- Que coisas gostaríamos de poder controlar nos/fazer com Brushes?
  - Controlar evento iniciador
  - Selecionar pontos na tela

# Brushes (Seleção com Mouse)

- O evento que se inicia o brush é definido na função *filter*

```
myBrush.filter(function(){
 return d3.event.button === 1;
})
```

# Brushes (Seleção com Mouse)

- Como Selecionar Pontos na Tela?
  - É possível executar ações ao início, durante e ao término da ação de brush

*brush.on("start",function(){})*

*brush.on("brush",function(){})*

*brush.on("end",function(){})*

# Brushes (Seleção com Mouse)

- Mais exemplos de brushes
  - <https://bl.ocks.org/mbostock/4349545>
  - <https://bl.ocks.org/mbostock/6232537>
  - <https://bl.ocks.org/mbostock/34f08d5e11952a80609169b7917d4172>

# Zoom e Pan

- Navegação em uma visualização

```
svg.append("rect")
```

```
.attr("width", width)
```

```
.attr("height", height)
```

```
.style("fill", "none")
```

```
.style("pointer-events", "all")
```

```
.call(d3.zoom().on("zoom", zoomed))
```

Cria um retângulo  
para capturar os  
eventos de mouse

Associa um  
compartamento de  
zoom a esse  
retângulo

# Zoom e Pan

```
function zoomed() {
 circleGroup.attr("transform", d3.event.transform);
 d3.select(".xAxis").call(xAxis.scale(d3.event.transform.re
scaleX(xScale)));

 d3.select(".yAxis").call(yAxis.scale(d3.event.transform.re
scaleY(yScale)));
}
```

# Transições

- Em aulas passadas vimos o papel das seleções `enter()` e `exit()` na atualização de visualizações
- Vamos falar um pouco mais sobre esse ponto considerando transições (**baixe [link](#) e [link](#)**)
- **Na atualização dos pontos, modifique para**

`circleSelection`

`.transition()`

`.attr(...`



# Transições

- Em aulas passadas vimos o papel das seleções `enter()` e `exit()` na atualização de visualizações
- Transições são uma ferramenta poderosa para controlar a atualização das nossas visualizações
- Uso

`selection`

`.transition()`

`.attr(...)`

# Atualização das Visualizações

- Depois de chamar `.transition()`, é possível incluir opções de atraso e duração com as funções

`.delay(1000) //1,000 ms or 1 second`

`.duration(2000) //2,000 ms or 2 seconds`

# Atualização das Visualizações

- Útil na atualização das escalas

```
d3.select("#xAxis")
 .transition()
 .call(xAxis);
```

# Reusable Plots

# Coordinated Views