

tt()

Schedule

1. Recap
2. Scales in D3
3. Drawing an axis

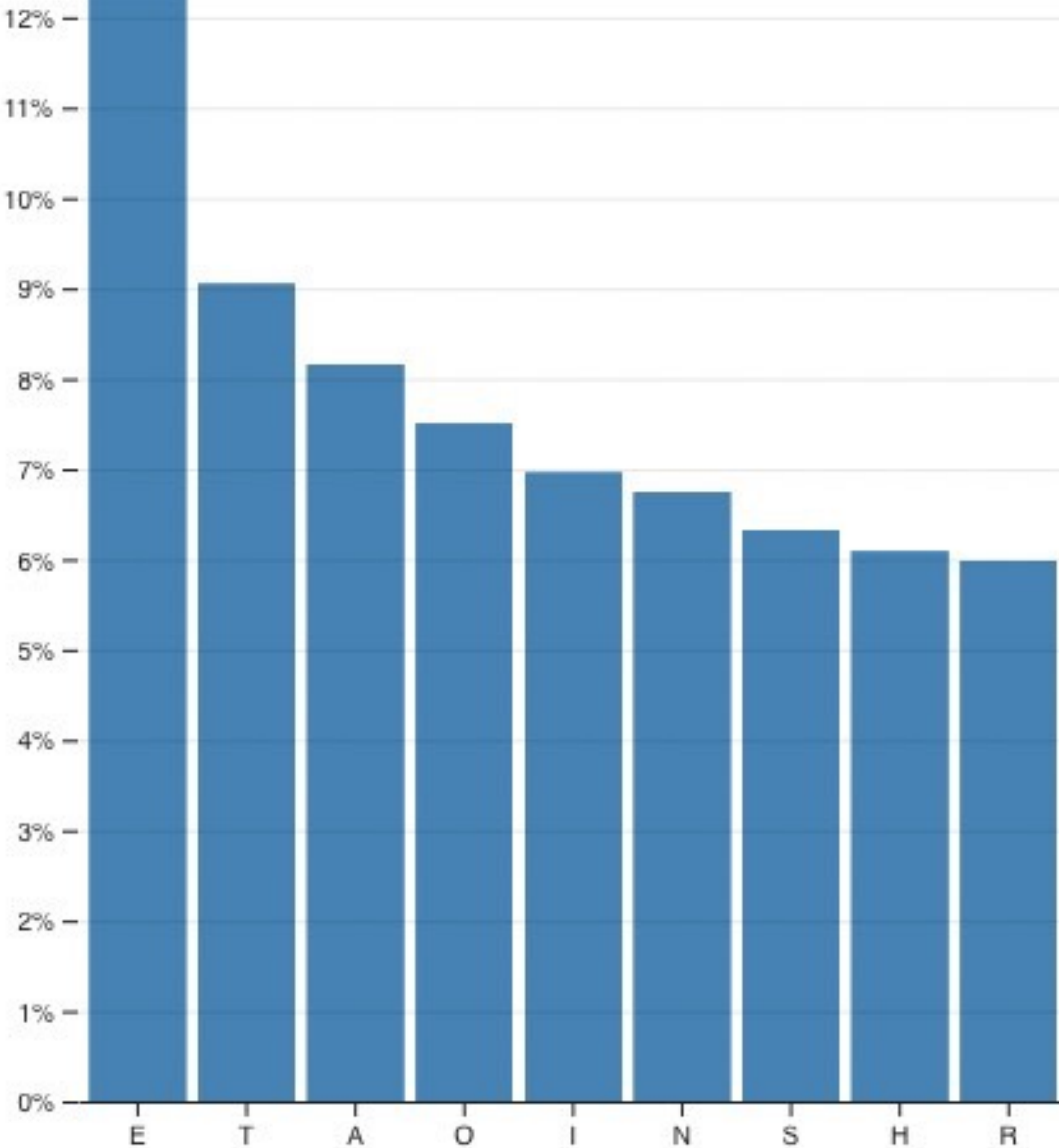


Schedule

1. Recap
2. Scales in D3
3. Drawing an axis



D3 Concepts



1. Selections

2. Joins

3. Accessor functions

4. Scales

5. Axes

6. Ticks

Selections – the purpose

We work with HTML elements, SVG ... and want to modify them

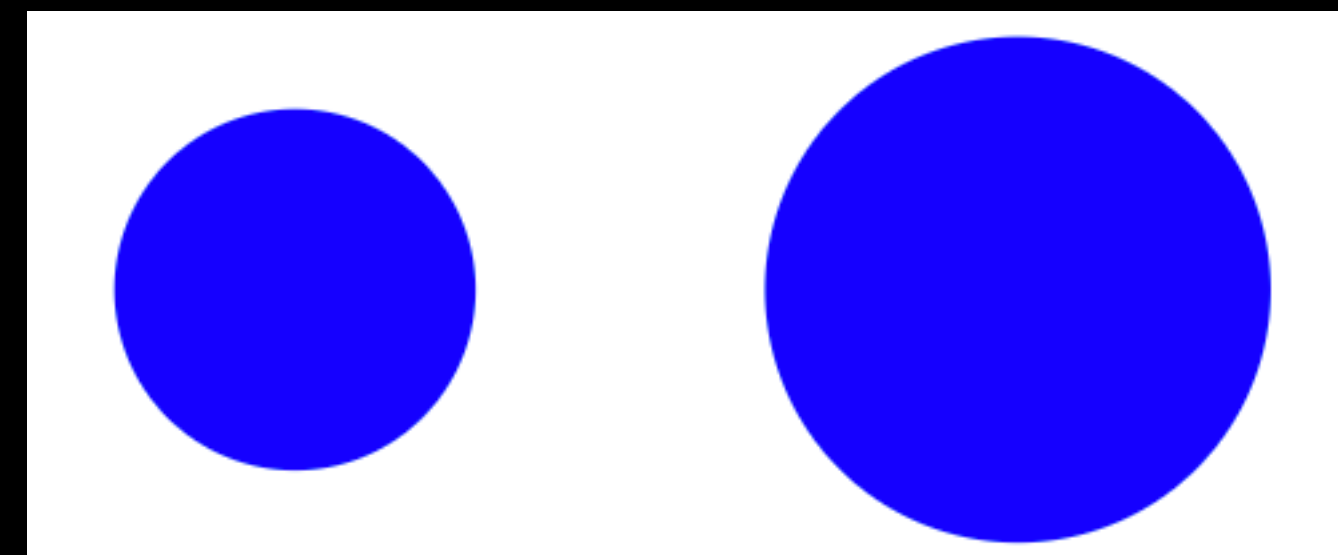
`d3.select()` is kinda like `document.querySelector()`

`d3.selectAll` is kinda like `document.querySelectorAll()`

```
d3.selectAll('circle')
```

```
<svg>  
  <circle cx=10 cy=10 r=5 />  
  <circle cx=30 cy=10 r=7 />  
</svg>
```

```
d3.selectAll('circle').style('fill', 'blue')
```



Selections

`d3.select()` is kinda like `document.querySelector()`

`d3.selectAll` is kinda like `document.querySelectorAll()`

Name	Behaviour	Example
<code>.style</code>	Update the style	<code>d3.selectAll('circle').style('fill', 'red')</code>
<code>.attr</code>	Update an attribute	<code>d3.selectAll('rect').attr('width', 10)</code>
<code>.classed</code>	Add/remove a class attribute	<code>d3.select('.item').classed('selected', true)</code>
<code>.property</code>	Update an element's property	<code>d3.selectAll('input[type=checkbox]').property('checked', true)</code>
<code>.text</code>	Update the text content	<code>d3.select('h1').text('Hello world')</code>
<code>.html</code>	Change the html content	<code>d3.select('form').html('<button>Turn off</button>')</code>

Joins – the purpose

We never know in advance how many elements of the DOM (HTML, SVG) we will need...

It depends on our data

Joins

Data joins are kinda like doing a mail merge in Office to create address labels based on a list in Excel



```
<svg id="chart"></svg>
```

```
<script>
```

```
let myData = [40, 10, 20, 60, 30];
```

```
d3.select('#chart')
```

```
  .selectAll('rect')
```

```
  .data(myData)
```

```
  .join('rect');
```

```
</script>
```

Here we use `d3.join()` to create a `<rect>` element for each item in our `myData` array

Accessor functions – the purpose



```
<svg id="chart"></svg>
```

```
<script>
```

```
  const myData = [
```

```
    { day: "Monday", cars: 40 },
```

```
    { day: "Tuesday", cars: 10 },
```

```
    { day: "Wednesday", cars: 20 },
```

```
    { day: "Thursday", cars: 60 },
```

```
    { day: "Friday", cars: 30 },
```

```
  ];
```

```
  d3.select("#chart")
```

```
    .selectAll("rect")
```

```
    .data(myData)
```

```
    .join("rect")
```

```
    .attr('width', d => d.cars);
```

```
</script>
```

If you're using JSON (an array of objects) you'll need to tell D3 which property you want to use

← Accessor function

About Sveltekit and D3

```
<script>
  import * as d3 from 'd3';
  import { onMount } from 'svelte';

  const myData = [4, 8, 15, 16, 23, 42];

  onMount(() => {
    console.log('D3 is running after mount!');

    d3.select('#chart')
      .selectAll('rect')
      .data(myData)
      .join('rect')
      .attr('x', (d, i) => i * 30)
      .attr('y', (d) => 100 - d * 2)
      .attr('width', 20)
      .attr('height', (d) => d * 2)
      .attr('fill', 'steelblue');
  });
</script>

<svg id="chart" width="300" height="100"></svg>
```

Exercise

Practise with selections, joins and accessor function:

https://codepen.io/Laura_B/pen/KKOBMgI

Show and tell

Schedule

1. Recap
- 2. Scales in D3**
3. Drawing an axis



SCAII

ING

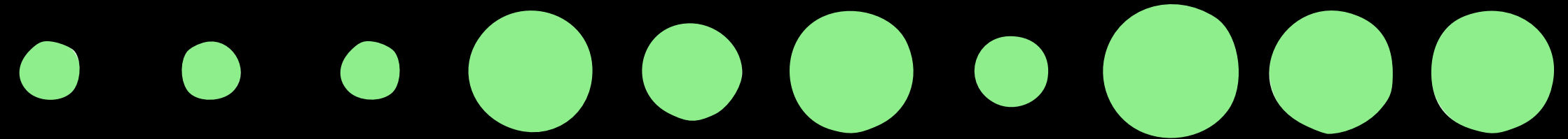
D3

Scales – the purpose

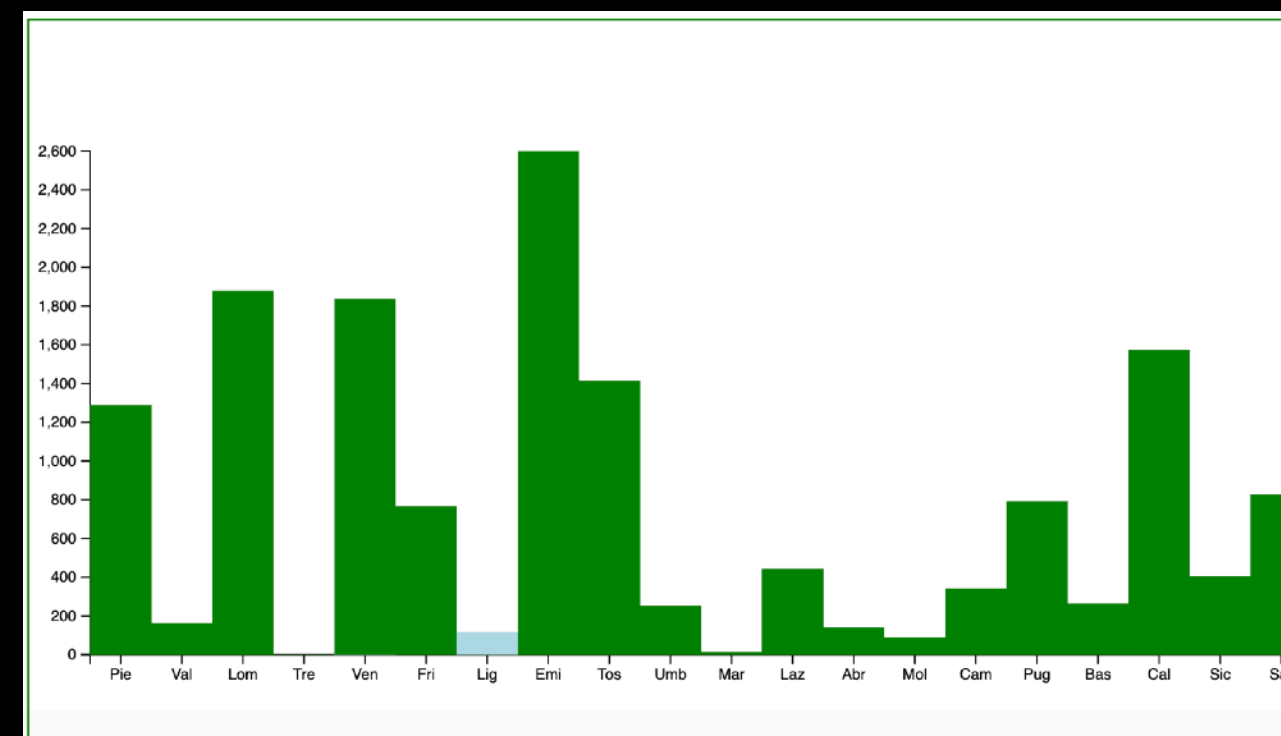
Scales allow to visualize data through:

size

position



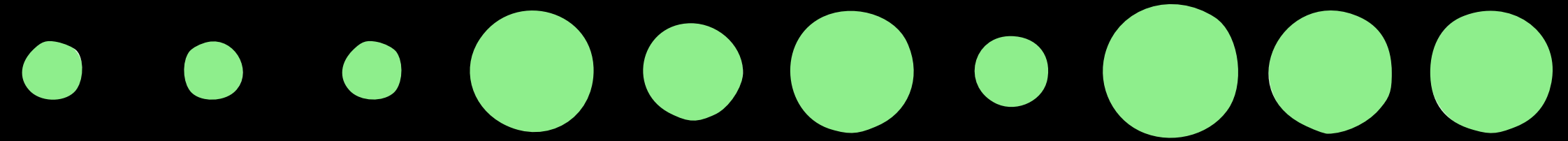
(In order to fit in your graphic view)



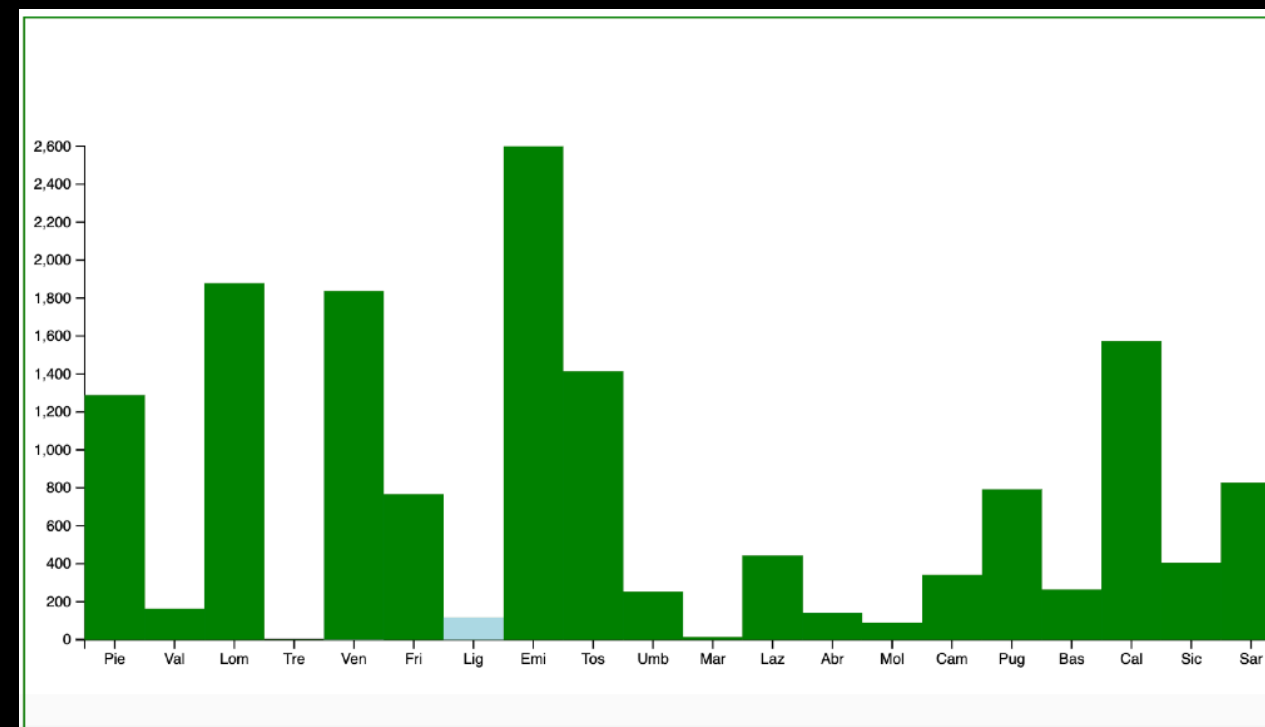
Scales – the purpose

Scales allow to visualize data through:

size



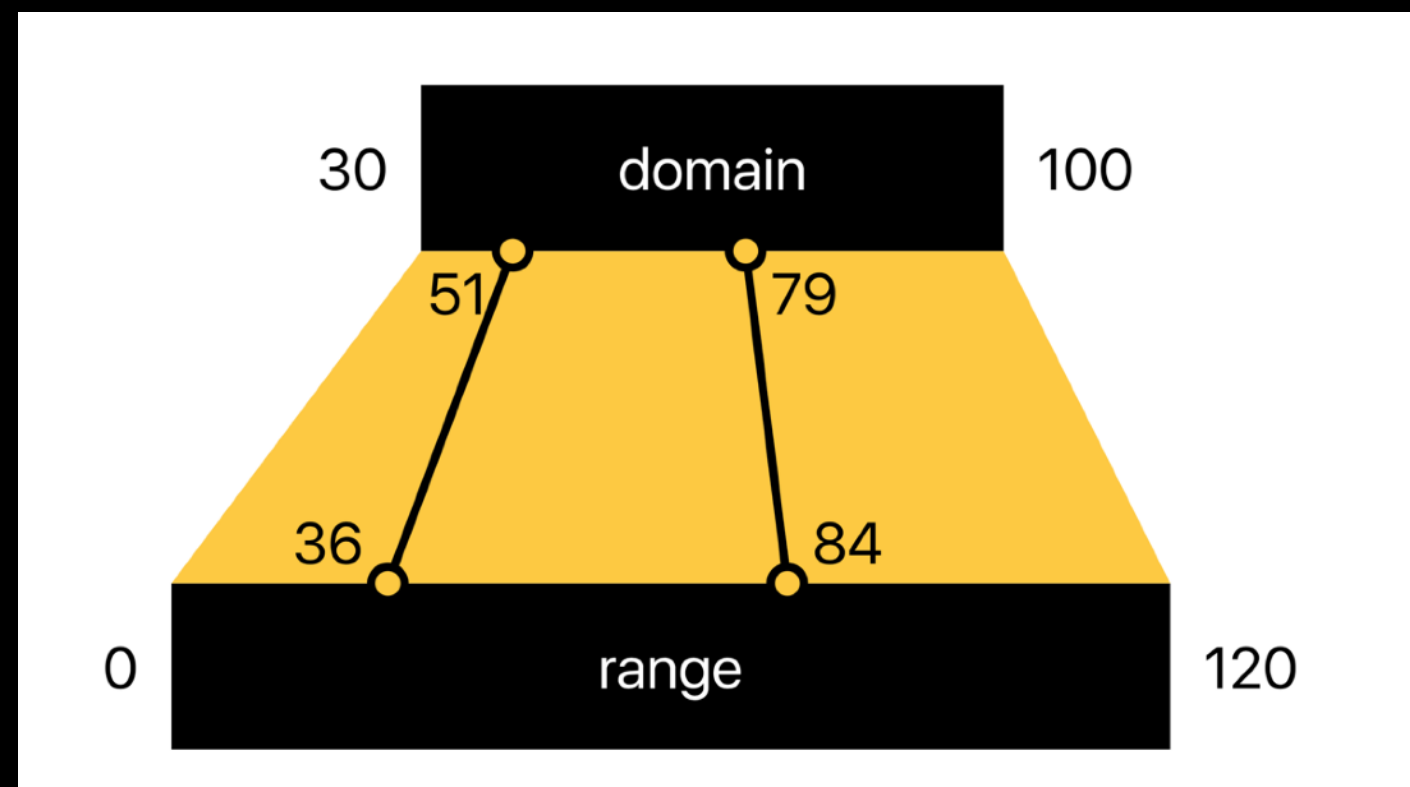
position



and **color**



Scales – the abstraction



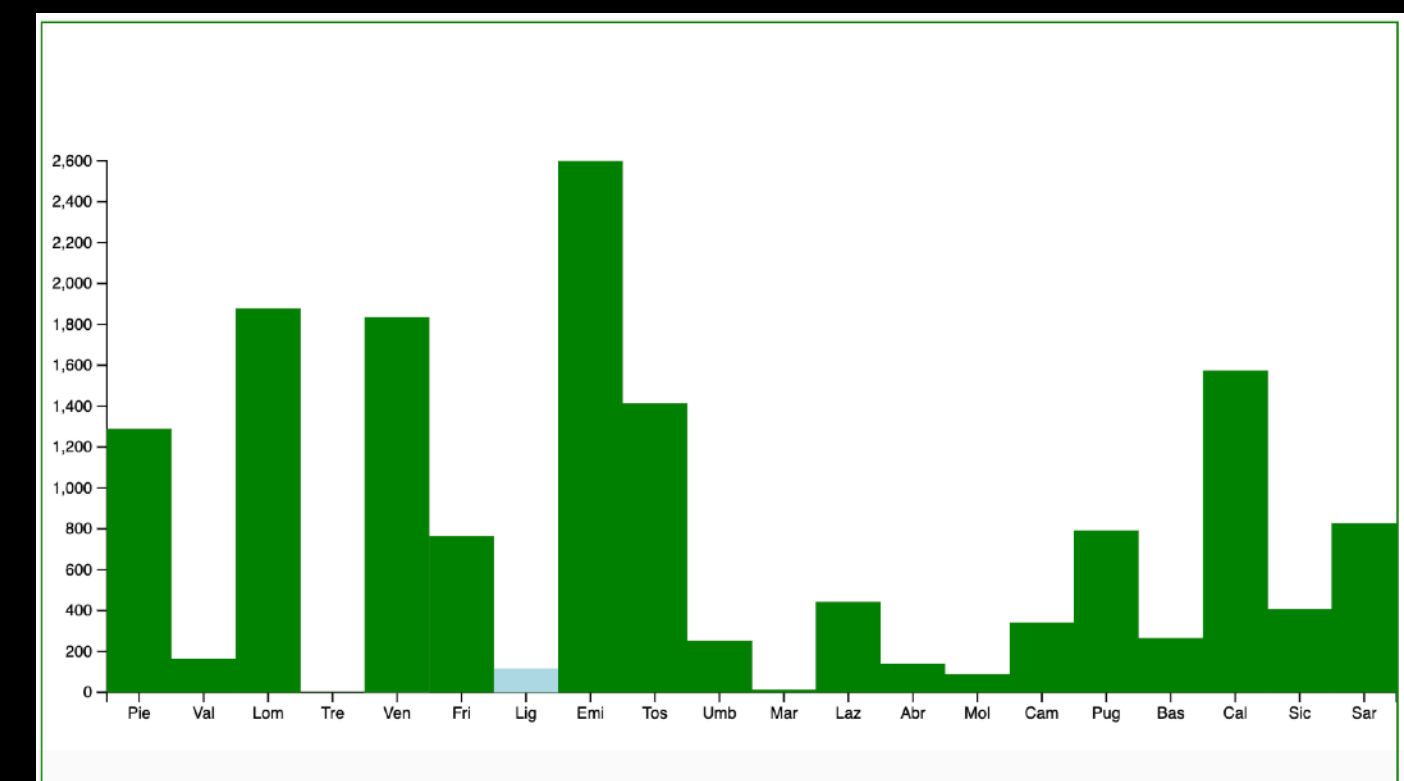
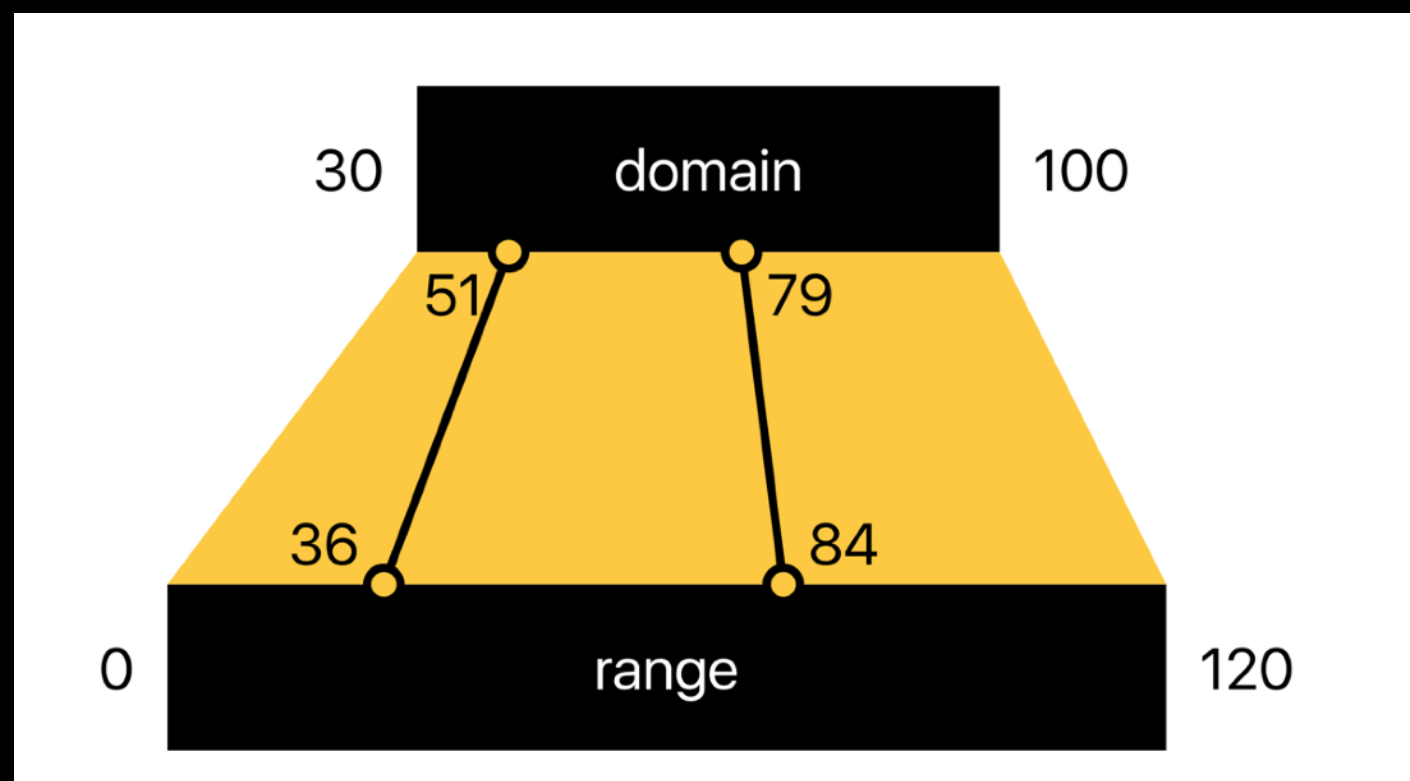
A **scale** is a **function**, that – given the values you want to display – helps you calculate:

- where the elements, representing each value are positioned
- how big they should be

It's all about numbers

Scales calculate how data is displayed on the screen:

- as a number on an axis
- as the size of an element of the DOM (that is a number)
- as the color of an element of the DOM (that is also a combination of numbers)

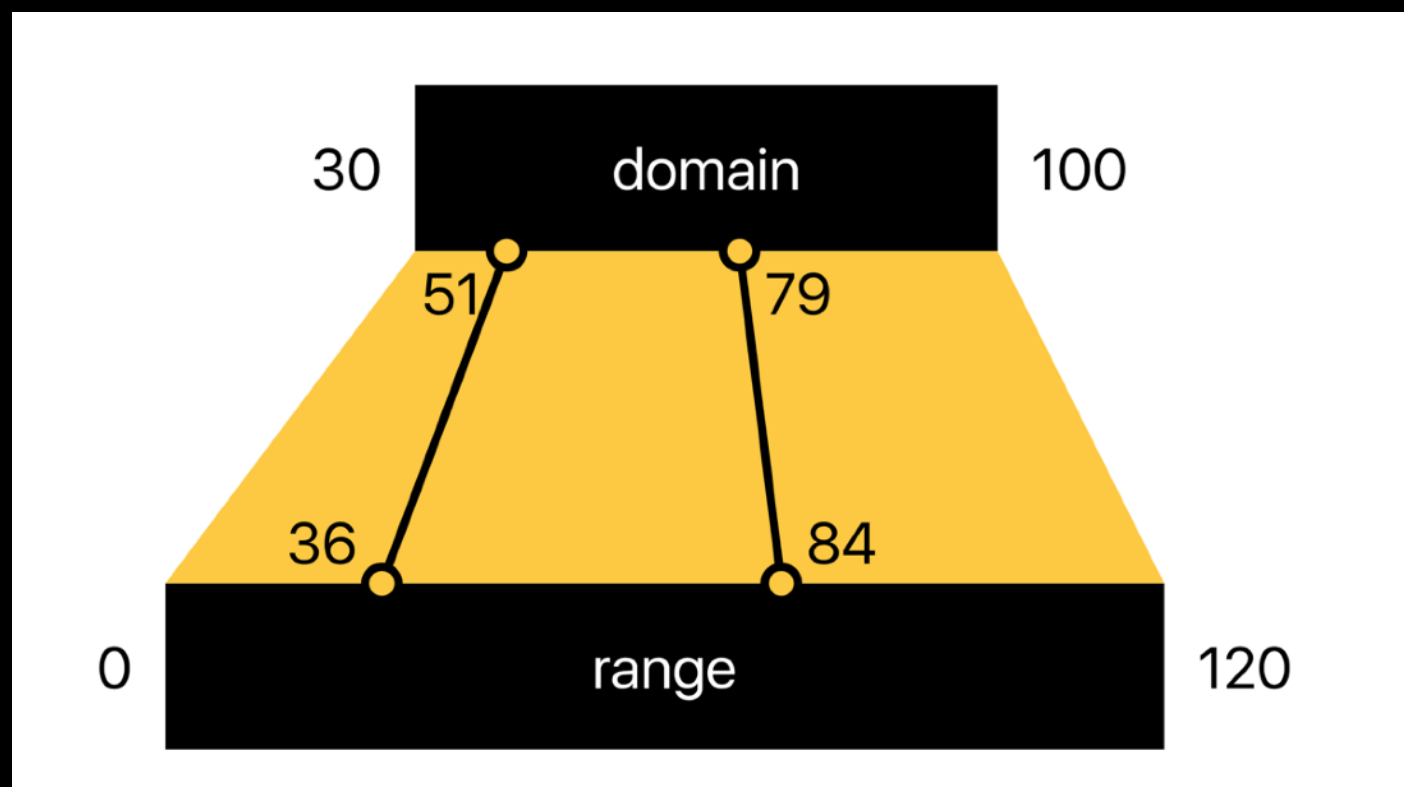


Scales—the Math (vocabulary, part 1)

A **scale** is a **function**, that – given a value you want to display – helps you calculate how the value is displayed on the screen

The set of values you want to display is called the **domain** of the scale function.

How you choose to visualize these values (size, position on on axis, color etc.) can also be seen as a set of numerical values. We call it the **range** of the scale function.



Scale examples

color scale

```
var color = d3.scaleLinear()  
  .domain([10, 100]) //data  
  .range(['brown', 'steelblue']) //colorrange
```

```
color(20) // Output: '#9a3439'
```

```
color(50) // Output: '#7b5167'
```

Scale examples

time scale

MDN reference: Date constructor

```
new Date(year, monthIndex, day)
```

```
new Date(year, monthIndex, day, hours)
```

```
var time = d3.scaleTime()  
    .domain([new Date(2000, 0, 1), new Date(2000, 0, 2)])  
    .range([0, 960])
```

```
time(new Date(2000, 0, 1, 5)) // Output: 200
```

```
time(new Date(2000, 0, 1, 16)) // Output: 640
```

Scale examples

```
var scale = d3.scaleLinear()  
  .domain([10, 130])  
  .range([0, 960])  
  
scale(-10) // Output: -160, outside range  
  
scale.clamp(true)  
scale(-10) // Output: 0, clamped to range
```

clamping scales


By default, D3 scales will try to use the still return a scaled value if the data you give it is outside the domain. This could be a weird outlier in your dataset for example or just a bug in your API.

If you want your scale to always stay within the range, you can add `.clamp(true)` to your scale function.

Choosing scales – (vocabulary, part 2)

variable	What do I want to display	How do I want to display it
	Domain	Range
X-coördinate	0..11 passengers	0..800 px
(Point scale)		Equally spaced points
Size (area) of a circle	0..11 passengers	0..45 px diameter
(scale Sqr)		
Color of a car	0..11 passengers	['red', 'orange', 'green']
(Scale quantile)	Array of values	0..3 red, 4..6 orange enz.

Hands-on...

- Werk in tweetallen
- Oefen met Joins, Scales (en Axes):
https://codepen.io/Laura_B/pen/KKOBMgL
- Maak een **point scale** die er voor zorgt dat, hoeveel ritten er ook worden afgebeeld door een auto, ze altijd passen in een rij van 800px
- Maak een **kleurscale** die de user laat zien hoe rendabel de rit is, bijvoorbeeld:

- Zie: <https://www.d3indepth.com/scales/>

Wij zijn benieuwd!!!!

Schedule

1. Recap
2. Scales in D3
- 3. Drawing an axis**



Axes & Ticks

4 different types of Axes:
axisLeft, axisRight, axisTop,
axisBottom

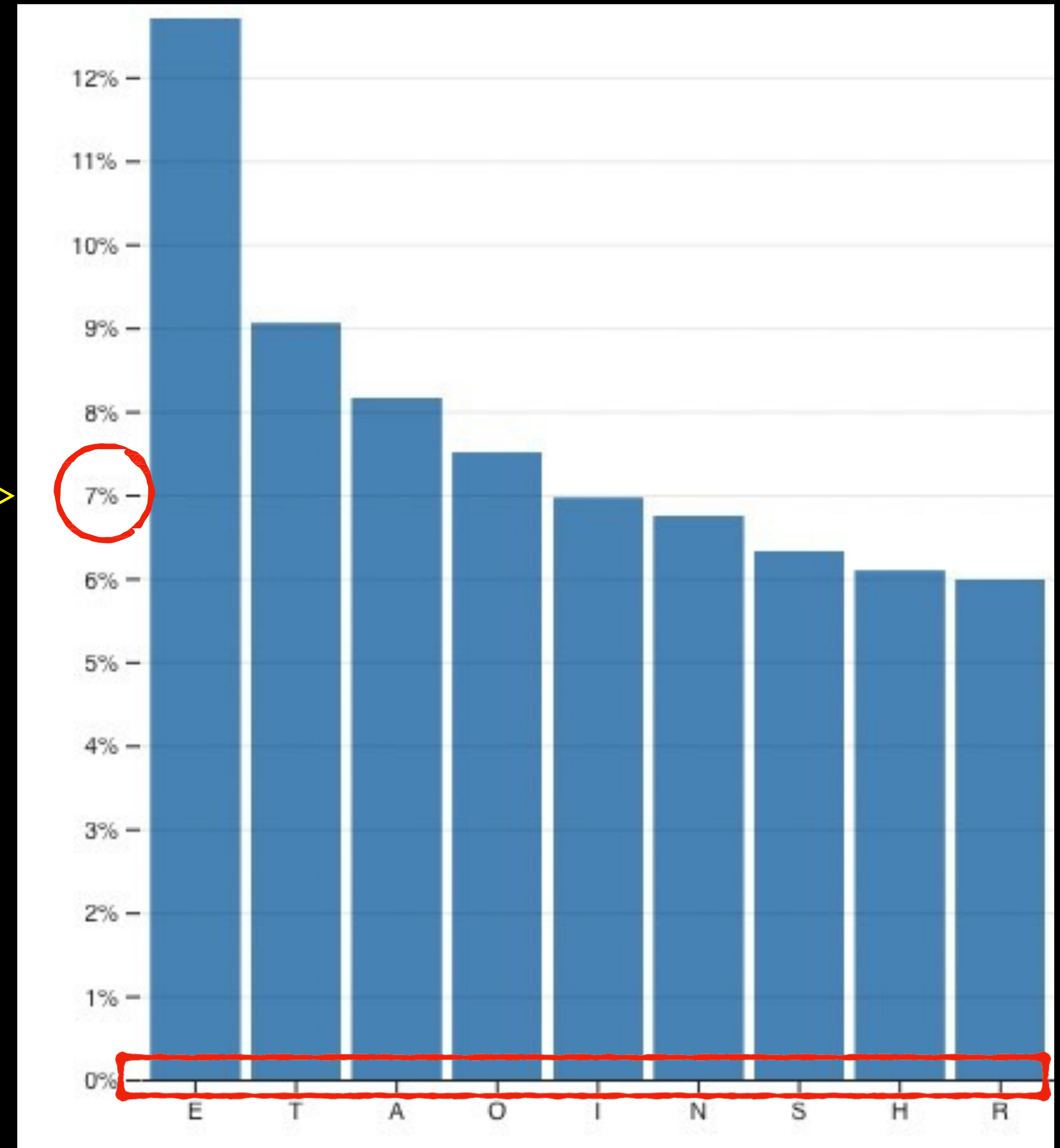
Axes refer to **scales**
that should have been
declared previously

```
24 const myAxisBottom = d3
25   .axisBottom(myScaleBand)
26   .tickFormat(d => d)
27
```

Ticks are gathered from data,
with accessor functions

Ticks ->

Axes ->



Axes & Ticks

Axes should be **positioned**

Different strategies:

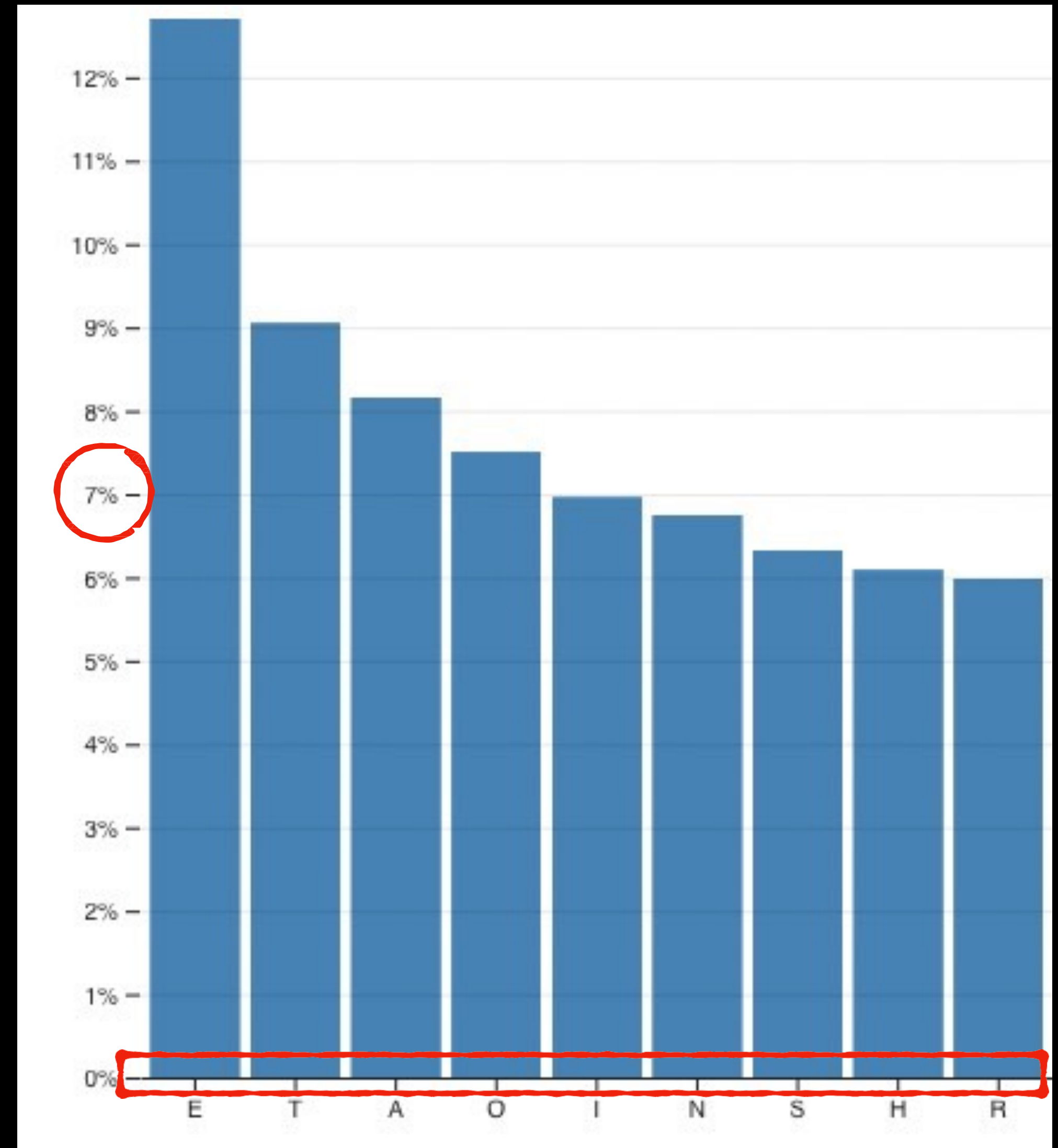
```
27 <svg id="axis">
28   <g transform="translate(50,0)"></g>
29
30 </svg>
31
```

In **HTML**: SVG el. have a *transform* attribute. Its syntax looks like the CSS property - but is different

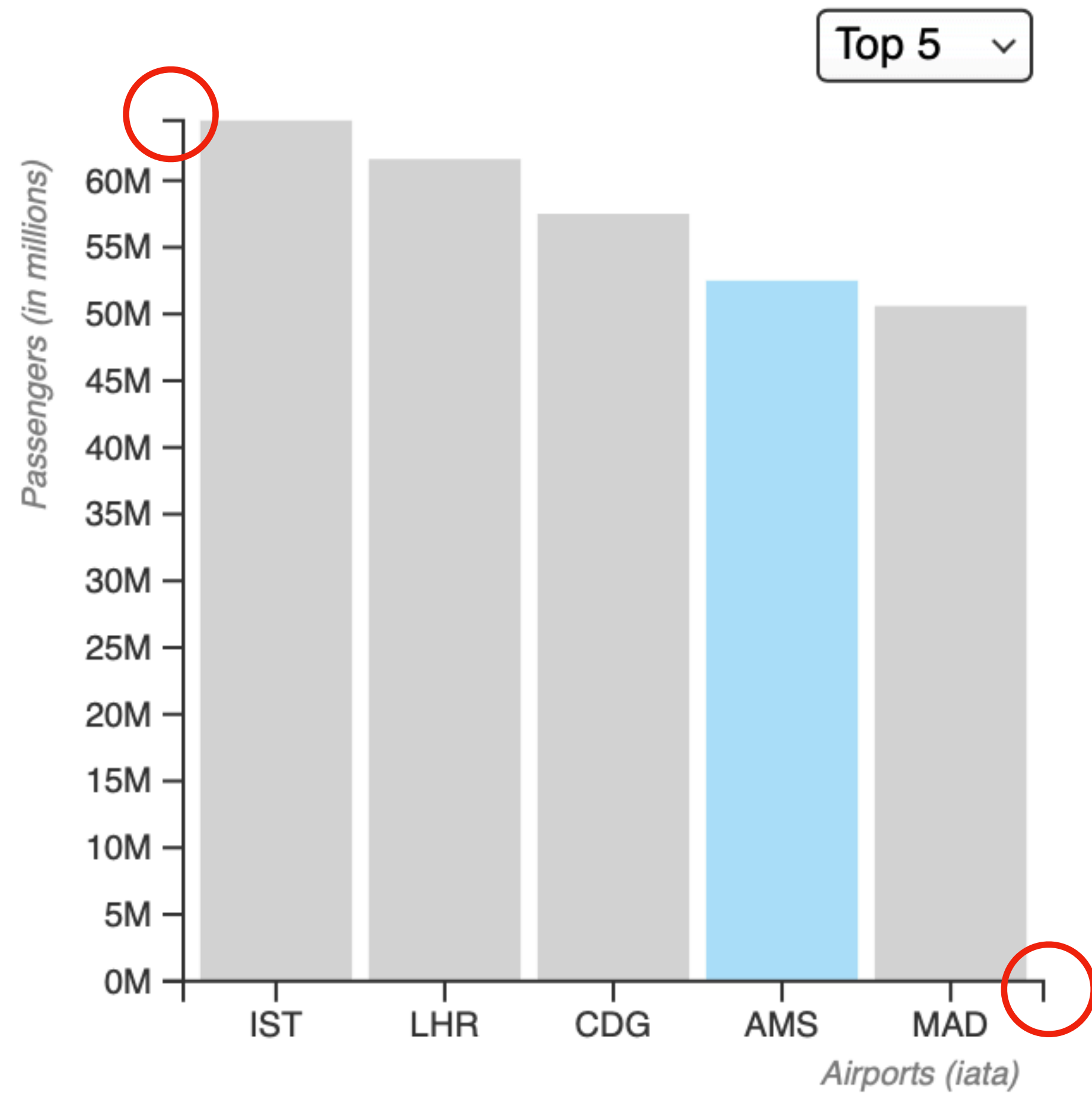
In **d3**: using
.attr("transform",...)

Avoid **CSS**

Axis-→



Axes



Sometimes your data doesn't fit on axes perfectly and you might see an extra tick without a label.

You can ask D3 to solve this by adding `.nice()` to your axis function

Uncaught SyntaxError
Unexpected end of input