Data Journalism

Introduction to D3

Graphs

Angelica Lo Duca
angelica.loduca@iit.cnr.it

D3 (Data Driven Documents)

A javascript library for manipulating documents based on data.

D3 is not a chart library, but it is a general purpose visualization library

Prerequisites

What's the Difference?

HTML

Hypertext Markup Language

Create the structure

- Controls the layout of the content
- Provides structure for the web page design
- The fundamental building block of any web page

CSS

Cascading Style Sheet

Stylize the website

- Applies style to the web page elements
- Targets various screen sizes to make web pages responsive
- Primarily handles the "look and feel" of a web page

Javascript

Increase interactivity

- Adds interactivity to a web page
- Handles complex functions and features
- Programmatic code which enhances functionality

HTML

the actual
content of a page

CSS

look of the page
(color, style)

JavaScript

easily control
and alter HTML

Example of HTML page

```
<html>
    <head>
        <title>Hello World</title>
    </head>
    <body>
        <p>Example of a paragraph</p>
    </body>
</html>
```

How to include D3.js

```
<html>
    <head>
        <title>Hello World</title>
        <script src="https://d3js.org/d3.v5.min.js"></script>
    </head>
    <body>
        <p>Example of a paragraph</p>
    <body>
<html>
```

Selections and Append

Selections

A selection is an array of elements pulled from the current document.

D3 uses CSS3 elements to select elements.

After selecting elements, you apply operators to them to do stuff. These operators can get or set attributes, styles, properties, HTML and text content.

Select an element from the document.

```
d3.select(selector), d3.select(node)
```

Select multiple elements from the document.

```
d3.selectAll(selector), d3.selectAll(nodes)
```

Example - single selection

index.html

```
<html>

      <head>

            <title>Hello World</title>

            <script
src="https://d3js.org/d3.v5.min.js"></script>

      </head>

      <body>

            <p>Example of a paragraph</p>

            <script src=”Main.js”></script>

      <body>

<html>
```

Main.js

```
const paragraph = d3.select("p");
```

Example - multiple selection

index.html

```
<html>
    <head>
        <title>Hello World</title>
        <script
src="https://d3js.org/d3.v5.min.js"
></script>
    </head>
    <body>
        <p>paragraph 1</p>
        <p>paragraph 2</p>
        <p>paragraph 3</p>
    <body>
<html>
```

Main.js

```
const paragraph = d3.selectAll("p");
```

Modify Selection

Modify attribute

```
d3.select("a").attr("name", "fred")
```

Modify style

```
d3.select("a").style("color", "red");
```

Append Element to another Element

Appends a new element with the specified name as the last child of each element in the current selection, returning a new selection containing the appended elements.

```
var svg = d3.select("svg");

var g = svg.append("g");
```

Style element with margins

```
var svg = d3.select(“svg”);

var margin = { top: 30, right: 50, bottom: 30, left: 30},

var g = svg.append(“g”)

    .attr(“transform”, “translate(“ + margin.left + “,” +
margin.top + “)”);
```

Data

Bind data to elements

```
<div id="content">

    <div></div>

    <div></div>

    <div></div>

</div>
```

```
var dataset = [ 10, 40, 20 ];

d3.select('#content')

  .selectAll('div')

  .data(dataset);
```

Note

In this example the dataset is the same length as the selection.

However, what happens if the array has more (or less) elements than the selection?

- if the array is longer than the selection there’s a shortfall of DOM elements and we need to add elements
- if the array is shorter than the selection there’s a surplus of DOM elements and we need to remove elements

.enter()

.enter identifies any DOM elements that need to be added when the joined array is longer than the selection. It's defined on an update selection (the selection returned by .data):

```
var dataset = [ 10, 40, 20 , 60, 34];

d3.select('#content')

  .selectAll('div')

  .data(dataset)

  .enter()

  .append('div'); // adds elements to the DOM
```

exit()

.exit returns an exit selection which consists of the elements that need to be removed from the DOM. It's usually followed by .remove:

```
var dataset = [ 10, 40 ];

d3.select('#content')

  .selectAll('div')

  .data(dataset)

  .exit()

  .remove();
```

Anonymous functions

```
.attr("width", function(d){})
```

For each of the elements in our selection, d represents the bound data point.

Example

```
var dataset = [ 10, 40, 20 , 60, 34];

d3.select('#content')
   .selectAll('div')
   .data(dataset)
   .enter()
   .append('div')
   .attr("width", function(d){ return d; })
```

Fetch external data

`d3.csv()` - load a CSV file

`d3.json()` - load a JSON file

Example

```
d3.csv('yourcsv.csv', function(d) {
     // preprocess your data
     return {Country: d.Country, Value: +d.Value} // +d converts to numeric
};)
  .then(function(data) {
      // do some stuff
  })
  .catch(function(error){
     // handle error
  })
```

Scales

Scales

Scales are functions that map from an input domain to an output range.

Scales are mainly used for transforming data values to visual variables such as position, length and colour.

Domain and Range

```
var dataset = [ 100, 200, 300, 400, 500 ];
```

A scale's **input domain** is the range of possible input data values. Given the apples data above, appropriate input domains would be either 100 and 500 (the minimum and maximum values of the data set) or zero and 500.

A scale's **output range** is the range of possible output values, commonly used as display values in pixel units. The output range is completely up to you, as the information designer. If you decide the shortest apple-bar will be 10 pixels tall, and the tallest will be 350 pixels tall, then you could set an output range of 10 and 350.

* source from here

Types of scales

D3 has around 12 different scale types (scaleLinear, scalePow, scaleQuantise, scaleOrdinal etc.) which can be classified into 3 groups:

- scales with continuous input and continuous output
- scales with continuous input and discrete output
- scales with discrete input and discrete output

* source from here

Scales with continuous input and continuous output

Linear Scale

Log Scale

Time Scale

...

Linear Scale

```
var scale1 = d3.scaleLinear()
             .domain([0, 500])
             .range([10, 350]);
```

This scale accepts inputs from 0 to 500 (domain) and maps them to outputs between 10 and 350.

OR

```
var Scale2 = d3.scaleLinear()
             .domain([0, d3.max(dataset, function(d) { return d[0]; })])
             .range([10, 350]);
```

Time Scale

```
var scale = d3.scaleTime()
    .domain([new Date(2021, 05, 1), new Date(2021, 05, 31)])
    .range(0,100);
```

Log Scale

```
var scale1 = d3.scaleLog()
             .domain([0, 500])
             .range([10, 350]);
```

Scales with continuous input and discrete output

Scale Quantize

...

Quantize Scale

```
var scale = d3.scaleQuantize()
   .domain([0, 100])
   .range(['lightblue', 'orange', 'lightgreen', 'pink']);
```

Scales with discrete input and discrete output

Scale Ordinal

Scale Band

...

Scale Ordinal

scaleOrdinal maps discrete values (specified by an array) to discrete values (also specified by an array). The domain array specifies the possible input values and the range array the output values. The range array will repeat if it's shorter than the domain array.

```
var myData = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug',
'Sep', 'Oct', 'Nov', 'Dec']

var scale = d3.scaleOrdinal()

  .domain(myData)

  .range(['black', '#ccc', '#ccc'])
```

Scale Band

When creating bar charts scaleBand helps to determine the geometry of the bars, taking into account padding between each bar.

```
var bandScale = d3.scaleBand()
  .domain(['Mon', 'Tue', 'Wed', 'Thu', 'Fri'])
  .range([0, 200]);
```

Data Map

d3.map groups items together, creating a hashed array that can be accessed using handy functions like array.get(value).

```
var y = d3.scaleBand()
    .range([ 0, height ])
    .domain(data.map(function(d) { return d.my_value; }))
```

Scale Band (cont.)

Two types of padding may be configured:

- **`paddingInner()`** which specifies (as a percentage of the band width) the amount of padding between each band
- **`paddingOuter()`** which specifies (as a percentage of the band width) the amount of padding before the first band and after the last band

Shapes

Shapes in SVG (Scalable Vector Graphics)

D3 provides some functions, called generators, to draw SVG shapes:

- Line
- Curve
- Area
- ...

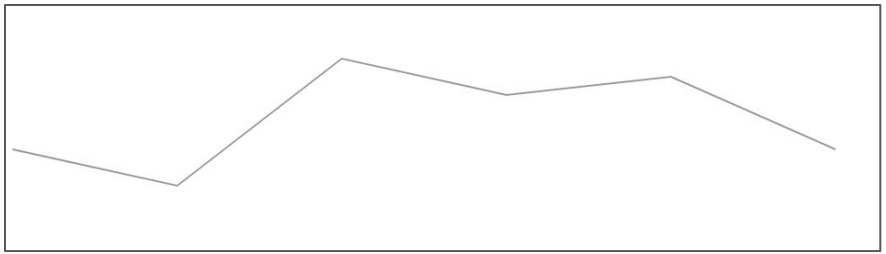
The shapes can be made up of SVG path elements.

Line Generator

```
<svg width="700" height="110">

    <path></path>

 </svg>
```

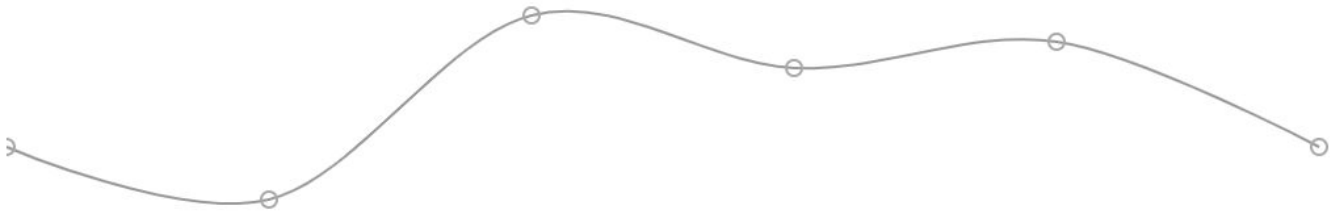


```
var lineGenerator = d3.line();
var points = [
    [0, 80],
    [100, 100],
    [200, 30],
    [300, 50],
    [400, 40],
    [500, 80]
];
var pathData = lineGenerator(points);
// Select the path element and set its
d attribute
d3.select('path').attr('d', pathData);
```

Curve Generator

We can also configure how the points are interpolated. For example we can interpolate each data point with a B-spline:

```
var lineGenerator = d3.line().curve(d3.curveCardinal);
```



Area Generator

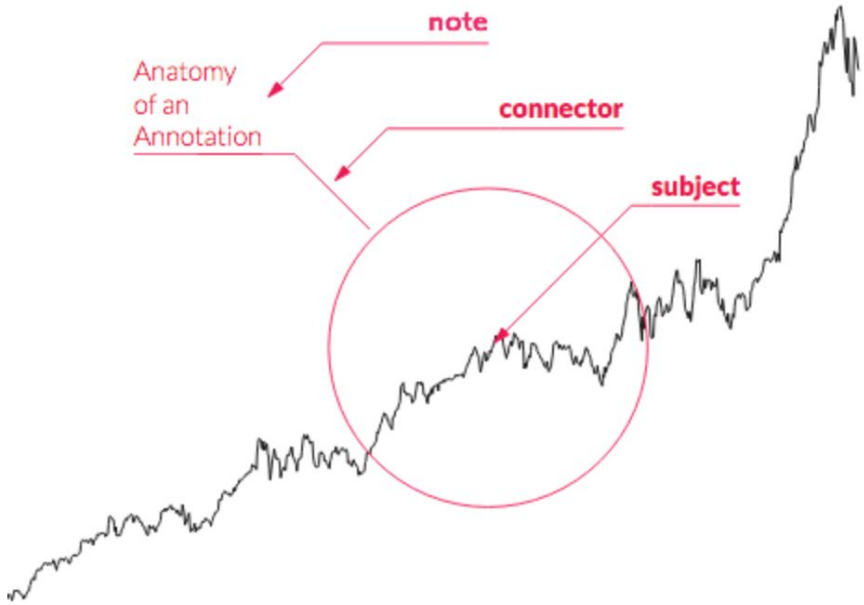
```
var areaGenerator = d3.area();
var points = [
  [0, 80],
  [100, 100],
  [200, 30],
  [300, 50],
  [400, 40],
  [500, 80]
];
var pathData = areaGenerator(points);
d3.select('g')
    .append('path')
    .attr('d', area);
```

Annotations

D3-Annotation

Annotations establish context, and direct our users to insights and anomalies.

All annotations are made of just three parts, a **note**, a **connector**, and a **subject**.



Including D3-Annotation

```
<html>
<head>
    <meta charset="utf-8">
        <script src="https://d3js.org/d3.v5.min.js"></script>
        <!-- Load d3-annotation -->
        <script
src="https://rawgit.com/susielu/d3-annotation/master/d3-anno
tation.min.js"></script>
</head>
<body>
</body>
</html>
```

Types of Annotations

```
d3.annotationLabel

d3.annotationCallout

d3.annotationCalloutElbow

d3.annotationCalloutCurve

d3.annotationCalloutCircle

d3.annotationCalloutRect

d3.annotationXYThreshold

d3.annotationBadge
```

Types of Annotations

d3.annotationLabel

d3.annotationCallout

d3.annotationCalloutElbow

d3.annotationCalloutCurve

d3.annotationCalloutCircle

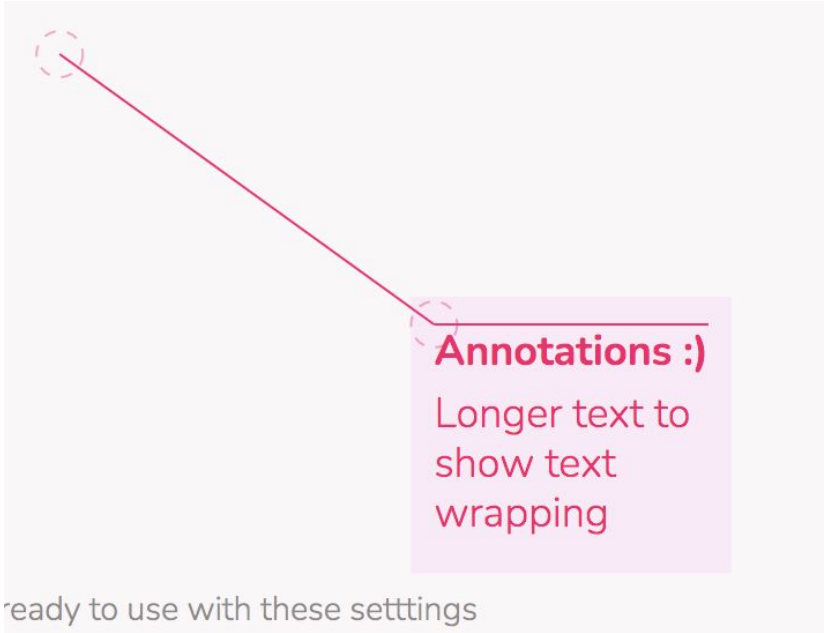
d3.annotationCalloutRect

d3.annotationXYThreshold

d3.annotationBadge

Types of Annotations

```
d3.annotationLabel
d3.annotationCallout
d3.annotationCalloutElbow
d3.annotationCalloutCurve
d3.annotationCalloutCircle
d3.annotationCalloutRect
d3.annotationXYThreshold
d3.annotationBadge
```



Types of Annotations

`d3.annotationLabel`

`d3.annotationCallout`

`d3.annotationCalloutElbow`

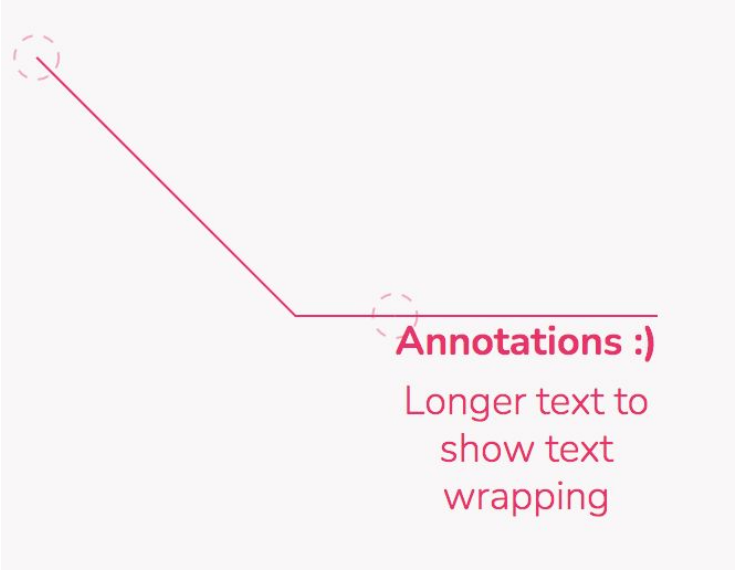
`d3.annotationCalloutCurve`

`d3.annotationCalloutCircle`

`d3.annotationCalloutRect`

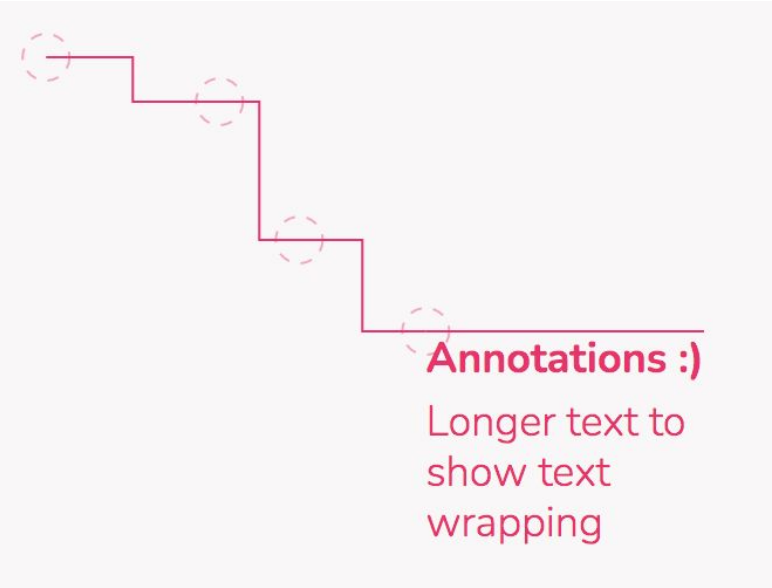
`d3.annotationXYThreshold`

`d3.annotationBadge`



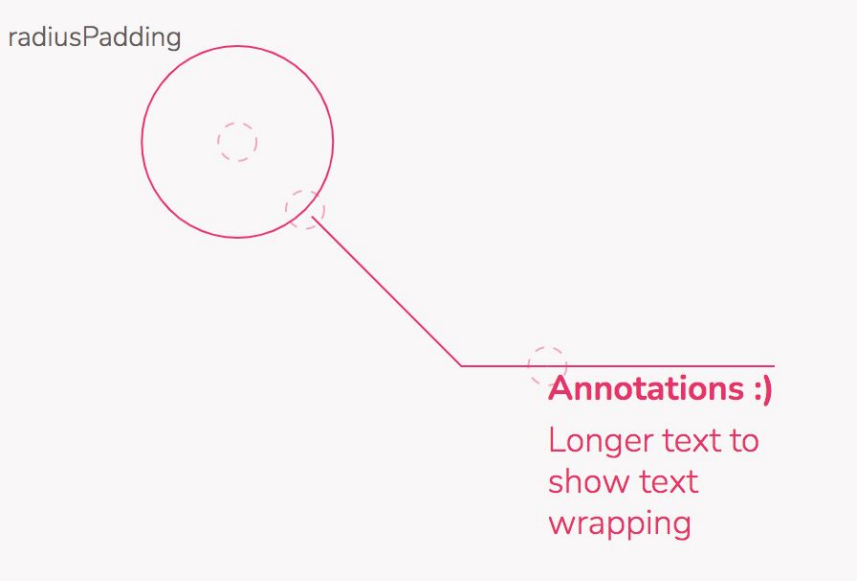
Types of Annotations

```
d3.annotationLabel
d3.annotationCallout
d3.annotationCalloutElbow
d3.annotationCalloutCurve
d3.annotationCalloutCircle
d3.annotationCalloutRect
d3.annotationXYThreshold
d3.annotationBadge
```



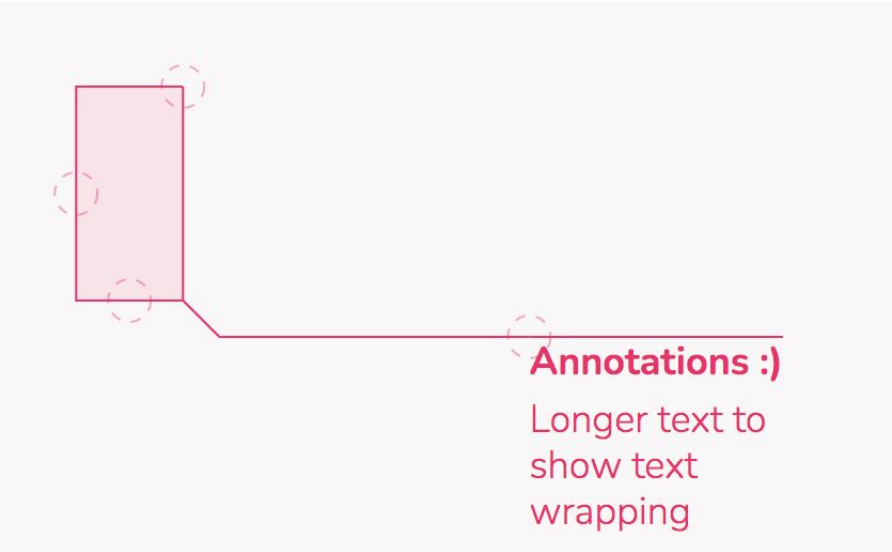
Types of Annotations

```
d3.annotationLabel
d3.annotationCallout
d3.annotationCalloutElbow
d3.annotationCalloutCurve
d3.annotationCalloutCircle
d3.annotationCalloutRect
d3.annotationXYThreshold
d3.annotationBadge
```



Types of Annotations

```
d3.annotationLabel
d3.annotationCallout
d3.annotationCalloutElbow
d3.annotationCalloutCurve
d3.annotationCalloutCircle
d3.annotationCalloutRect
d3.annotationXYThreshold
d3.annotationBadge
```



Types of Annotations

`d3.annotationLabel`

`d3.annotationCallout`

`d3.annotationCalloutElbow`

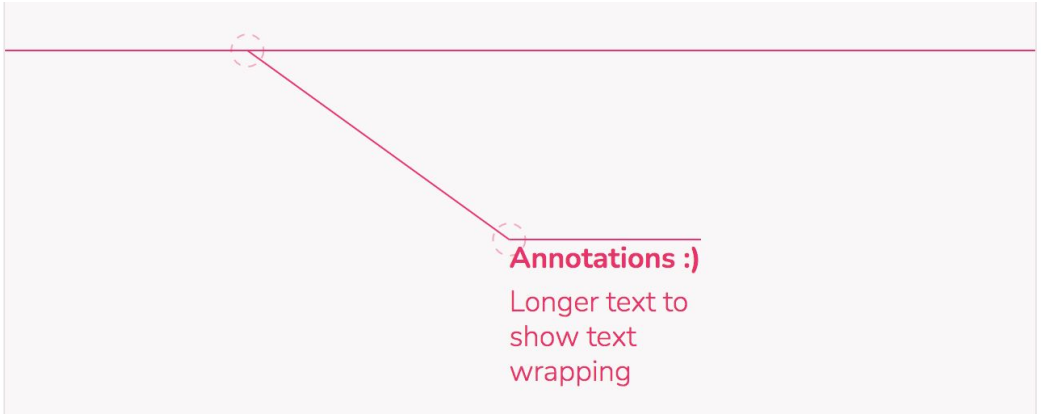
`d3.annotationCalloutCurve`

`d3.annotationCalloutCircle`

`d3.annotationCalloutRect`

`d3.annotationXYThreshold`

`d3.annotationBadge`



Types of Annotations

```
d3.annotationLabel
d3.annotationCallout
d3.annotationCalloutElbow
d3.annotationCalloutCurve
d3.annotationCalloutCircle
d3.annotationCalloutRect
d3.annotationXYThreshold
d3.annotationBadge
```

Example

```
const annotations = [
  {
    note: {
      label: "My Annotation",
      title: "My Table",
    },

    x: x(2500), // position with respect the x axis
    y: y(100),  // position with respect the y axis
  }
]
```

Example (cont.)

```
// Add annotation to the chart
const makeAnnotations = d3.annotation()
  .annotations(annotations)


  svg.append("g")
  .call(makeAnnotations)
```

Graphs

Axis

Regardless of orientation, axes are always rendered at the origin. To change the position of the axis with respect to the chart, specify a transform attribute on the containing element.

```
var axis = d3.axisLeft(scale);

d3.select("svg").append("g").call(axis);
```

Position of labels of an axis

```
d3.axisRight(scale)

d3.axisLeft(scale)

d3.axisTop(scale)

d3.axisBottom(scale)
```

Example

```
// set the dimensions and margins of the graph
var margin = {top: 10, right: 30, bottom: 30, left: 60},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;
```

Example

```
var x = d3.scaleLinear()
        .domain([0, 13000])
        .range([ 0, 100]);


  svg.append("g")
    .attr("transform", "translate(0," + height + ")")
    .call(d3.axisBottom(x))
```

Text Label for the x axis

```
svg.append("text")
    .attr("transform",
          "translate(" + (width/2) + " ," +
                         (height + margin.top + 20) + ")")
    .style("text-anchor", "middle")
    .text("Date");
```

Text Label for the y axis

```
svg.append("text")
      .attr("transform", "rotate(-90)")
      .attr("y", 0 - margin.left)
      .attr("x",0 - (height / 2))
      .attr("dy", "1em")
      .style("text-anchor", "middle")
      .text("Value");
```

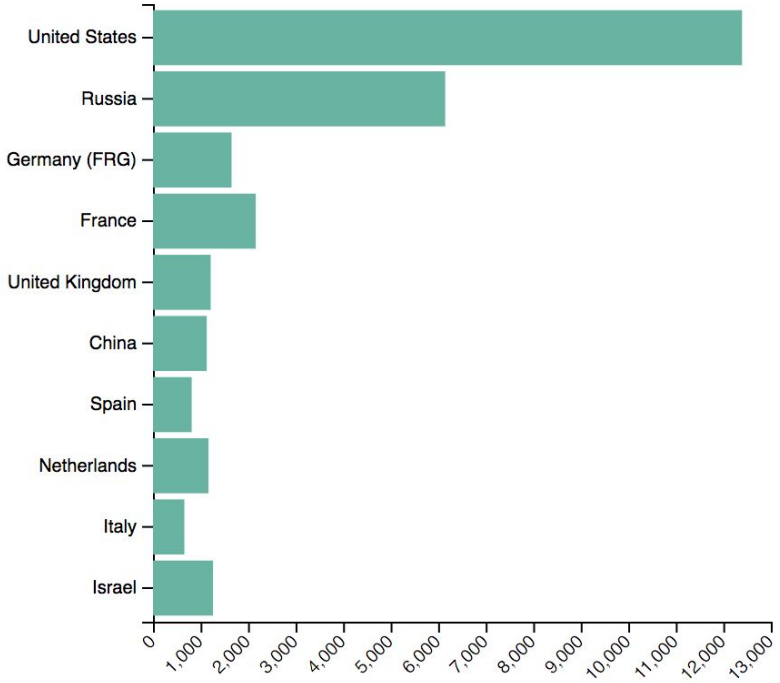
Types of graphs

Bar Chart

Line Chart

Other Graphs

Bar chart - Bad Chart



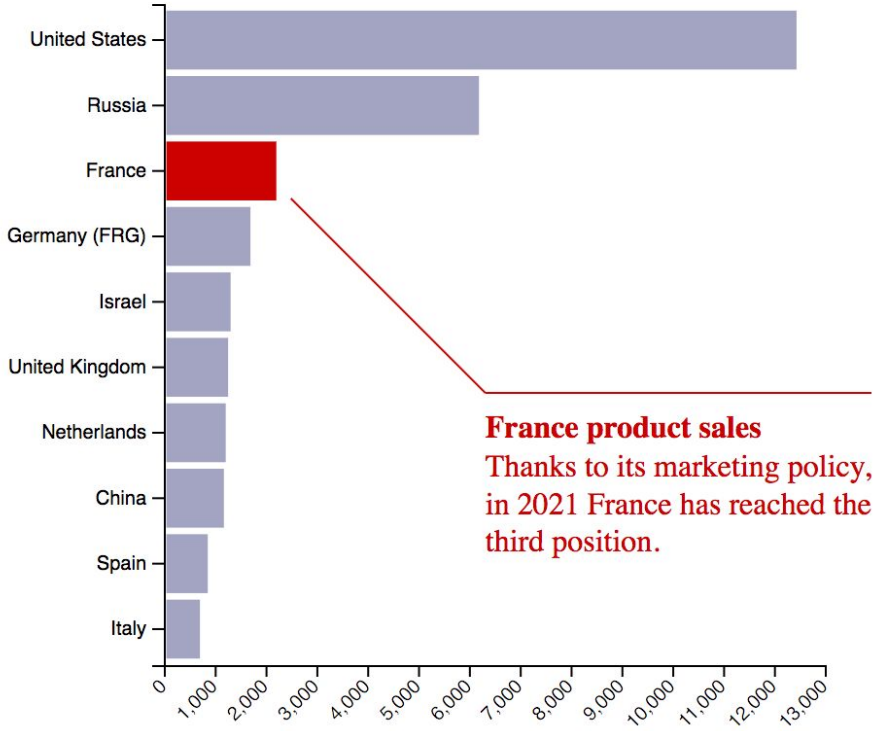
France Product Sales

Thanks to its marketing policy, in 2021 France has reached the third position.

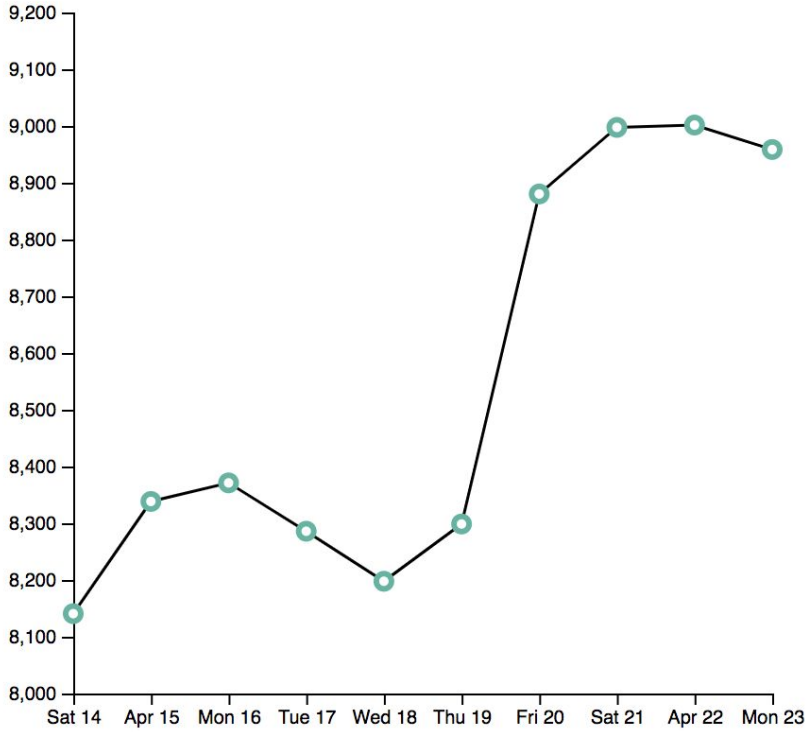
Where is France?

For the reader finding that France has reached the third position is difficult.

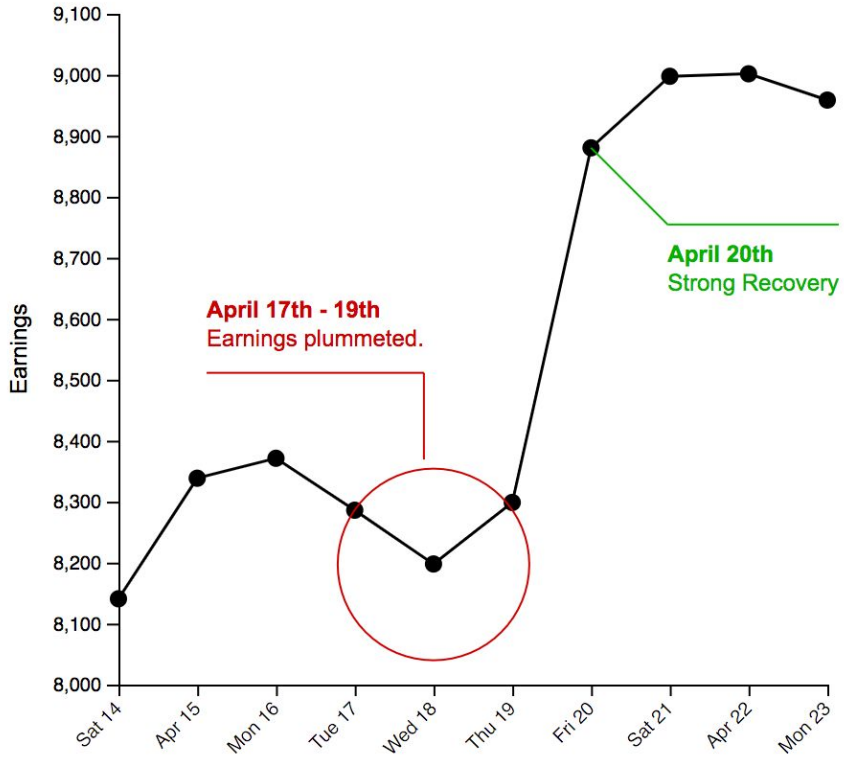
Bar chart - Chart with a narrative



Line Chart without a context



Line Chart with a narrative



References

https://www.d3indepth.com/