



Übung Open Data:

Skalen und Achsen mit D3.js realisieren

Termin 7, 14. April 2016

Dr. Matthias Stürmer und Prof. Dr. Thomas Myrach

Universität Bern, Institut für Wirtschaftsinformatik

Abteilung Informationsmanagement

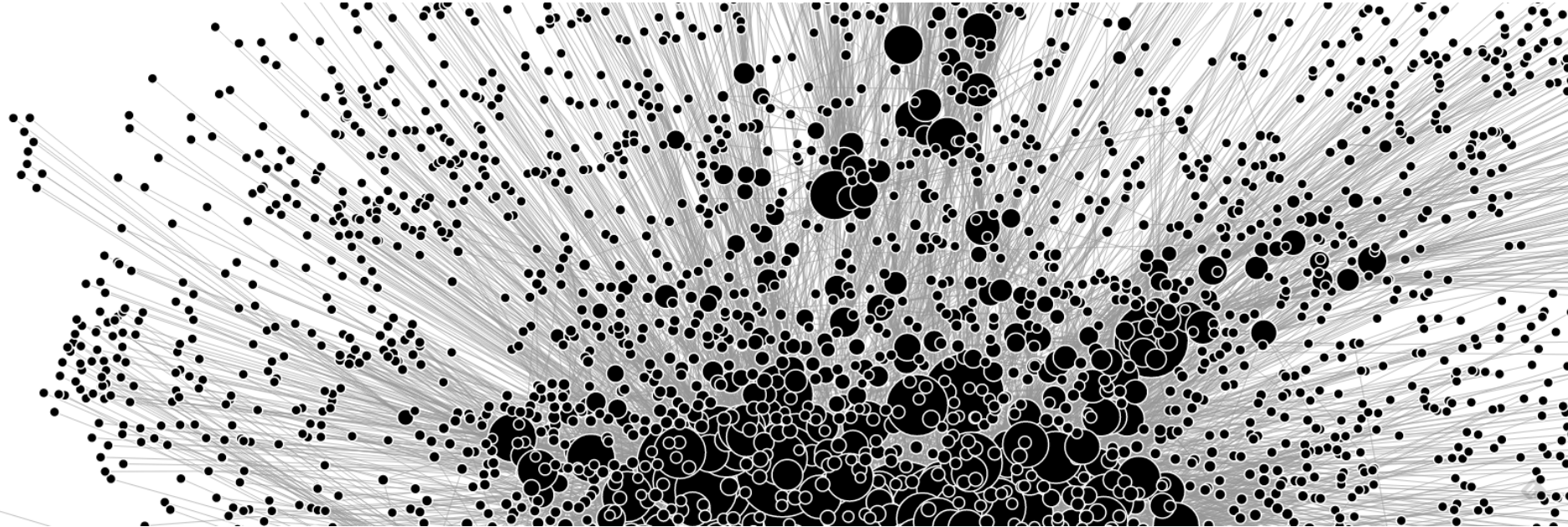
Forschungsstelle Digitale Nachhaltigkeit

Terminübersicht Übungen

- > 25.02.2016: Informationen zu den Übungen, App-Demos & Einführung in Tools
- > 03.03.2016: Einführung Web-Programmierung
- > 10.03.2016: Open Data Speed Dating
- > 17.03.2016: Einführung D3.js & Daten einbinden in D3.js
- > 24.03.2016: Anpassen von bestehenden Apps & Bibliotheken, die D3.js verwenden
- > 31.03.2016: Osterferien
- > 07.04.2016: Daten visualisieren & Layouts
- > **14.04.2016: Skalen und Achsen & Responsive Design**
- > 21.04.2016: User Experience, Usability Patterns
- > 28.04.2016: Zwischenpräsentation & Datenaktualisierung und Transitionen
- > 05.05.2016: Auffahrt
- > 12.05.2016: Interactivity & Geomapping
- > 19.05.2016: 3D Web-Programmierung mit Three.js & Programming Coaching
- > 26.05.2016: Abschlusspräsentationen
- > 02.06.2016: frei

Agenda

1. **Skalen**
2. Achsen



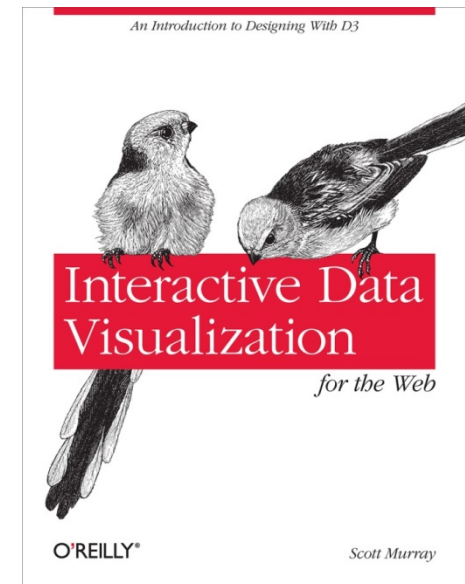
Interactive Data Visualization for the Web

Chapter 7. Scales:

> <http://chimera.labs.oreilly.com/books/123000000345/ch07.html>

Chapter 8. Axes:

> <http://chimera.labs.oreilly.com/books/123000000345/ch08.html>



Scales vs. Axes

> Scales

- `scale()` function: pass it a data value, it returns a scaled output value
- Scales are functions that map from an input domain to an output range
- A scale is a mathematical relationship, with no direct visual output
- Linear, ordinal, logarithmic, square root scales
- Used with domains and ranges

> Axes

- `axis()` function: they generate SVG elements
- Axes don't return a value, but generate the visual elements of the axis, including lines, labels, and ticks
- Linear, ordinal, logarithmic, square root Axes
- Each axis needs to be told on what *scale* to operate

API Reference of Scales

GitHub

This repository ▾ Search or type a command ⓘ

Explore Features Enterprise Blog

Sign up Sign in

PUBLIC

mbostock / d3

★ Star 25,424

🍴 Fork 5,484

Quantitative Scales

mbostock edited this page a month ago · 30 commits

Wiki ▸ API Reference ▸ Scales ▸ Quantitative Scales

Scales are functions that map from an input domain to an output range. **Quantitative** scales have a continuous domain, such as the set of real numbers, or dates. There are also **ordinal scales**, which have a discrete domain, such as a set of names or categories. Scales are an optional feature in D3; you don't have to use them, if you prefer to do the math yourself. However, using scales can greatly simplify the code needed to map a dimension of data to a visual representation.

A scale object, such as that returned by `d3.scale.linear`, is both an object and a function. That is: you can call the scale like any other function, and the scale has additional methods that change its behavior. Like other classes in D3, scales follow the method chaining pattern where setter methods return the scale itself, allowing multiple setters to be invoked in a concise statement.

Linear Scales

Linear scales are the most common scale, and a good default choice to map a continuous input domain to a continuous output range. The mapping is *linear* in that the output range value y can be

Pages (65)

Find a Page...

3.0

3.1

API Reference

API Reference (русскоязычная версия)

API参考

Arrays

Behaviors

Bundle Layout

Chord Layout

Cluster Layout

CN Home

Link: <https://github.com/mbostock/d3/wiki/Quantitative-Scales>

FS 2016

API Reference of Axes

GitHub

This repository ▾

Search or type a command 🔍

Explore

Features


Enterprise

Blog

Sign up

Sign in

PUBLIC

 mbostock / d3

★ Star

25,424

Fork

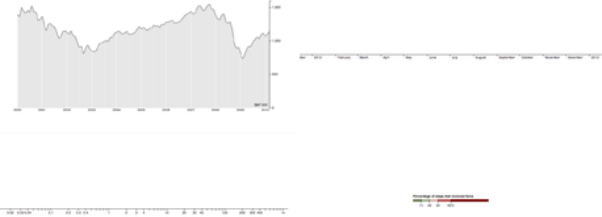
5,484

◀ SVG Axes

bruceharris edited this page 4 months ago · 25 commits

Wiki ▸ API Reference ▸ SVG ▸ SVG Axes

D3's [axis component](#) displays reference lines for scales automatically. This lets you focus on displaying the data, while the axis component takes care of the tedious task of drawing axes and labeled ticks.



▼ Pages (65)

Find a Page...

3.0

3.1

API Reference

API Reference (русскаяязычная версия)

API参考

Arrays

Behaviors

Bundle Layout

Chord Layout

Cluster Layout

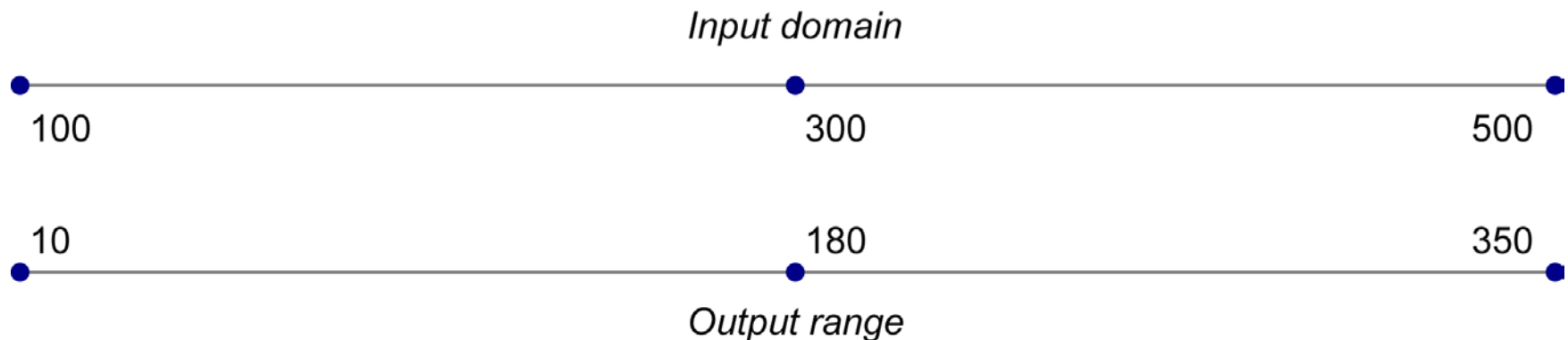
Link: <https://github.com/mbostock/d3/wiki/SVG-Axes>

Domains and Ranges

- > A scale's *input domain* is the domain of possible input data values.
- > A scale's *output range* is the range of possible output values, pixels.

Example:

A scale with *input domain* of [100,500] and *output range* of [10,350]:



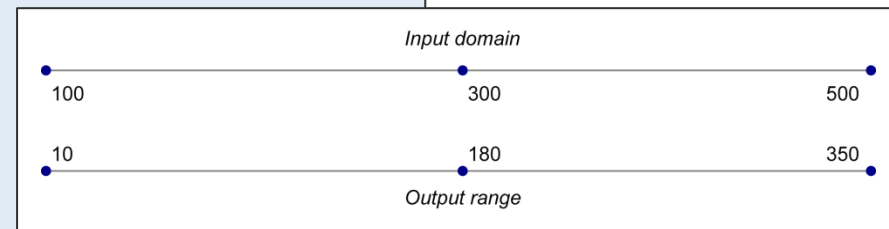
Creating a Scale

Einer Variable eine Skala zuweisen:

```
var scale = d3.scale.linear()  
                .domain([100, 500])  
                .range([10, 350]);
```

Folgende Werte können in der Konsole generiert werden:

```
scale(100); //Returns 10  
scale(300); //Returns 180  
scale(500); //Returns 350
```



d3.max()

Den Maximal-Wert bestimmen:

```
var simpleDataset = [7, 8, 4, 5, 2];  
d3.max(simpleDataset); // Returns 8
```

Maximal-Wert eines zweidimensionalen Arrays (array of arrays):

```
var dataset = [[5, 20],[480, 90],[250, 50],[100, 33],  
               [330, 95],[410, 12],[475, 44],[25, 67],  
               [85, 21], [220, 88]];  
d3.max(dataset, function(d) {  
    return d[0]; //Returns 480  
}));
```

Setting Up Dynamic Scales

Die x-Skala definieren:

```
var xScale = d3.scale.linear()  
  .domain([0, d3.max(dataset, function(d) {return d[0];})])  
  .range([0, w]);
```

- > **xScale** ist der Name der Skala
- > Domain und Range sind in zwei eckigen Klammern definiert
- > Input Domain beginnt bei 0 (könnte auch **min()** verwendet werden)
- > Input Domain endet bei höchsten Wert: **max(dataset)**
- > Output Range liegt zwischen 0 und w (width) der SVG-Fläche

Setting Up Dynamic Scales

Die y-Skala definieren:

```
var yScale = d3.scale.linear()  
  .domain([0,d3.max(dataset,function(d) {return d[1];})])  
  .range([0, h]);
```

D3.js Scatterplot (Chapter 6)

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Chapter 6 - Another Circles Example</title>
    <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
  </head>
  <body>
    <script type="text/javascript">

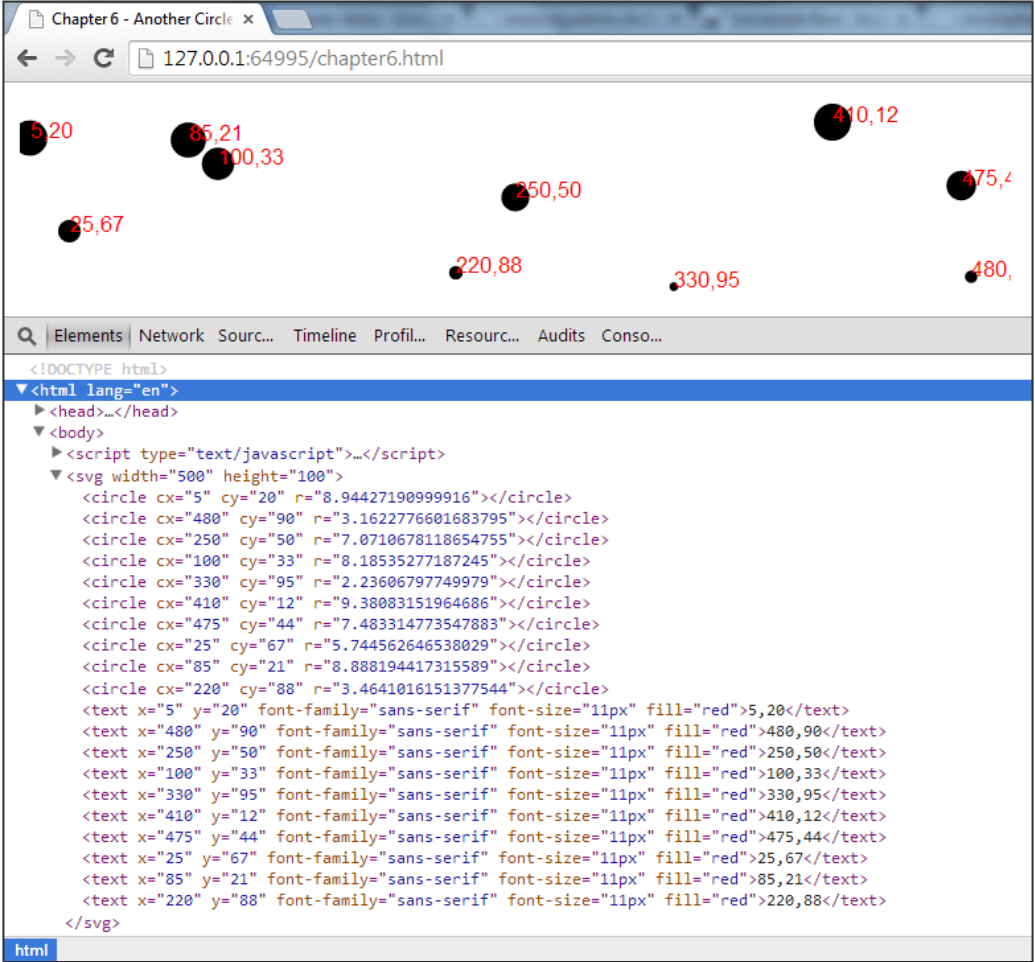
      var w = 500;
      var h = 100;
      var dataset = [
        [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
        [410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
      ];

      var svg = d3.select("body")
        .append("svg")
        .attr("width", w)
        .attr("height", h);

      svg.selectAll("circle")
        .data(dataset)
        .enter()
        .append("circle")
        .attr("cx", function(d) {
          return d[0];
        })
        .attr("cy", function(d) {
          return d[1];
        })
        .attr("r", function(d) {
          return Math.sqrt(h - d[1]);
        });

      svg.selectAll("text")
        .data(dataset)
        .enter()
        .append("text")
        .text(function(d) {
          return d[0] + "," + d[1];
        })
        .attr("x", function(d) {
          return d[0];
        })
        .attr("y", function(d) {
          return d[1];
        })
        .attr("font-family", "sans-serif")
        .attr("font-size", "11px")
        .attr("fill", "red");

    </script>
  </body>
</html>
```



Incorporating Scaled Values

Positionierung
ohne Skala:

```
.attr("cx", function(d) {  
    return d[0];  
})  
.attr("cy", function(d) {  
    return d[1];  
})
```

Positionierung
mit Skala:

```
.attr("cx", function(d) {  
    return xScale(d[0]);  
})  
.attr("cy", function(d) {  
    return yScale(d[1]);  
})
```

Incorporating Scaled Values

Text-Label
ohne Skala:

```
.attr("x", function(d) {  
    return d[0];  
})  
.attr("y", function(d) {  
    return d[1];  
})
```

Text-Label
mit Skala:

```
.attr("x", function(d) {  
    return xScale(d[0]);  
})  
.attr("y", function(d) {  
    return yScale(d[1]);  
})
```

D3.js Scatterplot mit dynamischer Skala

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>D3: Linear scales with a scatterplot</title>
    <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
    <style type="text/css">
      /* No style rules here yet */
    </style>
  </head>
  <body>
    <script type="text/javascript">

      //Width and height
      var w = 500;
      var h = 100;

      var dataset = [
        [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
        [410, 12], [475, 44], [25, 67], [85, 21], [220, 88]
      ];

      //Create scale functions
      var xScale = d3.scale.linear()
        .domain([0, d3.max(dataset, function(d) { return d[0]; })])
        .range([0, w]);

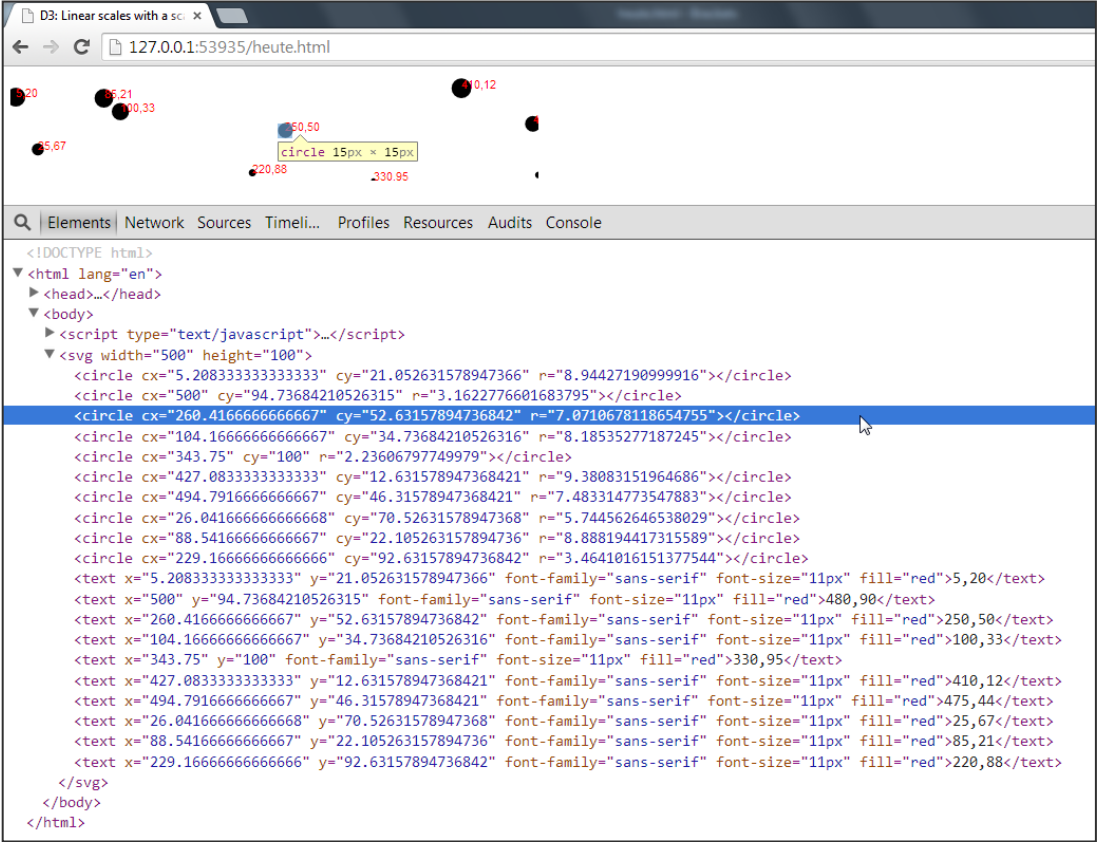
      var yScale = d3.scale.linear()
        .domain([0, d3.max(dataset, function(d) { return d[1]; })])
        .range([0, h]);

      //Create SVG element
      var svg = d3.select("body")
        .append("svg")
        .attr("width", w)
        .attr("height", h);

      svg.selectAll("circle")
        .data(dataset)
        .enter()
        .append("circle")
        .attr("cx", function(d) {
          return xScale(d[0]);
        })
        .attr("cy", function(d) {
          return yScale(d[1]);
        })
        .attr("r", function(d) {
          return Math.sqrt(h - d[1]);
        });

      svg.selectAll("text")
        .data(dataset)
        .enter()
        .append("text")
        .text(function(d) {
          return d[0] + "," + d[1];
        })
        .attr("x", function(d) {
          return xScale(d[0]);
        })
        .attr("y", function(d) {
          return yScale(d[1]);
        })
        .attr("font-family", "sans-serif")
        .attr("font-size", "11px")
        .attr("fill", "red");

    </script>
  </body>
</html>
```

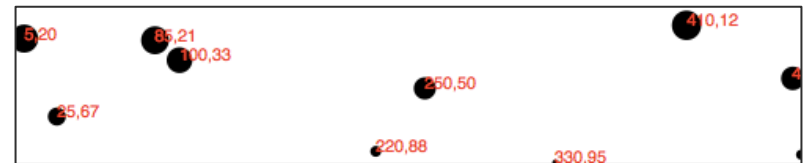


Refining the Plot

Smaller y values are at the top of the plot, and the larger y values are toward the bottom.

Original:

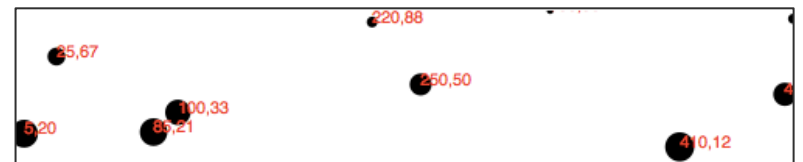
```
.range([0, h]);
```



Reverse that, so greater values are higher up.

Reversed:

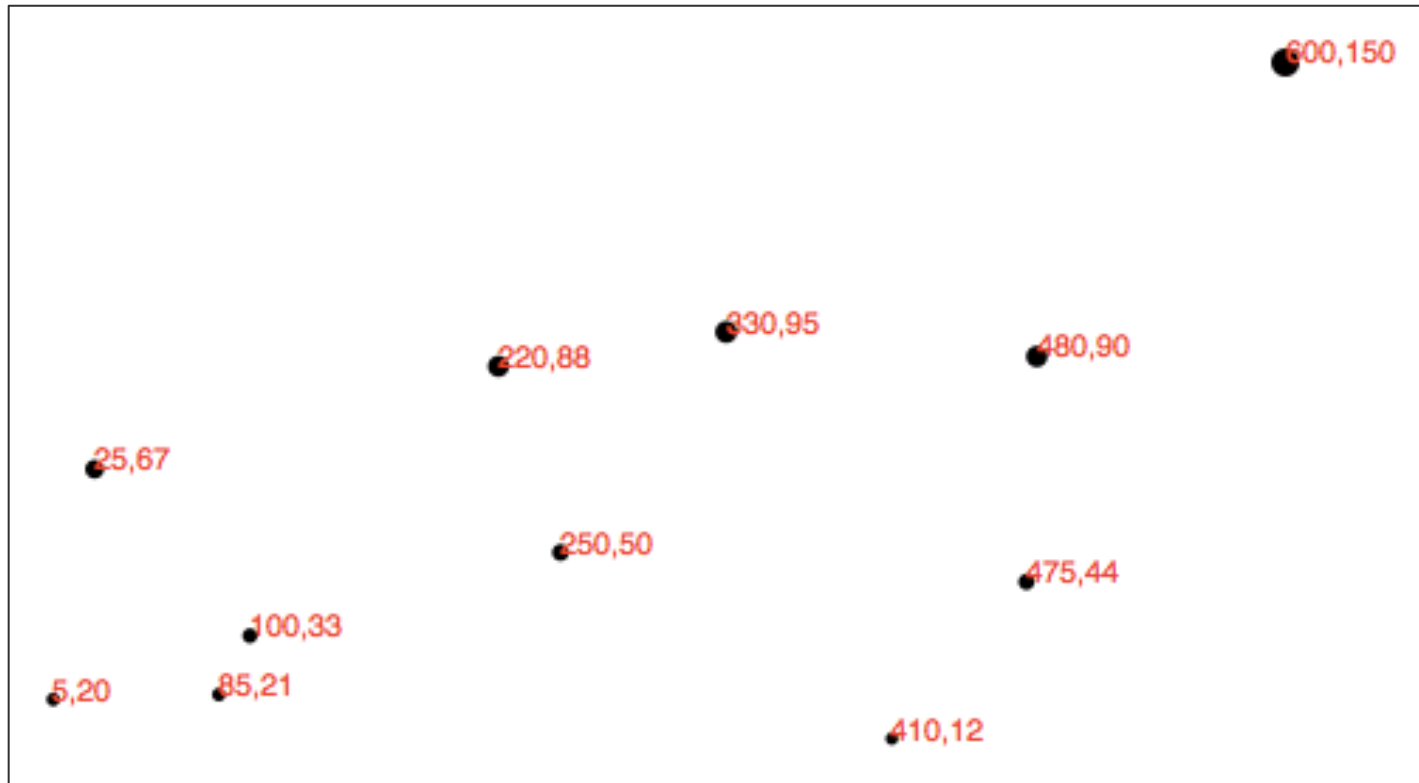
```
.range([h, 0]);
```



Now a smaller input to yScale will produce a larger output value, thereby pushing those circles and text elements down, closer to the base of the image.

Neuer Datenpunkt, grössere SVG-Fläche

- > Neuer Datenpunkt: [600 , 150]
- > Grössere SVG-Fläche: `var h = 300;`



D3.js verbesserter Scatterplot mit dynamischer Skala

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>D3: Adjusted radii</title>
    <script type="text/javascript" src="/js/vendor/d3.min.js"></script>
  </head>
  <body>
    <script type="text/javascript">

      //Width and height
      var w = 500;
      var h = 300;
      var padding = 20;

      var dataset = [
        [5, 20], [480, 90], [250, 50], [100, 33], [330, 95],
        [410, 12], [475, 44], [25, 67], [85, 21], [220, 88], [600, 150]
      ];

      //Create scale functions
      var xScale = d3.scale.linear()
        .domain([0, d3.max(dataset, function(d) { return d[0]; })])
        .range([padding, w - padding * 2]);

      var yScale = d3.scale.linear()
        .domain([0, d3.max(dataset, function(d) { return d[1]; })])
        .range([h - padding, padding]);

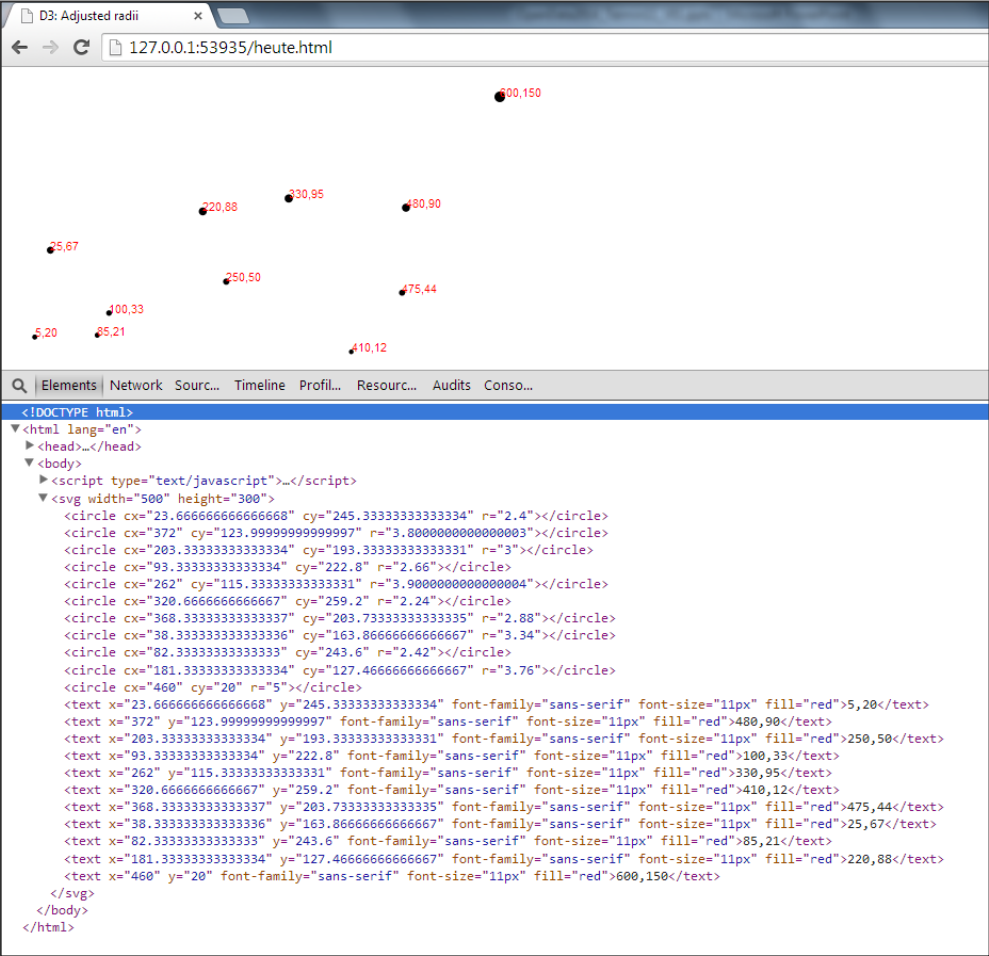
      var rScale = d3.scale.linear()
        .domain([0, d3.max(dataset, function(d) { return d[1]; })])
        .range([2, 5]);

      //Create SVG element
      var svg = d3.select("body")
        .append("svg")
        .attr("width", w)
        .attr("height", h);

      svg.selectAll("circle")
        .data(dataset)
        .enter()
        .append("circle")
        .attr("cx", function(d) {
          return xScale(d[0]);
        })
        .attr("cy", function(d) {
          return yScale(d[1]);
        })
        .attr("r", function(d) {
          return rScale(d[1]);
        });

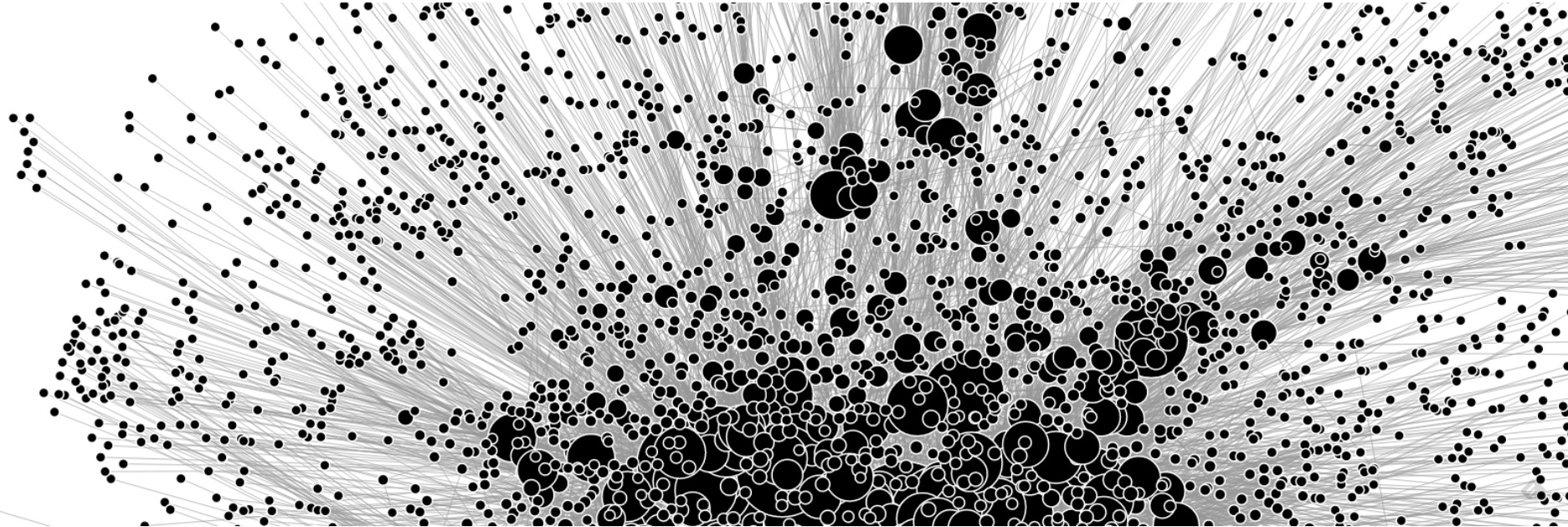
      svg.selectAll("text")
        .data(dataset)
        .enter()
        .append("text")
        .text(function(d) {
          return d[0] + ", " + d[1];
        })
        .attr("x", function(d) {
          return xScale(d[0]);
        })
        .attr("y", function(d) {
          return yScale(d[1]);
        })
        .attr("font-family", "sans-serif")
        .attr("font-size", "11px")
        .attr("fill", "red");

    </script>
  </body>
</html>
```



Agenda

1. Skalen
2. **Achsen**



Setting Up an Axis

Eine Achse erstellen:

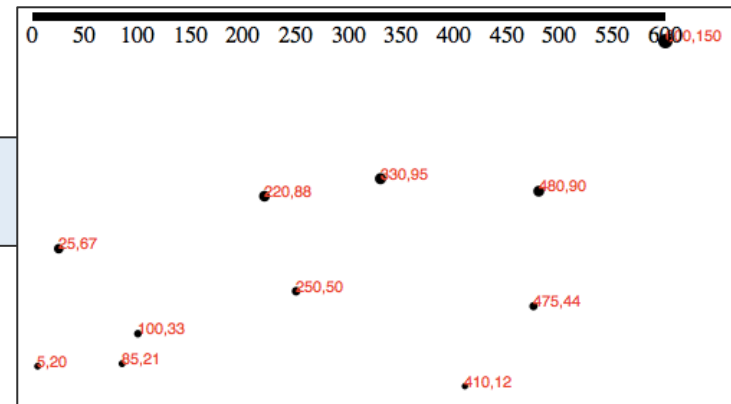
```
var xAxis = d3.svg.axis();
```

Der Achse eine Skala zuweisen:

```
xAxis.scale(xScale);
```

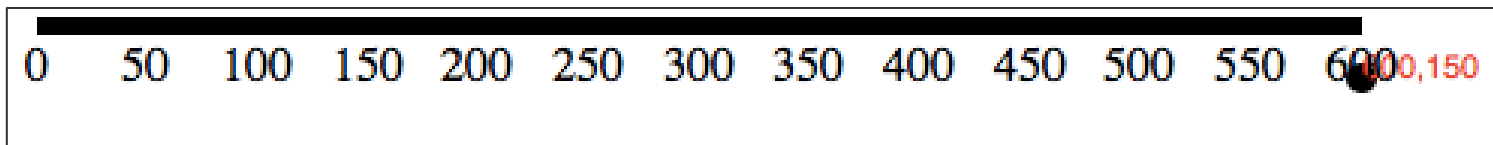
Die Achse zeichnen:

```
svg.append("g").call(xAxis);
```



Setting Up an Axis

> Gezeichnete Achse:



> Generierter SVG-Code:

```
<g>
  <g class="tick" transform="translate(20,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(56.66666666666667,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(93.33333333333334,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(130,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(166.66666666666669,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(203.33333333333334,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(240,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(276.66666666666667,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(313.33333333333337,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(350,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(386.66666666666667,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(423.33333333333337,0)" style="opacity: 1;">...</g>
  <g class="tick" transform="translate(460,0)" style="opacity: 1;">...</g>
  <path class="domain" d="M20,6V0H460V6"></path>
</g>
```

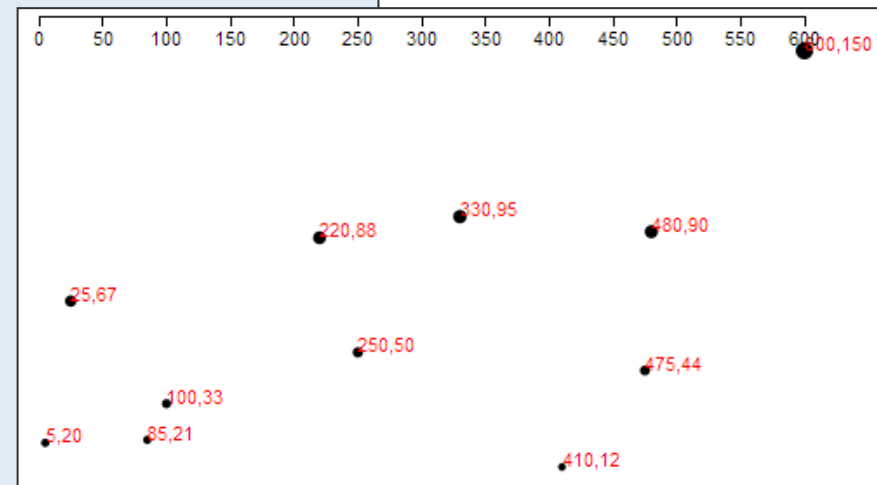
Cleaning It Up

Der Achse die CSS Klasse **axis** zuweisen:

```
svg.append("g")  
  .attr("class", "axis") //Assign "axis" class  
  .call(xAxis);
```

CSS Styles definieren:

```
.axis path,  
.axis line {  
  fill: none;  
  stroke: black;  
  shape-rendering: crispEdges;  
}  
.axis text {  
  font-family: sans-serif;  
  font-size: 11px;  
}
```



SVG transforms

Die Achse soll nach unten verschoben werden. Das wird bei SVG-Elementen mit `transform` und `translate` erreicht:

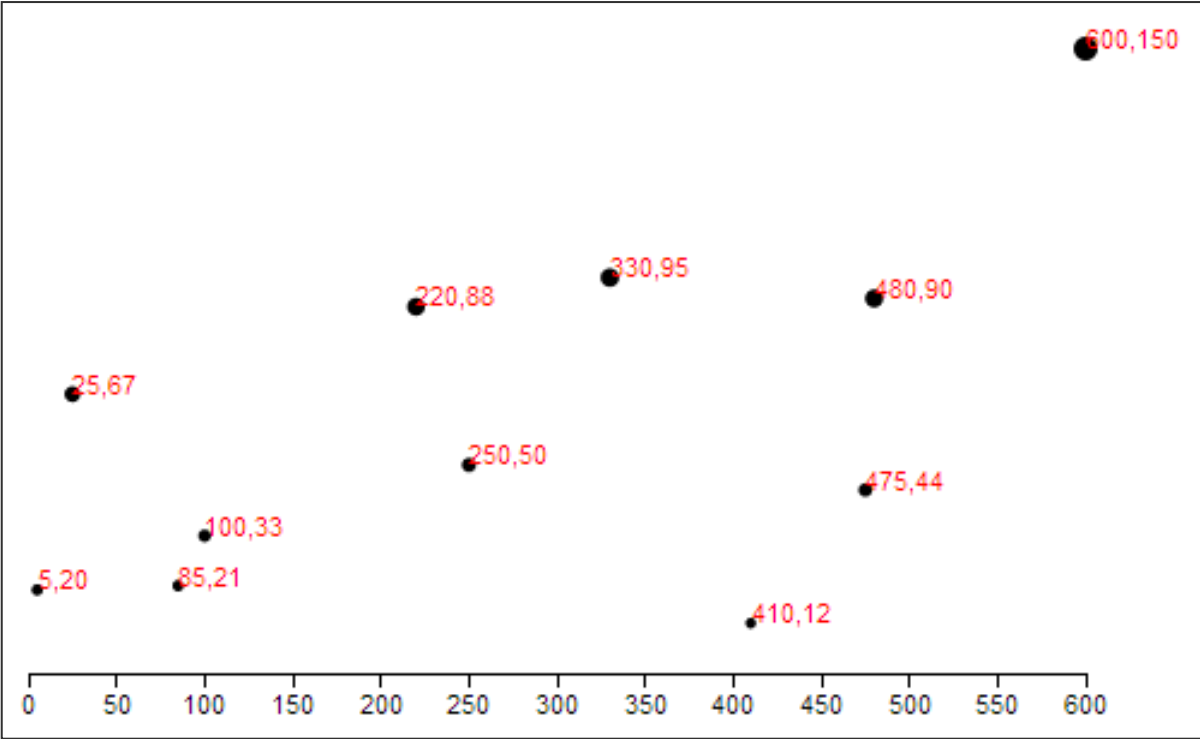
```
<g class="axis" transform="translate(0,280)">
```

Im JavaScript Code muss der Achse hinzugefügt werden:

```
.attr("transform","translate(0," + (h-padding) + ")")
```


SVG transforms

> Resultat:



> Transform:

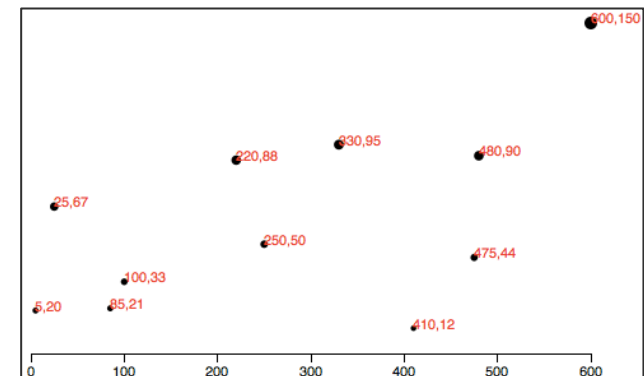
```
<text x=400 y=20 font-family=sans-serif font-size=10>  
  <g class="axis" transform="translate(0,280)">  
    <g class="tick" transform="translate(20,0)" style="stroke-width:2px;">
```

Check for Ticks

You can customize all aspects of your axes, starting with the rough number of ticks, using `ticks()`:

```
var xAxis = d3.svg.axis()  
                .scale(xScale)  
                .orient("bottom")  
                .ticks(5); //Set rough # of ticks
```

D3 interprets the `ticks()` value as merely a suggestion and will override your suggestion with what it determines to be the most clean and human-readable values:



Y Achse

Oben im Code Variable für die Y Achse erstellen:

```
var yAxis = d3.svg.axis()  
                .scale(yScale)  
                .orient("left")  
                .ticks(5);
```

Unten im Code die Y Achse hinzufügen:

```
svg.append("g")  
    .attr("class", "axis")  
    .attr("transform", "translate(" + padding + ",0)")  
    .call(yAxis);
```

D3.js Scatterplot mit Achsen

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>D3: Vertical axis added</title>
  <script type="text/javascript" src="https://d3js.org/d3.min.js"></script>
  <style type="text/css">
    .axis path,
    .axis line {
      fill: none;
      stroke: black;
      shape-rendering: crispEdges;
    }
    .axis text {
      font-family: sans-serif;
      font-size: 12px;
    }
  </style>
</head>
<body>
  <script type="text/javascript">

    //Width and height
    var w = 500;
    var h = 300;
    var padding = 30;

    var dataset = [
      [5, 20], [400, 90], [200, 90], [100, 30], [330, 95],
      [410, 12], [475, 44], [25, 67], [85, 21], [220, 88],
      [600, 150]
    ];

    //Create scale functions
    var xScale = d3.scale.linear()
      .domain([0, d3.max(dataset, function(d) { return d[0]; })])
      .range([padding, w - padding * 2]);

    var yScale = d3.scale.linear()
      .domain([0, d3.max(dataset, function(d) { return d[1]; })])
      .range([h - padding, padding]);

    var xScale = d3.scale.linear()
      .domain([0, d3.max(dataset, function(d) { return d[0]; })])
      .range([2, 5]);

    //Define X axis
    var xAxis = d3.svg.axis()
      .scale(xScale)
      .orient("bottom")
      .ticks(5);

    //Define Y axis
    var yAxis = d3.svg.axis()
      .scale(yScale)
      .orient("left")
      .ticks(5);

    //Create SVG element
    var svg = d3.select("body")
      .append("svg")
      .attr("width", w)
      .attr("height", h);

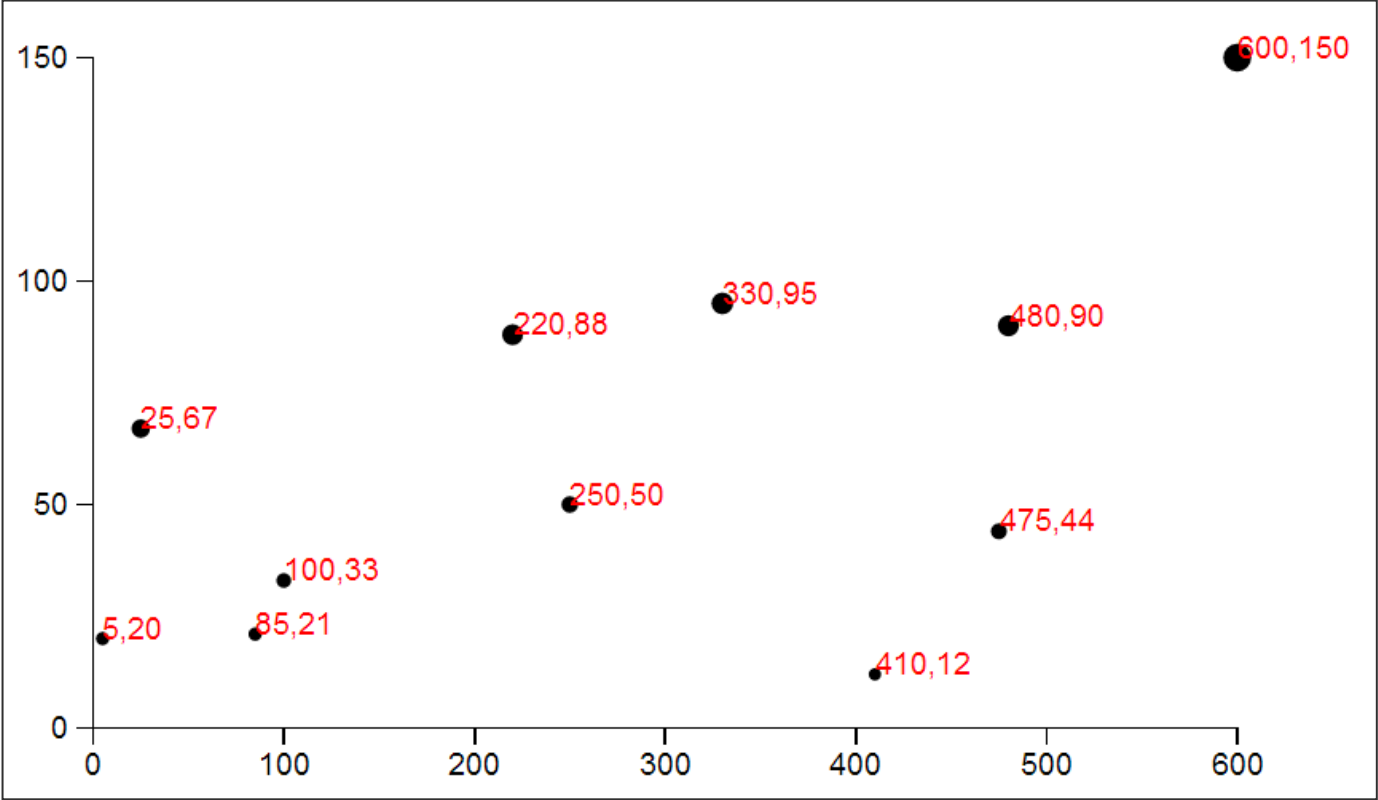
    //Create circles
    svg.selectAll("circle")
      .data(dataset)
      .enter()
      .append("circle")
      .attr("cx", function(d) {
        return xScale(d[0]);
      })
      .attr("cy", function(d) {
        return yScale(d[1]);
      })
      .attr("r", function(d) {
        return yScale(d[1]);
      });

    //Create labels
    svg.selectAll("text")
      .data(dataset)
      .enter()
      .append("text")
      .text(function(d) {
        return d[0] + ", " + d[1];
      })
      .attr("cx", function(d) {
        return xScale(d[0]);
      })
      .attr("cy", function(d) {
        return yScale(d[1]);
      })
      .attr("font-family", "sans-serif")
      .attr("font-size", "12px")
      .attr("fill", "red");

    //Create X axis
    svg.append("g")
      .attr("class", "xaxis")
      .attr("transform", "translate(0, " + (h - padding) + ")")
      .call(xAxis);

    //Create Y axis
    svg.append("g")
      .attr("class", "yaxis")
      .attr("transform", "translate(" + padding + ", 0)")
      .call(yAxis);

  </script>
</html>
```



http://chimera.labs.oreilly.com/books/1230000000345/ch08.html#_y_not