Line charts

SVG <path> elements

```
<svg width = "500" height = "400">
    <path d="M 50 400 L 100 300 L 150 300
             L 200 330 L 250 175 L 300 275
             L 350 250 L 400 125" fill="none"
             stroke="red" stroke-width="5">
    </path>
</svg>
```

d= attribute: **M = move to**
L = line to

```
<svg width = "500" height = "400">
    <path d="M 50 400 L 100 300 L 150 300
             L 200 330 L 250 175 L 300 275
             L 350 250 L 400 125" fill="none"
             stroke="red" stroke-width="5">
    </path>
</svg>
```

SVG editors

https://github.com/SVG-Edit

README.md

SVG-edit

SVG-edit is a fast, web-based, javascript-driven SVG drawing editor that works in any modern browser.

Try SVG-edit here

(Also available as a download in releases).

Recent news

- 2017-07 Added to Packagist: https://packagist.org/packages/svg-edit/svgedit

R

Save plots as SVG files:

```
> svg("xsquared.svg")
> plot(1:10, (1:10)^2, axes=F)
> dev.off()

> library(svglite)
> svglite("xsquared2.svg")
> plot(1:10, (1:10)^2, axes=F)
> dev.off()
```

R

Set graphics device to svg in code chunk options:

```{r, dev="svg"}
```

```{r, dev="svglite"}
```

example

Back to line charts

What we have:

Day	High Temp
April 1	60
April 2	43
April 3	43
April 4	56
April 5	45
April 6	62
April 7	49

Back to line charts

What we have:

Day	High Temp
April 1	60
April 2	43
April 3	43
April 4	56
April 5	45
April 6	62
April 7	49

What we need:

<path class="line" fill="none" d="M0,116.66666666666669C27.777777777777786,155.51317663817 66,55.555555555555557,194.35968660968655,83.33333333333333,215.83333333333331C111.11111111111109,237.30698005698008,138.88888888888889,241.40776353276354,166.66666666666666,215.83333333333331C194.44444444444443,190.2589031339031,222.22222222222223,135.00925925925924,250,140C277.77777777777777,144.99074074074076,305.55555555555554,210.22186609686608,333.33333333333333,204.16666666666666C361.11111111111111,198.11146723646723,388.88888888888889,120.76994301994304,416.66666666666667,105.00000000000003C444.44444444444446,89.23005698005701,472.22222222222223,135.03169515669518,500,180.83333333333334**"></path>**

Step 1. Create a line generator

Expects data in an array of 2-dimensional arrays, that is, an array of (x,y) pairs:

```
> var dataset = [ [0, 60], [1, 43], [2, 43],
                  [3, 56], [4, 45], [5, 62],
                  [6, 49] ];

> var mylinegen = d3.line()
```

Test it in the Console:

```
> mylinegen(dataset);
```

Add a path element

Just to see how it works:

```
> var mypath = mylinegen(dataset);
```

```
> d3.select("svg").append("path")
    .attr("d", mypath)
    .attr("fill", "none")
    .attr("stroke", "red")
    .attr("stroke-width", "5");
```

Add scales

An ordinal scale for x:

```
> var xScale = d3.scaleBand()
      .domain(d3.range(dataset.length))
      .range([0, 500])
```

A linear scale for y:

```
> var yScale = d3.scaleLinear()
      .domain([d3.min(dataset, d => d[1]) - 20,
               d3.max(dataset, d => d[1]) + 20)
      .range([400, 0]);
```

Add accessor functions .x( ) .y( )

```
> mylinegen
      .x(d => xScale(d[0]))
      .y(d => yScale(d[1]));
```

Test again:

```
> mylinegen(dataset);
```

Add a path element

Try the new and improved line generator:

```
> var mypath = mylinegen(dataset);
```

```
> d3.select("svg").append("path")
    .attr("d", mypath)
    .attr("fill", "none")
    .attr("stroke", "red")
    .attr("stroke-width", "5");
```

Step 2. Call the line generator

```
> d3.select("svg").datum(dataset)
      .attr("d", mylinegen)
      .attr("fill", "none")
      .attr("stroke", "teal")
      .attr("stroke-width", "5");
```

Create a line style

```
.line {
        fill: none;
        stroke: teal;
        stroke-width: 5px;
      }

d3.select("svg").datum(dataset)
   .attr("d", mylinegen)
   .attr("class", "line");

EDAV8_1_linegen.html
```