Line charts

SVG <path> elements

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



SVG editors

https://github.com/SVG-Edit

■ README.md



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"

7786,155.5131766381766,55.555555555555 ,194.35968660968655,83.333333333333333,21 5.8333333333331C111.11111111111109,237. 30698005698008,138.88888888888889,241.40 776353276354,166.666666666666666,215.8333 333333331C194.4444444444443,190.25890 31339031,222.222222222223,135.00925925 925924,250,140C277.77777777777777,144.99 074074074076,305.555555555555554,210.2218 6609686608,333.333333333333333,204.1666666 666666C361.11111111111111,198.111467236 46723,388.8888888888889,120.769943019943 04,416.6666666666667,105.00000000000003C 444.444444444446,89.23005698005701,472 .222222222223,135.03169515669518,500,1 80.8333333333334"></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 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:

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
```