DEVL.js :: "Damn Easy Visualization Library"

Kevin McVey, 12 Dec 2013, ARCH5590

With the advent of big data in a world growing ever more saturated with information, some means of organizing and making sense of it all is becoming necessary. Following in the footsteps of information communication pioneers William Playfair and Charles Minard, statisticians and scientists have looked towards the practical arts to construct a new discipline for making sense of our increased access to information. Today this field is referred to as Data Visualization. While it is indeed exciting to see large organizations such as IBM, Amazon.com and The Washington Post invest in Data Visualization experts; the tools used by these experts remain esoteric. A quick perusal of Wikipedia's list of Visualization tools provides further evidence of this claim:

- 29 of the 49 tools listed are marked as "proprietary."
- Of those 20 tools marked as "open source" all but one of them claim to be intended primarily for engineers, scientists, and programmers.
- Only paid, proprietary tools offer solutions for other, perhaps non-technical users including students, financial users, business users, managers, telecommunications personnel, government workers, coaches, and teachers.

It is becoming clear that there is a demand for Visualization as our world of data grows in complexity, but this demand is proving difficult to satisfy. Even the most popular visualization tools are often criticized for featuring large usability flaws or for only being accessible to skilled programmers. Data Driven Documents (D3) for instance is simultaneously hailed by the Visualization community for being one of the finest tools available and denounced for having one of the steepest learning curves of any JavaScript library out there. Some of my own personal visualization projects, written using JavaScript and an SVG library called Raphael fall victim to this as well. *Team Diversity* (found here: http://kevin.4mcveys.com/mhiviz/) for instance requires approximately 500 lines of JavaScript to properly visualize an interactive pie chart with a Modified Herfindahl Index slider and verbose "participant voice" chart. Similarly, *Major Regions of Virtual Water Trade* (found here: http://people.virginia.edu/~kmm4ce/water/) requires even more, reaching about 800 lines of JavaScript. This level of programming involvement might not frighten a seasoned computing professional with a degree in CS or CpE, but to many the language of Visualization is as foreign as ever.

Herein lays the rationale for this semester's software project, DEVL.js. DEVL, short for "Damn Easy Visualization Library," is a JavaScript library that seeks to demystify the development of common visualizations. DEVL.js serves two primary goals:

1. To simplify the development of visualizations to the point that technical / programming experience is no longer necessary.
2. To remain flexible enough to meet a variety of needs and avoid becoming a specialty tool.

These two goals certainly seem to conflict to some extent; flexible and customizable tools are rarely simple. Multiple iterations of DEVL.js' and its many features carefully walked this line in order to preserve its simplicity of use while offering a wide variety of features. It was discovered early on, in the first version of DEVL.js, that even the simplest of graphs have some level of inherent complexity in their design and formatting. Understanding this, DEVL.js' formatting was designed from the ground-up with five primary areas of customization: window, x-axis, y-axis, data, and legend, each of which containing several sub-items and properties that are each independently customizable.

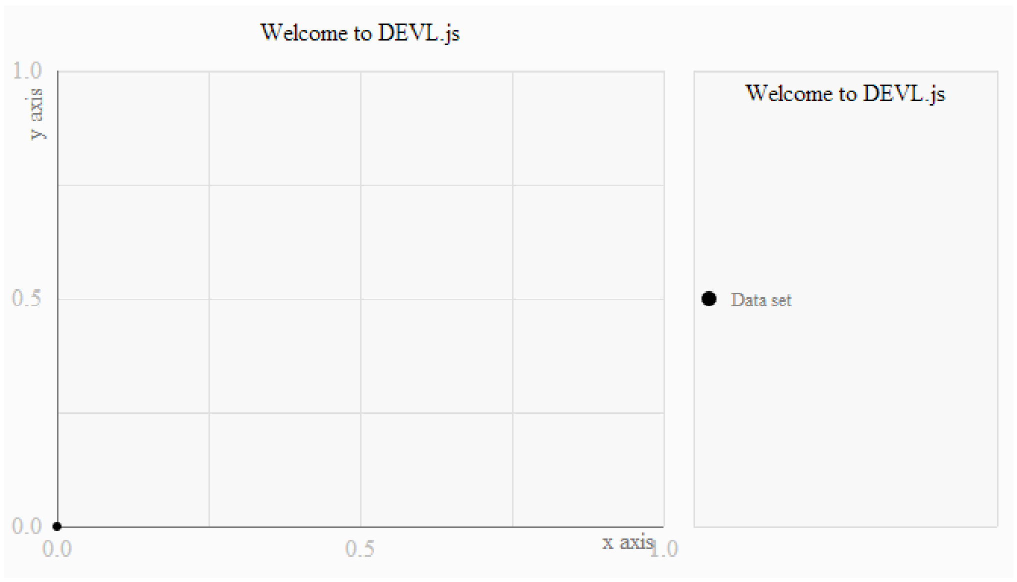
window: style
window: xposition
window: yposition
window: width
window: height
window: title: enabled
window: title: style
window: title: title
window: title: bufferx
window: title: buffery
window: title: align
window: borders: top: enabled
window: borders: top: style
window: borders: right: enabled
window: borders: right: style
window: borders: bottom: enabled
window: borders: bottom: style
window: borders: left: enabled
window: borders: left: style
xaxis: enabled
xaxis: style
xaxis: min
xaxis: max
xaxis: show
xaxis: position
xaxis: modulatebyindex
xaxis: label: enabled
xaxis: label: style
xaxis: label: rotate
xaxis: label: buffer
xaxis: label: label
xaxis: ticks: enabled
xaxis: ticks: style
xaxis: ticks: show
xaxis: ticks: position
xaxis: ticks: ticksperlabel
xaxis: ticks: tickcount
xaxis: ticks: tickdelta
xaxis: ticks: labels: enabled
xaxis: ticks: labels: style
xaxis: ticks: labels: rotate
xaxis: ticks: labels: bufferx
xaxis: ticks: labels: buffery
yaxis: enabled
yaxis: style
yaxis: min
yaxis: max
yaxis: show
yaxis: position
yaxis: modulatebyindex
yaxis: label: enabled
yaxis: label: style
yaxis: label: rotate
yaxis: label: buffer
yaxis: label: label
yaxis: ticks: enabled
yaxis: ticks: style
yaxis: ticks: show
yaxis: ticks: position
yaxis: ticks: ticksperlabel
yaxis: ticks: tickcount
yaxis: ticks: tickdelta
yaxis: ticks: labels: enabled
yaxis: ticks: labels: style
yaxis: ticks: labels: rotate
yaxis: ticks: labels: bufferx
yaxis: ticks: labels: buffery
data: radial
data: modulatesizebyindex
data: modulatecolorbyindex
data: modulatesizemin
data: modulatesizemax
data: modulatecolormin
data: modulatecolormax
data: values
legend: enabled
legend: style
legend: height
legend: width
legend: bufferx
legend: buffery
legend: xposition
legend: yposition
legend: borders: top: enabled
legend: borders: top: style
legend: borders: right: enabled
legend: borders: right: style
legend: borders: bottom: enabled
legend: borders: bottom: style
legend: borders: left: enabled
legend: borders: left: style
legend: entries: style
legend: entries: windowpaddingx
legend: entries: windowpaddingy
legend: entries: textpaddingx
legend: entries: textpaddingy
legend: entries: markersize
legend: entries: placement
legend: title: enabled
legend: title: style
legend: title: title
legend: title: bufferx
legend: title: buffery
legend: title: align

As shown here, the customizable fields within a given "DEVL Object" are many. These fields represent only the "top level" fields, whereas a further series remains for the customization of each DEVL data set; more on those later. Fortunately, DEVL does not require that all of these fields be present, only those that the user wishes to change from the DEVL's default plot. DEVL's data formatter handles filling in all values the user chooses not to provide.

As one of DEVL's core goals is use simplicity, DEVL strives to require as little code as possible to function. The DEVL constructor requires only the ID of any DIV in an HTML page and the DEVL plotter requires only the DEVL formatting JSON object. The following is a perfectly valid use of DEVL:

```
var DEVL = DEVL("svgContainer");
DEVL.draw_scatter_plot({ });
```

The plot that this draws is not a particularly interesting one. The JSON object passed to DEVL.draw_scatter_plot is completely empty and DEVL, therefore, fills all **104** editable fields with default values providing the following image.



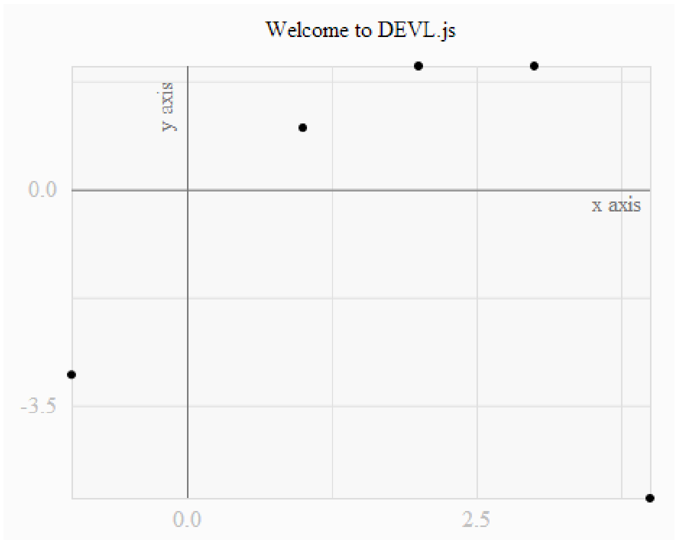
Similarly, DEVL tries to automate as much of the drawing process as it can behind the scenes. The plot automatically scales and translates in order to match a user's dataset. Consider the following:

```
var DEVL = DEVL("svgContainer");
DEVL.draw_scatter_plot({
    "data":{
        "values":{
            "0":{
                "values":[[1,1],[2,2],[-1,-3],[4,-5],[3,2]]
            }
        }
    },
    "legend":{
        "enabled":false
    }
});
```

The JSON object in this example is stylized for readability but would work just as well in one line:

```
({"data":{"values":{"0":{"values":[[1,1],[2,2],[-1,-3],[4,-5],[3,2]]}}},"legend":{"enabled":false}}
```

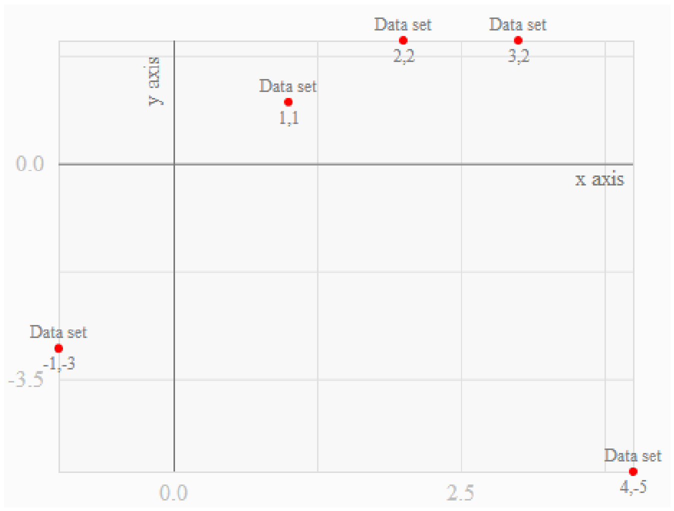
In this example, the legend has been removed and a data set of [[1,1],[2,2],[-1,-3],[4,-5],[3,2]] has been provided. In both cases (formatted and unformatted JSON) the chart drawn exhibits some desireable properties. The chart's minimum and maximum values on the x- and y- axes are scaled to match the user's input data and the origin is shifted to represent where (0,0) should lie. While these properties are independently editable should a power user wish to edit them, they default to "auto," a feature that lets DEVL do the work. This functionality was carefully designed in order to ensure DEVL's simplicity and utility – the average user should not be required to deal with the intricacies of their plot if they do not wish to.



Data in DEVL is easily entered and managed. DEVL handles data in units called "sets" in order to maximize organization. Each set has its own collection of independently modifiable fields and subfields, similar to the overall DEVL formatter as previously shown. Sets contain a few core areas of modification as well, these being interpolation, label, rawvalues, and the values to plot themselves. As before, these modifiable fields are all automatically filled by DEVL's data formatter if left unfilled by the user. These particular fields are hidden inside each set as they give the user to place multiple data sets in one plot with different colors, labels, and effects. As before, this demonstrates DEVL's flexibility as well as its dedication to organization and utility.

enabled
style
interpolation: enabled
interpolation: style
interpolation: modulatestroke
label: enabled
label: name
label: style
label: align
label: inlegend
label: buffer
rawvalues: enabled
rawvalues: style
rawvalues: align
rawvalues: buffer
values

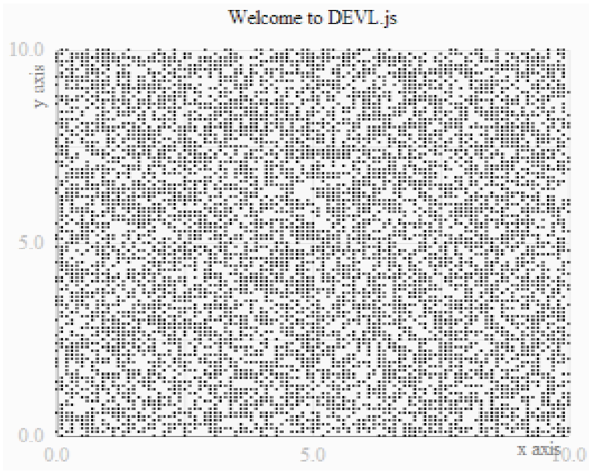
Modifying the previous DEVL JSON object can help show off some of these set-based features. Here, DEVL adds the exact data values themselves along with the set title to each plotted point, using user-provided alignment information. The JSON Object that draws this plot is as follows:

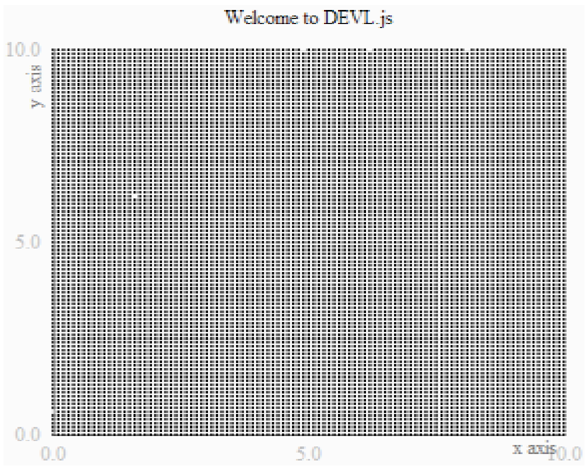


```
{"window":{"title":{"enabled":false}},"legend":{"enabled":false},"data":{"values":{"0":{"style":{"cx":370,"cy":50,"r":3,"fill":"#f00"},"label":{"enabled":true,"buffer":4,"align":"top"},"rawvalues":{"enabled":true,"align":"bottom","buffer":12},"values":[[1,1],[2,2],[-1,-3],[4,-5],[3,2]]}}}}
```

Even more exciting about DEVL is its ability to gracefully handle large amounts of data. Provided below are a few stress tests drawing large quantities of information at once. X and Y values are randomly generated from 0 to 10 and forced into only having one decimal place of accuracy (rather, each number has two significant digits). The code that generates these numbers is shown below with "n" as the variable deciding the size of the test.

```
var DEVL = DEVL("svgContainer");
var n = 10000;
var set = [];
for(var i = 0; i < n; i++) {
	set.push([parseFloat((Math.random() * 10).toFixed(1)), parseFloat((Math.random() *
10).toFixed(1))]);
}
var plot = {"xaxis": {"min": 0, "max": 10}, "yaxis": {"min": 0, "max": 10}, "data":
{"modulatesizemin":1,"modulatesizemax":1,"modulatesizebyindex":"0","values": {"0": {"values": {}}}}};
plot["data"]["values"]["0"]["values"] = set;
DEVL.draw_scatter_plot(plot);
```



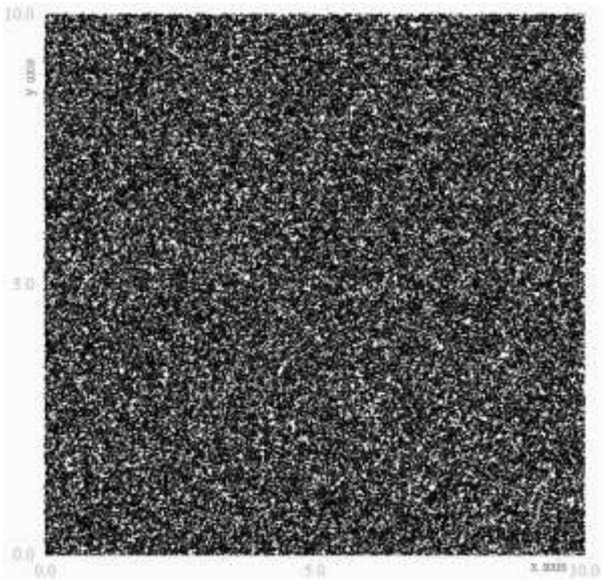


N=10000

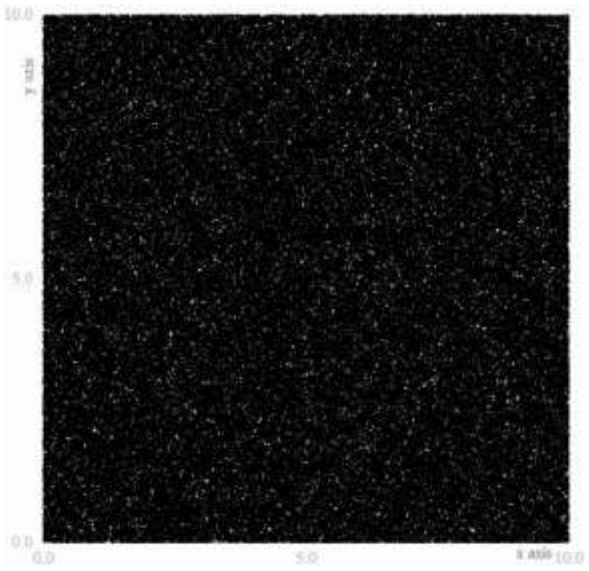
N=100000

Changing the plot and code a bit to no longer constrain the numbers to some amount of significant digits makes the information even more interesting as we continue to use DEVL to visualize randomness.

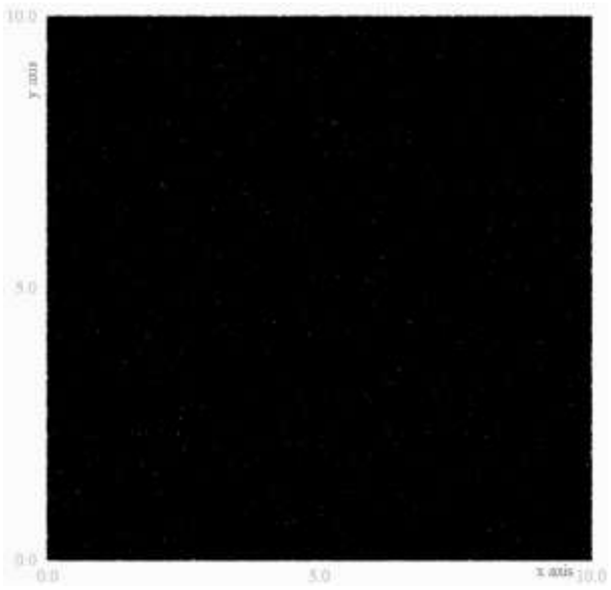
```
var DEVL = DEVL("svgContainer");
var n = 100000;
var set = [];
for(var i = 0; i < n; i++) {
	set.push([(Math.random() * 10), (Math.random() * 10)]);
}
var plot = {"xaxis": {"min": 0, "max": 10, "ticks":{"show":0,"position":"bottom","ticksperlabel":2}},
"yaxis": {"min": 0, "max": 10, "ticks":{"show":0,"position":"left","ticksperlabel":2}}, "data":
{"modulatesizemin":1,"modulatesizemax":1,"modulatesizebyindex":"0","values": {"0": {"values": {}}}}};
plot["data"]["values"]["0"]["values"] = set;
DEVL.draw_scatter_plot(plot);
```



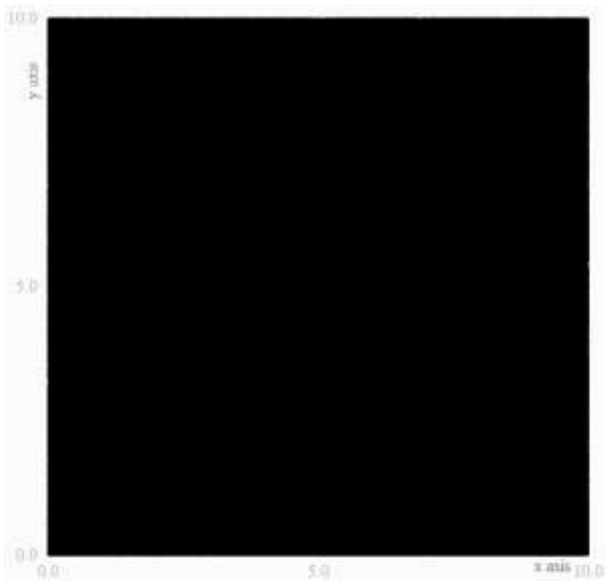
N = 100,000



N = 250,000

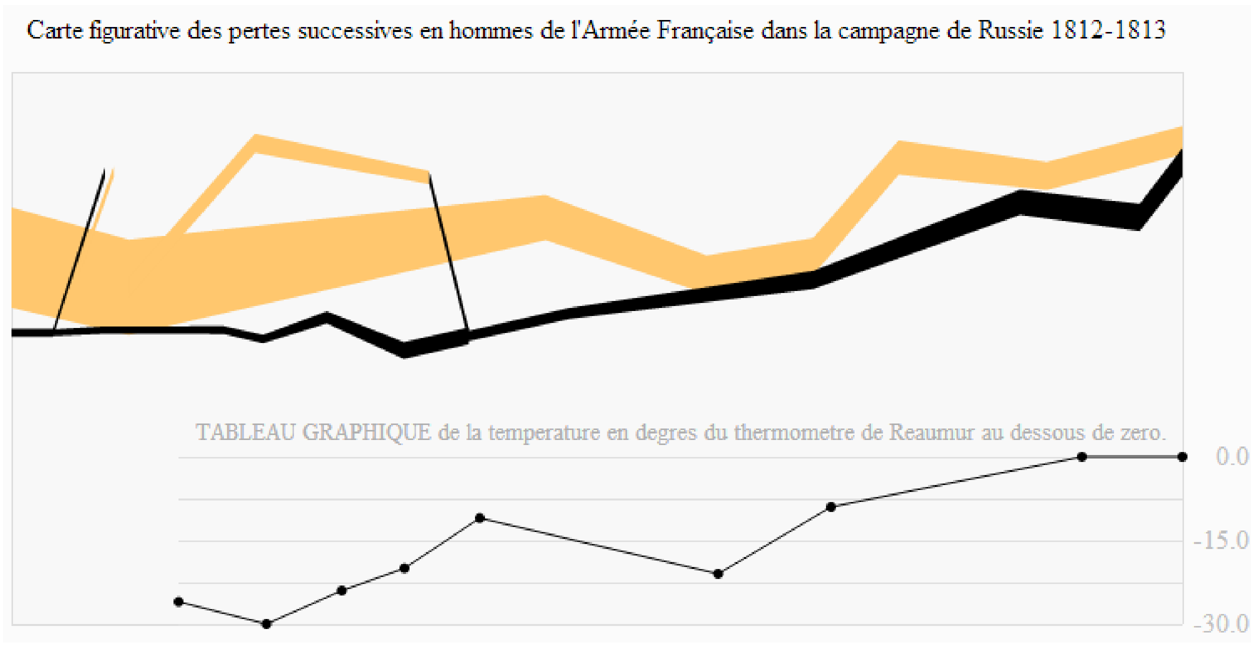


N = 500,000



N = 1,000,000

As one final test of DEVL's power, DEVL has been used to recreate the famous "Minard Graph," a visualization of Napoleon's march into Russia and back. This document concludes with it as a statement of DEVL's utility as well as its success in achieving its goals of simplified versatility. DEVL can be experimented with on the "DEVL Dashboard" at http://kevin.4mcveys.com/devl/devl_demo.html until a nicer DEVL.js page gets created. DEVL itself can be downloaded at http://kevin.4mcveys.com/devl/devl.js



(Source code available at http://kevin.4mcveys.com/devl/minard.html)