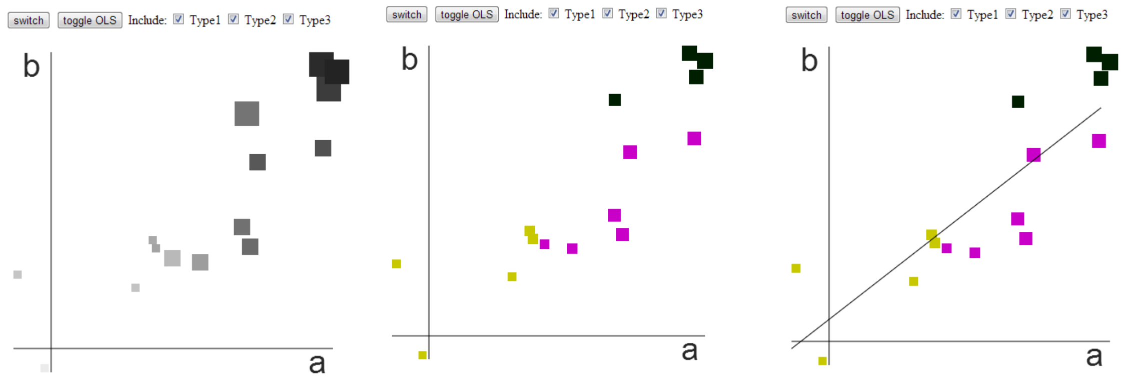
SI649 – Lab 1

September 10, 2014

Today we're going to practice building a mini-visualization using Javascript. It'll be limited in lots of ways and would never fly as an actual visualization, but it'll be fun to play with. **I do not expect you to finish this entire lab in class.** Work through as much as you can and then take it home to finish as practice. This will not be graded.

When you're done, you'll get something that looks like this:



(Different views depending on the buttons I clicked).

What we have is a scatter plot of some data. Each data point has 4 things associated with it:

a, b, xtype, and *c*

you should think of *a* and *b* as the *x* and *y* values respectively, and *xtype* as *c* as some additional pieces of data that we care about. *xtype* is categorical, and can be: type1, type2, and type3, *c* is numerical

Here's a snippet of the data definition:

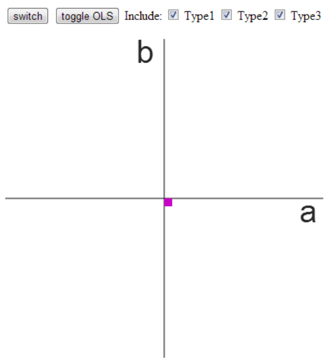
```
var alldata = [{a:-0.035333333,b:-0.053333333,xtype:"type1",c:-0.08},
      {a:-0.127333333,b:0.277333333,xtype:"type1",c:2.81}, … ]
```

It shows you two sample points.

Step 1:

To start, grab the scatter.html file from ctools. When you open it up, you should see something like:

(all points are drawn in the same place with the same color, axes are in the wrong place and buttons don't do anything useful)

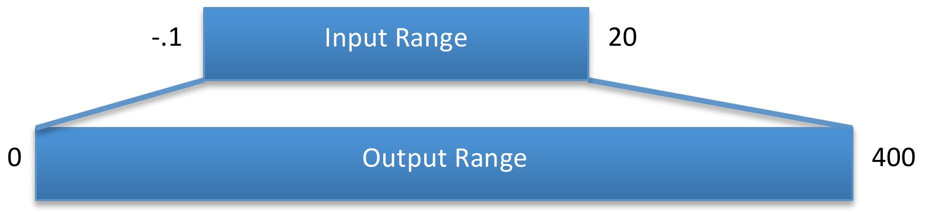


Take a look inside the file, there are a bunch of places labeled as "CHANGE ME." Those will be the places we try to fix things.   A few things to notice: there are some "global variables" defined… things like alldata, filtereddata, xmin, context, etc.   These are things that many of our methods will use, so we're just going to leave them "scoped" at this level (that means any function can access them without having to pass around a pointer). This isn't always safe to do, but it'll be ok for now.

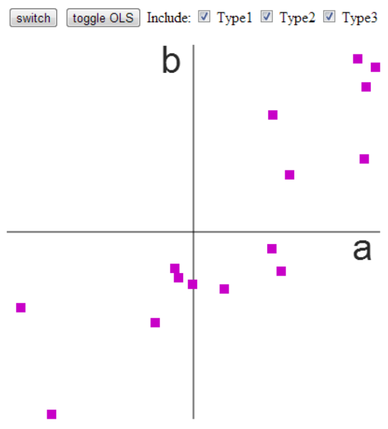
Step 2:

First things, first, the points are totally in the wrong place. Find the function called drawScatter and calcMinMax. Fix those two functions. calcMinMax should find the minimum value a, b, and c can take and store those in xmin, xmax, ymin, ymax, etc. (again, recall that “a” is our x, and “b is our y). These should not be normalized yet… so if the absolute minimum of c is .15, cmin should be .15. If you get really stuck on calcMinMax, just leave it hard coded the way it is now

When you’re done with that, go modify the drawScatter function so that xloc and yloc are set correctly. Here we need to convert the values of the points into pixel coordinates. I assume a canvas size of 400 x 400 (so 400 pixels in every direction). You’re going to want to calculate a function that maps the point into this space. Think of it this way… if the *a* values range from: -.1 to 20, we want to map -.1 to 0 and 20 to 400 (see figure). If you had a point that was .9 in the original scale it would get mapped to pixel coordinate “20” (approximately). Before you try programming this, write out the equation:

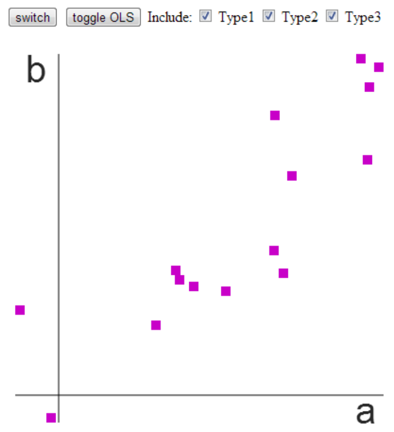


If you do this correctly, you should see something like this:



Step 3:

Fix the axes… go to the drawAxes function and modify the x1,y1,x2,y2 points (these are the end points of the two axes). You can use essentially the same function you used before. You hopefully get something like this:

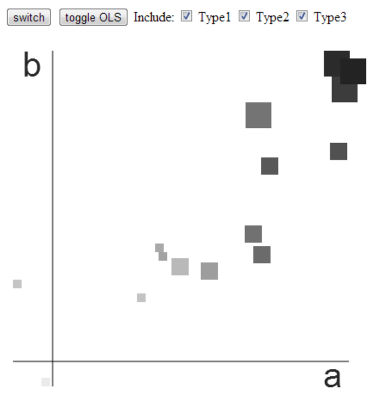


Step 4:

Ok, right now everything is hard coded so that all points are the same color size. The way this works is that drawScatter sends each point to a function that calculates the color and size and then uses the values returned by those (see the functions dummycolor and dummysize). Modify these so that :

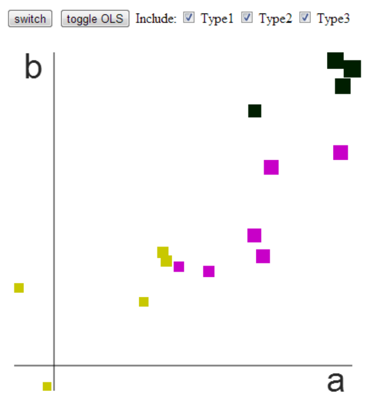
- each points are sized by *xtype* (let's say 10, 20, and 30 pixels respectively for type1, type2, and type3)
- and each points is colored by the c value. This is why calculating cmin/cmax will come in handy. It's more or less the same function as finding the xlocation and ylocation
- You should create two functions to do this similar to dummycolor and dummysize, and then modify the lines that say colorfunc = … and sizefunc = … to point at your new functions

You'll hopefully have something like this:



Step 5:

Now try to do the opposite, write a function that colors by xtype (pick your favorite 3 colors) and sizes by c (let's say between 10 pixels for the smallest and 20 for the biggest). You should see something like this:



Step 6:

There's a button at the top of the page that says "switch." Every time you press it, it calls a function (togglevarbs) that toggles an internal variable called mode. If the variable was set to true, it changes it to false. If it was false it goes to true.

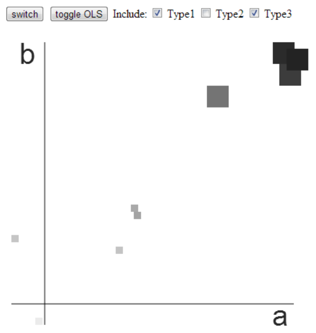
Modify the code so that every time you click the button (you can do this either by modifying drawScatter or togglevarbs (which is the function that gets called when you click). Have it switch configurations depending on the state of mode (true = size by xtype, color by c ; false = size by c, color by xtype).

Step 7:

There's a function that is used to filter the data so that only certain types are displayed. Every time you uncheck (or check) the checkboxes at the top, the drawIt function gets called (not the best name, sorry). drawIt makes use of a "utility" function called filterByCat. drawIt passes a list of "checked" items to filterByCat along with all the original points. What it expects back in a subset of those points that are "valid" (i.e. their category has been checked). Modify filterByCat so that it does this.

The two inputs to filterByCat are "points" which is basically our data in the original format, and valid, which is an array of strings of checked boxes (e.g., if Type1 and Type 3 are checked, valid will be ["type1","type3"].

If you do this correctly and hit untoggle type 2, I should see:

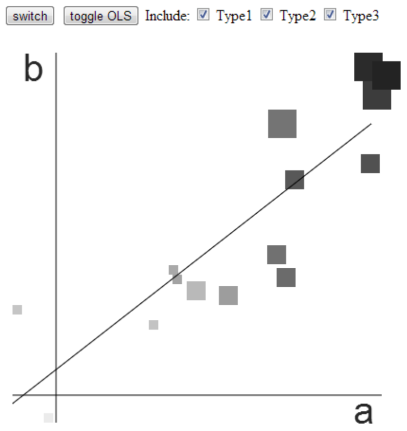


Step 8 (bonus):

When you press toggle OLS, a line is drawn from the bottom left to top right. This is supposed to be the linear fit. Modify the drawOLS code so that it calculates the correct fit depending on which points are displayed (the code is built right now to pass you the filtereddata you calculated in step 7 as "data").

Take a look at the Wikipedia page for OLS if you don't remember how to calculate it: http://en.wikipedia.org/wiki/Ordinary_least_squares.

If you did this correctly, you'll hopefully see something like the following when you click on "toggle OLS"


switch
toggle OLS
Include: ☑ Type1 ☑ Type2 ☑ Type3
b
a