

d3.js scatterplot with different colors and symbols - issues encountered

I am trying to create a scatterplot of hundreds of datapoints, each with about 5 different attributes. The data is loaded from a .csv as an array of objects, each of which looks like this:

```
{hour: "02",yval: "63",foo: "33", goo:"0", bar:"1"},
```

I want to display the scatterplot with the following attributes:

Shape for bar :

-circle to represent all points where bar=0, and a triangle-down to represent those where bar=1 (this is a dummy variable).

Color for foo and goo:

- All points start as grey. goo is categorical with values [0,1,2] while foo is quantitative with a range from 0-50. foo and goo are mutually exclusive, so only one of them has a value. In other words, for each data point either foo=0 or goo=0.
- Points with goo=1 should be orange; points with goo=2 should be red.
- foo should be mapped onto a linear color scale from light blue to dark blue, ie d3.scale.linear().domain([0, 50]).range(["#87CEFF", "#0000FF"]);

I can do each of these individually, but defining everything together is creating issues for me.

My code with reproducible data is here: http://jsfiddle.net/qy5ohw0x/3/

Issues

· For the symbol, i tried

```
.append("svg:path")
.attr("d", d3.svg.symbol())
```

which did not work. I tried a different approach altogether, but this did not map the values correctly:

```
var series = svg.selectAll("g.series")
    .data(dataSet, function(d, i) { return d.bar; })
    .enter()
    .append("svg:g")

series.selectAll("g.point")
    .data(dataSet)
    .enter()
    .append("svg:path")
    .attr("transform", function(d, i) { return "translate(" + d.hour + "," + d.yval + ")"; })
    .attr("d", function(d,i, j) { return d3.svg.symbol().type(symbolType[j])(); })
    .attr("r", 2);
```

• For the goo colors (grey/orange/red), i mapped the values to the 3 colors manually:

```
First define var colors = ["grey", "orange", "red"];
```

Then while drawing the data points chain

```
.style("fill", function (d) { return colors[d.type]; })
```

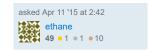
This worked alone, but not with the different symbols.

• Finally, can i chain a second color .attr for foo ? d3.scale.linear().domain([0, 50]).range(["#87CEFF", "#0000FF"]); would probably work if this is possible.

Again, the jsfiddle is here: http://jsfiddle.net/qy5ohw0x/3/

Thanks!!





2 Answers

Just do all the logic and comparisons in a function(d) for each attribute.

First set up some helpers:

```
// symbol generators
var symbolTypes = {
  "triangleDown": d3.svg.symbol().type("triangle-down"),
```

```
"circle": d3.svg.symbol().type("circle")
};

// colors for foo
var fooColors = d3.scale
    .linear()
    .domain([0, 50])
    .range(["#87CEFF", "#0000FF"]);
```

Then append a path for each symbol:

```
svg.selectAll("path")
     .data(dataSet)
     .enter().append("path")
     .attr("class", "dot")
     // position it, can't use x/y on path, so translate it
     .attr("transform", function(d) {
    return "translate(" + (x(d.hour) + (Math.random() * 12 - 6)) + "," + y(d.yval) +
     // assign d from our symbols
     .attr("d", function(d,i){
    if (d.bar === "0") // circle if bar === 0
        return symbolTypes.circle();
          else
               return symbolTypes.triangleDown();
     // fill based on goo and foo
     .style("fill", function(d,i){
    if (d.goo !== "0"){
        if (d.goo === "1")
                    return "red":
               else
                    return "orange";
          }else{
               return fooColors(d.foo);
    });
```

Updated fiddle.

On a side note, I actually think straight d3 is way more intuitive than nvd3 for this situation.



Ah, thanks! I guess I had the right idea for appending a path element and then using d3.svg.symbol(), but my syntax was wrong. – ethane Apr 14 '15 at 22:23

It's much simplier with nvd3.js

```
function prepareData (data) {
     return [{
          key:
                      'Group 1'
          values: data.map(function (item) {
   item.shape = item.bar == "0" ? 'circle' : 'triangle-down';
   item.x = Number(item.hour);
   item.y = Number(item.yval);
               item.size = 0.1;
item.disabled = Math.random() > 0.4;
               return item;
         })
    }]
nv.addGraph(function() {
  var chart = nv.models.scatterChart()
                    .showDistX(false)
                    .showDistY(true)
                    .showLegend(false)
  //Axis settings
chart.xAxis.tickFormat(d3.format('3.0f'));
  chart.yAxis.tickFormat(d3.format('3.0f'));
  d3.select('#chart svg')
        .datum(prepareData(dataSet))
        .call(chart)
  // A bit hacky but works
  var fooscale = d3.scale.linear().domain([0, 50]).range(["#87CEFF", "#0000FF"]);
  function colorer(d) {
       return 'orange';
else if (d.goo == '2')
return 'red';
else if (d.goo == '0')
           return fooscale(d.foo);
        return 'gray';
  d3.selectAll('.nv-point')
        .attr({
             'stroke': colorer,
             'fill': colorer
```

```
nv.utils.windowResize(chart.update);
return chart;
});
```

See https://jsfiddle.net/qy5ohw0x/4/

PS Unfortunately Nvd3 lacks docs, so use it's github instead



Thanks for replying, I'll look into the nvd3 library! The jsfiddle doesn't seem to render the plot correctly, though. Did you test it somewhere else? — ethane Apr 11 '15 at 5:57

May be because i'm adjusted you x,y calculation and removed scale? It's for simplicity, you easily can bring that back. – mbeloshitsky Apr 11 1 5 at 6:05