SOM - Creating hexagonal heatmaps with D3.js

Self Organizing Maps series

Posted: July 22, 2013 - Likes: 0 - Comments: 13 - Categories: D3.js, Self Organizing Maps, Tutorial - Tags: D3.js, Self Organizing Maps, SOM

In my previous post I spoke a bit about a program I wrote in R that helps me perform Self Organizing Map analyses and create heatmaps; from the cleaned data file all the way to the visualization and analysis of the heatmaps
If you want to know how to add boundaries to heatmaps, see my next post that deals specifically with the boundaries once you have your heatmap all set up

One minor issue with the R program was that I wasn't allowed to hand over the part to analyse the SOM heatmaps to the client (I'm still part of a commercial business so I couldn't give the client the intellectual property of the underlying code).
So all they got was a PowerPoint with static images and our strategic findings about the (client) segmentation. Still very interesting of course, since usually we show them insights they've never found before (or has never been proven or not shown in that amount of detail), but I wanted to develop something a bit more 'sexy' I guess

Having tried to use D3.js a bit in the past few months, I knew that creating a webpage where the client can go through all the heatmaps and have a bit of analyzing capabilities would be

Storytelling (3)

Tutorial (28)

Uncategorized (2)

Recent posts

Presenting the d3.loom chart

Coding a “Reasons to” logo remix on my iPad

It hardly ever “feels right”

My journey into data visualization

Hacking the Visual Norm

See all blog posts

Search...

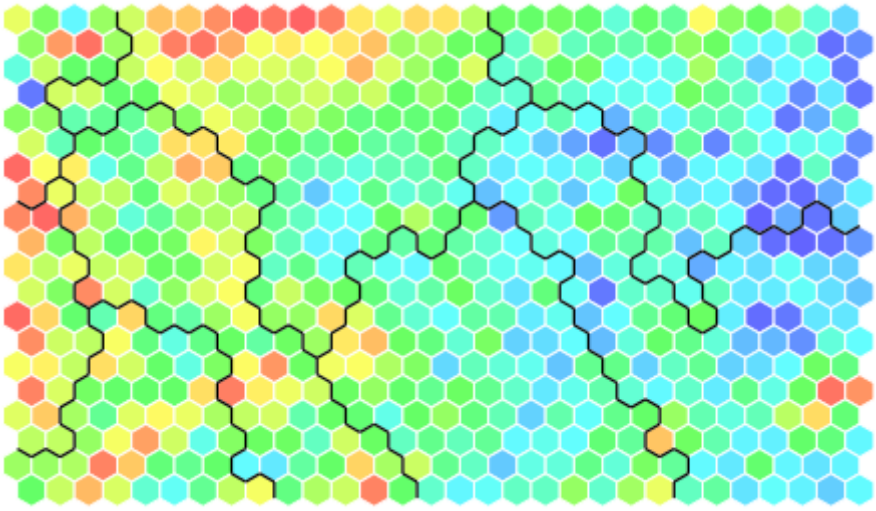
Recent projects

See all my work

just what I was looking for and D3.js makes almost everything look good ;)

I have to admit that D3 (but especially JavaScript) has been one of the most difficult programming languages I’ve tried to learn. Quite a different structure from R and IDL and even VBA and SAS came to me faster

Anyway, I used the D3 hexbin plugin, even though I have nothing to ‘bin’. By using this plugin in a slightly different way than intended I do not have to make a function that draws each hexagon myself.



Hexagons 101

There really is only one difficult step in the whole process and that is calculating the centers of each hexagon. But to do this I first need to know the map dimensions (number of rows and columns) and decide what the radius of one hexagon needs to be.

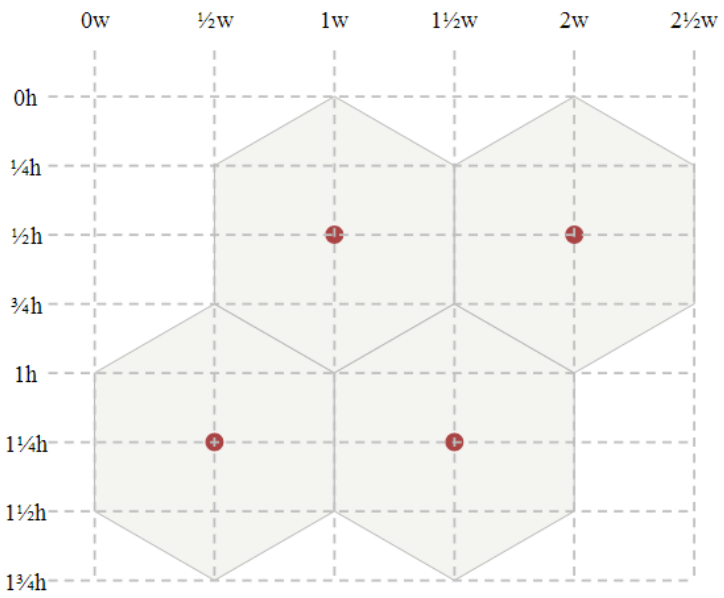
Say I want the heatmap to fit inside a rectangular shape of 850px by 350px and my map has 30 columns and 20 rows. The maximum radius that a single hexagon can have depends on which value is smaller; the radius so 30 hexagons will fit into 850px or the radius needed to still fit 20 hexagons inside 350px

Radius, height and width of a hexagon are thankfully all very much related. This site really helped me to get an understanding of hexagons and their dimensions in a grid. Say hexRadius is the radius of a hexagon, it is the circle that can be drawn so all six corners touch the circle

- The height of a hexagon: hexRadius * 2
- The width of a hexagon: hexRadius * sqrt(3)

The image below will give you an insight into what the final dimensions of the heatmap will be:

- The height of the total heatmap will be: #Rows * 3/2 * hexRadius + 1/2 * hexRadius = (#Rows + 1/3) * 3/2 * hexRadius
- The width of the total heatmap will be: #Columns * sqrt(3) * hexRadius + sqrt(3)/2 * hexRadius = (#Columns + 1/2) * sqrt(3) * hexRadius



My final formula, that calculates the 'best' hexagonal radius is the following:

```
//svg sizes and margins
var margin = {
    top: 50,
    right: 20,
    bottom: 20,
```

```
    left: 50
},
width = 850,
height = 350;

//The number of columns and rows of the heatmap
var MapColumns = 30,
    MapRows = 20;

//The maximum radius the hexagons can have to still fi
var hexRadius = d3.min([width/((MapColumns + 0.5) * Ma
   height/((MapRows + 1/3) * 1.5)]);
```

Getting the center coordinates

Well, now that we finally have a good radius, we can calculate the centers of each hexagon. Using the image above we can see that two hexagons are one width apart horizontally (sqrt(3) * hexRadius) and 3/4 height apart vertically (3/2 * hexRadius).

Strangely enough, for the hexbin plugin, you do not need to take the offset into account that occurs when going to a new row (moved 1/2 width to the left). Therefore, calculating the centers is easily done by the following for loop:

```
//Calculate the center positions of each hexagon
var points = [];
for (var i = 0; i < MapRows; i++) {
    for (var j = 0; j < MapColumns; j++) {
        points.push([hexRadius * j * 1.75, hexRadius *
    }//for j
}//for i
```

Where 1.75 ~ sqrt(3), strangely enough giving it exactly sqrt(3) resulted in a jump after about 12 hexagons. This does work, so I kept it at 1.75. Now we have an array that holds all [x,y] center coordinates, which is all you need. The hexbin plugin deals with converting this into an actual hexagon path with 6 sides!

Initiate the SVG and Hexbin

Now lets create a placeholder for the heatmap; an SVG

```
//Create SVG element
var svg = d3.select("#chart").append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottc
    .append("g")
    .attr("transform", "translate(" + margin.left + ",
```

And also initiate the Hexbin plugin:

```
//Set the hexagon radius
var hexbin = d3.hexbin()
            .radius(hexRadius);
```

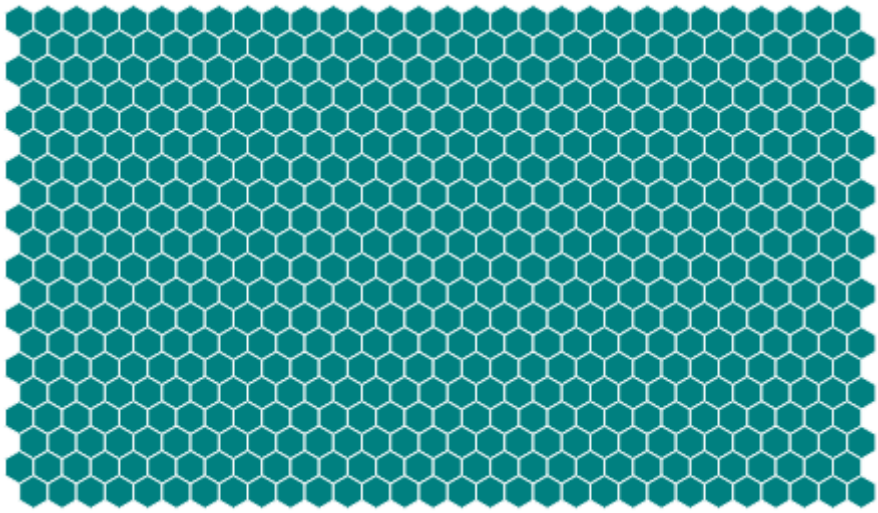
Hexagonal Grid

And the last piece of code that will draw the hexagonal grid for you. See how the data step gets “hexbin(points)”, not just “points”:

```
//Draw the hexagons
svg.append("g")
```

```
      .selectAll(".hexagon")
      .data(hexbin(points))
      .enter().append("path")
      .attr("class", "hexagon")
      .attr("d", function (d) {
    return "M" + d.x + "," + d.y + hexbin.hexagon();
      })
      .attr("stroke-width", "1px")
      .style("fill", "teal");
```

You should now have something like the image below, a uni color grid!

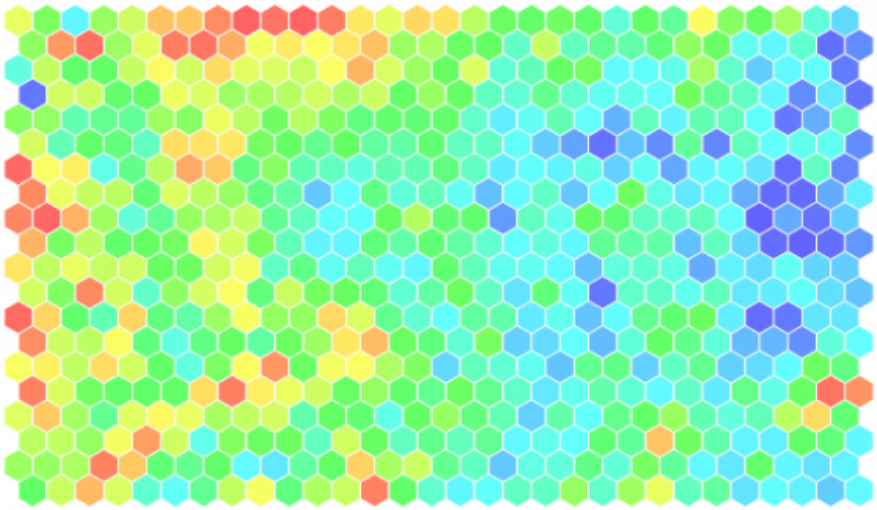


Colors

Now that we have a grid, let's add some color, which is the easy part actually. I started with an array of color hex codes, instead of numeric data, just to make my life easier (I calculated the hex codes in R).
Just replace the line where you have set the "fill" to one color for all hexagons, to one that depends on the iterator "i" of the hexagon number. Now the color of the ith hexagon will be filled with the ith entry of the array *color*:

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

The Heatmap below is the result of adding a splash of color. It actually represents the expected revenue of a supermarkets in the Netherlands, red being a lot of revenue and blue being not so much revenue



Code

You can find the code and example in this block (you might have to refresh it once though if nothing is showing up the 1st time): http://bl.ocks.org/nbremer/6052814
I added some other lines of code to have a nice mouse over event, a bit of styling, etc.

Final Result

Below you can see what I eventually made with the Heatmap code. It now has options to show multiple heatmaps (you can pick a variable from the drop down list), there is a legend that tells you which color belongs to which value and finally, there is a slider that gives you the option to tweak the color boundaries. Slide the left handle towards the right and the number below it shows where dark blue is reached, every value below the one from the handle will be dark blue as well. The Heatmap colors get updated while you slide it.
This gives me the chance to investigate a portion of the map

that is all primarily one color, by adjusting the handles I can tweak the colors to show more detail in that section. You can see the result of sliding a handle in the two images below

Self Organizing Maps

Heatmaps showing distributions per variable

Choose a Variable to show as Heatmap...

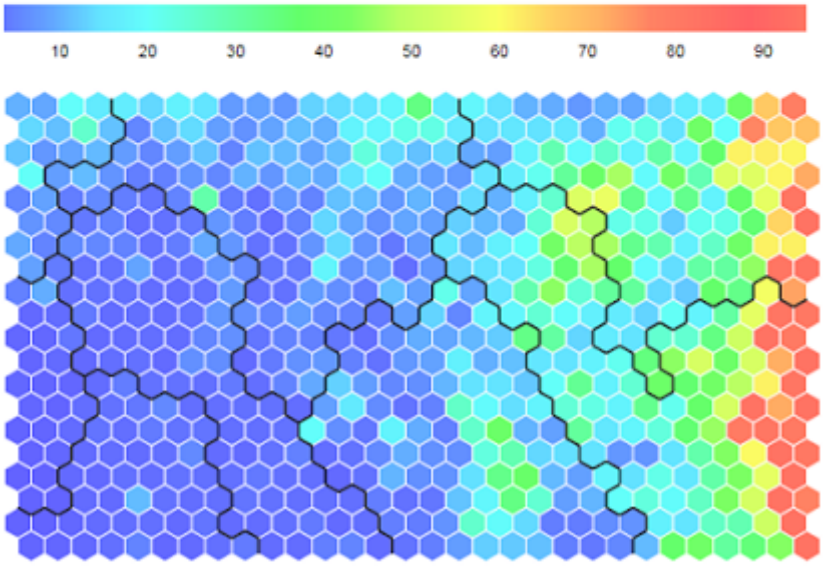
Local_Share__

SLIDER CONTROLLING THE VALUES OF THE COLOR RANGE

3.7 95

VARIABLE

Local_Share__



Self Organizing Maps

Heatmaps showing distributions per variable

Choose a Variable to show as Heatmap...

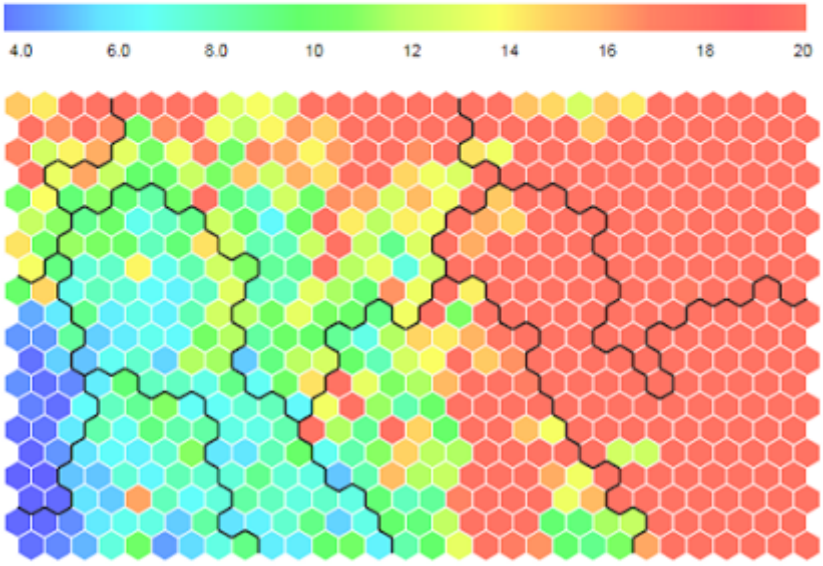
Local_Share__

SLIDER CONTROLLING THE VALUES OF THE COLOR RANGE

3.7 20

VARIABLE

Local_Share__



In my next post I explain how to calculate the black lines that form the boundary between the heatmap segments with just one ‘simple’ mathematical principle :)

Prev / Next Post

« Next | Previous »

Comments (13)

david.boaz - August 8, 2013 - Reply

Many thanks, this post saved me a lot of time.

One small comment: Instead of using the hexbin plugin, I defined a hexagon polygon and created a path from it:
var hexagonPoly=[[0,-1],[SQRT3/2,0.5],[0,1],[-SQRT3/2,0.5],[-SQRT3/2,-0.5],[0,-1],[SQRT3/2,-0.5]];

var hexagonPath = “m”+hexagonPoly.map(function(p) {return [p[0]*hexRadius, p[1]*hexRadius].join(‘,’)}).join(‘l’)+”z”;

Then, I bind to data:

vis.selectAll(‘path’) .data(nodes) .enter() .append(‘path’) .attr(“d”, function (d) { return “M” + d.x*hexRadius + “,” + d.y*hexRadius + hexagonPath; })

(Author) Nadieh - April 2, 2016 - Reply

Hi David,

Thank you for the code! I’ve started to use it on occasions when I want to plot hexagons in specific locations and it works like a charm :)

Anonymous - December 30, 2014 - Reply

Did you try this with 48+ columns? There’s a break in column 48…

merodeador - February 15, 2016 - Reply

i am also wondering why there's a gap at column 48. anyone know why this happens and/or how to fix it?

Luke - March 1, 2016 - Reply

because in "//Calculate the center positions of each hexagon "
it's not correct to
"points.push([hexRadius * j * 1.75, hexRadius * i * 1.5]);"
you can try this
var a;
var b = (3 * i) * hexRadius / 2;
if (i % 2 == 0) {
a = Math.sqrt(3) * j * hexRadius;
} else {
a = Math.sqrt(3) * (j – 0.5) * hexRadius;
}
points.push([a, b]);

Ian - June 12, 2015 - Reply

Hi I was wondering how to select a nth hexagon and change it's color for example, not via select(this).

yuler - December 14, 2015 - Reply

Would you know how to add lables into hexagonals? thank you.

sagar - October 25, 2016 - Reply

Hi,
Examples out there are plotted over maps of various countries. So,they do have a set of boundaries within which they can visualize heatmap.

But I have a data where only x,y,z co ordinates and time data is given. Are there any examples for heat map using d3.js just by using these? So how do we do a heat map if we have to read x,y,z and time co ordinates from a ".csv" file.

kindly share examples

---

paul - January 12, 2017 - Reply

Any idea how i can get this to work for smaller numbers of hexagons?
let's say start with just a single one?

mine ends up hanging in the top left with only the bottom right quarter visible:
i changed the margins to be 10 on all sides.

paul - January 12, 2017 - Reply

I had to add the following:

```
if(MapRows % 2 == 0) {
margin.left = hexRadius * 1.75;
}
else {
margin.left = hexRadius;
}

margin.top = hexRadius;
```

And

```
if(MapRows % 2 == 0) {
width = MapColumns * hexRadius *
Math.sqrt(3) + hexRadius;
}
else {
width = MapColumns * hexRadius *
Math.sqrt(3);
}

if(MapRows == 1)
width = hexRadius * 2 * MapColumns;
```

And finally change the svg object like:

```
var svg =
d3.select(".hexgrid").append("svg")
.attr("width", width)
.attr("height", height)
.append("g")
.attr("transform", "translate(" +
margin.left + "," + margin.top + ")");
```

Leave a Reply

Enter your comment here...

Recent Posts

- Presenting the d3.loom chart
- Coding a “Reasons to” logo remix on my iPad
- It hardly ever “feels right”
- My journey into data visualization
- Hacking the Visual Norm

Go to Blog

Latest Tweets

About to board my plane and fly to SF 🇺🇸 for a week YAY :D Very much looking forward to meeting friends again and d3.unconf ^_^
13 hours ago

YAY! Awesome to finally be on a #shirleychat :D Come join us this Sunday, right after I get off my plane in SF lol…
https://t.co/qqrccy74Wz
2 days ago

Hmmmm, not at all what I was expecting o_O
#d3BrokeAndMadeArt
https://t.co/n4tZ9CaQz0
2 days ago

Follow me on Twitter

Subscribe via Email

Enjoying this Website?

Be sure to subscribe to the Visual Cinnamon website and get an update when I post a new project, tutorial or blog!

Your email address

Sign up