Using SAS and D3.js to create interactive visualizations for ADaM and SDTM data

Nathan Mockler, Boehringer Ingelheim, Ridgefield, CT

ABSTRACT

While SAS® software excels at creating plain and rich text output, it struggles outside of JMP® software to create interactive visuals, and JMP limits the customization possible. In the field of data visualization, the gold standard is D3.js, a Javascript library created by Mike Bostock and used in such places as the New York Times and NPR. The advantage of D3.js is that it creates complex data-driven documents using open web standards, allowing information to be shared without having to install additional software. This paper will show how to "link" data created from SAS® using XML to this process, creating web pages that can be shared on internal or external networks to visualize clinical data.

INTRODUCTION

Every clinical trial generates tremendous amounts of data. A submission to the FDA may involve sending hundred and hundreds of tables, with meta-analyses of tens of trials often seen in submissions. With that in mind, effective methods of summarization are needed. The days of going through listings are archaic.

WHY THE FOCUS ON VISUALIZATIONS?

Visualizations also speak to our ability to see patterns (sometimes to our advantage, sometimes to our detriment). We have evolved to be visual creatures, and while it is possible to see patterns through raw listings or summarization tables, it is much easier to see patterns and hidden information through a graph or a table. Visualization is a way to map information using a series of rules. It is much more readable to laypeople as well. A PROC MEANS and a boxplot show similar information, but our brain is able to process the information quicker in a boxplot.

Drug Group	A	B	C	D
Average Age	45	67	213	78

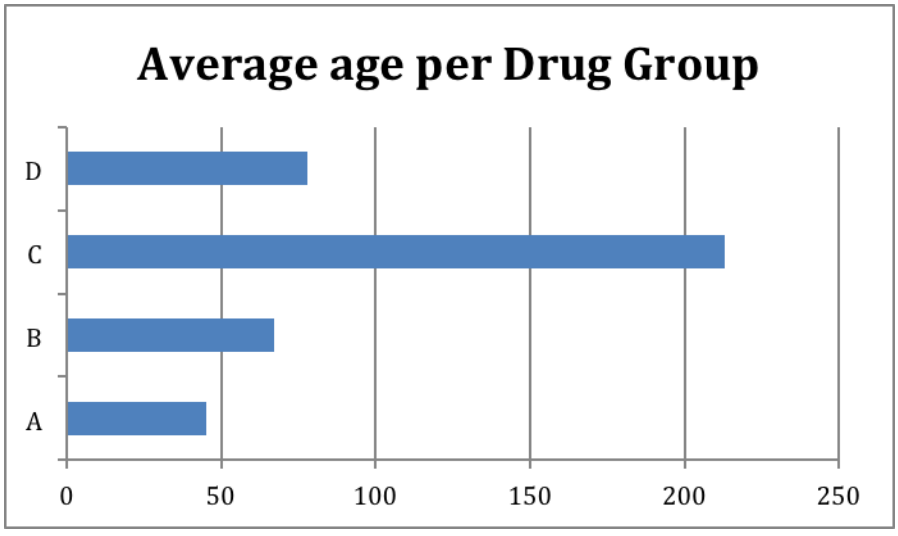


Figure 1: While the same information is displayed in both the table and figure, the information is easily assessed visually.

WHY THE FOCUS ON INTERACTION?

The bar plot example is straightforward, and can be done in SAS® (honestly, even Excel) very easily. The problem with that plot is that it is static. Any attempt to drill further into that data now requires another plot, a table, or a listing. This plot showed the problem, but once that problem is assessed, it is of little use. Now you need more help.

The good news is that interactive plots allow people to explore the data on their own. Building dynamic plots save a programmer modification time, and allows as many people to observe the data in a way that does not require an advanced degree or to learn another language. You want as many eyes digging into the data as possible (you know never know who is going to find that one anomalous thing that could've been disaster had nobody saw it). The basic needs of an interactive visualization are that it should give a broad overview at first, and then have the ability to zoom and filter, and grant details on demand.

WHY THE FOCUS ON D3 AND NOT JMP/TABLEAU/<FILL IN THE BLANK>

The good news is that there are numerous interactive visualization options out there, from Tableau to Spotfire to even SAS's own JMP®. They all do have some issues however:

- Most of these visualization options cost money, especially with site licenses.
- Some of them have no or limited capability to exporting these visualizations. Even something like JMP only allows to export as a Flash (.swf) file
- These are proprietary options, creating lock-in

One of the most interesting platforms that has come out recently is D3, a data visualization platform that stands for Data-Driven Documents. It is a free and open-source Javascript library developed by Mike Bostock at the New York Times. An offshoot of a previous visualization library, Protovis, that he developed as a graduate student in 2011. It is currently one of the fastest growing libraries in use, with multiple websites (such as the New York Times and NPR) using them for any sort of interactive display on their web.

The library allows a user to do the following:

- Load data into a browser's library using a multitude of formats through d3's functions (e.g. tab seperated values, comma separated, XML, etc.)
- Bind data into elements built into the document
- Transform these elements based on the user's input
- Use all open standards (SVG, HTML, Javascript)
-

There are limitations however. Everything is done through the web browser, so there's no "off-line" mode. Also, all the data is able to be accessed, so if looking at confidential information, it will have to be done through an intranet or online storage solution. It also only supports modern (non-IE) browsers. Also, D3 has no "canned" visualizations. If you're looking for something simple and quick SAS® itself or JMP® would be the tool. This is for high-powered, custom visualizations.

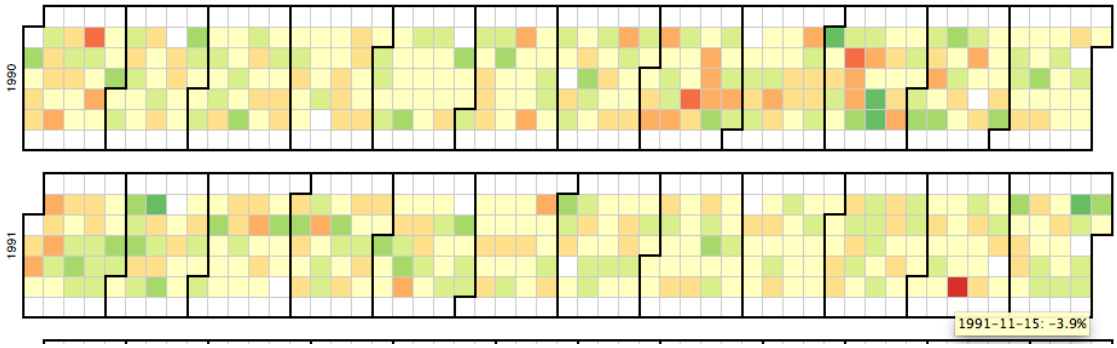


Figure 2: D3 projection of Airline Data (adapted from Rick Wicklin and Robert Allison's 2009 Data Expo presentation: http://bl.ocks.org/mbostock/4063318) Notice the tool-tip displaying each date and additional information

NOW I HAVE TO LEARN JAVASCRIPT, WEB PAGE DESIGN AND SVG? I DON'T HAVE TIME FOR THAT!

One does not need to be an expert in these to do so. Books such as *Designing Interactive Visualizations For the*

Web teach enough of each to get working. There are thousands of examples that one can use online and adapt, not to mention an active community online for using this software.

That being said, it would help if one had a basic understanding of the concepts behind the library before attempting to dive in. While that is well, well beyond the purview of this paper to do so, a quick one-paragraph overview for understanding of the following examples:

HTML is the wrapper used to structure content for web browsers, of which elements are added to it. These elements are rendered on the webpage. Javacript is a scripting language that manipulates the internal Document Object Model (DOM). One creates graphics in D3 using Scalar Vector Graphics (SVG), a cross-platform image format. In D3, you're using Javascript to manipulate the SVG using the inputs from Variables and Arrays. If you want more information, a book like *Javascript: The Easy Parts* would be useful.

We are not going to eliminate SAS® however, far from it. SAS® is still going to do the heavy lifting, and generate the CSV/HTML that we need for this. We'll use this as an export, like any other export that SAS® can do.

EXAMPLE 1: SCATTER PLOT WITH PATIENT PROFILE ON MOUSEOVER

SETTING UP

Since we're using SAS®, we need to set up a web server, at least for testing purposes. You'll need to download the library (from d3.org) as well. If you're using anything but Windows (Mac/Linux), it's very simple to do so in the terminal. If you're using Windows, try WAMPSERVER to set up your personal computer as a web server.

ACTUAL GRAPH

Let's look at the first example plot, Hba1C by Age and Race taken from DM and LB domains in SDTM, converted into ADSL and an analysis data set (Note: all data is simulated):

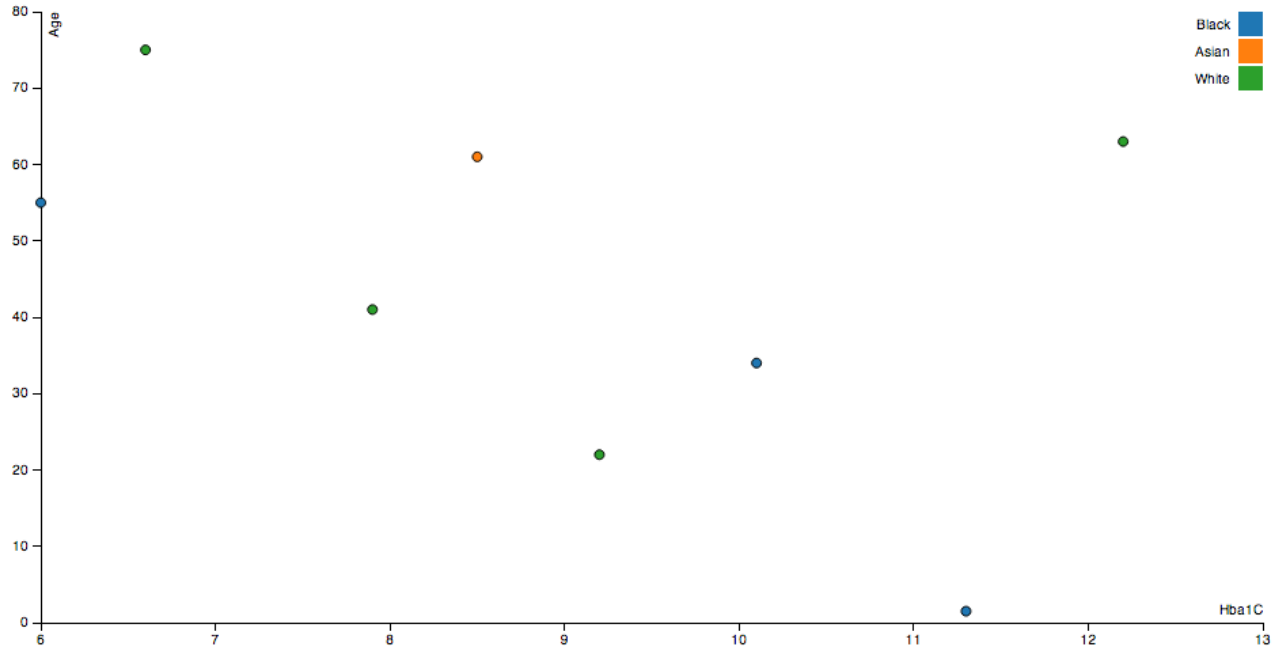


Figure 3: Basic plot in D3

This looks like a very simple plot. It has each patient represented by a dot on this scatterplot, with each race as a different color. This would be very simple to do in SAS®. However, if one looks at the plot, there is someone who has a very young. Normally, an email would be sent, asking to identify the patient and any relevant information. However, if the person hovers over that dot, the following shows up:

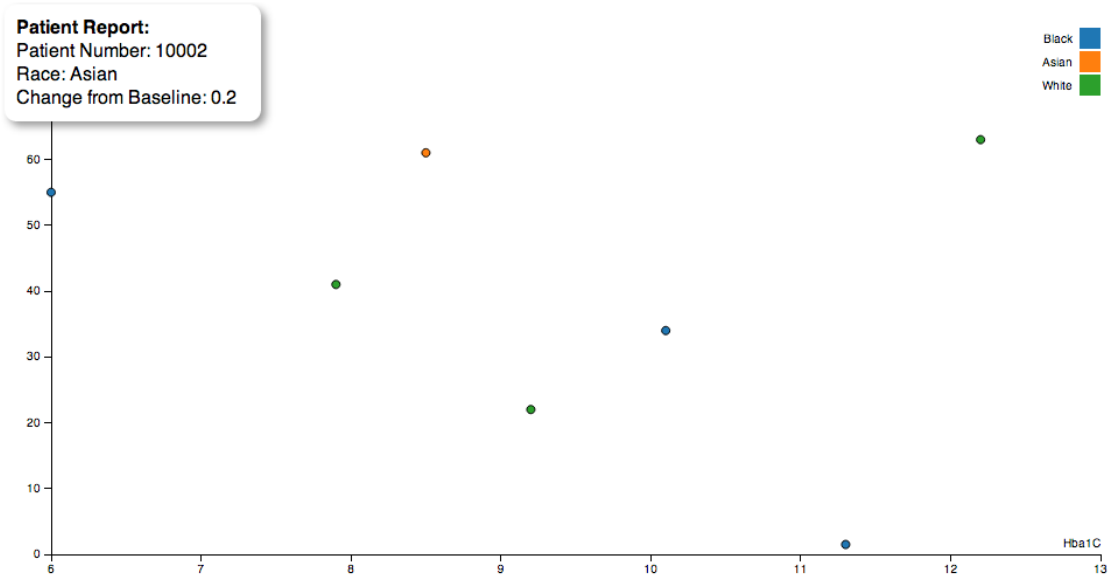


Figure 4: Hover over any dot, a real-time report comes up.

A patient profile, built right into the graph when the mouse hovered over that point. How did we do this?

First, let's look at the code to generate the dataset. D3 can utilize any type of data. For this exercise, it will just be a tab seperated values for simplicities' sake, but it could be applied to CSV/TSV/XML (taking it from XML would be ideal).

```
proc sql;
    create table d3plot as
        select
            adsl.subjid,
            adsl.age,
            adsl.Race,
            adlab.aval as HbA1C,
            adlab.chg as BLChange
        from adsl as a
             inner join
             adlab as b
             on a.subjid = b.subjid
        where adlab.paramcd = "HBA1C";
    quit;
```

This is a relatively simple join combining data from adsl (subjid, age, race) and adding lab parameters from an ADLAB Adam data set.

We have two options for generating the index.html that contains the Javascript needed to render this information

- Keep the HTML completely separate from the SAS. In that case, you can "hard-code" the variables that you need right into index.html, and that keeps it simplified. The code is in Appendix A
- Embed the HTML into the SAS code using a data _null_ statement. This is more difficult, but allows you greater customization and control of what the display is. You can then use the X command to launch this from Chrome.

A sample of this code is shown below:

```
%macro d3plot(data =, vars =, plotvars=, out = );
  proc sql;
     create table head (string char(250));
     insert into head
```

```
      set string = '        <!DOCTYPE html>     '
      set string = '        <head>     '
      set string = '        <meta charset="utf-8">     '
      set string = '        <style>     '
      set string = '        body {     '

…. Etc.
%Mend;
```

That will generate the HTML. The next step is using a PROC EXPORT for exporting the TSV file, and then having SAS open Chrome (or any other web browser) to generate the results.

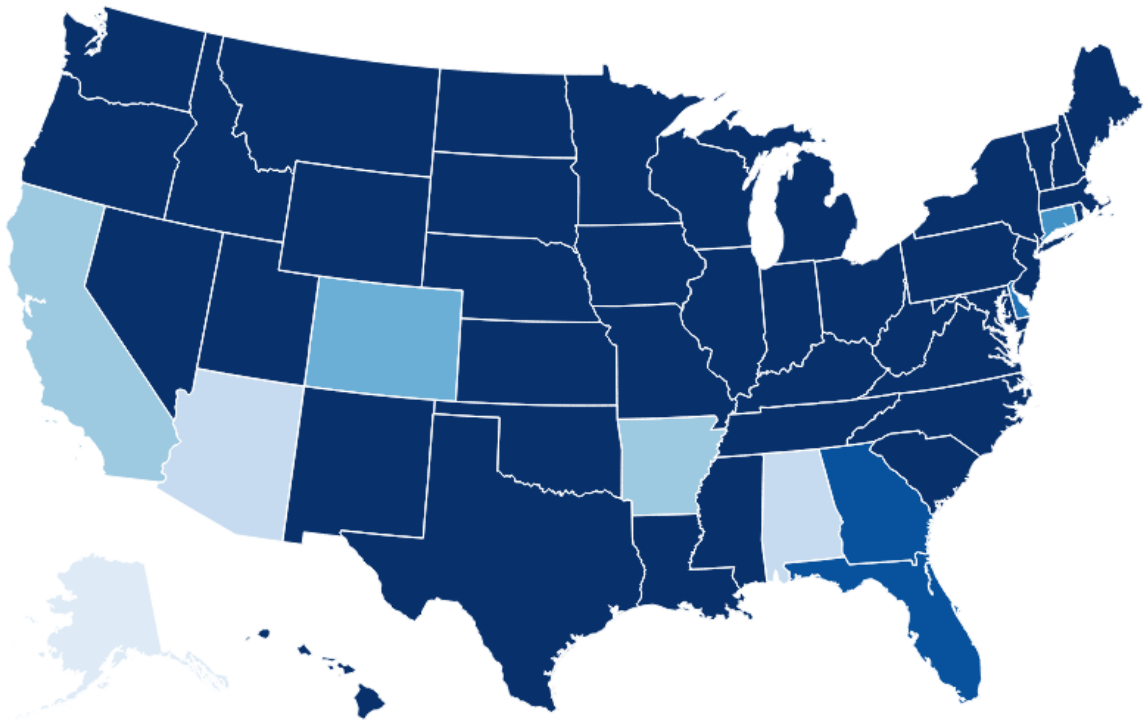
```
x 'chrome.exe <html file generated>';
```

In that case, needed variables are macro-tized (variables in the profile and plotted variables). In the interest of brevity (since it's mostly just putting the HTML together), the rest of the macro has been omitted. Hopefully, the power of this approach is clear at this point. It combines the data manipulation of SAS with the visual ability of D3. We'll go through another example:

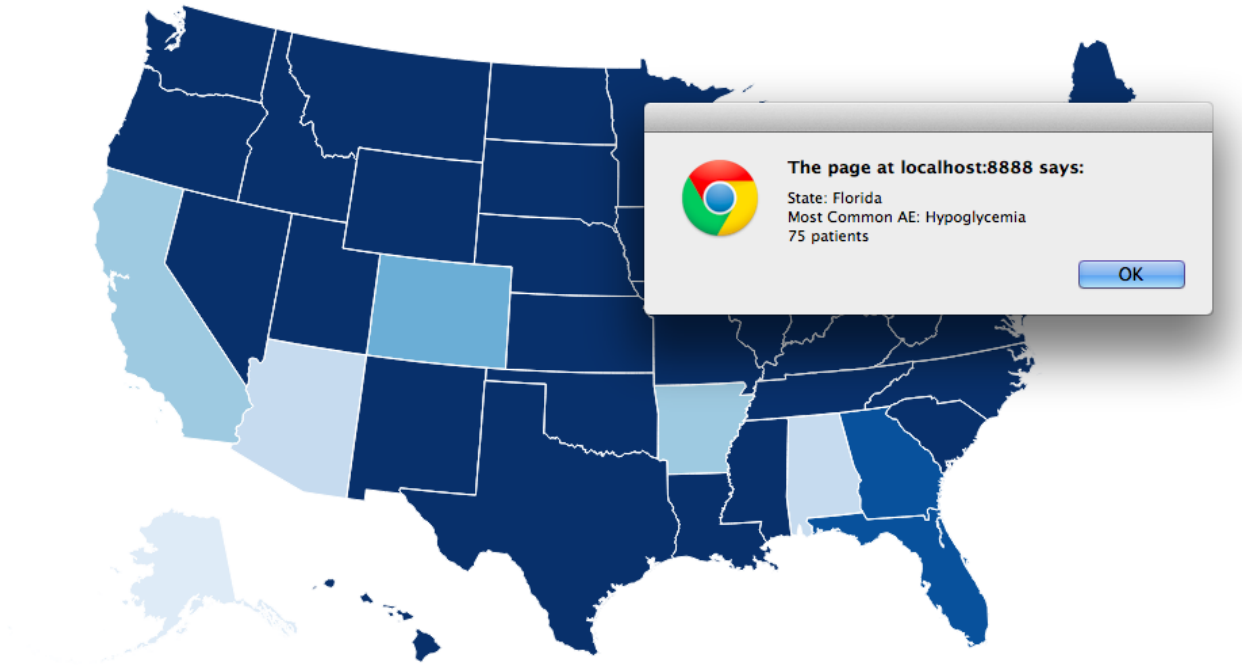
EXAMPLE 2: CHLOROPLETH MAP WITH INFORMATIONAL DIALOG BOX

You can do a Chloropleth map using PROC GMAP, but your interactivity options are limited. Using AE data from SDTM, it's pretty easy to generate this.

As a review, a chloropleth map is a map in which areas are shaded in proportion to a variable (in this case, incidence of AEs by state in a 50 state trial).



However, let's say we wanted to show the most prevalent AE in each state. We can do that as well.



That can be done by using an alert() statement in the code. The code is very similar to the first plot, if not much less verbose (Appendix B).

The process is very similar. We'll just do a frequency calculation from the AE SDTM dataset by state (since "state" is not collected in the AE domain, you can either take it from a supplemental domain, or just calculate it yourself)

```
proc freq data=sdtm.ae;
     table state;
run;
```

Of course, you'll want to filter it (do you only count each patient once, in that case, just grab a unique number of patients per state.

Then, in a similar process to the first example, you can either keep it as a separate .html file, or modify it right in SAS. That code is in Appendix B. The key variable for that is that you can set an alert() – a dialog box – to bind to the state that you click.

CONCLUSION

This only scratches the surfaces of what D3 can do. It specializes in doing transitions, animations, filtering and other modern functionality for infographics. With SAS as the computational backend, it is possible to create effective and attractive interactive graphs using standardized inputs that can help detect data quality issues and visualize difficult concepts both for experience programmers and laypeople.

REFERENCES

- "Interactive Data Visualization for the Web" (O'Reilly). Copyright 2013 Scott Murray, 978-1-449-33973-9.
- "Getting Started with D3 "(O'Reilly). Copyright 2012 Mike Dewar, 978-1-449-32879-5.
- "D3 Tips and Tricks" by Michael McLean. Copyright 2013 Michael McLean
- "Chloropleth Map" http://bl.ocks.org/mbostock/4060606
- "Scatterplot" http://bl.ocks.org/mbostock/3887118

ACKNOWLEDGMENTS

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Nathan Mockler
Boehringer Ingelheim
900 Ridgebury Road
Ridgefield, CT, 06776
Work Phone: (203) 791-6245
Email: Nathan.Mockler@boehringer-ingelheim.com

****

APPENDIX A: SCATTER PLOT WITH PATIENT PROFILE ON HOVER

Adapted from “Scatterplot” http://bl.ocks.org/mbostock/3887118

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

body {
 font: 10px sans-serif;
}

.axis path,
.axis line {
 fill: none;
 stroke: #000;
 shape-rendering: crispEdges;
}

.dot {
 stroke: #000;
}

      #tooltip {
        position: absolute;
        width: 200px;
        height: auto;
        padding: 10px;
        background-color: white;
        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        border-radius: 10px;
        -webkit-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
        -moz-box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
        box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.4);
        pointer-events: none;
      }

      #tooltip.hidden {
        display: none;
      }

      #tooltip p {
        margin: 0;
        font-family: sans-serif;
        font-size: 16px;
        line-height: 20px;
      }


</style>
<body>
 <div id="tooltip" class="hidden">
      <p><strong>Patient Report:</strong></p>
      <p><span id="value">100</span></p>
   </div>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script>
```

```
var margin = {top: 20, right: 20, bottom: 30, left: 40},
   width = 960 - margin.left - margin.right,
   height = 500 - margin.top - margin.bottom;

var x = d3.scale.linear()
   .range([0, width]);

var y = d3.scale.linear()
   .range([height, 0]);

var color = d3.scale.category10();

var xAxis = d3.svg.axis()
   .scale(x)
   .orient("bottom");

var yAxis = d3.svg.axis()
   .scale(y)
   .orient("left");

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

d3.tsv("data.tsv", function(error, data) {
 data.forEach(function(d) {
   d.Ptno = +d.Ptno;
   d.Hba1C = +d.Hba1C;
   d.Age   = +d.Age;
   d.BLChange = +d.BLChange
 });

 x.domain(d3.extent(data, function(d) { return d.Hba1C; })).nice();
 y.domain(d3.extent(data, function(d) { return d.Age; })).nice();

 svg.append("g")
     .attr("class", "x axis")
     .attr("transform", "translate(0," + height + ")")
     .call(xAxis)
   .append("text")
     .attr("class", "label")
     .attr("x", width)
     .attr("y", -6)
     .style("text-anchor", "end")
     .text("Hba1C");

 svg.append("g")
     .attr("class", "y axis")
     .call(yAxis)
   .append("text")
     .attr("class", "label")
     .attr("transform", "rotate(-90)")
     .attr("y", 6)
     .attr("dy", ".71em")
     .style("text-anchor", "end")
     .text("Age")

 svg.selectAll(".dot")
```

```
      .data(data)
    .enter().append("circle")
      .attr("class", "dot")
      .attr("r", 3.5)
      .attr("cx", function(d) { return x(d.Hba1C); })
      .attr("cy", function(d) { return y(d.Age); })
      .style("fill", function(d) { return color(d.Race); })
      .on("mouseover", function(d) {

          //Get this bar's x/y values, then augment for the tooltip
          var xPosition = parseFloat(d3.select(this).attr("x")) - 10;
          var yPosition = parseFloat(d3.select(this).attr("y")) - 10 ;

          //Update the tooltip position and value
          d3.select("#tooltip")
            .style("left", xPosition + "px")
            .style("top", yPosition + "px")
            .select("#value")
            .html("Patient Number:  " + d.subjid + "<br> Race: " + d.Race +
"<br> Change from Baseline: " + d.BLChange);

          //Show the tooltip
          d3.select("#tooltip").classed("hidden", false);

         })

       .on("mouseout", function() {

          //Hide the tooltip
          d3.select("#tooltip").classed("hidden", true);

         })


 var legend = svg.selectAll(".legend")
      .data(color.domain())
    .enter().append("g")
      .attr("class", "legend")
      .attr("transform", function(d, i) { return "translate(0," + i * 20 + ")";
});

 legend.append("rect")
      .attr("x", width - 18)
      .attr("width", 18)
      .attr("height", 18)
      .style("fill", color);

 legend.append("text")
      .attr("x", width - 24)
      .attr("y", 9)
      .attr("dy", ".35em")
      .style("text-anchor", "end")
      .text(function(d) { return d; });

});

</script>
```

APPENDIX B: CHLOROPLETH WITH AE DATA ON CLICK

Adapted from “Chloropleth Map” http://bl.ocks.org/mbostock/4060606

```
<!DOCTYPE html>
<meta charset="utf-8">
<head> <title>D3: Chloropleth layout</title> </head>
<style>


.counties {
 fill: none;
}

.states {
 fill: none;
 stroke: #fff;
 stroke-linejoin: round;
}

.q0-9 { fill:rgb(247,251,255); }
.q1-9 { fill:rgb(222,235,247); }
.q2-9 { fill:rgb(198,219,239); }
.q3-9 { fill:rgb(158,202,225); }
.q4-9 { fill:rgb(107,174,214); }
.q5-9 { fill:rgb(66,146,198); }
.q6-9 { fill:rgb(33,113,181); }
.q7-9 { fill:rgb(8,81,156); }
.q8-9 { fill:rgb(8,48,107); }

</style>
<body>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="http://d3js.org/queue.v1.min.js"></script>
<script src="http://d3js.org/topojson.v1.min.js"></script>
<script>

var width = 960,
   height = 500;

var rateById = d3.map();

var quantize = d3.scale.quantize()
   .domain([0, .15])
   .range(d3.range(9).map(function(i) { return "q" + i + "-9"; }));

var path = d3.geo.path();

var svg = d3.select("body").append("svg")
   .attr("width", width)
   .attr("height", height);

queue()
   .defer(d3.json, "us.json")
   .defer(d3.tsv, "unemployment.tsv", function(d) { rateById.set(d.id,
+d.rate); })
   .await(ready);
```

```
function ready(error, us) {
 svg.append("g")
     .attr("class", "states")
   .selectAll("path")
     .data(topojson.feature(us, us.objects.states).features)
   .enter().append("path")
     .attr("class", function(d) { return quantize(rateById.get(d.id)); })
     .attr("d", path)
           .on('click', function(d){
      alert("State: d.State\nMost Common AE: d.TopAE\nd.PtCount patients");
     });

 svg.append("path")
     .datum(topojson.mesh(us, us.objects.states, function(a, b) { return a !==
b; }))
     .attr("class", "states")
     .attr("d", path);
}

</script>
```