SVG Bar Chart Tutorial

Eduardo Navas (ean13@psu.edu)

The basic code for the following instructions can be found in the book *Interactive Data Visualization* by Scott Murray

You can develop a basic bar-chart visualization, using SVGs as follows:

You need to link the D3 library to the html page you will be editing. Make sure to move your d3.v3.js page to the same folder in which you placed your html page. Inside the <head></head> tag of your html page include the following:

<head>

<script type = "text/javascript" src = "d3.v3.js"></script>

</head>

Next type your tags declaring javacript: <script type = “text/javascript”></script> inside the <body></body> tags.

It should look like this:

<body>

<script type = "text/javascript">

</script>

</body>

Next, inside your Javascript tags type the variables you will be using. Also declare the size of your SVG. You will also need some bar padding to make sure that each element is visible and well defined The declaration of your variables should look as follows:

```
var dataset = [25,7,5,100,11,30,7,5, 80,11,35,7,5,50,11, 25,40,5,60,11]
var w = 1000;
var h = 500;
var barPadding = 5;
```

Now we are going to create the svg:

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

Note that in the above code the svg is declared as a variable. Then attritutes are attached by using D3's own syntax of .() Note that we instruct the browser to place the SVG within the body tag, to which "svg" is appended. Next width and height are respectively attributed by declaring the properties (width and height), to which the variables of w and h are passed. Note that these variables were previously declared above.

Now we can design the content that will appear within the SVG. Write the following:

```
svg.selectAll("rect")
        .data(dataset)
        .enter()
        .append("rect")
        .attr("x", function(d, i){
                return i * (w / dataset.length) ;
                })
        .attr("y", function(d) {
                return h - d*4;
        })
        .attr("width", w / dataset.length - barPadding)
        .attr("height", function(d){
                return d *4;
        })
        .attr("fill", function(d){
        return "rgb(255, 0, " + (d * 3) + ")";
        })
        .attr("stroke", "rgba(155, 255, 50, 0.66)")
        .attr("stroke-width", function(d){
                                return d/4;
        });
```

That's a lot of code. Let's take a closer look. The code set up is straight forward. It's basically a call for an SVG followed by a series of chained attributes: "svg.( ).( ).( ).( ).()" But it looks complex because it includes functions that call up previously declared variables (methods found in the D3 library document) to define the size of the bar as well as the colors of the bars. Let's look at the first part closer:

```
svg.selectAll("rect")
        .data(dataset)
        .enter()
        .append("rect")
        .attr("x", function(d, i){
```

```
            return i * (w / dataset.length) ;
            })
        .attr("y", function(d) {
            return h - d*4;
        })
```

Here we have the SVG attributed the form of a rect with the attribute .selectAll("rect") which means that all SVG elements will be rects. Next we need to pass the data that will define the size of the rects .data(dataset). Next we enter this information .enter() and append it to the SVG with .append("rect"). Next we need to define the width and the height of each rectangle on the x, y axis and we do this with the x, y attributes highlighted above. Note that on the left we have x and y in quotes, while on the right we have a function which takes the dataset information in the order listed to create the respective bars.

Next we need to define the fill color as well as the stroke color. This is shown in the code below:

```
.attr("fill", function(d){
        return "rgb(255, 0, " + (d * 3) + ")";
        })
        .attr("stroke", "rgba(155, 255, 50, 0.66)")
        .attr("stroke-width", function(d){
                                return d/4;
        });
```

Note that the fill has a function which uses the dataset of numbers to define the color for each bar. Notice that the stroke has the similar attributes except that its color does not vary according to the dataset. Instead it has a straightforward decimal color code. The stroke's width is defined according to the data for each bar created divided by 4. If you change this number the y placement for each bar will change. And will not be flush to the bottom of the SVG.

(Continues on next page)

Now we need to display the actual number that defines the bars on top of the bars themselves. The code is below:

```
svg.selectAll("text")
        .data(dataset)
        .enter()
        .append("text")
        .text(function(d) {
        return d;
        })
        .attr("x", function(d, i){
```

```
            return i *(w / dataset.length) + (w / dataset.length - barPadding) /2;
            })
        .attr("y", function(d) {
            return h - (d*4) + 16;
        })
        .attr("font-family", "sans-serif")
        .attr("font-size", "11 px")
        .attr("fill", "white")
        .attr("text-anchor", "middle")
```

Let's take a closer look. Again we call up the svg and in this case we use the attribute .selectAll() to declare that we will be using text in this case. Then we append the dataset and follow it up with .enter(), next we use the attribute .text() to return the dataset as text. We place the text in similar fashion to the rects on the x, y axis. Finally we need to define the font, font-size, the fill and the text-anchor to make sure the numbers will display properly on top of the bars. And we should have a barchart with some variations of the same color.

The entire code appears below. In the next section below I explain how to make detailed changes to redesign your barchart.

```
<script type = "text/javascript">

var dataset = [25,7,5,100,11,30,7,5, 80,11,35,7,5,50,11, 25,40,5,60,11]
var w = 1000;
var h = 500;
var barPadding = 5;

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

svg.selectAll("rect")
        .data(dataset)
        .enter()
        .append("rect")
        .attr("x", function(d, i){
            return i * (w / dataset.length) ;
            })
        .attr("y", function(d) {
            return h - d*4;
        })
        .attr("width", w / dataset.length - barPadding)
```

```
        .attr("height", function(d){
                return d *4;
        })
        .attr("fill", function(d){
        return "rgb(255, 0, " + (d * 3) + ")";
        })
        .attr("stroke", "rgba(155, 255, 50, 0.66)")
        .attr("stroke-width", function(d){
                                return d/4;
        });

svg.selectAll("text")
        .data(dataset)
        .enter()
        .append("text")
        .text(function(d) {
        return d;
        })
        .attr("x", function(d, i){
                return i *(w / dataset.length) + (w / dataset.length - barPadding) /2;
                })
        .attr("y", function(d) {
                return h - (d*4) + 16;
        })
        .attr("font-family", "sans-serif")
        .attr("font-size", "11 px")
        .attr("fill", "white")
        .attr("text-anchor", "middle")


</script>
```

Redesigning your barchart

You can redesign your SVG barchart in many ways. Below are some suggestions. To make the SVG smaller or bigger, enter a different number for the width and the height variables:

```
var w = 1000;
var h = 500;
```

You can also change the separation between the bars by changing the number of the barPadding:

```
var barPadding = 5;
```

When you look over the code above, experiment with the attributes for both the text and the barcharts. Make sure to save a copy of your work as you achieve the effects you desire.

There is one last thing that you can do to your barchart to make it more exciting. You can create a simple hover effect by placing the attributes in the CSS styles at the top of your web page between the <head></head> tags. It should look as follows:

```
<style type="text/css">

        rect:hover {  fill: orange;
                        }

                .style1 {
        font-family: Arial, Helvetica, sans-serif;
        font-size: 14pt;
}
</style>
```

This will create a hover effect. Note that you can choose the color of the fill for the hover effect.