E6893 Big Data Analytics Lecture 8:

*Big Data Visualization — II*

Ching-Yung Lin, Ph.D.
Adjunct Professor, Dept. of Electrical Engineering and Computer Science

October 25th, 2018
Outline

Part I: SVG and its applications
Part II: From SVG to D3
SVG Introduction

SVG Examples

Basic Shapes

- Lines
- Rectangles
- Circles
- Ellipses
- Polygons
- Polylines

Advanced Shapes (Paths)

- Arcs
- Quadratic Bezier Curves
- Cubic Bezier Curves
SVG Examples

Text

Text

Text along a curved path

Images

?
SVG Examples

Layering + Opacity

Text on Shape

Transformations

SVG Transformations

Rotated Text
SVG Examples

Animations

Use Cases

Graphs

Bar Charts

Pie Charts
A Simple SVG Example

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="10" y="10" height="100" width="100"
       style="stroke:#ff0000; fill: #0000ff"/>
</svg>
```
SVG as Background Images

Since the browsers treat SVG images just like bitmap images, you can use SVG images as background images via CSS. Here is an example:

```html
<svg>
  <circle cx="40" cy="40" r="24" style="stroke:#006600; fill:#00cc00"/>
</svg>
</div>
```

svg Element Inside HTML

Embedding an SVG image using the `svg` element can be done by embedding an `svg` element directly in an HTML page, like this:
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <g>
    <line x1="10" y1="10" x2="85" y2="10"
          style="stroke: #006600;"/>
    <rect x="10" y="20" height="50" width="75"
          style="stroke: #006600; fill: #006600"/>
    <text x="10" y="90" style="stroke: #660000; fill: #660000">
      Text grouped with shapes</text>
  </g>
</svg>

Text grouped with shapes
SVG strokes and fill

```xml
<circle cx="40" cy="40" r="24"
  style="stroke:#006600;
           stroke-width: 3;
           fill:#00cc00"/>
```

```xml
<circle cx="40" cy="40" r="24"
  style="stroke:#006600;
           stroke-width: 3;
           stroke-dasharray: 10 5;
           fill:#00cc00"/>
```

```xml
<circle cx="40" cy="40" r="24"
  style="stroke: none;
           fill:#00cc00"/>
```

```xml
<circle cx="40" cy="40" r="24"
  style="stroke: #006000;
           fill: none"/>
```

```xml
<circle cx="40" cy="40" r="24"
  style="stroke: #660000;
           fill: #cc0000"/>
<circle cx="64" cy="40" r="24"
  style="stroke: #000066;
           fill: #0000cc"
           fill-opacity: 0.5"/>
```
SVG path Element

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
    <path d="M50,50
           A30,30 0 0,1 35,20
           L100,100
           M110,110
           L100,0"
           style="stroke:#660000; fill:none;"/>
</svg>
```

A: Arc
M: Move
L: Line
The radius of the arc is set by the two first parameters set on the A command. The first parameter is \( r_x \) (radius in x-direction) and the second is \( r_y \) (radius in y-direction). Setting \( r_x \) and \( r_y \) to the same value will result in a circular arc. Setting \( r_x \) and \( r_y \) to different values will result in an elliptical arc. In the example above \( r_x \) was set to 30 and \( r_y \) to 50.

The third parameter set on the A command is the \( x\text{-axis-rotation} \). This sets the rotation of the arc's x-axis compared to the normal x-axis. In the above example the \( x\text{-axis-rotation} \) is set to 0. Most of the time you will not need to change this parameter.

The fourth and fifth parameters are the \text{large-arc-flag} and \text{sweep-flag} parameters. The \text{large-arc-flag} determines whether to draw the smaller or bigger arc satisfying the start point, end point and \( r_x \) and \( r_y \).
SVG path Element

```
<path d="M40,20 A30,30 0 0,0 70,70"
     style="stroke: #cccc00; stroke-width:2; fill:none;"/>

<path d="M40,20 A30,30 0 1,0 70,70"
     style="stroke: #ff0000; stroke-width:2; fill:none;"/>

<path d="M40,20 A30,30 0 1,1 70,70"
     style="stroke: #00ff00; stroke-width:2; fill:none;"/>

<path d="M40,20 A30,30 0 0,1 70,70"
     style="stroke: #0000ff; stroke-width:2; fill:none;"/>
```
<text x="20" y="40"
    style="fill: #000000; stroke: none; font-size: 48px;">
  Fill only
</text>
<text x="20" y="100"
    style="fill: none; stroke: #000000; font-size: 48px;">
  Stroke only
</text>
<text x="20" y="150"
    style="fill: #999999; stroke: #000000; font-size: 48px;">
  Fill and stroke
</text>

Fill only
Stroke only
Fill and stroke
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

  <text x="20" y="40"
       transform="rotate(30 20,40)"
       style="stroke:none; fill:#000000;"
       >Rotated SVG text</text>

</svg>

Rotated SVG text

<text x="10" y="20" style="writing-mode: tb;"
      >Vertical</text>

Vertical
SVG Transformation

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
    <rect x="50" y="50" height="110" width="110"
         style="stroke:#ff0000; fill: #ccccff"
         transform="translate(30) rotate(45 50 50)"
    />
    <text x="70" y="100"
         transform="translate(30) rotate(45 50 50)"
         >Hello World</text>
</svg>
```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

    <rect x="10" y="10" height="110" width="110"
         style="stroke:#ff0000; fill: #0000ff">

        <animateTransform
            attributeName="transform"
            begin="0s"
            dur="20s"
            type="rotate"
            from="0 60 60"
            to="360 60 60"
            repeatCount="indefinite"
        />
    </rect>

</svg>
### Overview of Animation Options

1. `<set>`
2. `<animate>`
3. `<animateColor>`
4. `<animateTransform>`
5. `<animateMotion>`

---

**set**

```xml
<circle cx="30" cy="30" r="25" style="stroke: none; fill: #0000ff;">
  <set attributeName="r" attributeType="XML"
       to="100"
       begin="0s" />
</circle>
```

---
 animate

```
<circle cx="30" cy="30" r="25" style="stroke: none; fill: #0000ff;">
    <animate attributeName="cx" attributeType="XML"
        from="30" to="470"
        begin="0s" dur="5s"
        fill="remove" repeatCount="indefinite"/>
</circle>
```
animateTransform

```xml
<rect x="20" y="20" width="100" height="40"
    style="stroke: #ff00ff; fill:none;" >
    <animateTransform attributeName="transform"
        type="rotate"
        from="0 100 100" to="360 100 100"
        begin="0s" dur="10s"
        repeatCount="indefinite"
    />
</rect>
```
animateMotion

```xml
<rect x="0" y="0" width="30" height="15"
     style="stroke: #ff0000; fill: none;">
  <animateMotion
      path="M10,50 q60,50 100,0 q60,-50 100,0"
      begin="0s" dur="10s" repeatCount="indefinite"/>
</rect>
```
SVG Gradients

Linear Gradients

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

<defs>
    <linearGradient id="myLinearGradient1"
        xlink:href="#myLinearGradient1"

        x1="0%" y1="0%"
        x2="0%" y2="100%"
        spreadMethod="pad">
        <stop offset="0%" stop-color="#00cc00" stop-opacity="1"/>
        <stop offset="100%" stop-color="#006600" stop-opacity="1"/>
    </linearGradient>
</defs>

<rect x="10" y="10" width="75" height="100" rx="10" ry="10"
      fill: url(#myLinearGradient1);
      stroke: #005000;
      stroke-width: 3; />

</svg>
```
SVG Gradients

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

<defs>
    <linearGradient id="myLinearGradient1"
        x1="0%" y1="0%"
        x2="100%" y2="0%"
        spreadMethod="pad">
        <stop offset="10%" stop-color="#00cc00" stop-opacity="1"/>
        <stop offset="30%" stop-color="#006600" stop-opacity="1"/>
        <stop offset="70%" stop-color="#cc0000" stop-opacity="1"/>
        <stop offset="90%" stop-color="#000099" stop-opacity="1"/>
    </linearGradient>
</defs>

<rect x="10" y="10" width="500" height="50" rx="10" ry="10"
     style="fill:url(#myLinearGradient1); stroke: #005000;
             stroke-width: 3;" />
</svg>
```
SVG Gradients

Radial Gradients

```xml
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

  <defs>
    <radialGradient id="myRadialGradient4"
                    fx="5%" fy="5%" r="65%"
                    spreadMethod="pad">
      <stop offset="0%" stop-color="#00ee00" stop-opacity="1"/>
      <stop offset="100%" stop-color="#006600" stop-opacity="1"/>
    </radialGradient>
  </defs>

  <rect x="340" y="10" width="100" height="100" rx="10" ry="10"
        style="fill:url(#myRadialGradient4);
                stroke: #005000; stroke-width: 3;"/>

</svg>
```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <rect x="10" y="10" height="130" width="500" style="fill: #000000"/>
  <image x="20" y="20" width="300" height="80"
  <line x1="25" y1="80" x2="350" y2="80"
       style="stroke: #ffffff; stroke-width: 3;"/>
</svg>
SVG Maps

https://www.tutorialspoint.com/svg/maps.htm
D3 Introduction and Examples
D3 Website

http://d3js.org

Data-Driven Documents
D3 Installation

Installing

For NPM, `npm install d3`. For Yarn, `yarn add d3`. Otherwise, download the latest release. The released bundle supports AMD, CommonJS, and vanilla environments. Create a custom bundle using Rollup or your preferred bundler. You can also load directly from d3js.org:

```html
<script src="https://d3js.org/d3.v5.js"></script>
```

For the minified version:

```html
<script src="https://d3js.org/d3.v5.min.js"></script>
```

You can also use the standalone D3 microlibraries. For example, `d3-selection`:

```html
<script src="https://d3js.org/d3-selection.v1.min.js"></script>
```
D3 Selections

In Javascript:

```javascript
var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
    var paragraph = paragraphs.item(i);
    paragraph.style.setProperty("color", "white", null);
}
```

D3 employs a declarative approach, operating on arbitrary sets of nodes called selections. For example, you can rewrite the above loop as:

```javascript
d3.selectAll("p").style("color", "white");
```
D3 Dynamic Properties

Randomly color paragraphs:

```javascript
let randomColor = function() {
  return `hsl(${Math.random() * 360}, 100%, 50%)`;
}

d3.selectAll('p').style('color', function() {
  return randomColor();
});
```

Alternate shades of gray for even and odd nodes:

```javascript
let evenOddColor = function(d, i) {
  return i % 2 ? '#fff' : '#eee';
}

d3.selectAll('p').style('color', function(d, i) {
  return evenOddColor(d, i);
});
```
Randomly color paragraphs:

```javascript
d3.selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .style("font-size", function(d) { return d + "px"; });
```

Computed properties often refer to bound data. Data is specified as an array of values, and each value is passed as the first argument (d) to selection functions. With the default join-by-index, the first element in the data array is passed to the first node in the selection, the second element to the second node, and so on. For example, if you bind an array of numbers to paragraph elements, you can use these numbers to compute dynamic font sizes:
Using D3’s `enter` and `exit` selections, you can create new nodes for incoming data and remove outgoing nodes that are no longer needed.

When data is bound to a selection, each element in the data array is paired with the corresponding node in the selection. If there are fewer nodes than data, the extra data elements form the enter selection, which you can instantiate by appending to the `enter` selection. For example:

```javascript
d3.select("body")
  .selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .enter().append("p")
  .text(function(d) { return "I’m number " + d + "!"; });
```
```javascript
// Update...
var p = d3.select("body")
  .selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .text(function(d) { return d; });

// Enter...
p.enter().append("p")
  .text(function(d) { return d; });

// Exit...
p.exit().remove();
```
D3 Transformation

For example, to fade the background of the page to black:

```javascript
d3.select("body").transition()
  .style("background-color", "black");
```

Or, to resize circles in a symbol map with a staggered delay:

```javascript
d3.selectAll("circle").transition()
  .duration(750)
  .delay(function(d, i) { return i * 10; })
  .attr("r", function(d) { return Math.sqrt(d * scale); });
```
D3 Bar Chart Tutorial

https://bost.ocks.org/mike/bar/

```javascript
var data = [4, 8, 15, 16, 23, 42];
```

Selecting an Element

**Javascript:**

```javascript
var div = document.createElement("div");
div.innerHTML = "Hello, world!";
document.body.appendChild(div);
```

**D3:**

```javascript
var body = d3.select("body");
var div = body.append("div");
div.html("Hello, world!");
```
Chaining Methods

```js
var body = d3.select("body");
body.style("color", "black");
body.style("background-color", "white");

d3.select("body")
 .style("color", "black")
 .style("background-color", "white");

var section = d3.selectAll("section");

section.append("div")
 .html("First!");

section.append("div")
 .html("Second.");
```
D3 Bar Chart Tutorial

Coding a Chart, Manually

```html
<!DOCTYPE html>
<style>
.chart div {
  font: 10px sans-serif;
  background-color: steelblue;
  text-align: right;
  padding: 3px;
  margin: 1px;
  color: white;
}
</style>
<div class="chart">
  <div style="width: 40px;">4</div>
  <div style="width: 80px;">8</div>
  <div style="width: 150px;">15</div>
  <div style="width: 160px;">16</div>
  <div style="width: 230px;">23</div>
  <div style="width: 420px;">42</div>
</div>
```
D3 Bar Chart Tutorial

Coding a Chart, Automatically

```javascript
d3.select(".chart")
    .selectAll("div")
    .data(data)
    .enter().append("div")
    .style("width", function(d) { return d * 10 + "px"; })
    .text(function(d) { return d; });
```

First, we select the chart container using a class selector.

```javascript
var chart = d3.select(".chart");
```

Next we initiate the data join by defining the selection to which we will join data.

```javascript
var bar = chart.selectAll("div");
```
D3 Bar Chart Tutorial

Coding a Chart, Automatically

```javascript
var barUpdate = bar.data(data);

var barEnter = barUpdate.enter().append("div");

barEnter.style("width", function(d) { return d * 10 + "px"; });

barEnter.text(function(d) { return d; });
```
Scaling to Fit

```javascript
var x = d3.scale.linear()
    .domain([0, d3.max(data)])
    .range([0, 420]);
```
D3 Bar Chart Tutorial

```html
<!DOCTYPE html>
<style>
  .chart rect {
    fill: steelblue;
  }

  .chart text {
    fill: white;
    font: 10px sans-serif;
    text-anchor: end;
  }
</style>

<svg class="chart" width="420" height="120">
  <g transform="translate(0,0)"
      transform-origin="0 0">
    <rect width="40" height="19"></rect>
    <text x="37" y="9.5" dy=".35em">4</text>
  </g>
  <g transform="translate(0,20)"
      transform-origin="0 0">
    <rect width="80" height="19"></rect>
    <text x="77" y="9.5" dy=".35em">8</text>
  </g>
  <g transform="translate(0,40)"
      transform-origin="0 0">
    <rect width="150" height="19"></rect>
    <text x="147" y="9.5" dy=".35em">15</text>
  </g>
  <g transform="translate(0,60)"
      transform-origin="0 0">
    <rect width="160" height="19"></rect>
    <text x="157" y="9.5" dy=".35em">16</text>
  </g>
  <g transform="translate(0,80)"
      transform-origin="0 0">
    <rect width="230" height="19"></rect>
    <text x="227" y="9.5" dy=".35em">23</text>
  </g>
  <g transform="translate(0,100)"
      transform-origin="0 0">
    <rect width="420" height="19"></rect>
    <text x="417" y="9.5" dy=".35em">42</text>
  </g>
</svg>

Full code to do it manually

![Graph showing bar chart with values 4, 8, 15, 16, 23, 42]
Full code to do it automatically

```html
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.chart rect {
  fill: steelblue;
}
.chart text {
  fill: white;
  font: 10px sans-serif;
  text-anchor: end;
}
</style>
<svg class="chart"></svg>
<script src="/d3js.org/d3.v3.min.js" charset="utf-8"></script>
```
D3 Bar Chart Tutorial

Full code to do it automatically

```javascript
<script>
var data = [4, 8, 15, 16, 23, 42];

var width = 420,
    barHeight = 20;

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

var chart = d3.select(".chart")
    .attr("width", width)
    .attr("height", barHeight * data.length);

var bar = chart.selectAll("g")
    .data(data)
    .enter().append("g")
    .attr("transform", function(d, i) { return "translate(0," + i * barHeight + ")"; });

bar.append("rect")
    .attr("width", x)
    .attr("height", barHeight - 1);

bar.append("text")
    .attr("x", function(d) { return x(d) - 3; })
    .attr("y", barHeight / 2)
    .attr("dy", ".35em")
    .text(function(d) { return d; });
</script>
```
Load data

```javascript
// 1. Code here runs first, before the download starts.
d3.tsv("data.tsv", function(error, data) {
    // 3. Code here runs last, after the download finishes.
});

// 2. Code here runs second, while the file is downloading.
```

The equivalent of Javascript code:
```
var data = [
    {name: "Locke", value: 4},
    {name: "Reyes", value: 8},
    {name: "Ford", value: 15},
    {name: "Jarrah", value: 16},
    {name: "Shephard", value: 23},
    {name: "Kwon", value: 42}
];
```
<!DOCTYPE html>
<meta charset="utf-8">
<style>
.chart rect {
    fill: steelblue;
}

.chart text {
    fill: white;
    font: 10px sans-serif;
    text-anchor: end;
}
</style>
<svg class="chart"></svg>
<script src="//d3js.org/d3.v3.min.js" charset="utf-8"></script>
D3 Bar Chart Tutorial

```javascript
<script>
    var width = 420,
        barHeight = 20;

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

    var chart = d3.select(".chart")
        .attr("width", width);

    d3.tsv("data.tsv", type, function(error, data) {
        x.domain([0, d3.max(data, function(d) { return d.value; })]);

        chart.attr("height", barHeight * data.length);

        var bar = chart.selectAll("g")
            .data(data)
            .enter().append("g")
            .attr("transform", function(d, i) { return "translate(0, " + i * barHeight + ")"; });

        bar.append("rect")
            .attr("width", function(d) { return x(d.value); })
            .attr("height", barHeight - 1);

        bar.append("text")
            .attr("x", function(d) { return x(d.value) - 3; })
            .attr("y", barHeight / 2)
            .attr("dy", ".35em")
            .text(function(d) { return d.value; });

    function type(d) {
        d.value = +d.value; // coerce to number
        return d;
    }
</script>
```
D3 Example Circles

https://bost.ocks.org/mike/circles/

```html
<svg width="720" height="120">
  <circle cx="40" cy="60" r="10"></circle>
  <circle cx="80" cy="60" r="10"></circle>
  <circle cx="120" cy="60" r="10"></circle>
</svg>
```

```javascript
var circle = d3.selectAll("circle");

circle.style("fill", "steelblue");
circle.attr("r", 30);
```
D3 Example Circles

https://bost.ocks.org/mike/circles/

```html
<svg width="720" height="120">
    <circle cx="40" cy="60" r="10"></circle>
    <circle cx="80" cy="60" r="10"></circle>
    <circle cx="120" cy="60" r="10"></circle>
</svg>
```

```javascript
var circle = d3.selectAll("circle");
circle.style("fill", "steelblue");
circle.attr("r", 30);

circle.attr("cx", function() { return Math.random() * 720; });
```
D3 Gallery

Box Plots

Bubble Chart

Bullet Charts

Calendar View

Non-contiguous Cartogram

Chord Diagram

Dendrogram

Force-Directed Graph

Circle Packing

Population Pyramid 2000

Stacked Bars

Streamgraph
D3 Gallery

<table>
<thead>
<tr>
<th>Sunburst</th>
<th>Node-Link Tree</th>
<th>Treemap</th>
<th>Voronoi Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Sunburst" /></td>
<td><img src="image" alt="Node-Link Tree" /></td>
<td><img src="image" alt="Treemap" /></td>
<td><img src="image" alt="Voronoi Diagram" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hierarchical Edge Bundling</th>
<th>Voronoi Diagram</th>
<th>Symbol Map</th>
<th>Parallel Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Hierarchical Edge Bundling" /></td>
<td><img src="image" alt="Voronoi Diagram" /></td>
<td><img src="image" alt="Symbol Map" /></td>
<td><img src="image" alt="Parallel Coordinates" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scatterplot Matrix</th>
<th>Zoomable Pack Layout</th>
<th>Hierarchical Bars</th>
<th>Epicyclical Gears</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Scatterplot Matrix" /></td>
<td><img src="image" alt="Zoomable Pack Layout" /></td>
<td><img src="image" alt="Hierarchical Bars" /></td>
<td><img src="image" alt="Epicyclical Gears" /></td>
</tr>
</tbody>
</table>
D3 Gallery

Collision Detection

Collapsible Force Layout

Force-Directed States

Azimuthal Projections

Choropleth

Collapsible Tree Layout

Zoomable Treemap

Zoomable Partition Layout

Zoomable Area Chart

Drag and Drop Collapsible Tree Layout

Radial Cluster Layout

Sankey Diagram

© 2018 CY Lin, Columbia University
### D3 Gallery

<table>
<thead>
<tr>
<th>Fisheye Distortion</th>
<th>Hive Plot</th>
<th>Co-occurrence Matrix</th>
<th>Motion Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Fisheye Distortion" /></td>
<td><img src="image2.png" alt="Hive Plot" /></td>
<td><img src="image3.png" alt="Co-occurrence Matrix" /></td>
<td><img src="image4.png" alt="Motion Chart" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chord Diagram</th>
<th>Animated Béziers</th>
<th>Zoomable Sunburst</th>
<th>Collatz Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5.png" alt="Chord Diagram" /></td>
<td><img src="image6.png" alt="Animated Béziers" /></td>
<td><img src="image7.png" alt="Zoomable Sunburst" /></td>
<td><img src="image8.png" alt="Collatz Graph" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parallel Sets</th>
<th>Word Cloud</th>
<th>Obama's Budget Proposal</th>
<th>Facebook IPO</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image9.png" alt="Parallel Sets" /></td>
<td><img src="image10.png" alt="Word Cloud" /></td>
<td><img src="image11.png" alt="Obama's Budget Proposal" /></td>
<td><img src="image12.png" alt="Facebook IPO" /></td>
</tr>
</tbody>
</table>
D3 Gallery

US Trade Deficit

Sequences sunburst

NFL Predictions

Koalas to the Max

Sankey Creator

Convert any page into bubbles

D3 Builder