D3 - Data Driven Documents

Presented by:

Advait Ramesh Iyer

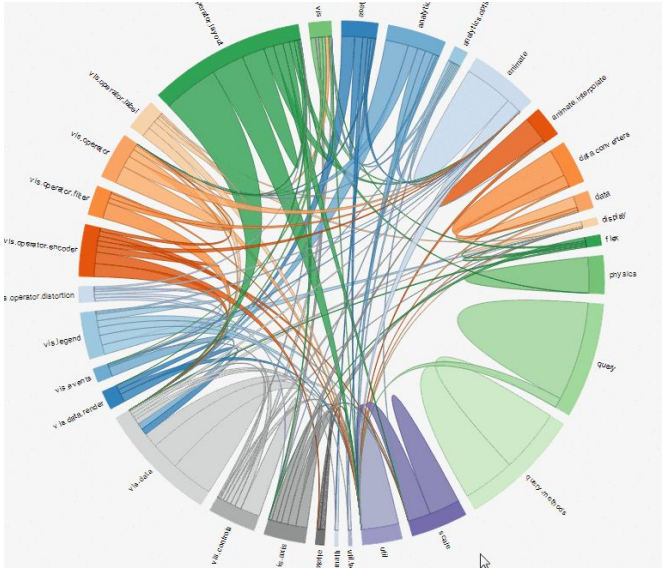
Information Visualization (IST 719) M001

What is D3?

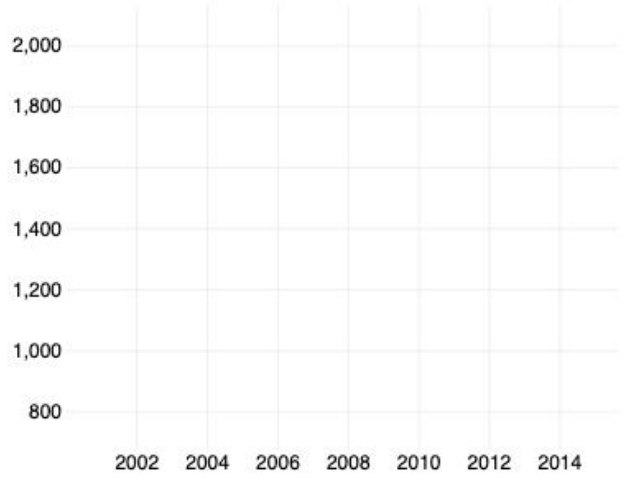
- D3 is a JavaScript library used for creating interactive visualizations on web-pages
- D3 adds graphics to SVG (Scalable Vector Graphic) element in HTML, and binds data to its elements dynamically
- It is important to note that a lot of CSS code that is used for regular stylizing of HTML web-pages do not work the same for SVG graphics

Use cases

D3 can be used to develop really creative visualizations, wherein high-dimensional data requires interactivity for an easy reader-understanding

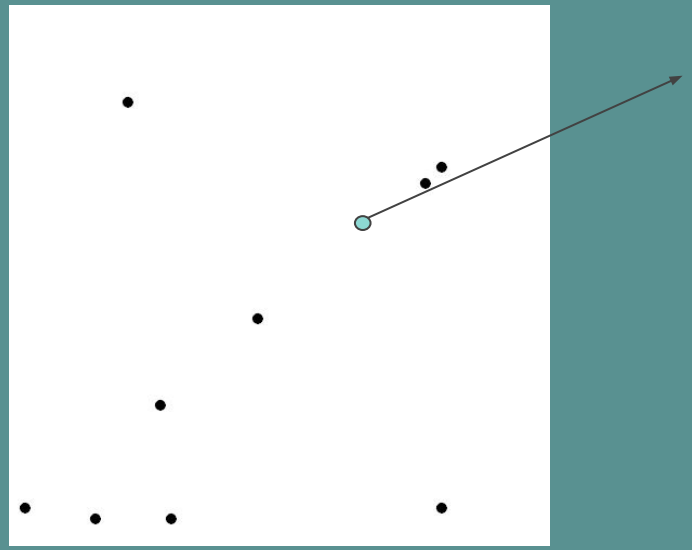


Interactive Chord diagram



Interactive Time-series

Let’s create an interactive Voronoi diagram!



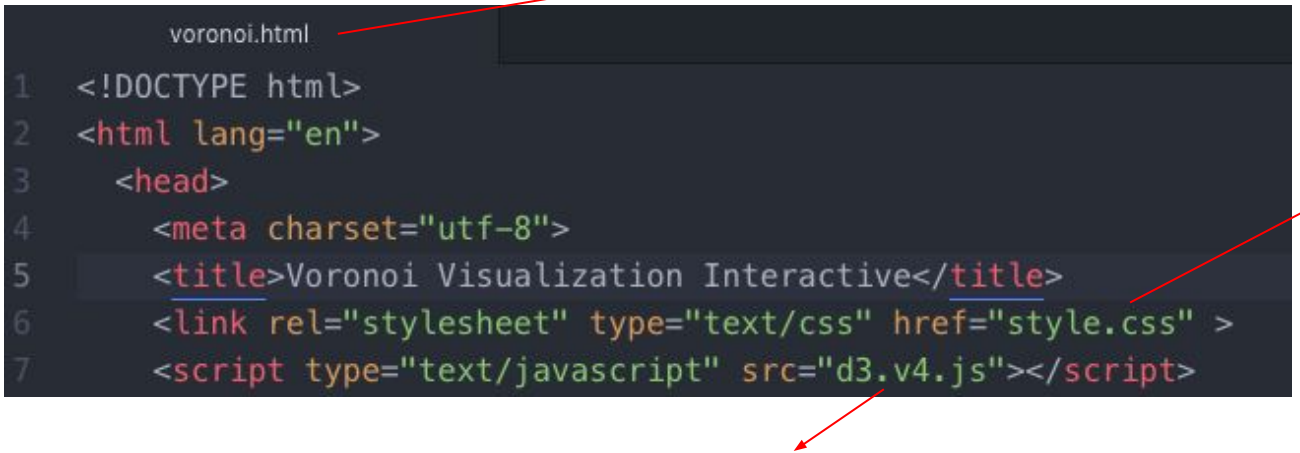
If any random point is drawn, the voronoi diagram (in this case based on Euclidean distance), tells us the respective closest data-point

Use Case:

- Classification Visualizations
- Proximity Analysis

Getting started

01



NOTE: It is important to import the D3 library in the head tag itself, else the visualizations will not be readable by the browser

Defining styles in the <head>

02

- We are considering analyzing proximity from fuel stations
- We define the polygon, and the fuel group, which are the elements of our Voronoi diagram
- “polygon” defines the space closest to each fuel station, and “fuel” represents the centroid points which represent each fuel station

```
    <style>
    g.polygons path {
      fill: white;
      stroke: lightsteelblue;
    }

    g.fuel circle {
      fill: steelblue;
    }
    </style>
</head>
```

Setting up the Voronoi
03

Random number generator function

```
var vertices = d3.range(100)
                 .map(function(d){ return [Math.random()*width,Math.random()*height]; });

var voronoi = d3.voronoi()
                .size([width,height]);
var svg = d3.select("body").append("svg")
            .attr("width","100%")
            .attr("height","100%");
```

SVG Space Allocation on <body>

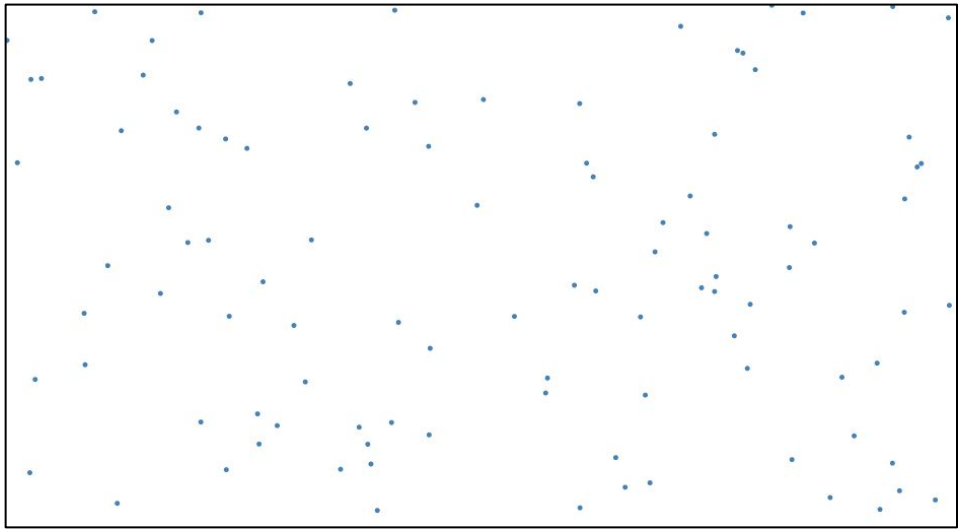
Voronoi element

Creating Centroids
04

Defining the class: fuel

```
svg.append("g").attr("class","fuel")
.selectAll("circle")
  .data(vertices)
  .enter().append("circle")
          .attr("cx",function(d){ return d[0]; })
          .attr("cy",function(d){ return d[1]; })
          .attr("r","2.5");
```

Defining coordinates x, y and radius r of the circle



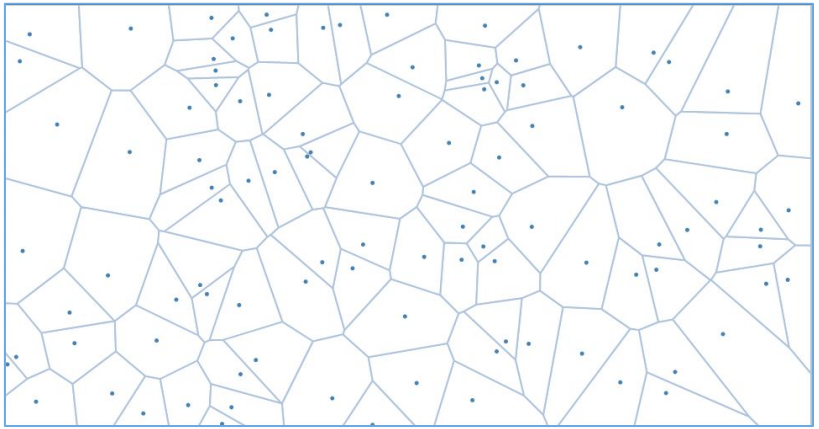
Output of HTML web-page containing circles across the SVG space

Creating Polygons
05

Defining the class polygons

```
svg.append("g").attr("class","polygons")
.selectAll("path")
  .data(voronoi.polygons(vertices))
  .enter().append("path")
          .attr("d",function(d){ return "M"+d.join("L")+"Z"; });
```

Defining the polygonic contours for the voronoi diagram



Output of HTML web-page containing circles across the SVG space

Let us assign interactivity to our Voronoi diagram!

Resizing our diagram by window-width

06

We call the function resize from JavaScript library

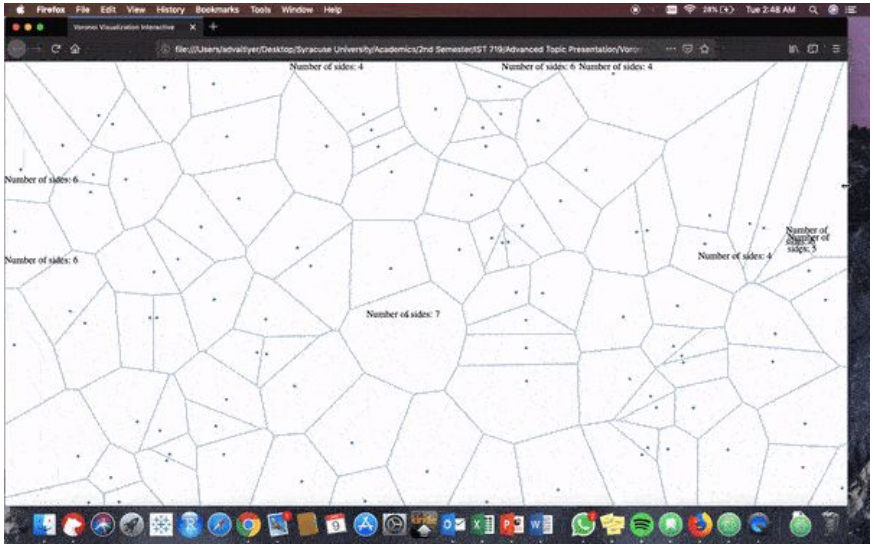
```
d3.select(window).on("resize",callFunction);
callFunction();
function callFunction() {

function dragged() {
  d3.select(this).attr("transform","translate("+d3.event.x+","+d3.event.y+")");
}

      var tooltip = d3.select("body")
                    .append("div").style("opacity","0").style("position","absolute");

      var svgtest = d3.select("body").select("svg");
      if (!svgtest.empty()) {
        svgtest.remove();
      }

      var width = window.innerWidth||document.documentElement.clientWidth||document.body.clientWidth;
      var height = window.innerHeight;
```

Defining it for dragging of window screen

Defining behavior of the mouse tip

Using svgtest variable to remove white-space after resizing our diagram

Defining our SVG space for all types of devices using or statement



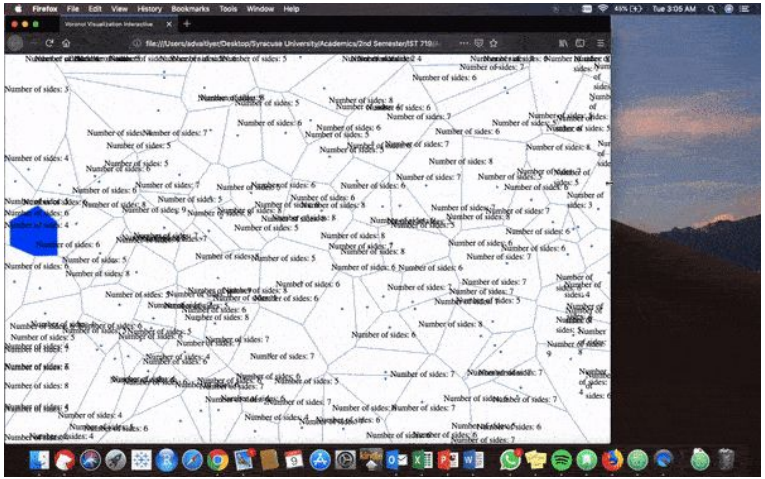
Animating Polygons
07

Selecting the 31st child

```
      d3.select("g.polygons").select("path:nth-child(30)")
        .transition().duration(1000)
        .style("fill","blue")
        .attr("transform","translate(10,10)");


      d3.select("g.polygons").select("path:nth-child(30)").dispatch("mousemove");
}
    </script>
  </body>

</html>
```

Setting duration and color

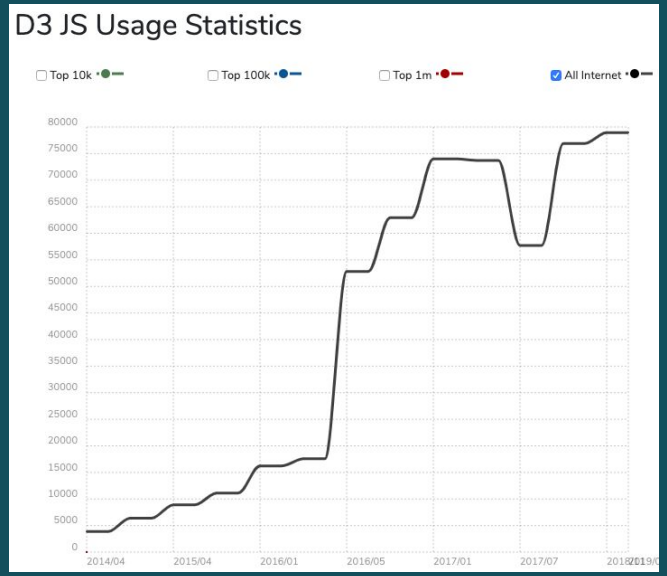
Dispatching the child with movement



Comparison with R

- D3 is based on JavaScript, whereas R is a programming language itself
- D3 is not a visualization library
- D3 provides the option of developing interactive graphics. Some of the newer packages in R have this kind of capability
- We cannot import D3 visualizations, as they are created on the web-page

D3: Market Dynamics and Future Scope



The growth rate for D3 has been very high since mid-2016

Top Competitors

Chart.js

Raphael.js

Highcharts

JQuery UI Tabs

References

- https://lynda.com
- https://wikipedia.org
- https://bl.ocks.org
- https://trends.builtwith.com

Presented by:

Advait Ramesh Iyer

Information Visualization (IST 719) M001

THANK YOU!