CSE 564
Visualization & Visual Analytics

Introduction to D3

Klaus Mueller
Computer Science Department
Stony Brook University
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intro, schedule, and logistics</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Applications of visual analytics</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Basic tasks, data types</td>
<td>Project #1 out</td>
</tr>
<tr>
<td>4</td>
<td>Data assimilation and preparation</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Introduction to D3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Bias in visualization</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Data reduction and dimension reduction</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Visual perception</td>
<td>Project #2(a) out</td>
</tr>
<tr>
<td>9</td>
<td>Visual cognition</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Visual design and aesthetics</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Cluster analysis: numerical data</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Cluster analysis: categorical data</td>
<td>Project #2(b) out</td>
</tr>
<tr>
<td>13</td>
<td>High-dimensional data visualization</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Dimensionality reduction and embedding methods</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Principles of interaction</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Midterm #1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Visual analytics</td>
<td>Final project proposal call out</td>
</tr>
<tr>
<td>18</td>
<td>The visual sense making process</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Maps</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Visualization of hierarchies</td>
<td>Final project proposal due</td>
</tr>
<tr>
<td>21</td>
<td>Visualization of time-varying and time-series data</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Foundations of scientific and medical visualization</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Volume rendering</td>
<td>Project 3 out</td>
</tr>
<tr>
<td>24</td>
<td>Scientific and medical visualization</td>
<td>Final Project preliminary report due</td>
</tr>
<tr>
<td>25</td>
<td>Visual analytics system design and evaluation</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Memorable visualization and embellishments</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Infographics design</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Midterm #2</td>
<td></td>
</tr>
</tbody>
</table>
The material presented in these slides is derived from this book:

Also available online
What is D3.js?

D3 = Data Driven Documents

JavaScript library for manipulating documents based on data
- frequent tool to support data journalism (New York Times)

D3 helps you bring data to life using HTML, SVG, and CSS
- great library to construct animated visualizations (D3 website)

Runs in any modern web browser (Chrome, Firefox, IE)
- no need to download any software
- independent of OS (Linux, Windows Mac)
Makes Use Of

HTML  Hypertext Markup Language
CSS   Cascading Style Sheets
JS    JavaScript
DOM   The Document Object Model
  - tree structured organization of HTML objects
SVG   Scalable Vector Graphics
WHAT YOU NEED

A text editor
- Visual; Studio Code, Atom, sublime text 2, or your browser
- need an editor with syntax highlighting. else it’s easy to get lost

The d3 library
- from http://d3js.org

Data files for your code

A web server
- use python -m http.server 8000

A browser
- to run the code
Your folder structure should look like this:

project-folder/
  d3/
    d3.v3.js // D3 library
    d3.v3.min.js (optional) // minified D3 library
  index.html
Your initial webpage (index.html) should look like this:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>D3 Page Template</title>
    <script type="text/javascript" src="d3/d3.v3.js"></script>
  </head>
  <body>
    <script type="text/javascript">
      // Your beautiful D3 code will go here
    </script>
  </body>
</html>
```
**MAMP = My Apache, MySQL, PHP**
- really only need Apache for now
- MS Windows = WampServer and XAMPP for Windows
- Mac = MAMP or XAMPP for Mac

**Procedure**
- install package (Linux has it already installed)
- find webserver folder (only files residing there will be served)
- put project files there
Let’s Use Some Simple Data

var dataset = [ 5, 10, 15, 20, 25 ];
Consider the following js code ... all methods are chained:

d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text("New paragraph!");

which gives this output
  - how did this happen?
Consider the following js code ... all methods are chained:

d3.select("body").selectAll("p") // selects all paragraphs in the DOM (none so far ...)
.data(dataset) // counts and parses the data values
.enter() // creates new, data-bound elements (placeholders) for the data values
.append("p") // takes the empty placeholder and adds a p-element
.text("New paragraph!"); // takes the p-element and inserts a text value

which gives this output

- how did this happen?
Change the last line to:

d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return d; });

which gives this output
  ▪  how did this happen?
Change the last line to:

```javascript
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return d; }); // used the data to populate the contents of each paragraph of the data-driven document
```

which gives this output

- how did this happen?
Using Functions

Change the last line to:

d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return "I can count up to " + d; });

which gives this output
  - how did this happen?
Change the last line to:

```javascript
  d3.select("body").selectAll("p")
    .data(dataset)
    .enter()
    .append("p")
    .text(function(d) { return "I can count up to " + d; })
    .style("color", "red");
```

which gives this output
- how did this happen?
More Complex Functions

Replace the last line with:

```javascript
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text(function(d) { return "I can count up to " + d; })
  .style("color", function(d) { if (d > 15) {return "red"; } else { return "black"; } });
```

which gives this output

- how did this happen?
Let’s draw some bar charts

For this, put this embedded style in the document head

div.bar {
    display: inline-block;
    width: 20px;
    height: 75px; /* We'll override height later */
    background-color: teal;
}

Run this code:

```javascript
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
  .data(dataset)
  .enter()
  .append("div")
  .attr("class", "bar");
```

which gives this output
- five bars with no space between them
- how did this happen?
Run this code:

```javascript
var dataset = [5, 10, 15, 20, 25];

d3.select("body").selectAll("div")
  .data(dataset)
  .enter()
  .append("div")
  .attr("class", "bar")
  .style("height", function(d) { return d + "px"; });
```

which gives this output
- how did this happen?
Run this code:

```javascript
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
  .data(dataset)
  .enter()
  .append("div")
  .attr("class", "bar")
  .style("height", function(d) { return d + "px"; }) // adds text "px" to specify that the units are pixels → heights are 5px, 10px, 15px, 20px, and 25px
```

which gives this output

- how did this happen?
Run this code: (also add margin-right: 2px; to the css style)

```javascript
var dataset = [ 5, 10, 15, 20, 25 ];

d3.select("body").selectAll("div")
    .data(dataset)
    .enter()
    .append("div")
    .attr("class", "bar")
    .style("height", function(d) { var barHeight = d * 5; return barHeight + "px"; });
```

which gives this output
- how did this happen?
Optionally define some variable beforehand, e.g.:

```javascript
// width and height
var w = 500;
var h = 50;
```

Define the svg object:

```javascript
var svg = d3.select("body")
  .append("svg")
  .attr("width", w)
  .attr("height", h);
```
Define the circles as variables for ease of reference:

```javascript
var circles = svg.selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle");
```

But could so this just as well:

```javascript
drawCircles = svg.selectAll("circle")
  .data(dataset)
  .enter()
  .append("circle"); // now circles are appended to the end of the SVG element
```
Now Draw The Circles

Run this code (still using var dataset = [ 5, 10, 15, 20, 25 ];)

circles.attr("cx", function(d, i) {return (i * 50) + 25;})
  .attr("cy", h/2)
  .attr("r", function(d) {return d;});

or append it to the .append("circle") method

This gives this output
  - how did this happen?
Run this code (still using var dataset = [ 5, 10, 15, 20, 25 ];)

circles.attr("cx", function(d, i) {return (i * 50) + 25;})  // i increments by 1 each time, starting at 0
    .attr("cy", h/2)
    .attr("r", function(d) {return d;});

or append it to the .append("circle") method

This gives this output
  - how did this happen?
Run this code (still using var dataset = [ 5, 10, 15, 20, 25 ];)

circles.attr("cx", function(d, i) {return (i * 50) + 25;})
 .attr("cy", h/2)
 .attr("r", function(d) {return d;})
 .attr("fill", "yellow")
 .attr("stroke", "orange")
 .attr("stroke-width", function(d) {return d/2;});

This gives this output
• how did this happen?
BAR CHARTS

Code
This will update the bar chart on a **mouse click**:

```javascript
// New values for dataset
dataset = [ 11, 12, 15, 20, 18, 17, 16, 18, 23, 25, 5, 10, 13, 19, 21, 25, 22, 18, 15, 13 ];

// Update all rects
svg.selectAll("rect")
  .data(dataset)
  .attr("y", function(d) {
    return h - yScale(d);
  })
  .attr("height", function(d) {
    return yScale(d);
  });
```

```
Smooth animations are desirable:

```javascript
svg.selectAll("rect")
  .data(dataset)
  .transition()
  .attr("y", function(d) {
    return h - yScale(d);
  })
  .attr("height", function(d) {
    return yScale(d);
  })
  .attr("fill", function(d) {
    return "rgb(0, 0, " + (d * 10) + ")";
  });
```
Now run **this code:**

```javascript
svg.selectAll("rect")
    .data(dataset)
    .transition()
    .duration(1000) // <-- Now this is new!
    .attr("y", function(d) {
        return h - yScale(d);
    })
    .attr("height", function(d) {
        return yScale(d);
    })
    .attr("fill", function(d) {
        return "rgb(0, 0, " + (d * 10) + ")";
    });
```
Facilitated by event handlers (listeners), e.g.:

d3.select("p")
    .on("click", function() {
        //Do something on click
    });

others react on
  - mouse hovering
  - mouse over
  - mouse out
  - and others

Example
Assume you selected a certain item by mouseover

.on("mouseover", function() {
    //Do something on mouseover of any bar
});

Keyword “this” maps the action to the selected item

.on("mouseover", function() {
    d3.select(this)
        .attr("fill", "orange");
});
D3 layouts take data that you provide

- remap or otherwise transform it
- and so generating new data that is more convenient for a specific visual task

The supported layouts are:

- Bundle and Chord
- Cluster
- Force
- Histogram
- Pack, Partition, and Pie
- Stack
- Tree and Treemap
```javascript
var dataset = {
  nodes: [
    { name: "Adam" },
    { name: "Bob" },
    { name: "Carrie" },
    { name: "Donovan" },
    { name: "Edward" },
    { name: "Felicity" },
    { name: "George" },
    { name: "Hannah" },
    { name: "Iris" },
    { name: "Jerry" }
  ],
  edges: [
    { source: 0, target: 1 },
    { source: 0, target: 2 },
    { source: 0, target: 3 },
    { source: 0, target: 4 },
    { source: 1, target: 5 },
    { source: 2, target: 5 },
    { source: 2, target: 5 },
    { source: 3, target: 4 },
    { source: 5, target: 8 },
    { source: 5, target: 9 },
    { source: 6, target: 7 },
    { source: 7, target: 8 },
    { source: 8, target: 9 }
  ]
};
```
```javascript
var force = d3.layout.force()
  .nodes(dataset.nodes)
  .links(dataset.edges)
  .size([w, h])
  .linkDistance([50]) // <-- New!
  .charge([-100]) // <-- New!
  .start();

Next, we create an SVG line for each edge:

```javascript
var edges = svg.selectAll("line")
  .data(dataset.edges)
  .enter()
  .append("line")
  .style("stroke", 
  .style("stroke-width", 1);
```