

CSE 564
VISUALIZATION & VISUAL ANALYTICS

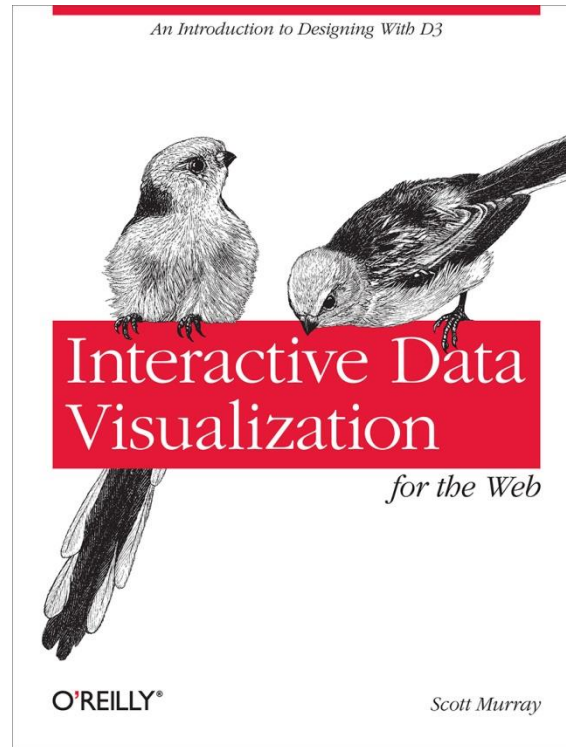
INTRODUCTION TO D3

KLAUS MUELLER

COMPUTER SCIENCE DEPARTMENT
STONY BROOK UNIVERSITY

Lecture	Topic	Projects
1	Intro, schedule, and logistics	
2	Applications of visual analytics, basic tasks, data types	
3	Introduction to D3, basic vis techniques for non-spatial data	Project #1 out
4	Visual perception and cognition	
5	Visual design and aesthetics	
6	Data types, notion of similarity and distance	
7	Data preparation and reduction	Project #1 due
8	Introduction to R, statistics foundations	Project #2 out
9	Data mining techniques: clusters, text, patterns, classifiers	
10	Data mining techniques: clusters, text, patterns, classifiers	
11	Computer graphics and volume rendering	
12	Techniques to visualize spatial (3D) data	Project #2 due
13	Scientific and medical visualization	Project #3 out
14	Scientific and medical visualization	
15	Midterm #1	
16	High-dimensional data, dimensionality reduction	Project #3 due
17	Big data: data reduction, summarization	
18	Correlation and causal modeling	
19	Principles of interaction	
20	Visual analytics and the visual sense making process	Final project proposal due
21	Evaluation and user studies	
22	Visualization of time-varying and time-series data	
23	Visualization of streaming data	
24	Visualization of graph data	Final Project preliminary report due
25	Visualization of text data	
26	Midterm #2	
27	Data journalism	
	Final project presentations	Final Project slides and final report due

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

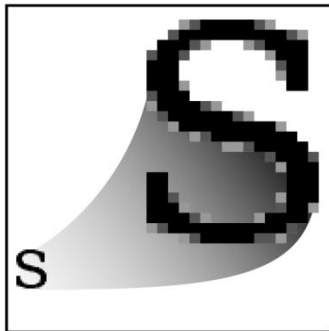
CSS Cascading Style Sheets

JS JavaScript

DOM The Document Object Model

- tree structured organization of HTML objects

SVG Scalable Vector Graphics



Raster

.jpeg .gif .png



Vector

.svg

WHAT YOU NEED

A text editor

- textMate, eclipse/aptana, 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 (recommended)

- if your visualization is reading data from files or a database (XMLHttpRequest)
- many options: EasyPHP (windows), Mac OS X Server, MAMP, Python
- else need to specify the data in the code

A browser

- to run the code

SETUP

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
```

SETUP

Your initial webpage (index.html) should look like this:

```
<!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>
```


SETTING UP A WEBSERVER

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
- open browser and point to <http://localhost/> or <http://localhost:8888/> or <http://localhost:8888/project-folder/>

LET'S USE SOME SIMPLE DATA

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

CHAINING

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?

New paragraph!

New paragraph!

New paragraph!

New paragraph!

New paragraph!

CHAINING

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?

New paragraph!

New paragraph!

New paragraph!

New paragraph!

New paragraph!

USING THE DATA

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?

5
10
15
20
25

USING THE DATA

Change the last line to:

```
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?

5
10
15
20
25

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?

I can count up to 5

I can count up to 10

I can count up to 15

I can count up to 20

I can count up to 25

ADDING AESTHETICS

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; })  
  .style("color", "red");
```

which gives this output

- how did this happen?

I can count up to 5

I can count up to 10

I can count up to 15

I can count up to 20

I can count up to 25

MORE COMPLEX FUNCTIONS

Replace the last line with:

```
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?

I can count up to 5

I can count up to 10

I can count up to 15

I can count up to 20

I can count up to 25

DRAWING WITH DATA

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;  
}
```

SIMPLE BAR CHART

Run this code:

```
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?



MORE COMPLEX BAR CHART

Run this code:

```
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?



MORE COMPLEX BAR CHART

Run this code:

```
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?



MAKE BARS TALLER AND ADD SPACES

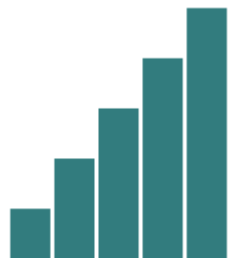
Run this code: (also add `margin-right: 2px;` to the css style)

```
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?



GET READY TO DRAW SOME SVGs

Optionally define some variable beforehand, e.g.:

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

Define the svg object:

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

GET READY TO DRAW SOME SVG CIRCLES

Define the circles as variables for ease of reference:

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

But could so this just as well:

```
svg.selectAll("circle")  
    .data(dataset)  
    .enter()  
    .append("circle"); // now circles are appended to the end of the SVG element
```


NOW DRAW THE CIRCLES

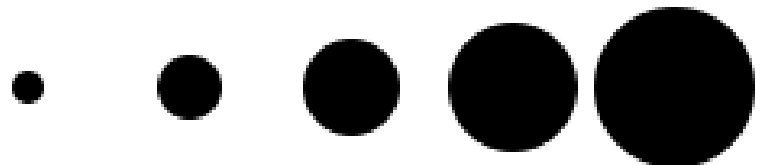
Run this code (still using `var dataset = [5, 10, 15, 20, 25];`)

```
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?



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;}) // i increments by 1 each time,  
                                                         starting at 0
```

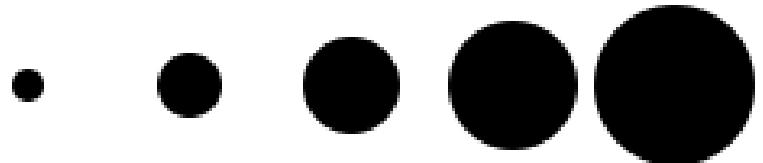
```
.attr("cy", h/2)
```

```
.attr("r", function(d) {return d;});
```

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

This gives this output

- how did this happen?



ADDING COLORS

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

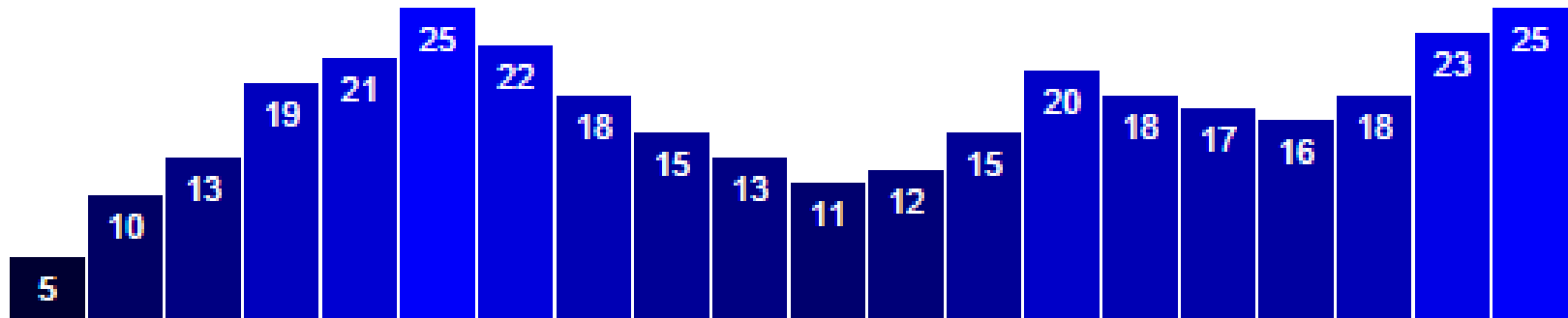
```
circles.attr("cx", function(d, i) {return (i * 50) + 25;})  
  .attr("cy", h/2)  
  .attr("r", function(d) {return d;})  
  .attr("fill", "yellow")  
  .attr("stroke", "orange")  
  .attr("stroke-width", function(d) {return d/2;});
```

This gives this output

- how did this happen?



BAR CHARTS



[Code](#)

UPDATES

This will update the bar chart on a [mouse click](#):

```
d3.select("p")
  .on("click", function() {

    //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);
      });

  });
```

ADDING ANIMATED TRANSITIONS

Smooth animations are desirable:

```
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) + ")";
  });
```

CONTROL DURATION

Now run [this code](#):

```
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) + ")";
  });
```

INTERACTION

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](#)

DIRECTING ACTION TO A SPECIFIC ITEM

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");  
});
```

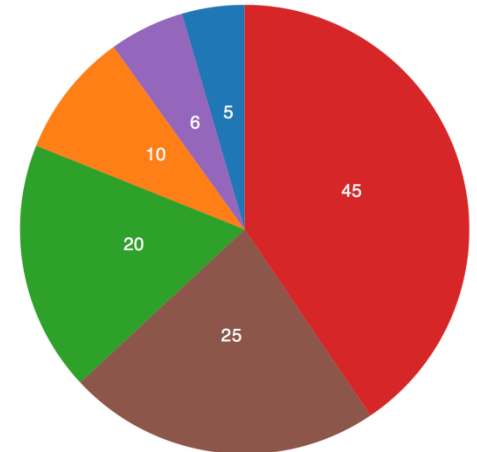
LAYOUTS

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



FORCE-DIRECTED LAYOUT

```
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 }
  ]
};
```

```
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:

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

Note that I set all the lines to have the same stroke color and weight, but of course you could set this dynamically based on data (say, thicker or darker lines for “stronger” connections, or some other value).

[Demo](#)

Then, we create an SVG circle for each node:

```
var nodes = svg.selectAll("circle")
    .data(dataset.nodes)
    .enter()
    .append("circle")
    .attr("r", 10)
    .style("fill", function(d, i) {
        return colors(i);
    })
    .call(force.drag);
```

MINI PROJECT #1

Get some CSV-based data from [here](#) or [here](#) or elsewhere

- if the data is too big you can down-sample it

Your D3-based visual interface should be able to (all 10 pts):

1. pick a variable and bin it into a fixed range (equi-width) of your choice
2. create a bar chart of the variable you picked in 1.
3. using a menu, allow users to select a new variable and update chart
4. only on mouse-over display the value of the bar on top of the bar
5. on mouse-over make the bar wider and higher to focus on it
6. on mouse-click transform the bar chart into a pie chart (and back)
7. mouse moves left (right) should decrease (increase) bin width/size

An additional 10 pts for elegant implementation/function

Submission on SBU HotCRP (see email)

Due Tuesday, Feb. 14, 2016

Extra credit (10 pts):

- on mouse-click create a force-directed layout using a chosen distance

DELIVERABLES

You need to upload the following by the due date

- 2-3 page report with illustrated description of your program's capabilities and implementation detail
- add code snippets to show how you did things
- mpeg or avi video file that shows all features of your software in action
- **important:** do not reveal your identity on the submission itself (to enable double-blind peer feedback)

Grading

- TA will pick students at random for thorough code review sessions
- you better know your code !!!
- so, please do not just copy code beyond the D3 templates
- or even worse, videotape someone else's program

SEEKING OUTSIDE HELP

Aka, cheating

Discussion with your class mates (but not others) is OK

Cut and paste from any source is not OK

- any suspected activity of this kind will result in zero points
- also for the person providing the original
- two-strikes and out rule is in effect (including an academic misconduct report)
- this includes any feeble attempt to cover the tracks somehow

Stay honest and resist the temptation!