Data-Driven Documents

MassMutual DSDP 2017:

INTRODUCTION TO D3

June 12, 2017

R. Jordan Crouser & Amelia McNamara
Statistical & Data Sciences
Smith College

D3.js

- JavaScript library designed to make web-based visualization easier
- Data *transformation*, rather than new representation
- Visualization problems reduced to constructing and updating a DOM
- DOM responds to changes in underlying data (stream or interaction)

Much of what follows comes from Mike Bostock's VIZBI 2012 workshop

Why use D3?

- Uses existing web standards (HTML and SVG)
  - SVG visual primitives: `rect`, `circle`, `path`, `etc.`
  - Low point of entry
  - Future-proofing
- Compatibility with existing tools (like CSS and debuggers)
  - Use what you already know!
  - Use the browser's JavaScript console to debug interactively

Selections

- D3's atomic operand (much like jQuery)

```
d3.selectAll("circle")    // return all elements with
                          // the circle tag
```

- Can select by any facet:

```
#foo          // <any id="foo">
 foo          // <foo>
.foo          // <any class="foo">
[foo=bar]     // <any foo="bar">
 foo bar      // <foo><bar></foo>
```

- Result is an array of elements that match description

Selection example

```
// Select all <circle> elements
var circle = d3.selectAll("circle");

// Set some attributes and styles
circle.attr("cx", 20);          // Center x value = 20
circle.attr("cy", 12);          // Center y value = 12
circle.attr("r", 24);           // radius = 24px
circle.style("fill", "red");
```

Selection example

Method chaining allows shorter (and more readable) code

```
// Select all <circle> elements
// and set some attributes and styles

d3.selectAll("circle")
      .attr("cx", 20)
      .attr("cy", 12)
      .attr("r", 24)
      .style("fill", "red");
```

Selection.append

- To build dynamic visualizations, we need to be able to add new elements as well

```
var h1 = d3.selectAll("section")
      .style("background", "steelblue")
      .append("h1")
      .text("Hello!");
```

- Use caution with method chaining: `append` returns a new selection!

Mapping data to elements

- Data are arrays
- Could be numbers:

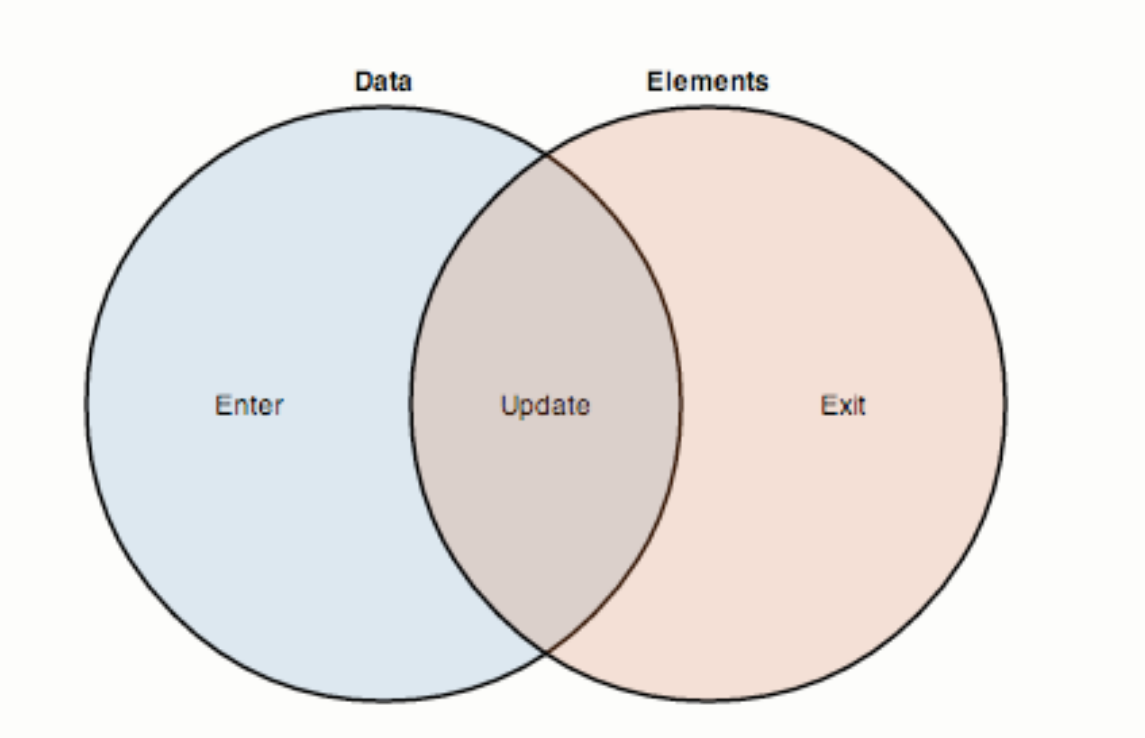
```
var data = [1, 1, 2, 3, 5, 8];
```

- Or objects:

```
var data = [
      {x: 10.0, y: 9.14},
      {x:  8.0, y: 8.14},
      {x: 13.0, y: 8.74}
];
```

- D3 provides a pattern for managing the mapping from data to elements: `enter()`, `update()`, `exit()`

Thinking with joins (in 60 seconds)



- `.enter()` figure out what's changed in the dataset
- `.update()` assign data items to visual elements
- `.exit()` delete items no longer being displayed

D3 example

```
svg.selectAll("circle")
      .data(data)
  .enter().append("circle")
      .attr("cx", x)
      .attr("cy", y)
      .attr("r", 2.5);
```

- The first line may be surprising: why select `<circle>` elements that don't exist yet?

- =>You're telling D3 that you want `circles` to correspond to `data` elements

D3 example (breakdown)

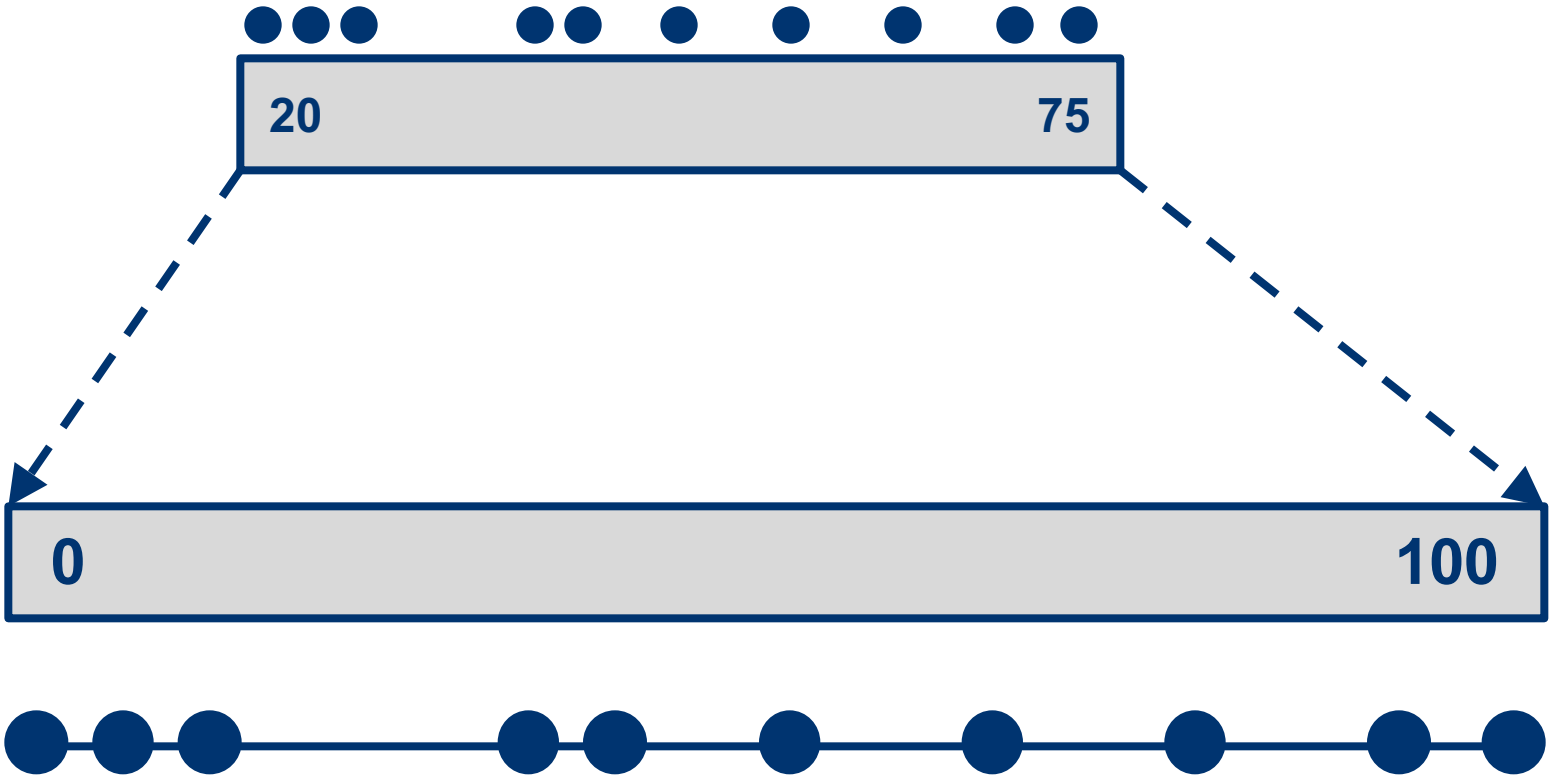
```
var circle = svg.selectAll("circle")
    .data(data);    // Data method computes the join,
                    // defining enter() and exit()

// Appending to the enter selection
// creates the missing elements

circle.enter().append("circle")
    .attr("cx", x(d) {
      return d.x;           // New elements are bound to
         })                 // data, so we can compute
    .attr("cy", y(d) {      // attributes using accessor
      return d.y;           // functions
    })
    .attr("r", 2.5);
```

Working with external data

- CSV: `d3.csv()`
- JSON: `d3.json()`
- Data loaded asynchronously (browser won't stall)
- Everything is returned as an array => JavaScript's built in array functions work:
  - `array.{filter,map,sort,…}`
- Additional data-transform methods; explore the API
  - `d3.{nest,keys,values,…}`

Scales

- Map from data space to visual space



Ordinal scales

- Map a discrete domain to a discrete range
- Essentially an explicit mapping

```
var x = d3.scaleOrdinal()
    .domain(["A", "B", "C", "D"])
    .range([0, 10, 20, 30]);

x("B"); // 10
```

- Often used to assign categorical colors

```
d3.scaleOrdinal()
    .range(d3.schemeCategory10)
```

Predefined range
(perceptually sound)

Quantitative scales

- Map a continuous (numeric) domain to a continuous range

```
var x = d3.scaleLinear()  // Others include scaleSqrt(),
      .domain([12, 24])   // scaleLog(), etc.
      .range([0, 720]);

x(16); // 240
```

- Typically, domains are derived from data:

```
.domain([0, d3.max(numbers)])
```

or

```
.domain(d3.extent(numbers))      // To compute min/max
                                 // simultaneously
```

- Need a compound (“polylinear”) scale? No problem: the domain and range can have more than two values!

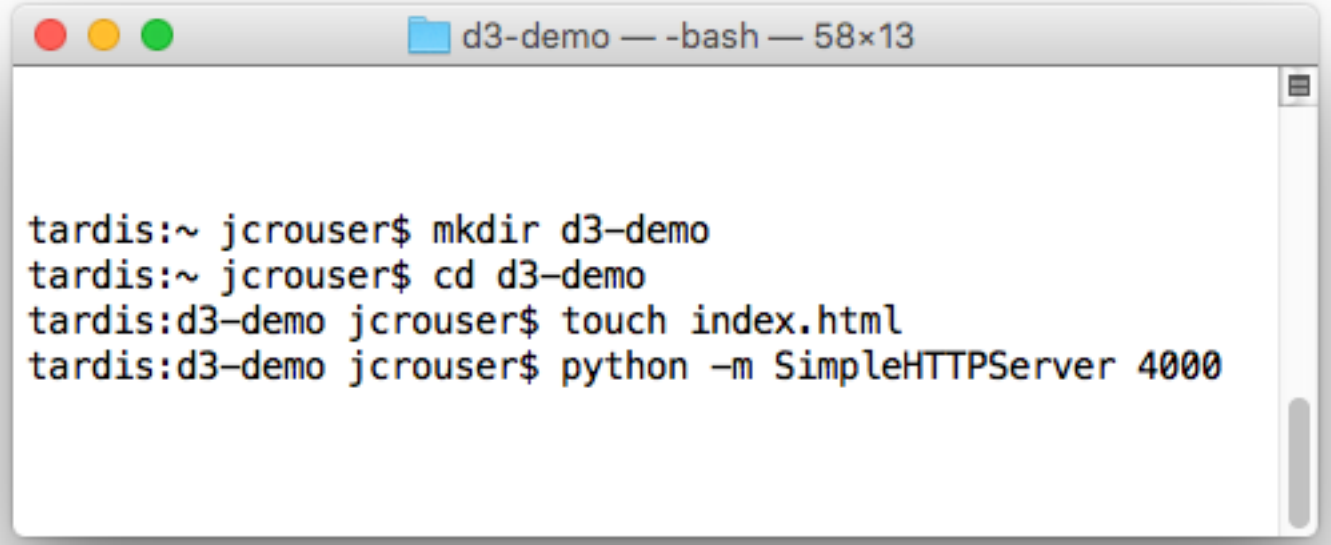
Line (and Area) Generators

- Can use scales to generate lines and regions

```
var x = d3.scale.linear(),
    y = d3.scale.linear();

var line = d3.svg.line()  // Line generator
     .x(function(d) { return x(d.x); })
     .y(function(d) { return y(d.y); });

svg.append("path")   // Pass data to the line generator
     .datum(objects)
     .attr("class", "line")
     .attr("d", line);
```

Let's play!

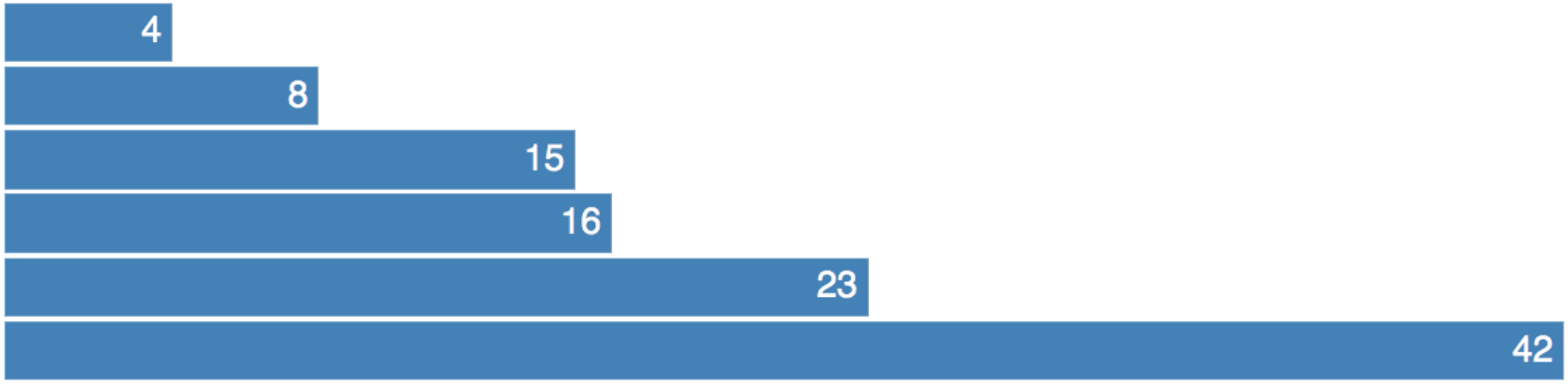
- Create a new directory
- Create an empty **index.html** file inside the directory
- Start up a little web server



Our data

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

Our goal



Attempt #1: pure HTML

```
<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>

<style>
.chart div {
    font: 10px sans-serif;
    background-color: steelblue;
    text-align: right;
    padding: 3px;
    margin: 1px;
    color: white;}
</style>
```

Attempt #2: with d3.js (setup)

```
<div class="chart"></div>

<style>
.chart div {
    font: 10px sans-serif;
    background-color: steelblue;
    text-align: right;
    padding: 3px;
    margin: 1px;
    color: white;}
</style>

<script src="https://d3js.org/d3.v4.min.js"></script>
```

Attempt #2: with d3.js

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



d3.select(".chart")   // Select the container
  .selectAll("div")   // Select the elements we want to bind
    .data(data)       // Bind the data to the selection
  .enter()
    .append("div")    // Append/style new elements
    .style("width",
           function(d) {return d * 10 + "px"; })
    .text(function(d)  { return d; });

</script>
```

Attempt #3: with d3.js and scales

```
<script>
var data = [4, 8, 15, 16, 23, 42];
var x = d3.scaleLinear()
    .domain([0, d3.max(data)])
    .range([0, 430]);

d3.select(".chart")   // Select the container
  .selectAll("div")   // Select the elements we want to bind
    .data(data)       // Bind the data to the selection
  .enter()
    .append("div")    // Append/style new elements
    .style("width",
           function(d) {return x(d) + "px";})
    .text(function(d) { return d; });

</script>
```

Attempt #4: with d3.js and SVG (setup)

```
<svg class="chart"></svg>

<style>
.chart rect {
  fill: steelblue;}

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

<script src="https://d3js.org/d3.v4.min.js"></script>
```

Attempt #4: with d3.js and SVG (setup)

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

var width = 420,
    barHeight = 20;

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

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

Attempt #4: with d3.js and SVG (setup)

```
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>
```

Discussion

- What are some of D3's strengths? Weaknesses?
- What are some of R's strengths? Weaknesses?