Visualization in D3

Cole Crawford
Jess Cohen-Tanugi

Schedule

- 1:45-1:55: Introductions; overview of D3
- 2:00-2:15: Development Environment; SVGs; Exercise 1
- 2:15-2:30: Binding Data; Using the Console; Exercise 2
- 2:30-3:00: Importing and Processing Data; Margins, Scales, Axes; University dataset introduction; Exercise 3
- 3:00-3:15: Break
- 3:15-4:45: D3 project structure; Enter, Exit, Remove; Interactivity (mouseover and filters); Exercise 4

What can you do with D3?

- https://www.hotel-frida.com/
- http://nycfoodiverse.com/
- https://hellosunapp.com/
- And many more … https://d3js.org/

Why use D3?

- From the D3 website:
  - D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.
- It's incredibly flexible: **D3 is more like a set of building blocks to help us build a chart or data visualisation from the ground up, rather than a library of pre-built charts. This means that there's very little limit (besides what's possible in a web browser) to what can be built with D3.**
- It's free!
- Tons of examples online you can piggyback from

When to reconsider using D3

- If you're not interested in coding
  - D3 takes time and Javascript knowledge to learn effectively
- If you don't know what you want to visualize yet
  - No prebuilt options (custom or nothing)
  - Not good for exploratory visualizations
- If you don't have time to tweak things
  - Code isn't always very reusable
- If your data is big (>5-10,000 records), it can be slow
  - Workarounds:
    - Server-side rendering
    - "Bucket" your data
- If you want to anonymize your data - difficult

Tools to compare with D3

- Highcharts: ready to go, off the shelf visualizations where you choose type of chart, data, some plugins, and you're good to go. Has a boost for large datasets. Not free for commercial use.
- React Vis (by Uber): has more high level functionality (many types of chart built in) while retaining customizability; most importantly, it integrates well with React, a popular javascript library for building user interfaces.
- RAWgraphs: D3 without the code. You can make similar things, but they're not going to be interactive. Sometimes the interface is wonky/slow, not the best for lots of data; limited choice of graphs.
- C3: D3 wrapper that allows you to customize by adding D3 if you want. Still coding, but less in the weeds.
- Mapbox: a place to build and host custom-made maps. Pay for premium access.

Here's a nice summary of pros and cons of some of these and other data visualization tools.

Getting started: SVGs.

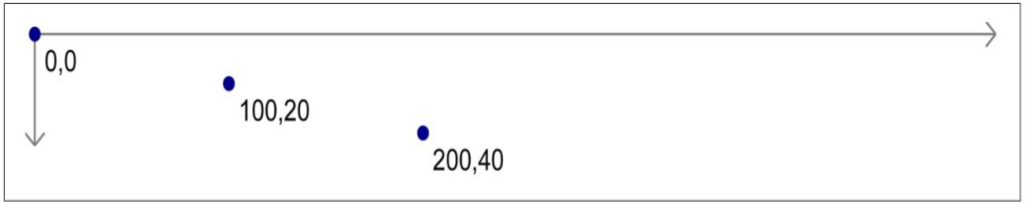
Before you can start drawing anything, you need to create an SVG element.

> “SVG is a text-based image format. Each SVG image is defined using markup code similar to HTML, and SVG code can be included directly within any HTML document, or inserted dynamically into the DOM. Every current web browser supports SVG.” Scott Murray, *Interactive Data Visualization*.

You can create shapes and put text in SVG elements, both of which we will do in our first exercise.

First, you create an empty SVG element, with a size in pixels, that you can fill with rectangles, circles, ellipses, lines, paths, and text.

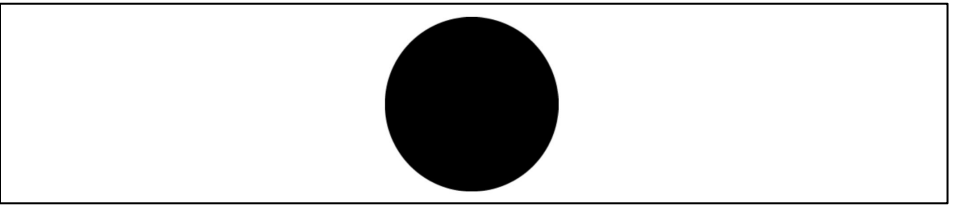
The SVG coordinate system.



If I created an SVG element of width=”500” height=”50” it would define a space this large with coordinates as shown.

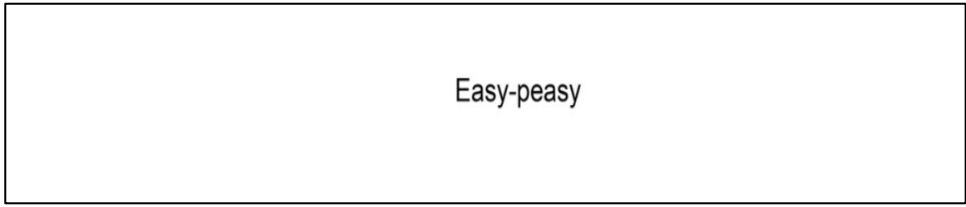
Fig. 3-13, Scott Murray, *Interactive Data Visualization*.

Then, we could add a circle within the SVG element like: <circle cx="250" cy="25" r="25"/> and get...



Whereas for text, you have to define where the text *starts* with the x, and the bottom line (the line the text “rests on” with the y:

<text x="250" y="25">Easy-peasy</text>:

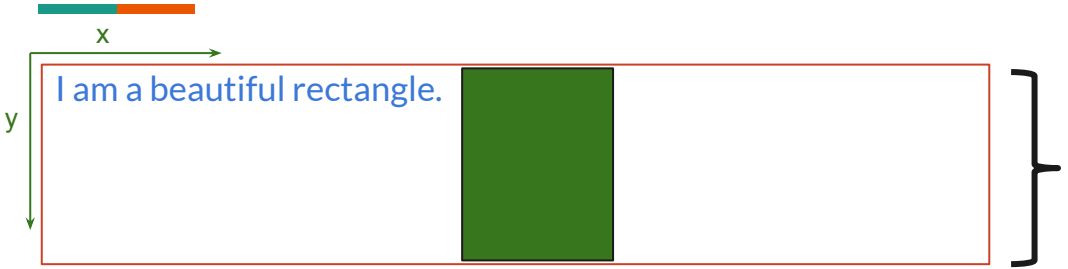


First exercise

Hello, SVG World! Let's get started.

- Clone the Datafest GitHub repo, or download and unzip our workshop materials zip from https://github.com/IQSS/datafest/tree/master/DataFest-2019/data-vis-d3
- Open up a terminal, navigate to that directory, and run
  - `python3 -m http.server 2222`
  - This starts up a python server running on port 2222 to serve our materials
- Open a web browser (Chrome or Firefox preferred) and go to http://localhost:2222/

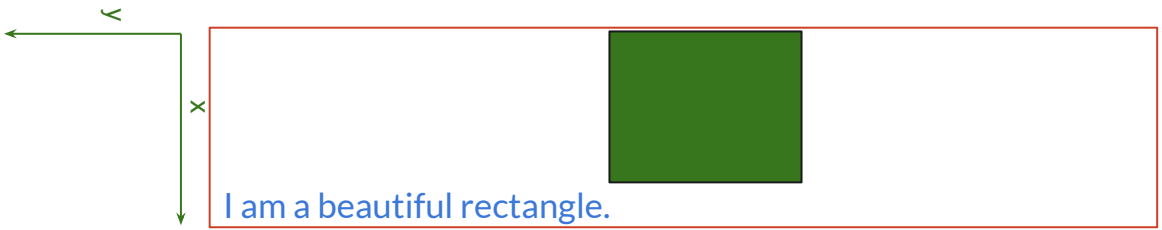
One last thing about SVGs: Transforms!




```
var h = 50
var w = 500

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

svg.append("text")
  .text("I am a beautiful rectangle.")
  .attr("x",0)
  .attr("y",15);

svg.append("rect")
    .attr("x",250)
    .attr("y",0)
    .attr("height",50)
    .attr("width",25)
    .attr("fill","black");
```

….

```
svg.append("text")
  .text("I am a beautiful rectangle.")
  .attr("transform", "translate(0,"+h+")");

svg.append("rect")
    .attr("x",0)
    .attr("y",-250)
    .attr("height",50)
    .attr("width",25)
    .attr("fill","green")
    .attr("transform", "rotate(90)");
```



Binding Data

- Data visualizations map data to visuals, using different visual cues to represent data attributes
- In D3, we want to map our data input values to elements in the DOM
- **Binding** is attaching or associating data to specific elements, so we can reference those values later to apply mapping rules

Binding Data

```
var dataset = [ 5, 10, 15, 20, 25 ];
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text("New paragraph!");
```

New paragraph!

New paragraph!

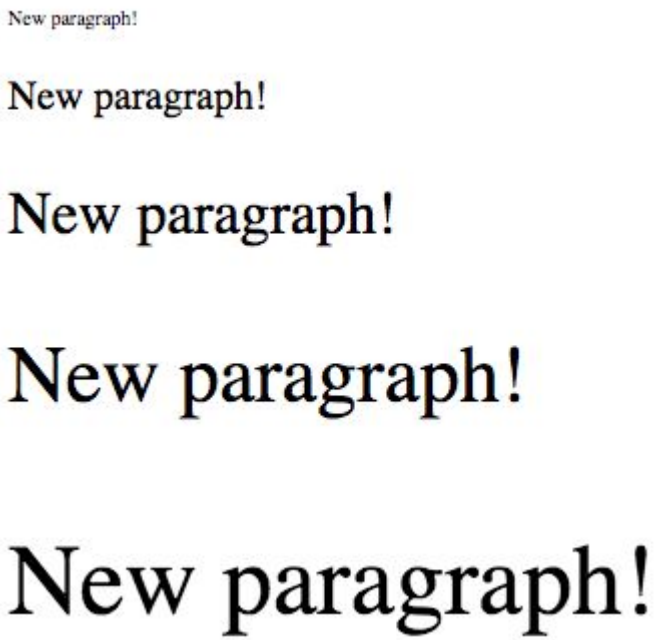
New paragraph!

New paragraph!

New paragraph!

Binding Data

```
var dataset = [ 5, 10, 15, 20, 25 ];
d3.select("body").selectAll("p")
  .data(dataset)
  .enter()
  .append("p")
  .text("New paragraph!")
    .style('font-size', function(d){
      return d;
    });
```



The Console

- Browser DevTools (especially Chrome / Firefox) are incredibly useful for developers
- Right click / Control click an element and click Inspect to see it in the DOM
- Command + Option + I / Ctrl + Shift + J to open Console
- Run commands in the console
- Use `console.log()` in your development code
- More on Chrome's DevTools: https://developers.google.com/web/tools/chrome-devtools/

Second exercise

- Hello SVG Worlds: `enter()` planetary data!
- http://localhost:2222/exercise2_helloWorlds/
- Challenges
  - Combine the data into one array of JS objects
    - https://www.w3schools.com/js/js_objects.asp
  - Join the new dataset to our elements
  - Label the planets with SVG `<text>` elements
  - Use the data to change the distances and colors of the planets to something meaningful

Importing Data

- `d3.csv()` and `d3.json()`
- File , callback function

```
d3.csv('/file_to_import.csv', function(imported_csv){
     console.log(imported_csv);
}

d3.json('https://url-to-api?param=123', function(data){
     console.log(data);
});
```

Processing Data

- Use the unary operator to cast strings into integers:

```
d.parameter = +d.parameter;
```

- Sort data: https://www.w3schools.com/js/js_array_sort.asp

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

- Filter data: https://www.w3schools.com/jsref/jsref_filter.asp

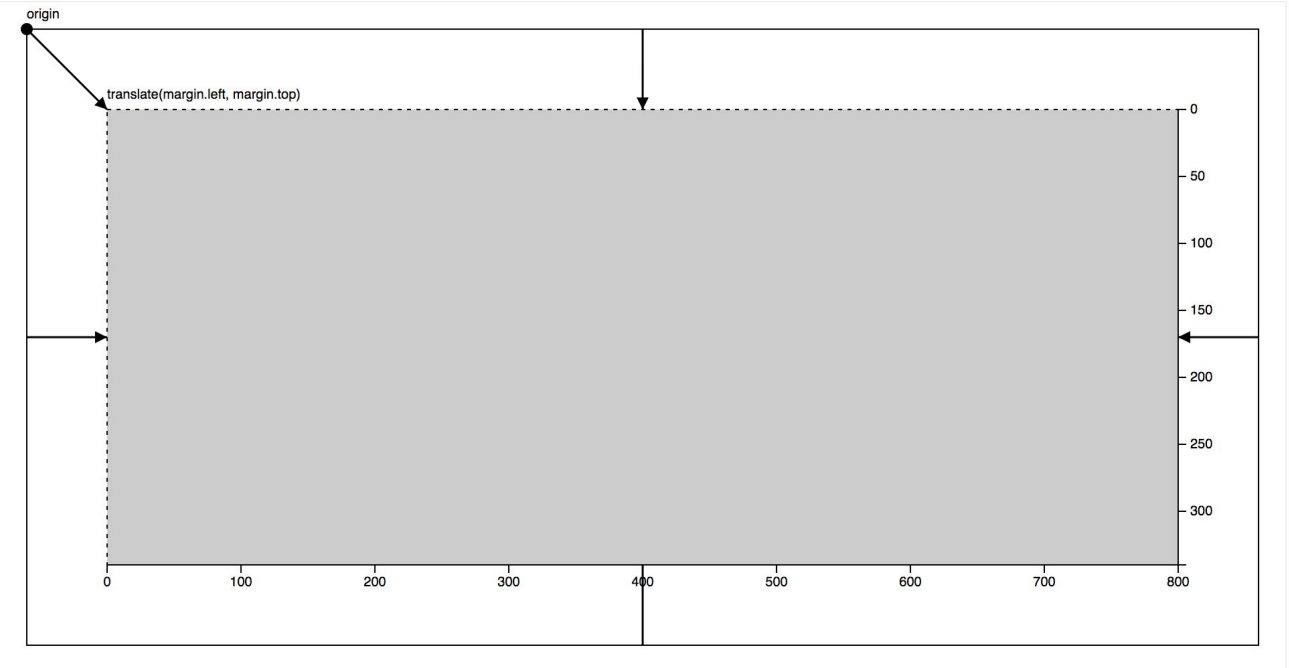
```
function checkProfessor(name){
     return name.indexOf('Dr') > -1;
}
data.filter(checkProfessor);
```

D3 Margins

- Mike Bostock's margin convention preserves space for axes without affecting the rest of the vis

```
var margin = {top: 20, right: 10, bottom: 20, left: 10};

var width = 960 - margin.left - margin.right,
    height = 500 - margin.top - margin.bottom;

var svg = d3.select("body").append("svg")
   .attr("width", width + margin.left + margin.right)
   .attr("height", height + margin.top + margin.bottom)
   .append("g")
   .attr("transform", "translate(" + margin.left + "," +
margin.top + ")");
```

D3 Margins


origin
translate(margin.left, margin.top)
0
50
100
150
200
250
300
0
100
200
300
400
500
600
700
800

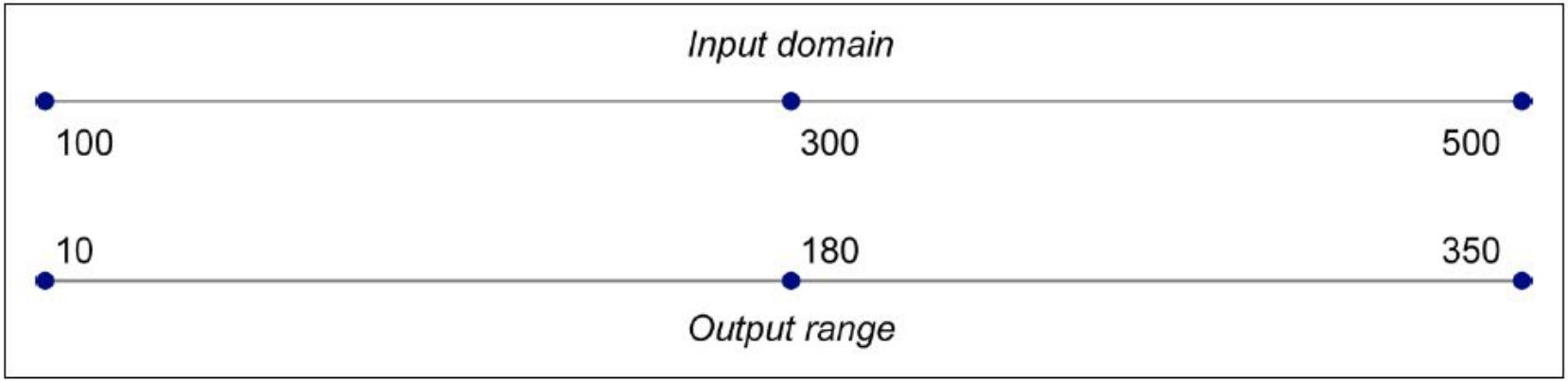
D3 Scales

"Scales are functions that map data from an input domain to an output range."

-Mike Bostock

- Scales make code easier to change
- Scales mean less time thinking about turning data into visual elements, and more time thinking about the data as data

D3 Scales



D3 Scales: Set a Domain and Range

```
var scale = d3.scaleLinear();

console.log(scale(2.5));

scale.domain([100,500])
     .range([10,350]);
console.log(scale(300));
```

D3 Scales: Max and Min

```
var dataset = [4, 7, 22, 111];
var max_number = d3.max(dataset, function(d) {
    return d;
});
console.log(max_number);
```

D3 Scales

- Basic scale (which we just used): linear
- Other scales:
  - Continuous input to continuous output: `scaleLinear, scalePow, scaleSqrt, scaleLog, scaleTime, scaleSequential`
  - Continuous input to discrete output: `scaleQuantize, scaleQuantile, scsleThreshold`
  - Discrete input to discrete output: `scaleOrdinal, scaleBand, scalePoint`
  - Color scales: `d3.schemeCategory10, d3.schemeInterpolateBlues, etc:` https://github.com/d3/d3-scale-chromatic
- More on scales
  - https://medium.com/@mbostock/introducing-d3-scale-61980c51545f
  - https://d3indepth.com/scales/

D3 Axes

- Scale: map data from an input domain to an output range
- Axis: visual representation of a scale
- Axes easily show how relationships between variables

D3 Axes

```
var svg = d3.select("body").append("svg")
                    .attr("width", 600)
                    .attr("height", 200);
var xScale = d3.scaleLinear();
xScale.domain([100,500])
     .range([10,350]);
var xAxis = d3.axisBottom().scale(xScale);
var xAxisGroup = svg.append('g')
                    .call(xAxis);
```

Third Exercise: Our Data

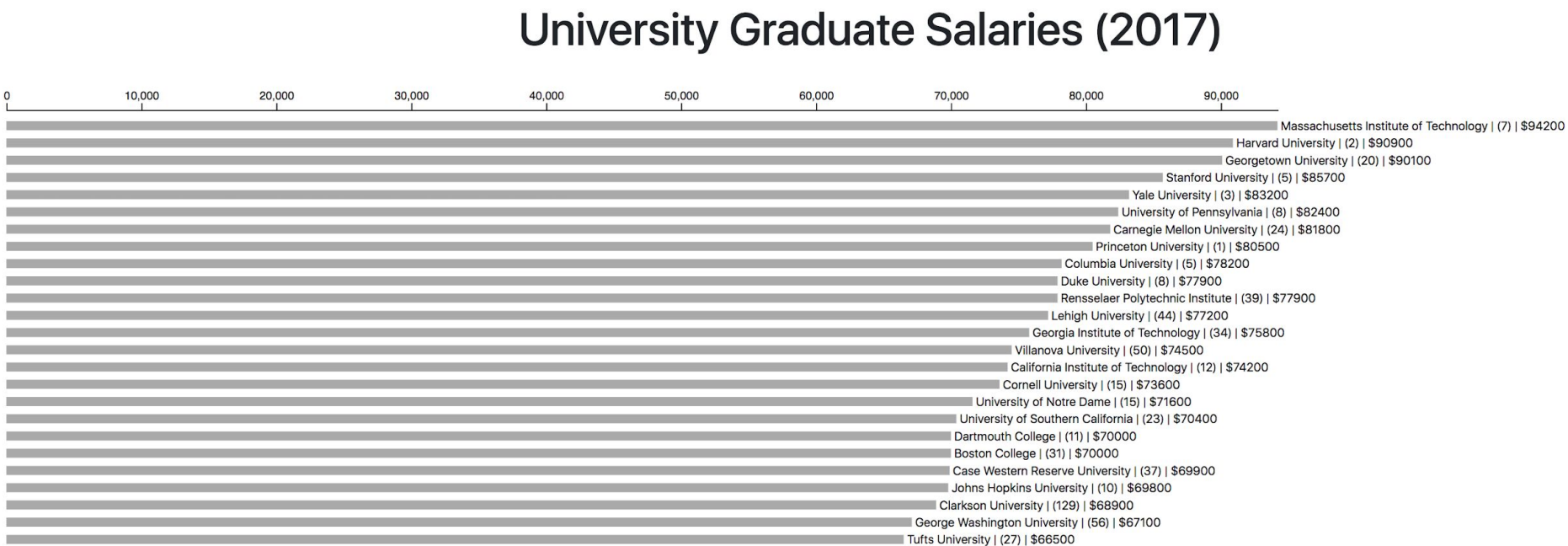
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	percentage_	four_year_gr	admit_rate	avg_grad_de	avg_needbas	avg_nonneec	instate_tuitic	location	name	rank	value_rank	grad_salary	state	total_cost_p	tuition_and_	undergrad_enrollment	
2	0	89	7	8908	41998	0		Princeton, NJ	Princeton Un	1	1	80500	NJ	61140	45320	5402	
3	1	86	5	16702	48598	23954		Cambridge, M	Harvard Univ	2	4	90900	MA	66609	47074	6699	
4	5	86	11	22256	47133	63649		Durham, NC	Duke Univers	8	5	77900	NC	69598	51265	6639	
5	21	87	11	24122	41331	22357		Nashville, TN	Vanderbilt U	15	7	64500	TN	64542	45610	6883	
6	17	83	15	22497	36772	19019		Houston, TX	Rice Univers	15	8	64300	TX	60658	43918	3910	
7	0	88	11	17849	46770	0		Hanover, NH	Dartmouth C	11	12	70000	NH	69787	51438	4307	
8	0	86	6	13625	50565	0		New Haven,	Yale Univers	3	13	83200	CT	70570	49480	5532	
9	0	85	8	24954	41767	0		Cambridge, M	Massachuset	7	17	94200	MA	65612	48452	4527	
10	0	81	8	18219	41901	5000		Pasadena, CA	California Ins	12	18	74200	CA	66027	47577	1001	
11	1	75	5	21987	45318	13697		Stanford, CA	Stanford Uni	5	20	85700	CA	66184	47940	6999	
12	0	86	6	23463	50733	0		New York, NY	Columbia Un	5	24	78200	NY	72627	55056	6102	
13	0	87	14	23389	39595	0		Ithaca, NY	Cornell Unive	15	29	73600	NY	68163	50953	14315	
14	7	85	25	29217	40816	25795		Atlanta, GA	Emory Unive	20	30	61500	GA	64510	47954	6867	
15	9	67	46	21645	7083	2531	6389	Gainesville, F	University of	50	32	53100	FL	39779	28666	35043	
16	3	88	12	23593	38238	39352		Baltimore, M	Johns Hopkin	10	34	69800	MD	69310	50410	6524	
17	0	84	9	23810	44105	8704		Providence, R	Brown Unive	14	35	63100	RI	69010	51367	6652	
18	14	88	17	23577	40509	6111		St. Louis, MO	Washington	19	40	66300	MO	68531	49770	7504	
19	0	85	9	24536	43899	0		Philadelphia,	University of	8	41	82400	PA	69880	51464	9726	

Dataset: Fields expanded

percentage_nonneedbased_aid
four_year_grad_rate
admit_rate
avg_grad_debt
avg_needbased_aid
avg_nonneedbased_aid
instate_tuition
location
name
rank
value_rank
grad_salary
state
total_cost_per_year
tuition_and_fees
undergrad_enrollment

The goal for exercise 3 is to make this.

Exercise 3: D3 rendering



Structuring a D3 Project: Files

- Common structure is useful
  - Don’t need to relearn every time
  - Organization
  - Easier to share and maintain

Structuring a D3 Project: Files

```
/project
    /css
        style.css, bootstrap.min.css, etc
    /data
        university_data.csv, test_scores.json, etc
    /fonts
    /images
    /js
        d3.min.js, jquery.min.js, main.js, scatterplot.js, etc
    index.html
```

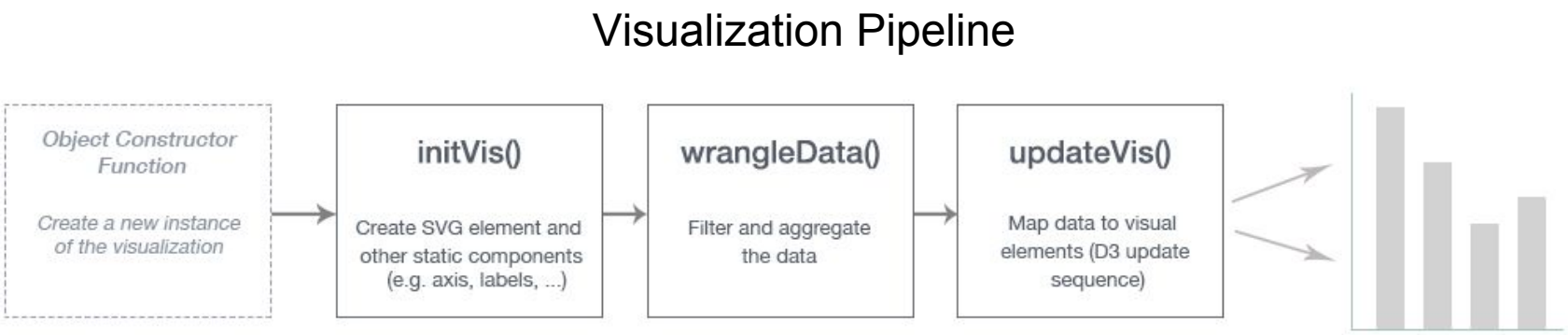
Structuring a D3 Project: Code

- Multiple Javascript files
  - Keep the codebase readable
  - Encapsulation
  - Dashboards
  - One main, one per visualization
- `main.js`
  - Global variables
  - Load the data
  - Maintain the state (if necessary)
- `vis.js`
  - Allows each visualization to exist separately
  - Allows for code reuse, if doing small multiples
  - State gets maintained internally for each vis

Structuring a D3 Project: Code

- Multiple Javascript files
  - Keep the codebase readable
  - Encapsulation
  - Divide and conquer: one main.js, one JS file per visualization
  - Use case: dashboards
- `main.js`
  - Global variables
  - Load the data
  - Maintain the state (if necessary)
  - Creates instances of all visualizations
- `vis.js`
  - Allows each visualization to exist separately
  - Allows for code reuse, if doing small multiples
  - State gets maintained internally for each vis

Structuring a D3 Project: Reusable Components



Credit: CS 171

Structuring a D3 Project: Reusable Components

```
var barchart = new BarChart("bar-chart-container", data);

BarChart = function(_parentElement, _data){
      this.parentElement = _parentElement;
      this.data = _data;

      this.initVis();
}

BarChart.prototype.initVis = function(){
      ...
}
```

Credit: CS 171

Structuring a D3 Project: Reusable Components

```
BarChart.prototype.initVis = function(){
      var vis = this;

      vis.margin = { top: 20, right: 0, bottom: 60, left: 60 };

      vis.width = 800 - vis.margin.left - vis.margin.right,
      vis.height = 400 - vis.margin.top - vis.margin.bottom;

      vis.svg = d3.select("#" + vis.parentElement).append("svg")
            .attr("width", vis.width + vis.margin.left + vis.margin.right)

      ...

      vis.wrangleData();
}
```

Credit: CS 171

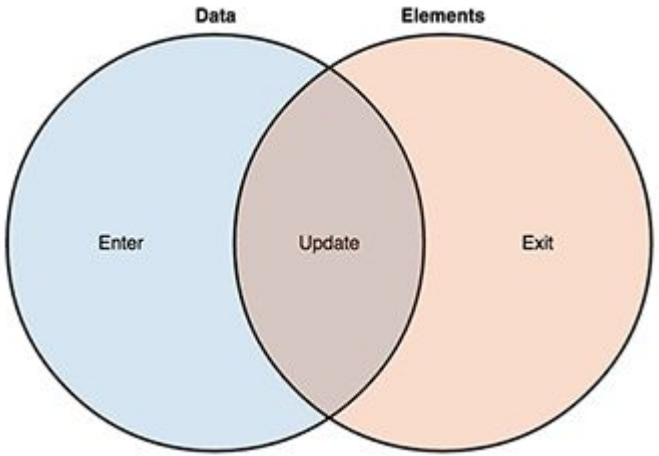
Structuring a D3 Project: Reusable Components

```
BarChart.prototype.wrangleData= function(){
      var vis = this;
      // Filter example
      vis.displayData = vis.data.filter(function(d){
            return d.year > 2000;
      });
      // Draw visualization
      vis.updateVis();
}
```

```
BarChart.prototype.updateVis =function(){
      var vis = this;

      // Update domains
      vis.y.domain([0, d3.max(vis.displayData, function(d) {
            return d.price; })]);
      ...

      // Draw the actual bar chart
      var bar = svg.selectAll(".bar")
      .data(data);

      bar.enter().append("rect")
            .attr("class", "bar")
            ...
}
```

Credit: CS 171

Dynamic Data: Enter, Update, Remove



Credit: CS 171

Dynamic Data: Enter, Update, Remove

```
var svg = d3.select("body").append("svg")
      .attr("width", 600)
      .attr("height", 200);

var circle = svg.selectAll("circle")
     .data([5, 10, 15]);

circle = circle.enter().append("circle")
      .attr("r", function(d) { return d; })
      .attr("cx", function(d, index) { return (index * 80) + 50 })
      .attr("cy", 80);
```

Dynamic Data: Enter, Update, Remove

```
var svg = d3.select("body").append("svg")
      .attr("width", 600)
      .attr("height", 200);

updateChart([5, 10, 15]);
updateChart([20, 30]);

function updateChart(data) {
      // Data-join (circle now contains the update selection)
      var circle = svg.selectAll("circle")
            .data(data);
      // Enter (initialize the newly added elements)
      circle.enter().append("circle")
            .attr("class", "dot")
            .attr("fill", "#707086")
            // Enter and Update (set the dynamic properties of the elements)
            .merge(circle)
            .attr("r", function(d) { return d; })
            .attr("cx", function(d, index) { return (index * 80) + 50 })
            .attr("cy", 80);
      // Exit
      circle.exit().remove(); }
```

Interacting With Data: Tooltips

- Tooltips
  - Show details about specific data elements
  - Append an element to hold the tooltip
  - Use mouseover() and mouseout()

```
var tooltip = d3.select("body").append("div")
    .attr("class", "details")
    .style("opacity", 0);

circle …
      .on("mouseover", function(d){
        tooltip
          .style("opacity", .9)
          .html("<span>" + d + "</span>")
          .style("left", (d3.event.pageX) + "px")
          .style("top", (d3.event.pageY) + "px");
```

Interacting with Data: Tooltips

```
var tooltip = d3.select("body").append("div")
    .attr("class", "details")
    .style("opacity", 0);

function updateChart(data) {
        var circle = svg.selectAll("circle")
                .data(data);
        circle.enter().append("circle")
                .attr("class", "dot")
                .attr("fill", "#707086")
                .merge(circle)
                .attr("r", function(d) { return d; })
                .attr("cx", function(d, index) { return (index * 80) + 50 })
                .attr("cy", 80)
                .on("mouseover", function(d){
                        tooltip
                          .style("opacity", .9)
                          .html("<span>" + d + "</span>")
                          .style("left", (d3.event.pageX) + "px")
                          .style("top", (d3.event.pageY) + "px");
                });
        circle.exit().remove();
}

var data = [10, 20, 30];
updataChart(data);
```

Dynamic Data: Filtering

- We can use other HTML elements (checkbox, selectbox, etc) to filter our data by triggering the wrangleData() part of our visualization pipeline
- jQuery ($) can be useful for doing non-vis specific DOM manipulation

index.html

```
<select multiple class="form-control" id="filter"></select>
```

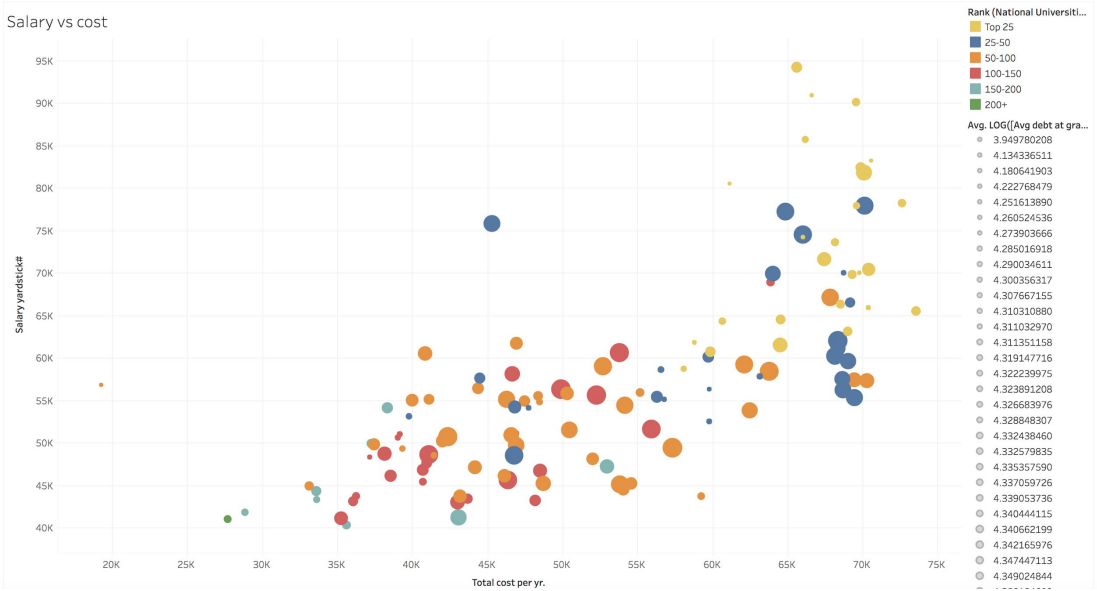
main.js

```
var data = [5, 10, 20, 50]
data.forEach(function(d){
	$("#filter").append(`<option value="${d}" selected="selected">${d}</option>`);
});
```

vis.js wrangleData()

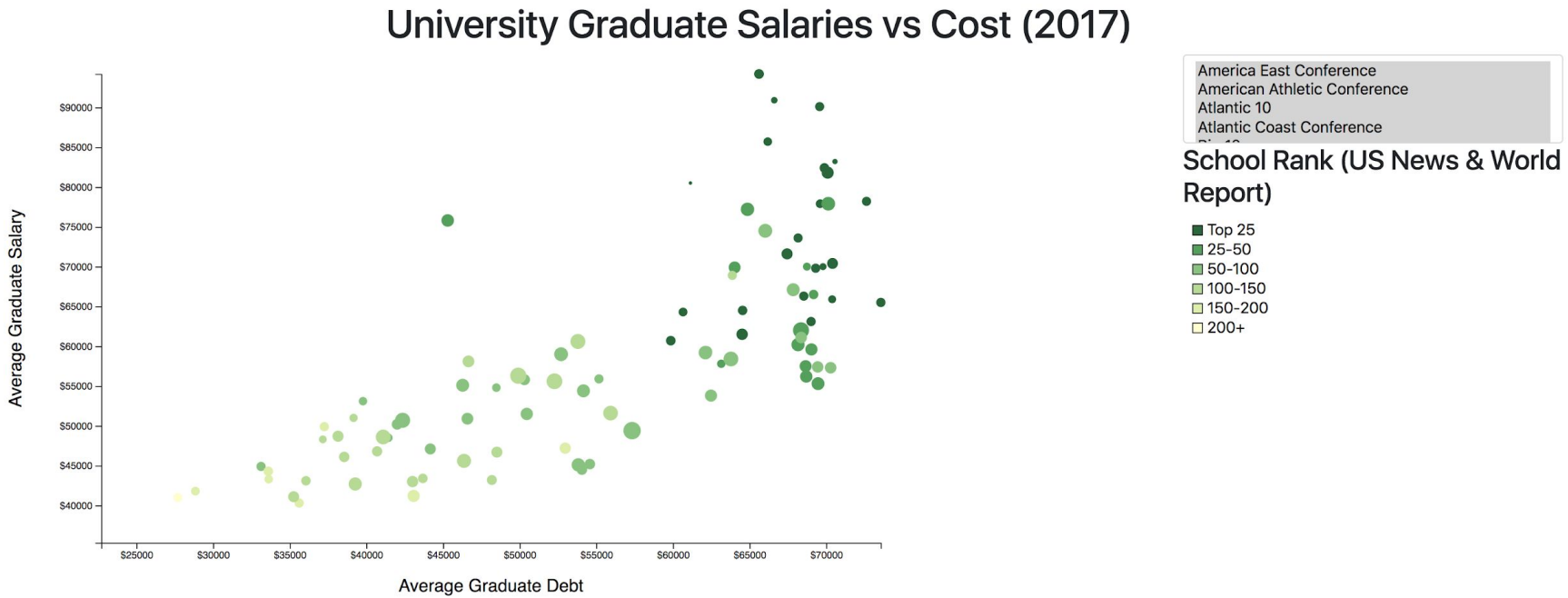
```
var filtered = $('#filtered').val();
var displayData = data.filter(function(d){
	return selected.includes(d);
});
```

Capstone exercise: Tableau representation



https://public.tableau.com/profile/jess.ct#!/vizhome/D3trials/Salaryvscost?publish=yes

Capstone Exercise: D3 representation



Capstone Challenges

- Add a legend
- Add tooltips or labels to each school dot
- Add a filter for school rank

More Resources

- Interactive Data Visualization for the Web (Scott Murray)
- D3 in Depth
- Bl.ocks (especially Mike Bostock)
- Read the docs

Digital Scholarship Support Group

The Digital Scholarship Support Group brings together Harvard faculty and staff with expertise in a host of technical, pedagogical, and subject-specific areas across disciplinary and divisional borders, to provide faculty, students, and staff interested in incorporating digital methods into their teaching and research with a single point of entry to the many resources available at Harvard.

https://dssg.fas.harvard.edu/