Data-Driven Documents

Authors: Troy Walton, Nai Wang, Minhua Wu, Marshall Wurangian

Lecture Summary

In this lecture, we saw a presentation from our guest speaker, Diego Garcia-Olano, about data visualization. We got to hear about D3 and see examples of visualizations using the D3 library. Our guest speaker shared some of his own work and discussed some visualization techniques. We also got to see how we could go about visualizing data ourselves whether we build from scratch using D3 or take advantage of existing visualization codes as a framework.

Background

D3 stands for Data-Driven Documents. It is a JavaScript library that allows you to create data-driven visualizations using HTML, CSS and SVG. D3 can support large dataset for dynamic interactions. Its code can be applied and resued in a variety of modules.

`.select()`

Selecting different **D**ocument **O**bject **M**odel (DOM) elements requires the use of `.select` and `.selectAll`. D3 is selecting all the `li` elements inside of a `ul` tag in an HTML document. After execution, the code returns a selection object containing the `li` elements from the DOM.

```
d3.select("ul").selectAll("li")
```

Manipulating individual nodes: Here you are selecting nodes by tag name "body".

```
d3.select("body").style("background-color", "black");
```

`.data()`

You can also chain `.data()` to the selector, then add an array to the mix:

```
d3.select("ul").selectAll("li").data()
```

```
var arr = [20, 25, 53, 59, 98];
d3.select("ul").selectAll("li").data(arr)
d3.select("ul").selectAll("li").data()
```

`.text()`

After data has been bound to an element, you can manipulate the elements using `.text` with a callback function. This callback function is called with each element in the selection.

```
var arr = [20, 25, 53, 59, 98];

d3.select("ul").selectAll("li")
  .data(arr)
  .text(function (d) {
    return d;
  });
```

.enter() & .append()

You can use `.enter()` to create a sub-selection for data that hasn't been mapped to an element. `.append()` will then pair an element to the sub-selection.

```
var arr = [20, 25, 53, 59, 98]

// Update existing elements
d3.select("ul")
  .selectAll("li")
  .data(arr)
  .text(function (d) {
    return d;
  });

// Create new elements for additional points
d3.select("ul")
  .selectAll("li")
  .data(arr)
  .enter()
  .append("li")
  .text(function (d) {
    return d;
  });
```

.exit() & .remove()

You can remove an element: `.exit()` will create a selection of the surplus. `.remove()` will remove them from the DOM.

```
var arr = [20, 25, 53, 59, 98]
d3.select("ul")
  .selectAll("li")
  .data(arr)
  .exit()
  .remove()
```

`.attr() & .style()`

Attributes and styling can be applied with `.attr()` and `.style()`. Here we're selecting the `li` elements and setting the text color to blue. Additionally, with `.attr()`, we added a class to each `li` element.

```
d3.select("ul").selectAll("li")
  .style("color", "blue")
  .attr("class", "myList")
```

D3 is composed of several components:

- HTML - layout of a web page
- JavaScript - the code that you input
- CSS (Cascading Style Sheets) - describes the presentation i.e. colors, fonts, spacing, etc.
- SVG (Scalable Vector Graphics) - web standard for geometric i.e. circles, lines, arrows
- Your Data such as .csv file of election results, topojson for creating maps, json, tsv, etc.

What is great about D3 is that it is open-source with many community-developed resources. There is much space for creative freedom with D3's many features: treemap, hierarchical edge bundling, Sankey diagram, density contours, force-directed graphs, map projections, heatmaps. D3 is much more focused on composable primitives such as shapes and scales than configurable charts. It is boundless without constraints and provides application for animation and interaction with end-users.

Here are a few examples:

Example 1

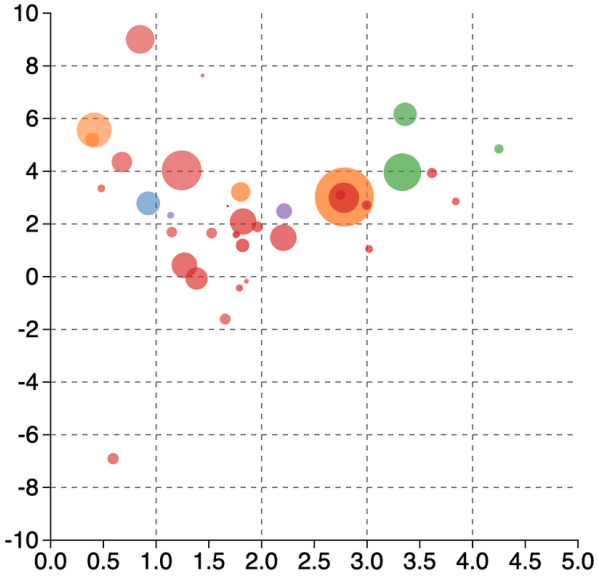
Let's say you load a .csv file called data.csv which details a country's name, continent, population, GDP growth, Gross Domestic Expenditure on R&D (GERD), and GDP Growth. An incredibly well-put information regarding D3 could be found on https://observablehq.com/@d3/learn-d3?collection=@d3/learn-d3 (https://observablehq.com/@d3/learn-d3?collection=@d3/learn-d3) which details an introduction to D3, tutorials, and examples.

```
country,continent,population,GDPcap,GERD,growth
Australia,Oceania,22319.07,40718.78167,2.21367,2.48590317
Austria,Europe,8427.318,42118.46375,2.74826,3.10741128
Belgium,Europe,10590.44,38809.66436,1.96158,1.89308521
Canada,America,33909.7,39069.91407,1.80213,3.21494841
Chile,America,17248.45,15106.73205,0.39451,5.19813626
Czech Republic,Europe,10286.3,25956.76492,1.52652,1.65489834
Denmark,Europe,5495.246,40169.83173,3.01937,1.04974368
Estonia,Europe,1335.347,22403.02459,1.44122,7.63563272
Finland,Europe,5366.482,37577.71225,3.84137,2.85477157
```

```
d3.csv("data.csv", function(csv) {
    csv.sort(function(a,b) {return b.population-a.population;}); //
    svg.selectALL("circle").data(csv).enter()
        .append("circle")
        .attr("cx", function(d) {return x(+d.GERD);})
        .attr("cy", function(d) {return y(+d.growth);})
        .attr("r", function(d) {return r(Math.sqrt(+d.population));})
        .style("fill", function(d) {return c(d.continent);})
        .style("opacity", function(d) {return o(+d.GDPcap);})
        .append("title")
        .text(function(d) {return d.country;})

})
```

The visualization output will be as follows:



When you create these visualizations on html, you could double click "inspect" on the graph and see each line of code that corresponds with the visual. Furthermore, you could edit the code on "inspect".

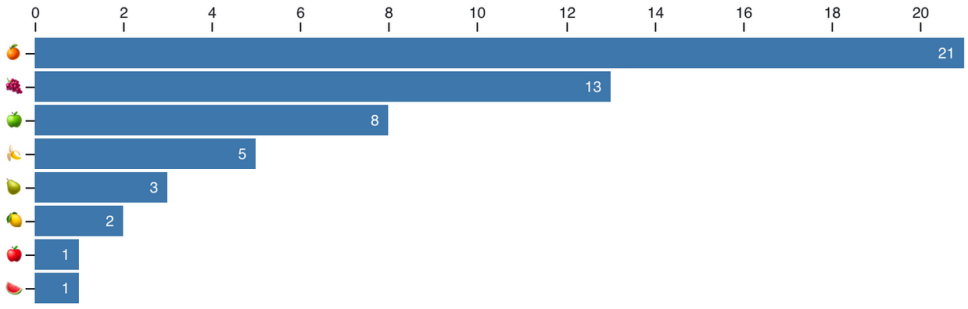
Example 2

Let's say you make your own dataset

```
fruits = [
  {name: "🍊", count: 21},
  {name: "🍇", count: 13},
  {name: "🍏", count: 8},
  {name: "🍌", count: 5},
  {name: "🥭", count: 3},
  {name: "🍋", count: 2},
  {name: "🍎", count: 1},
  {name: "🍉", count: 1}
]

fruits.map(d => d.count) // the count dimension (quantitative)
fruits.map(d => d.name) // the name dimension (nominal)
x = f(n)
x = d3.scaleLinear()
    .domain([0, d3.max(fruits, d => d.count)])
    .range([margin.left, width - margin.right])
    .interpolate(d3.interpolateRound)
y = f(i)
y = d3.scaleBand()
    .domain(fruits.map(d => d.name))
    .range([margin.top, height - margin.bottom])
    .padding(0.1)
    .round(true)
```

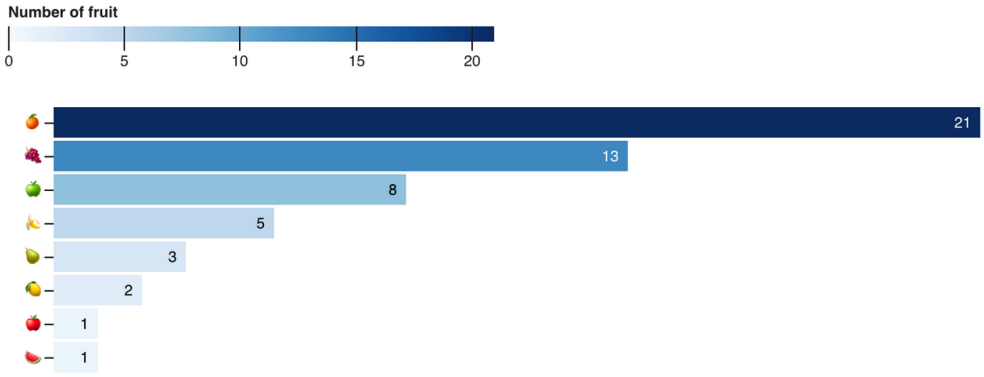
The output will be as follows:



A much complicated code using the same consepts above in html follows:

```
<svg viewBox="0 ${margin.top} ${width} ${height - margin.top}" style="max-width:
 ${width}px; font: 10px sans-serif;">
  <g>
    ${fruits.map(d => svg`<rect fill="${color(d.count)}" y="${y(d.name)}" x="${x
(0)}" width="${x(d.count) - x(0)}" height="${y.bandwidth()}"></rect>`)}
  </g>
  <g text-anchor="end" transform="translate(-6,${y.bandwidth() / 2})">
    ${fruits.map(d => svg`<text fill="${d3.lab(color(d.count)).l < 60 ? "white" : "
black"}" y="${y(d.name)}" x="${x(d.count)}" dy="0.35em">${d.count}</text>`)}
  </g>
  ${d3.select(svg`<g transform="translate(${margin.left},0)">`)
    .call(d3.axisLeft(y))
    .call(g => g.select(".domain").remove())
    .node()}
</svg>`
```

...which produces this sophisticated visual that maps count to color.



Grouping & Normalization

We talked about different visualization techniques such as grouping and normalization at the end of the class. Our group summarized of pros and cons of the techniques and thought it would be helpful to the class.

Grouping

Pros: Could be a good way to add dimensions to the story you're tell without using additional computing power. For example, through the COVID example during the class we learned grouping the COVID case graph of each state in its geographical position in the U.S conveys the geographical distribution of COIVD better the plain text or graphs along. **Cons:** One drawback of this approach is the assumption of people sharing the same common knowledge, let's say if the audience is not from the U.S the geographical lay out of the graphs won't add any value.

Normalization

Instead of writing pros and cons for normalization, we though it is better to call it trade-off more than anything else. They key to normalization is to know the audience and to know the highlight of the message being conveyed. For example, in terms of the severity of COVID in each state, we would use the absolute number of the cases if we want to highlight the states with the most cases. On the other hand, we could use a normalized version such as infection/1000 people to give the audience a aspect that is more fair.