Data visualization software

Andrew Peng
October 23, 2017

Introduction to Data visualization software

Data visualization is the creation and visual representation of data. The primary goal of data visualization is to communicate to a viewer the trends and information behind the data using graphics and plots. It is often used as a way to quickly summarize a conclusion that could take paragraphs to explain. Early data vizualization was done by hand, but in the present day most data visualization is created with computer software. In the age of "Big Data" it is imperative that we can understand the large amount to unstructured data being collected. Data visualization plays a big role in machine learning and predictive statistics in order to see if the model is performing as intended. It is also important to be able to view and monitor metrics for large scale distributed systems. A good data visualization will present the required information to the reader in a simple and elegant manner. It should not be confusing to read, and it should be pleasing to the eye. Many software packages exist today to create beautiful interactive visualizations. I will focus on one in particular, D3.js, which is a frontrunner in data visualization because of its power and flexibility, as well as its ease of use,

Examples of data visualization

Which fish are OK to eat?

Swimming World Records

Google search volume by language

Introducing D3 - Data Driven Documents

In 2011, Mike Bostock, Jeff Heer, and Vadim Ogievetsky of Stanford's Visualization Group developed D3.js. D3 is a Javascript library specifically created to produce dynamic interactive charts on the web. It uses SVG, HTML, and CSS3 to create the graphs and is used on thousands of websites. It is fully open source and available for use for anyone under the BSD Software Liscence. Many other data visualization libraries, such as Plotly and epoch are built on top of D3.

Using D3

D3 allows users to bind data the DOM (Document Object Model) and then apply transformations to the document based on the data. To include D3, you can download the library and include it in your source or put this in your HTML

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

Data

D3 accepts many kinds of data that R does. It will handle arrays of integers, numeric values, strings as well as JSON and GeoJSON data. Data can be loaded as TSV, CSV, or through HTTP requests, making D3 very formidable when creating web applications.

```
// basic hand created dummy data
var dataset = [1, 2, 3, 4, 5, 6, 7, 8, 9]

// data can be loaded from CSV, TSV, text, HTML etc
d3.csv("/data/cities.csv", function(data) {
  console.log(data[0]);
});

d3.tsv("/data/cities.tsv", function(data) {
  console.log(data[0]);
});

d3.html("/data/cities.html", function(data) {
  console.log(data[0]);
});

// data can be loaded from HTTP requests
d3.csv("http://www.example.com/data.csv", function(error, data) {
  // the code in here will be run when the asynchronous request finished
  // the data parsed from the CSV will be available in data
});
```

Selections

The main benefit of D3 is the ability to select certain elements of the DOM and apply transformations or data-binding similar to jQuery syntax. It can select plain HTML5 or CSS classes. For example, to select all the `h1` headers of a file and change the color to hex value `#7e9560`

```
d3.selectAll("h1").style("color", "#7e9560");
```

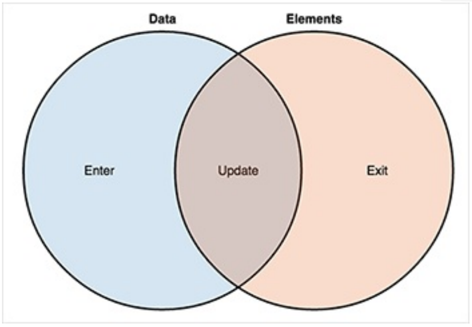
You can also select individual nodes

```
d3.select("#my-class").style("color", "#7e9560");
```

Databinding

D3 allows you to bind data to elements on the DOM, allowing them to be manipulated and updated easily. This makes it possible for charts to become increasingly dynamic and interactive. Data binding maps our data to specific DOM elements, which means the elements attributes can be modified through the data itself.

D3 works using the concept of the *data join*. D3 declares a relationship between data and DOM elements and the overlapping areas are the `update` section. Data not yet bound to elements are the `enter` section and elements not bound to data are the `exit` section.



Here is an example

```
var dataset = [1, 2, 1, 2, 1, 1, 2, 2, 1]

d3.select('svg').selectAll('circle')
  .data(dataset)
  .enter()
  .append('circle')
  .attr('r', function(d) {
    return d;
  })
```

1. `d3.select('svg').selectAll('circle')` will create an empty selection under the `svg` container of `circles`. This

will soon be populated!

2. `.data(dataset).enter()` will bind our data `dataset` to the current selection. It will result in three selections `enter()`, `exit()`, and `update()`. The update selection is returned with the `.data()` call. The enter and exit selections are returned with an additional call to `enter()` or `exit()`.
3. The missing elements are added to the SVG container with `.append('circle')`.
4. `.attr(,callback)` will allow you to modify the created or updated element corresponding to the attributes of the data. In the code above the radius of each circle is determined by the data value itself.

Drawing graphs

We can use all of the above to draw a very simple bar chart. Credit to Mike Bostock, the creator of D3, for creating many tutorials for different types of charts and uses for d3. His website can be found here.

Consider this fake dataset counting the frequency of letters in the alphabet

Data.tsv

```
letter  frequency
A      .08167
B      .01492
C      .02782
D      .04253
E      .12702
F      .02288
G      .02015
H      .06094
I      .06966
J      .00153
K      .00772
L      .04025
M      .02406
N      .06749
O      .07507
P      .01929
Q      .00095
R      .05987
S      .06327
T      .09056
U      .02758
V      .00978
W      .02360
X      .00150
Y      .01974
Z      .00074
```

Here is the code to create a bar chart

Base HTML

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

.bar {
  fill: steelblue;
}

.bar:hover {
  fill: brown;
}

.axis--x path {
  display: none;
}

</style>
<svg width="960" height="500"></svg>
<script src="https://d3js.org/d3.v4.min.js"></script>
```

This is just boilerplate HTML with some CSS styling to add colors etc. Don't forget to include the d3 script. The javascript code below can be included inline or through another external script.

```
var svg = d3.select("svg"),
    margin = {top: 20, right: 20, bottom: 30, left: 40},
    width = +svg.attr("width") - margin.left - margin.right,
    height = +svg.attr("height") - margin.top - margin.bottom;

var x = d3.scaleBand().rangeRound([0, width]).padding(0.1),
    y = d3.scaleLinear().rangeRound([height, 0]);

var g = svg.append("g")
    .attr("transform", "translate(" + margin.left + "," + margin.top + ")");
```

`x` and `y` are the axes of the graph. They can be specified as continuous or discreet values and the sizing and margins of the graph can also be configured using d3. The width and height here are taken from the size of the SVG container.

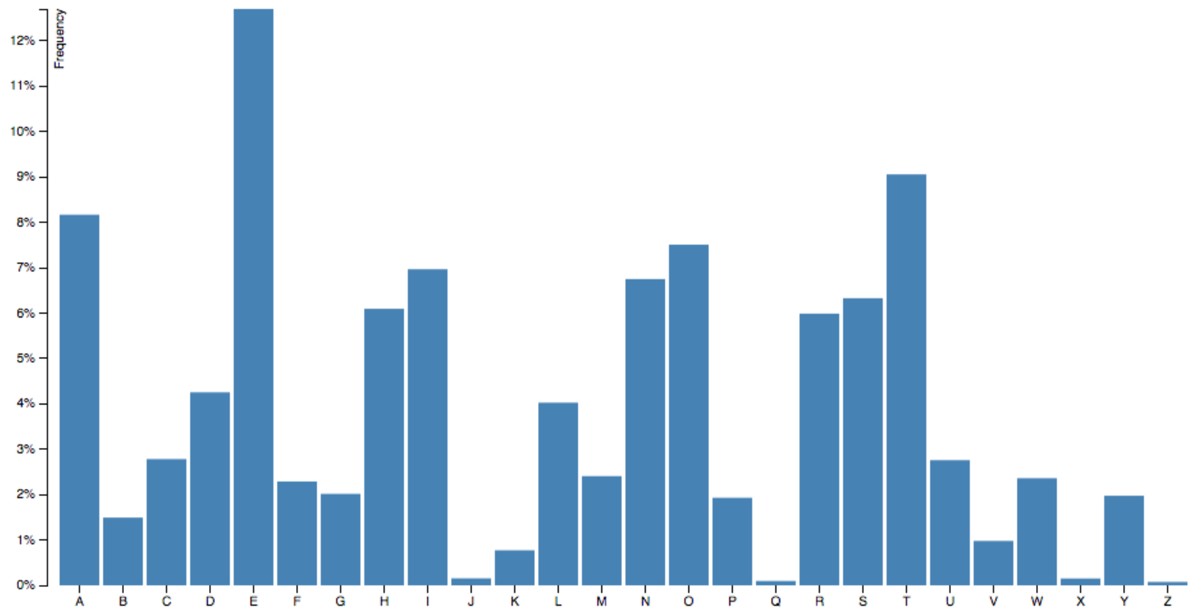
The data is loaded with `d3.tsv` and as a sanity check all frequencies are changed to be positive when the data is loaded. A anonymous callback function is used to handle errors and perform data joins.

```
d3.tsv("data.tsv", function(d) {
  d.frequency = +d.frequency;
  return d;
}, function(error, data) {
  if (error) throw error;

  x.domain(data.map(function(d) { return d.letter; }));
  y.domain([0, d3.max(data, function(d) { return d.frequency; })]);

  g.append("g")
      .attr("class", "axis axis--x")
      .attr("transform", "translate(0," + height + ")")
      .call(d3.axisBottom(x));

  g.append("g")
      .attr("class", "axis axis--y")
      .call(d3.axisLeft(y).ticks(10, "%"))
    .append("text")
      .attr("transform", "rotate(-90)")
      .attr("y", 6)
      .attr("dy", "0.71em")
      .attr("text-anchor", "end")
      .text("Frequency");

  g.selectAll(".bar")
    .data(data)
    .enter().append("rect")
      .attr("class", "bar")
      .attr("x", function(d) { return x(d.letter); })
      .attr("y", function(d) { return y(d.frequency); })
      .attr("width", x.bandwidth())
      .attr("height", function(d) { return height - y(d.frequency); });
});
```

`x.domain` and `y.domain` are used to scale the x and y axes. `d3.max` and `d3.min` can be used to find the max and min in a dataset, providing another anonymous function similar to python lambda functions. `g.append('g')` calls are to place the axes in the correct location and add labels. You can see the use of `.attr` to give attributes. `.text` gives labels.

`g.selectAll('bar').data(data).enter()` is where the data join happens. You can see we append a rectangles to the SVG with class `.bar`. The x value of the rectangle is mapped using a function to the discreet value on the x axis and the y value is determined by the frequency. We need to use the height as `height - y(d.frequency)` because the height starts at 0 from the top of the SVG.

Here is the resulting graph.

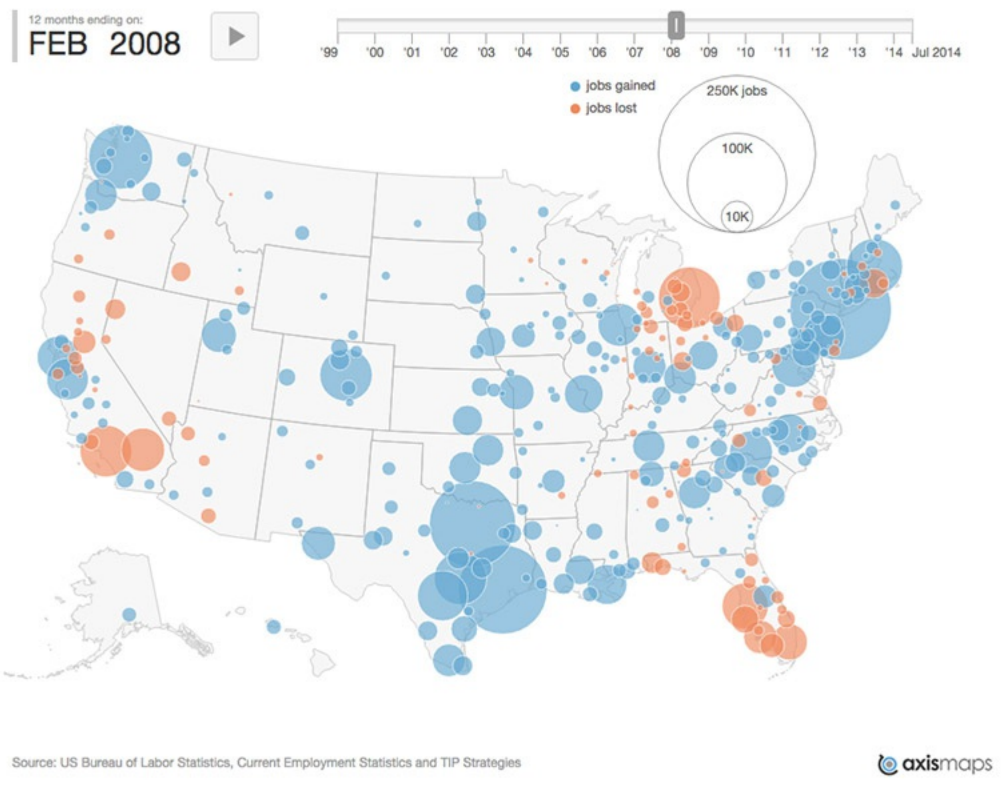


Further examples

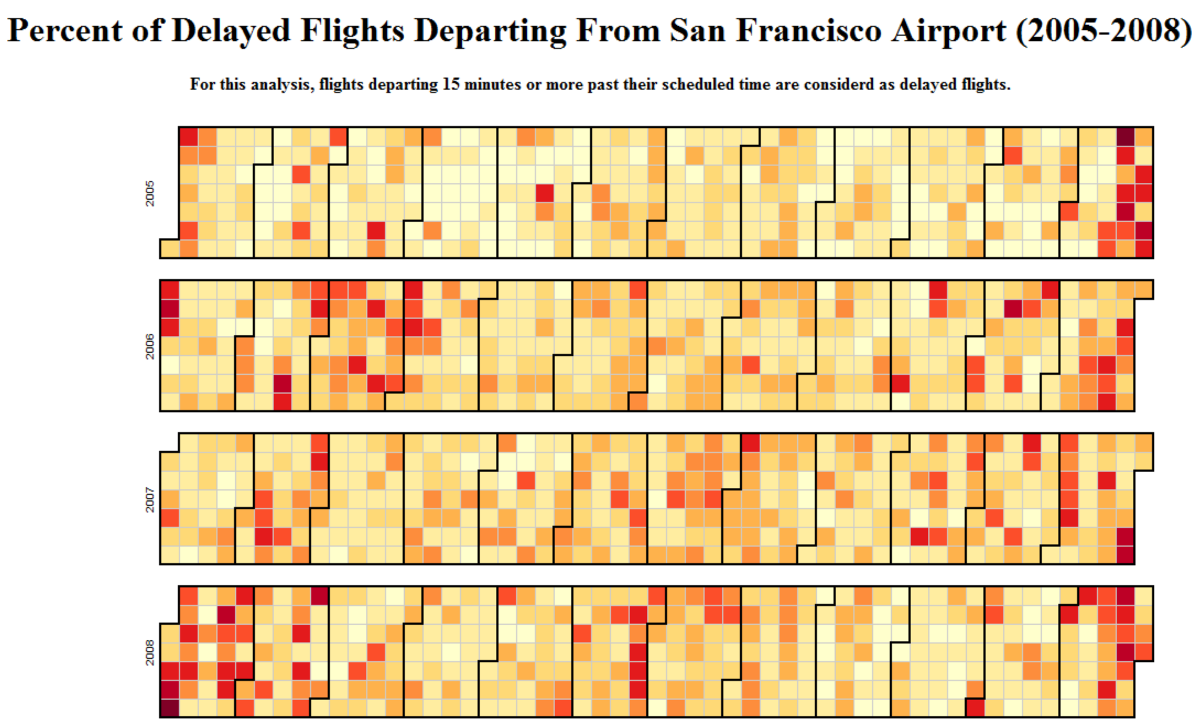
There are many more examples of graphs that can be created with D3. Of course there are basic line charts, pie charts and bar charts, but with the use of SVG and databinding, many different types of visualizations can be drawn. The only limit is your imagination.

Here are a couple

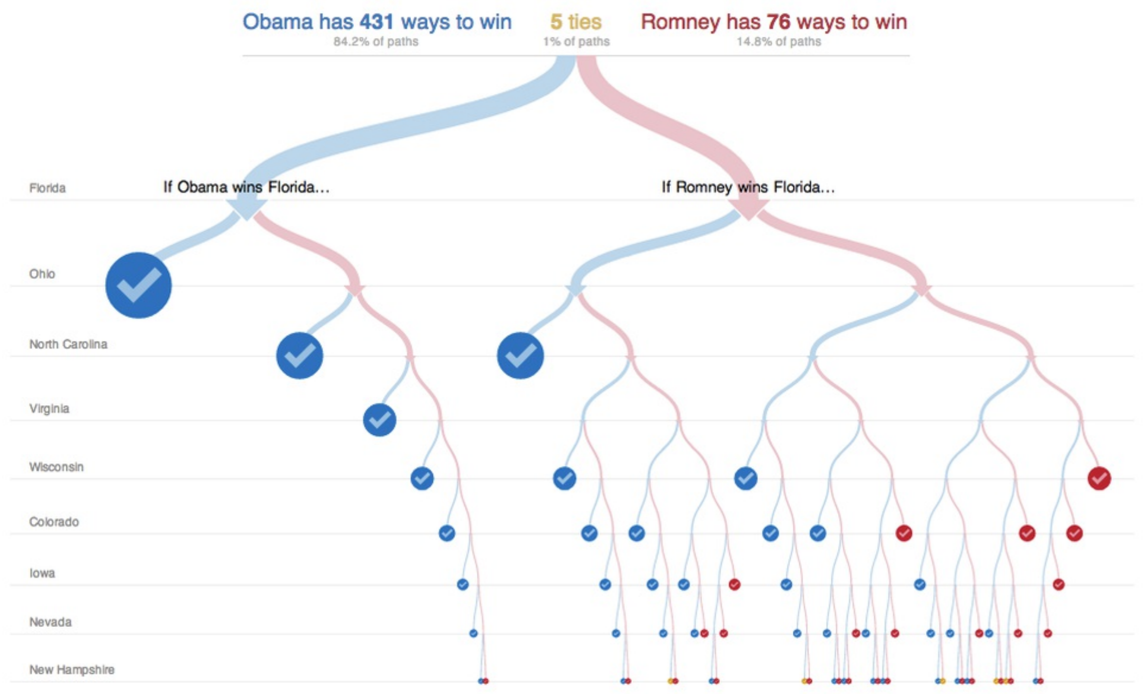
Geo data example


12 months ending on:
FEB 2008
'99 '00 '01 '02 '03 '04 '05 '06 '07 '08 '09 '10 '11 '12 '13 '14 Jul 2014
jobs gained
jobs lost
250K jobs
100K
10K
Source: US Bureau of Labor Statistics, Current Employment Statistics and TIP Strategies
axismaps

Calendar data example



Tree based data example



Final Message

Data visualization is a very important tool for statisticians and computer scientists. We need data visualization to be able to quickly convey a message in a large dataset in an elegant manner. Today we use software to aid us in creating beautiful visualizations. One of the biggest data visualization software packages is `d3.js`. D3 is a great visualization tool because of the flexibility in creating different types of visualizations and the power it gives you with data joins. It is also great because it is written in javascript, making it the preferred tool to visualize data on the web.

References

https://en.wikipedia.org/wiki/D3.js

https://d3js.org/

http://www.informationisbeautiful.net/visualizations/what-makes-a-good-data-visualization/

http://searchbusinessanalytics.techtarget.com/definition/data-visualization

https://website.education.wisc.edu/~swu28/d3t/index.html

https://www.dashingd3js.com/why-build-with-d3js

http://vis.stanford.edu/files/2011-D3-InfoVis.pdf