**Data Visualization Resources**

[**http://www.thefunctionalart.com/**](http://www.thefunctionalart.com/)

[**http://visualisingdata.com/**](http://visualisingdata.com/)

[**http://www.storytellingwithdata.com/**](http://www.storytellingwithdata.com/)

[**http://helpmeviz.com/**](http://helpmeviz.com/) -submit your vis and get feedback

[**http://d3js.org/**](http://d3js.org/)

[**http://www.tableausoftware.com/public/**](http://www.tableausoftware.com/public/)

[**http://raw.densitydesign.org/**](http://raw.densitydesign.org/)

[**http://www.adobe.com/products/illustrator.html**](http://www.adobe.com/products/illustrator.html)

<http://www.perceptualedge.com/images/Effective_Chart_Design.pdf>

<http://extremepresentation.typepad.com/files/choosing-a-good-chart-09.pdf>

<https://bost.ocks.org/mike/bar/>

**https://bost.ocks.org/mike/**

http://datajournalismhandbook.org/1.0/en/getting\_data\_0.html

The placement of the circle on the x and y axis was determined by the type of scale used to describe the data (linear or log), the domain values (max and min values of data) and range (max and min pixel values available on each axis). We inverted the range values for the y axis to account for the fact that the top of the y axis has a pixel value of 0. The max value extends down the y-axis rather than up the y-axis.

Here are the steps you took to accomplish this in the JavaScript console while viewing a course viewer page in Chrome. Note that the quotes used in expressions are straight, plain quotes (text-only) not curly, smart quotes (rich text format).

1. Paste contents of d3.min.js file into console.
2. File can be found [**here**](https://d3js.org/d3.v3.min.js).
3. Expected Output: True
4. Clear content from div in course viewer: Input: d3.select('.main').html('');
5. Expected Output: >[Array[1]]
6. Define svg variable: Input: var svg = d3.select('.main').append('svg')
7. Expected Output: undefined
8. Assign y axis linear scale to y variable. This axis describes life expectancy. Note that while the height of the svg is 300 pixels, only 250 are used for the y axis to leave a buffer. Also, for the y axis the max value comes first because of a quirk with how objects are drawn in the browser: highest value at bottom of axis. Input: var y = d3.scale.linear().domain([15,90]).range([250,0]);
9. Expected Output: undefined
10. Assign X axis scale to x variable. This axis describes annual income. Input: var x = d3.scale.log().domain([250,100000]).range([0,600]);
11. Expected Output: undefined
12. Assign radius scale to r variable. The radius corresponds to the square root of the population. Input: var r = d3.scale.sqrt().domain([52070, 1380000000]).range([10, 50]);
13. Check scaling with console.log by plugging in life expectancy for China in y variable, and annual income per person for China in x variable, and population for China in r variable. Input: console.log(y(77), x(13330), r(1380000000));
14. Expected Output appox: 43.33333333333314 398.1976156961321 50
15. Append circle with attribute values for radius, fill color, center x and center y of circle: Input: svg.append('circle').attr('r', r(1380000000)).attr('fill','red').attr('cx', x(13330)).attr('cy', y(77));
16. Expected Output: >[Array[1]]
17. Look at placement of the circle in the svg, and compare with the placement in the original Gapminder World graph: [**http://www.gapminder.org/world/**](http://www.gapminder.org/world/)

## Grammar of Graphics

The Grammar of Graphics is a visualization theory developed by [**Leland Wilkinson**](http://en.wikipedia.org/wiki/Leland_Wilkinson) in 1999 with the publication of the eponymous [**book**](http://www.springer.com/statistics/computational+statistics/book/978-0-387-24544-7).

It is quite an extensive theory which has influenced the development of graphics and visualization libraries alike (including D3 and its precursors), but in this class you will focus on 3 of its key principles:

1. Separation of data from aesthetics
2. Definition of common plot/chart elements
3. Composition of these common elements

### Separation of Concerns

You just saw some of the benefits of separating the data from the visual presentation of that data in the previous videos. The main take-aways are:

* Independently transform data and present data
* Delegate work and responsibilities
  + Engineer focuses on data manipulation
  + Designer focuses on visual encoding of data
* Present multiple visual representations of a dataset
  + Ex: Bubble chart and line chart show [**different**](http://dc-js.github.io/dc.js/) facets of a dataset.

### Common Elements

When thinking about creating a chart or graphic, it is often helpful to visually decompose what you want to achieve. In previous videos you saw how to abstract a chart into more basic visual encodings. In the map example, you saw that a choropleth is a combination of geography and color while a cartogram is a combination of geography and size. When talking about **composable** elements, a few of the most common are:

* Coordinate System (cartesian vs. radial/polar)
* Scales (linear, logarithmic, etc.)
* Text annotations
* Shape (lines, circles, etc.)
* Data Types (Categorical, Continuous, etc.)

### Composition

The beauty of the Grammar of Graphics surfaces when you combine these common components. For example, you can create a bar chart by mapping a value in the data to the height of the bar in cartesian space, but you can also can also map these values in polar coordinates, in which the data value corresponds to the radial degree of a slice, to get a pie chart.

* Categorical + Continuous x Cartesian = Bar Chart
* Categorical + Continuous x Polar = Pie Chart
* Continuous + Continuous x Cartesian = Scatter Chart

And you can create a plethora of other charts by combining these common components in different ways. How might you achieve a line plot with a logarithmic scale from these common components?