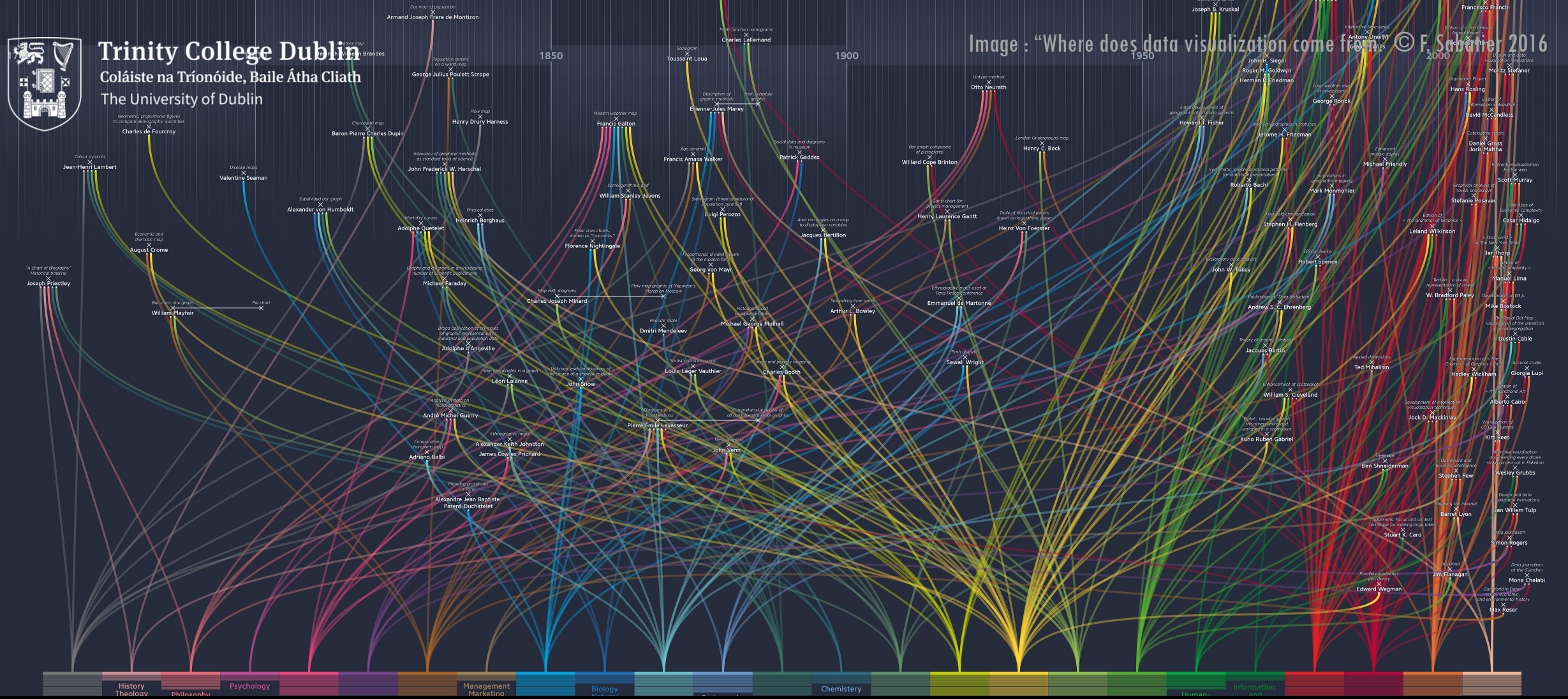
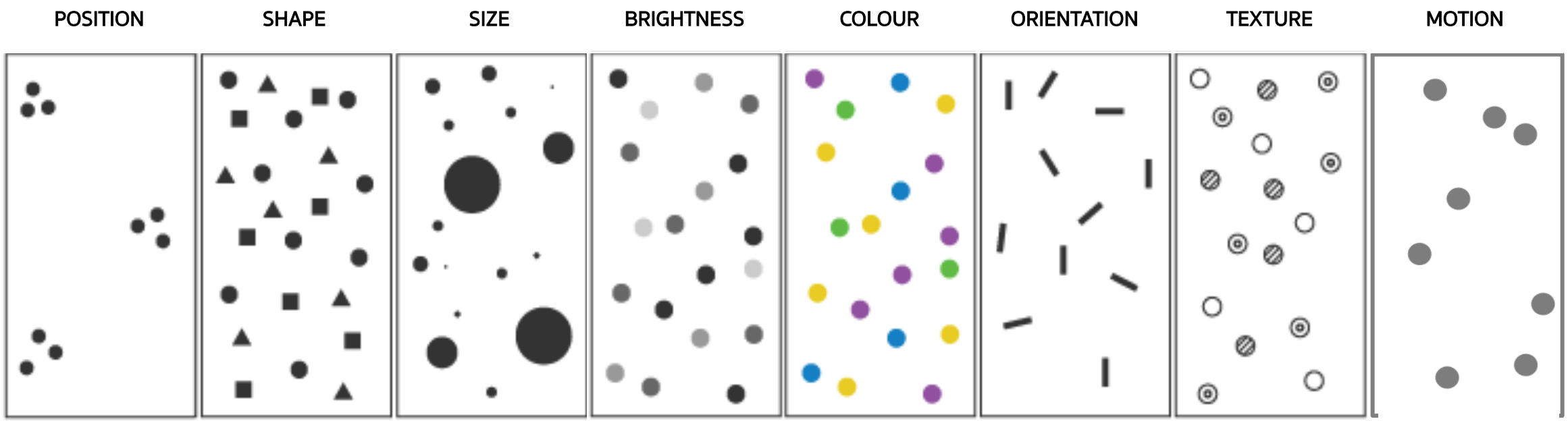


Visual Encoding – Part 2

29/09/2023

Recap: Visual Encoding Channels



The Eight Visual Variables (Ward et al 2010)

Graphics primitives used to depict data in visualisation

Which one to use?

Summary Encoding Channel Characteristics

Based on [Ward et al 2014]

ENCODING CHANNELS

CHARACTERISTICS

	POSITION	SHAPE	SIZE	BRIGHTNESS	HUE	ANGLE	TEXTURE	MOTION
SELECTIVE	✓	✓	✓	✓	✓	≈ (somewhat)	✓	✓
ASSOCIATIVE	✓	≈ (somewhat)	✓	✓	✓	≈ (somewhat)	✓	✓
ORDINAL	✓	✗	✓	✓	✗	≈ (somewhat)	✗	✓
QUANTITATIVE	✓	✗	✓	≈ (somewhat)	✗	≈ (somewhat)	✗	✓
RANGE	High	High	High	Limited	Limited	Limited	High	Limited

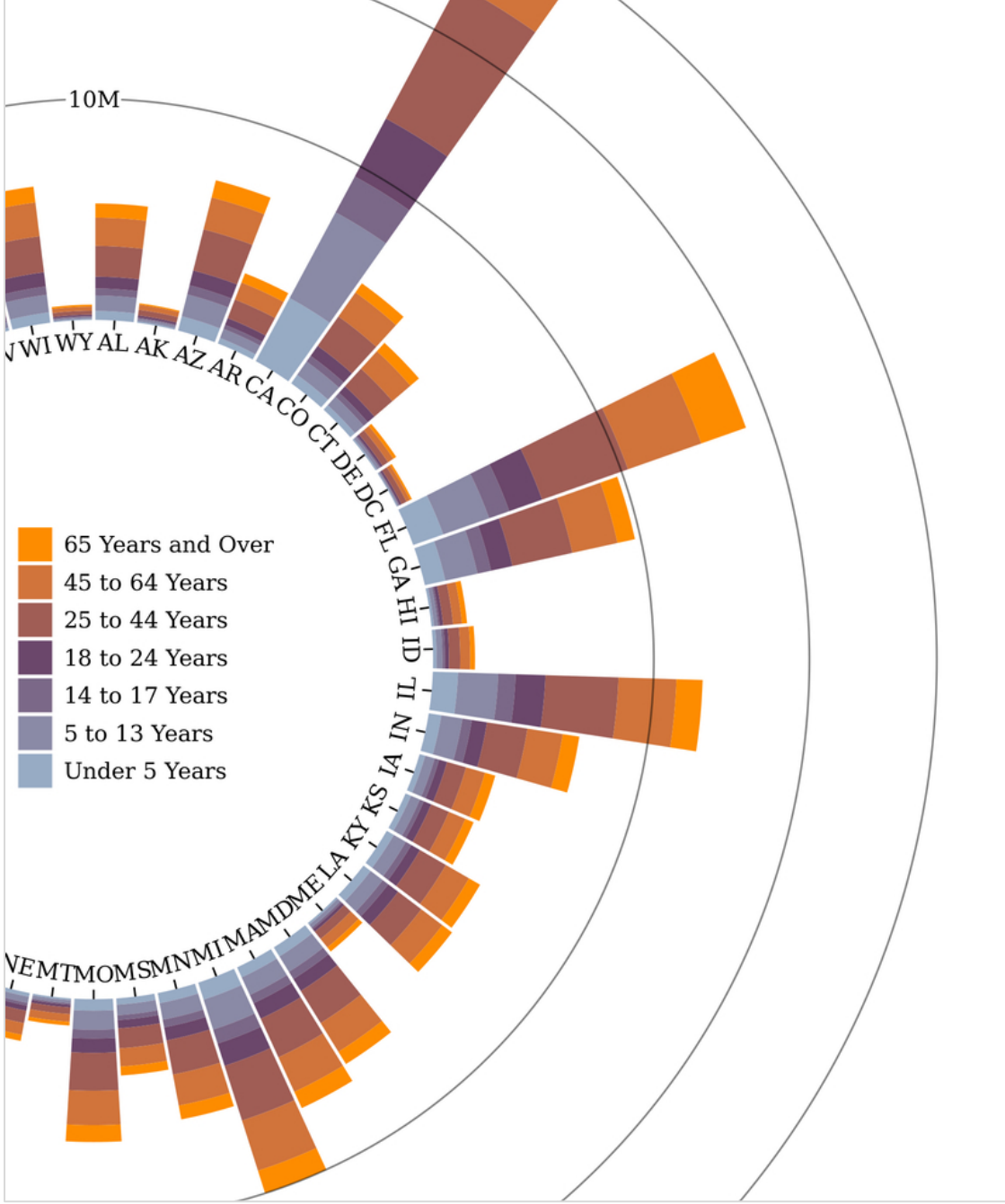
Effectiveness

If multiple channels are applicable, which one is "best"?

Effectiveness Principle: when there are multiple data attributes and we need to encode them with several visual channels, the importance of the data attribute should match the ranking of the channel used to express it.

How to rank Channels [Munzner 2014]:

- Accuracy: how precise can a value be read
- Discriminability: essentially range [discussed previously]
- Separability: degree to which channels impact each other
- Pop-out: ability to enhance one element over another
- Grouping: strength at which associations can be depicted



Accuracy [Stevens 1961]

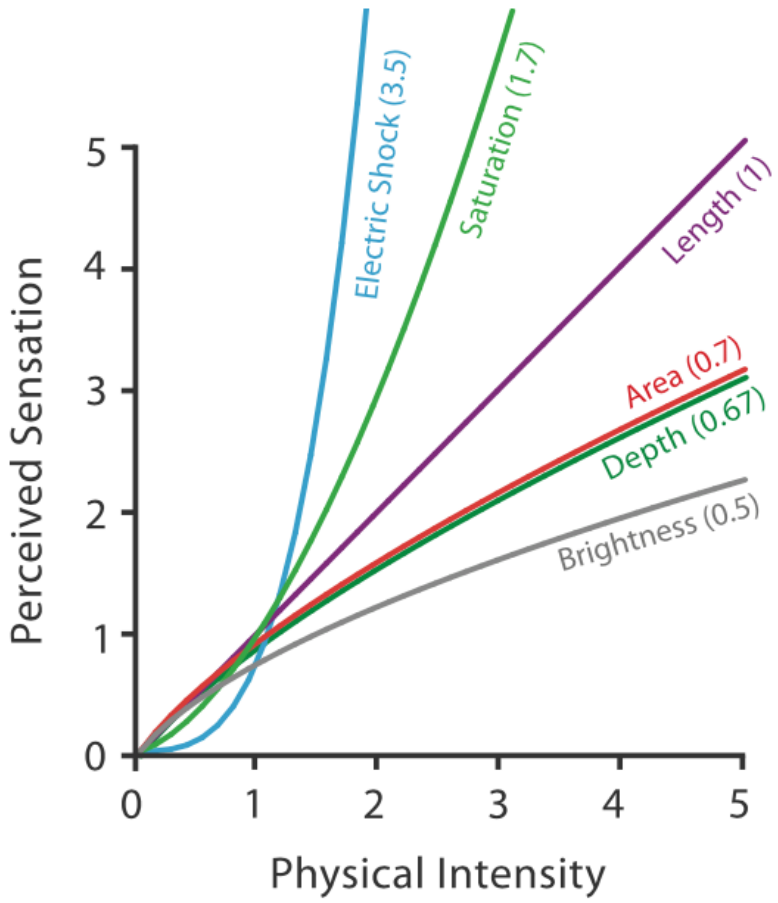


Image from Munzner (2014) after Stevens [1961]

Scientists have extensively studied how how strong is the perceived sensation is as the intensity of the input increases in a particular sensory modality. Some of these provide clues to the relative effectiveness of visual encoding channels.

Steven's Power Law: relationship between magnitude of stimulus I vs. its perceived intensity ψ:

$$\psi(I) = I^a$$

a is an exponent that depends on the type of stimulus

https://en.wikipedia.org/wiki/Stevens%27s_power_law

Accuracy [Munzner 2014]

Magnitude Channels:

- for Ordinal and Quantitative attributes, typically associated with “How much” tasks.
- require intuitive ordering, range

Identity Channels:

- for categorical attributes, associated with “What”, “Where” tasks.
- require selective/Associative accuracy

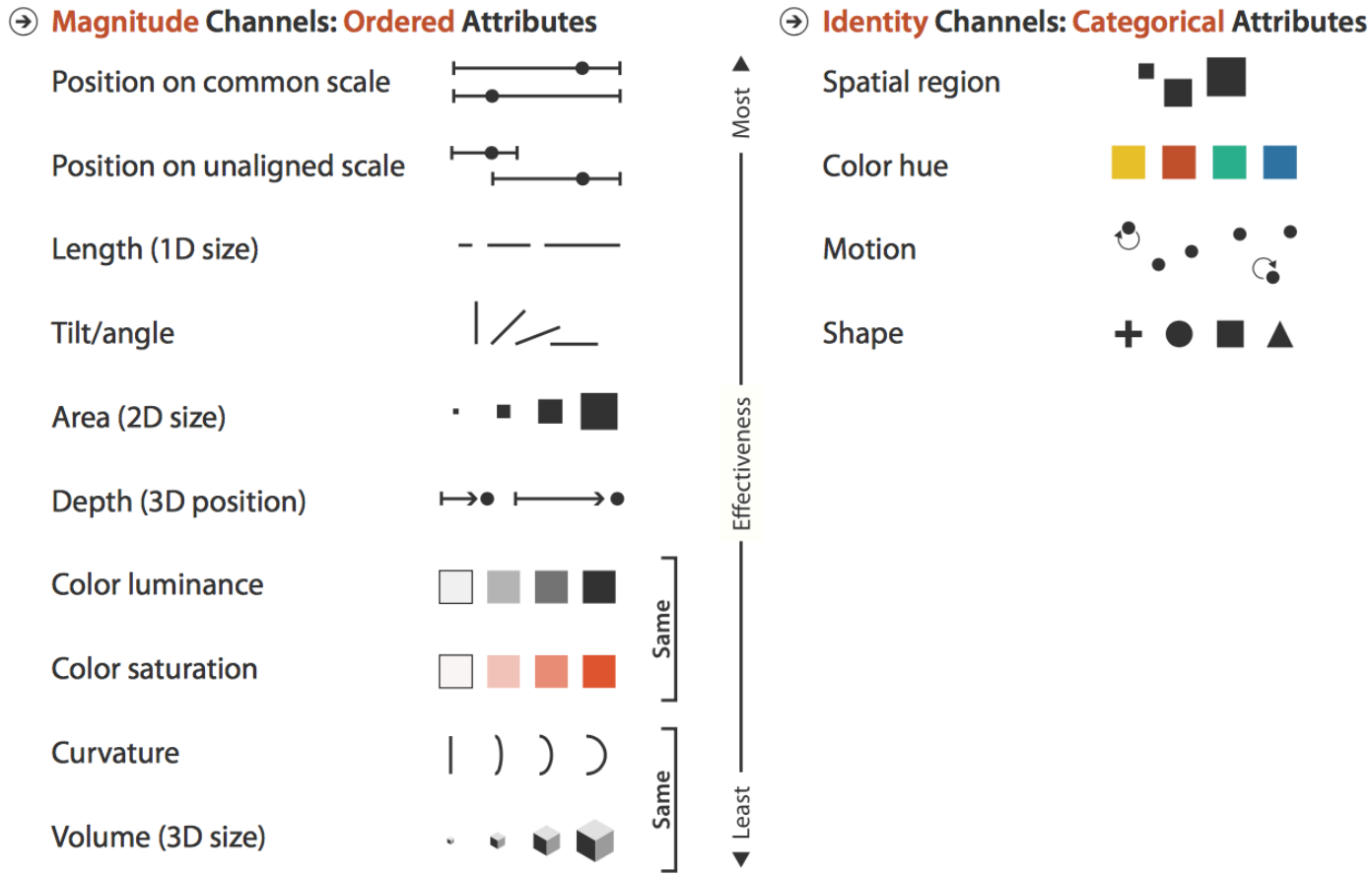
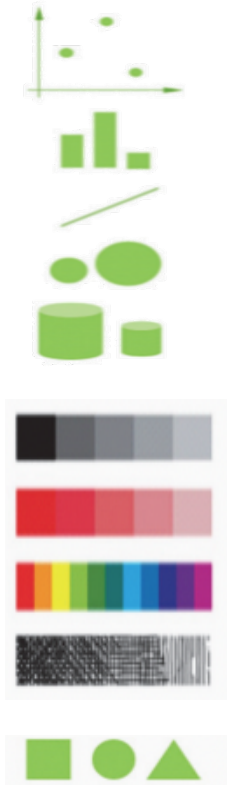
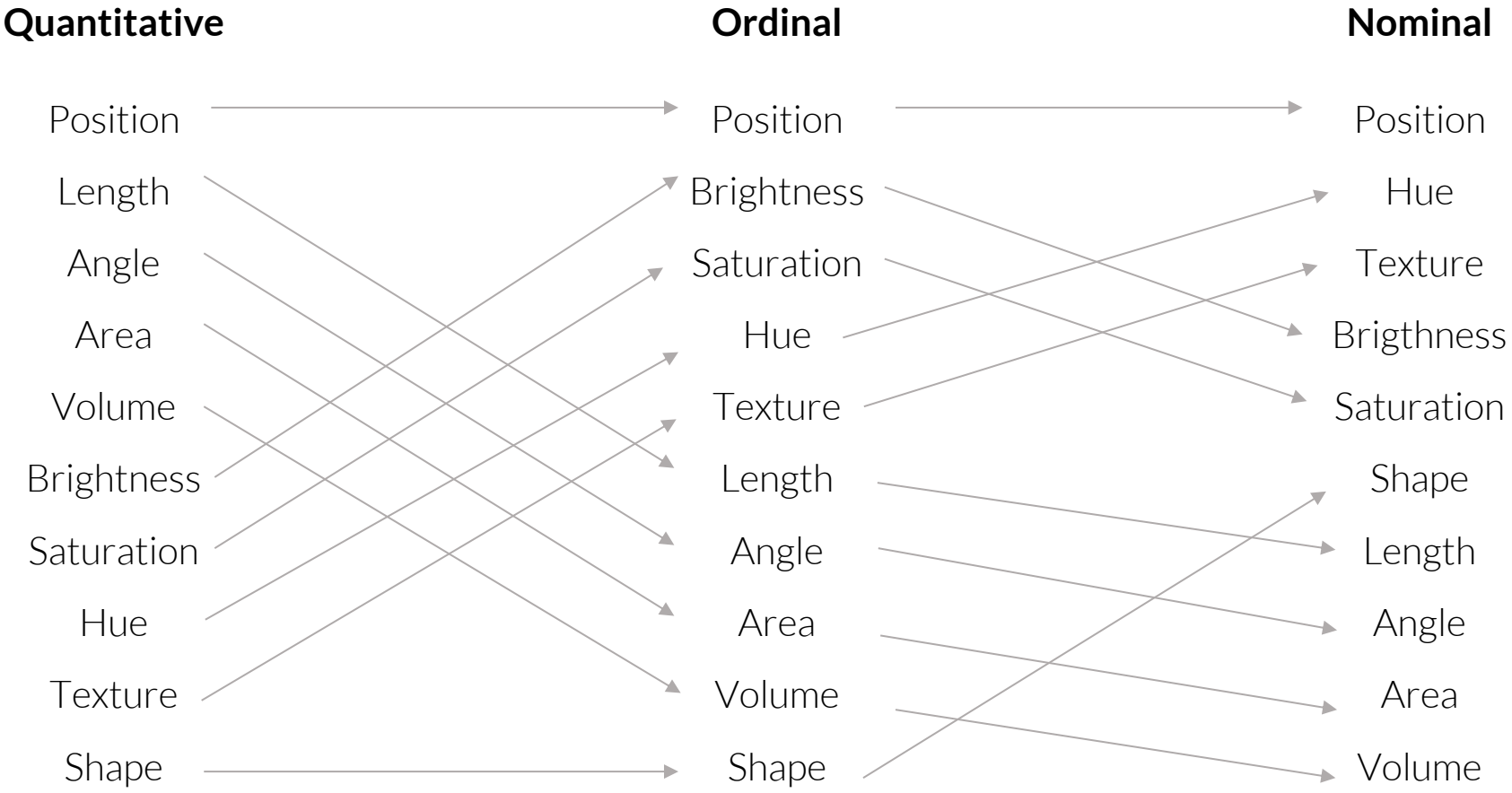


Image from Munzner (2014)

Accuracy [Mackinlay 1986]

Amongst the more often cited works, in the specific context of Visualization are the studies of Mackinlay





Note: (1) The term “Nominal” in Mackinlay’s paper is synonymous to what we have previously described as the categorical (or qualitative) attribute type. (2) The arrows are merely to highlight how the rankings change across different attribute types

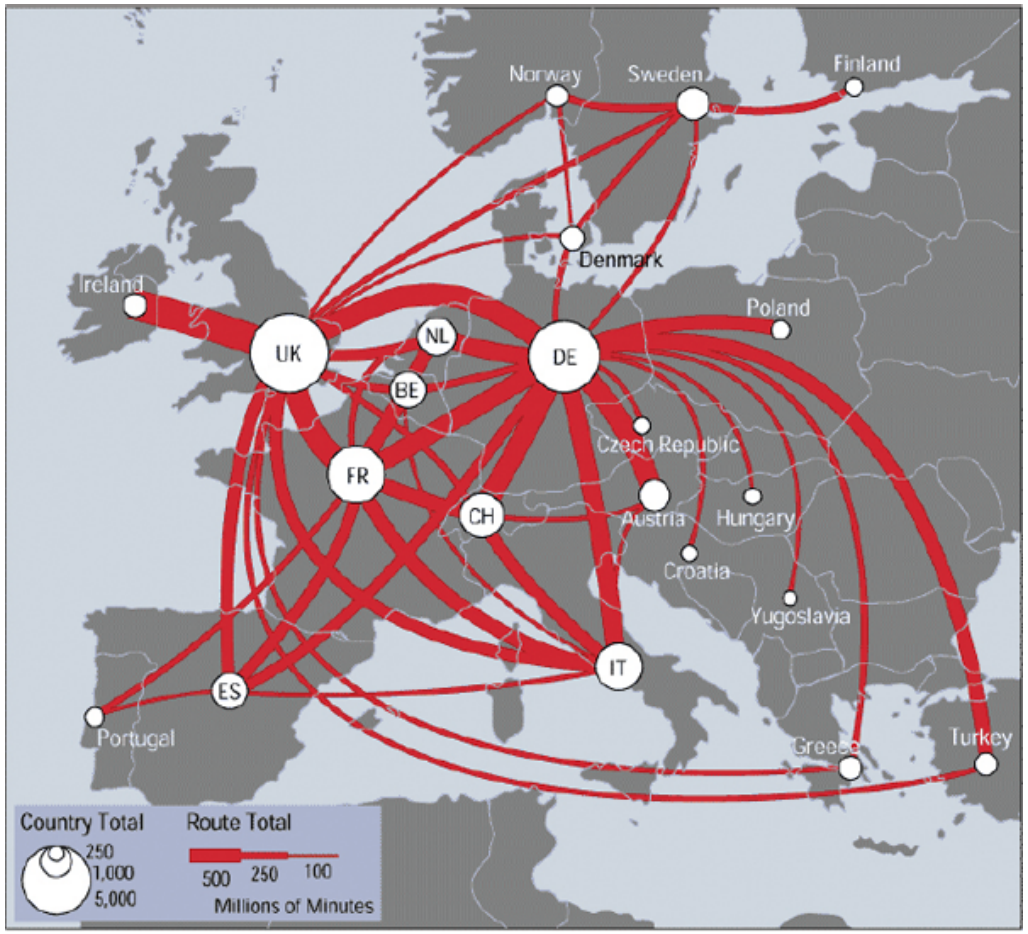
Discriminability/Range

For a given visual channel, are differences between items perceptible as intended?

- Range of the channel: the number of bins of a distinguishable step or level from the other.
- number of different values that need to be shown must not be greater than the number of bins available

Sometimes it is dependent on specifics the chart type

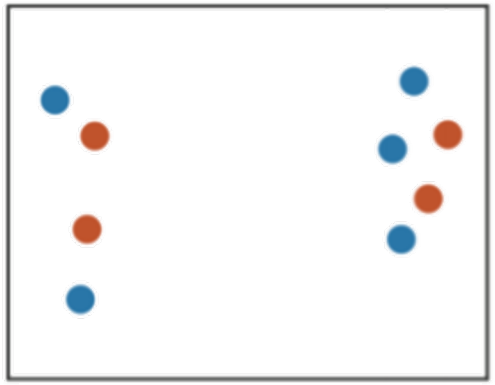
- e.g. size is thought to have fairly good range (many different sizes can be used), but changing line width in something like a sankey chart only works for a fairly small number of steps. Increasing further leads to line being perceived as an area.



Separability Of Multiple Channels

- Visual channels sometimes have dependencies and interactions with others.
- Some channels are very independent whilst others are inextricably related (may lead to unanticipated combinations being perceived)

Position + Hue



Fully separable

Size + Hue

Some interference; size of objects may lead to perceived shift in hue or brightness

Width + height

Interference; objects appear wider/narrower depending on their height and vice versa

Red + Green

Major interference: the degree of redness or greenness of some points is very unclear

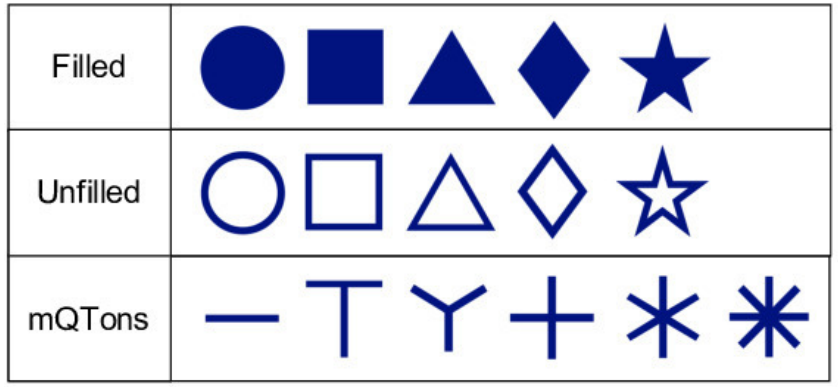
Image from [Munzner 2014]

Separability of Channels

- Shape affects colour difference perception: e.g., colours more discriminable with filled shapes
- Colour has some effect on shape perception: e.g., very light colors affect perceived shape, against a white background
- Shape significantly affects size perception:
  - Filled shapes perceived as larger than unfilled.
  - Shapes that contain more visual density on the top or bottom of the shape (e.g., T, ■, □) appear relatively bigger (than e.g., +)
- Size has some effect on shape perception: extremely small sizes affect shape perception
- Thin shapes/small sizes may affect texture, brightness and hue perception
- Texture recognition is somewhat affected by orientation of textures
- Sizes of objects may affect their perceived speed of motion
- Fast motion makes it difficult to process details e.g. texture patterns

For related reading, see [Smart and Szafir 2019]. However, some added points in this slide are based on conventional wisdom

Image from [Smart and Szafir 2019].



	Motion	Texture	Angle	Hue	Brightness	Size	Shape
Position	⚠	✅	✅	✅	✅	✅	✅
Shape	✅	⚠	⚠	⚠	⚠	⚠	
Size	⚠	⚠	⚠	⚠	⚠		
Brightness	✅	⚠	✅	⚠			
Hue	✅	⚠	✅				
Angle	✅	⚠					
Texture	⚠						

Note: intensity of effect varies. Furthermore, use of sub-channels can improve separability e.g,. x/y position, width/height

Grouping

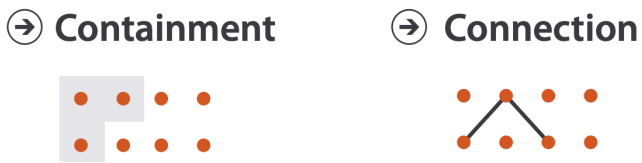
Similar to previously discussed *associative-ness* criteria of [Ward 2014]

Perceptual grouping can arise from the implicit characteristics of the encoding channel e.g.

- proximity i.e. placing items within the same spatial region
- similarity of hue
- similarity of shape
- similarity of motion (N.B. multiple levels of motion may overwhelm capacity for selective attention)

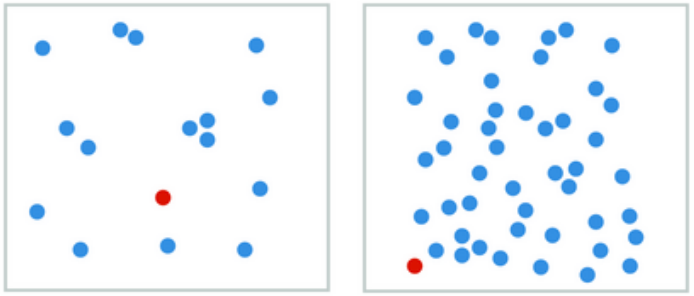
However, in some cases this can be explicitly encoded e.g.

- using containment, child node inside the enclosed area of its parent
- using connection i.e. link marks / connector
- often these are more effective, however it is at the cost of clutter



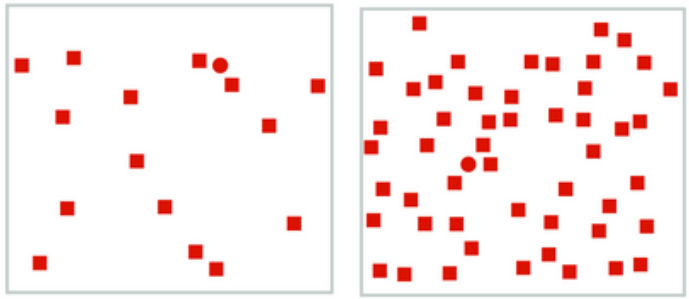
Pop-out

Property whereby a distinct item stands out from many others immediately.

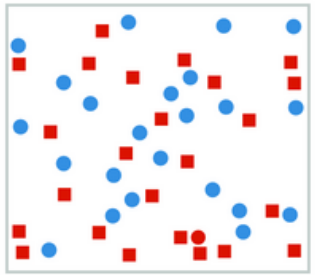


Strong effect: discovery independent of complexity

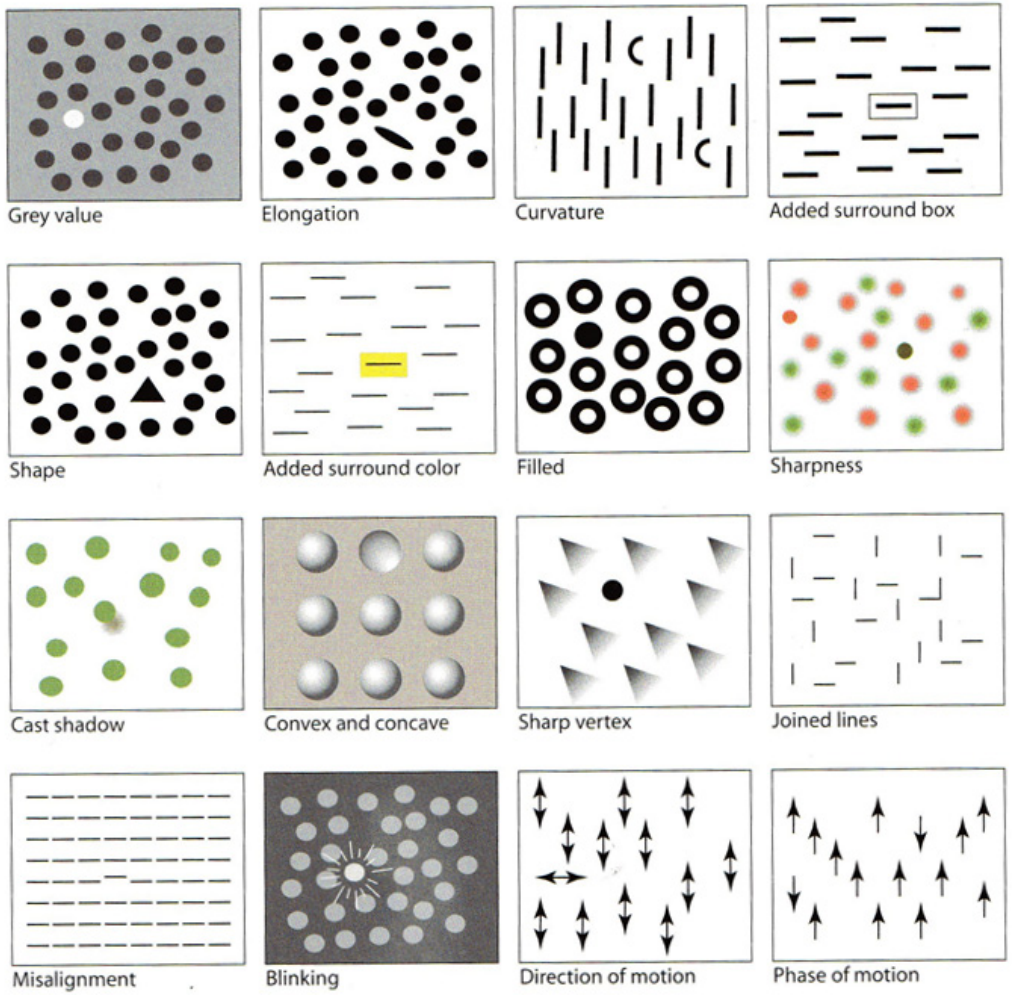
Based on low-level, highly parallel processing: time taken to distinguish objects does not depend on number of objects.



Less effective



Effects cannot always simply be combined. Some effective pairs: are space and color, motion and shape.



Most visual channels provide visual pop-out. Effectiveness depends on both the channel and how different the target item is from its surroundings.

Recommended Readings

- Chapter 1 – "Foundations for a science of Data Visualization" in **Information Visualization (4th Edn.)**, Colin Ware 2019
  - single chapter excerpt available at (click on preview):
    - https://www.ebooks.com/en-ie/book/209920967/information-visualization/colin-ware/?_c=1
    - http://www.ifs.tuwien.ac.at/~silvia/wien/vu-infovis/articles/book_information-visualization-perception-for-design_Ware_Chapter1.pdf

- Chapter 5 – "Marks and Channels" in **Visualization Analysis and Design**, Tamara Munzner 2014

  [Available as e-book in Library Reading Rooms]

  [A free low-quality pre-publication draft of is available on Author's web page: https://web.cse.ohio-state.edu/~machiraju.1/teaching/CSE5544/ClassLectures/PDF-old/book.120803.pdf#page=88]

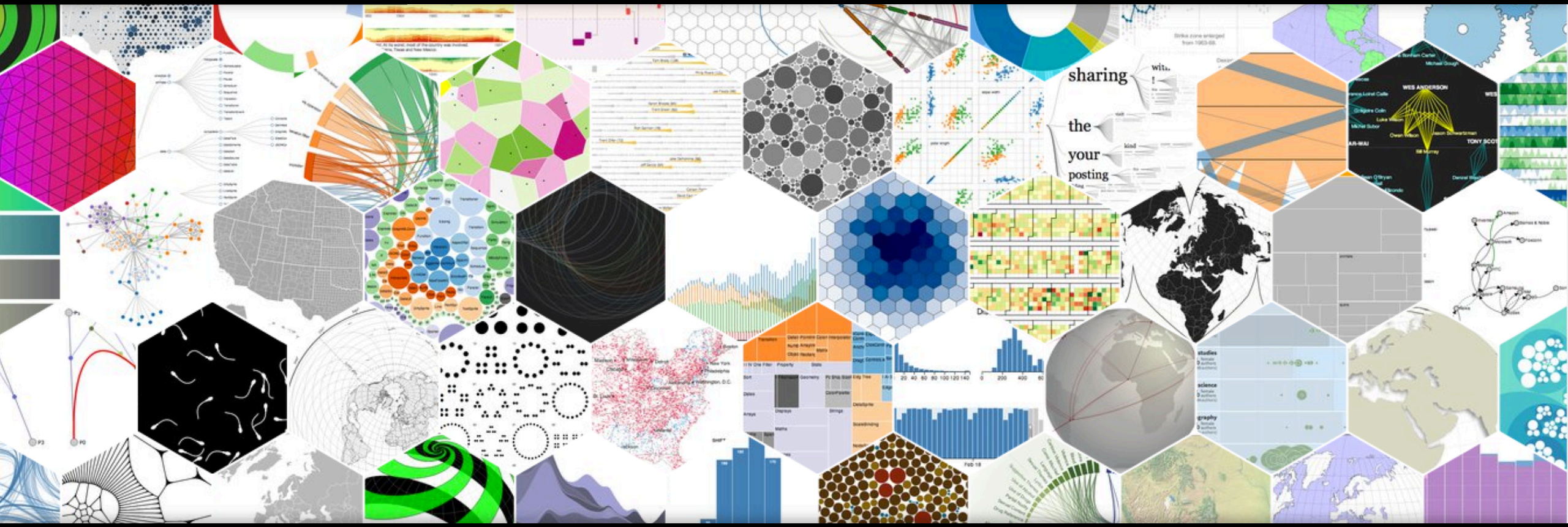
- Chapter 4 – "Visualization Foundations" in **Interactive Data Visualization**, Ward et al 2010 [Available as e-book in Library Reading Rooms]

Other References [relevant studies]

[Mackinlay 1986] Jock Mackinlay. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2):110-141, 1986.]

[Smart & Szafir 2019] S Smart and DA Szafir. Measuring the Separability of Shape, Size, and Color in Scatterplots. CHI 2019

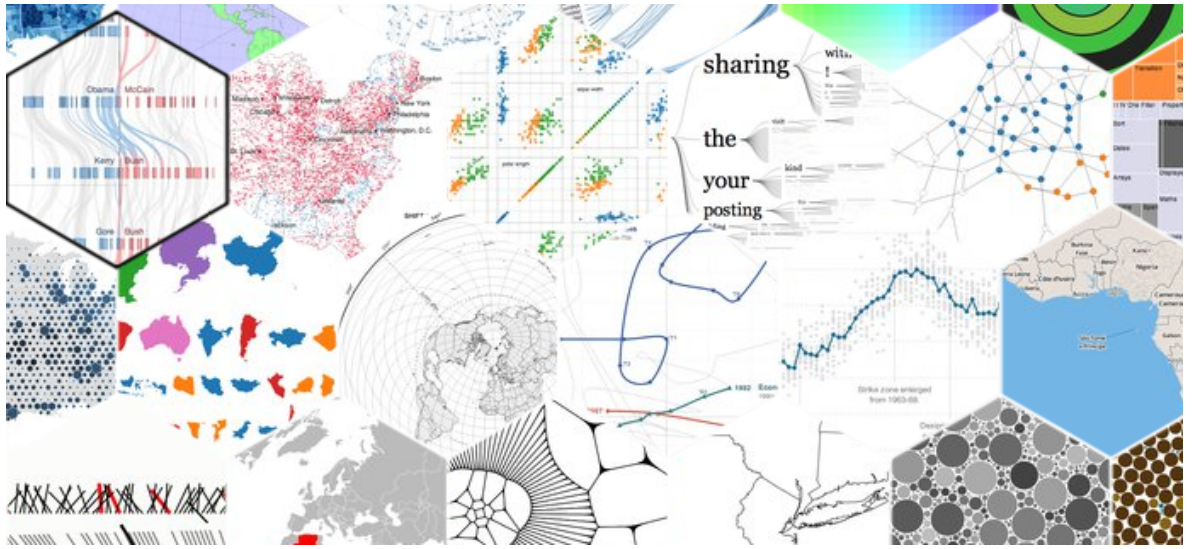
[Stevens 1961]. SS Stevens. The psychophysics of sensory function. In: Rosenblith WA, editor. Sensory communication. Cambridge, MA: MIT Press; pp. 1–34.



Vis TooIS#2 : d3.js

Data-Driven Documents (d3.js)

- a javascript library that allows you to bind data to browsers (DOM) and apply transformations to the document e.g. to create interactive visualizations with transitions and interactions.
- Not a graphics language per se, but a library of utility functions for visualization based directly from web standards: HTML, SVG, and CSS
- provides a more flexible, efficient, dynamic, fast alternative to doing things directly through these standards



More info: https://d3js.org/

Gallery:

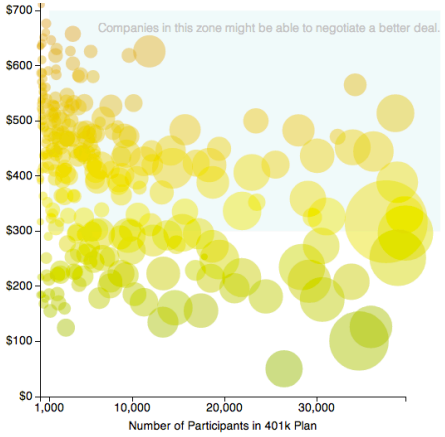
- https://www.d3-graph-gallery.com/
- https://github.com/d3/d3/wiki/Gallery

[slightly more complex examples]

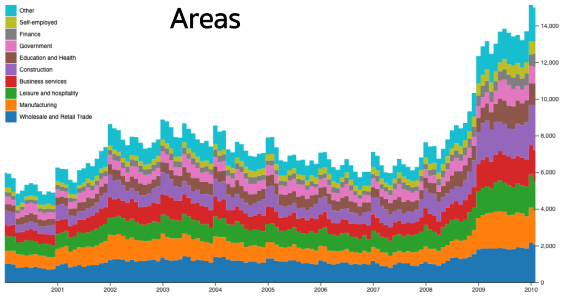
D3 Graphical Functions

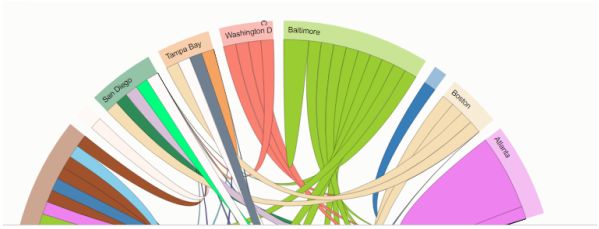
- Axes
- Brushes
- Chords
- Colors, Color Schemes
- Contours
- Dragging
- Easings
- Forces
- Geographies (Paths, Projections, Spherical Math, Spherical Shapes, Streams, Transforms)
- Paths
- Polygons
- Quadtrees
- Shapes (Arcs, Pies, Lines, Areas, Curves, Links, Symbols, Stacks)
- Transitions
- Voronoi Diagrams
- Zooming

Brush and Axes

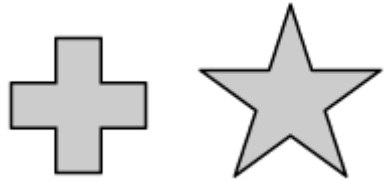


Areas

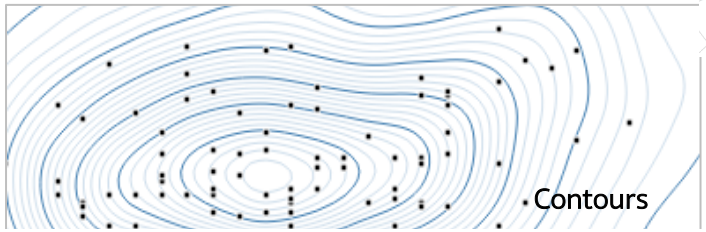




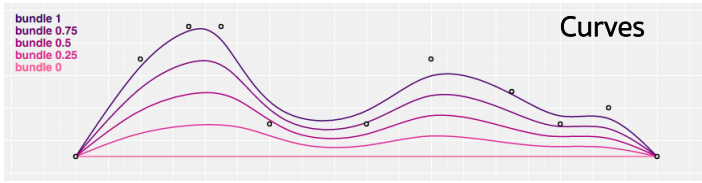
Chords and Colours



Symbols

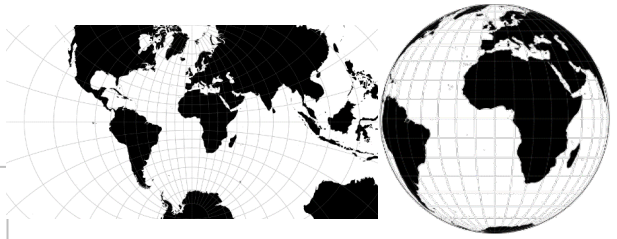


Contours

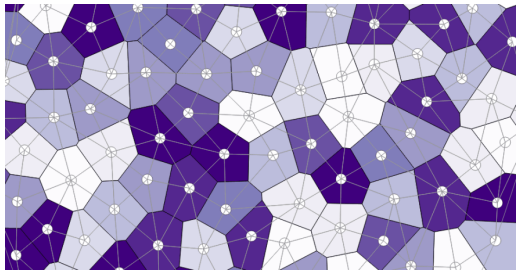


Curves

Spherical Shapes & Projections

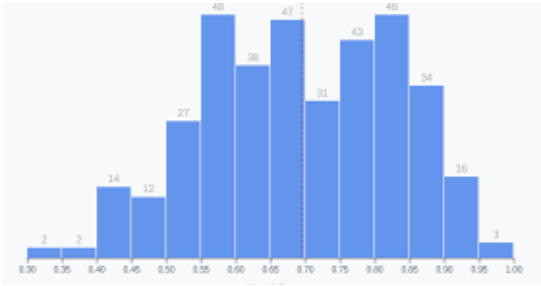


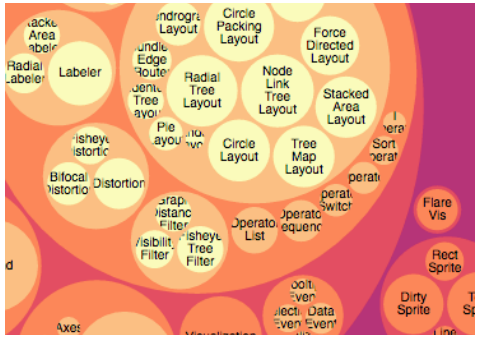
Voronoi Diagrams

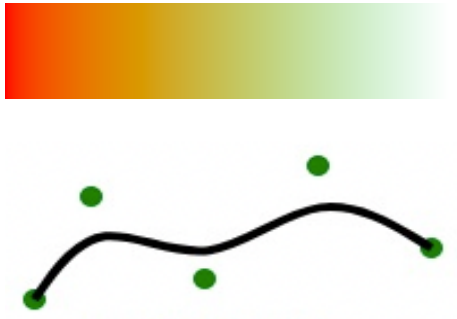


D3 Data Management Functions

- Arrays (Statistics, Search, Transformations, Histograms)
- Collections (Objects, Maps, Sets, Nests)
- Dispatches
- Delimiter-Separated Values
- Fetches
- Number Formats
- Hierarchies
- Interpolators
- Random Numbers
- Scales (Continuous, Sequential, Diverging, Quantize, Ordinal)
- Selections (Selecting, Modifying, Data, Events, Control, Local Variables, Namespaces)
- Time Formats
- Time Intervals
- Timers







Elements of d3.js and why we need them

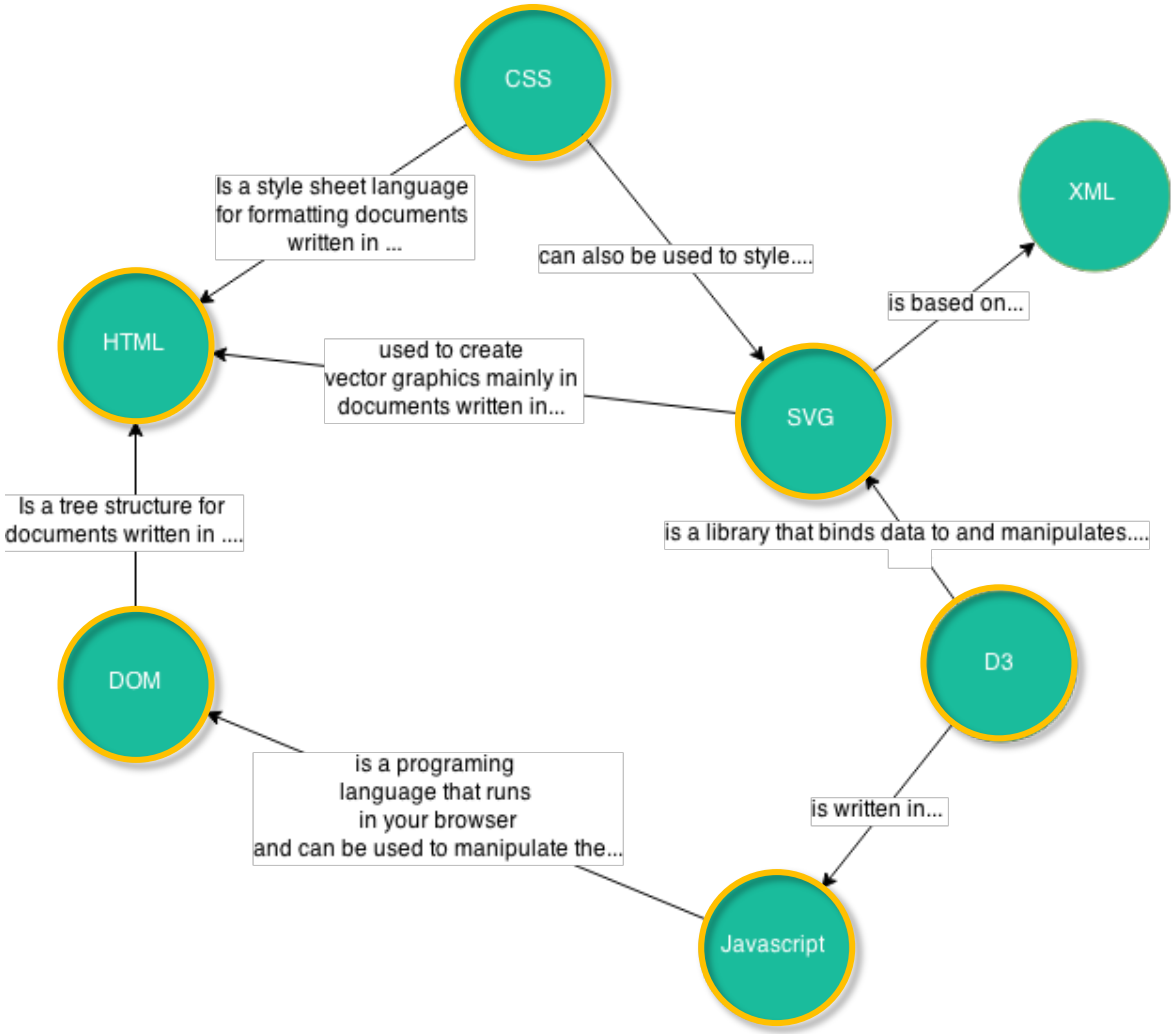


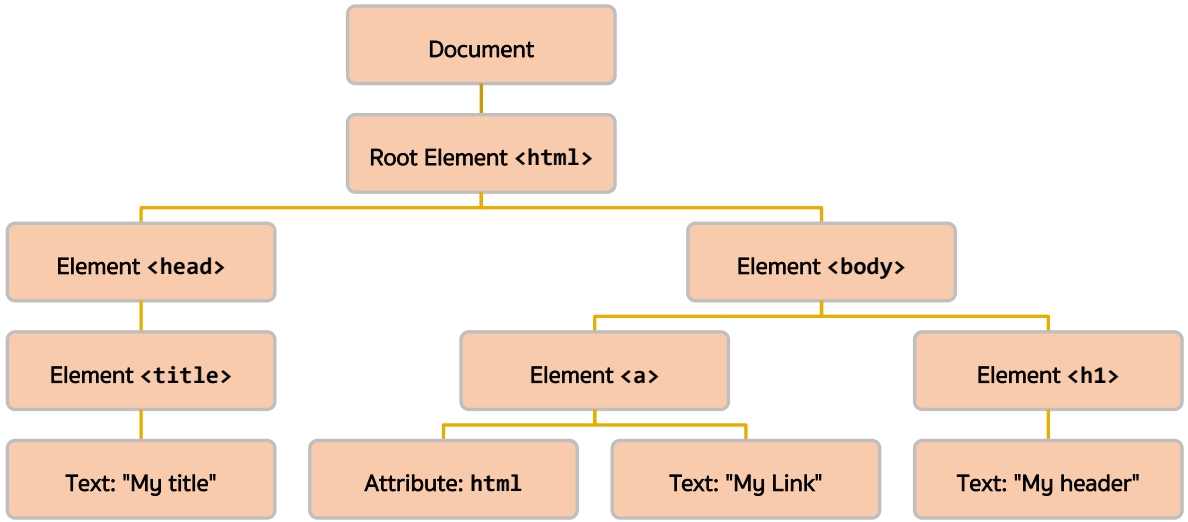
Image © Simon Raper

- DOM encodes the data structure and functionalities of a webpage accessed through HTML and JS
- Everything on a webpage is scripted in HTML at the top most level (i.e. wrapped inside some HTML)
- CSS used to define look and feel of the page e.g. colours, styles
- SVG defines standard vector graphics, the basic building blocks in d3js for creating graphical objects
- Javascript is a basis for coding any dynamic elements and programmatic elements, loops, classes, functions, variables. JS used to generate HTML/CSS code dynamically (there are also other alternatives e.g. php)
- d3.js is a javascript library used to provide more generalized and high-level functions to create visualizations

Document Object Model (DOM)

Document Object Model

- cross-platform and language-independent application programming interface for webpages that run in your browser
- essentially a data structure that holds all the elements of a webpage e.g. graphics, link, text blocks, form
- These are …
  - created with HTML
  - styled with CSS
  - and can be altered at run-time using javascript



Short intro

- https://www.w3schools.com/js/js_htmldom_elements.asp

Full reference (you don't really need this):

- https://www.w3schools.com/js/js_htmldom_elements.asp

HTML

Hypertext Markup Language (HTML)

- Building blocks of web pages
- Mainly for defining content i.e., specifying the text, images, videos, forms, links that exist on a web page

- NOT really intended for
  - Look and feel of elements (see CSS instead)
  - Low level specification of visual elements (see SVG and D3js instead)

```
<html>
  <head>
     <title>Hello world</title>
  </head>
  <body>
     <p>Some basic paragraph text.</p>
     <a href="http://www.tcd.ie">A LINK</a>
     <img src="https://www.scss.tcd.ie/John.Dingliana/TCD-logo.png">
  </body>
</html>
```

TRY THIS AT HOME: Save this code in a text file with **.html** extension and open it in a browser

CSS

Cascading Style Sheets

- Language for scripting appearance of HTML pages
- **Layout** of elements e.g. images, blocks of text, etc.
- **Styling** of content e.g. text colour, font, position, borders, table appearance, fill colour

```
<html>
  <head>
    <title>Hello world</title>
  </head>
  <body>
    <p style="color:red; font-family:arial;
      font-style:italic; font-size:28px;">Some
       basic paragraph text.</p>
    <a href="http://www.tcd.ie">A LINK</a>
    <img style="width:300px;" src="tcdlogo.jpg" >
  </body>
</html>
```

Scalable Vector Graphics (SVG)

Vector-based open standard graphics definition in XML format

TRY THIS AT HOME: download this file and open it in a browser (to see the image) and text-editor (to see the data)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
   width="152.56285mm"
   height="225.93279mm"
<g
    id="layer1"
     transform="translate(-103.99722,-103.51612)">
    <path
       style="fill:#000000"
      d="m 456.07482,898.96824 c -2.16368,-4.1443 -7.03082,-
     11.2266 -10.81585,-15.7386 -11.09946,-13.2311 -
     12.43481,-18.1357 -11.02939,-40.511 0.91784,-14.6126
     0.49609,-22.9514 -1.67322,-33.0823 -1.59521,-7.4498 -
     2.9004,-17.8021 -2.9004,-23.0048 l 0,-9.4599
     …
     …
     2.53245,4.73185 -0.0432,4.09552 -5.54592,-1.36998 z"
   />
  </g>
</svg>
```

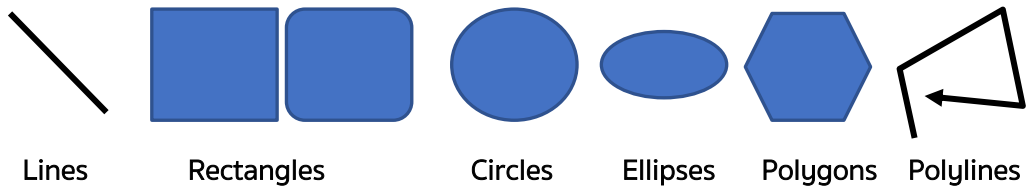
https://freesvg.org/marilyn-monroe-116834.

Scalable Vector Graphics (SVG)

Vector-based open standard graphics definition in XML format

Includes definitions of

- Basic shapes, Text, Color
- Styling attributes
- Coordinate systems and transforms
- Gradients and patterns



Reference: https://www.w3.org/TR/SVG11/

Simple Example within HTML:

```
<html>
<body>

<svg height="100" width="100">
  <ellipse cx="50" cy="50" rx="40" ry="40"
        style="fill:yellow; stroke:green;
        stroke-width:2" />
  Sorry, your browser does not support inline SVG.
</svg>

</body></html>
```

JavaScript

- High-level interpreted programming language
- ECMAScript specification
- Mainly for Client-side scripting
- Relatively easy to learn.
- Supports event-driven, functional, and imperative paradigms
- HTML and CSS are static definitions, JS provides means of programmatically creating HTML & CSS elements

Basic Tutorial:
https://www.w3schools.com/js/default.asp

```
var x, y, z;
x = 5; y = 6; z = x + y;

function myFunction(p1, p2){
  return p1 * p2;
}

var text = "The temperature is " + toCelsius(77)
        + " Celsius";

var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function()   {
    return this.firstName + " " + this.lastName;
  }
};

var cars = ["Saab", "Volvo", "BMW"];

var i;
for (i = 0; i < cars.length; i++)
{
  text += cars[i] + "<br>";
}
```

JavaScript

- High-level interpreted programming language
- ECMAScript specification
- Mainly for Client-side scripting
- Relatively easy to learn.
- Supports event-driven, functional, and imperative paradigms
- HTML and CSS are static definitions, JS provides means of programmatically creating HTML & CSS elements

Basic Tutorial:
https://www.w3schools.com/js/default.asp

```
<html>
   <head>
      <title>Hello world</title>
   </head>
   <body>
      <p style="color:red; font-family:arial;
      font-style:italic; font-size:28px;">Some basic
      paragraph text.</p>

      <a href="http://www.tcd.ie">A LINK</a>

      <img style="width:300px;" src="tcdlogo.jpg" >

      <script>
         for (i=0; i<10; i++){
           document.write("<h2>Hello World " +i+ "</h2>");
         }
      </script>
</body>
</html>
```

d3.js Example

From: https://www.tutorialspoint.com/d3js/

Direct SVG Example

```
<html>
<body>

<svg height="100" width="100">
  <ellipse cx="50" cy="50" rx="40" ry="40"
         style="fill:yellow; stroke:green;
         stroke-width:2" />

</svg>

</body></html>
```

Why would we want to use this longer version? SVG is great for a single static instance BUT js allows us to proceduralize, reuse or change content at run-time.

Equivalent d3.js Example

```
<html> <head>
<script type = "text/javascript" src =
"https://d3js.org/d3.v4.min.js"></script>
<style> body { font-family: Arial; } </style>
</head>
<body>
<div id = "svgcontainer"> </div>
<script language = "javascript">
         var width = 100;
         var height = 100;
         var svg = d3.select("#svgcontainer")
                  .append("svg")
                  .attr("width", width)
                  .attr("height", height);
               svg.append("ellipse")
                  .attr("cx", 40)
                  .attr("cy", 50)
                  .attr("rx", 40)
                  .attr("ry", 40)
                  .style("fill", "yellow")
                  .style("stroke", "green")
                  .style("stroke-width", 2);
                  </script>

</body>
</html>
```

Running d3.js Programs

d3.js intended for online web visualizations in a browser.

- you will generally need to deploy it on a web server

Option 1: Put it on a web folder and open it in a browser

Option 2: Run it locally through a local web server.

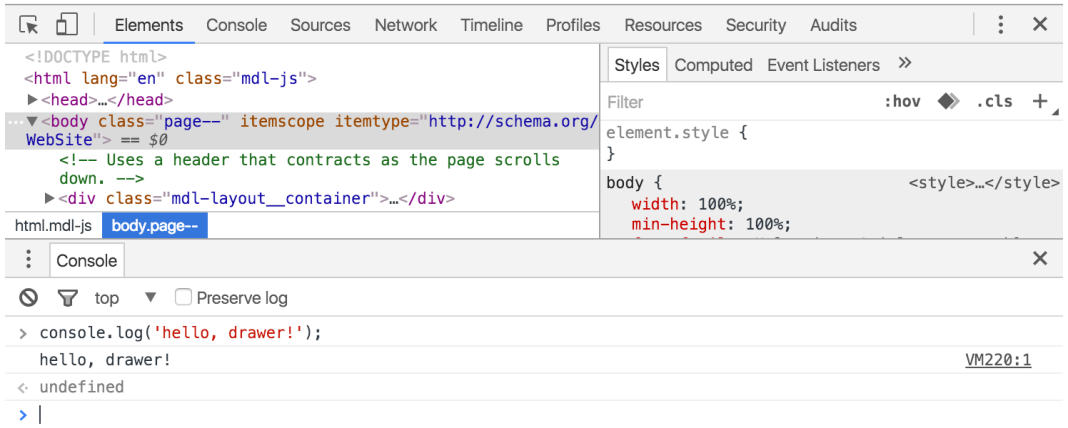
- Mac has built in support. See: http://goo.gl/u1DcQk
- For Windows, one popular option is WAMP: http://www.wampserver.com/en/
- You will need to either download the d3.js library or link to the library online [https://d3js.org/d3.v4.min.js]

Option 3: Various wholly-online editors/sketch books available for js:

- http://jsfiddle.net/tommy351/P4Z75
- https://beta.observablehq.com/playground

JS debugging supported in most browsers

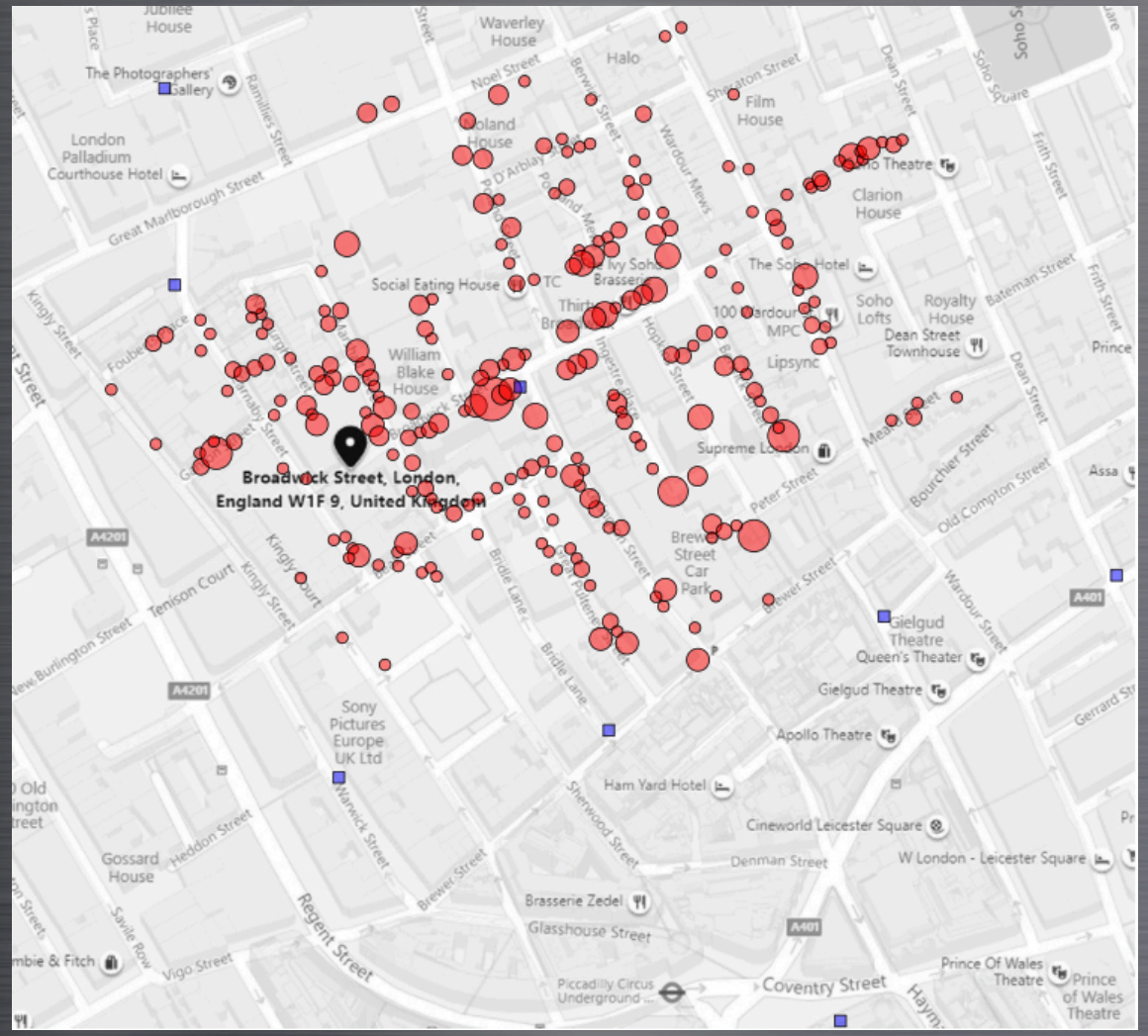
- on Chrome/Firefox, use keyboard shortcut Ctrl+Shift+ J (on Windows) or Option+Command+J (on Mac).



- On Safari: on the menu bar, go to Develop > Show Javascript Console or Shortcut (Option+Command+C). If you don't see the Develop menu, you'll need to turn it on using use the Safari> Settings > Advanced : Tick “Show Develop menu in menu bar”.

Summary: Getting Started With D3.JS

1. Choose a Text Editor
2. Write some basic code (see tutorials below)
3. Execute
   - Upload to a webfolder OR Use local webserver (see previous slide)
   - Run in a browser
4. Work through documentation/tutorials
   - Then the following are some recommended ones
     - d3 12-hour tutorial video by freeCodeCamp.org (intrduces js, html, svg, csss as well as d3): https://www.youtube.com/watch?v=_8V5o2UHG0E
     - d3.js beginners tutorial at tutorialspoint.com: https://www.tutorialspoint.com/d3js/d3js_data_join.htm
     - d3 tutorial by Mike Bostock (creator of d3): https://observablehq.com/@d3/learn-d3
   - Many others listed on https://github.com/d3/d3/wiki/Tutorials
   - Many chart examples with step-by-step code at: https://www.d3-graph-gallery.com/



Snow's cholera map

Solutions in processing, d3js and svg

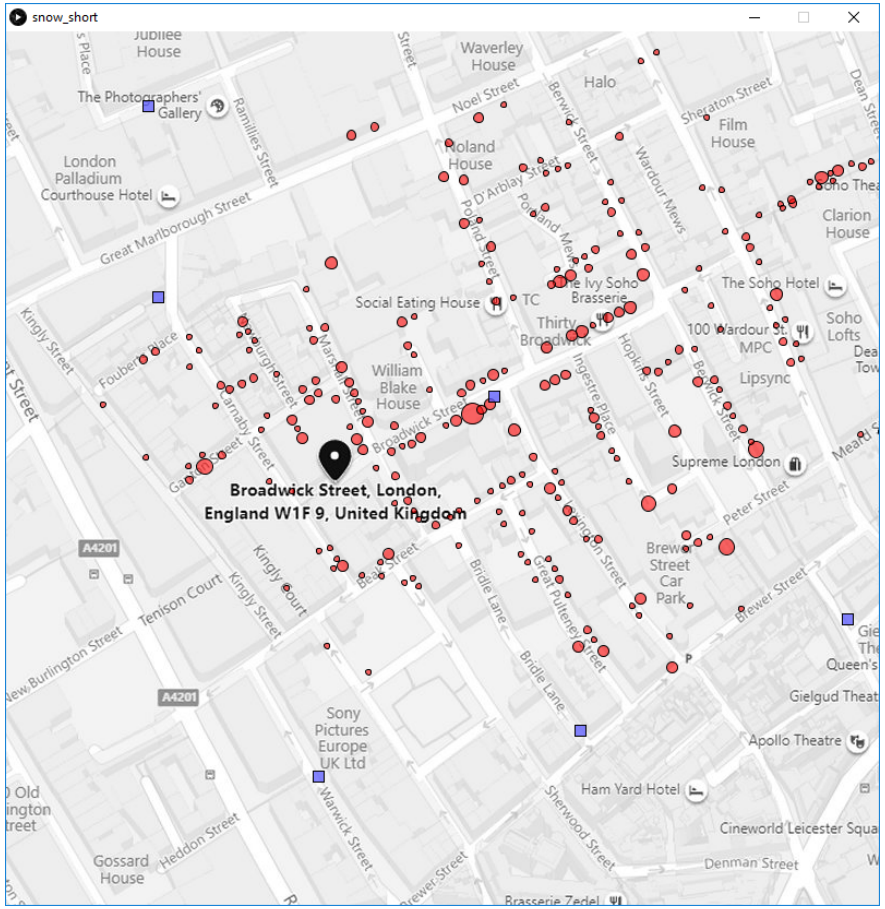
Last Week's Solution: Snow's Cholera Map p5

```
PImage map_image = loadImage("mapclean.jpg"); //load map image
Table table = loadTable("snow_pixelcoords.csv", "header"); //load data

size (800, 800);          //create window of 800 x 800 pixels dimension
image(map_image, 0, 0); //draw the map image starting on top-left corner

for (TableRow row : table.rows()) //iterate through rows of the data
{
  int num = row.getInt("count");
  float xx= row.getFloat("x_screen");
  float yy= row.getFloat("y_screen");

  if (num == -999)                //count of -999 indicates a WATER PUMP
  {
    fill(90, 90, 255, 150);       //set fill colour to bright blue
                                  //N.B. colours defined as Red, Green, Blue, Opacity (0-255)
    rect(xx-5, yy-5, 10, 10);     /draw rectangle of 10x10 pixels centered at xx yy
  }
  else
  {
    fill(255, 0, 0, 150);                          //set fill colour to bright blue
    ellipse(xx, yy, 5*sqrt(num), 5*sqrt(num));   //draw circle centred at xx yy
                                                   //N.B. area proportional to num_deaths
  }
}
```



USEFUL NOTE: Processing will let you save to SVG and PDF files so that you can output visualizations in vector format. See:
https://processing.org/reference/libraries/svg/index.html

Snow Example SVG

TRY THIS AT HOME: Look at the example and View Source at:
https://www.scss.tcd.ie/john.dingliana/d3js/snow/snowsvg.html

```
<svg width=1000 height=911>
<g>

<image href="./mapclean.jpg" />

<circle fill=#ff0000 fill-opacity=0.5 cx=314.4917326 cy=321.4091085 r=8.66025403784439 stroke=#000000 />
<circle fill=#ff0000 fill-opacity=0.5 cx=319.4455326 cy=331.0991085 r=7.07106781186548 stroke=#000000 />
<circle fill=#ff0000 fill-opacity=0.5 cx=322.6075326 cy=338.5791085 r=5 stroke=#000000 />
<circle fill=#ff0000 fill-opacity=0.5 cx=326.9289326 cy=347.9291085 r=5 stroke=#000000 />
<circle fill=#ff0000 fill-opacity=0.5 cx=331.6719326 cy=357.7891085 r=10 stroke=#000000 />
<!-- About 230 more lines of code -->
<circle fill=#ff0000 fill-opacity=0.5 cx=332.1989326 cy=586.9491085 r=5 stroke=#000000 />

<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=447.5065326 y=334.4991085 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=139.9493326 y=243.5491085 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=130.9903326 y=68.44910853 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=978.5117326 y=502.2891085 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=771.5061326 y=538.8391085 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=526.4511326 y=640.3291085 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=732.7189326 y=899.2391085 />
<rect fill=#0000ff fill-opacity=0.5 stroke=#000000 width=10 height=10 x=286.1391326 y=682.3191085 />

</g>
</svg>
```

SNOW Example d3.js

TRY THIS AT HOME: Look at the example and View Source at:
https://www.scss.tcd.ie/john.dingliana/d3js/snow/snowd3.html

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

<script>

// set the dimensions of the graph
  var width = 1000, height = 911;
// append the svg object to the body of the page
var svg = d3.select("#my_dataviz")
  .append("svg")
    .attr("width", width )
    .attr("height", height );

svg.append ("image")
.attr("x", 0)
.attr("y", 0)
.attr("y", 0)
.attr("href","./mapclean.jpg");

//Read the data
d3.csv("./snow_pixelcoords.csv", function(data) {

  // Add marks
  svg.append('g')
    .selectAll("deaths")
    .data(data)
    .enter()
    .append("circle")
     .filter(function(d){ return d.count>0; })
      .attr("cx", function (d) { return d.x_screen; } )
      .attr("cy", function (d) { return d.y_screen; } )
      .attr("r", function (d) { return
                              5*Math.sqrt(d.count); })
      .style("fill", "#ff5050a0")
      .style("stroke","#000000");

     svg.append('g')
    .selectAll("wells")
    .data(data)
    .enter()
    .append("rect")
      .filter(function(d){ return d.count<0; })
      .attr("x", function (d) { return d.x_screen-5; } )
      .attr("y", function (d) { return d.y_screen-5; } )
      .attr("width", 10)
      .attr("height", 10)
      .style("fill", "#0000ffa0")
      .style("stroke","#000000");
})
</script>
```

Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

The End

For now