D3 Tips and Tricks Notes

1. What is d3.js?
   1. D3 - JS lib for manipulating docs based on data
2. Introduction
   1. Break it down by components
3. What do we need to get started?
   1. HTML, JS, CSS, Web Servers
   2. HTML – Hypertext Markup Language
      1. Stuff the web pages are made of
   3. JS – scripting language that will make D3 do the fanciness
   4. CSS – Cascading Style Sheets
   5. Web Server – Allow access to HTML files and provide structure that allows it to be displayed on a web browser
   6. Setting up a local web server
      1. Python m- http.server
   7. Getting D3
      1. <script src="https://d3js.org/d3.v6.min.js"></script>
4. Starting with a Simple Graph
   1. HTML
   2. CSS
      1. The period (.) in front of line indicates the selector is a class
   3. D3 JS
      1. Setting up the margins and the graph area
         1. Margins
            1. *var margin = {top: 20, right: 20, bottom: 50, left: 70}*
            2. Origin is in the top left hand corner of the screen
            3. top: 20 will move the origin point down 20 pxs and left: 70 will move the origin point to the right 70 pxs
         2. Width
            1. *width = 960 - margin.left - margin.right*
            2. Width of the inner block of the area where the graph will be drawn
         3. Height
      2. Getting the Data
         1. Fetch – introduced delay to the smooth loading of a webpage
         2. Promise – allows an object to process on the assumption that the values retrieving will be available in due course
         3. Fetch and Promise gone since v5 of D3
         4. Can take in
            1. Text
            2. JSON
            3. XML
            4. HTML
            5. CSV
            6. TSV
            7. DSV
         5. Request – d3.csv
         6. Data File – (data.csv) -> URL of the file
         7. Promise – then(function(data) {
            1. Loads the data that is ingested as the array ‘data’
         8. Data.forEach(function(d) {
            1. All values that are pulled out of the csv file are set and formatted correctly
            2. For each row
            3. d.date = parseTime(d.date)
            4. d.close = +d.close

+ => makes the variable numeric if not already \*\* GOOD PRACTICE

* + 1. Formatting the Date/Time
       1. *d3.timeParse(specifier)* EX of specifier - “%d-%b-%y”
          1. % -> prefixes to each separate format type
          2. - -> literals for actual ‘-‘
          3. d -> zero padded day of the month [01,31]
          4. b -> abbreviated month name
          5. y -> year without the centuries as a decinam number
          6. [d3-time-format/README.md at main · d3/d3-time-format · GitHub](https://github.com/d3/d3-time-format/blob/main/README.md#locale_format)
    2. Setting Scales Domains and Ranges
       1. Range
          1. Range – the size of the chart
          2. For X axis .range([0, width])
          3. For Y axis .range([height, 0])
          4. 0,0 maxX, 0
          5. 0, maxY maxX, maxY
          6. Range.([closer\_to\_origin, further\_from\_origin])
       2. Domain – what the scope of the data will be
          1. D3.extent for x domain

Returns the min and max of a given array

*x.domain(d3.extent(data, function(d) { return d.date; }));*

* + - * 1. y Domain

*y.domain([0, d3.max(data, function(d) { return d.close; })]);*

* + 1. Adding data to the Line Function
       1. D3.line()
          1. Assign a set of coordinates in a form that can be used to draw a line
          2. Uses the d3.scaleSomething to transform the coordinates to screen cords
    2. Adding SVG element
       1. SVG element – an element designed for drawing graphics on
          1. Have width + left margin + right margin
          2. Have Height + top margin + bottom margin
       2. ‘g’ group element – top left corner of the actual graph area on the page, minus the margins
    3. Actually Drawing Something
       1. Drawing the Line
          1. append.(“path”)

[Paths — SVG 2 (w3.org)](https://www.w3.org/TR/SVG/paths.html)

path

Moveto

Lineto

Curveto

Arc

Closepath – z

d property

string value

* + - * 1. data vs datum

data

takes the specified array and joins it to a selection of data

datum

exists as a static element no joining

* + - * 1. class

Used CSS styling

* + - * 1. Attribute d

What function to call to give the coordinates of data

* + - 1. Drawing the Axes
         1. Both x and Y requires a ‘g’ element
         2. Y Axis

Can be drawn at default placement

* + - * 1. X Axis

Needs to be drawn at the bottom of the graph

Transform -> trnalsate

* + - * 1. D3 Axis functions

axisTop – ticks and values above horz axis

axisBottom - ticks and values below horz axis

axisRight - ticks and values right vert axis

axisLeft - ticks and values left vert axis

1. Things We can do with a Simple Graph
   1. Setting up and configuring the Axes
      1. Change the Text Size
         1. Default Size – 10px with font sans-seriff
         2. Use classes for more than one element with same font size
      2. Changing the number of Ticks on a Axis
         1. Use .ticks(#) to specify number of ticks along axis but there are certain intervals for time specifically
         2. Specifying Own Intervals
            1. D3.timeDay.every(4)
            2. Standard units Milliseconds to Year
      3. Rotating text labels for a graph axis
         1. Text-anchor – style and if “end” then the end of the label is attached to the tick origin point
         2. dx and dy
            1. Shifts along the corresponding axis
         3. Transform – rotate 65
      4. Formatting a date/time axis with specified values
         1. D3.timeFormat()
   2. Adding Axis Labels
      1. X Axis Label
         1. Bad example
            1. Text element with x and y attributes
            2. Text-anchor middle
            3. .text(“Date”) adds the actual text
            4. BAD PART – hard coded the location
         2. Good example
            1. Make x and y attributes relative to the width and height of the graph
         3. Another good example
            1. Use transform instead of x and y attributes
      2. Y Axis Label
         1. Rotation changes the coordinate system
         2. Bottom left hand corner is now 0,0 y,x
         3. 1em – a unit of measure that equals exactly one unit of the currently specified text point size, in other words this shifts the Value label to the right by the height of the text
   3. How to Add a title to your graph
   4. Changing a line chart into a ScatterPlot
      1. selectAll(“dot”) -> suitable group label for svg circle elements
      2. data(data) -> what data we are binding
      3. .enter().append(“circle”) -> adds a circle for each data point in data
      4. Radius, cx, cy
   5. Smoothing Out Graph Lines
      1. On d3.line().curve(d3.curveBasis)
      2. Other options
         1. curveLinear – normal jagged
         2. curveLinearClosed – closed in a loop
         3. <https://github.com/d3/d3-shape/blob/master/README.md#curves>
   6. Making A Dashed Line
      1. On append(“path”) add .style("stroke-dasharray", ("3, 3"))
         1. In which 3, 3 is 3 pixels on and 3 pixels off
   7. Filling an Area under the graph
      1. Define the Area Function d3.area()
         1. For all xs from y0 to y1
      2. Draw the area
   8. Adding A Drop Shadow to Allow Text to stand out on graphics
   9. Adding grid lines to a graph
      1. Add .tickSize([size]) to the axisLeft and axisBottom call
      2. .tickFormat(“”) suppresses printing of any label
   10. Adding more than one line to a graph
       1. Adjust y scale domain to account for points being higher or lower on second line
   11. Labeling Multi Lines on A graph
       1. How can we put text at the end of each line on the graph?
   12. Multiple Axes for a Graph
       1. Create 2 y scales and then add the scales to each side of the graph
2. Elements, Attributes, and Styles
   1. Framework
      1. Circle – cx, cy, radius
   2. Elements
      1. Subset of SVG Elements
      2. Circle
         1. Cx,cy,r
      3. Ellipse
         1. Cx, cy, rx, ry
      4. Rectangle
         1. X,y,width,height,rx,ry
         2. Rx and ry curve the corners of the rectangle
      5. Line
         1. X1,y1,x2,y2, stroke (style)
      6. Polyline
         1. Points attribute
         2. Stroke style
         3. Fill style
      7. Polygon
         1. Points attribute
         2. Stroke style
         3. Fill style
      8. Path
         1. D attribute of mini language of commands
      9. clipPath
         1. use in combination with another shape to remove any parts of the combined shape
         2. We have a cookie cutter ellipse and a rectangular dough, place ellipse on the dough to get the intersection of the shape
         3. Append to rectangle the attr(“clip-path”, “url(#ellipse-clip)”)
      10. Text
          1. X,y,dx,dy,text-anchor
          2. Fill style black
          3. Text anchor
             1. Middle start end – y justification bottom left middle or right of cords
             2. Use dy .71 em for top, 0.35 em for bottom
   3. Attributes
      1. Transform (translate, scale, rotate)
         1. Scale – not only changes the size but also the position of the shape
         2. Rotate – applies to both text element and x and y attributes
      2. Dx, dy – offset of text elemenets from the anchor of x and y
         1. Can be em , px, pt, and %
         2. 0.35em dy – add half the height of the text to the dimension y
      3. Textlength – adjusts the length of the text to fit a specified value
         1. Letter and word spacing are adjusted, the letters remain the same size
      4. lengthAdjust – have spacing of text element controlled by either spacing or spacingAndGlyphs
         1. spacing : letters remain same size but spacing between letters and words are adjusted
         2. spacingAndGlyphs: text is stretched or squeezed to fit
   4. Styles
      1. Fill
         1. Hex or rgb
      2. Stroke
         1. Hex or rgb
      3. Opacity
         1. Element’s transparency
         2. 0 complete see through to 1 complete solid color
         3. Fill-opacity and stroke-opacity
      4. Stroke-width – in pixels
      5. Stroke-dasharray
         1. Dashes rather than a straight line 10, 3 10 pixels dash and 3 pixels space
      6. Stroke-linecap
         1. Control of shape of the ends of the lines
         2. Butt
            1. Butts up to the starting or ending position and is cut off squarely
         3. Round
            1. Line is rounded in proportion to its width
         4. Square
            1. Line squared off but extended in proportion to its width
      7. Stroke-linejoin
         1. Specifies shape of the join of two lines
         2. Path polyline and polygons
         3. Miter
            1. Squared off
         4. Round
            1. Outside portion of the join is rounded in proportion to its width
         5. Bevel
            1. Straight edged outer portion slipped off slightly more contoured effect while still being angular
      8. Writing-mode
         1. Orientation of the text so it prints out top to bottom -tb
      9. Using CSS styles
         1. Good for when many elements have the same style consider using classes
3. Bar Charts and Histograms
   1. Bar – each col rep a group defined by a category while histogram – each col rep a group defined by a range
   2. Bar Charts
      1. The data
      2. The code
      3. The Bar chart explained
         1. D3.scaleBand()
            1. Allows creation of a series of unfiorm bands that can be computed from the assigned range
            2. Padding(0.1) – space made between bars is 1/10 of the width of the space available for each band/bar.
            3. Then can get .bandwidth() for width of rectangle
   3. Histograms
      1. The data
      2. The code
      3. The histogram explained
         1. D3.bin()
            1. Form our data into bins that form continuous non overlapping intervals
            2. Thresholds are used for the bins in this case monthly
4. Tree Diagrams
   1. What is a tree diagram?
      1. Produce a node link diagram that lays out connection between nodes in a method that displays the relationship of one node to another
      2. Data needs to be json with a name and children when appropriate
   2. Tree Diagram Explained
      1. D3.tree()
         1. Size([width,height]) – size of diagram
      2. D3.hierarchy() – constructs a root node from hierarchical dat
         1. Node.data
            1. Data associated with a node
         2. Node.depth
            1. Zero for root, Number of hops from initial root node
         3. Node.height
            1. Zero for leaves, greatest distance from any descendant leaf nodes
         4. Node.parent
            1. Null if root node
         5. Node.children
            1. Undefined for any leaf nodes
      3. Map the node data to d3.tree()
      4. Draw links
         1. .data( nodes.descendants().slice(1))
            1. Descendants returns the array of descendant nodes
            2. Slice(1) doesn’t include the main root node since links will be drawn from child to parent
         2. .append(“path”)
         3. .attr("d", function(d) {

return "M" + d.x + "," + d.y

+ "C" + d.x + "," + (d.y + d.parent.y) / 2

+ " " + d.parent.x + "," + (d.y + d.parent.y) / 2

+ " " + d.parent.x + "," + d.parent.y;

})

* + - * 1. M – moveto Absolute coordinates
        2. C – drawBezierCurve using control points from current position to end point absolute coordinates
        3. "C" + d.x + "," + (d.y + d.parent.y) / 2

Draw cubic Bezier curve 1st control point d.x for x and halfway between starting node and parent for y

* + - * 1. d.parent.x + "," + (d.y + d.parent.y) / 2

Sets 2nd control point with parent node x for x and halfway between starting node and parent for y

* + - * 1. d.parent.x + "," + d.parent.y;

end point of curve at the parent node location

* + 1. Create node as a group var node =
    2. Draw node Circle
    3. Draw node text
  1. Horz tree diagram explained
     1. Need to flip the x and y coordinates
  2. Styling nodes in tree
     1. Changing node and link colors
     2. Changing the nodes to different shapes
        1. D3.symbolGenerator
           1. Circle
           2. Cross
           3. Diamond
           4. Square
           5. Triangle
           6. Star
           7. Y
     3. Using images as Nodes
        1. Xlink:href defines the location of the referenced image
  3. Generating a tree diagram from external data
     1. D3.json
  4. Generating a tree diagram from flat data
     1. D3.stratify()
        1. Requires a unique identifier to be used for each node and will be declared as .id
        2. Also need parent id
  5. Generating a tree diagram from CSV file
  6. Interactive Tree Diagram
     1. Enter – update -exit
     2. D3.merge
        1. Flattens the iterable of iterables into a new array

1. Sankey Diagrams
   1. What is a Sankey Diagram
      1. Flow diagram where the flow represented by arrows of varying thickness depending on the quantity of flow
      2. Useful in demonstrating proportionality to a flow where diff parts of diagram rep diff quantities
   2. How d3.js Sankey diagrams want their data formatted
      1. Nodes list with node (id) and nodeName
      2. Links list with source target and value
   3. Description of code
      1. D3.sankey()
         1. Sets node width
         2. Node padding
         3. And size diagram
         4. Makes the links between the nodes and the node locations
   4. Formatting Data for Sankey
2. Bullet Charts
   1. Introduction to Bullet Chart Structure
      1. Text Label
         1. IDs the performance being measured
      2. Three Background fill colors that show qualitative ranges like low medium and high
         1. No fewer than two and no more than 5
      3. Symbol marker that shows a comparative value
      4. Quantitative scale
      5. Bar that shows the performance measure
   2. Code
      1. D3.bullet()
         1. Define the chart size
         2. Additional file used
   3. Adapting and Changing Bullet Chart Components
3. Mapping with d3.js
   1. GeoJSON and TopoJSON
      1. GeoJSON
         1. Encodes the data in JSON type hierarchy
         2. Often have too much detail
      2. TopoJSON
         1. Geometries are not encoded directly
         2. Saves 20% of the file size wise
   2. Starting with a Simple Map
      1. Projection
         1. Way that the geographic coordinate syste is adjusted for display in flat screen
      2. D3.geoMercator()
         1. Center
            1. Projections center as 2 array longitude and latitude in degrees
         2. Scale
            1. Default is 150
         3. Rotate
            1. 3 axis rotation yaw pitch roll
      3. D3.geoPath()
         1. Used to specify a projection type .projection(projection)
   3. Zooming and Panning
      1. D3.zoom()
   4. Display Points on map
4. Assorted Tips and Tricks
   1. Adding tooltips
      1. Transitions
         1. Gradually interpolating styles and attributes over time
      2. Events
         1. Mousedown
         2. Mouseup
         3. Mouseover
         4. Mousemove
         5. Click
         6. Contextmenu
         7. Dblclick
      3. Get tipping
   2. What are the predefined colors?
   3. Selecting and filtering a subset of objects
      1. Filter function
   4. Select items with an if statement
   5. Applying a color gradient to a line based on value
      1. Mike Bostock
      2. Smooth color transitions along a vector form one colour to another
   6. Applying a color gradient to an area fill
   7. Transitions
      1. Chaining
         1. Add a transition, duration pair for each attribute
      2. Easing
         1. Linear motion vs slowing up and down
         2. .ease(d3.easeLinear)
            1. Quad
            2. Cubic
            3. Poly
            4. Sin
            5. Exp
            6. Circle
            7. Bounce
            8. Back
            9. Elastic
         3. Can be modified with In or Out or stopping/starting at different points in the curves
      3. Looping
   8. Show/hide an element by clicking on another element
   9. Using html inputs with d3.js
   10. Add an html table to your graph
       1. Table
       2. Tr – row
       3. Th – header
       4. Td – col cell
   11. More table madness sorting prettifying and adding columns
   12. How to use data imported from a csv file with spaces in the header
   13. Extracting data from a portion of a string
   14. Selecting a random string from an array
   15. Adding web links to d3.js objects
   16. Export an image from a d3.js page as a svg or bitmap
       1. Bitmap
          1. Compose of lots of discrete individual dots
       2. SVG
          1. Image is created by describing they way the image is created
          2. Zooming in everything is still smooth
       3. Exporting – final result bit map
          1. Copying the image from the web page and save it as a SVG file
             1. SVG Crowbar

Extracts SVG nodes and accompanying styles from HTML and downloads them as svg file

Means drag the bookmarklet from web page to bookmarks

Click on svgcrowbar bookmark and will be prompted for a location to save image as svg

* + - 1. Open SVG image in a program designed to use vector images and edit if required
      2. Export image as bitmap
  1. Understanding JSON

1. D3.js examples explained
   1. Multi-line graph with automatic legend and toggling show/ hide lines
      1. Array.from(target, mapFunc, thisarg)
      2. D3.group(data, d => d.symbol)
      3. \s – whitespace
      4. g – global flag
      5. + - contiguous string
      6. /\s+/g – any contiguous string of space of characters
   2. Difference chart: Science vs Style
      1. clipPath
      2. difference chart
   3. My Favorite Tooltip method for a line graph
      1. Line chart tooltip with grid axises