Ideas

Area Chart

* HTML in chrome shows 960 pixels. This is per index.html.

.classed("region", true) //add a class, without erasing any existing classes

.attr("fill", colour(d[1]) ); //colour based on the data

Show series of many lines for the various HS10 categories

User hovers over each these lines and then we can the details.

Can display say about 1000 lines.

***To Do***

Fast movers for current period (country / industry)

Multiple line chart of countries / industries (like the expected death)

http://projects.flowingdata.com/life-expectancy/

Notice in the above how you can click on the various buttons and they will display different colours.

Some move interactivity when the brush moves – display the % change in growth in the area graph.

Use some of the meta data contained in the Geo Json to slice and dice.

***Time Series with missing values***

<http://bl.ocks.org/cgroll/c5e7bdb5dffb12818623>

***Consider a steamgraph***

***To Do With the Map***

Double click to zoom in.

Organise a concordance with three digit codes

Add as many countries as you can with the data.

Get rid off New Zealand and some of the other sovient islands.

A legend for the map

A line graph when hover on the map.

Some type of category display when hover over the map.

Possibilly of animating the changes when the brush stops. DONE

***zoom effect on map***

<https://bl.ocks.org/john-guerra/43c7656821069d00dcbc>

***bostock defer example***

bostock choropleth

d3.queue() .defer(d3.json, "/mbostock/raw/4090846/us.json") .defer(d3.tsv, "unemployment.tsv", function(d) { rateById.set(d.id, +d.rate); }) .await(ready);

***To Do With the Area Chart***

Can you break this up by continent -- like an area chart but with different colours for each continent.

We get some new data

Then we do the d3 dance (joining).

And then we can slot the class straight in

// map.keys() returns ["foo", "bar"]

// map returns i {$foo: 1, $bar: 2}

// map.entries() returns an array of objects

// map.get("foo") returns 1

// map.has("foo") returns true

// map.has("foooo") returns false.

// map.values() returns [1, 2]

var map = d3.map()

.set("foo", 1)

.set("bar", 2);

***Nests Tutorial***

http://bl.ocks.org/phoebebright/raw/3176159/

***Area chart - initial***

Get a variable that contains all columns (keys)

set the x domain (using date)

set the z domain (keys)

set stack stack.keys(keys)

g.selectAll(“.layer”).data(stack(data))

layer.append(“path

***Area chart - initial***

Initial called with fn\_initial(gg\_data) This means that layers contains the data. Layers is defined as follows:

layers = stack(nest.entries(l\_data));

The data set should have 37 rows. The data set is defined as follows Key, Value, Date

Layers is an array of three objects. layers.length == 3

layers[1].key == “Group2”

layers[1].values.length == 12

layers[0].values[0] Object {key: "Group1", value: 17, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST), y0: 0, y: 17}

layers[1].values[0] Object {key: "Group2", value: 5, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST), y0: 17, y: 5}

layers[2].values[0] Object {key: "Group3", value: 46, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST), y0: 22, y: 46}

fn\_initial(gg\_data)

Tried the following:

layers = nest.entries(l\_data);

But fails on call to area.

Looking inside .attr("d", function(d) ***without the stack***

Without the stack means: ***layers = nest.entries(l\_data);***

***We get***

console.log(d.key) is Group1, Group2, Group3

console.log(d.values) is [object, object, object……12 objects…3 times].

The result of d.value was saved to a variable…here is a result:

initial\_data[0]

Object {key: "Group3", value: 46, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST)}

Looking inside .attr("d", function(d) ***with the stack***

With the stack means: ***layers = stack(nest.entries(l\_data));***

console.log(d.key) is Group1, Group2, Group3

initial\_data[0]

Object {key: "Group3", value: 46, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST), y0: 22, y: 46}

***So including the stack means we get the two extra values: y0 and y***

***Looking at layers again with and without the stack***

***With stack***

layers[0].values[0] Object {key: "Group1", value: 17, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST), y0: 0, y: 17}

***Without stack***

layers[0].values[0] Object {key: "Group1", value: 17, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST)}

So, once again we see that the ***stack produces the two extra values.***

Here are some values from layers with stack (group3)

layers[2].values[0] Object {key: "Group3", value: 46, date: Mon Apr 16 2012 00:00:00 GMT+1000 (AEST), y0: 22, y: 46}

layers[2].values[1] Object {key: "Group3", value: 56, date: Tue Apr 17 2012 00:00:00 GMT+1000 (AEST), y0: 34, y: 56}

layers[2].values[3] Object {key: "Group3", value: 16, date: Wed Apr 18 2012 00:00:00 GMT+1000 (AEST), y0: 40, y: 16}

***Here is a trace of various groups of layers***

layers[0].values[3] Object {key: "Group1", value: 37, date: Thu Apr 19 2012 00:00:00 GMT+1000 (AEST), y0: 0, y: 37}

layers[1].values[3] Object {key: "Group2", value: 19, date: Thu Apr 19 2012 00:00:00 GMT+1000 (AEST), y0: 37, y: 19}

layers[2].values[3] Object {key: "Group3", value: 39, date: Thu Apr 19 2012 00:00:00 GMT+1000 (AEST), y0: 56, y: 39}

**Inpsect the data**

d3.selectAll(".layer").data()

***Steps required to create the area chart***

***Creating quantiles based on growth***

If either start or end are zero….then color is neutral

We need to extract negative and positive values and set the domain for ScaleQuantile before applying it.