Add a tooltip (using d3)

https://bl.ocks.org/d3noob/257c360b3650b9f0a52dd8257d7a2d73

Copy CSS

Add tooltip div

Add mouse events

Add a tooltip (using Bootstrap)

https://www.w3schools.com/bootstrap/bootstrap_ref_js_tooltip.asp

Add the Bootstrap files and HTML links from the example in week 3

Replace .on('mouseover'… with the following:

```
.attr ( 'data-toggle' ,  'tooltip' )
.attr ( ' title ' ,  function ( d )  {
          return d.women;
});
```

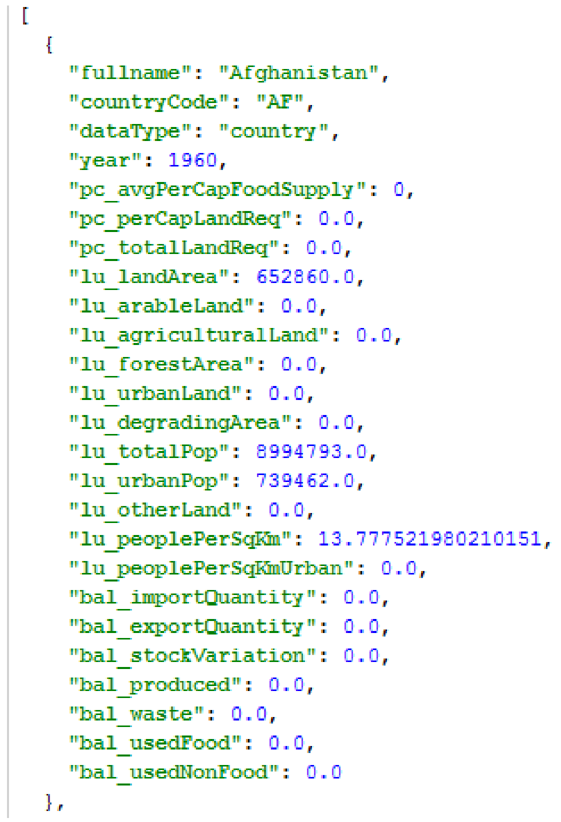
Initialize the tooltips, using JQuery

```
$(' [ data-toggle =  "tooltip" ] ' ) .tooltip(  ) ;
```

Comment out the old CSS tooltip styling (interferes with Bootstrap), and add your own, if desired

Other data formats: JSON

JSON files are Javascript-formatted data objects, and are often used in place of .csv files for big projects.



```
[
  {
    "fullname": "Afghanistan",
    "countryCode": "AF",
    "dataType": "country",
    "year": 1960,
    "pc_avgPerCapFoodSupply": 0,
    "pc_perCapLandReq": 0.0,
    "pc_totalLandReq": 0.0,
    "lu_landArea": 652860.0,
    "lu_arableLand": 0.0,
    "lu_agriculturalLand": 0.0,
    "lu_forestArea": 0.0,
    "lu_urbanLand": 0.0,
    "lu_degradingArea": 0.0,
    "lu_totalPop": 8994793.0,
    "lu_urbanPop": 739462.0,
    "lu_otherLand": 0.0,
    "lu_peoplePerSqKm": 13.777521980210151,
    "lu_peoplePerSqKmUrban": 0.0,
    "bal_importQuantity": 0.0,
    "bal_exportQuantity": 0.0,
    "bal_stockVariation": 0.0,
    "bal_produced": 0.0,
    "bal_waste": 0.0,
    "bal_usedFood": 0.0,
    "bal_usedNonFood": 0.0
  },
```

http://www.convertcsv.com/csv-to-json.htm

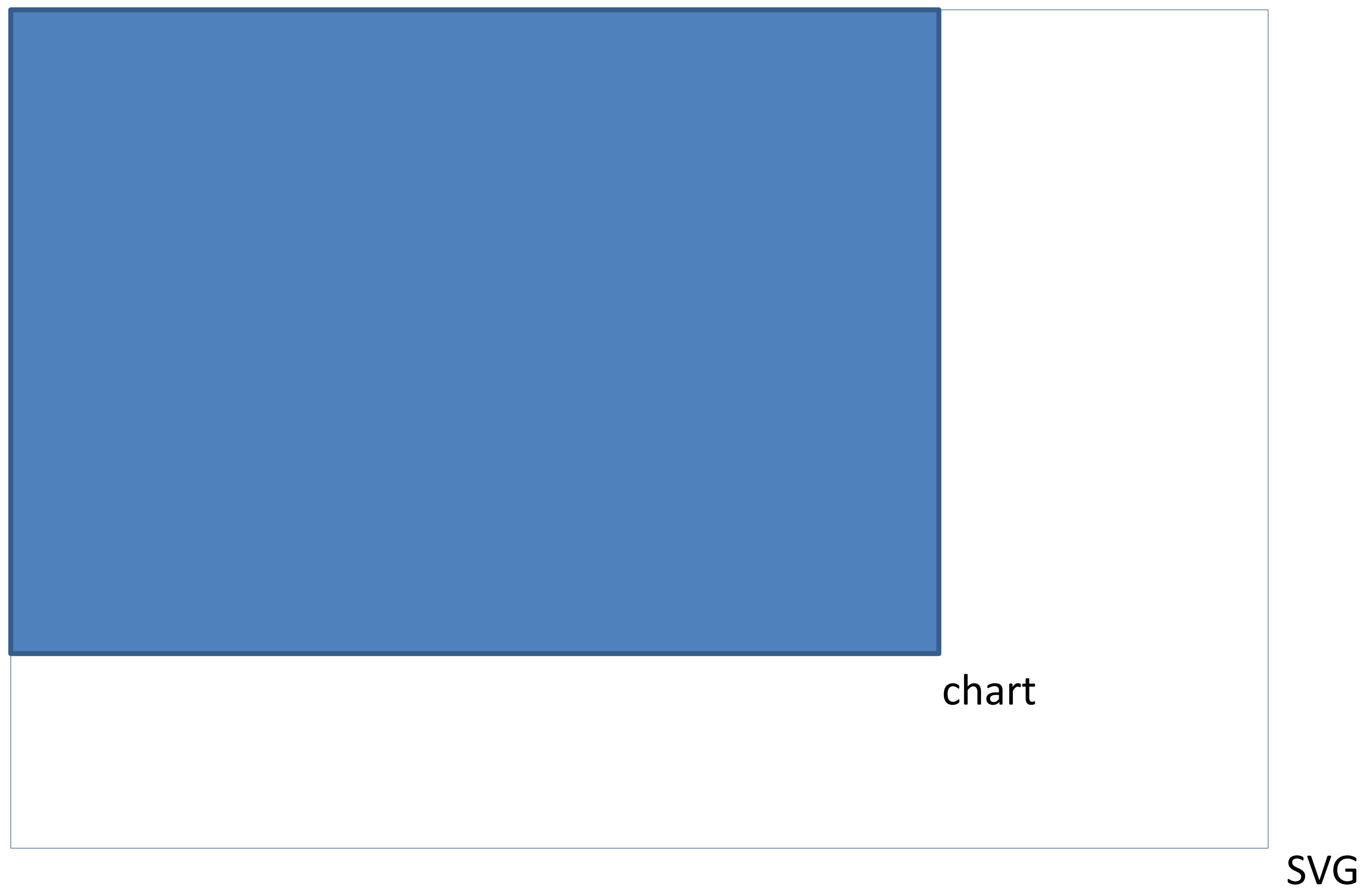
Side note: Crossfilter

Crossfilter is another library that's used for handling really big datasets efficiently, especially if you want to do a lot of filtering, sorting, and aggregating within the data.

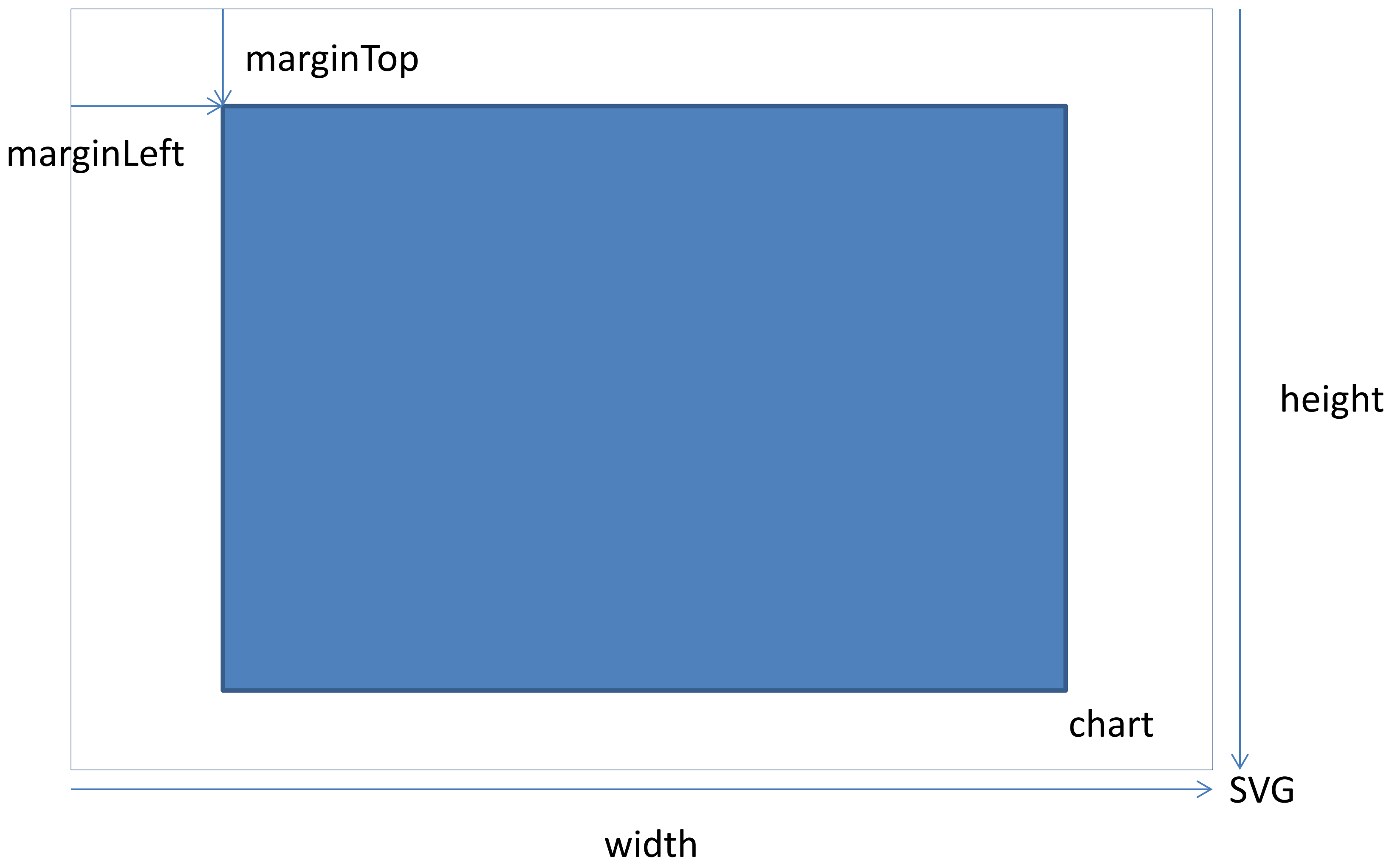
http://square.github.io/crossfilter/

I recommend sticking with simple JS functions unless you really need it, but it's worth knowing that the library exists, for those who have really big data or want to do complicated comparisons.

Setting up a basic (generic) chart position



Setting up a basic (generic) chart position



Setting up a basic chart position

Put the chart items in a group, and translate the whole group over. First, get the SVG width and height:

```
var width = d3 .select ( 'svg‘ ) .attr ( 'width‘ );
var height = d3 .select ( 'svg‘ ) .attr ( 'height‘ );

var marginLeft = 100;
var marginTop = 100;

var svg = d3.select ( 'svg‘ )
    .append (‘g‘ )
    .attr (‘ transform ‘ , ‘ translate ( ' + marginLeft  +  ‘ , '  +  marginTop  +  ‘ ) ‘ );
```

Now, put everything into this group, using svg.append( ) as usual

Drawing a bar chart: switching out data files

Update d3.csv file name to countryData_topten.csv

Comment out everything else

Update slider positions to match year values in new file

Before we start: Automating axis scaling

Want to make the x-axis more general; don’t want to keep filling in all the values by hand, like we did before:

```
var scaleX = d3 .scalePoint() .domain ( ["16-19"  , "20-24" , "25-34" , "35-
44" , "45-54" , "55-64" ,"65+" ] ) .range( [ 0 , 600 ]);
```

Instead, declare scaleX and scaleY at the top of the document, but don’t set the domain yet:

```
var scaleX = d3 .scalePoint ( ) .range ( [ 0 , 600 ] );
var scaleY = d3 .scaleLinear ( ) .range ( [ 400 , 0 ] );
```

Then, set the domain and call the axis *after* the data loads (come back to this in a minute)

Automating axis scaling: using loaded data for a categorical axis

Want axis values to automatically populate based on the data: need a way to find out what’s inside the data when it loads.

Array mapping returns a list of values for each object in an array:

```
loadData .map ( function ( d ) {
            return   d .countryCode ;
} );
```

This .map() function returns an array with a list of all the country codes for the loadData array.

Use this to set the domain for the x axis:

```
scaleX .domain (
            loadData .map ( function ( d ) { return  d.countryCode ; } )
)
```

Automating axis scaling: using loaded data for a quantitative axis

For y, we want something a little different. Instead of a list of all the values, we really only want the maximum value in the array.

For this, we do the same thing to get a list of values for the variable that we want to measure:

```
loadData .map ( function ( d ) { return   d .totalPop  } );
```

And then wrap it up inside a d3 .max( ) function to get only the biggest value.

```
scaleY .domain( [ 0 ,
        d3.max (
                loadData .map ( function ( d ) { return   d .totalPop  } )
        );
] );
```

(Note that there is a more general Math .max ( ) function in JS, but d3's .nest function adds an array element that messes it up)

Drawing a bar chart: adjusting x axis scaling

For a bar chart, we also need to change the scale type for the x-axis:

```
scaleX = d3 .scaleBand ( ) .rangeRound ( [ 0, 600 ] ) .padding ( 0.1 )
```

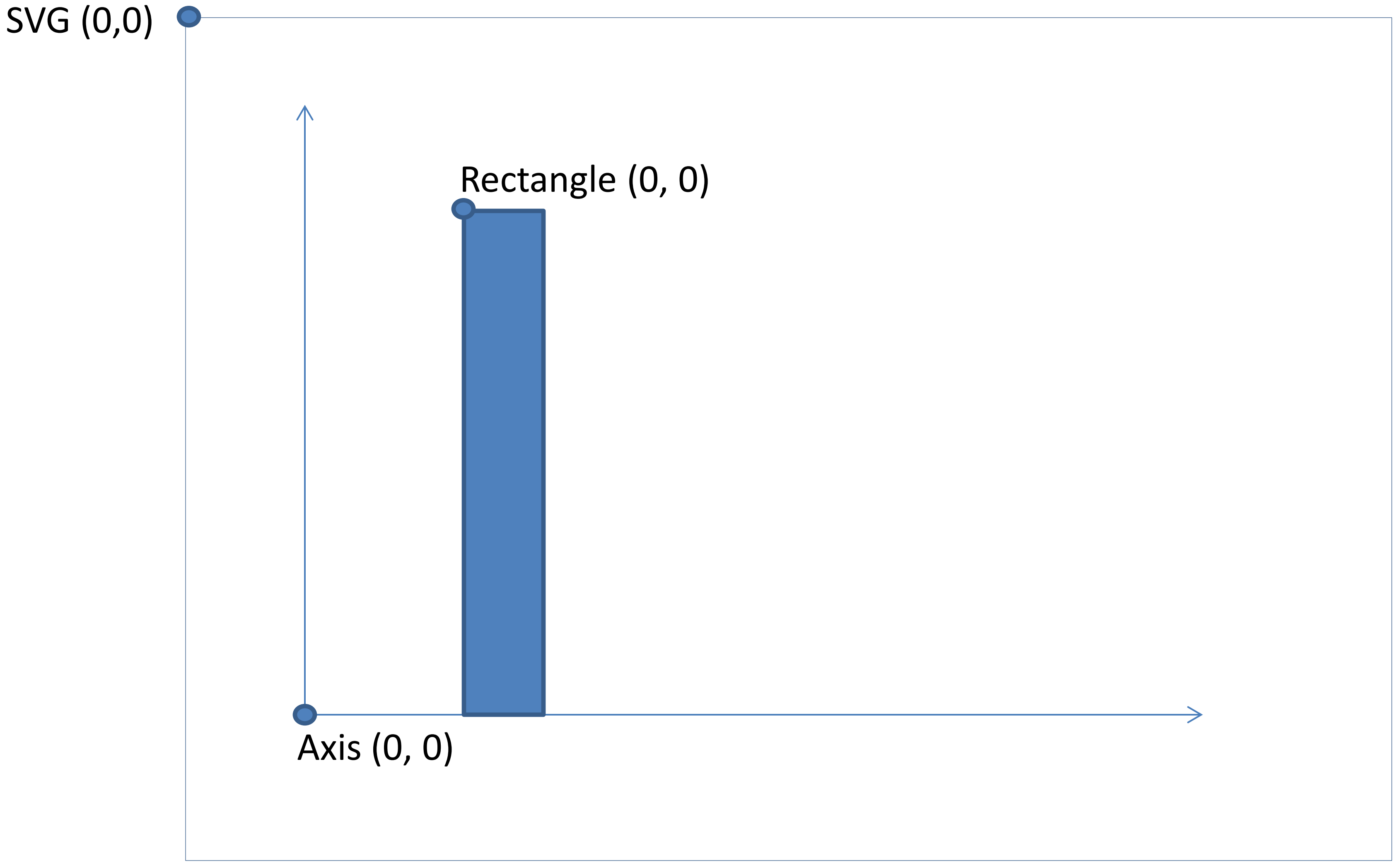
.scaleBand() positions bars evenly along the axis,
rangeRound() is used instead of .range(), and
.padding() determines the amount of space between bars.

Drawing a bar chart: adjusting the draw function

Bind to the data, as before:

```
svg .selectAll ( ‘rect‘ )
       .data ( loadData )
       .enter ( )
       .append ( ‘rect‘ )
       .attr ( ‘class‘ , ‘bars‘ )
       .attr ( 'fill‘ , ‘slategray’ );
```

Drawing a bar chart: update the draw function



Drawing a bar chart: update the draw function

```
svg.selectAll('rect')
    .data(pointData)
    .attr('x',function(d){
        return scaleX(d.countryCode);
    })
    .attr('y',function(d){
        return scaleY(d.totalPop);
    })
    .attr('width',function(d){
        return scaleX.bandwidth();
    })
    .attr('height',function(d){
        return 400 - scaleY(d.totalPop);
        //400 is the beginning domain value of the y axis, set above
    });
```

Update the axes

Notice what happens when the slider values change: bars update, but axes don’t!

Need to update the axis scaling in the update function!

Move scale function to drawPoints()

Add a class to the y axis group in the d3.csv function, then select it in drawPoints() and call the axis again to rescale

```
d3 .selectAll ( ‘ .yaxis ‘ )
      .call ( d3 .axisLeft ( scaleY ) );
```

Object constancy -- switch to barChart_exit

Each bar in a bar chart represents a piece of data. What happens to a bar if it’s data disappears (a top ten chart, where one country falls off the list?)

```
d3 .csv (‘ . / countryData_topten .csv‘ ,  function ( dataIn ) {…
```

Object constancy is a principle of visualization that states that each object (bar in a bar chart) should be related to one unique piece of data.

Ensuring object constancy requires use of a **key function**. It pairs each bound DOM element to the key property specified in the function:

```
 svg.selectAll(‘rect‘ )
       .data ( pointData ,  function ( d ) { return d. countryCode } )
```

Change in both the original data bind and the update function.

The “Exit” in enter, exit, update:

When your data is bound to a DOM element using a key function, you can’t just rebind a new array to the same element if the list of keys isn’t the same; you’d have an orphaned bar left over.

Instead, you need to add an .exit() function to the .enter() and update functions that we’ve been using.

First, move the data bind into the drawPoints( ) function. Separate the selection and data bind from the rest, and store it in a variable:

```
var rects = svg.selectAll('.bars')
      .data(pointData, function(d){return d.countryCode;});
```

Next, get rid of any bars that are no longer needed:

```
rects .exit ( )
      .remove ( );
```

Enter, exit, update:

Update the remaining bars, just like before:

```
rects
        .attr ( 'x‘ , function ( d ) { …
```

And then add in any new bars needed:

```
rects
        .enter ( )
        .append ( 'rect‘ )
        .attr ( 'class‘ ,‘bars‘ )
        .attr ( 'fill‘ , "slategray“ )
        .attr ( 'x‘ , function(d){ …
```

Update scales to match new data:

Because the contents of your data array are changing, you also need to reset the x axis domain each time the data updates (otherwise your new bars don’t have a position on the x axis)

Add a class to x axis in d3.csv function

Move scaleX domain def’n to drawPoints( )

Select the axis group by its class, and call the axis function

```
d3 .selectAll ( ' .xaxis ‘ )
      .call ( d3 .axisBottom ( scaleX ) );
```

Add transitions!

Add a transition to the update pattern to make the object constancy more noticeable:

```
rects
        .transition ( )
        .duration ( 200 )
        .attr ( 'x‘ , function ( d ) {….
```

Linked charts ?

http://square.github.io/crossfilter/