Why do the dots fly in from the origin?

You haven't set coordinates before drawing. If this is not a desired feature, duplicate the positioning code immediately after appending the circle (and before drawing)

This is more obvious because I put a transition on the points:

```
.transition ( )
.ease ( d3 .easeSin  )
.duration ( 400 )
```

Review code in switchToRealData

Lines 57 and 63: variable name passing (global, local)

Separating drawing function (drawPoints) from data update (buttonClicked) – why do this?

By the end of the day, move update function outside of buttonClicked and into an updateData function instead; makes it more general.

For now, start where we ended last week, and build toward a more general system step by step.

Switch over to real data: Median income for men and women by age

Change d3.csv file name

Change year values to match new data (2000 and 2016)

Update scales: use scalePoint ( ) for x; map age group onto x axis position. Use the values from the .csv file, and list them as *strings* ("16-19", etc)

Update titles: 'Weekly median income by age', 'age group' and 'income (dollars)'

Adjust the drawing

Split the drawing into two series (one for women, one for men); change both the enter ( ) and the update section of the code.

X position is determined by d.age
Y position is determined by d.women (or men, respectively)
Fill color and radius are uniform across a series (can be moved into enter ( ) section of the code, since not changing with the data)

Adding more years: Automate the separation using d3.nest

Switch to updateAllYears folder

Update the .csv link to use allYears data file

Replace the manual year arrays using d3.nest:

```
nestedData = d3.nest ( )
      .key ( function ( d ) { return d.year } )
      .entries ( dataIn );

console.log (
            nestedData.filter ( function ( d ) {
                        return d.key == "2016“
             } ) [ 0 ] .values
);
```

Can use filtered data to populate data bind array

Adding more years: Making a more general update function

Create new global variables nestedData and yearData (empty arrays)

Make an updateData function, which takes a year value as an input and returns the filtered data:

```
function updateData ( selectedYear ) {

        return  nestedData .filter ( function ( d ) {
                return d.key == selectedYear
        } ) [ 0 ] .values;
}
```

Call updateData and drawPoints on button click:

```
newData = updateData ( '2016‘ ) ;

drawPoints ( newData ) ;
```

Adding more years: Replace button with a slider

Replace the HTML button with a slider:

< input type = “range“ min ="2000“ max ="2016“ value = "1"
class = "slider" id="mySlider“ oninput = “ sliderMoved(value)” >

(if you only want the chart to update when the user lets go of the slider, use onchange instead of oninput)

Change the name of the buttonClicked function in JS, allow it to accept an input, and use that input in the updateData call:

```
function sliderMoved ( value ) {

        newData = updateData ( value );
        drawPoints ( newData );

}
```

Combine with setInterval to auto advance

Use setInterval to update the data.

```
var year = 2000;

setInterval ( function ( ) {

    newData = updateData ( year );
    drawPoints ( newData );

    //Increment the year until it hits 2016, then reset to 2000.
    if ( year < 2016 ) {
        year++
    }
    else {
        year = "2000";
    }

}, 1000);
```

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
  },
```

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.

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.

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

Free coding session

1) Work on incorporating your final project data into this structure. You can use my file of countryData if you don't have final project data yet.

   I recommend that you go back to a simple version, and then add the pieces in one by one, rather than starting with the complicated one all at once

2) If you don't have your final project data, then start compiling it

3) Open consultation time for other questions, problems, or concerns about your final project

Final project: Sketches crit

Briefly introduce your project:

- topic
- what you're hoping to do
- specific questions you'd like to ask
- sketches to show what you're currently thinking about