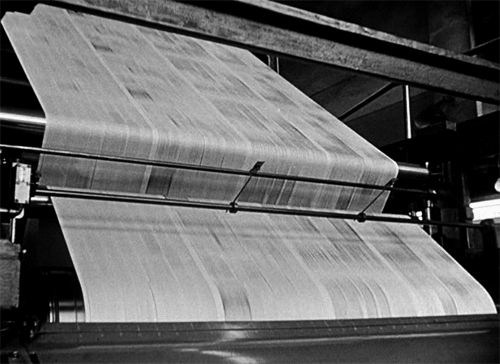
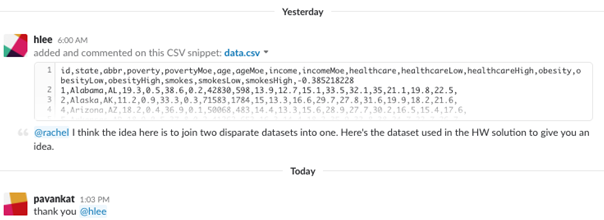
# **Unit 16 | Assignment - Data Journalism and D3**

****

*[via GIPHY](https://giphy.com/gifs/newspaper-press-v2xIous7mnEYg)*

## **Tip One:**



## **Background**

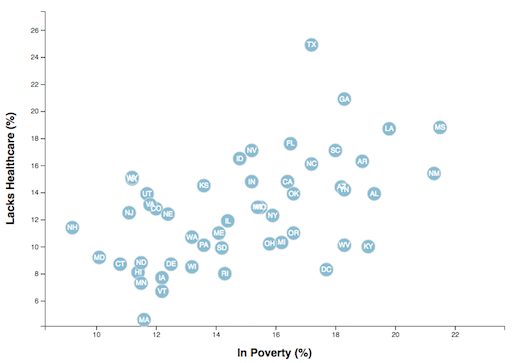
Welcome to the newsroom! You've just accepted a data visualization position for a major metro paper. You're tasked with analyzing the current trends shaping people's lives, as well as creating charts, graphs and interactives to help readers understand your findings.

The editor wants to run a series of feature stories about the health risks facing particular demographics. She's counting on you to sniff out the first story idea by sifting through the latest information from the U.S. Census Bureau and the Behavioral Risk Factor Surveillance System.

The beat reporters will go out and investigate the relationship you find, sourcing experts and finding anecdotes to back up your research. That's tough work, and they won't be happy about digging for a story that doesn't exist—you'll need to find a convincing correlation before you even think about pitching your first article. The editor also wants you to make a scatter plot to show the correlation—you *are* the data visualizer, after all.

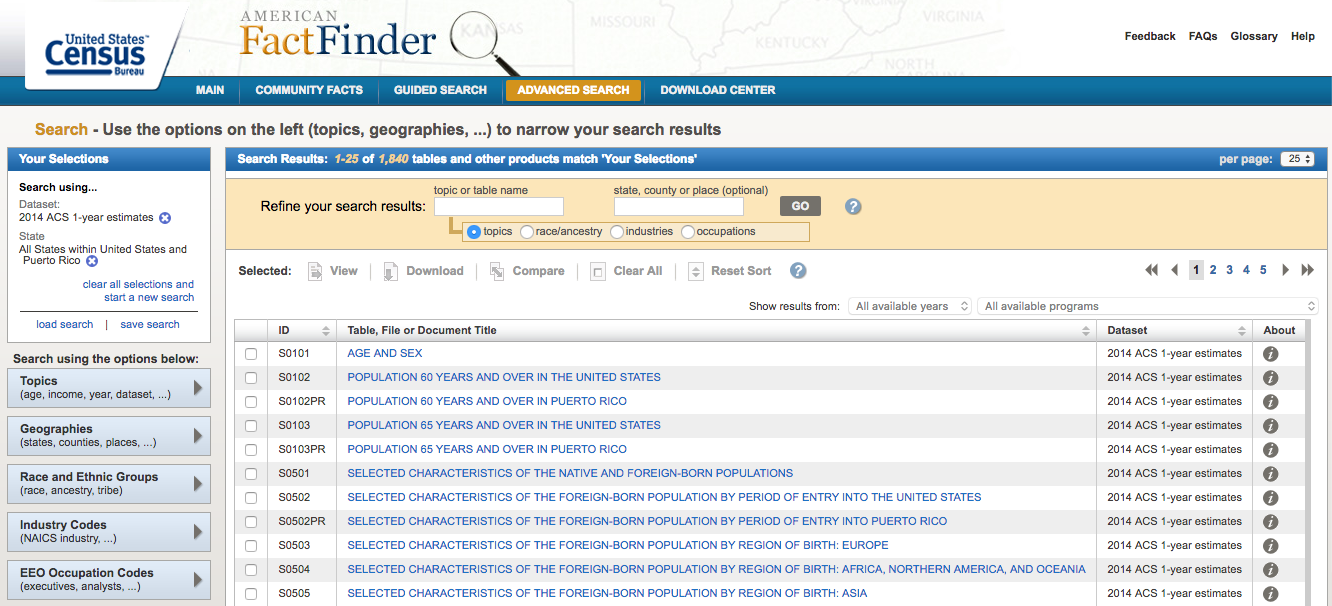
## **Your Task**

### **Level 1: D3 Dabbler**

****

You need to find a correlation between two data variables, each measured state by state and taken from different data sources. You'll then visualize the correlation with a scatter plot and embed the graphic into an .html file. You need to accomplish these four steps.

#### **1. Find the Data**

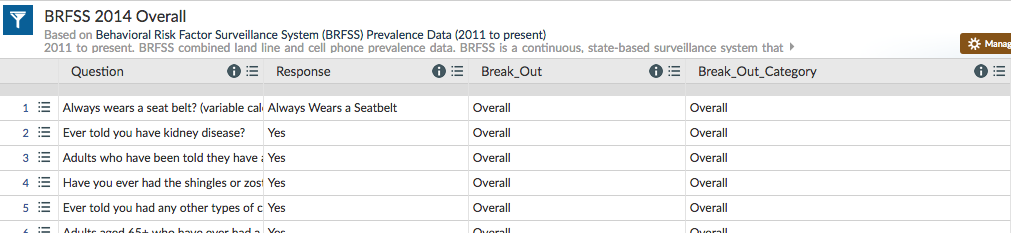
****

Look for demographic information using the 2014 one-year estimates from the U.S. Census Bureau's American Community Survey. You can specify your information using the [American FactFinder](http://factfinder.census.gov/faces/nav/jsf/pages/searchresults.xhtml) tool. When searching through the data, be sure to select these options in the left sidebar:

* Topics -> Dataset -> 2014 ACS 1-year estimates
* Geographies -> Select a geographic type -> State - 040 -> All States within United States and Puerto Rico

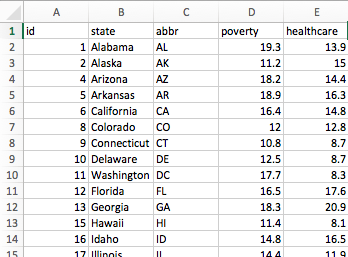
When you select those filters, use the search bar to chose the demographic of your choice, or browse through the entries already shown. Click the data that interests you and then download the .csv file.

Next, you'll search for data on health risks using 2014 survey data from the [Behavioral Risk Factor Surveillance System](https://chronicdata.cdc.gov/Behavioral-Risk-Factors/BRFSS-2014-Overall/5ra3-ixqq). Note that we already filtered the data by year and break-out—you just need to find the behavioral risk you want to use. Filter the Question data on the site before downloading a specified .csv, or simply download the whole .csv file and use Excel's filtering tools.

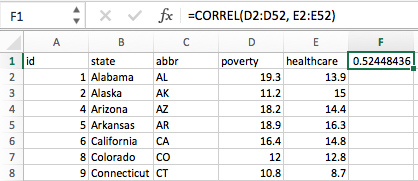


#### **2. Format and Test the Data**

Let's format your data for D3. With your two data types chosen, grab the value columns from each and paste them into a new Excel document. Create header names that you can easily call with JavaScript (concise, lowercased, camelCased). Make sure that your rows and columns line up—You may need to delete Guam from your datasheet so that your Census and BRFSS data matches.



To make sure you have a solid trend, you need to test for correlation with Excel's =CORREL() function. Aim for a value either less than -0.5 or more than 0.5—these values would indicate a moderate correlation and a story that might be worth pursuing (shoot for -0.75 or 0.75 if you're feeling super diligent).



* If you don't find a value that matches, try at least four other demographic-risk combinations—if you can't find one that hits -0.5 or .5, just go with the most striking mix.

When you find a suitable match, delete any correlation cells from your sheet and save the file as data.csv. Place it in the data folder of your homework directory and move onto the next step.

#### **3. Visualize the Data**

Using the D3 techniques we taught you in class, create a scatter plot that represents each state with circle elements. You'll code this graphic in the app.js file of your homework directory—make sure you pull in the data from data.csv by using the d3.csv function. Your scatter plot should ultimately appear like the image at the top of this section.

* The x-values of the circles should match the demographic census data, while the y-values should represent the risk data.
* Include state abbreviations in the circles.
* Create and situate your axes and labels to the left and bottom of the chart.
* Generate this chart in the d3.html file in your assignment directory.
* Note: You'll need to use http-server to display the graphic since you're pulling data in from a source outside of your app.js file.

#### **4. Embed into an iframe**

When your d3.html displays the graphic just as you'd like it to, embed it in index.html with an iframe. Add a quick written analysis of your data below the graphic.

* You'll notice that at certain screen sizes, the iframe may cut off sections of your scatter plot. Don't worry about this unless you want to take on the final challenge.

**By Mike Bostock: how do you create a label to display the 'name' of each node over the circle?**

Typically by adding an svg:text element. Something like this in terms of SVG:

  <svg:g class="node" transform="translate(d.x, d.y)">  
    <svg:circle r="5"/>  
    <svg:text x="10" dy=".31em" text="[d.name](http://d.name/)"/>  
  </svg:g>

Or, in D3:

  var g = svg.selectAll("g.node")  
      .data(nodes)  
    .enter().append("svg:g")  
      .attr("class", "node")  
      .attr("transform", function(d) { return "translate(" + d.x + ","  
+ d.y + ")"; })  
      .call(force.drag);

  g.append("svg:circle")  
      .attr("r", 5);

  g.append("svg:text")  
      .attr("x", 10)  
      .attr("dy", ".31em")  
      .text(function(d) { return [d.name](http://d.name/); });

This does have a drawback, though, which is that later nodes' circles  
will be drawn over earlier nodes' text labels. Also, there's a nice  
technique where you draw the text twice, first with a thick white  
stroke, such that the text has a white halo that improves legibility.  
I used this in yesterday's example of mobile patent suits:

<http://bl.ocks.org/1153292>

This involves adding two different svg:g layers: one for the circles,  
and one on top for the labels. In terms of SVG:

  <svg:g>  
    <svg:circle class="node" cx="d.x" cy="d.y" r="5"/>  
    <svg:circle class="node" cx="d.x" cy="d.y" r="5"/>  
    …  
  </svg:g>  
  <svg:g>  
    <svg:text class="label" x="d.x + 10" y="d.y" dy=".35em" text="[d.name](http://d.name/)"/>  
    <svg:text class="label" x="d.x + 10" y="d.y" dy=".35em" text="[d.name](http://d.name/)"/>  
    …  
  </svg:g>

In terms of D3, first you add the svg:g for the circles:

  var circle = svg.append("svg:g").selectAll("circle")  
      .data(nodes)  
    .enter().append("svg:circle")  
      .attr("r", 6)  
      .call(force.drag);

Then a second svg:g for the labels:

  var text = svg.append("svg:g").selectAll("g")  
      .data(nodes)  
    .enter().append("svg:g");

  text.append("svg:text")  
      .attr("x", 8)  
      .attr("y", ".31em")  
      .attr("class", "shadow")  
      .text(function(d) { return [d.name](http://d.name/); });

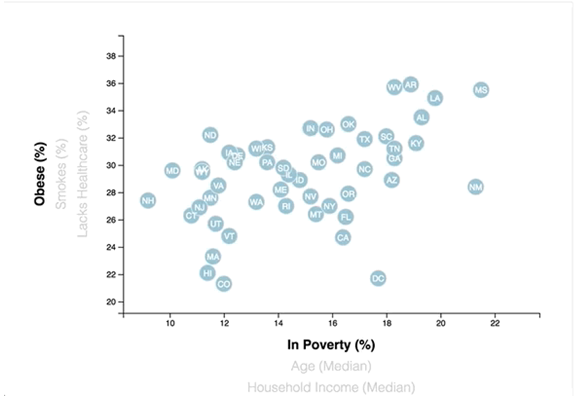
  text.append("svg:text")  
      .attr("x", 8)  
      .attr("y", ".31em")  
      .text(function(d) { return [d.name](http://d.name/); });

It's a bit more work because you're mapping the nodes twice (once for  
the circles, and once for the labels). Versus in the simpler example,  
you map once to an svg:g element that contains both a circle and a  
label. But the benefit of the second approach is more legible text,  
which is a big plus.

Mike

### **Level 2: Impress the Boss**

Why make a static graphic when D3 lets you interact with your data?



#### **1. More Data, More Dynamics**

You're going to include two more demographics and two more risks. Place additional labels in your scatter plot and give them click events so that your users can decide which data to display. Animate the transitions for your circles' locations as well as the range of your axes.

* Hint: Try binding all of the .csv data to your circles. This will let you easily determine their x or y values when you click the labels.
* Note: Don't worry about the correlation values for all your variable combos. Focus on the graphic’s interactivity for this stage (unless you're feeling super, *super* diligent).

<http://code.hazzens.com/d3tut/lesson_4.html>

08-Stu\_Hair\_Metal\_Solved, 09-Stu\_Transitions\_Solved,

10-Hair\_Metal\_Modularize

<http://bl.ocks.org/1153292>

**D3 Selection:**

svg.selectAll(".class1, circle, #id2")

**multiple classes on an element** - .attr("class", "axis-text x-axis") \*\* this has an axis-text and x-axis class so I can select either two only get the x-axis or axis-text to get all elements. Add attr("class", "axis-text x-axis") to x axes labels and attr("class", "axis-text y-axis") for the y axes.

Transitioning Text:

<https://bl.ocks.org/mbostock/7004f92cac972edef365>

<https://bl.ocks.org/mbostock/f7dcecb19c4af317e464>

<https://bost.ocks.org/mike/transition/>

d3.selectAll("text")

.append("text")

.style("opacity", 0)

.transition()

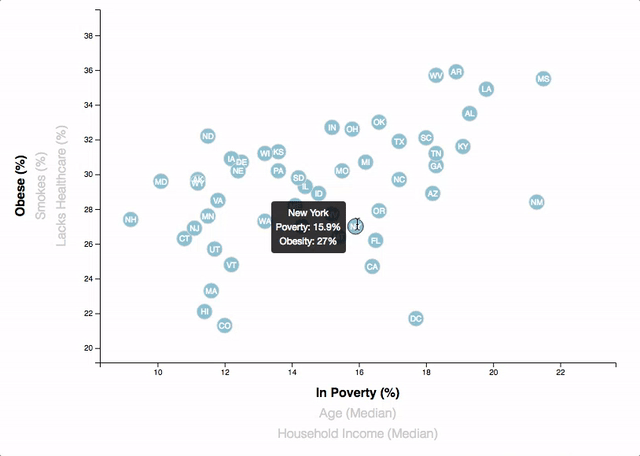
.style("opacity", 1)

.transition()

.delay(1500);

#### **2. Incorporate d3-tip**

While the ticks on the axes allow us to infer approximate values for each circle, it's impossible to determine the true value without adding another layer of data. Enter tooltips: developers can implement these in their D3 graphics to reveal a specific element's data when the user hovers their cursor over the element. Add tooltips to your circles and display each tooltip with the data that the user has selected. Use the d3-tip.js plugin developed by [Justin Palmer](https://github.com/Caged)—we've already included this plugin in your assignment directory.

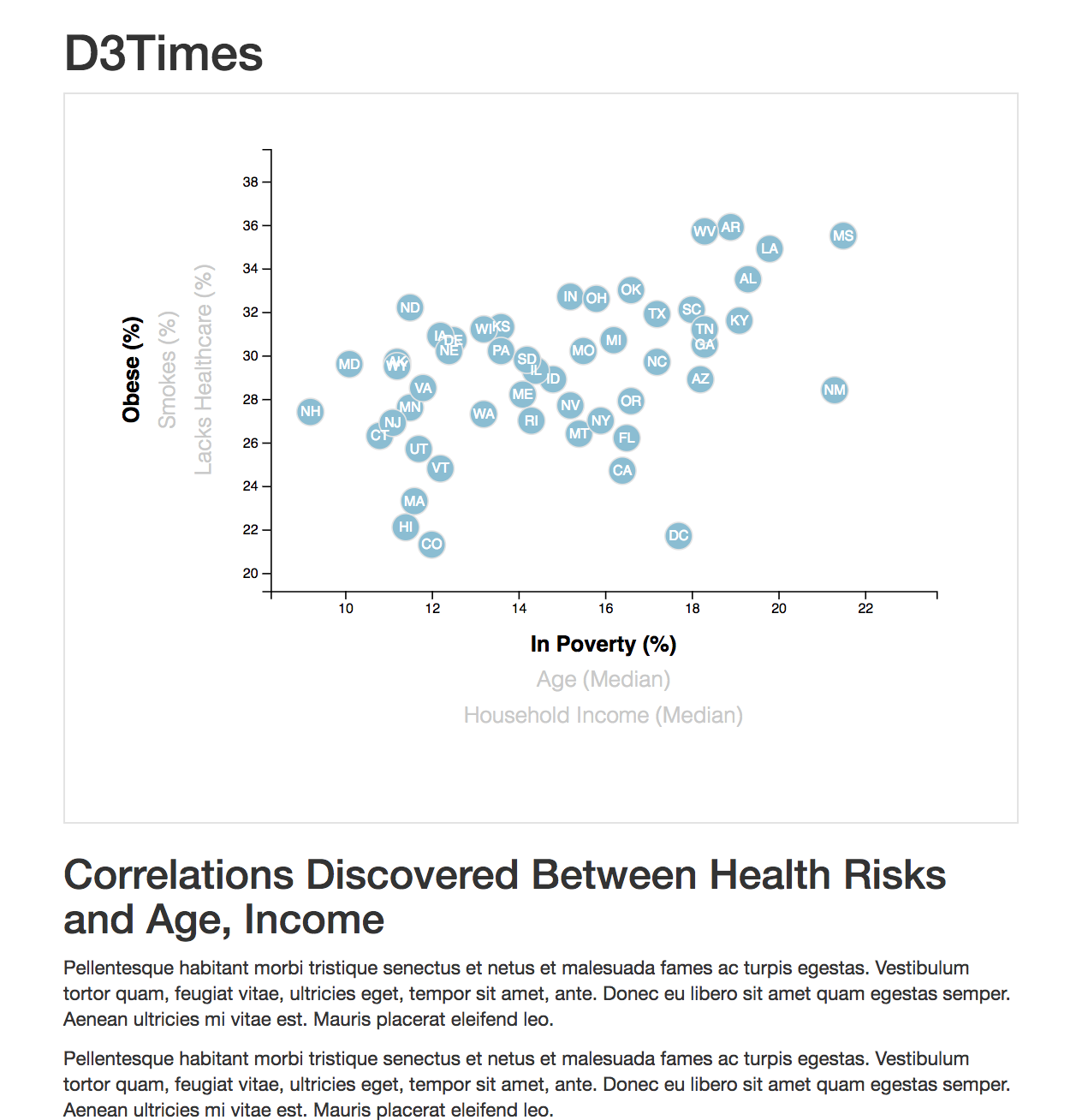


* Check out [David Gotz's example](https://bl.ocks.org/davegotz/bd54b56723c154d25eedde6504d30ad7) to see how you should implement tooltips with d3-tip.

### **Level 3: Mobile Moxie (Tough, but Rewarding)**

Sure, there's plenty of desktop readers out there, but what about tablet and phone users? How would you optimize your scatter plot so that its data is accessible to all users? This will be one of the toughest tasks when making most D3 graphics—but taking it on will also help you stand out as a thorough data visualizer.

Add a function to your scatter plot that changes the look of your svg whenever the size of your window/iframe changes. D3 lacks a one-size-fits-all approach to mobile since every graph will look different depending on the topic and developer. You'll need to decide what attributes to morph for whenever the screen size changes. Make sure that the graph is visible in the index.html iframe.



Visit these links for advice on how to accomplish this:

* [Responsive Charts with D3 - Chris Amico](http://eyeseast.github.io/visible-data/2013/08/28/responsive-charts-with-d3/)
* [Responsive D3 article from Safari Books Online](https://www.safaribooksonline.com/blog/2014/02/17/building-responsible-visualizations-d3-js/)
* Note: Some of the code in these articles will be outdated, like the axis code—each was written before D3 4.0. Regardless, the core concept remains the same: build a function to resize your chart and fire it with d3.select(window).on('resize', [name of your function]).

### **Assessment**

Your final product will be assessed on the following metrics:

* Completion of all steps in chosen level
* Coherency of scatter plot (labels, ticks)
* Visual attraction
* Professionalism

Good luck!

## **Copyright**

Coding Boot Camp (C) 2016. All Rights Reserved.