Lab Assignment 5: Interactive Data Visualization

Due: 11:59 PM Feb 18, Wednesday

In this lab, you will learn the basics of D3.js, another popular tools for interactive visualization. It's better if you have a background knowledge of HTML, CSS, JavaScript. An excellent 10 minutes overview of these fundamentals is on this website, alignedleft.com, by Scott Murray. To be more proficient with D3.js, you are encouraged to go through the tutorial by the same person,visualization with D3.js(time estimates: 10-15 hours).

1 D3.js Tutorial

This tutorial illustrates how to plot circles, bind data with the plots, plot word cloud.

A simple example of plotting circles

1. Open a brower, e.g., Chrome, FireFox or IE. Chrome is the best.
2. Open the web page main page.
3. Stay on the page, open `Chrome -> More tools -> Developer Tool`. Or click `F12` button on the keyboard to open `Developer Tool`. You will see the `Elements, Sources, Console, ...` tabs. Click on `Elements` tab. If you mouse hovers on any tag, the corresponding content will be highlighted on the web page with a blue background. Internal CSS styles are included within `<style></style>` tag under the `<head>`. External CSS sheets are linked in the head too.
4. Select a DOM element using plain JavaScript. In the console, type `document.getElementById(‘footer’);`, you will get a DOM element returned. DOM stands for **D**ocument **O**bject **M**odel.
5. Select a DOM element using D3.js. First, copy all the text on D3.js page in the console. Second, type `var header = d3.select(‘#header’);`, you will get an array of DOM nodes. Third, change the header of the page by typing `header.select(‘h1’).text(‘Hello World!’);`; change the header background color by typing `header.select(‘h1’).style(‘background-color’, ‘green’);`
6. Draw a dot with D3. D3 draws **S**calable **V**ector **G**raphics, which is a text-based image format. First, define a variable called `var svg = d3.select(‘#footer’).append(‘svg’);`. Second, set the width and height attributes of the SVG canvas by typing `svg.attr( ‘width’,60).attr( ‘height’,50)`. Third, draw a circle on the canvas by typing `var circle = svg.append(‘circle’);`. Forth, you can also set the attributes of the circle to specify its appearance, type `circle.attr( ‘cx’,’25).attr( ‘cy’,’25’).attr( ‘r’,22)` to see the circle in the top right corner. Continue to set `circle.attr(‘fill’,‘blue’).attr(‘stroke’,‘gray’).attr(‘stroke-width’,’2’);`.

(Notice the difference between `attr()` and `style()`). For more appearance of circles, here is the reference. How D3 SVG place your plots? See SVG primer.

7. D3 data binding. Suppose you need to draw 5 dots, with radius of each specified by a different value. Would you append 5 circles by typing the above codes 5 times? Is there any easier way? Yes. Let's do it. First, reload the page to clear what you did before. Then copy all the text again on D3.js page to include D3.js. Now, draw a bigger canvas by typing `var svg = d3.select('#footer').append('svg'); svg.attr( 'width',500).attr( 'height',80);`

8. D3 data binding continued. First, define the radius array, `var radius= [10, 15,20, 25, 30];`. Then, declare 5 place holders on the canvas for the 5 circles in the future, typing `var circles = svg.selectAll('#varycircle').data(radius).enter()`. Third, for each place holder, append an circle node, type `circles = circles.append('circle');`. Forth, binding the data with the plot appearance, type `circles.attr( 'r', function(d) {return d;})`. Fifth, specify the x and y position so that the 5 circles are next to each other, `circles.attr('cx',function(d,i){return (i*50) + 25; }`. It's the same syntax to specify any other appearance, as if you are manipulating one circle at a time. To specify y position, `circles.attr('cy',30)` or `circles.attr('cy',function(d){return 30;}`. (Data types you should know in JavaScript, which are kind of similar to Python.)

More complicated word cloud example [1]

The plot in Figure 1 shows the word cloud of the sample tweets data. The size of each word is proportional to the word count of all the tweets. You can see the basketball is with the largest size, that is because the example twitter data is crawled by keyword 'basketball'. The program is set to show up to 500 words. We also see kobe, LSU, and some video links be the top 500 most frequent words.

Now, we illustrate how to plot the word cloud based on your own data.

1. Reload the page to clear what you did before. Then copy all the text again on D3.js page to include D3.js.

2. To plot word cloud, you need to write hundreds of code, but the good news is someone already did it. Let's copy and paste all the text into the console on the page word cloud auxiliary functions. Now, let's call those functions to create a simple word cloud.

3. Open the link word cloud generation code. You can copy and paste all the code directly to the console. When you press enter, you will see a word cloud for a sample text. The last two lines specify a string variable `var textdata` and call the `load()` function. Try to modify the `textdata` to another string and call `load()` function again.

2 Assignments

In Lab 2, you downloaded tweets data from different locations, e.g., Philadelphia and Boston.

1. Show the word clouds of tweets respectively that you crawled from two places.

2. Show the word cloud of the most recent two days of the two locations. You can preprocess data in Python to create the files needed for the word cloud program.

[1] Based on Jason David's code

3. Show any word cloud that might interest you.
4. Write up your discoveries.

3 Extra Credits

Now, you've get the basic ideas of D3.js, it's time to explore yourself.

Three are a lot of awesome projects or applications based on

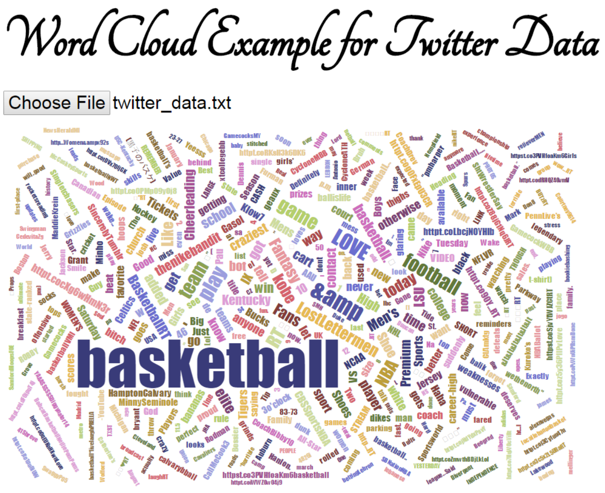

Figure 1: Word cloud for tweets data