Old Dominion Univeristy

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 4:20pm - 7:10pm R, ECSB 2120

Assignment # 7

GeorgeC. Micros UIN: 00757376

Honor Pledge
I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagIiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed ________________________________
November 10, 2014

George C. Micros

Written Assignment 7

Fall 2014
CS 495: Introduction to Web Science
Dr. Michael Nelson

November 10, 2014

Contents

Chapter 1
Written Assignment 7

1.1 Question 1

1.1.1 The Question

Using D3, create a graph of the Karate club before and after the split.
- Weight the edges with the data from:
`http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/zachary.dat`
- Have the transition from before/after the split occur on a mouse click.

1.1.2 The Answer

This assignment was an opportunity for using the departments server to host a webpage. In the back of my mind I have wanted to devote some time into a small project that I can host on the server and make user of my student account. There is no bettter motivation than a deadline to force a proper work ethic to be born. The main challenge was finding out all the little details that are required to get a simple “hello world” page running. Once the general html file format was established teh example for the tutorials were useful in establishing some familiarity with the bizarre syntax and mechanics of D3. It resembles the written form of a series of mouse click, if one would ever be compelled to take notes on how to use a GUI and then make that into a programming language.

The main advance between the tutorial examples and the final product was learning how to load data files. The approach I used stores graphs in json files. These files contain lists of the nodes and edges as well as their respective attributes. These items can be then used to within the script to read data and dynamically produce items in the webpage, such as nodes and edges.

In order to generate the json formatted graph files I had to revisit assignment 6 again and reimplement some of the logic in python, rather than R which was the original approach. The networkx package has functions that generate json files of the graph. The following script meets this requirement by reading in the weighted graph gml and producting the correspoding json.

```python
#! /usr/bin/python


import json
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from networkx.readwrite import json_graph

g = nx.read_gml("weighted_karate.gml")

data = json_graph.node_link_data(g)
with open('wgraph.json', 'w') as f:
    json.dump(data , f, indent=4)
```

Listing 1: Python script to make JSON files

The HTML standard provides node and link objects that serve the purpose of creating graphs. These items can be dynamically created and populated based on the data with D3. The foloowing code segments illustrate the procedure to creating new nodes and new links.

```javascript
var node = svg.selectAll(".node")
      .data(graph.nodes)
    .enter().append("circle")
      .attr("class", "node")
      .attr("r", 10)
      .text(function(d){ return d.name})
      .style("fill", "gray")
```

Listing 2: Generating nodes based on data

```
var link = svg.selectAll(".link")
      .data(graph.links)
    .enter().append("line")
      .attr("class", "link")
      .style("stroke-width", function(d){return Math.log(d.value)})
      .style("stroke", function(d) { return lcolor(Math.exp(d.value) + 10)});
```

Listing 3: Generating links based on data

These code segments are enclosed in a function. The function is called when the page is laoded, but also when a mouse click is registered. Based on the mouse click that function creates the nodes and links with different color schemes to indicate the before and after split graph.

The full HTML file follows:

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

.node {
  stroke: #fff;
  stroke-width: 1.5px;
}

.link {
  stroke: #999;
  stroke-opacity: .6;
}

</style>
<body>
<p>Karate Klub</p>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script>

var width = 960,
    height = 500;

var color = d3.scale.category10();

var force = d3.layout.force()
    .charge(-600)
    .linkDistance(60)
    .size([width, height]);

var svg = d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);


var lcolor = d3.scale.linear()
  .domain([0,100])
  .range(["#FDFFCB", "#232942"])
//  .interpolate(d3.cie.interpolateLch);

var dataJson = ["wlgraph.json", "test.json", "e.json", "d.json"];
var k = 0;

d3.select("svg")
  .on("click", changer);

```

```
var json = dataJson[0];

function changer()
{
  k = k + 1
  json=dataJson[0];

  refresher();

}


function refresher() {

d3.json(json, function(error, graph) {
  svg.selectAll(".link").remove();
  svg.selectAll(".node").remove();
  d3.selectAll("#p1").remove()
  force
      .nodes(graph.nodes)
      .links(graph.links)
      .start();


  var link = svg.selectAll(".link")
      .data(graph.links)
    .enter().append("line")
      .attr("class", "link")
      .style("stroke-width", function(d){return Math.log(d.value)})
      .style("stroke", function(d) { return lcolor(Math.exp(d.value) + 10)});
if(k%2 ===0)
{
d3.select("body").append("p").html("Before").attr("id","p1");

  var node = svg.selectAll(".node")
      .data(graph.nodes)
    .enter().append("circle")
      .attr("class", "node")
      .attr("r", 10)
      .text(function(d){ return d.name})
      .style("fill", "gray")
      //.style("fill", function(d) { return color(d.club); })
      //.call(force.drag);
}
if(k%2 ===1)
{
d3.select("body").append("p").html("After").attr("id","p1")
  var node = svg.selectAll(".node")
      .data(graph.nodes)
    .enter().append("circle")
      .attr("class", "node")
      .attr("r", 10)
      .style("fill", function(d) { return color(d.club); })
      //.call(force.drag);
}



force.on("tick", function() {
    link.attr("x1", function(d) { return d.source.x; })
        .attr("y1", function(d) { return d.source.y; })
        .attr("x2", function(d) { return d.target.x; })
        .attr("y2", function(d) { return d.target.y; });

    node.attr("cx", function(d) { return d.x; })
```

```
        .attr("cy", function(d) { return d.y; });
  });
}



);
}
refresher();

</script>
```

Listing 4: HTML file of the karate club graph

The resulting webiste can be viewed at:
`http://www.cs.odu.edu/~gmicros/karate/`

References

1. Michael Bostock. Michael Bostock‘s D3 Tutorials. `https://github.com/mbostock/d3/wiki/Tutorials`.
2. Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309, 2011.
3. Jerome Cukier. Selections in D3. `http://www.jeromecukier.net/blog/2013/03/05/d3-tutorial-at-strata-redux/`, 2013.
4. Michael Bostock. Force-Directed Graph. `http://bl.ocks.org/mbostock/4062045`, 2012.
5. Christophe Viau. The Big List of D3.js Examples. `http://christopheviau.com/d3list/`.
6. NetworkX. `https://networkx.github.io/`.