CS595 Intro to Web Science, Assignment #7

Valentina Neblitt-Jones

November 7, 2013

Question 1

Using D3, create a graph of the Karate club before and after the split

Weight the edges with 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.

Answer to Question 1

To begin, I needed to get zachary.dat into a JSON format so I needed a dictionary with two lists inside. i read in each line - skipping the header data and the matrix showing mere connections - of the dat file and assigned it to person. Then I looped through the lines to create the "name" (we did not have their names so numbers were used). An second loop was needed in order to go through each line and access the weight. I needed to ignore the weights that equaled zero so only the weights of 1 or higher were included. I did not exclude the duplicate pairings (e.g. 1, 32, 4 and 32, 1, 4). My guess is that the duplicates would not cause a problem, but I remain unsure. I then used json.dumps to output the data in the JSON format. (Listing 1)

Listing 1: CreateJSONFile01.py

```
import json
import pprint

g = open('zachary1.json', 'w')
pp = pprint.PrettyPrinter()

adict = {}
adict["nodes"] = []
adict["links"] = []

id = 0 # name

with open('zachary.dat', 'r') as f:
   good = f.readlines()[41:]
for line in good:
        person = line.split()
        id = id + 1 # generates name
        adict["nodes"].append({'id':str(id)})
        for i in range(0, len(person)):
                weight = int(person[i])
                source = id - 1
                target = i
                if weight != 0:
                        adict["links"].append({'source': source, 'target': target, 'weight':
                            weight})

pp.pprint(adict)
output = json.dumps(adict, indent=4)
```

```
g.write(output)

f.close()
g.close()
```

The next step was to use the JSON file from the previous step in order to create the graph of the club after the split. I opened the file using json.load and used node_link_graph to turn the input into something that networkX could work with. In the previous assignment, we did something similar, but I implemented it in R so I needed to translate the steps from R to Python. Because we were emulating the group splitting into two groups, the while looped kept functioning until the number of connected components reached 2. First, I calculated the edge betweenness. Then I looped through the edge betweenness values while keeping track of the max edge betweenness value. Finally, it deleted the edge for the highest value until the connected components was equal to 2. Next, the result was converted back to data with node_link_data and outputted as JSON with json.dumps. (Listing 2)

Listing 2: CreateJSONFile02.py

```
import json
import pprint
import networkx
from networkx.readwrite import json_graph

pp = pprint.PrettyPrinter()

f = open('zachary1.json')
g = open('zachary2.json', 'w')

data = json.load(f)
# pp.pprint(data)
f.close()

G = json_graph.node_link_graph(data)

while networkx.number_connected_components(G) < 2:
        edgy = networkx.edge_betweenness_centrality(G)
        maxb = 0
        for edge in edgy:
                #print('key ' + str(edge) + ' value ' + str(edgy[edge]))
                if edgy[edge] > maxb:
                        maxb = edgy[edge]
                        maxedge = edge

        G.remove_edge(maxedge[0],maxedge[1])

result = json_graph.node_link_data(G)

g.write(json.dumps(result, indent=4))
g.close()
```

I followed an example in Scott Murray's book *Interactive Data Visualization*, which is how I ended up with the colorful unlabeled circles. He covered how to make one graph, but not how to do the transformation from one to the other on mouse click. So looking at Listing 3, I changed the height and width because the graphs would need more room in order to show the split. I added a transformation attribute to create the call for transformation. I load the original graph first. The transforming function needed to remove the original graph's elements before loading the new elements. The elements of the code that were true for both graphs are contained within d3.json(jsonfile, function(error, dataset). I adjusted force.gravity to a lower number since the split was not clear. The split components were initially still too close to each other to see that they had actually split apart. I could be mistaken, but adding labels to the circles seemed to lack support and perhaps I should have followed the Bostock example at `http://bl.ocks.org/mbostock/950642`. Figure 1 shows the club before the split and Figure 2 shows the club after the split. The animation demonstration can be found at `http://www.cs.odu.edu/~vneblitt/cs595/`.

Listing 3: index.html

```
<script type="text/javascript">

        //Width and height
        var w = 800;
        var h = 800;

         //Create SVG element
        var svg = d3.select("body")
                .append("svg")
            .attr("width", w)
            .attr("height", h)
            .on("click", transformit);

        loadjson('zachary1.json');

                function transformit() {
                        d3.selectAll("line").remove();
                        d3.selectAll("circle").remove();
                        loadjson('zachary2.json');

                }

                function loadjson(jsonfile) {

        d3.json(jsonfile, function(error, dataset) {


        //Initialize a default force layout, using the nodes and edges in
            dataset
        var force = d3.layout.force()
                .nodes(dataset.nodes)
            .links(dataset.links)
            .size([w, h])
            .linkDistance(100)
            .charge(-100)
            .gravity(0.01)
            .start();

        var colors = d3.scale.category10();

        //Create edges as lines
        var links = svg.selectAll("line")
                .data(dataset.links)
            .enter()
            .append("line")
            .style("stroke", "#ccc")
            .style("stroke-width", 1);

        //Create nodes as circles
        var nodes = svg.selectAll("circle")
            .data(dataset.nodes)
            .enter()
            .append("circle")
            .attr("r", 10)
            .style("fill", function(d, i) {
                return colors(i);
                })
            .call(force.drag);

        //Every time the simulation "ticks", this will be called
        force.on("tick", function() {

                links.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; });

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

```
                .attr("cy", function(d) { return d.y; });
        });
        });
        }
</script>
```

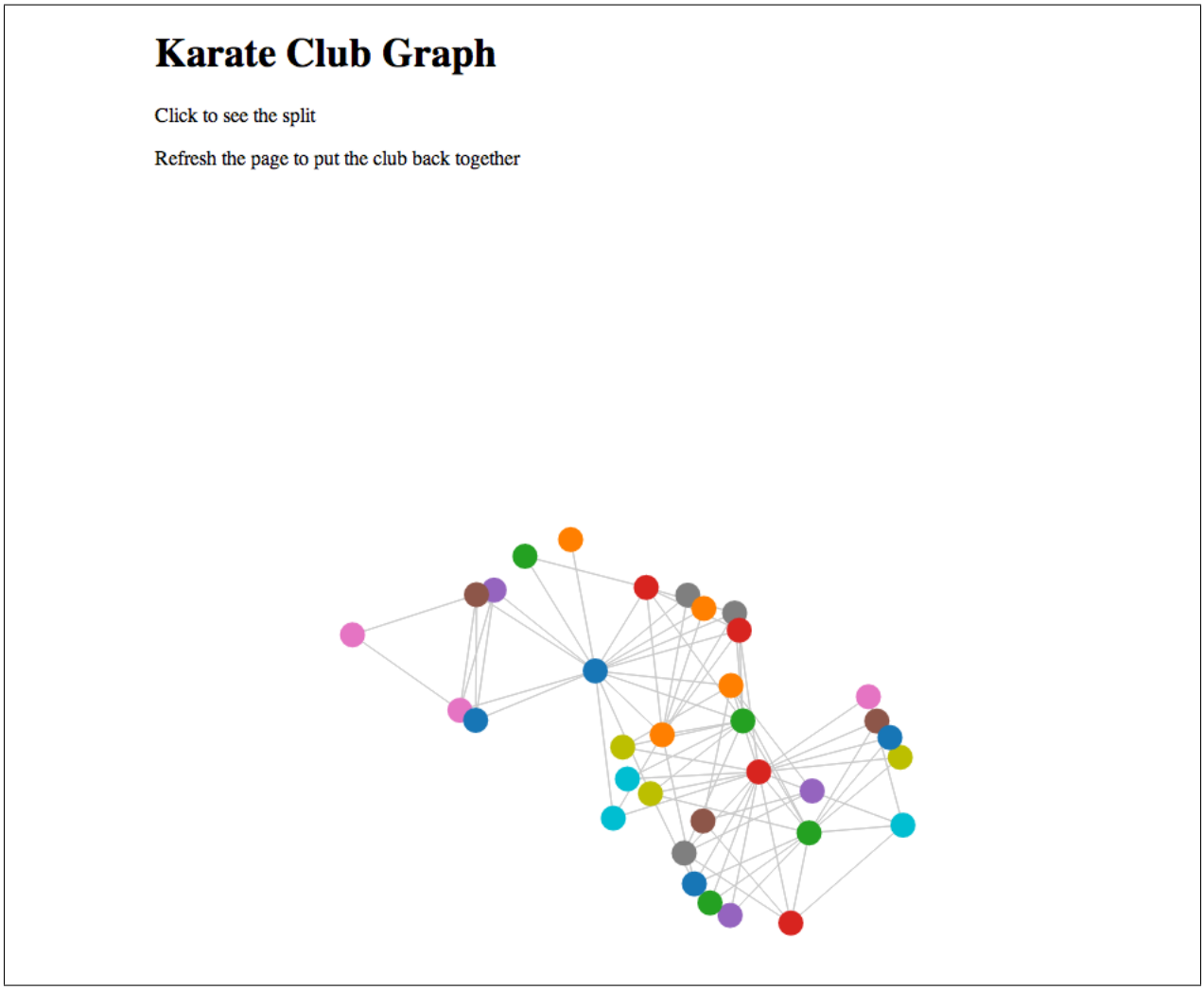

Figure 1: Karate Club Before Split

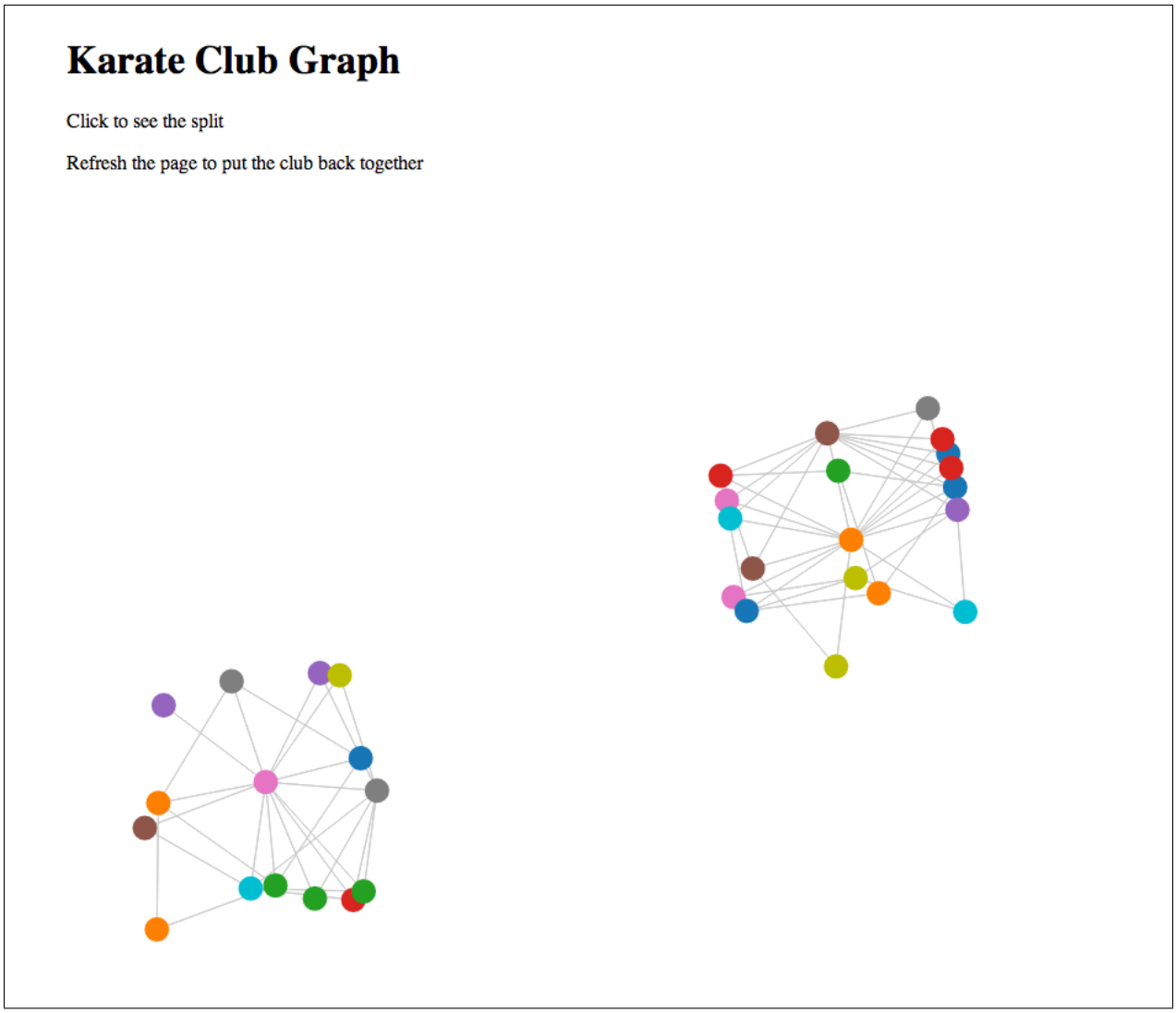

Figure 2: Karate Club After Split

Extra Credit, 3 Points

Use D3 to create a who-follows-whom graph of your Twitter account. Use my Twitter account (phone-dude_mln) if you do not have an interesting number of followers.

Answer to Extra Credit

Not attempted.

Resources

Note: Apologies. I did not have time to implement BibTex for this assignment, but I will on Assignments 8, 9, and 10.

- Bostock, Michael. Data-Driven Documents. http://d3js.org/
- Bostock, Michael. Labeled Force Layout. http://bl.ocks.org/mbostock/950642
- Bostock, Michael. Selections. https://github.com/mbostock/d3/wiki/Selections#wiki-on
- Bostock, Michael. mbostock/d3. https://github.com/mbostock/d3
- Bostock, Michael. mbostock/d3: API Reference. https://github.com/mbostock/d3/wiki/API-Reference
- Murray, Scott. About these tutorial. http://alignedleft.com/tutorials/d3/about
- Murray, Scott. Interactive Data Visualization: An Introduction to Designing with D3. Via Safari Books Online
- NetworkX developer team. NetworkX. http://networkx.github.io/
- NetworkX developer team. NetworkX: edge_betweenness_centrality. http://networkx.github.io/documentation/latest/reference/generated/networkx.algorithms.centrality.edge_betweenness_centrality.html#networkx.algorithms.centrality.edge_betweenness_centrality
- NetworkX developer team. NetworkX: Graph - Undirected graphs with self loops. http://networkx.github.io/documentation/latest/reference/classes.graph.html#networkx.Graph
- NetworkX developer team. NetworkX: node_link_data. http://networkx.lanl.gov/reference/generated/networkx.readwrite.json_graph.node_link_data.html#networkx.readwrite.json_graph.node_link_data
- NetworkX developer team. NetworkX: node_link_graph. http://networkx.lanl.gov/reference/generated/networkx.readwrite.json_graph.node_link_graph.html#networkx.readwrite.json_graph.node_link_graph
- NetworkX developer team. NetworkX: number_connected_components. http://networkx.github.io/documentation/latest/reference/generated/networkx.algorithms.components.connected.number_connected_components.html#networkx.algorithms.components.connected.number_connected_components
- PyMOTW. json: JavaScript Objection Notation Serializer. http://pymotw.com/2/json/
- Python.org. json: JSON encoder and decoder. http://docs.python.org/3.3/library/json.html
- Python.org. print: Data pretty printer. http://docs.python.org/2/library/pprint.html
- Stack Overflow. Parsing values from a JSON file in Python. http://stackoverflow.com/questions/2835559/parsing-values-from-a-json-file-in-python
- Stack Overflow. Skip the first couple of lines while reading lines in Python file. http://stackoverflow.com/questions/9578580/skip-first-couple-of-lines-while-reading-lines-in-python-file