Assignment 7

Fall 2014
CS595 Web Science
Dr. Michael Nelson

Mathew Chaney

November 5, 2014

Contents

List of Figures

Listings

1 Question 1

1.1 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.
```

1.2 Answer

Using an example from the primary author of the D3 JavaScript library, Mike Bostok [1], a graph was created using Zachary's Karate Club as the dataset. The edge weights were obtained from the pickled [2] dataset found at http://nexus.igraph.org/api/dataset_info?id=1&format=html. The D3 library provides a force-directed graphing layout [3], which was used to display the graph and to allow for a transition from the initial graph, shown in Figure 1, to the graph after the split of the karate club, shown in Figure 2.
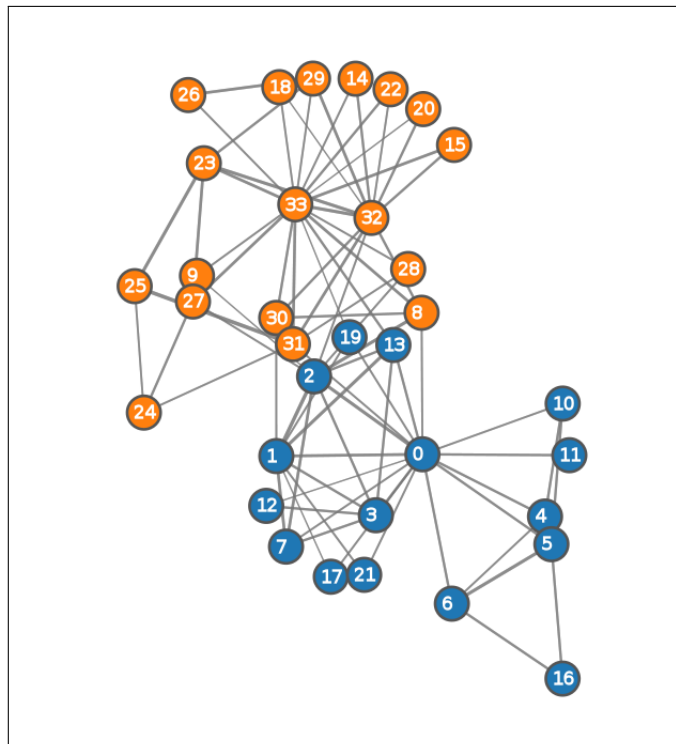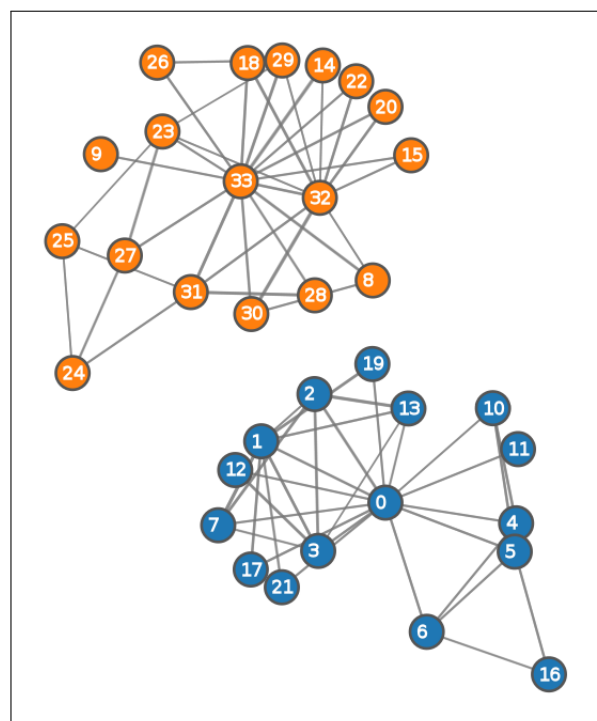

Figure 1: Initial Graph

Figure 2: Graph After Split

The dataset was first parsed into matrices using the `build_matrix` function, shown in Listing 1. These matrices were converted into python dictionary objects, which are pickled [2] into the json format [4]. This output was used as the input for the JavaScript code, which uses the D3 library [5] to create the graphs.

The python code to produce the json data is shown in Listing 1.

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

import pickle
import json

def build_matrix(raw, n):
    """Builds a matrix from a list of strings where each string
    is a space-separated row of the matrix"""
    matrix = [[0 for y in range(n)] for x in range(n)]
    for idx, line in enumerate(raw):
        for idy, val in enumerate(line.split()):
            matrix[idx][idy] = int(val)
    return matrix

def build_nodes(clubs, names):
    """Builds a list of nodes represented as a python dict"""
    return [{"id": i, "club": c, "name": names[i]} for i, c in enumerate(clubs)]

def build_links(egraph, cgraph):
    """Builds a list of links represented as a python dict"""
    links = []
    for idx, line in enumerate(egraph):
        for idy, val in enumerate(line):
            if val == 1:
                links.append({"source": idx, "target": idy, "value": cgraph[idx][idy]})
    return links

if __name__ == '__main__':
    data = pickle.loads(open('karate.pickle').read())['karate']
    clubs = data.vs['Faction']
    names = data.vs['name']

    # Read lines from input data file
    lines = [line.strip() for line in open('karate.txt').readlines()]

    # parse size of nxn matrix
    n = int(lines[1].split()[0].replace('N=', ''))

    # build matrices using input data
    egraph = build_matrix(lines[7:41], n)
    cgraph = build_matrix(lines[41:], n)

    m = {}
    m['nodes'] = build_nodes(clubs, names)
    m['links'] = build_links(egraph, cgraph)
    with open('out.json', 'w') as output:
        output.write(json.dumps(m, indent=1, separators=(',', ': ')))
```

Listing 1: Data Converter

The javascript code to produce the graph is shown in Listing 2.

```
27    var width = 960,
28        height = 700;
29
30    var color = d3.scale.category10();
31
32    var svg = d3.select("body").append("svg")
33        .attr("width", width)
34        .attr("height", height);
35
36    var force = d3.layout.force()
37        .charge(-400)
38        .gravity(0.1)
39        .linkDistance(90)
40        .size([width, height]);
41
42    d3.json("out.json", function(error, graph) {
43        var link = svg.selectAll(".link")
44            .data(graph.links)
45            .enter().append("line")
46            .attr("class", "link")
47            .style("stroke-width", function(d) { return Math.sqrt(d.value); });
48
49        var node = svg.selectAll(".node")
50            .data(graph.nodes)
51            .enter().append("g")
52            .attr("class", "node")
53            .call(force.drag);
54
55        node.append("circle")
56            .attr("r", 12)
57            .style("fill", function(d) { return color(d.club); });
58
59        node.append("text")
60            .attr("dx", -7)
61            .attr("dy", ".35em")
62            .text(function(d) { return d.id; });
63
64        function update() {
65            node.data(graph.nodes)
66                .exit().remove();
67            link.data(graph.links)
68                .exit().remove();
69            force.nodes(graph.nodes)
70                .links(graph.links)
71                .on("tick", tick)
72                .start();
73        };
74
75        function tick() {
76            link.attr("x1", function(d) { return d.source.x; })
77                .attr("y1", function(d) { return d.source.y; })
78                .attr("x2", function(d) { return d.target.x; })
79                .attr("y2", function(d) { return d.target.y; });
80
81            node.attr("transform", function(d) {
82                return "translate(" + d.x + "," + d.y + ")";
83            });
84        };
85
86        var done = false;
87        svg.on("click", function(d) {
88            if (!done) {
89                done = true;
90                graph.links = graph.links.filter(function(d) {
91                    return graph.nodes[d.source.id].club == graph.nodes[d.target.id].club;
92                });
93                update();
94            }
95        });
96        update();
97    });
```

Listing 2: Building the Graph

2 References

[1] Mike Bostok. Force-Directed Graph. http://bl.ocks.org/mbostock/4062045, 2012.

[2] The Python Software Foundation. Python pickle Module. https://docs.python.org/2/library/pickle.html, 2014.

[3] Mike Bostok. d3 - Force Layout. https://github.com/mbostock/d3/wiki/Force-Layout, 2014.

[4] European Computer Manufacturers Association. ECMA-404 The JSON Data Interchange Standard. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf, 2013.

[5] Mike Bostok. Data-Driven Documents. http://d3js.org/, 2014.