CS595 Assignment 7

Jon Robison

November 6, 2013

Q1.
Using D3, create a graph of the Karate club before and after the split. Have the transition from before/after the split occur on a mouse click.

See Appendix A for code to generate json given to d3
See Appendix B for d3 code

Figure 1: Before

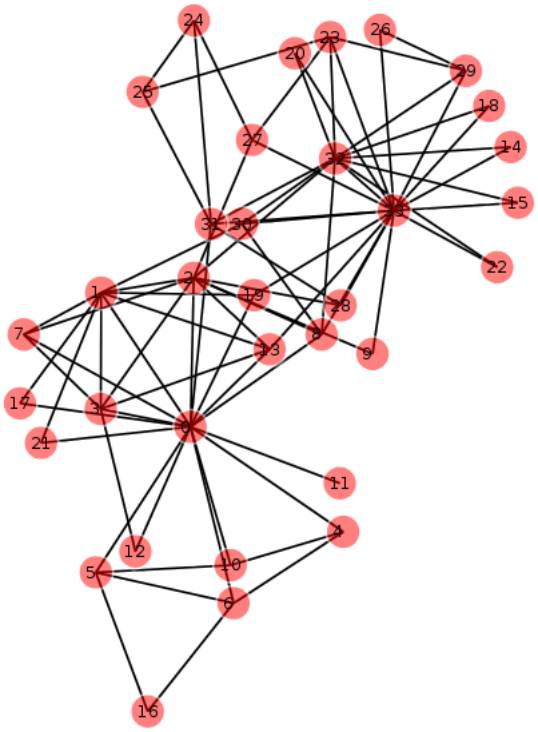
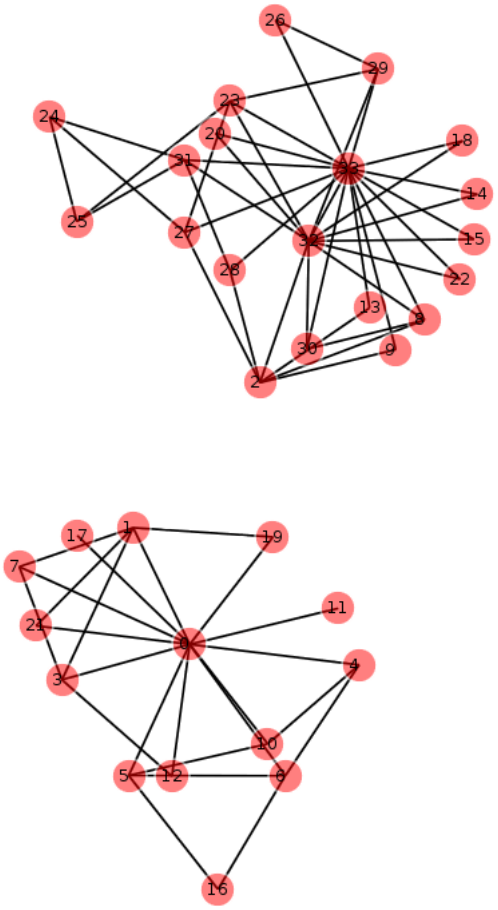


Figure 2: After



Appendix A

```
#!/bin/python3
import sys

HEADER_LINES=8
DEFAULT_INPUT='zachary.dat'
DEFAULT_OUTPUT='zachary.json'
if len(sys.argv) != 2:
    print('Pass the path to your input dat then output json file. ' +
        'Defaulting to input:' + DEFAULT_INPUT + ' output:' +
        DEFAULT_OUTPUT)
    input=DEFAULT_INPUT
    output=DEFAULT_OUTPUT
else:
    input=sys.argv[1]
    output=sys.argv[2]

def writeNodes(o, count):
    o.write('{\n  "nodes":[\n')
    for x in range(0, count+1):
        o.write('    {"name":"'+str(x)+'","group":'+str(x)+'}')
        if x != count:
            o.write(',')
        o.write('\n')
    o.write('  ],\n  "links":[\n')

with open(output, 'w') as o:
  with open(input) as f:
    lineNumber=0
    currentPerson=0
    for line in f:
        lineNumber += 1
        if(lineNumber == HEADER_LINES):
            total=len(line.strip().split(' '))-1
            print('Total entries: ' + str(total))
            writeNodes(o, total)
        elif(lineNumber > HEADER_LINES and
             lineNumber > HEADER_LINES + total):
            currentPersonConnections = []
            for connection in line.strip().split(' '):
                currentPersonConnections.append(int(connection))
            currentConnection=0
            for connection in currentPersonConnections:
                if currentConnection > currentPerson and connection > 0:
                    o.write('    {"source":' + str(currentPerson) + ',
```

```
                    "target":' + str(currentConnection) + ',"value":' +
                    str(connection)+'},\n')
            currentConnection+=1
        currentPerson+=1
o.write('  ]\n}')
```

Appendix B

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

.link {
  stroke: #000;
  stroke-width: 1.5px;
}

circle {
  fill: #F00;
  stroke: #fff;
  stroke-width: 1.5px;
  opacity: .5;
}
circle:hover{
  opacity: 1;
}
text {
  fill: #000;
  font: 12px sans-serif;
  pointer-events: none;
}
</style>
<body>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script>

var width = 960,
    height = 800;

var color = d3.scale.category10();

var nodes = [],
    links = [];

var force = d3.layout.force()
    .nodes(nodes)
    .links(links)
    .charge(-400)
    .linkDistance(100)
    .size([width, height])
    .on("tick", tick);
```

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

var link = svg.selectAll(".link");

d3.json("zachary.json", function(error, data) {
  data.nodes.forEach(function(d, i) {
      d.id = i;
      nodes.push(d);
  });
  data.links.forEach(function(d, i) {
      links.push(d);
  });
  start();
})

var clicked = false
svg.on("click", function(d){
  if(!clicked){
    clicked = true
    links.splice(52,1);
    links.splice(34,1);
    links.splice(25,1);
    links.splice(24,1);
    links.splice(23,1);
    links.splice(20,1);
    links.splice(19,1);
    links.splice(16,1);
    links.splice(15,1);
    links.splice(11,1);
    links.splice(7,1);
    links.splice(1,1);
    start();
  }
})

function start() {
  link = link.data(force.links(), function(d) {
    if(!isNaN(d.source)){
      return d.source + "-" + d.target;
    }else{
      return d.source.id + "-" + d.target.id;
    }
  });
  link.enter().insert("line", ".node").attr("class", "link");
```

```
  link.exit().remove();

  var node = svg.selectAll(".node")
      .data(nodes)
      .enter().append("g")
      .attr("class", "node")
      .call(force.drag);
  node.append("circle")
    .attr("r", 13);
  node.append("text")
      .attr("dx", -8)
      .attr("dy", ".35em")
      .text(function(d) { return d.id.toString(); });

  force.start();
}

function tick() {
  var node = svg.selectAll(".node")
  node.attr("transform", function(d) {
    return "translate(" + d.x + "," + d.y + ")";
  });

  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; });
}
</script>
```