CS-432/532 Introduction to Web Science:
Assignment #6:
Data Visualization

Due on Thursday, March 17, 2016

Dr. Michael L. Nelson

Plinio Vargas
pvargas@cs.odu.edu

Contents

List of Figures

Listings

List of Tables

Problem 1

D3 graphing (5 points)

Use D3 to visualize your Twitter followers. Use my twitter account ("@phonedude_mln") if you do not have >= 50 followers. For example, @hvdsomp follows me, as does @mart1nkle1n. They also follow each other, so they would both have links to me and links to each other.

To see if two users follow each other, see: `https://dev.twitter.com/rest/reference/get/friendships/show`

Attractiveness of the graph counts! Nodes should be labeled (avatar images are even better), and edge types (follows, following) should be marked.

Note: for getting GitHub to serve HTML (and other media types), see: `http://stackoverflow.com/questions/6551446/can-i-run-html-files-directly-from-github-instead-of-just-viewing-their-source`

Be sure to include the URI(s) for your D3 graph in your report.

1.1 Approach

The Twitter account for **Jose Antonio Olvera**, whom follows Dr. Nelson, was used as our data-set $\boldsymbol{S}$ for problem 1. This problem was divided into two sub-problems: extracting the data and plotting the graph.

1.1.1 Extracting the data

Listing 1 is a python program implemented to extract the data. It accomplishes the following:

a. Obtains S's followers Twitter's ID account using tweepy API [1]. Lines 10-16

b. All IDs, including S, are placed into an array $\boldsymbol{A}$. Lines 20-21.

c. Iterates through all elements in A, extracting its followers and other important information, such as name, screen name, number of friends, etc. Followers of each element are placed into an array $\boldsymbol{B}$. Lines 28-56

d. Only B's friends in A are needed, then $B = A \cup B$. Lines 36-39.

e. The collection of all B arrays are saved into a JSON file. Lines 59-61.

Listing 1: TwitterAPI.py

```
10 auth = tweepy.OAuthHandler('w11PMTOUlj5sGIKjnSII7AcT9', '4
       if8m3WWrWvHFZ4wHKRXAgrb7mBuRBRboHKUDwoT4AfuoL97hB')
11
12 api = tweepy.API(auth)
13
14 # screen_name = 'phonedude_mln'
15 screen_name = 'joc7188'
```

```
try:
    user = api.get_user(screen_name)
except tweepy.TweepError as e:
    if isinstance(e, tweepy.TweepError):
        print('RateLimitError... Wait 15min')
    sys.exit(1)

print(user.screen_name, user.id)
ids = [f for f in user.followers_ids()]

ids.append(user.id)    # add user_id to comparison list
user_friends = {}      # dictionary keeping all followers and user nodes
print(user.followers_count, ids)

counter = 0
excluded = []

for id_ in ids:
    friend = api.get_user(id_)
    print(counter, friend.name, friend.profile_image_url, friend.id, end=' ---> ')
    counter += 1
    user_connected_friends = []

    try:
        friend_followers = [f for f in api.get_user(friend.id).followers_ids()]
        for connected in friend_followers:
            if connected in ids:
                print(connected, end=',')
                user_connected_friends.append(connected)
        print()
        user_friends[friend.id] = {'name': friend.name, 'avatar': friend.
            profile_image_url,
                                   'screen_name': friend.screen_name, '
                                       followers_count': friend.followers_count,
                                   'friends_count': friend.friends_count, '
                                       connected_to': user_connected_friends}

    except tweepy.TweepError as e:
        print('\n This is -->', e)
        if isinstance(e, tweepy.TweepError):
            time.sleep(60 * 15)

            try:
                # retry after 15 mins.
                friend_followers = [f for f in api.get_user(friend.id).
                    followers_ids()]
                for connected in friend_followers:
                    if connected in ids:
                        print(connected, end=',')
                        user_connected_friends.append(connected)
                print()
                user_friends[friend.id] = {'name': friend.name, 'avatar': friend.
                    profile_image_url,
```

Problem 1 continued on next page...

```
64                                          'screen_name': friend.screen_name, '
                                                followers_count': friend.
                                                followers_count,
65                                          'friends_count': friend.friends_count,
                                                'connected_to':
                                                user_connected_friends}
66                  continue
67
68              except tweepy.TweepError as e:
69                  print('\n This is Other error in line 62-->', e)
70                  excluded.append(friend.id)
71
72          else:
73              print('\n This is Other error in line 68 -->', e)
74              excluded.append(friend.id)
```

At the time our data-set was retrieved, its size was of 53 elements. There is a request number limitations of 15, to extract Twitter's user information. Lines 50-52 catch that error and wait for 15 minutes to continue with the information extraction.

Some Twitter users may have privacy settings in their accounts, so their information cannot be retrieved through the API. See section Excluded Data. Lines 68-74.

1.1.2 Plotting the Graph

Although we have the data from 1.1.1 to plot the graph, it does not have detailed information on how nodes link. < *ReshapeData.py* > accomplishes this job. It creates two JSON dictionaries within a file: **nodes**, containing all element characteristics and **links**, a source pointing to a target. It also adds a new characteristic to the node: color. It determines how close the friendship is between any node with $\boldsymbol{A}$.

Figure 1: Color Scheme

Figure 1 represents a color scheme showing how similar any particular node with data-set A is in relation to its followers.

Listing 2: ReshapeData.py

```
46
47 """
48 Write to file
49 """
50 outputfile = 'jose.json'
51 with open(outputfile, 'w') as file:
52     file.write('{\n\t"nodes": [\n')
53     for record in data:
54         friendship = len(data[record]['connected_to']) / network_size
55         if friendship > 0.20:
56             friend_color = colors[0]
57         elif friendship > 0.14:
58             friend_color = colors[1]
59         elif friendship > 0.10:
60             friend_color = colors[2]
61         elif friendship > 0.04:
62             friend_color = colors[3]
63         else:
64             friend_color = colors[4]
65
66         print(friendship)
67         file.write('\t\t{\n\t\t\t"id": "%s",\n\t\t\t"name": "%s",\n\t\t\t"
               followers_count": "%s",'
68                 '\n\t\t\t"friends_count": "%s",\n\t\t\t"screen_name": "%s",\n\t
                    \t\t"avatar": "%s",'
69                 '\n\t\t\t"color": "%s"\n\t\t},\n' %
```

```
70              (record, data[record]['name'], data[record]['followers_count'], data
                    [record]['friends_count'],
71               data[record]['screen_name'], data[record]['avatar'], friend_color))
72      file.write('\t],\n\n\t"links": [\n')
73
74      for record in data:
75          for link in data[record]['connected_to']:
76              if str(link) in cross_idx:
77  #                file.write('\t\t{"source": %s, "target": %s, "type": "followed-by
        ", "value": %d},\n' %
78  #                           (cross_idx[str(record)], cross_idx[str(link)], 1))
79                  file.write('\t\t{"source": %s, "target": %s, "type": "followed-by
                        ", "value": %d},\n' %
80                             (str(record), str(link), 1))
81      file.write('\t]\n}')
```

In order to make the graph appealing in terms of friendship colors, some tests were conducted to capture different similarity levels. As a result, if any particular node has:

a. If $> 20\%$ of N followers are also following A, then N is color coded as having similar followers as A.

b. If node N has $> 14\%$ and $< 20\%$ of A followers, then N is color coded as having most of its followers following A.

c. If node N has $> 10\%$ and $< 14\%$ of A followers, then N is color coded as having some its followers following A.

d. If node N's has $> 4\%$ and $< 14\%$ of A followers then N is color coded as having few followers following A.

e. If node N's has $< 4\%$ of A followers, then N is color coded as very few of N's followers following A.

The color code scheme is implemented in lines 55-64 of $<$ *ReshapeData.py* $>$. The rest of the code just dump the data into ***jose.json***, which is the input file for our Data-Driven Document. **The color code scheme is not accurate in an English grammatical context, but it gives a quick view of how close the followers are in the network**

Data Visualization:
Various features from different D3JS sites were used to enhance graph appearance.

Zooming and dragging	`http://bl.ocks.org/mbostock/6123708` [2]
Mouseover Tip-tool	`http://bl.ocks.org/Caged/6476579` [3]
Directed graph	`http://bl.ocks.org/mbostock/1153292`[4]
Mouseover link highlight	`http://p.migdal.pl/wizualizacja-wolnych-lektur/polish_books_themes.html` [5]
D3 Markers	`http://bl.ocks.org/dustinlarimer/5888271` [6]

D3 Markers obtained from [6] defines a relationship direction between two nodes using an arrow. An arrow pointing to a node means the target is a source's follower. This feature is included as D3-JavaScript in the html page. See below:

Listing 3: Placing D3 Markers in d3directed.hmtl

```
75
```

```
76              // Per-type markers, as they don't inherit styles.
77              svg.append("defs").selectAll("marker")
78                  .data(["followed-by", "licensing", "resolved", "center"])
79                  .enter().append("marker")
80                  .attr("id", function(d) { return d; })
81                  .attr("viewBox", "0 -5 10 10")
82                  .attr("refX", function(d) {if (d == "center") return 45; else
                        return 30; })
83                  .attr("refY", -1.5)
84                  .attr("markerWidth", 3)
85                  .attr("markerHeight", 3)
86                  .attr("orient", "auto")
87                  .append("path")
```

The attribute "RefX" line 82 points out how far from the target the arrow is going to be placed. We use this attribute to make a larger separation between any regular node and the main node A.

Mouseover link highlight obtained from [5] highlights the links and nodes related to the node when the mouse hovers over it. This feature modifies the CSS characteristic of link-nodes by iterating over all the nodes in the graph, filtering the modifications to those having a relationship with the node where the mouse is hovering. Below D3-JavaScript coding to create this effect:

Listing 4: Highlighting Links and Nodes in d3directed.hmtl

```
152             //-------------------------------------------------
153             //              mouse-over
154             //-------------------------------------------------
155             var mouseover = function(z){
156                 tip.show(z);
157                 var neighbors = {};
158                 neighbors[z.index] = true;
159
160                 path.filter(function(d){
161                     if (d.source == z) {
162                         neighbors[d.target.index] = true
163                         return true
164                     } else if (d.target == z) {
165                         neighbors[d.source.index] = true
166                         return true
167                     } else {
168                         return false
169                     }
170                 })
171                     .style("stroke-opacity", 1);
172
173                 circle.filter(function(d){ return neighbors[d.index] })
174                     .style("stroke-width", 3);
175
176                text.filter(function(d){ return !neighbors[d.index] })
177                     .style("fill-opacity", 0.2);
178
179                text.filter(function(d){ return neighbors[d.index] })
180                     .style("font-size", 16 + "px")
181
```

```
182             };
```

Directed Graph is the main D3 template obtained from [4] to generate our solution. The major difference between the template and our solution is the incorporation of extra features not included in the template, and using an outside data source file instead of being embedded with in the JavaScript. The data is loaded at the beginning for the external JSON file: jose.json. See below:

Listing 5: Loading JSON file into d3directed.hmtl

```
19          d3.json("jose.json", function(error, json) {
20            if (error) return console.warn(error);
21            var data = json['links'];
22            var vertices = json['nodes'];
23            visualize(data, vertices);
24          });
```

Mouseover Tip-tool is a mouse-over effect feature obtained from [3] that pops information of a node as the user hovers the mouse over it. The function is included in the mouseover effect, adding a "CSS" attribute to our DOM object to include the desired node information:

Listing 6: Poping Node info in d3directed.hmtl

```
99              var tip = d3.tip()
100                 .attr('class', 'd3-tip')
101                 .offset([-10, 0])
102                 .html(function(d) {
103                     var index = vertices.findIndex(x => x.id==d.id);
104                     return "<strong>Name:</strong> <span style='color:yellow'>" +
                            vertices[index].name + "</span><br>" +
105                         "<strong>Pseudo:</strong> <span style='color:yellow'>"
                                + vertices[index].screen_name + "</span><br>" +
106                         "<strong>Followers:</strong> <span style='color:yellow
                            '>" + vertices[index].followers_count + "</span><br>
                            " +
107                         "<strong>Following:</strong> <span style='color:yellow
                            '>" + vertices[index].friends_count + "</span><br>"
                            +
108                         "<img src='" + vertices[index].avatar + "'>";
109                 });
```

Zooming and dragging is an interesting and useful effect obtained from [2], but at the point when this document was edited, it only worked in Mozilla FireFox Web-Browsers. The description of its operation is listed below:

Listing 7: Zooming and Dragging d3directed.hmtl

```
199             function zoomed() {
200               svg.attr("transform", "translate(" + d3.event.translate + ") scale("
                      + d3.event.scale + ")");
201               slider.property("value",  d3.event.scale);
202             }
203
```

```
204             function dragstarted(d) {
205               d3.event.sourceEvent.stopPropagation();
206               d3.select(this).classed("dragging", true);
207             }
208
209             function dragged(d) {
210              d3.select(this).attr("transform", transform);
211             }
212
213             function dragended(d) {
214               d3.select(this).classed("dragging", false);
215             }
216
217             function slided(d){
218               zoom.scale(d3.select(this).property("value"))
219                 .event(svg);
220             }
```

The DOM graph object is modified as the mouse while is been rotated. A good and detailed explanation of this effect can be found at [7]

1.2 Excluded Data

Table 1: Jose Olvera Twitter Followers Excluded Data

ID	NAME	Screen-Name
509427317	Paulo Carrillo	panicape
271081991	monsorcas	monsorcas

Values shown in Table 1 are followers of Jose Olvera Twitter's Account, but were not included because these accounts have a privacy setting and cannot be extracted without authentication.

1.3 Solution

Figure 2: Jose A. Olvera Followers Network Graph

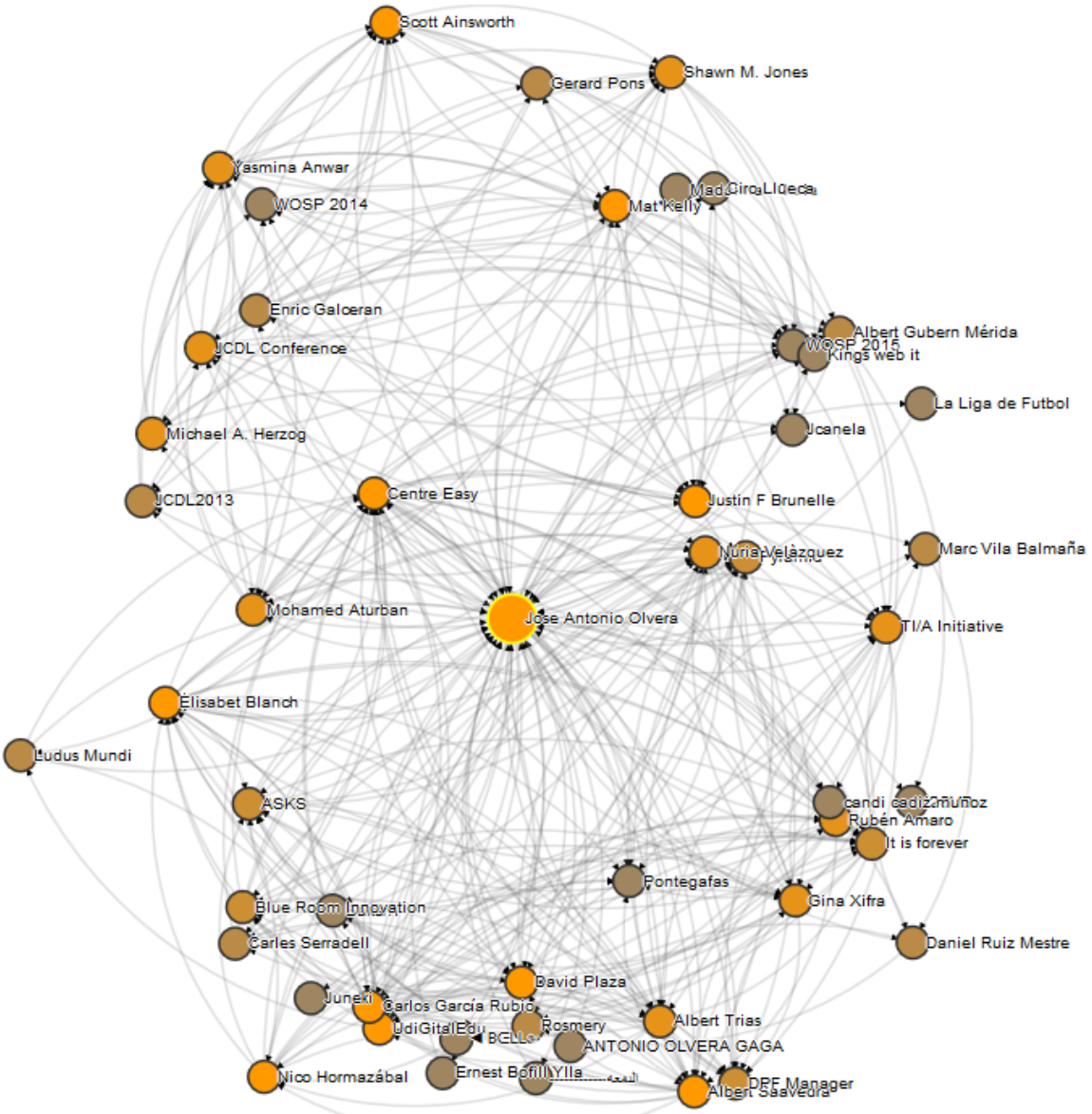

Figure 2 shows (A) Jose A. Olvera's Followers' Twitter Network Graph featuring a color scheme representing how similar the followers of any particular node are with A followers. Node A (Jose Antonio Olvera) has a larger radius and a yellow stripe around to distinguish it from the rest of the nodes in the graph. **URI:** `http://test-mysite.us/d3directed.html`

URI:`http://test-mysite.us/d3directed.html`

Problem 2

Gender homophily in your Twitter graph (5 points)

Take the Twitter graph you generated in question #1 and test for male-female homophily. For the purposes of this question you can consider the graph as undirected (i.e., no distinction between "follows" and "following"). Use the twitter name (not "screen name"; for example "Michael L. Nelson" and not "@phonedud_mln") and programatically determine if the user is male or female. Some sites that might be useful:

https://genderize.io/
https://pypi.python.org/pypi/gender-detector/0.0.4

Create a table of Twitter users and their likely gender. List any accounts that can't be determined and remove them from the graph.

Perform the homophily test as described in slides 11-15, Week 7.

Does your Twitter graph exhibit gender homophily?

2.1 Approach

We we used the same data generated in problem 1 to solve this problem. However, we filtered the data set by injecting the first name of each node with gender_detector python library. The iteration is shown in the listing below:

Listing 8: GetGender.py

```
20 detector = GenderDetector('us')
21
22 with open("jose.json", 'rb') as file:
23     data = json.load(file)
24
25 counter = 0
26 unknown = []
27 with open("jose-gender.data", 'w') as file:
28     for name in data['nodes']:
29         counter += 1
30         try:
31             print(counter, name['name'].split()[0].encode('ascii'), detector.guess
                   (name['name'].split()[0]), name['id'])
32             file.write('%s, %s, %s\n' % (str(name['id']), name['name'].split()[0].
                   encode('ascii'), detector.guess(name['name'].split()[0])))
33         except:
34             print(counter, name['name'].split()[0], 'unknown')
35             file.write('%s, %s, unknown\n' % (str(name['id']), name['name'].split
                   ()[0].encode('utf-8')))
```

Problem 2 continued on next page...

The result file can be seen on Table 3. To complete our graph and generate a solution we use the same approach as in 1.1.2, but we added an extra field to the nodes: gender, and we placed all linking edges into a set, thus eliminating the extra edge between two nodes having a bi-directional relationship. Since the JSON generation file approach is the same as in the previous problem, only where they differ will be pointed out.

Listing 9: GenerateGender.py

```
52 counter = 0
53 linked_nodes = set()
54 for node in data['links']:
55     if str(node['target']) in gender_nodes and str(node['source']) in gender_nodes
       :
56         if (node['source'], node['target']) not in linked_nodes and (node['target'
           ], node['source']) not in linked_nodes:
57             counter += 1
58             print('\t\t{"source": %s, "target": %s, "type": "followed-by"},\n' %
59                   (node['source'], node['target']), end='')
60             linked_nodes.add((node['source'], node['target']))
61 print('\t]\n}', end='')
```

As mentioned before, a set object was created in line 53 and we checked in both sides of the linked nodes (lines 55-56) to verify if the relationship exists.

Data Visualization:

We used the same approach as in 1.1.2, the difference is our color scheme was based on two colors: male (blue) and female (pink).

2.2 Excluded Data

Table 2: Gender Graph Excluted Data

ID	First-Name	Gender
467244543	It	unknown
22826489	Shawn	unknown
1919132468	Ludus	unknown
3094776599	WOSP	unknown
20786017	Nico	unknown
4255657283	Blue	unknown
204062965	Rubén	unknown
320655858	Centre	unknown
2986305078	JCDL	unknown
49576758	Enric	unknown
606550203	JCDL2013	unknown
3200537140	TI/A	unknown
506173851	ASKS	unknown
80852622	La	unknown

ID	First-Name	Gender
1364325193	Juneki	unknown
861118038	Jcanela	unknown
3271976762	Kings	unknown
256914642	Pyramid	unknown
2341223632	unicode	unknown
299702944	UdiGitalEdu	unknown
2585451218	I2CVB	unknown
4070742673	unicode	unknown
2512551954	WOSP	unknown
3200493898	DPF	unknown
295664883	Núria	unknown
898245566	Pontegafas	unknown
2220440436	unicode	unknown

Gender for data-set above were not able to be recognized by the python library gender_detector

2.3 Solution

2.3.1 Table of Twitter Users Likely Gender

Table 3: Likely Gender for Jose Olvera's Twitter Followers

ID	First-Name	Gender
120440596	David	male
467244543	It	unknown
3787923975	candi	female
22826489	Shawn	unknown
387831845	Michael	male
117034312	Ciro	male
1919132468	Ludus	unknown
1511472330	Mohamed	male
479894950	Mat	male
3094776599	WOSP	unknown
397587860	Rosmery	female
199226328	Marc	male
373298265	Gina	female
551758619	Carlos	male
367167714	Ernest	male
20786017	Nico	unknown
551038996	Albert	male
4255657283	Blue	unknown
204062965	Rubén	unknown
320655858	Centre	unknown
2986305078	JCDL	unknown
49576758	Enric	unknown
365952591	Jose	male
606550203	JCDL2013	unknown
32747579	Scott	male
3200537140	TI/A	unknown

ID	First-Name	Gender
857229727	ANTONIO	male
180473295	Justin	male
48938481	Albert	male
506173851	ASKS	unknown
80852622	La	unknown
862861328	Daniel	male
1364325193	Juneki	unknown
861118038	Jcanela	unknown
3271976762	Kings	unknown
256914642	Pyramid	unknown
2341223632	unicode	unknown
299702944	UdiGitalEdu	unknown
3082680478	Carles	male
2585451218	I2CVB	unknown
4070742673	unicode	unknown
2512551954	WOSP	unknown
3200493898	DPF	unknown
282698602	Elisabet	female
48842695	Gerard	male
295664883	Núria	unknown
114584669	Albert	male
898245566	Pontegafas	unknown
11938602	Madalina	female
2220440436	unicode	unknown
221460308	Yasmina	female

Most likely gender was generated utilizing python library gender_detector

Problem 2 continued on next page. . .

2.3.2 Gender Graph URI

Figure 3: Gender Graph for Jose A. Olvera Twitter Followers

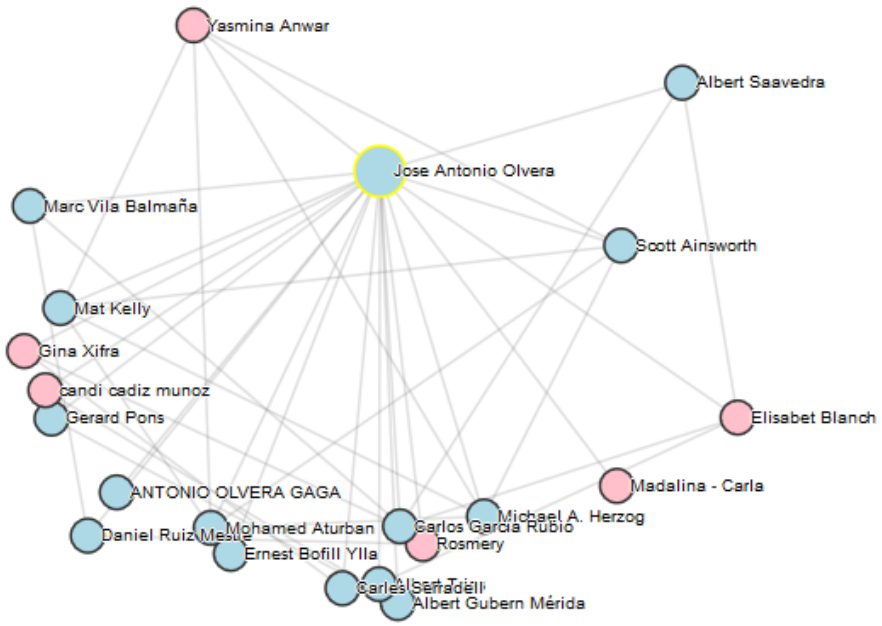

Figure 3 visualize gender relationship among Jose Olvera Followers. URI: `http://test-mysite.us/d3undirected.html`

URI: `http://test-mysite.us/d3undirected.html`

2.3.3 Test for Gender Homophily

To test our graph for gender homophily, we implemented < *homophily.py* >. The result of running the program is shown below:

```
Starting Time: Wed, Mar 16, 2016 at 21:50:36

Number of females: 6
Number of males: 18
p: 0.750
q: 0.250
2pq: 0.375

Cross-gender edges:
-------------------

Elisabet Blanch <--> David Plaza
```

```
Yasmina Anwar <--> Michael A. Herzog
Yasmina Anwar <--> Mohamed Aturban
Yasmina Anwar <--> Mat Kelly
Jose Antonio Olvera <--> Rosmery
Daniel Ruiz Mestre <--> Rosmery
Marc Vila Balmana <--> Rosmery
Carlos Garcia Rubio <--> Gina Xifra
Jose Antonio Olvera <--> Gina Xifra
Carles Serradell <--> Gina Xifra
Elisabet Blanch <--> Carlos Garcia Rubio
Elisabet Blanch <--> Albert Trias
candi cadiz munoz <--> Jose Antonio Olvera
Yasmina Anwar <--> Jose Antonio Olvera
Madalina - Carla <--> Jose Antonio Olvera
Elisabet Blanch <--> Jose Antonio Olvera
Yasmina Anwar <--> Scott Ainsworth
Yasmina Anwar <--> Justin F Brunelle
candi cadiz munoz <--> Albert Gubern Merida
Albert Saavedra <--> Elisabet Blanch

Summary of Cross-gender edges: 20 out of 52
Percentage of Cross-gender edges 0.385

End Time: Wed, Mar 16, 2016 at 21:50:36
Execution Time: 0.00 seconds
```

As we can see 2pq $<$ cross-edges, $0.375 < 0.385$ $\therefore$ there is no evidence of homophily.

Problem 3

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. This is a toggle, so the graph will go back and forth between connected and disconnected.

References

[1] Tweepy API. (n.d.) Retrieved March 10, 2016, from http://docs.tweepy.org/en/latest/api.html

[2] Zooming and dragging. (n.d.) Retrieved March 10, 2016, from http://bl.ocks.org/mbostock/6123708

[3] d3-tip tool. (n.d.) Retrieved March 10, 2016, from http://bl.ocks.org/Caged/6476579

[4] Directed graph. (n.d.) Retrieved March 10, 2016, from http://bl.ocks.org/mbostock/1153292

[5] Polish Books. (n.d.) Retrieved March 10, 2016, from http://p.migdal.pl/wizualizacja-wolnych-lektur/polish_books_themes.html

[6] D3 Markers. (n.d.) Retrieved March 10, 2016, from http://bl.ocks.org/dustinlarimer/5888271

[7] Explaining D3 Zooming and Dragging. (n.d.) Retrieved March 13, 2016, from http://stackoverflow.com/questions/21344340/sematic-zooming-of-force-directed-grap