LONDON GANG NETWORK ANALYSIS

SOCIAL NETWORK ANALYSIS ASSIGNMENT, WINTER SEMESTER 2020-21

Stefanos Kypritidis 8170050

Table of Contents

Introduction – Network Choice

The network that was chosen to be analyzed is a London street gang. First of all, London as most of the major cities has a plethora of gangs operating at its regions. In fact, according to theguardian.com London has become the global capital of money-laundering and the beating heart of European organized crime. In addition, it is reported that around 5,000 criminal gangs are operating in the UK alone. That means a huge amount of lawlessness in London and the UK stems from the existence of gang organizations.

Coming to the dataset on its own, it was found in Sites Google and it includes information on co-offending in a London-based inner-city street gang from 2005 till 2009. Moreover, the data comes from anonymized police arrest and conviction data for all the confirmed members of the gang. Furthermore, the data showcases the relationships between gang members. In detail, there are 4 different kind of relationships: just hanging out together, co-offending together, co-offending together serious crimes and co-offending together serious crimes and being relatives. Last but not least, there is also information about the age, birthplace, residence, arrests, convictions, prison, music and ranking of each gang member.

Looking at the dataset from a graph analysis perspective, each node represents a member of the street-gang and each edge represents the type of relationship that there is between two gang members (if there is any). By creating a network of all the gang-member we can get the whole picture of the gang. As a result, we can draw a variety of conclusions regarding the organization of the gang, popularity of members, roles of members, importance of members, homophily in the gang, triadic closure, cliques and communities. Also, we can find out a lot about the most dangerous members and their partners in crime. Furthermore, by including the different birthplaces of each member we can make assumptions on the race of the member and try to find out patterns on the crime tendency and crime associations between races. In addition, by using clustering for different attributes of the members, it is possible to create profiles of gang members and thus get valuable insights for each profile. Lastly, the goal is to gain more insight and a better understanding in gangs in general by scrutinizing this particular case.

Having the option to choose any dataset I pleased, I quickly realized that I wanted to analyze something different, more exciting and mysterious. Finding myself in tons of options online, it was not easy to decide which network to analyze. However, I have always been interested in gangs and the bonds and relationships that are formed between the members, which can be characterized as family relations and sometimes even more than family. Nonetheless, every criminal activity connected to a gang is criticized and should be penalized accordingly. To sum it up, I decided to analyze and throw light on this street-gang dataset because it definitely caught my attention and made me feel like a detective trying to solve a crime puzzle.

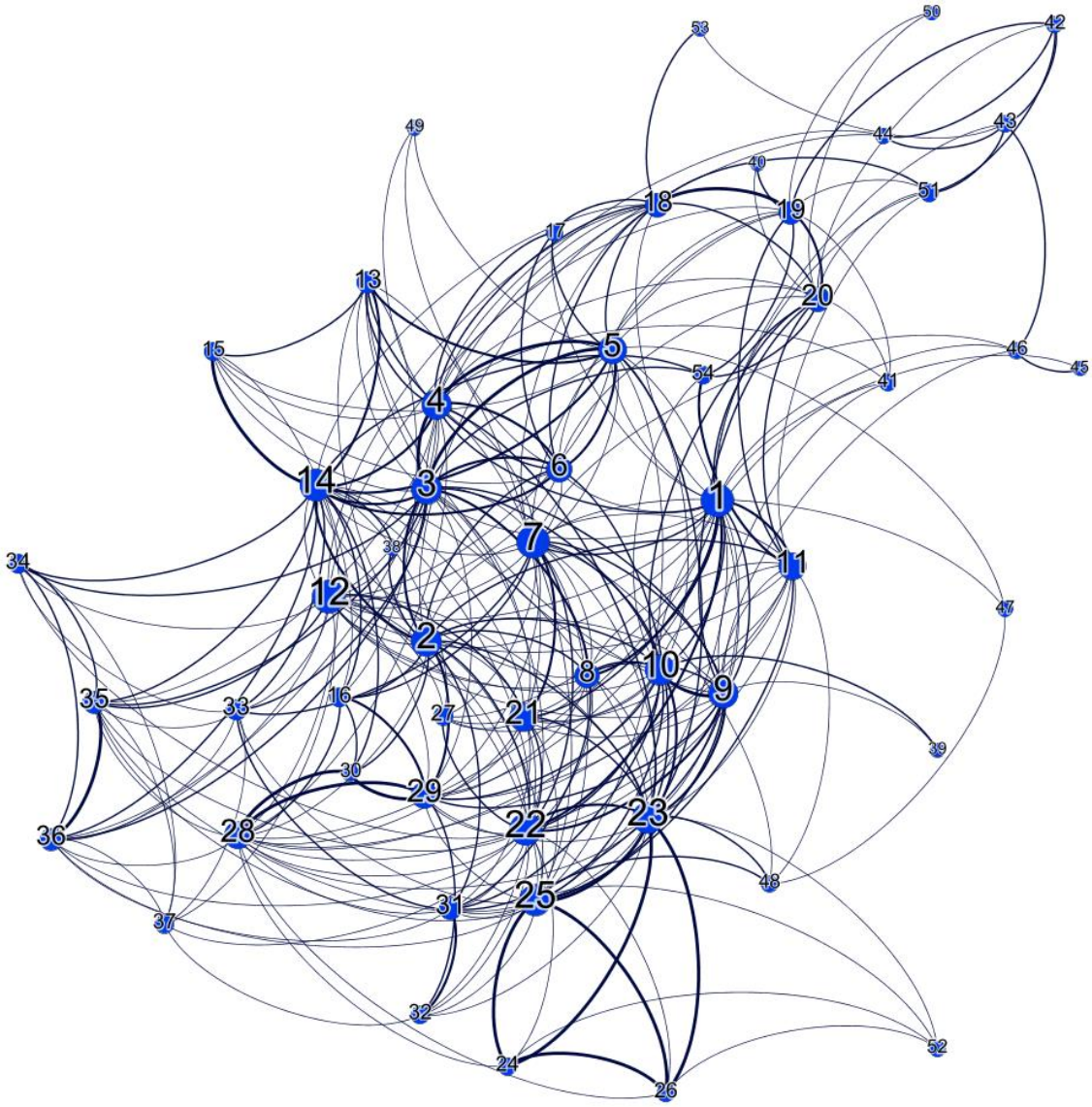
Graphical representation

Here we can observe a graphical representation of the complete weighted and undirected graph. Firstly, the graph is undirected because the relationships developed are the same for both nodes involved. Moreover, the graph's edges are weighted because of the different types of relationships, meaning that each type of relationship will be represented with a unique weight. In detail, we can see those weight-relationships correspondences here:

- The weight 1 describes the relationship of just hanging out together.
- The weight 2 describes the relationship of co-offending together.
- The weight 3 describes the relationship of co-offending serious crimes together.
- The weight 4 describes the relationship of co-offending serious crimes together and also having family bonds.

Note: it is considered that pair of nodes with heigher weights also include the relationships of the lower weights. For example, a pair that is connected with the weight 2 (co-offending together), also hangs out together and a pair with an edge of 4, means that the two nodes have all four relationships developed between them

In the graph representation it is visible that there are lots of diverse kinds of relationships developed from different nodes. Furthermore, the edge line thickness defines the type of relationship. That means that the thicker the line is the stronger the relationship holds, as described above. Furthermore, the size of the nodes is based on the degree of centrality of the node, that shows how connected a node is. In other words, the bigger the size of the node looks, the more connected the node is inside the network. Lastly, due to privacy reasons, the names of the members are not available so each member-node has a unique id number to be distinguished.


59
42
53
49
44
43
40
51
18
19
17
13
20
15
5
46
45
54
41
4
14
3
6
1
7
34
39
11
12
47
2
10
8
9
35
33
16
27
21
39
30
29
23
22
36
28
48
31
25
37
32
52
24
26

Basic topological properties

According to Gephi, the graph contains 54 different nodes and 315 edges. In other words, in this particular London gang are found 54 unique gang members. Also, there have been developed 315 relationships between all the gang members.

That is also confirmed by using python and its library networkX. Furthermore, the graph's diameter is calculated to be 4. As known, the diameter is the longest shortest path of the graph. That means in order to travel from any node to another one we would have to traverse maximum 4 edges (so 4 or less). Lastly, the average path length, which is defined as the average of the shortest paths for all pairs of nodes, is found to be 2.05.

- Showcasing the results

- Importing the right libraries

```
In [62]: import pandas as pd
         import numpy as np

         import networkx as nx
```

- Reading the adjacency matrix that describes the edges of the graph

```
In [63]: edges = pd.read_csv('CSV/LONDON_GANG.csv', index_col=0)
         edges.columns = edges.columns.astype(int) #changing columns names to integer type
         edges
```

- Creating a networkx graph from the advancency matrix

```
In [64]: G = nx.from_pandas_adjacency(edges)
```

- Printing basic infos

```
In [49]: print(nx.info(G))
Name:
Type: Graph
Number of nodes: 54
Number of edges: 315
Average degree:  11.6667
```

- Here we can see the graph's diameter

```
In [66]: nx.diameter(G)
Out[66]: 4
```

- Also the average path length is calculated

```
In [68]: nx.average_shortest_path_length(G)
Out[68]: 2.0538085255066387
```

- Now we calculate the times each shortest path appears.

A script is created to find the times each shortest path length appears .

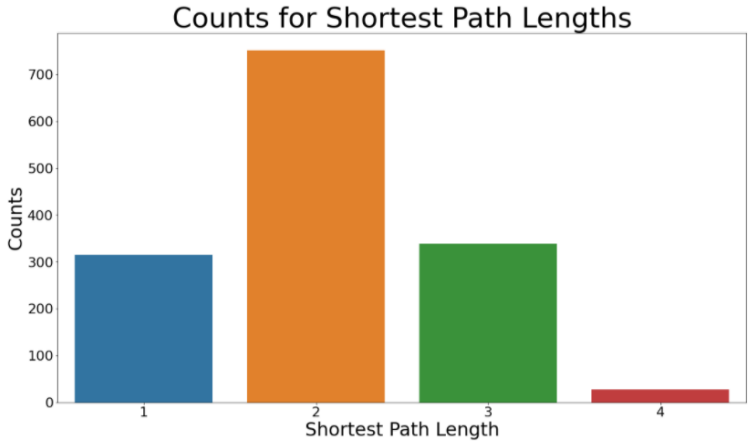
In [121]:
```
list_counts = [0,0,0,0] #list with counts for each path length, 1 path length is found in the first spot and so on
for i in range (1,55):
    for j in range (i+1,55):
        list_counts[(nx.shortest_path_length(G, source=i, target=j)) - 1] += 1
list_counts
```

Out[121]: [315, 751, 338, 27]

That means that from the shortest paths between all nodes, 315 of them had length one, 751 of them had length two, 338 of them had length three and 27 of them had length four

In [170]:
```
font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
matplotlib.rc('font', **font)

fig_dims = (20, 11)
fig, ax = plt.subplots(figsize=fig_dims,)
ax = sns.barplot(x= ['1','2','3','4'] ,
                 y =list_counts, palette='tab10', )
plt.title('Counts for Shortest Path Lengths', fontdict ={'size': 45}, loc='center')
plt.xlabel('Shortest Path Length', fontdict ={'size': 30})
plt.ylabel('Counts',fontdict ={'size': 30})
```

Out[170]: Text(0, 0.5, 'Counts')

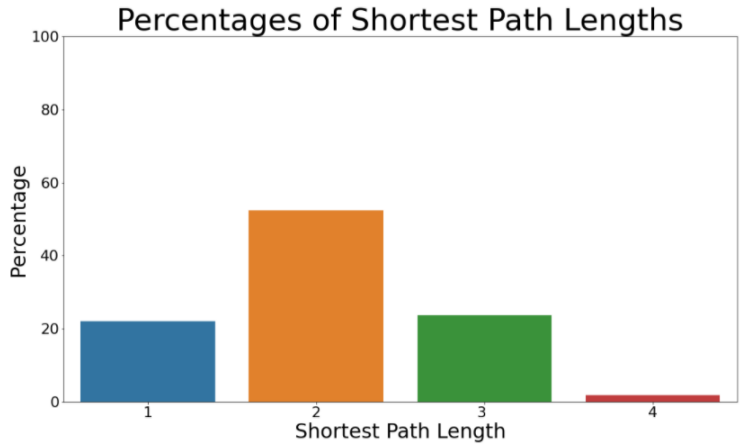


- Also calculating the percentages

In [155]:
```
per_counts = [x / sum(list_counts)*100 for x in list_counts]
per_counts
```

Out[155]: [22.012578616352, 52.48078266946191, 23.619846261355697, 1.8867924528301887]

In [169]:
```
font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
matplotlib.rc('font', **font)

fig_dims = (18, 10)
fig, ax = plt.subplots(figsize=fig_dims,)
ax = sns.barplot(x= ['1','2','3 ','4'] ,
                 y =per_counts, palette='tab10', )
plt.title('Percentages of Shortest Path Lengths', fontdict ={'size': 45}, loc='center')
plt.xlabel('Shortest Path Length', fontdict ={'size': 30})
plt.ylabel('Percentage',fontdict ={'size': 30})
ax.set_ylim(0, 100)
plt.show()
```



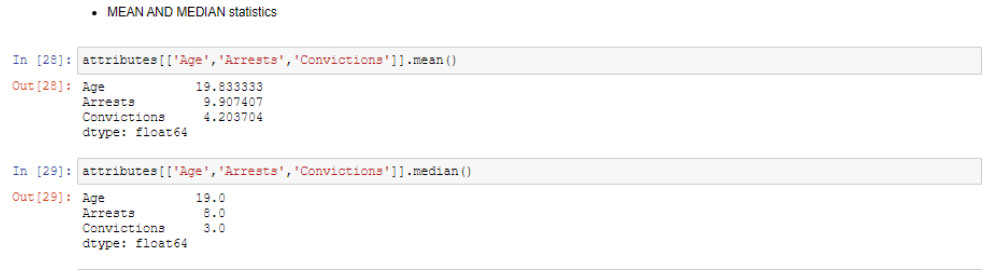
It is clearly visible that most pair of nodes have a shortest path of 2. Actually, the shortest paths with length two are more than all the other counts of lengths summed up. Lastly, it is highly unlikely for a pair of nodes to have a shortest path of length 4 as the likelihood is less than 2%. That means that even if two members do not have any relationship between them, they will probably have one mutual ‘friend’ (member that each of those members have some relationship with) or one ‘friend’ each, who have some relationship between them.

- Also, the graph’s density is calculated to be 0.22 and thus our graph is a sparse one:

Attributes Analysis

For each different gang member there are some extra information that describes them. Specifically, we know the birthplace, age, number of arrests and convictions, if they served in prison already, if they make music and their ranking in the gang.

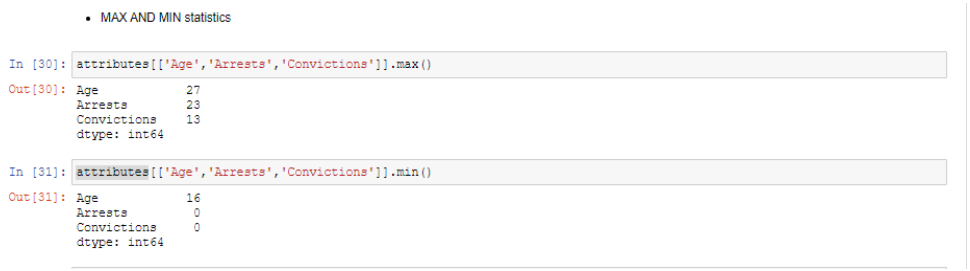
- Let's find the mean and median for the age and the number of arrests and convictions:



It is observed that the mean age of the gangs is almost 20 and the median is 19. That means that there are probably some higher "outlier" values that pull the mean to the right (higher in value). We can also make the conclusion that many youngsters are part of the gang (even underaged youngsters) because of the median and the mean being relatively low.

Moving on, the mean number of arrests is almost 10 and of convictions is around 4. Again, the medians are lower than the means. In conclusion, it is clear that those gang members have quite a criminal record. Knowing that the average gang member has been arrested 10 times and was convicted 4, makes those members extremely dangerous.

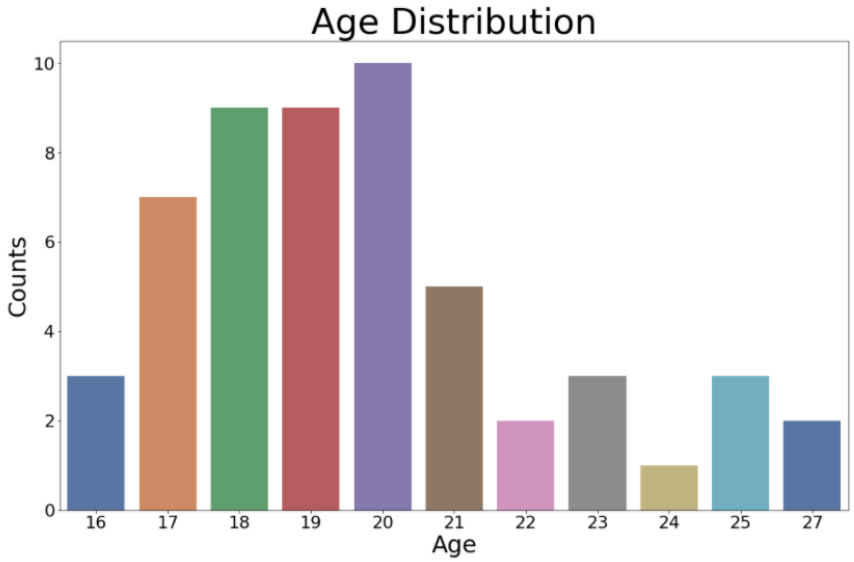
- Now the maximum and minimum values for the age and the number of arrests and convictions are shown:



As suspected, the minimum age is 16 (underaged) and the maximum one is 27 years. Also, the minimum number of arrests and convictions are 0 but the maximum are 23 and 13 respectively.

- Here we can also see the age distribution which confirms the observation of many members beings extremely young and even underaged.

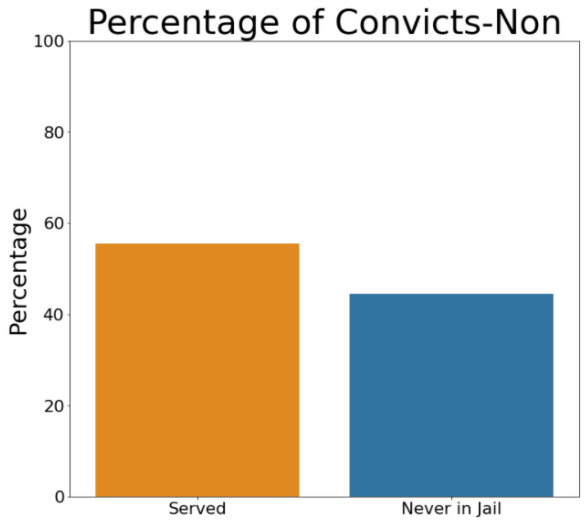
```
In [217]: font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
          matplotlib.rc('font', **font)

          fig_dims = (18, 11)
          fig, ax = plt.subplots(figsize=fig_dims,)
          ax = sns.barplot(x= attributes.Age.value_counts().index ,
                           y =attributes.Age.value_counts().values,
                           palette='deep', )
          plt.title('Age Distribution', fontdict ={'size': 45}, loc='center')
          plt.ylabel('Counts',fontdict ={'size': 30})
          plt.xlabel('Age',fontdict ={'size': 30})

Out[217]: Text(0.5, 0, 'Age')
```



- Now the number of the member-convicts will be showcased:

  - Number of members who served time in jail

```
In [228]: font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
          matplotlib.rc('font', **font)

          fig_dims = (12, 11)
          fig, ax = plt.subplots(figsize=fig_dims,)
          ax = sns.barplot(x= ['Served','Never in Jail'] ,
                           y =attributes.Prison.value_counts()/sum(attributes.Prison.value_counts())*100,
                           palette=['darkorange','tabiblue'], )
          plt.title('Percentage of Convicts-Non', fontdict ={'size': 45}, loc='center')
          plt.ylabel('Percentage',fontdict ={'size': 30})
          ax.set_ylim(0, 100)

Out[228]: (0.0, 100.0)
```

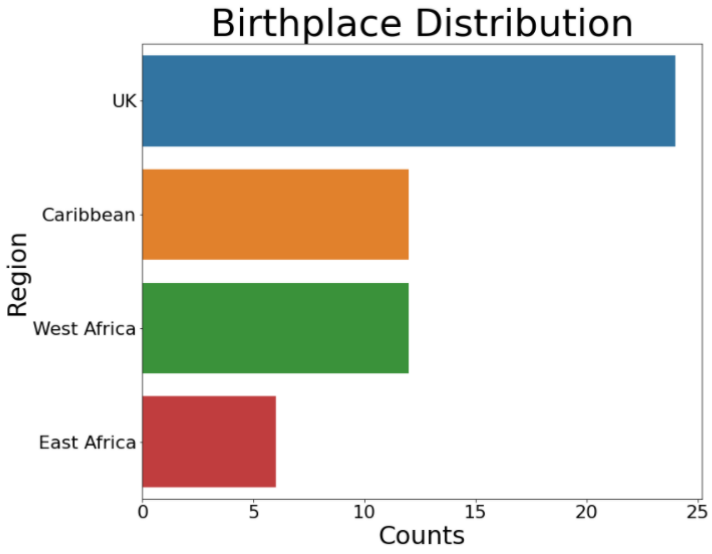


We observe that more than 50% of the members have served in prison, which confirms the perilousness of those members.

- Moreover, the birthplace information of the members will be analyzed:

  - Members per birthplace

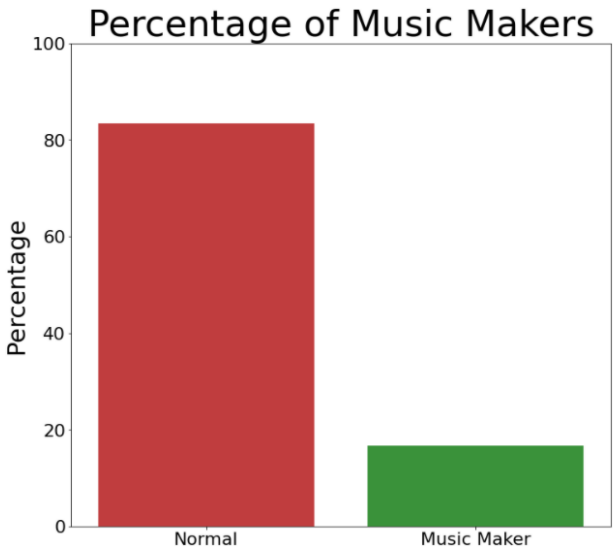
```
In [287]: font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
          matplotlib.rc('font', **font)

          fig_dims = (12, 10)
          fig, ax = plt.subplots(figsize=fig_dims,)
          ax = sns.barplot(y= attributes.Descript.value_counts().index ,
                           x =attributes.Descript.value_counts().values,
                           palette='tab10', )
          plt.title('Birthplace Distribution', fontdict ={'size': 45}, loc='center')
          plt.xlabel('Counts',fontdict ={'size': 30})
          plt.ylabel('Region',fontdict ={'size': 30})
Out[287]: Text(0, 0.5, 'Region')
```



Most gang-members were born in the UK (around 45%) which makes sense because of the gang operating in London. Furthermore, around 20% of the members were born in the Caribbean and around 30% in Africa. In conclusion, it is understandable that the gang is very multicultural, bringing together members from different continents and cultures (Africa and Caribbean)

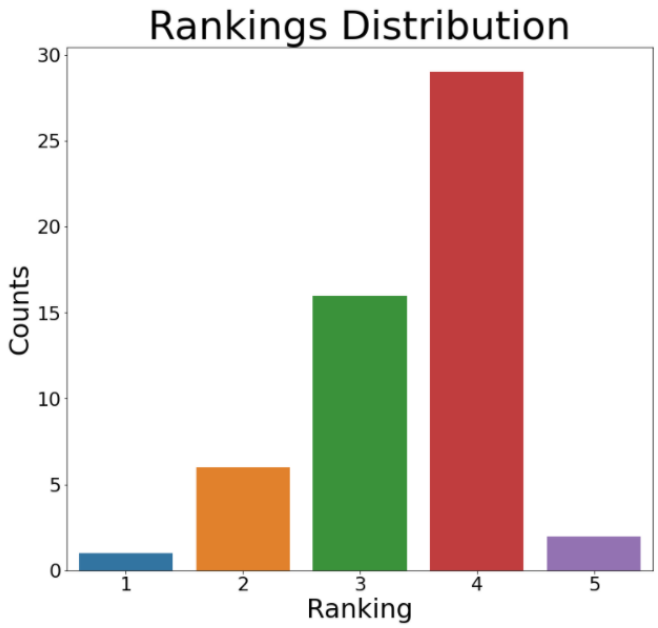
- Additionally, the percentage of music makers inside the gang can be seen here:

```
In [69]: font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
         matplotlib.rc('font', **font)

         fig_dims = (12, 11)
         fig, ax = plt.subplots(figsize=fig_dims,)
         ax = sns.barplot(x= ['Normal','Music Maker'] ,
                          y =attributes.Music.value_counts()/sum(attributes.Music.value_counts())*100,
                           palette=['tab:red','tab:green'], )
         plt.title('Percentage of Music Makers', fontdict ={'size': 45}, loc='center')
         plt.ylabel('Percentage',fontdict ={'size': 30})
         ax.set_ylim(0, 100)

Out[69]: (0.0, 100.0)
```



- Finally, we can see the rankings distributions:

```
In [80]: font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
         matplotlib.rc('font', **font)

         fig_dims = (12, 11)
         fig, ax = plt.subplots(figsize=fig_dims,)
         ax = sns.barplot(x= attributes.Ranking.value_counts().index ,
                          y =attributes.Ranking.value_counts(),
                          palette='tab10', )
         plt.title('Rankings Distribution', fontdict ={'size': 45}, loc='center')
         plt.ylabel('Counts',fontdict ={'size': 30})
         plt.xlabel('Ranking',fontdict ={'size': 30})

Out[80]: Text(0.5, 0, 'Ranking')
```

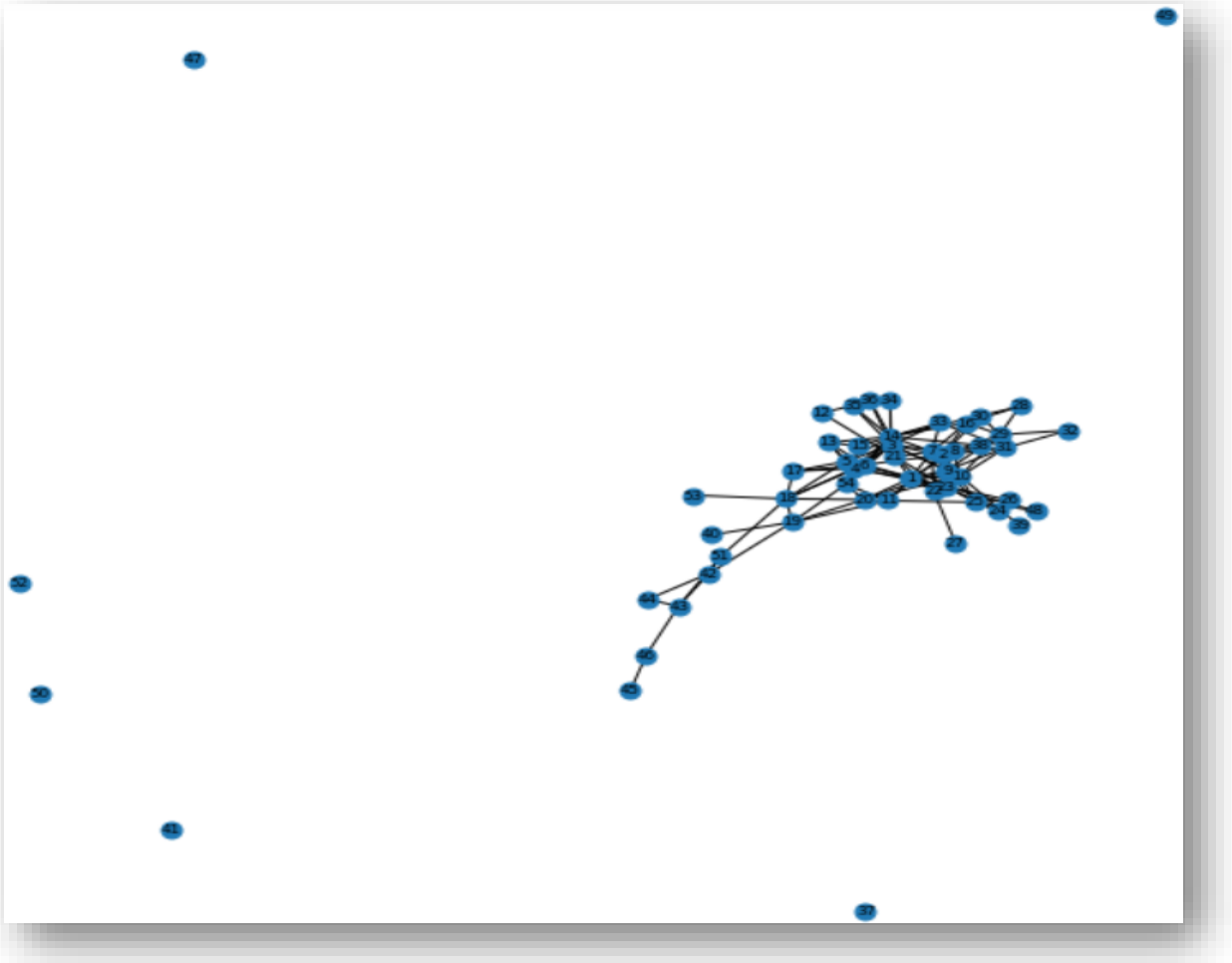


Based on the ranking information, there is one member with ranking 1 (theoretically the leader), six members with ranking 2, sixteen members with ranking 3, twenty-nine members with ranking 4 and one member with ranking 5. There is no information or description on the ranking, so we suppose that the lower ranking is, the more the power owned. In fact, we concluded to that because the member with ranking 1 (most power) has the highest centrality measures.

Component

As we can easily understand and notice, the network only consists of one component of 54 nodes. That makes much sense because the network concerns a gang organization and its relationships. Therefore, each member would have at least one connection to another member, who would be connected to the component. In any other case it would not make sense to keep one or more nodes, which are not connected to the rest of the network because they would simply not belong to the gang. Having explained that, that could only be the case for a network analysis for multiple gangs, which does not apply in this analysis.

- The graph's relationships of co-offending together (edge weights 2,3 and 4) will be drawn now:



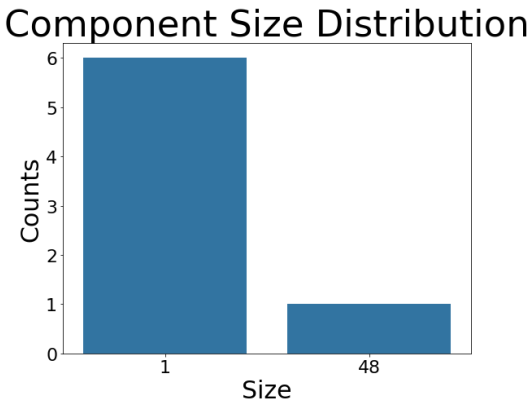
- Components number:

```
In [156]: nx.number_connected_components(G2)
Out[156]: 7
```

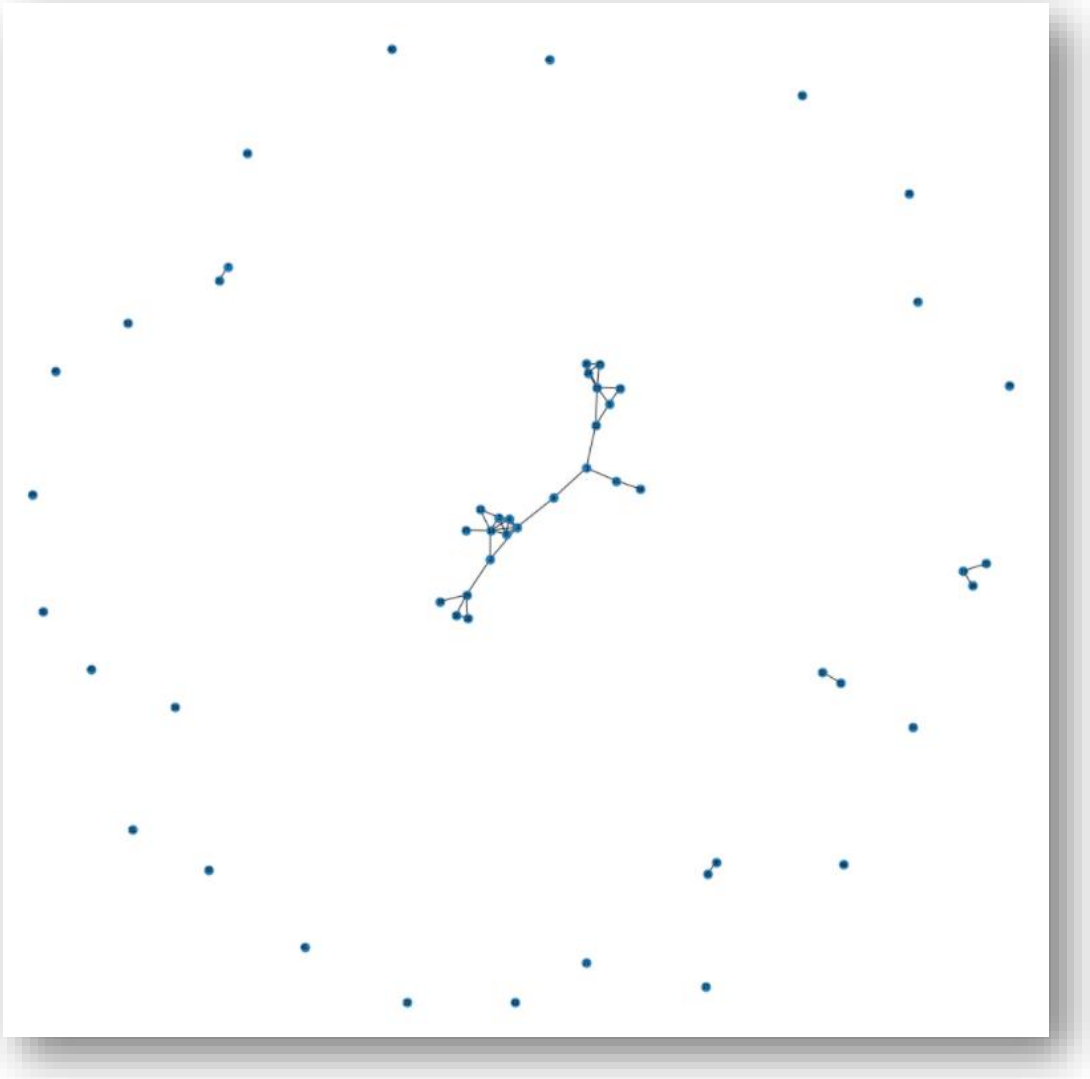
In this case, the number of connected components is 7, with one giant component in the middle. Specifically, the 6 components are of the size 1 and thus have no connections. That could either mean that those 6 members do not offend at all and are just hanging out with the other members of the gang or they could only offend by their own (the first possibility is more probable). In addition, the giant component means that the rest 48 co-offend together. Because those members are in a gang, it is believed that they could co-offend in groups (e.g., the clique in the middle of the giant component) or switch up co-offending partners due to the trust between members.

The component size distribution can be seen below:

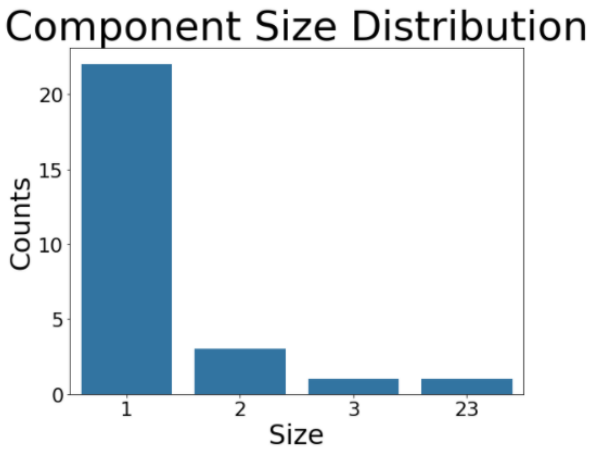
```
In [180]: from collections import Counter

          font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
          matplotlib.rc('font', **font)

          fig_dims = (9, 7)
          fig, ax = plt.subplots(figsize=fig_dims,)
          ax = sns.barplot(y= list(Counter(size_components).values()) ,
                           x = list(Counter(size_components).keys()),
                           palette=['tab:blue'], )
          plt.title('Component Size Distribution', fontdict ={'size': 45}, loc='center')
          plt.xlabel('Size',fontdict ={'size': 30})
          plt.ylabel('Counts',fontdict ={'size': 30})
Out[180]: Text(0, 0.5, 'Counts')
```



- The graph of only co-offending serious crimes is made now (edge weights 3 and 4):



There is a different picture now: many more components can be found. Here's the size distribution:



In the case of co-offending serious crimes, there are different conclusions to be made. There are 22 members (components of size 1) that either do not commit any serious crimes or do commit them on their own (probably the first case is true). Also, there are 3 components of size 2 (3 pairs of members) that only commit serious crimes with their partner in crime (not other members). Moreover, there is a component of size 3 (3 members that co-offend serious crimes). It is interesting that two of those members in that component only commit serious crimes with the 'mutual friends' (member 19), who can be called the bad influence. We could suppose that a triangle would have been formed in that component, but maybe it is just a matter of time for it to happen. Lastly, the giant component has size of 23. We can see different communities being formed there, as the graph is less dense. That means that members do not tend to commit serious crimes with many other members; they have their partners. Despite that, some members have a variety of partners, and hence keeping the giant component connected.

From the above graph, we can also understand the most perilous members of the gang. We can assume that those are the members that commit the most serious crimes in groups of at least two. In detail, the members: 1-11, 13-16, 18-26, 28-32, 35-36 and 54 (e.g., 1-11 means the members with IDs in the range of 1 to 11) are the most likely to commit serious crimes and especially when they are with their partners in crime, horrible actions could be taken.

Degree Measures

- Calculating the degree measures with networkX library

  - A new dataframe is created with the degree for each node

```
In [28]: degrees = pd.DataFrame.from_dict(nx.degree(G))
         degrees = degrees.rename(columns={0:'node', 1:'degree'})
         degrees
```

Out[28]:

	node	degree
0	54	6
1	53	2
2	52	4
3	51	7
4	50	2
5	49	3
6	48	5
7	47	3
8	46	5
9	45	2
10	44	4
11	43	6

  - Average degree

```
In [29]: degrees.degree.mean()
Out[29]: 11.666666666666666
```

  - Maximum degree

```
In [33]: degrees.degree.max()
Out[33]: 25
```

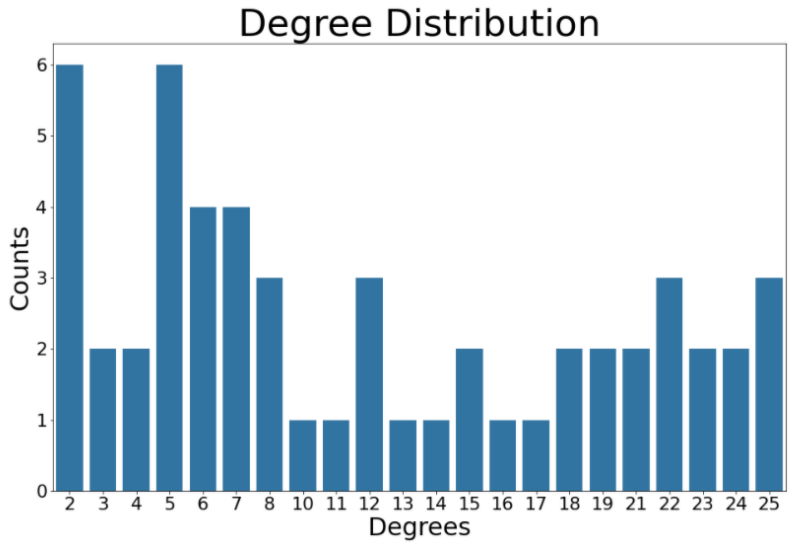
  - Minimum degree

```
In [31]: degrees.degree.min()
Out[31]: 2
```

First of all, the degree shows how connected a node is. In other words, the higher the degree gets, the more gang-members does one member know and is friends with. It is noticed that the average degree is 11.6, which means that on average a gang member has relationships with 11 to 12 other members of the gang. Also, those relationships can be of any kind of those described in the beginning. Furthermore, the maximum degree found is 25 and the minimum is 2. Interestingly enough, each and every member has developed relationships with at least 2 other members. It could be argued that a person joining the gang must know at least two members (or know one and get to know another one) in order to sustain the trust inside the group.

- Plotting the degree distribution

```
In [51]: font = {'size': 22, 'family':'DejaVu Sans'} #increasing the size of the different political parties text
         matplotlib.rc('font', **font)

         fig_dims = (16, 10)
         fig, ax = plt.subplots(figsize=fig_dims,)
         ax = sns.barplot(y= degrees_valuecounts['count'] ,
                          x =degrees_valuecounts.degree,
                          palette=['tab:blue'], )
         plt.title('Degree Distribution', fontdict ={'size': 45}, loc='center')
         plt.xlabel('Degrees',fontdict ={'size': 30})
         plt.ylabel('Counts',fontdict ={'size': 30})
Out[51]: Text(0, 0.5, 'Counts')
```



Looking at the plot, it is noted that the majority of nodes have values below the average degree which is 11.6. However, the existence of some nodes that have extremely high degrees (e.g., 22 and above) compared to the majority, increases the average degree. We could also see that by calculating the median.

```
In [52]: degrees.degree.median()
Out[52]: 9.0
```

As expected, the median, which is not affected much from extreme values, is lower than the average degree. From an analyst perspective that can be explained as:

- Lots of gang members have very low degrees, meaning that they are not that connected in the graph. In other words, they do not have relationships with many other gang-members. Those could be new members or members that are not very involved in the mob activities.
- Some few gang members have extremely high degrees (higher than 20) meaning that they are very connected and thus have lots of relationships with other members. We could imagine that those people are the ones to run the show, keep the gang together and also form the main core of the organization.

Centrality measures

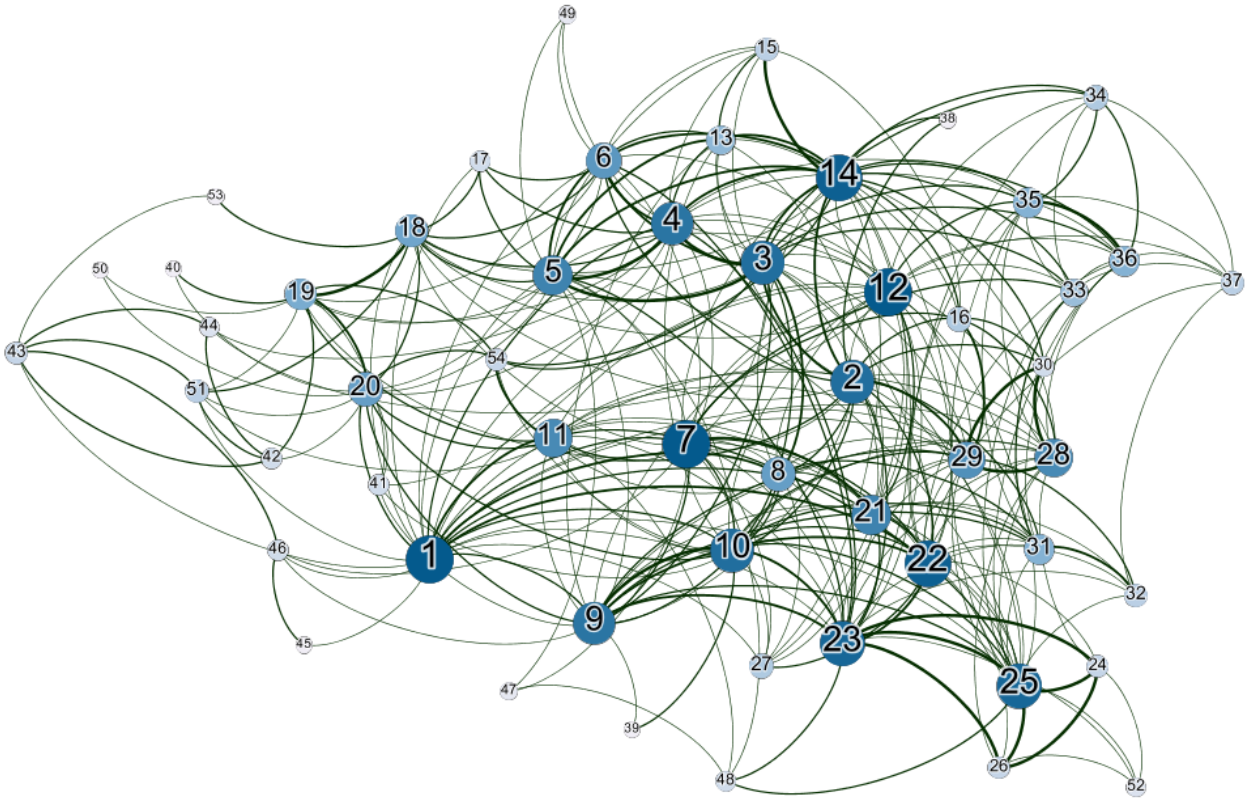
Degree

Now we get a better view of the nodes based on their degree. Not only the size of the node, but also the color defines the degree of the node in the diagram. The higher the degree of a node is, the more intense the color blue becomes. The gangsters with the highest degrees are the dark blue ones. As expected, they are located in the center of the graph or close to it because of the many connections they sustain. On the other side, the members with the lowest degrees are colored white. Specifically, most of those gang members are located on the edges of the graph.

In detail, based on the degree the members are divided in:

- The most famous and outgoing members (most friends) with a degree of 21 or more: Nodes 1, 2, 3, 4, 7, 9, 10, 12, 14, 22, 23, 25.
- The popular members with degree of 13 to 20. Nodes 5, 6, 8, 11, 18, 19, 20, 21, 28, 29
- The normal members with degree of 5 to 12. Nodes 54, 51, 48, 46, 43, 42, 41, 37, 36, 35, 34, 33, 32, 31, 30, 27, 26, 24, 17, 16, 15, 13
- The newcomers or least involved members are considered to have a degree of 4 or less: Nodes 38, 39, 40, 44, 45, 47, 49, 50, 52, 53.

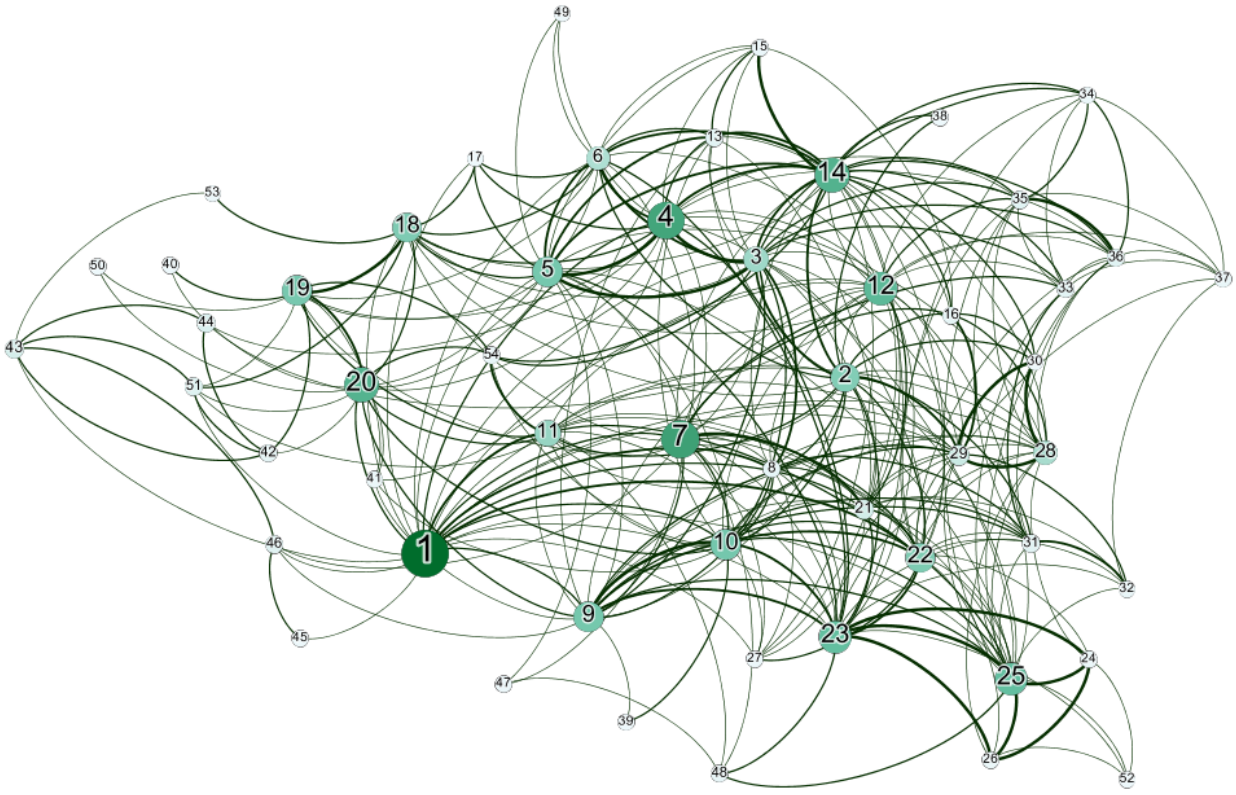
Note: the degree thresholds were selected by personal choice. One could set different thresholds based on their preferences.



Betweenness Centrality

Betweenness centrality is the measure to show how important a node is in terms of connecting other nodes. Again, based on the intensity of the color and the size of the node we can identify the nodes with the highest, normal and lowest betweenness centralities in the graph.

It is clear that the Node 1 has the highest betweenness centrality with around 0.11 (as calculated). After that node, the nodes 7 and 4 have the next highest betweenness centralities. As seen at the degree section, those nodes also have some of the highest degrees in the graph. That means that not only do those members know many different members, but they also connect plenty of different members by being the middleman. From a more street view, because they hang out with different people, who do not hang out with each other, they can transfer information from one member to another when needed or even bring different members together because of their connections. In fact, those are the members that could convey valuable information quickly to a plethora of different members for example for a new criminal activity or ongoing police investigation. In conclusion, the member with the highest betweenness centrality can be considered to be the unifiers of the gang and also coordinators of actions due to their ease to pass on information and updates.



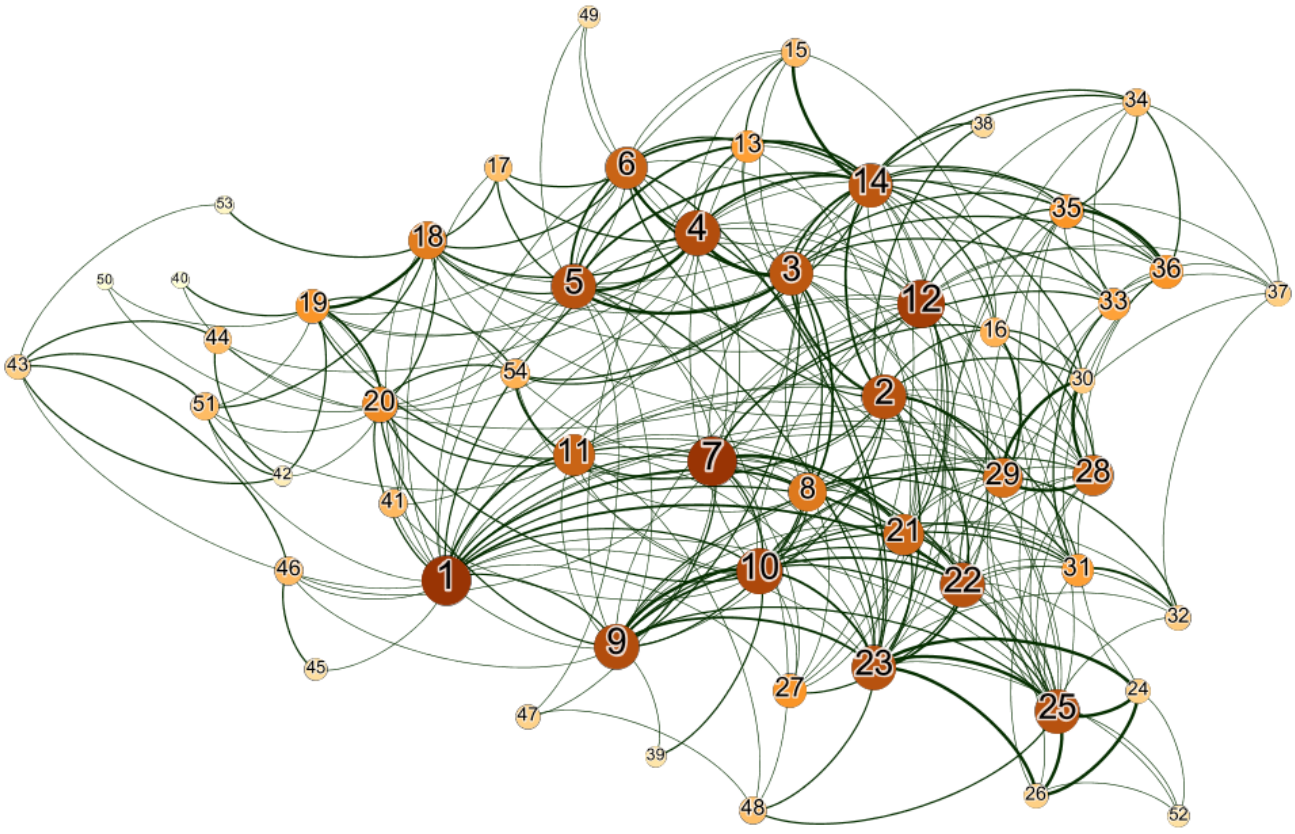
Closeness Centrality

As known, closeness centrality shows how easily a node can reach other nodes. As usually showcased, the more intense the color gets, the higher the closeness centrality for the node becomes. The same applies for the size of the node.

According to networkX, the nodes with the highest closeness centrality are 1, 7, 12, 10, 9, 4 etc. Accordingly, those nodes are the closest to the center of the network.

As observed, the nodes 1, 4 and 7 have been calculated to have one of the highest betweenness centralities as well. Therefore, it is not a surprise to also have a high closeness centrality. On the other hand, the nodes 9, 10, 12 do not have the highest betweenness centralities but have one of the highest closeness centralities. That implies that even though those nodes may not connect a great number of nodes (as of short paths), they find themselves in the center of the graph. That happens because they are connected to the central nodes 1, 4 and 7 (which are both in the center of the graph and connect many nodes)

In more practical words, even though the nodes 9, 10, 12 may not be able to connect members from the whole gang, they hang out with the ‘coordinators’. Therefore, they are able to learn news, calls and occurrences faster than the other gangsters, as well as pass on their own news quicker.

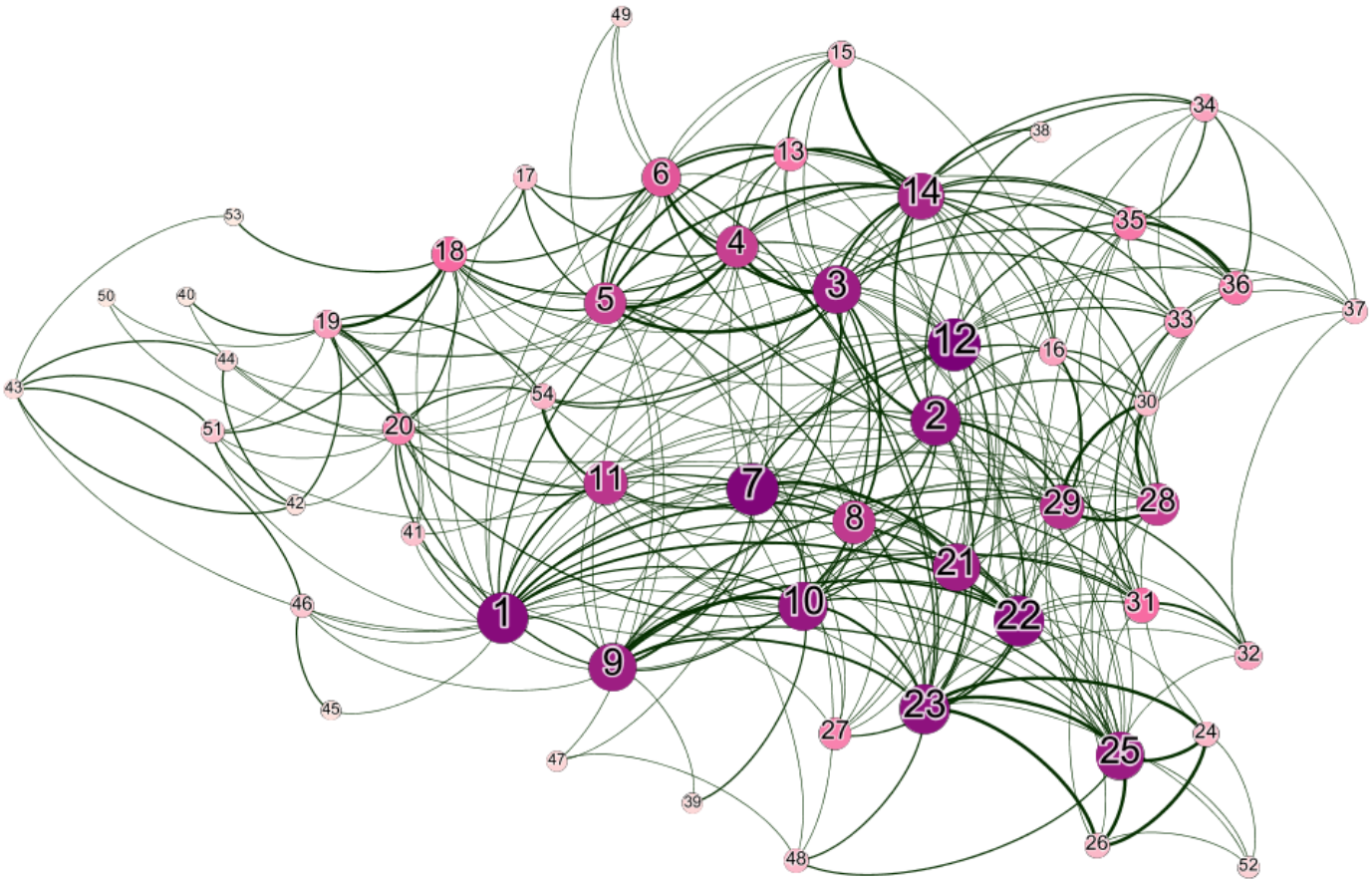


Eigenvector Centrality

Eigenvector centrality is the metric to show how connected a node is to other important nodes in the network. Again, the size of the node and the intensity of the color correspond to the level of eigenvector centrality.

It is noticed that the nodes 1, 2, 3, 7, 10, 12, 21, 22, 23, 25 are the ones with the highest eigenvector centrality. It is noticed that all those nodes (apart from node 21) belong to the most outgoing members as characterized in the degree section.

In the gang case, those members are the most important ones in the network, hence are believed to consist the organization of the gang. They are connected to the other most important gangsters, so they probably know a lot of precious information about the operations, crimes and plans of the mob. Lastly, the node 1 is considered to be the leader of the organization because he has the highest scores in all the centralities and also based on his ranking attribute (only member to have the ranking 1).



Clustering Effects

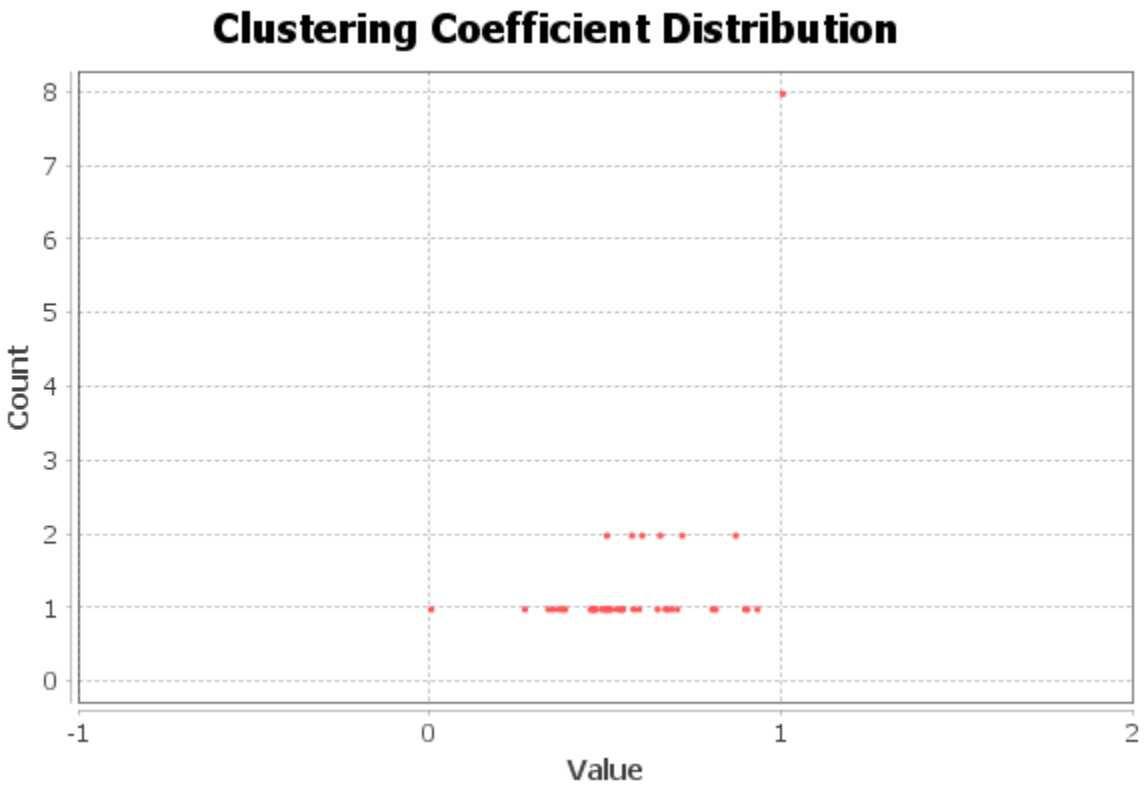
- Firstly, the average clustering coefficient is calculated with networkX to be 0.63. This can indicate how complete a network is. The closer the average clustering coefficient is to 1, the more complete the graph will be. Also, that is a sign of triadic closure because the more complete the graph is, the more triangles will usually arise.

- Calculating the average clustering coeeficient

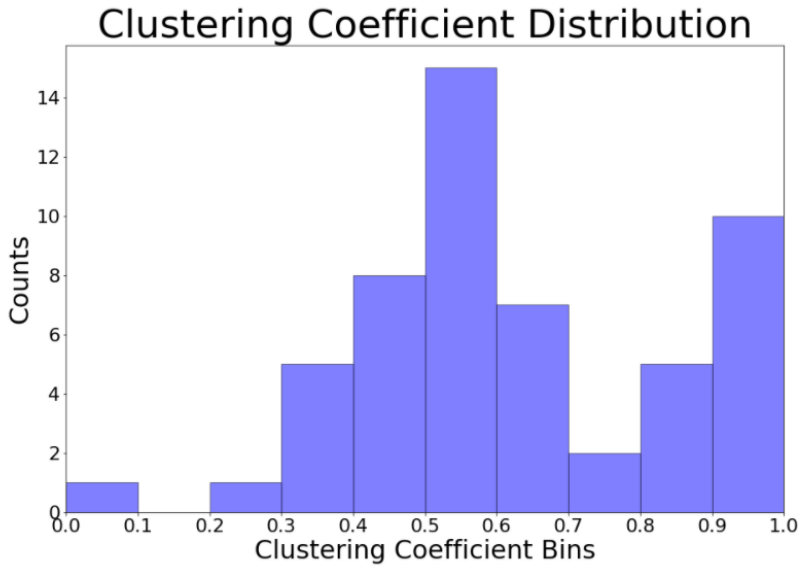
```
In [44]: nx.average_clustering(G,)
Out[44]: 0.6331465409311988
```

As known, the average clustering coefficient is the average of the local clustering coefficients of all the nodes.

- Here we can see the local clustering coefficient distribution in Gephi and networkX.



```
In [86]: clust_values = list(nx.clustering(G).values()) #creating list from dictionary values of clustering coefficients
         fig_dims = (15, 10)
         fig, ax = plt.subplots(figsize=fig_dims,)
         plt.hist(bins=np.linspace(0., 1., 11), x=cluster_values,
                                color='blue', edgecolor='black', alpha=0.5, )
         ax.set_xlim(0, 1)
         plt.xticks(np.linspace(0., 1., 11))
         plt.title('Clustering Coefficient Distribution', fontdict ={'size': 45}, loc='center')
         plt.xlabel('Clustering Coefficient Bins',fontdict ={'size': 30})
         plt.ylabel('Counts',fontdict ={'size': 30})
         plt.show()
```



In order to showcase the distribution, it was decided to create 10 bins of 0.1 interval each due to numbers being real. Also, it is well-known that the clustering coefficient of a node N is the probability that two randomly selected friends of N are friends with each other.

It is noticed that the bin with the highest counts by far, concerns nodes with local clustering coefficients from 0.5 to 0.6. Also, for around 43% of the members there is a probability of 40-60% that two of their friends are also friends with each other. Very interesting is that around 28% (more than one quarter) of the members have a probability of more than 80%, that two randomly selected friends of theirs are friends with each other. Finally, for 72% (7 out of 10 members) of all members there is a probability of 50% and higher, that two randomly selected friends of the member are also friends with each other. In conclusion, that shows that it is very likely for friends of friends to also be friends, which is an important sign of triadic closure.

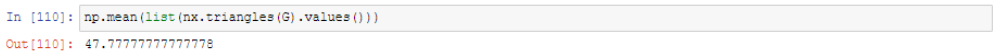
- Now we can see the number of unique triangles found in the network. There are 860 different triangles.

  - Calculating the sum of all unique triangles

```
In [90]: sum(list(nx.triangles(G).values()))/3 #deviding with 3 because each triangle is counted once for each node
Out[90]: 860.0
```

- Also, the average number of triangles that a node is a part of. On average, a node is part of almost 48 triangles.

- Average number of triangles a node belongs to



That is a strong indication that very often the friends of member are also friends. Let's call a specific node N. If just two of his friends are also friends between them, that would count as 1 triangle. On average there are 48 triangles, so 48 times two friends of the member's N friends are also friends.

That could be explained from the triadic-closure operation. As time goes by, new relationships tend to form between two members who have a common friend. In detail, it can be explained with an example. Let's imagine there is a member A that hangs out with the member B and C and members B and C do not hang out or know each other. Over time, because the member A spends time with the two other members, it is almost certain that the member A will acknowledge the existence of B to C and the other way around. That could create pressure for the members B and C to get to know each other because especially in gang environments members tend to know each other. Furthermore, it is very likely for the members B and C to get to meet each other sooner or later because of their mutual friend A. For example, if the member A decides to organize an operation or meetup, he would probably invite most of his friends. In addition, the trust factor is very crucial in the gang setting. In other words, because B and C are friends with A, trust, confidence and reliance exists between them. That makes it much easier for them to trust each other than in the case of not having any mutual friends.

Same applies for new members as well. Every member has at least two 'neighbors' (friends) and all members are part of at least one triangle (except for member 53). That could mean a couple of things:

- In order for a person to become a member, the person needs to know at least two members of the gang, who are usually friends between them.
- Otherwise, when a new member is recruited from one member, the new member quickly becomes friends with one of the recruiter's friends as the recruiter and the new member spend time together.

Bridges

It is clear that there are no bridges in the network.

- However, it is also validated with the use of networkX.

BRIDGES

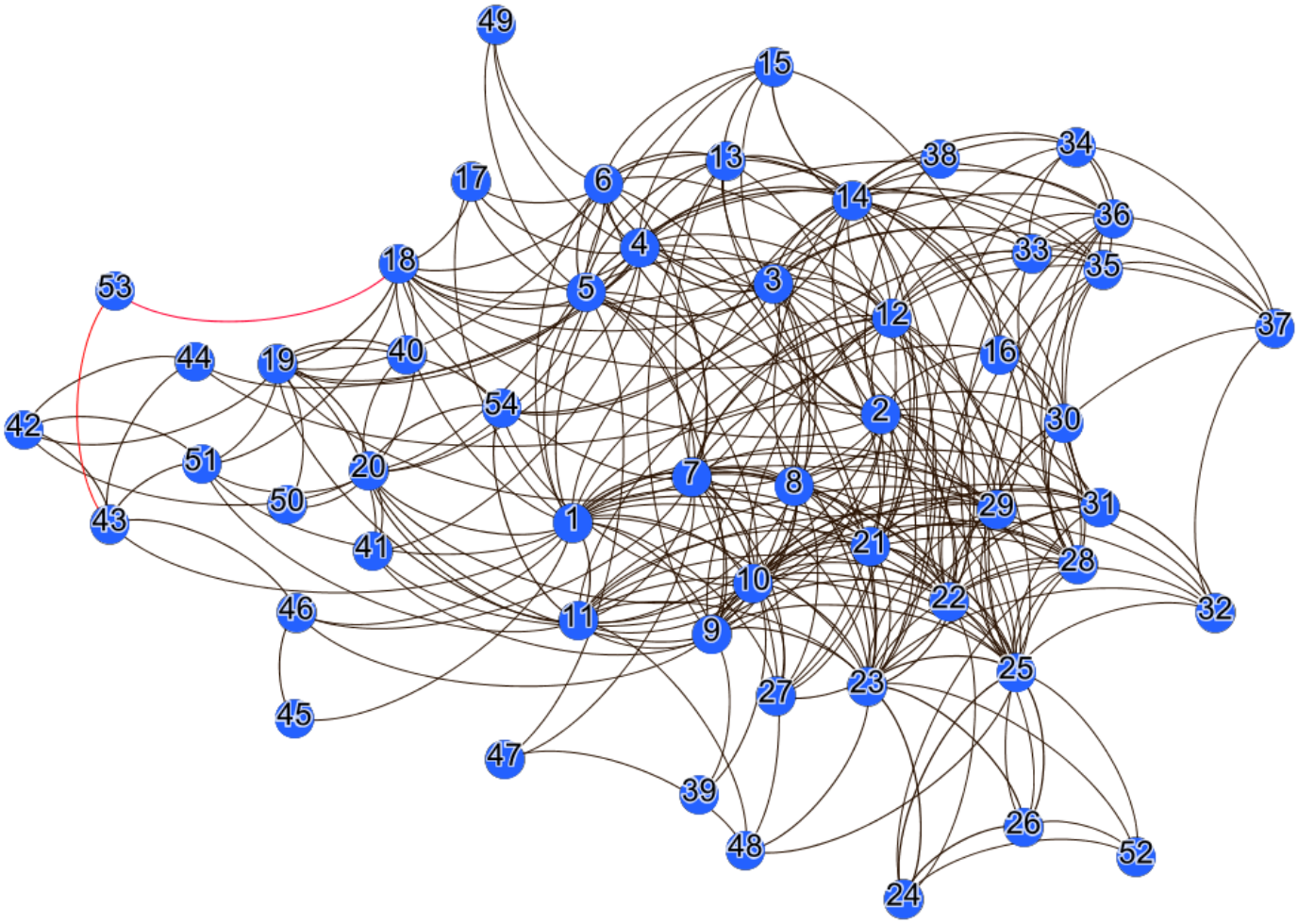
```
In [139]: nx.has_bridges(G)
Out[139]: False
```

- Now the local bridges will be found with networkX.

- Finding the local bridges

```
In [142]: for local in nx.local_bridges(G):
              print (local)

(53, 43, 3)
(53, 18, 3)
```

Interestingly enough, only two local bridges are found. In fact, those two bridges concern the only member not to be involved in any triangles (member 53). We can see in the graph picture below, the local bridges drawn with red color. Moreover, the span of the local bridge is calculated as well in networkX. It is set to be 3 for both local bridges (as the span is the third tuple number of the above output). Looking at the graph, if the edge between the nodes 43 and 53 was deleted, the path from one to the other node would become: 43-51-18-53 (nodes). On the other hand, if the edge between the nodes 18 and 53 was deleted, the path from one to the other node would become: 53-43-51-18 (nodes). Indeed, those paths have a length of 3 edges.



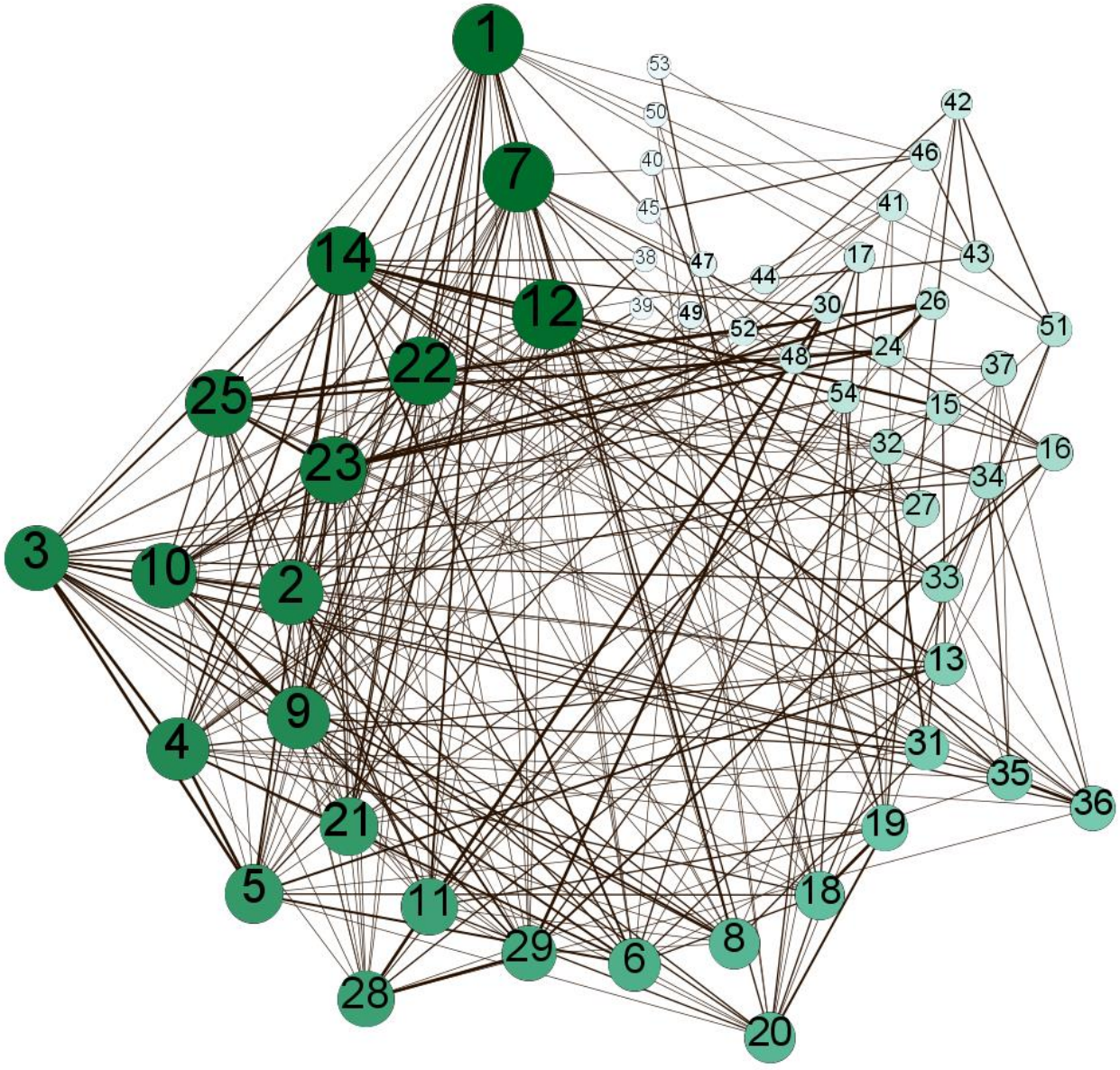
The lack of bridges can be explained by the triadic closure and the structure of a gang. In detail, the triadic closure, as explained above, results in members with mutual friends becoming friends over time. That means that if there was a bridge (normal or local) in the past, new edges would form and further connect the different segments of the network creating triangles. In other words, friends of the nodes of the bridge would become friends with the other node of the bridge and so on. Moreover, the structure of a gang would hardly allow normal bridges to happen. Due to many different members getting involved in different operations and the factor of trust playing a huge role, an existence of a normal bridge could lead to a tear of the network into two different components in case of a disagreement or conflict of the bridge nodes.

Homophily - Assortativity

Assortativity and homophily describes the preference for a network's nodes to attach to others that are similar in some way.

Assortativity Degree

- We start by examining assortativity in terms of nodes degrees.



The above layout is a radial axis layout. That means that nodes with the same degree are grouped together in the same line. We can imagine it like a clock. 1 o'clock is the section with the nodes with lowest degrees. On the other side, 12 o'clock is the line with the nodes of the highest degrees. With the same idea, all different node degrees are set. Furthermore, the size of the node and the intensity of the color also describe the degree of the node. It is clear that the nodes in the same line (section) have identical sizes and colors.

Now we calculate the degree assortativity coefficient:

HOMOPHILY

- Setting up the attributes for each different node with `set_node_attributes`

```
In [57]: node_attr = attributes.set_index('Id').to_dict('index') #creating dictionary to pass for the set attributes function
         nx.set_node_attributes(G, node_attr)
```

- Let's check the attribute `Birthplace` for all nodes

```
In [60]: nx.get_node_attributes(G,'Descript')
```

```
19: 'Caribbean',
18: 'Caribbean',
17: 'Caribbean',
16: 'UK',
15: 'UK',
14: 'UK',
13: 'UK',
12: 'UK',
11: 'West Africa',
10: 'West Africa',
9: 'West Africa',
8: 'West Africa',
7: 'East Africa',
6: 'UK',
5: 'Caribbean',
4: 'Caribbean',
3: 'Caribbean',
2: 'Caribbean',
1: 'West Africa'}
```

- Firstly, the degree assortativity coefficient is found

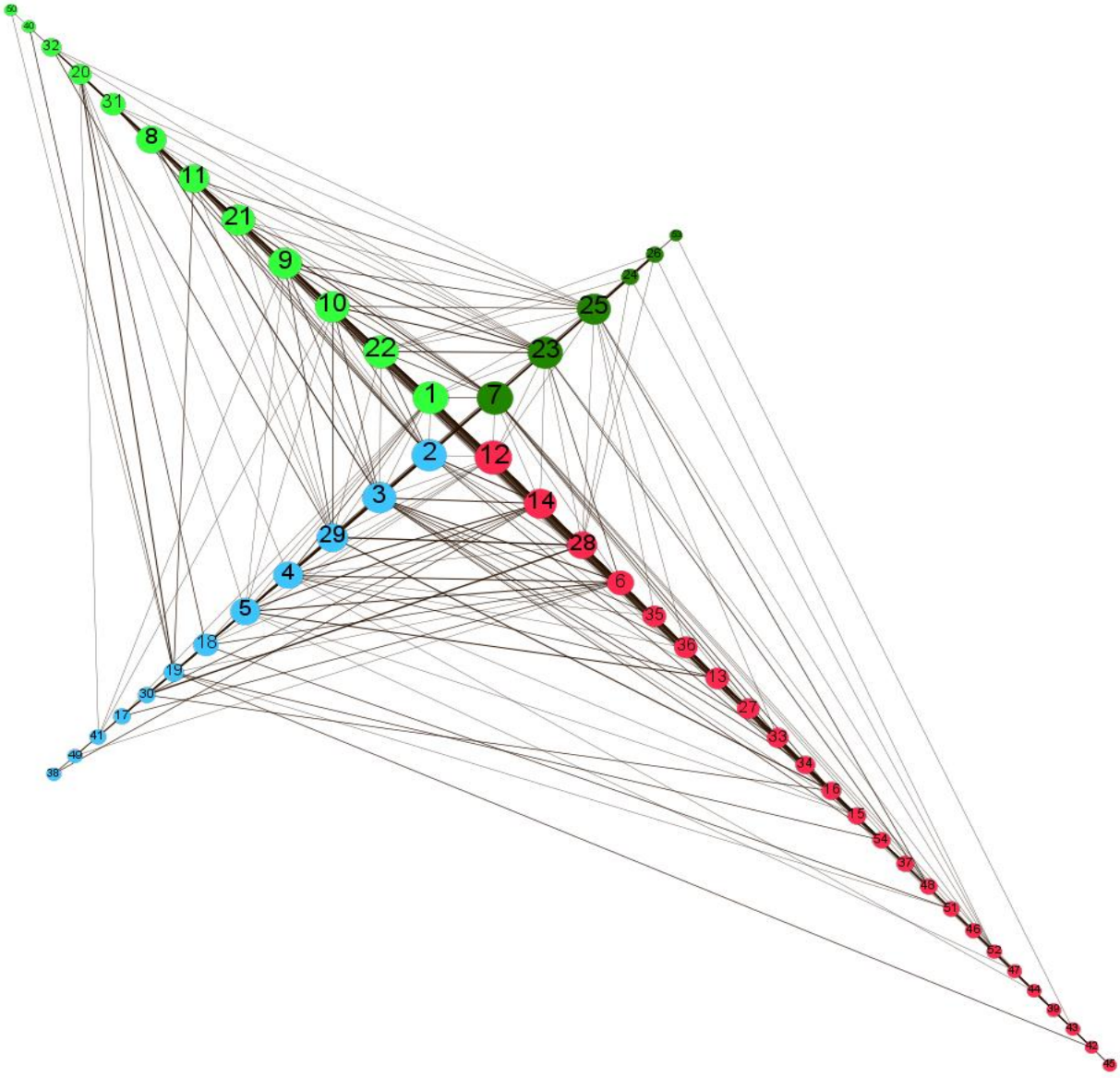
```
In [47]: nx.degree_assortativity_coefficient(G)
Out[47]: 0.0226245269341033
```

In fact, the assortativity coefficient is the Pearson correlation coefficient of degree between pairs of linked nodes. That means that it takes values from -1 to 1. In detail, a positive assortativity coefficient indicates a correlation between nodes of similar degree, while a negative indicates relationships between nodes of different degree.

In our case the assortativity coefficient is around 0.023, which is almost 0. That means that the network is almost non-assortative, and we cannot correlate linked nodes based on their degrees. In other words, we can not draw conclusions on the friends of members based on their number of friends. Lastly, that could be also seen in the above layout as edges have been formed between all kind of nodes, no matter their degree.

Assortativity Birthplace

- Next up is the assortativity in terms of birthplace.



The same layout of radial axis is used again. Each different color is a line of nodes of a particular birthplace. In fact, the nodes closest to the center are the ones with the highest eigenvector centrality, meaning they are connected to more important nodes. Also, the size of the nodes is based on the eigenvector centrality.

The light green color corresponds to members born in West Africa and the dark green one to members born in East Africa. On the other side, the red color defines members born in the UK and the blue one members born in the Caribbean.

Now the attribute ‘birthplace’ assortativity coefficient is calculated:

Now we calculate the assortativity coefficient for categorical attributes

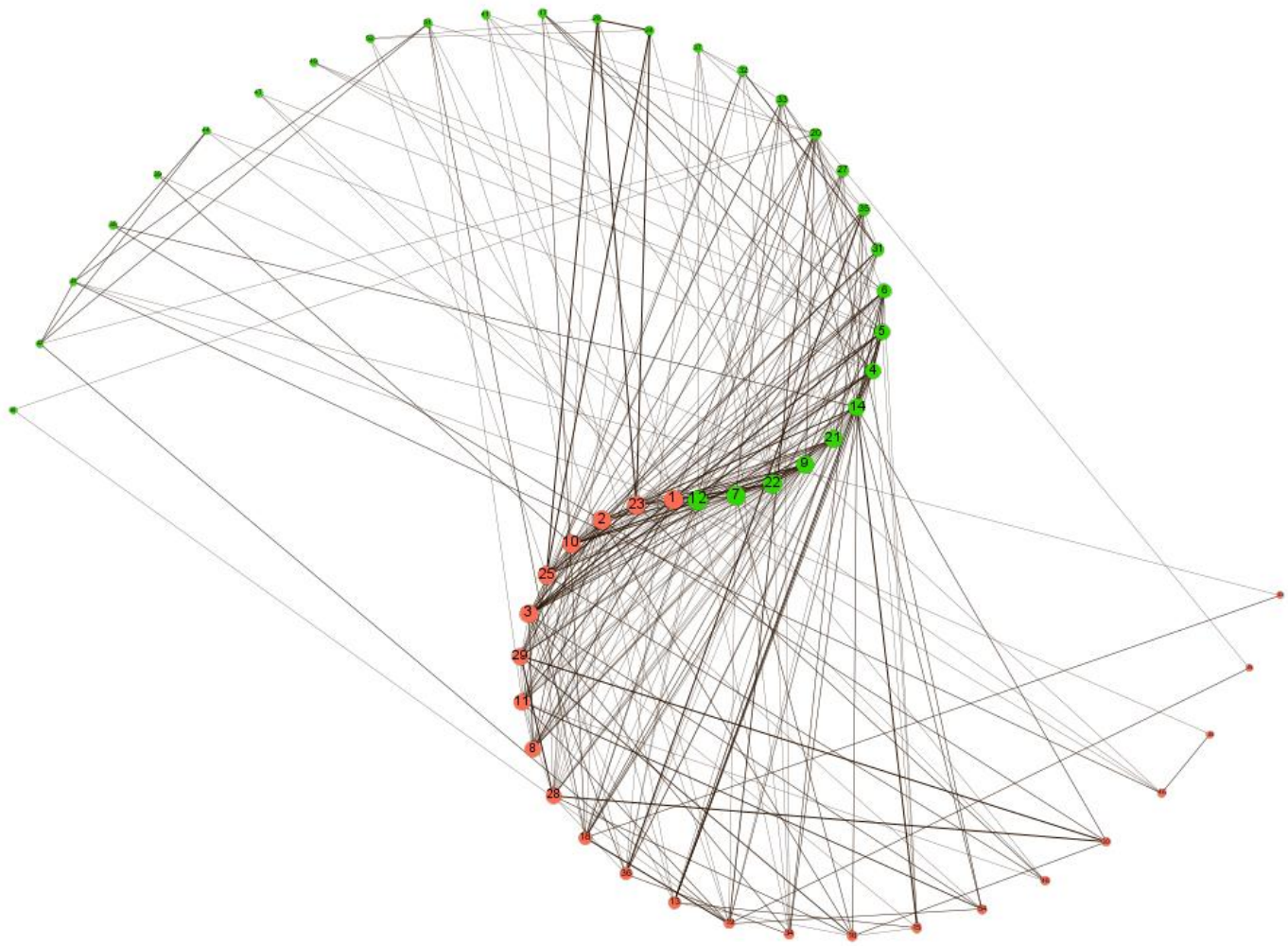
- Attribute `Birthplace`

```
In [48]: nx.attribute_assortativity_coefficient(G,'Descript')
Out[48]: 0.11317444577736785
```

The assortativity coefficient is computed to be 0.11. Again, that is very close to 0. Someone could argue that there is a just very slight correlation between members who were born in the same region. However, the network is considered almost non-assertive when it comes to the birthplace of the members. Furthermore, we can see in the layout that there are plenty of edges from a birthplace to different birthplaces. In conclusion, we can not argue that members of one birthplace prefer to make friends only with member of the same birthplace. In other words, no strong association of the similar birthplace values between connected nodes.

Assortativity Prison

- Now the assortativity coefficient for the attribute Prison will be examined.



The layout radial axis is used again but with spiral axis in order to see all the edges from the two different attribute values (if the categories were in the same line, that would not be possible). In detail, the red-orange color corresponds to members that have been to prison and the green one to members that have never been to prison.

Again, the assortativity coefficient is calculated with networkX:

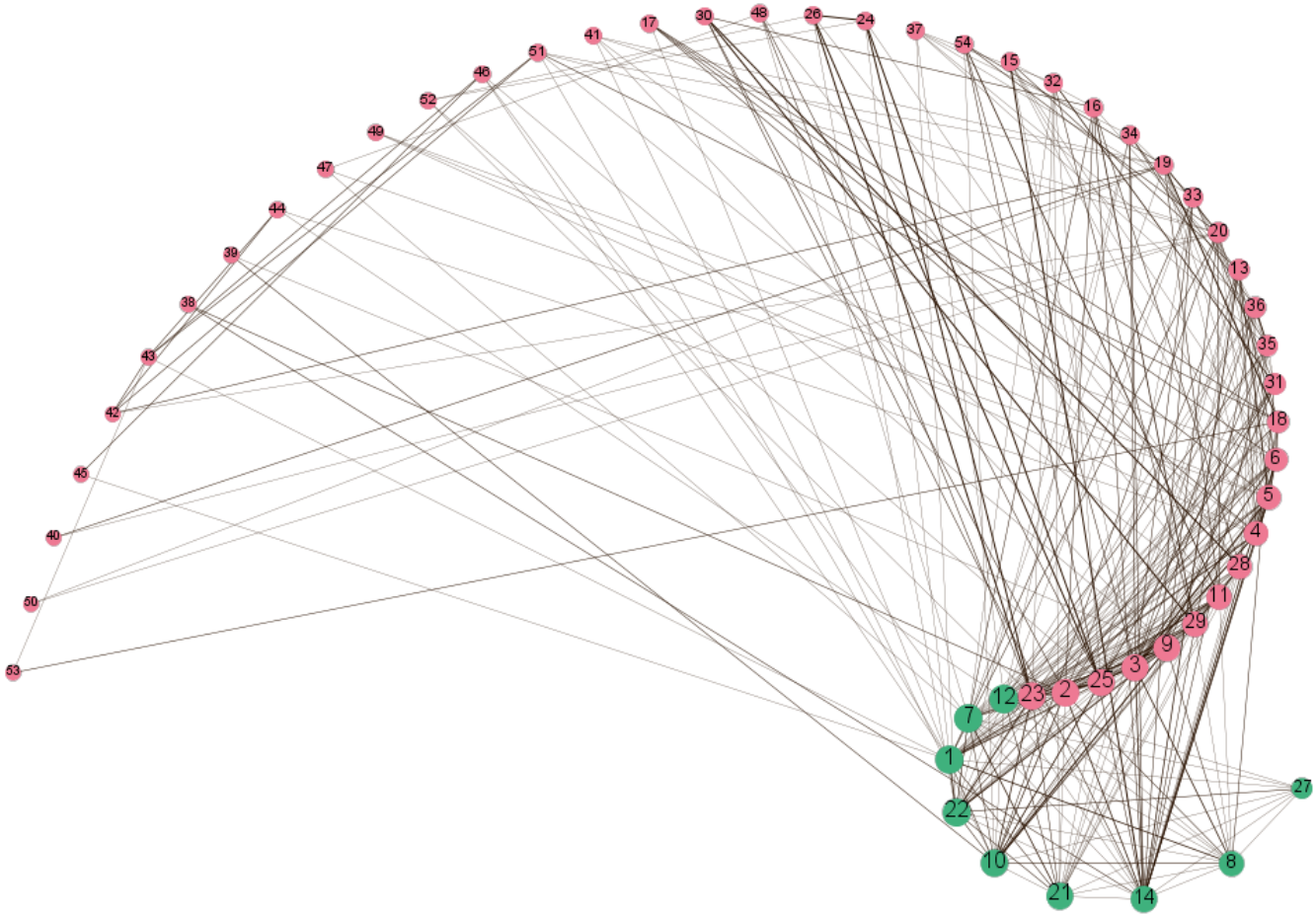
- Attribute `Prison`

```
In [49]: nx.attribute_assortativity_coefficient(G,'Prison')
Out[49]: 0.014880201765447642
```

Again, the assortativity coefficient is almost 0. That means that the network is non-assertive when it comes to the attribute of prison serving. Furthermore, we can see that from the network layout above as edges can be found from all kind of nodes to all kind of nodes despite their prison experience. To sum up, we can not draw conclusions on the friends of members based on if they have been to prison.

Assortativity Music

- The assortativity coefficient for the attribute music will be examined.



The same layout is used once more. With the green color are the nodes that create music and with the red color are the nodes that do not produce music.

Let's calculate the coefficient with networkX:

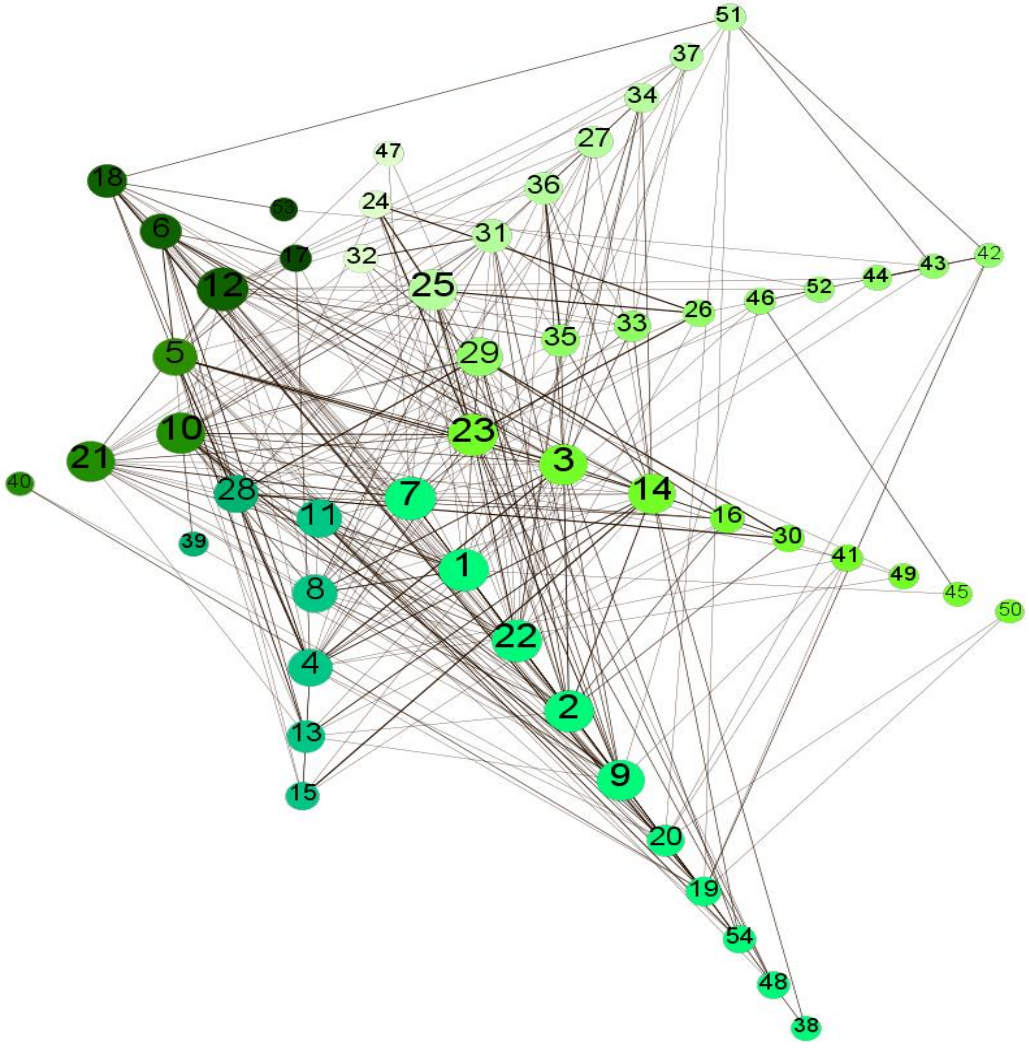
- Attribute `music`

```
In [50]: nx.attribute_assortativity_coefficient(G,'Music')
Out[50]: 0.09501333880566394
```

The attribute assortativity coefficient is found 0.09 which is very close to 0 again. We can not conclude that all members prefer to hang out with members like themselves when it comes to music producing as edges are found between all kind of nodes. Maybe that could be the case for a couple of members only, like member 27 (in the bottom right corner) who only has friends, other members that also produce music.

Assortativity Age

- Now the age attribute assortativity coefficient will be seen:



The members with the same age are seen in the same line and with the same color. The lowest age members (16yo) are with the lightest green color and are found in the place of 1 o'clock. On the other hand, the members with the highest age (27 yo) are found with the darkest green color at the place of 12 o'clock. Again, we can see that there are edges between all kind of ages.

NetworkX is used again to calculate the assortativity coefficient for the numerical variable age.

The assortativity coefficient for numerical attributes is found next

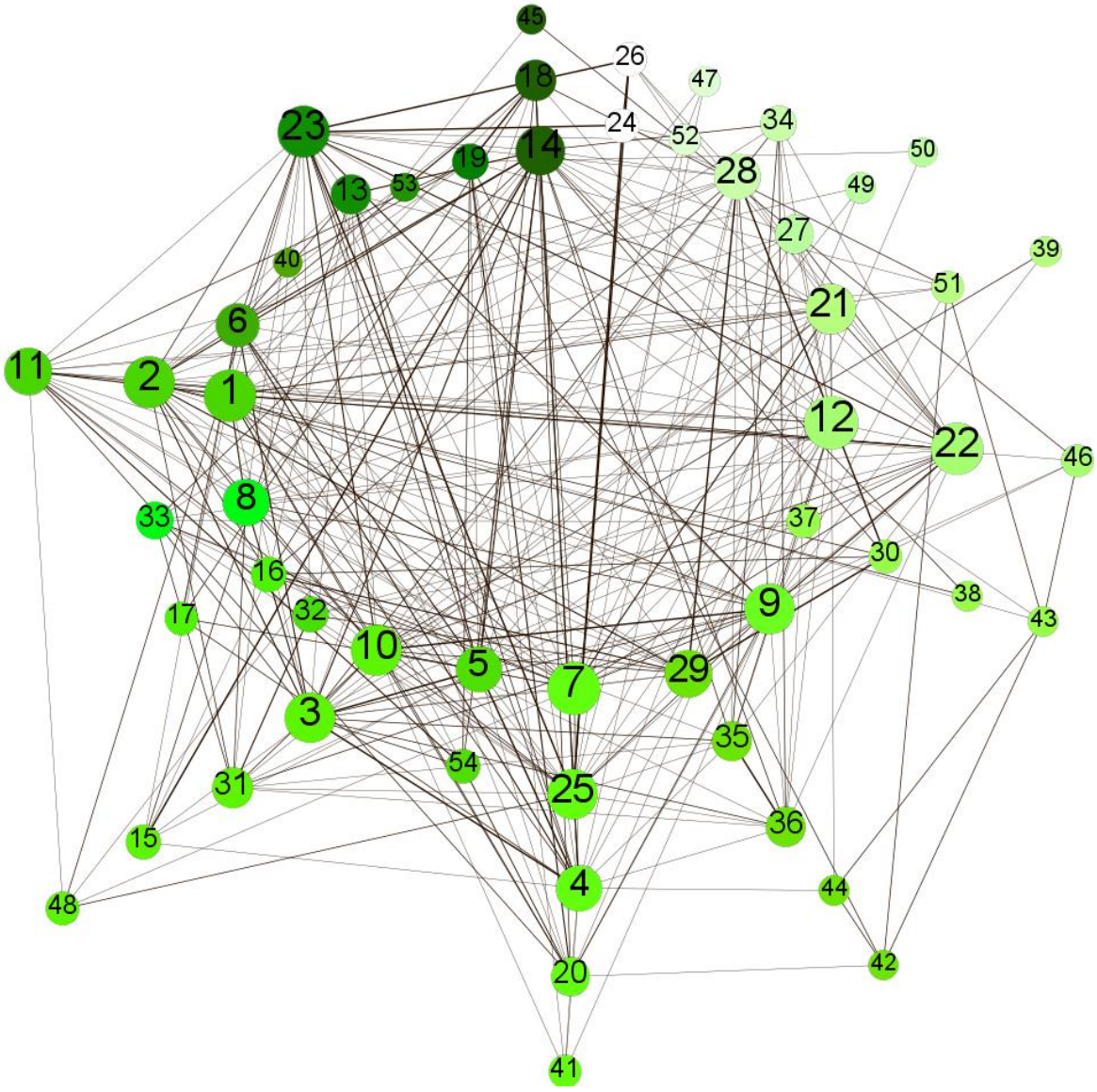
- Attribute `Age`

```
In [51]: nx.numeric_assortativity_coefficient(G,'Age')
Out[51]: 0.15167532832882372
```

The assortativity coefficient is the highest so far with the value 0.15. Nevertheless, it is still very low and based on the layout, we can not conclude that members of one age prefer to be friends with other members of the same/similar age. Possibly, some members hang out with members of similar ages but that is not significant enough to make a conclusion on strong associations of similar ages between connected nodes.

Assortativity Arrests

- Next the arrests attribute assortativity coefficient will be seen.



Members with the same number of arrests are found in the same line and with the same color. The lowest number of arrests (0) is defined with the white color, whereas the largest (23) is set with the darkest green color. Once more, no homophily is seen as edges are drawn everywhere despite the number of arrests.

NetworkX calculates the assortativity coefficient:

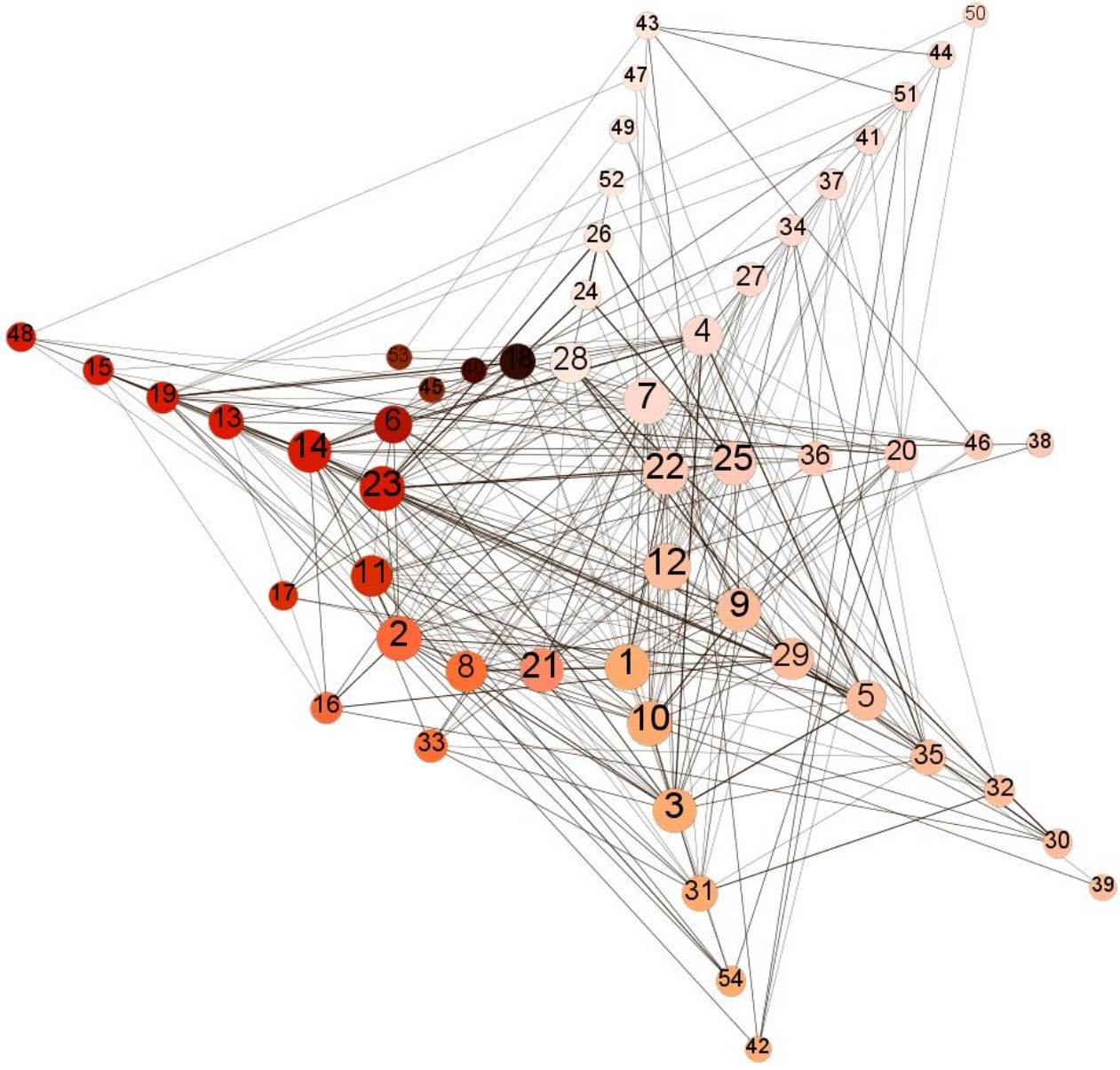
- Attribute `Arrests`

```
In [52]: nx.numeric_assortativity_coefficient(G,'Arrests')
Out[52]: 0.07194567096332875
```

Once more, a very low assortativity coefficient. No conclusion can be made on the friends of a member based on the member's number of arrests (we can not argue that a member tends to be friends with members with similar or different number of arrests)

Assortativity Convictions

- Lastly, the convictions attribute assortativity coefficient will be checked:



In the layout, the darker the color the more convictions it showcases. The lowest number of convictions (0) is found at 1 o'clock with the light orange color and so on. Once again, edges can be seen between all different sections of the graph in spite of the number of convictions.

NetworkX finds the attribute assortativity coefficient:

- Attribute `Convictions`

```
In [53]: nx.numeric_assortativity_coefficient(G,'Convictions')
Out[53]: 0.09718640151746134
```

Again, the assortativity coefficient is too close to 0 in order to make any valuable conclusions. The network is almost non-assortative when it comes to the attribute of convictions. The connections between the members do not lead to any strong or weak association of similar convictions between connected nodes.

Conclusions

No assortativity/homophily can be found in this network. Having checked all the different attributes of the members, the highest assortativity coefficient found corresponds to the age attribute with the value of 0.15. That correlation is basically a Pearson Correlation coefficient of r=0.15, which means a very little correlation. It could be argued that people tend to hang out with other similar aged ones; hence making it no surprise that it has the highest value so far. Moreover, the lack of homophily could be a result of the triadic closure. Members become friends and thus hang out or co-offend together with friends of their current friends despite differences in their characteristics.

Finding Assortativity in the network

After failing to find any homophily in the whole network, it was decided to check specific type of relationships only.

- Assortativity was found in the network only containing relationships of co-offending together. A new graph was created that did not contain any relationships of hanging out (no edges with weight 1, but edges with weights 2,3 or 4)

- Creating a graph that only includes the relationships of co-offending (edge values of 2, 3 and 4)

```
In [87]: edges_crimes = edges.replace([1],0)
         G2 = nx.from_pandas_adjacency(edges_crimes)
         node_attr = attributes.set_index('Id').to_dict('index') #creating dictionary to pass for the set attributes function
         nx.set_node_attributes(G2, node_attr)
         G2
Out[87]: <networkx.classes.graph.Graph at 0x1b465dafd60>
```

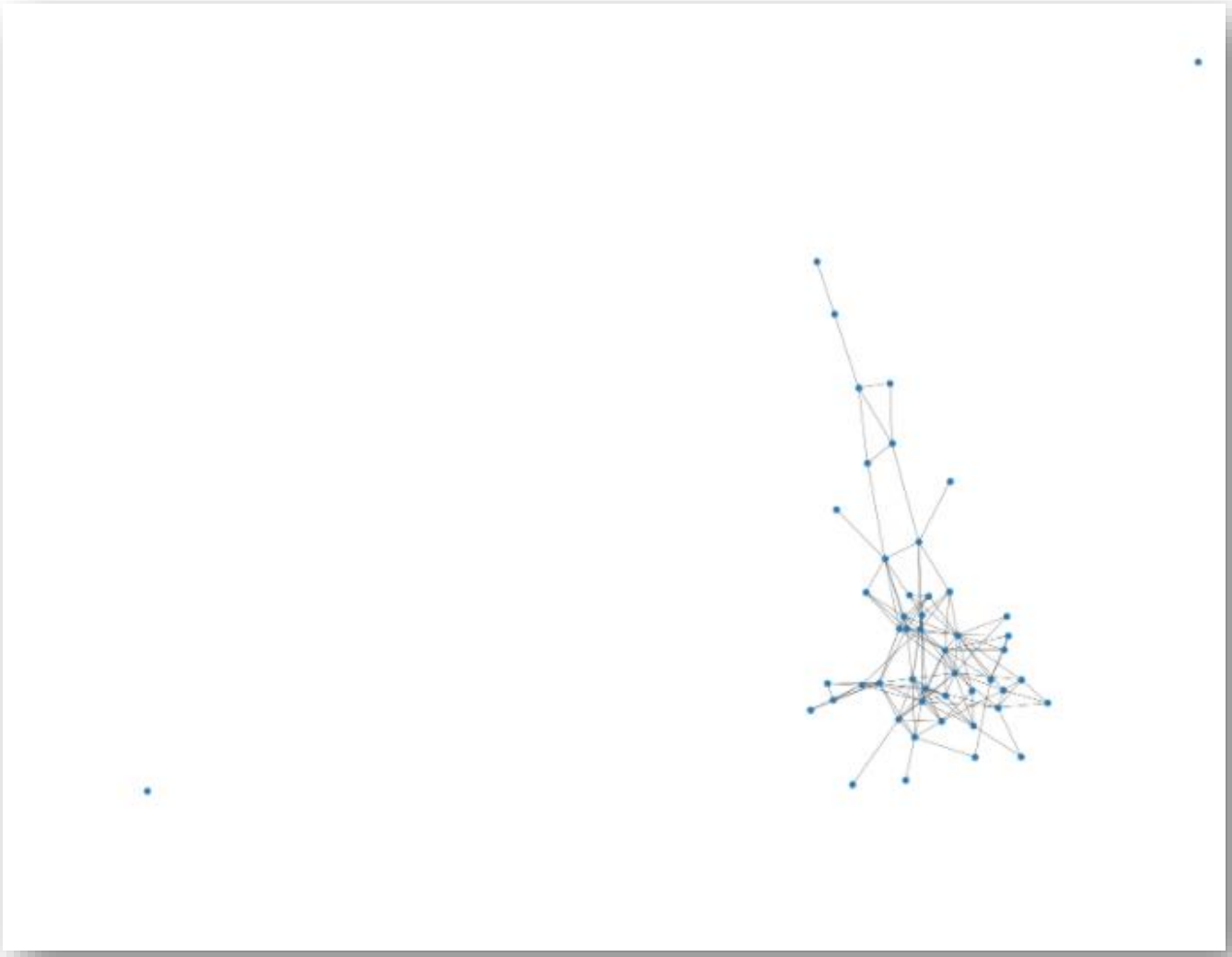
As we can see below, the assortativity coefficient of the attribute age is around 0.37. That could be called a low Pearson correlation, meaning that members slightly tend to

co-offend with other similar aged members. In conclusion, a weak association was found of similar age values between connected nodes.

- Attribute `Age` assortativity coefficient

```
In [104]: nx.numeric_assortativity_coefficient(G2,'Age')
Out[104]: 0.3711122206232127
```

We can see the graph below:



- Assortativity was also found in the network only containing relationships of co-offending serious crimes. A new graph was created that did not contain any relationships of hanging out and co-offending small out (no edges with weight 1 and 2, but edges with weights 3 or 4)

- Creating a graph that only includes the relationships of executing serious crimes (edge values of 3 and 4)

```
In [105]: edges_crimes = edges.replace([1,2],0)
          G3 = nx.from_pandas_adjacency(edges_crimes)
          node_attr = attributes.set_index('Id').to_dict('index') #creating dictionary to pass for the set attributes function
          nx.set_node_attributes(G3, node_attr)
          G3

Out[105]: <networkx.classes.graph.Graph at 0x1b4664afe20>
```

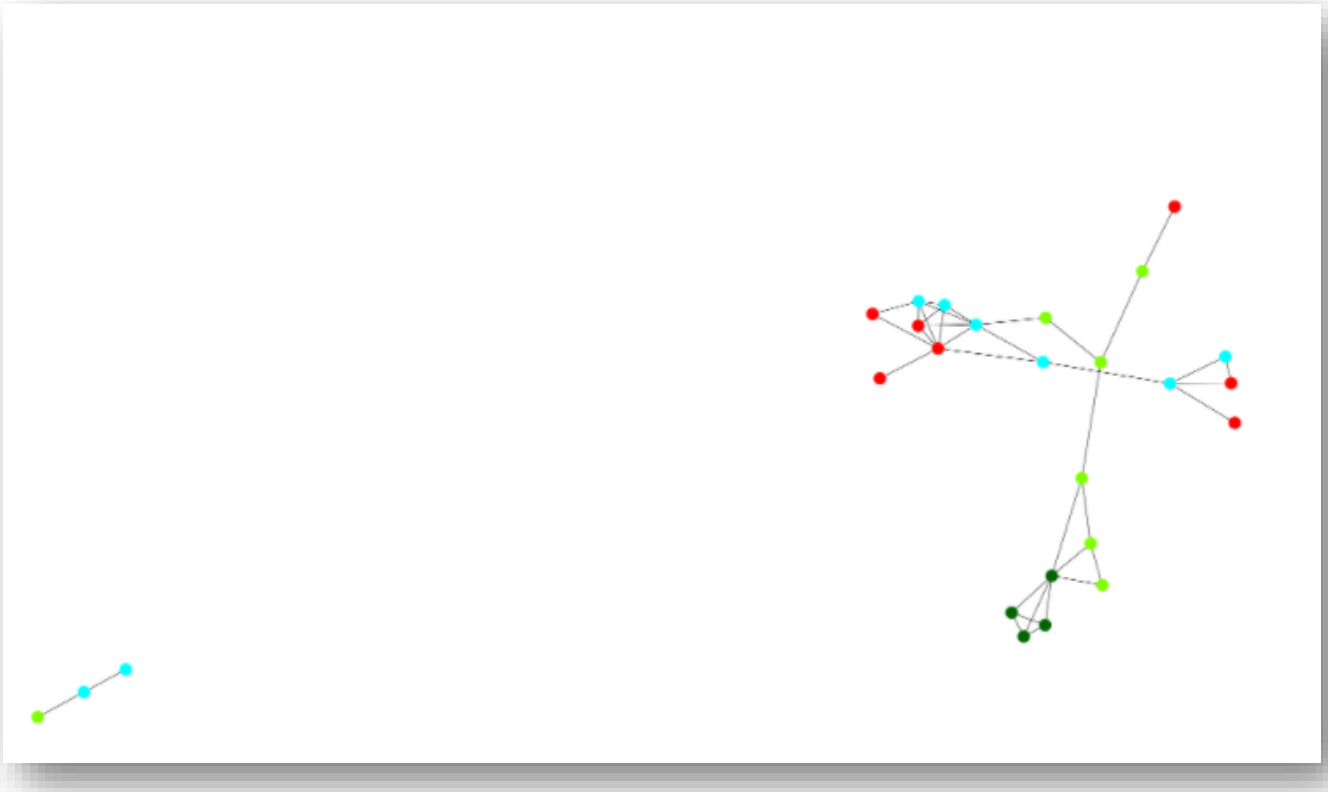
- Attribute `Birthplace` assortativity coefficient

```
In [107]: nx.attribute_assortativity_coefficient(G3,'Descript')

Out[107]: 0.4069907593411008
```

As it is clear above, the assortativity coefficient of the attribute Birthplace has a value of 0.41. That is considered a weak correlation. In other words, members slightly tend to commit serious crimes with members that were born in the same geographic regions.

Also, in the layout below we can see the network that only contains relationships of committing serious crimes together. The light green color describes members born in West Africa, the dark green members born in East Africa, the red members born in the UK and the blue members born in the Caribbean. We can see that members do not commit serious crimes with many other members and usually the members they co-offend with have the same color; hence the same birthplace.

```
In [179]: node_color=[]
          for node in G3.nodes(data=True):

              # if the node has the attribute group1
              if 'West Africa' in node[1]['Descript']:
                  node_color.append('chartreuse')

              # if the node has the attribute group1
              elif 'East Africa' in node[1]['Descript']:
                  node_color.append('darkgreen')

              # if the node has the attribute group1
              elif 'UK' in node[1]['Descript']:
                  node_color.append('red')

              # if the node has the attribute group1
              elif 'Caribbean' in node[1]['Descript']:
                  node_color.append('aqua')


          fig_dims = (70, 70)
          fig, ax = plt.subplots(figsize=fig_dims,)
          nx.draw(G3, node_color=node_color, )
```



Note: The members with no edges have been cut for visualization purposes.

- Lastly, assortativity was found in the network only containing relationships of co-offending serious crimes and being family. A new graph was created, that only contained the edges of value 4.

Creating a graph that only includes the relationships of executing serious crimes and being family (edge values of 4)

In [115]:
```
edges_crimes = edges.replace([1,2,3],0)
G4 = nx.from_pandas_adjacency(edges_crimes)
node_attr = attributes.set_index('Id').to_dict('index') #creating dictionary to pass for the set attributes function
nx.set_node_attributes(G4, node_attr)
G4
```

Out[115]:
```
<networkx.classes.graph.Graph at 0x1b4664affd0>
```

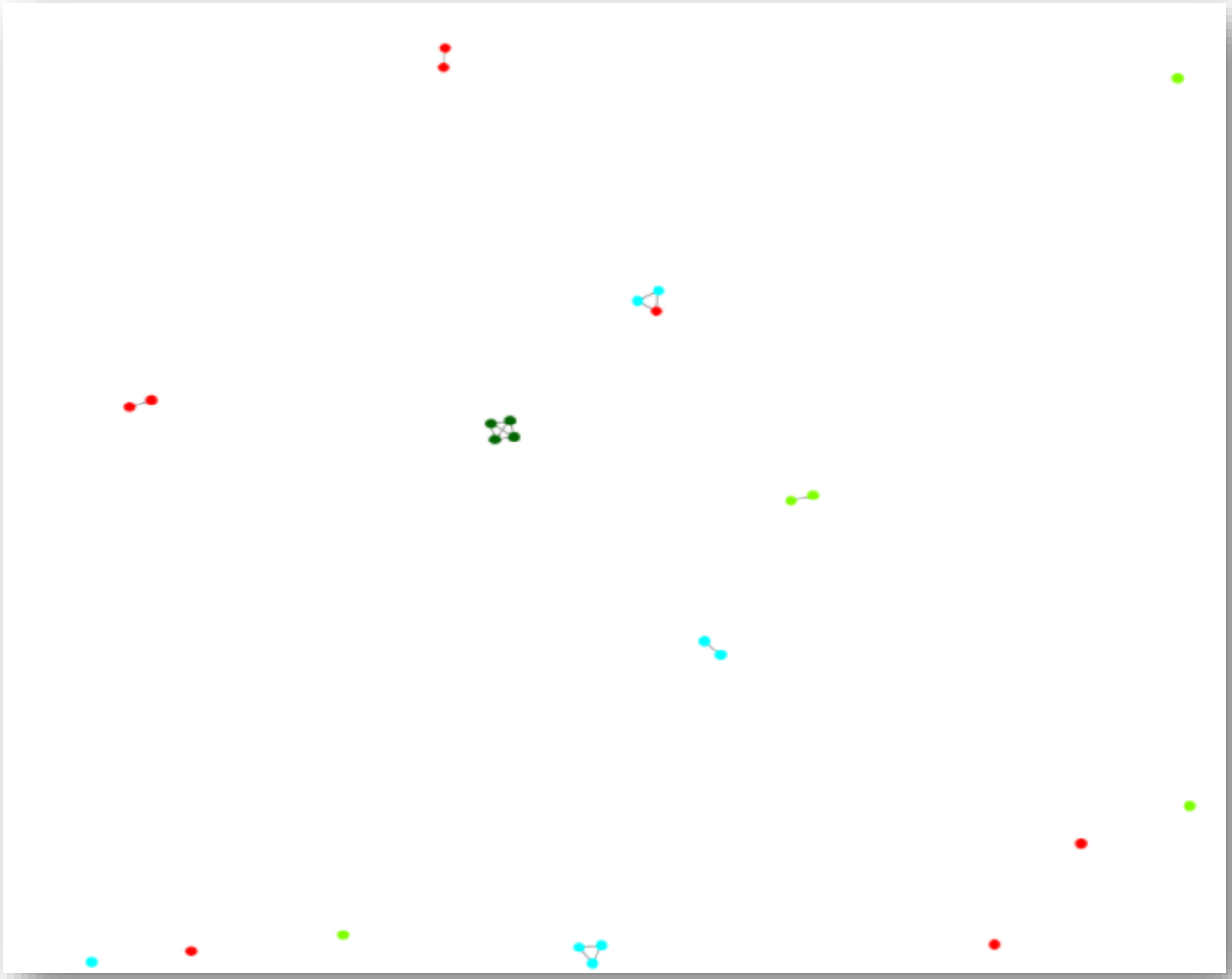
- Attribute `Birthplace` assortativity coefficient

In [117]:
```
nx.attribute_assortativity_coefficient(G4,'Descript')
```

Out[117]:
```
0.8160919540229885
```

It is noticed that the assortativity coefficient of the attribute birthplace has a value of 0.81. As a Pearson correlation that shows that the variables are highly correlated. In other words, members, that commit serious crimes and are also family, have a high tendency to share the same birthplace. That makes a lot of sense, since family members are usually born in the same geographic regions.

Here we can see the different components. Actually, the only time, that there is an exception in the tendency, is in the middle triangle, where one member has been born in the UK and the other two in the Caribbean.

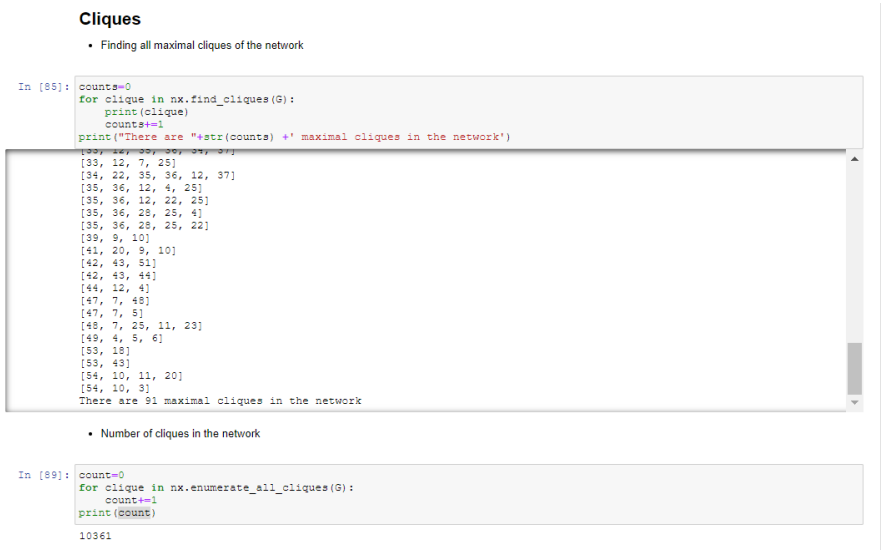


Network Communities

Cliques

The maximal cliques of the network will be found right now. In fact, a clique is a complete subgraph, meaning each pair of nodes are connected. Moreover, a maximal clique is one that can not be extended by adding one more node to it.

- Using networkX to find all the maximal and normal cliques:



Cliques

- Finding all maximal cliques of the network

```
In [85]: counts=0
         for clique in nx.find_cliques(G):
             print(clique)
             counts+=1
         print("There are "+str(counts) +' maximal cliques in the network')
```

```
[33, 12, 7, 25]
[34, 22, 35, 36, 12, 37]
[35, 36, 12, 4, 25]
[35, 36, 12, 22, 25]
[35, 36, 28, 25, 4]
[35, 36, 28, 25, 22]
[39, 9, 10]
[41, 20, 9, 10]
[42, 43, 51]
[42, 43, 44]
[44, 12, 4]
[47, 7, 48]
[47, 7, 5]
[48, 7, 25, 11, 23]
[49, 4, 5, 6]
[53, 18]
[53, 43]
[54, 10, 11, 20]
[54, 10, 3]
There are 91 maximal cliques in the network
```

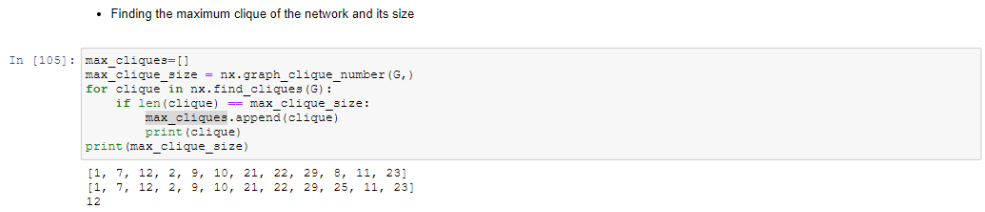
- Number of cliques in the network

```
In [89]: count=0
         for clique in nx.enumerate_all_cliques(G):
             count+=1
         print(count)
```

```
10361
```

We can see that our network has lots of maximal cliques. Actually, there are 91 different maximal cliques. Additionally, the network has even more normal cliques, as there are 10361 normal cliques, which would include all the different combinations of the maximal cliques by reducing the number of nodes.

- The maximum cliques are also found:



- Finding the maximum clique of the network and its size

```
In [105]: max_cliques=[]
          max_clique_size = nx.graph_clique_number(G,)
          for clique in nx.find_cliques(G):
              if len(clique) == max_clique_size:
                  max_cliques.append(clique)
                  print(clique)
          print(max_clique_size)
```

```
[1, 7, 12, 2, 9, 10, 21, 22, 29, 8, 11, 23]
[1, 7, 12, 2, 9, 10, 21, 22, 29, 25, 11, 23]
12
```

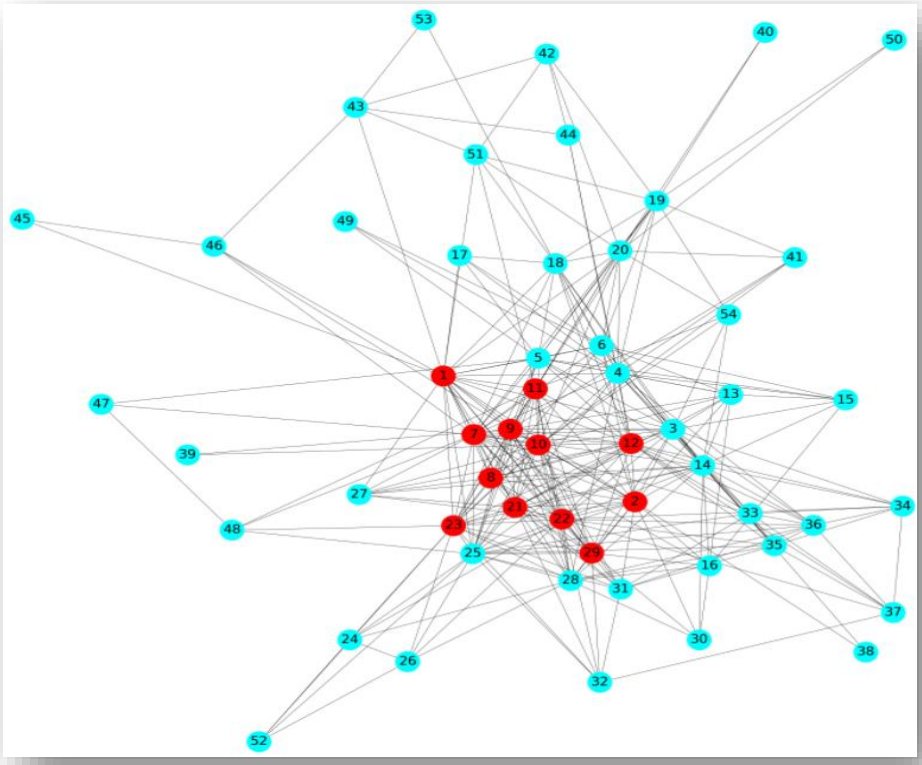
The maximum cliques are the ones with the largest possible size in the network. We can see that the maximum size of a clique is 12 and many familiar members are part of the maximum cliques e.g., member 1 (leader), 7 etc. Most of the members of the maximum cliques have the highest eigenvector centralities.

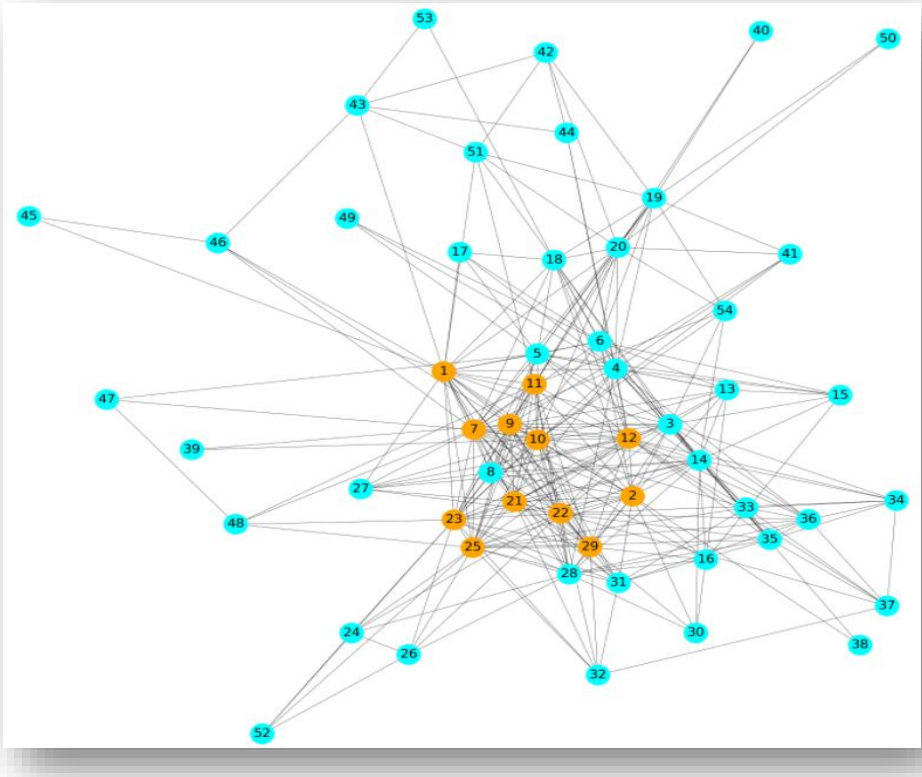
- The mean attributes of the maximum cliques are calculated:

Both the maximum cliques' members have an average of a bit less of 21 years old. The first maximum clique has an average of 11 arrests per member and 4.5 convictions per member. Slightly reduced numbers are found in the second maximum clique.

Definitely do those cliques play an important role in the running of the gang. Not only do all those members hang out (and probably co-offend) together but most of them are also considered very important ones based on the eigenvector centrality. Knowing their mean attributes could assist in recognizing the members and understanding how they intend to operate (e.g., a clique with a higher average of age could be more experienced and thus plan better their criminal activities)

- Now we can see the two maximum cliques inside the network. The red color is the first maximum clique and the orange color is the second one.


53
40
50
42
43
44
51
19
45
49
46
17
18
20
41
54
6
5
4
1
11
13
15
47
7
9
3
10
12
39
14
8
27
2
34
21
22
33
36
48
23
25
29
35
16
28
31
37
24
30
38
26
32
52


53
40
50
42
43
44
51
19
45
49
46
17
18
20
41
54
6
5
4
1
11
13
15
47
7
9
3
10
12
39
14
8
27
2
34
21
22
33
36
48
23
25
29
35
16
28
31
37
24
30
38
26
32
52

Cliques conclusions

The existence of many cliques inside the graph is not a surprise for us. Having seen the triadic closure existence, it is expected for members with mutual friends to become friends over time. Furthermore, the network describes a gang, that is a group of people, so we can expect people to make cliques in their social setting. Lastly, the network also describes co-offending together. Due to the gang setting, it is more likely for members to offend together, especially more serious crimes, in order to execute their actions as planned. Now, if a group of members co-offends together, they become a clique immediately because they all work together and thus edges should be formed between all the pair of members.

Communities

After having scrutinized the cliques, we can examine the communities of the graph. A community is a group of nodes, so that nodes inside the group are connected with many more edges than between groups. The modularity measure will be used to detect communities:

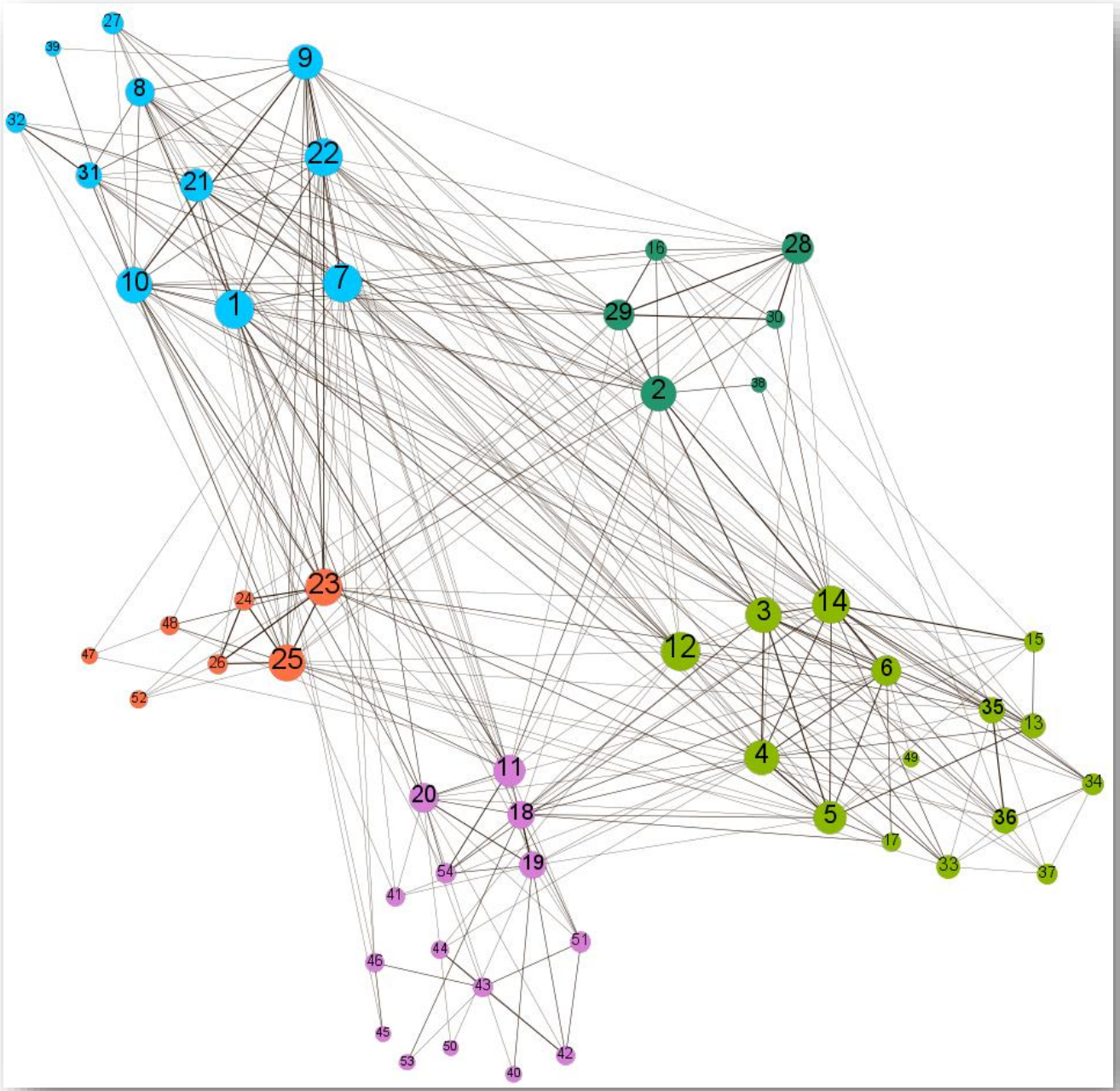
Modularity: 0,336
Modularity with resolution: 0,336
Number of Communities: 5

Algorithm:

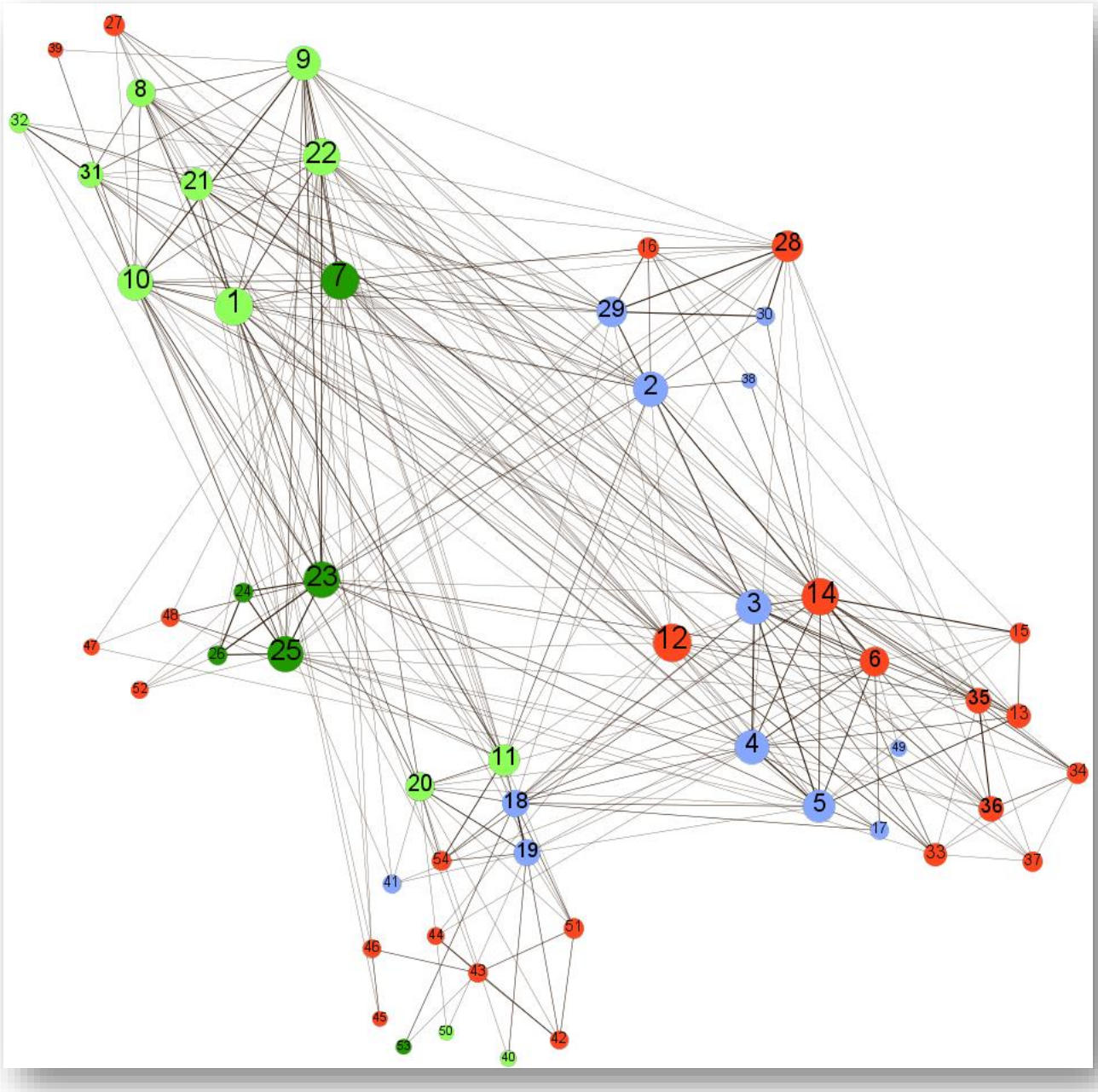
Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre, *Fast unfolding of communities in large networks*, in Journal of Statistical Mechanics: Theory and Experiment 2008 (10), P1000

As it is clear to see, five communities were detected. In addition, the modularity score is 0.336, which is very decent as the maximum modularity score usually lies between 0 and 1.

- Now let's see the communities in the graph:



Each community has a different color and they come with different sizes. Despite the existence of communities, there are also many edges between the communities. That is because of the triadic closure and the gang setting. However, we can not understand much from the communities until we include the birthplace of the nodes:



The colors are based on the birthplace: light green for West Africa, dark green for East Africa, red for the UK and blue for the Caribbean. Also, it has to be explained that those colors just describe the birthplace. In other words, a member born in the UK could have any kind of background like Caribbean, East or West African. The sure assumption we can make is that members, that were born in West Africa, East Africa or Caribbean, definitely are of that background.

Community conclusions

It is believed that those communities are mostly based on ethnic background:

- The top left community consists of 11 members. Two of those members were born in the UK, seven were born in West Africa and one in East Africa. It is assumed that this community concerns African background members and especially of the region of West Africa. The two members born in the UK are assumed to have a West African background.
- The middle-left community consists of 7 members, three of whom were born in the UK and 4 of whom were born in East Africa. This community is believed to concern East African background members. In fact, the members born in the UK are most likely East African because they only have edges to other East African members.
- The top-right community has a size of 6 members. In detail, four were born in the Caribbean and two in the UK. This community is most likely of Caribbean background. The two members that were born in the UK are possibly Caribbean because many of their edges are connected to other Caribbean members.
- The bottom-right community has the largest size found with 15 members. Five of those members were born in the Caribbean and the other ten were born in the UK. This community is probably of mixed backgrounds. Because many members were born in the UK, we can not know their background. One case would be to assume that because they are all in the same community, their background must be the same too. That can not be said because many members of this community, who were born in the UK, also have links to other communities. Moreover, it has to be understood that intercultural ties are very likely to happen as well because all those members belong to the same gang.
- The bottom-middle community has also the size of 15. This community is considered to be multi-cultural because of the various birthplaces of its members. Specifically, one member was born in East Africa, three in the Caribbean, four in West Africa and seven in the UK. That also shows that close intercultural relationships are possible and very likely to happen in the gang.

To sum up, many communities are based on their ethnic background. That can assist us to predict the exact background of members that were born in the UK and thus their real background is unknown. For example, if there was an Id check of a member (and the member has a British ID), where only the birthplace is written, we can figure out the member's background in most cases by the community he belongs to or the members he hangs out with. On the other hand, intercultural relationships are also likely to be formed in the network because of the co-existing of members of different cultures.

Association Rules between backgrounds

At this point, we will try to find association rules for different backgrounds when co-offending together. In detail, we will use the graph with the edges of weight 2, 3 and 4 (co-offending relationships)

- Data preprocessing

Association rules

Creating a dataframe that contains the believed background for each member. The rules are:

- A member born in East Africa, West Africa and Caribbean are surely of that background
- Members born in the UK, that the majority of their edges to a particular background, are considered of the same background (can be seen with communities)
- The rest of the members born in the UK are considered unknown background

In [106]:
```
background= attributes[['Id','Descript']].copy()
background = background.sort_values(by='Id')
background = background.reset_index(drop=True)
background.index += 1 #setting index to be the same as ID
background
```

Out[106]:

	Id	Descript
1	1	West Africa
2	2	Caribbean
3	3	Caribbean
4	4	Caribbean
5	5	Caribbean
6	6	UK
7	7	East Africa
8	8	West Africa
9	9	West Africa
10	10	West Africa
11	11	West Africa
12	12	UK

- Fixing the values that were born in the UK, based on the relationships they have developed and the community they belong to.

In [107]:
```
background.loc[47,'Descript'] = 'East Africa' #only has edges to other east african nodes
background.loc[48,'Descript'] = 'East Africa' #only has edges to other east african nodes
background.loc[52,'Descript'] = 'East Africa' #only has edges to other east african nodes
background.loc[39,'Descript'] = 'West Africa' #only has edges to other West african nodes
background.loc[27,'Descript'] = 'West Africa' #majority of edges to other West african nodes
background.loc[16,'Descript'] = 'Caribbean'   #only has edges to other Caribbean nodes
background.loc[28,'Descript'] = 'Caribbean'   #has family edges to other Caribbean nodes
background.loc[15,'Descript'] = 'Caribbean'   #majority of edges to other Caribbean nodes
background.loc[6,'Descript'] = 'Caribbean'    #majority of edges to other Caribbean nodes
background.loc[13,'Descript'] = 'Caribbean'   #majority of edges to other Caribbean nodes
uk_index = background.loc[background.Descript == 'UK'].index #setting the rest as 'unknown'
background.loc[uk_index,'Descript']='unknown'
background
```

Out[107]:

We are only interested in the backgrounds of East Africa, West Africa, and Caribbean. Members born in the UK can be any of the above three backgrounds. We tried to find the background of members, that were born in the UK, through their edges and communities but that was not always possible.

- Importing the right library

```
In [152]: from apyori import apriori
```

- Creating a list of all the edges

```
In [153]: edges_list = list(nx.edges(G2))
```

- Creating a list containing the edges. Each edge consists of a pair of nodes. Instead of the nodes ids, we use the nodes backgrounds in order to examine the association rules for the backgrounds.

```
In [154]: edges_background=[]
          for edge in edges_list:
              first_background = background.loc[background.Id == edge[0]]['Descript'].values[0]
              second_background = background.loc[background.Id == edge[1]]['Descript'].values[0]
              edges_background.append([first_background,second_background])
          edges_background
```

```
Out[154]: [['unknown', 'West Africa'],
           ['unknown', 'Caribbean'],
           ['unknown', 'unknown'],
           ['unknown', 'West Africa'],
           ['unknown', 'Caribbean'],
           ['East Africa', 'Caribbean'],
           ['unknown', 'unknown'],
           ['unknown', 'unknown'],
           ['unknown', 'Caribbean'],
           ['East Africa', 'East Africa'],
           ['East Africa', 'East Africa'],
           ['unknown', 'unknown'],
           ['unknown', 'unknown'],
           ['unknown', 'unknown'],
           ['unknown', 'unknown'],
           ['unknown', 'unknown'],
           ['unknown', 'Caribbean'],
           ['West Africa', 'Caribbean'],
           ['West Africa', 'West Africa'],
           ['Caribbean', 'unknown'],
```

- Only keeping the known background edges:

```
In [155]: edges_background_known=[]
          for edge in edges_background:
              if (edge[0]!='unknown') & (edge[1] != 'unknown'):
                  edges_background_known.append(edge)
          edges_background_known
```

```
           ['Caribbean', 'Caribbean'],
```

A list is created with the different edges (pair of nodes), which is necessary for the apriori algorithm to run. However, instead of the id of the nodes, we keep the background of the nodes. Consequently, we have a list of edges that connect backgrounds (pairs of backgrounds). Furthermore, we only keep the edges of known backgrounds (cutting edges that contain at least one unknown background)

- Now we can see the association rules between the same backgrounds:

- Association rule: East Africa -> East Africa

In [157]:
```
counter = 0
counter_sum = 0
for place in edges_background_known :
    if (place[0] == place[1]) & (place[1]=='East Africa'):
        counter+=1
    if (place[1]=='East Africa') | (place[0]=='East Africa'):
        counter_sum += 1
print('Support: '+ str(counter/len(edges_background_known)))
print('Confidence: '+ str(counter/counter_sum))
```

```
Support: 0.08791208791208792
Confidence: 0.3076923076923077
```

- Association rule: West Africa -> West Africa

In [158]:
```
counter = 0
counter_sum = 0
for place in edges_background_known :
    if (place[0] == place[1]) & (place[1]=='West Africa'):
        counter+=1
    if (place[1]=='West Africa') | (place[0]=='West Africa'):
        counter_sum += 1
print('Support: '+ str(counter/len(edges_background_known)))
print('Confidence: '+ str(counter/counter_sum))
```

```
Support: 0.24175824175824176
Confidence: 0.44
```

- Association rule: Caribbean -> Caribbean

In [159]:
```
counter = 0
counter_sum = 0
for place in edges_background_known :
    if (place[0] == place[1]) & (place[1]=='Caribbean'):
        counter+=1
    if (place[1]=='Caribbean') | (place[0]=='Caribbean'):
        counter_sum += 1
print('Support: '+ str(counter/len(edges_background_known)))
print('Confidence: '+ str(counter/counter_sum))
```

```
Support: 0.2967032967032967
Confidence: 0.5510204081632653
```

The conclusions based on the known backgrounds are:

- If a member is East African and co-offends, there is a 26% chance that the member will co-offend with another East African.
- If a member is West African and co-offends, there is a 44% probability that the member will co-offend with another West African.
- Lastly, if a member is Caribbean and co-offends, there is a 55% probability that the member will co-offend with another Caribbean.

➢ Now the association rules between different backgrounds will be seen:

- Finding the association rules

```
In [156]: association_rules = apriori(edges_background_known, min_support=0.05, min_confidence=0.28, min_lift=0.1, min_length=2)
          association_results = list(association_rules)
          association_results
```

```
Out[156]: [RelationRecord(items=frozenset({'Caribbean'}), support=0.5384615384615384, ordered_statistics=[OrderedStatistic(item
          s_base=frozenset(), items_add=frozenset({'Caribbean'}), confidence=0.5384615384615384, lift=1.0)]),
           RelationRecord(items=frozenset({'East Africa'}), support=0.2857142857142857, ordered_statistics=[OrderedStatistic(it
          ems_base=frozenset(), items_add=frozenset({'East Africa'}), confidence=0.2857142857142857, lift=1.0)]),
           RelationRecord(items=frozenset({'West Africa'}), support=0.5494505494505495, ordered_statistics=[OrderedStatistic(it
          ems_base=frozenset(), items_add=frozenset({'West Africa'}), confidence=0.5494505494505495, lift=1.0)]),
           RelationRecord(items=frozenset({'West Africa', 'Caribbean'}), support=0.17582417582417584, ordered_statistics=[Order
          edStatistic(items_base=frozenset({'Caribbean'}), items_add=frozenset({'West Africa'}), confidence=0.326530612244898,
          lift=0.5942857142857143), OrderedStatistic(items_base=frozenset({'West Africa'}), items_add=frozenset({'Caribbean'}),
          confidence=0.32, lift=0.5942857142857143)]),
           RelationRecord(items=frozenset({'East Africa', 'West Africa'}), support=0.13186813186813187, ordered_statistics=[Ord
          eredStatistic(items_base=frozenset({'East Africa'}), items_add=frozenset({'West Africa'}), confidence=0.4615384615384
          6156, lift=0.84)])]
```

The most valuable conclusions, that based on the known backgrounds, are:

- If a member is Caribbean and co-offends, there is a probability of 33% for it to co-offend with a West African member. Likewise, if a member is west African and co-offends, there is a 32% probability that member will co-offend with another Caribbean.
- If a member is East African and co-offends, there is a probability of 46% that the co-offending partner is West African.

In conclusion, based on the known information, West African and Caribbean members are far more likely to co-offend with similar background members with 44 and 55 percent, respectively. On the other hand, East African members are more likely to co-offend with a West-African member with 46% (knowing that the member co-offends).

Clustering Profiles

At this point, different profiles of the gangsters will be discovered based on the age and the number of arrests and convictions by using clustering and specifically the k-means algorithm.

The steps are shown below:

CLUSTERING

At this point a clustering will be executed in order to make different profiles for the members

- Only the numerical attributes will be used

In [228]:
```
cluster_attr = attributes.sort_values(by='Id', ascending=False)[['Arrests','Convictions','Age']]
#sorted values for right order in the plot
```

- The attributes need to be standarized in order to execute the k-means algorithm.
- StandardScaler transforms the data so that each attribute has a mean of 0 and standard deviation of 1

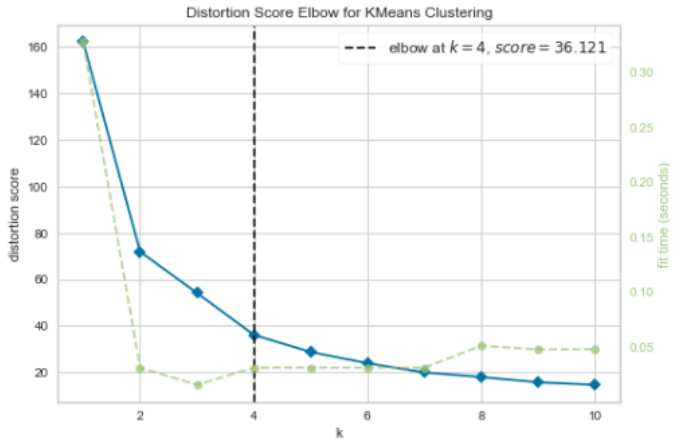
In [218]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X = scaler.fit_transform(cluster_attr)
scaled_data = pd.DataFrame(X, index=cluster_attr.index,
                           columns=cluster_attr.columns)
scaled_data
```

Out[218]:

	Arrests	Convictions	Age
47	0.171311	-0.056148	0.062541
53	1.582451	1.873303	2.689251
46	-1.396622	-1.158691	-0.687948
45	-0.926242	-0.883055	-1.063192
11	-1.083035	-0.883055	-0.312704
23	-1.083035	-1.158691	-0.312704
44	0.328105	1.322031	0.062541
43	-1.396622	-1.158691	-1.438437
42	-0.769449	-0.607420	-0.687948
41	2.052831	1.873303	-0.312704
40	-0.455862	-0.883055	-0.687948
39	-0.612655	-1.158691	-0.687948
38	-0.455862	-0.056148	-0.687948

- Elbow method is used in order to find the optimal number of clusters

In [219]:
```
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans


kmeans = KMeans(random_state=0)
visualizer = KElbowVisualizer(kmeans, k=(1,11))

visualizer.fit(scaled_data)
visualizer.show()
```



Out[219]: <matplotlib.axes._subplots.AxesSubplot at 0x25b5a926370>

- The kmeans algorithm is executed for 4 clusters as the elbow method advised.

In [230]:
```
kmeans = KMeans(n_clusters=4, random_state=1).fit(scaled_data)
pd.DataFrame(kmeans.cluster_centers_, columns=scaled_data.columns)
```

Out[230]:

	Arrests	Convictions	Age
0	1.124132	1.025192	0.004811
1	-0.377465	-0.430225	0.571801
2	1.331582	1.818175	2.088860
3	-0.726687	-0.745238	-0.841457

The centroids are the most important factor of this data mining execution. Knowing that the values are standardized, meaning they have a mean of 0 can lead us to discover different profiles.

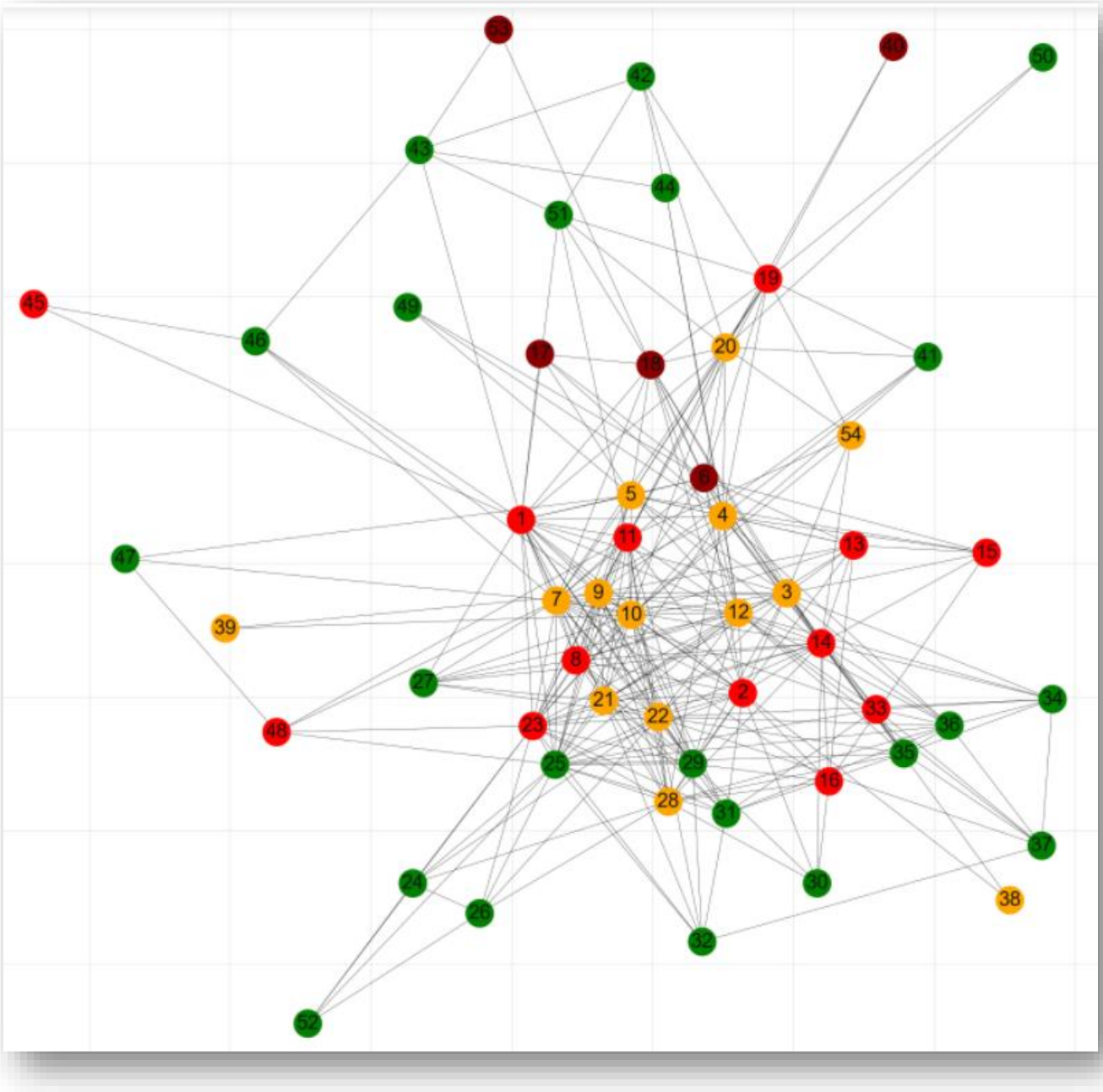
- The first cluster centroid describes members with far above the average number of arrests and convictions and average values of age.
- The second cluster describes members with below the average number of arrests and convictions and ages above the average
- The third cluster describes members with far higher than the average number of arrests, convictions and age
- The fourth cluster describes members with far lower than the average number of arrests, convictions and age.

Here we can see the average numbers of each category

In [232]:
```
attributes[['Arrests','Convictions','Age']].mean()
```

Out[232]:
```
Arrests          9.907407
Convictions      4.203704
Age             19.833333
dtype: float64
```

The graph can be seen here:



Each color defines a different profile:

- The green color describes members with far less arrests, convictions and age than the average. That means each has much less than 10 arrests, 4 convictions and 20 years old of age, which are the averages. Those members can be called the fresh, young, most innocent and inexperienced ones and thus the ‘green’ color. Lastly, most of them are found at the borders of the graph.

- The yellow color corresponds to members with slightly below the average arrests and convictions in their past and with age slightly above the average (20 + years old). Those are the members, that have had some experience in the gang, but they have not developed a huge criminal record. They could be considered rising stars in the mob organization. Also, they are found in the center of the graph. Many of those members have the highest eigenvector centralities and possibly could consist the organization of the gang and thus not commit many crimes by themselves (but organize them).
- The red color corresponds to members with far above the average arrests and convictions in their past and average age (around 20 years old). Despite their young age, they have already turned into criminals as their criminal records can indicate. Those members could also be the executers of the gang plans and consequently have more arrests and convictions.
- The dark-red color describes members with the highest number of arrests and convictions and are also the oldest members. Those surely have the most experience in the gang. Very interesting is that those members also have the lowest eigenvector centralities, making them the least significant members. It is very likely that those members directly execute the plans of the organization; hence they get arrested and convicted.

Conclusions

This network analysis scrutinized a London street gang from the years of 2005 – 2009. In detail, the network was depicted as a graph, where each member of the gang is a node and the relationships between members are the edges. Depending on the relationship, the edge has a different weight. Also, the basic topological properties were presented in order to get a first understanding of the graph. After, the different attributes of birthplace, age, arrests, convictions, prison, music and ranking, that each member has, were introduced and examined. Then, in the component section different versions of the network were showcased based on the type of relationship we wanted to emphasize. Specifically, members that do not co-offend could be spotted as well as the members that commit the serious crimes.

Moving forward, the centrality measures of the graph were explored. The degree helps us find newcomers in the gang as well as the popularity of each member. Furthermore, the betweenness centrality relates with the member's ability to transfer knowledge throughout the gang. The highest betweenness centralities are considered to be the unifiers and the coordinators. Moreover, closeness centrality is connected to the ability of a member to reach others. Lastly, the eigenvector centrality assists us to discover the most important members, which could consist the organization of the gang. Nonetheless, node member 1 is considered to be the leader of the gang due to his high scores in all the centrality measures.

The high clustering coefficients, together with the plethora of unique triangles and the lack of bridges indicate the existence of triadic closure in the gang. The reasons behind the triadic closure existence are also explained in this particular case. Moreover, valuable insight was created for the way new-members are embraced in the mob.

No homophily and assortativity was found in the normal network, that contains all types of relationships, after examining each and every attribute of the members. However, assortativity was found in the different versions of the graph which contained particular types of relationships only. In fact, a weak association was found between similar aged members when it comes to co-offending. Also, a weak association was found between members, that were born in the same place, when it comes to committing serious crimes.

The cliques and communities were also identified in the graph. The high number of cliques and maximal cliques validates the existence of triadic closure. Also, the maximum cliques are showcased in the graph, after being described. The communities were found with the use of modularity and they are directly connected to the backgrounds of the members. Often members of the same background hang out / co-offend together, thus making their own community. Other times, more intercultural relations are made; hence multicultural communities are formed. Most importantly, the

communities can help us identify the culture of members that were born in the UK, whose background is unknown to us.

Last but not least, data mining functions were used in order to extract extra intelligence from the dataset. According to the association rules and the known backgrounds, members of one background are more likely to co-offend with members of a particular background. In other words, knowing the background of a member that co-offends, can direct us to the background of the member's partner in crime. In addition, the k-means clustering algorithm identified specific profiles of members based on their age, arrests and convictions, which can help recognize not only the members orchestrating the actual criminal activities but also the members with the most violent past and the members with the most violent future.

Bibliography

Campbell, D. (2019, July 4) Inside the 21st-century British criminal underworld. Theguardian. Retrieved from https://www.theguardian.com/world/2019/jul/04/inside-the-21st-century-british-criminal-underworld

(2016, August 31) London Gang. Sites Google. Retrieved from: https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/londongang