Problem Set 4 Part II

By: Kali Benavides

Collaborators: Kailin Graham, Christian Cmehil-Warn, Lan Ha, Les Armstrong, Maja Svanberg

Nodes = Offenders

Edges = Arrested for the same crime event

Weight of edge = # of crimes for which any two individuals were arrested together

Data set = Cooffending.csv, Quebec Arrest records 2003-2010

Description of data = DataDescription.txt

a) How many data points in the data set?
1,280,459 data points with 15 parameters, of these 467 are duplicates. Which makes up 0.036% of the data set. I will remove these duplicate values and keep the first incidence. This leaves a total number of 1,279,992 sample points. There were no NaN values in the dataset.
There were two errors noted in birth year (Column='Naissance'), however these were left in under the assumption that the other data values of these rows were accurate.

	NoUnique	Naissance	SEXE	SeqE	dateInf	NCD1	NCD2	NCD3	NCD4	MUN	ED1	Jeunes	Adultes	Date	annee
0	1	1007	F	1085034	20051217	3530				58227	2	0	1	12/17/2005	2005
1	2	1828	F	1431379	20080423	1430				94068	5	0	1	04/23/2008	2008

b) How many different offenders?
Case number (NoUnique) refers to the offender. There are 539,593 offenders

c) How many different crimes?
Event number (SeqE) refers to the crime event. There are 1,164,836 crime events.
There are 302 different crime categories

a. How many crimes per year? Utilizing the pandas group by function to group by year (Column titled 'annee') and then counting the number of crime events for each year.

Year	Number of Crime Events
2003	122263
2004	133675
2005	188055
2006	203305
2007	214245
2008	220690
2009	197752
2010	7

*It appears that 2010 only has data for the month of January

d) Which crimes involved the greatest number of offenders?

SeqE	Number of Offenders Involved	Municipality
27849	156	66023
876159	102	12072
445040	77	66023
60815	60	54048
23526	60	75017

a. Crimes/Number of offenders involved and which municipality it occurred in?

The crime SeqE #27849 has the most offenders involved and occurred in municipality 66023, and the crime type is “TOUTE AUTRE INF. AU C.CR.”

Code for Parts a-d

```
#Drop all duplicate values keeping only the first instance
df.drop_duplicates(keep = 'first', inplace = True)
df.shape
```

```
(1279992, 15)
```

```
#Count the number of unique values in each column
print(df.nunique())
```

```
NoUnique      539593
Naissance        110
SEXE               2
SeqE         1164836
dateInf         2561
NCD1             295
NCD2             244
NCD3             178
NCD4             116
MUN             1342
ED1               99
Jeunes            14
Adultes           50
Date            2561
annee              8
dtype: int64
```

```
#Look at how many different values there are in the 4 crime rows to count the number of different crime types
CrimeTypes = pd.unique(df[['NCD1', 'NCD2', 'NCD3','NCD4']].values.ravel())
len(CrimeTypes)
```

```
df.loc[df['annee'] == 2010]
```

```
#Group crime events and count the number of offenders associated with each crime event
df_crimecount = df.groupby(['SeqE'])['NoUnique'].count().reset_index(
  name='Count').sort_values(['Count'], ascending=False)
```

```
#Displaying crime events sorted from largest number of associated co-offenders to lowest number
df_crimecount.head()
```

```
#Group the crime events by municipality to view the municipality with the greatest number of crime events
df_city = df.groupby(['MUN'])['SeqE'].count().reset_index(
  name='Count').sort_values(['Count'], ascending=False)
df_city.head()
```

E)

Co-offending network

- Created a dataframe of just the offenders (No. Unique) and Crime Event (SeqE)
- Merged a duplicate of that dataframe onto itself with the Crime Events (SeqE) in common
- This yielded a dataframe with all of the crime events and then a column for offender i and another column for offender j.
- Then I removed those rows where offender i = offender j, these represent rows listing crime events that an offender has in common with themselves, these do not represent an edge we want to include in the adjacency matrix
- Also removed those rows which were just a duplicate of an already listed connection between offenders for a given crime event.
- In the entire dataset there are 539,593 offenders. After removing solo offenders there are now a total 121,159 different offenders. This indicates a total of 418,434 solo offenders.
- There are 84,038 different crime events that involve multiple offenders

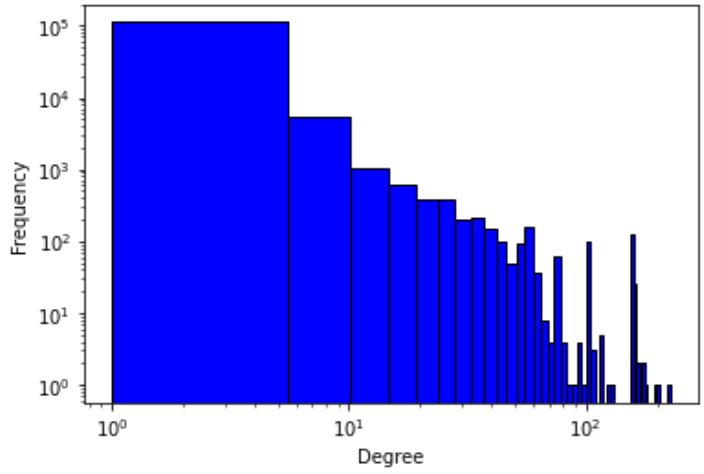
Co-offending Network Attributes:

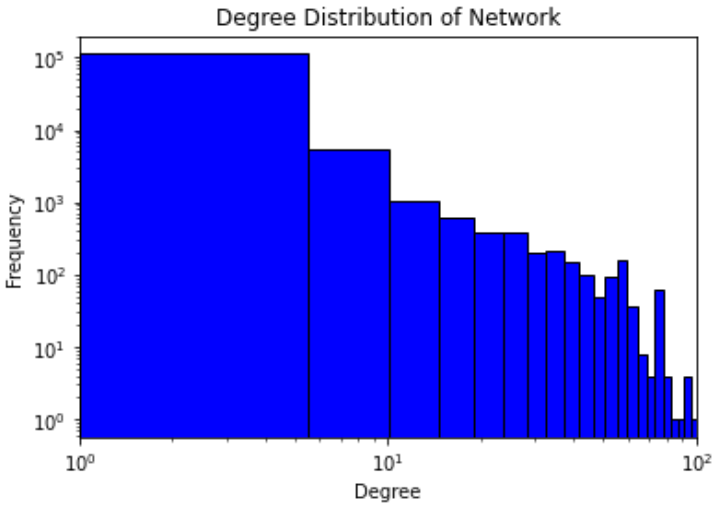
There are 121,159 nodes in the network

There are 418,434 solo offenders (These were not included in the network)

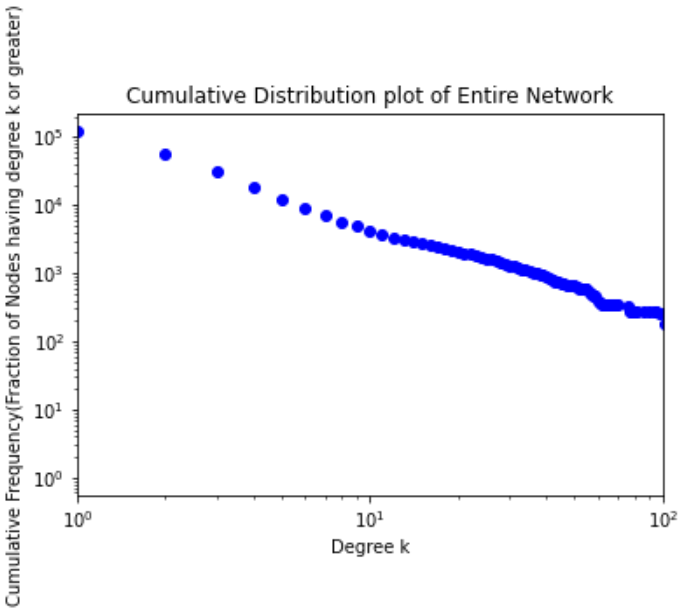
There are 178,413 edges

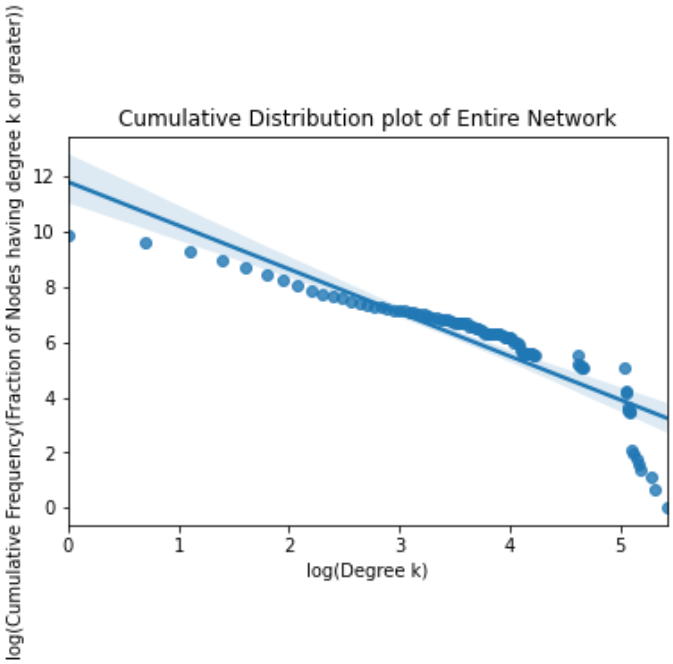
F)





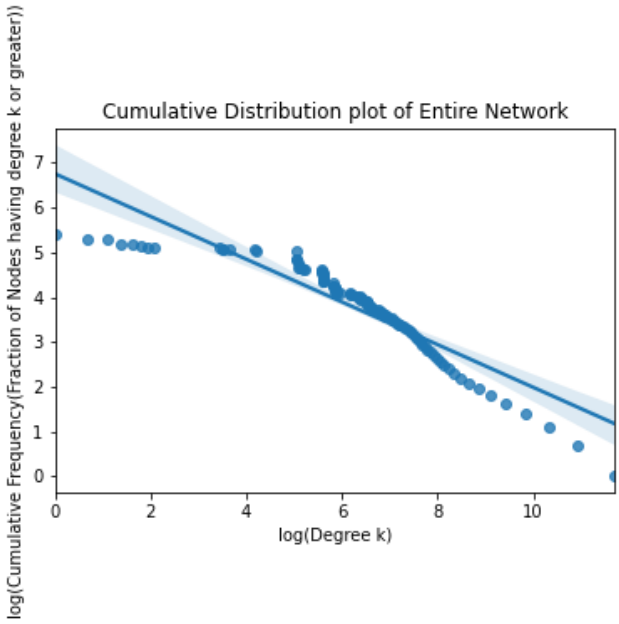
**Plotted only up to degree 10^2





Regression Line on log(Degree) and log(Cumulative Frequency):

log(Cumulative Frequency) = -1.577*(log(degree))+11.8



Regression Line on log(Degree) and log(Cumulative Frequency), Only up to log(degree) = 10^2

log(Cumulative Frequency) = -0.477*(log(degree))+6.758

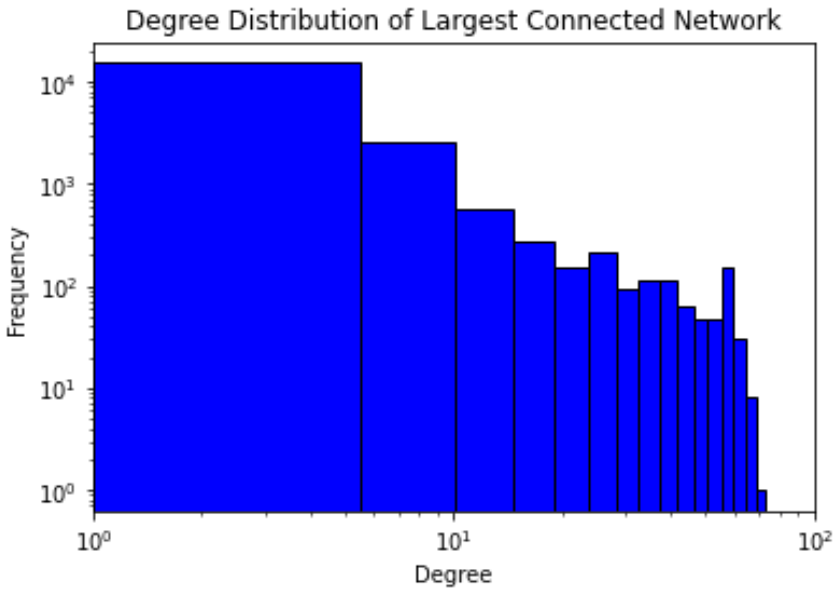
The degree distribution of the entire network does appear to exhibit a power law degree distribution which follows a linear relationship when represented in log-log form.

There is more noise in the right-hand side of the distribution associated with few instances of nodes having much higher degrees.

G) 36,098 connected components exist in this network.

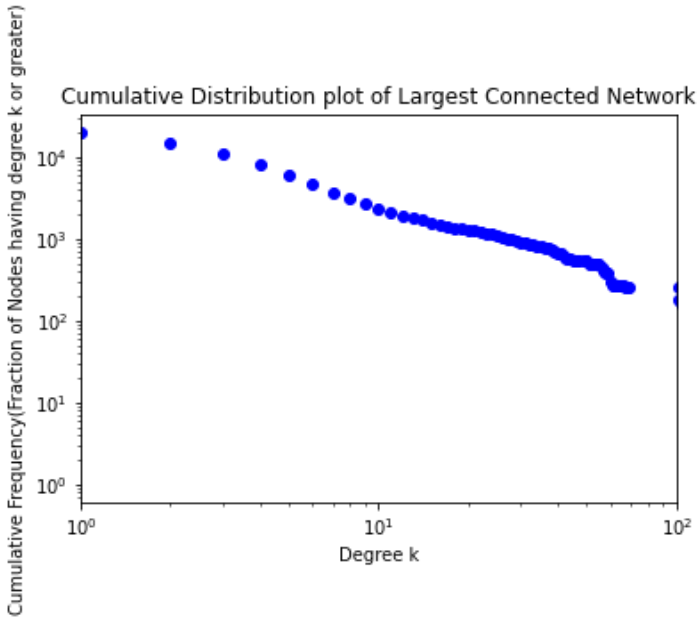
H) The largest connected network has 19,924 nodes. (55% of all of the nodes in the network)

I)



**Plotted up to degree 10^2

*This degree distirbution for the largest connected component of the network also appears to follow the log power distirbution law because the frequency and degree follow a linear relationship when plotted on a log-log scale.



J) The largest connected component of the network has 70,586 edges. (This represents ~40% of all of the edges in the network)

The edge density in the largest connected network is 0.0003556. (The edge density of the entire network is: 2.43e-05)

The diameter of the largest connected network is 48 and the average clustering coefficient is 0.478.

```
#Creating data frame of just the Offender ID and Crime Event Case Number
df_NoUnique_i = df[['NoUnique','SeqE']]
#Merging the above dataframe with a duplicate of itself to list all crime events and the involved offenders
df_ij = pd.merge(df_NoUnique_i, df_NoUnique_i, on='SeqE')
```

```
df_ij.shape
```

```
#Removing all rows where the offender ID number in both columns is the same
#Removing all rows where it is justa duplicate of another row with the same case number and offenders
df_ij_filtered = df_ij.loc[(df_ij['NoUnique_x'] < df_ij['NoUnique_y'])]
```

```
#Determining the size of this new filtered dataframe that consists of pairs of offenders and their common Crime Event case number
print(df_ij_filtered.shape)
#Counting how many unique crime event case numbers are included in this data set
print(df_ij_filtered.nunique())
#Counting how many unique offender IDs are included in this data set
ijcount = pd.unique(df_ij_filtered[['NoUnique_x', 'NoUnique_y']].values.ravel())
print(len(ijcount))
```

```
(216705, 3)
NoUnique_x    75308
SeqE          84038
NoUnique_y    74769
dtype: int64
121159
```

```
#Generate a column that includes the weights for the edges
#i.e. count how many crime event case numbers any given pair of offenders has in common
edgelist = df_ij_filtered.value_counts(subset=['NoUnique_x','NoUnique_y'])

# converting to df and assigning new names to the columns
edgelist_w = pd.DataFrame(edgelist)
edgelist_w = edgelist_w.reset_index()
edgelist_w.columns = ['NoUnique_x', 'NoUnique_y', 'weights'] # change column names


print(edgelist_w.head())
print(edgelist_w.shape)
```

```
   NoUnique_x  NoUnique_y  weights
0      253577      440431      356
1      614546      623487      204
2      303644      318895      106
3      207865      253979       95
4      170099      317918       80
(178413, 3)
```

```
GG = nx.from_pandas_edgelist(edgelist_w, source='NoUnique_x',target='NoUnique_y', edge_attr='weights')
```

```
nx.is_directed(GG)
```

```
False
```

#Code Determining largest connected component

```
GG.order()
```

```
GG.size()
```

```
nx.number_connected_components(GG)
```

```
largest_cc = max(nx.connected_components(GG), key=len)

Large_G = GG.subgraph(largest_cc).copy()
```

```
print('Number of nodes', Large_G.order())
```

K) As the municipality is an attribute of the crime event, I created a new network where the crime event is the node and the edges represent common offenders of two crime events.

Created an edge list with pairs of Crime Events, where the weights of the edges are the number of common co-offenders.

Then created a dictionary of crime events and their corresponding municipality.

This was added to the network graph as a node attribute.

Modularity was calculated based on similar municipality, Using Networkx modularity function.

Modularity = 0.95

This value indicates that crime events tend to be connected to crime events in the same municipalities. This geographically makes sense as certain crime events would tend to be clustered geographically. We would anticipate that the correlation between municipalities crime events would decrease with distance. There could also be associated similarities between municipalities with respect to population density, industries or other aspects which also contribute to similar crime events occurring in those cities.

```
#Municipality Homophily
#Will generate a network with the crime events as the nodes and the offenders as edges
#Will then assign the municipality as a node attribute of the crime event

#Creating edgelist of offenders just in the Municipality 66023
#Creating data frame of just the Offender ID and Crime Event Case Number
df_Mun_i = df[['NoUnique','SeqE']]
df_Mun_ij = pd.merge(df_Mun_i, df_Mun_i, on='NoUnique')
df_Mun_ij.head()
```

	NoUnique	SeqE_x	SeqE_y
0	1	1085034	1085034
1	2	1431379	1431379
2	4	167174	167174
3	5	1179096	1179096
4	17	1270690	1270690

```
#Removing all rows where the Crime Event Number in both columns is the same
#Removing all rows where it is just a duplicate of another row with the same case number and offenders
df_ij_filteredMUN = df_Mun_ij.loc[(df_Mun_ij['SeqE_x'] < df_Mun_ij['SeqE_y'])]

#Determining the size of this new filtered dataframe that consists of pairs of crime events
#and their common offenders
print(df_ij_filteredMUN.shape)
#Counting how many unique crime event case numbers and offenders are included in this data set
print(df_ij_filteredMUN.nunique())
#Counting how many unique crime events are included in this data set
ijcountMUN = pd.unique(df_ij_filteredMUN[['SeqE_x', 'SeqE_y']].values.ravel())
print(len(ijcountMUN))
#Generate a column that includes the weights for the edges
#i.e. count how many common offenders that any two pairs of crime events has in common
edgelistMUN = df_ij_filteredMUN.value_counts(subset=['SeqE_x', 'SeqE_y'])

# converting to df and assigning new names to the columns
edgelist_wMUN = pd.DataFrame(edgelistMUN)
edgelist_wMUN = edgelist_wMUN.reset_index()
edgelist_wMUN.columns = ['SeqE_x', 'SeqE_y', 'weights'] # change column names


print(edgelist_wMUN.head())
print(edgelist_wMUN.shape)
```

```
(5923161, 3)
NoUnique    184475
SeqE_x      687081
SeqE_y      690071
```

```
GG_mun = nx.from_pandas_edgelist(edgelist_wMUN, source='SeqE_x',target='SeqE_y', edge_attr='weights')
```

```
#Creating data frame of municipality and crime event (SeqE)
df_muni.drop(df_muni.columns[[0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14]], axis = 1, inplace = True)


df_muni.set_index(df_muni['SeqE'], inplace=True)
df_muni.drop(df_muni.columns[[0]], axis = 1, inplace = True)
print(df_muni.head())
#Converting dataframe into dictionary
dict_muni = df_muni.to_dict()
```

```
          MUN
SeqE
1085034  58227
1431379  94068
167174   49058
1179096  65005
1270690  23027
```

```
#Adding municipality as a node attribute to the network, the nodes are in this case Crime Events
nx.set_node_attributes(GG_mun, dict_muni, 'MUN')

largest_cc_mun = max(nx.connected_components(GG_mun), key=len)

Large_G_mun = GG_mun.subgraph(largest_cc_mun).copy()
print('Number of nodes', Large_G_mun.order())
```

```
Number of nodes 120603
```

```
nx_comm.modularity(Large_G_mun, nx_comm.label_propagation_communities(Large_G_mun))
```

```
0.951118897680136
```

L) Will determine modularity of the largest connected component based on the gender of the offenders.

If the offenders are both female or are both male then this will be counted as a similar node connection.

I will use modularity to quantify the number of similar connections in the largest connected component of the co-offender network.

Created a dictionary where the key is the Offender (SeqE) and then the corresponding gender listed as 0 for Male and 1 for Female. Then added this attribute to the existing Largest connected component as a node attribute.

- Utilized nx_comm.modularity function to calculate the modularity = 0.866
- This indicates a high degree of connectivity between similar node types (Male or Female) that exceeds random chance. For this network this indicates that females are more likely to co-offend with females and males are more likely to co-offend with males.

```
#Created function to convert genders to 0 and 1
def label_sex(df):
    if df['SEXE'] == 'F' :
          return 1
    elif df['SEXE'] == 'M' :
           return 0
```

```
#Creating data drame of gender assignment and Offender (NoUnique)
df_Gender = pd.read_csv(path)
df_Gender.drop(df_Gender.columns[[1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]], axis = 1, inplace = True)
#Applying function to convert M and F designation to 0 and 1
df_Gender['SEXE'] = df_Gender.apply(label_sex, axis=1)

df_Gender.set_index(df_Gender['NoUnique'], inplace=True)
df_Gender.drop(df_Gender.columns[[0]], axis = 1, inplace = True)
df_Gender.head()
#Converting dataframe into dictionary
dict_Gender = df_Gender.to_dict()
```

```
#Adding attributes to inital network
nx.set_node_attributes(GG, dict_Gender, 'gender')
#Designating new largest connected component network but with gender attributes
largest_cc_g = max(nx.connected_components(GG), key=len)

Large_G_gender = GG.subgraph(largest_cc_g).copy()
```

```
#Calculating modularity on this largest connected component with gender designated as the similarity determination
nx_comm.modularity(Large_G_gender, nx_comm.label_propagation_communities(Large_G_gender))
```

```
0.8661153086822764
```

M) Will look at the municipality with the most crimes over this time period (2003-2010). And identify which are the central offenders within that co-offender network.

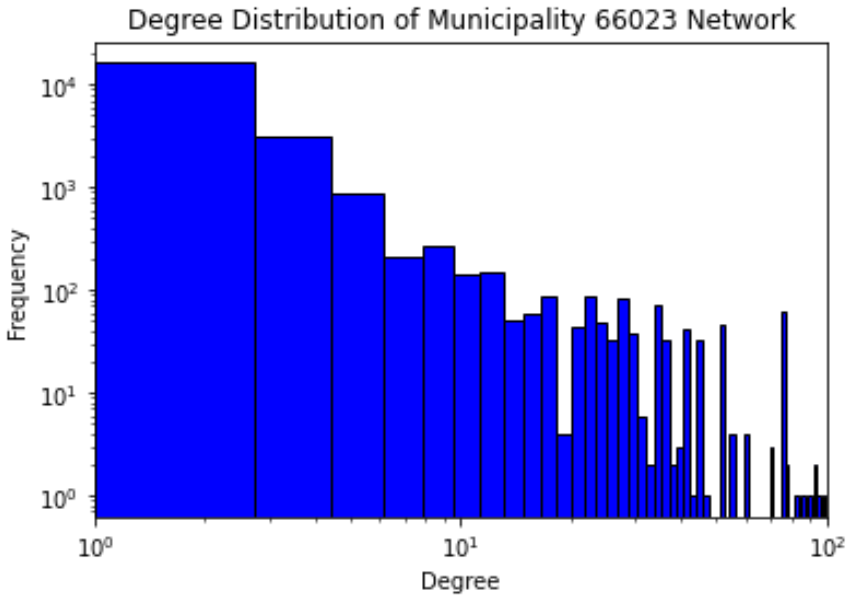
That is MUN # 66023 with a total of 330,376 crime events.

Created a co-offender network for only those crime events which occurred in this municipality.

Municipality 66023 co-offender network has 21,983 nodes (Offenders) and 47,654 edges(Crime event connections).

The edge density is: 0.000197

It consists of 7,100 connected components.



The largest connected component in the Municipality No. 66023 Co-offender network has 1,060 nodes (~4.8% of the total nodes in this municipality co-offender network) and 3,341 edges (~7% of the total edges in this municipality co-offender network). This could indicate that not one connected component dominates this municipality.

The diameter of the largest connected network is 32.

The edge density of the largest connected network is 0.00595. This density is much greater than the density in the entire network which was 0.000197.

Centrality:

Betweenness Centrality (Node which lies on the most paths between other ndoes, i.e. the node whose removal could most effectively break up the network)

The node with the largest betweenness centrality is offender with NoUnique # 597339. The betweenness centrality measurement is 0.341

This is a male born in 1988, who has been arrested for 35 offenses. (Crimes spanning 2004-2009)

Degree Centrality (The node with the greatest number of connections)

The node with the largest degree centrality is offender with NoUnique #538039. The degree centrality measurement is 0.0359.

This is a male born in 1985, who has been arrested for 8 offenses. (Crimes spanning 2006-2009)

Eigenvector Centrality (Identifies the node with the most connections to other nodes who are also well connected.)

The node with the largest degree centrality is offender with NoUnique #537522. The degree centrality measurement is 0.187.

This is a male born in 1985, who has been arrested for 11 offenses (Crimes spanning 2005-2009)

In Municipality #66023, the average birth year of offenders is 1973. The offenders identified with highest centrality are all slightly younger than the average offender in the network.

The three offenders that were identified as having different measurements of high centrality were arrested for their first recorded crimes at the ages of 16, 21 and 20. This could indicate something about the demographics of offenders in this municipality.

```
#Create dataframe of just municipality 66023
df_MUN_scenario = pd.read_csv(path)
df_MUN_scenario.drop_duplicates(keep = 'first', inplace = True)
#Create dataframe of just municipality 66023

df_MUN66023 = df_MUN_scenario.loc[df['MUN'] == 66023]

print(df_MUN66023['Naissance'].max())
print('Average year of birth in the offender network of Municipality 66023',df_MUN66023['Naissance'].mean())
df_MUN66023.sort_values('Naissance', ascending=False)
```

```
#Creating edgelist of offenders just in the Municipality 66023
#Creating data frame of just the Offender ID and Crime Event Case Number
df_NoUnique_i_MUN = df_MUN66023[['NoUnique','SeqE']]
df_ij_MUN = pd.merge(df_NoUnique_i_MUN, df_NoUnique_i_MUN, on='SeqE')
#Removing all rows where the offender ID number in both columns is the same
#Removing all rows where it is justa duplicate of another row with the same case number and offenders
df_ij_filteredMUN = df_ij_MUN.loc[(df_ij_MUN['NoUnique_x'] < df_ij_MUN['NoUnique_y'])]

#Determining the size of this new filtered dataframe that consists of pairs of offenders and their
#common Crime Event case number
print(df_ij_filteredMUN.shape)
#Counting how many unique crime event case numbers are included in this data set
print(df_ij_filteredMUN.nunique())
#Counting how many unique offender IDs are included in this data set
ijcountMUN = pd.unique(df_ij_filteredMUN[['NoUnique_x', 'NoUnique_y']].values.ravel())
print(len(ijcountMUN))
#Generate a column that includes the weights for the edges
#i.e. count how many crime event case numbers any given pair of offenders has in common
edgelistMUN = df_ij_filteredMUN.value_counts(subset=['NoUnique_x','NoUnique_y'])
```

```
# converting to df and assigning new names to the columns
edgelist_wMUN = pd.DataFrame(edgelistMUN)
edgelist_wMUN = edgelist_wMUN.reset_index()
edgelist_wMUN.columns = ['NoUnique_x', 'NoUnique_y', 'weights'] # change column names


print(edgelist_wMUN.head())
print(edgelist_wMUN.shape)
```

```
#Creating network graph of co-offender network only in Municipality 66023
GG_MUN66023 = nx.from_pandas_edgelist(edgelist_wMUN, source='NoUnique_x',target='NoUnique_y', edge_attr='weights')
#Number of connected components in the graph
nx.number_connected_components(GG_MUN66023)
#Determining number of nodes and edges
print('Number of nodes', GG_MUN66023.order())
print('Number of edges', GG_MUN66023.size())

densityMUN = nx.density(GG_MUN66023)
print('The edge density is: ' + str(densityMUN))
```

```
degree_sequenceMUN = sorted([d for n, d in GG_MUN66023.degree()], reverse=True)
dmax = max(degree_sequenceMUN)
plt.hist((degree_sequenceMUN), bins=100,edgecolor="black", color="blue")
plt.xscale('log')
plt.yscale('log')
plt.ylabel('Frequency')
plt.xlabel('Degree')
plt.xlim(10**0, 10**2)
plt.title('Degree Distribution of Municipality 66023 Network')
```

```
#Largest connected component of the Municipality 66023 graph
MUN_comp = sorted(nx.connected_components(GG_MUN66023), key=len, reverse=True)
G0 = GG_MUN66023.subgraph(MUN_comp[0])

len(MUN_comp)
print('Number of nodes in largest connected component of MUN 66023', G0.order())
print('Number of edges in largest connected component of MUN 66023', G0.size())
#Diameter of the largest connected component of Mun. 66023
nx.algorithms.distance_measures.diameter(G0)
#Edge density
densityMUN_
0 = nx.density(G0)
print('The edge density is: ' + str(densityMUN_0))
#Calculating Centrality Measurements of the largest connect component in the Municipality  # 66023

eig_cen = nx.eigenvector_centrality(G0)
btwn_cen = nx.betweenness_centrality(G0)
deg_centrality = nx.degree_centrality(G0)
```

```
#Determining the node (Offender) with the greatest centrality measurement
max_btwn = max(btwn_cen, key = btwn_cen.get)
print('Node with the largest betweenness centrality is', max_btwn)
max_deg_centrality = max(deg_centrality, key = deg_centrality.get)
print('Node with the largest Degree centrality is', max_deg_centrality)
max_eig_cen = max(eig_cen, key = eig_cen.get)
print('Node with the largest Eigenvector centrality is', max_eig_cen)

#Getting the centrality measurement of the node with the greatest centrality
print(btwn_cen.get(597339))
print(deg_centrality.get(max_deg_centrality))
print(eig_cen.get(max_eig_cen))
```