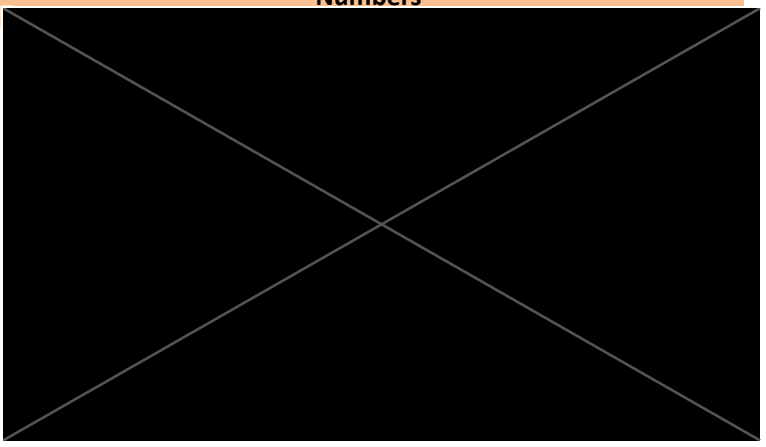
# Social Networks – Literature Review & Analysis of DataSet

## MA214 – Network Analysis Group Assignment

**Done By:** ***Network Ninjas***

| # | Names | E-mails | Registration Numbers | PRIDs |
|---|---|---|---|---|
| 1 | Muhammad Ashfaq Tahir | | | |
| 2 | Mahesh Kodepati | | | |
| 3 | Rohan Jadhav | | | |
| 4 | Aanandhan Nachiyar Manimagalai Murugan | | | |
| 5 | Roja Annamalai Rajanbabu | | | |



# Contents

# Abstract

Social network analysis is now recognised as a potent instrument for exploring the complex network of connections that influence our social environment. This report discusses the literature review as well as the analytical part of a chosen dataset in the field of social networks. The aim is to research multiple papers related to the field of study and use the inferences derived to do the analysis on the chosen dataset. This report consists of a dataset about the trolls collected on Twitter by the Internet research agency to find their influence in politics. We use various methodologies to analyse the data set, such as size, degree, density, centrality, modularity, and other related factors and aim to check if there is any external community trying to influence the 2016 US presidential elections. Post analysis, we found that there is a high probability of Russian interference on US elections.

***Keywords**: Social Network Analysis, Russian Troll, Degree, Density, Centrality, Modularity, Community*

# 1. Introduction

Social network analysis examines the patterns of connections between individuals or social entities, like organisations, as 'points' and the connections between behaviours or attitudes that are influenced by these social relationships as lines (1,2). In simpler terms, social network analysis takes the concept that interactions build a network of connections and turns it into a formal representation to model social relationship structures. Viewing a social structure as a network is the key idea behind social network analysis (2). Fig 1 shows various examples in our daily life.

Social networks offer a rich source of data for analysis due to their inherent structure of connections between individuals or entities. By studying social networks, one can gain insights into patterns of communication, influence, information flow, and behaviour. Understanding these dynamics can be valuable in various fields such as sociology, psychology, anthropology, business, and public health. Social network analysis allows for the identification of key influencers, communities, and structural features within networks, enabling researchers to explore complex social phenomena and make informed decisions. Therefore, we choose social networks for analysis purposes and provide a powerful framework for studying and understanding social interactions and relationships.

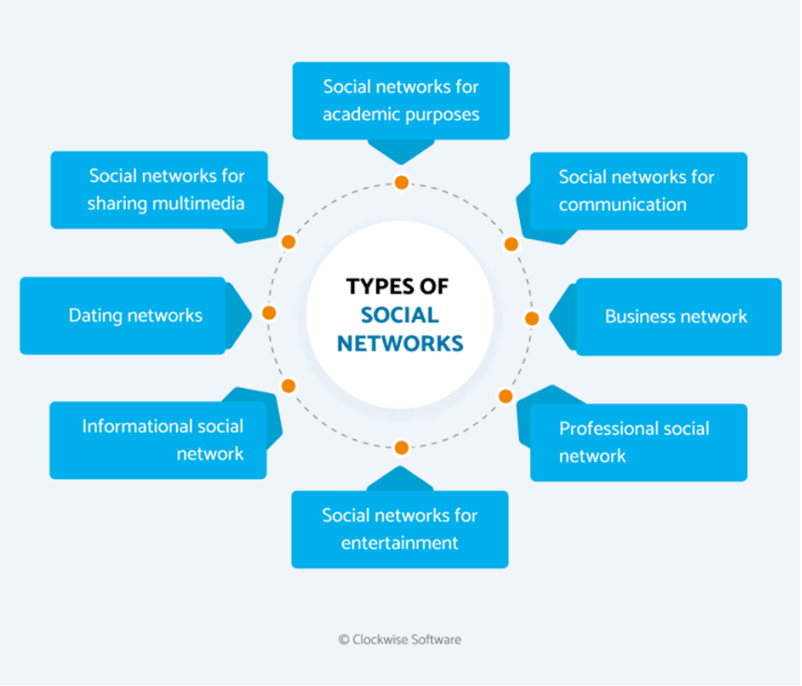



***Fig. 1 - Types of Social Networks (3)***

Several social science disciplines, especially anthropology and sociology, have long engaged in social network analyses (4). The interest in network analysis has grown among statisticians due to advancements in computing power, enabling the development of new models and methods for analysing networks. This increased interest is driven by the understanding that networks play a crucial role in various research areas involving individuals, communities, policy domains, workplaces, and schools, leading to a broader application of social network analysis (1).

Two main types of network models are commonly used: individual-level models and relational-level models. Individual-level models focus on outcomes at the individual level, using network data to define explanatory variables. On the other hand, relational-level models analyse the relationships between individuals in a network, treating it as a multivariate dependent variable with individual linkages as its elements. Relational analyses consider network structure by looking at network statistics related to relational properties and covariates like characteristics of units within the network (1). Social network analysis uses three main mathematical approaches that have been widely influential in the field: graph theory, algebraic approaches, and spatial approaches (7).

# 2. Literature Review

James O'Malley and Peter V. Marsden analysed the healthcare data to find the physician influence on social networks in primary care practice The article explores the progression of social network analysis from social science to statistics, emphasising the advancement of sophisticated models and techniques for examining physician influence networks. It delves into fundamental network statistics, descriptive measures, and two categories of statistical models: individual-outcome models and relational models. The challenges in estimating these models stem from complex correlation patterns among outcome measures. Utilising social network analysis effectively in health services and outcomes research will not only provide fresh perspectives on these phenomena but also aid in advancing social network methodology further. They concluded that the changes in networks and figuring out ways to pick samples of networks haven't been researched a lot yet. (1)

Sancheng Peng, Yongmei Zhou, Lihong Cao, Shui Yu d, Jianwei Niu e, and Weijia Jia used a survey to find Influence analysis on social networks. The analysis of influence plays a vital role in supporting the practical use of social networks and has garnered notable interest in recent times. This survey focuses on the theoretical and practical obstacles encountered by researchers in this evolving domain. Its goal is to establish a thorough groundwork by showcasing the most recent studies. The survey encompasses fundamental aspects of social networks, their varieties, social influence analysis across various levels, assessment criteria, existing models, strategies for maximising influence, existing algorithmic issues, and upcoming trends within this field. (5) Pietro A. Bianchi et al., used social network analysis (SNA) to explore how people, teams, and organisations interact with each other from patterns like social networks, and if these patterns can explain important business outcomes. They reviewed 162 articles from 2000 to 2021 in accounting and finance, summarising key elements and organising findings. The authors propose a more systematic way of studying networks (SNA) in accounting and finance. This new approach would involve: 1. Building the network based on the research question to make sure the measures used are accurate; 2. Examining the properties and structure of the network to confirm the validity of the theory being tested; 3. Considering and discussing the influence of other existing networks that might be affecting the results. By following these steps, the authors believe that SNA can be used more effectively to understand how networks influence organisations and individuals. This will not only improve our understanding of current network effects, but also lead to new research questions that can advance the field. (6)

Xiangjie Kong, Yajie Shi, Shuo Yu, Jiaying Liu and Feng Xiav found that within the rapidly expanding realm of scholarly big data, social network technologies have recently garnered significant interest in both

academic and industrial sectors. Academic social networks, situated in the realm of scholarly big data, encompass complex academic connections between entities and their interactions. Various methodologies are employed to process extensive scholarly data for analysing the diverse structures and associated information within these networks. The abundance of academic data available today simplifies the analysis and exploration of these networks. This study explores the origins, current state, and future directions of academic social networks, addressing the concept itself, models based on node characteristics and timeliness, analytical approaches encompassing metrics and network attributes, tools for academic analysis, essential mining technologies for academic social networks, and a comprehensive examination of research tasks at actor, relationship, and network levels. While dealing with limited data presents a challenge, extensive datasets can be enriched by leveraging the interconnections within the data. Academic data platforms are typically tailored to specific subjects, primarily in Computer Science, which hinders interdisciplinary research. Assessing the impact of research continues to be a persistent difficulty for scholars. (7)

According to Madelaine Castles, Robert Heinsohn, Harry H. Marshall, Alexander E.G. Lee, Guy Cowlishaw, and Alecia J. Carter the utilisation of social network analysis in animal populations provides valuable insights into group dynamics and individual social roles, aiding in the comprehension of the advantages and disadvantages of social behaviours and evolutionary strategies within societies. A research study compared social networks constructed using diverse interaction and proximity methods in two troops of chacma baboons over a three-year period. Findings revealed notable disparities between networks formed through different techniques, underscoring the caution needed when comparing studies employing varied sampling approaches. While interaction techniques may be suitable for comparison on a broader scale, proximity methods were not, emphasising the importance of careful evaluation when using proximity as a proxy for social interactions in social network studies. The study also stressed the significance of considering temporal changes and the influence of sampling methods when examining social networks. (8)

## 2.1 Literature review - Conclusion

Social network analysis (SNA) has emerged as a powerful tool for examining the intricate web of connections that shape our social world. SNA has gained traction in statistics due to advancements in computing power and has fuelled the development of innovative models and methods for analysing networks across various domains, from healthcare to business. The literatures used various methods for SNA such as, Size, Density, Degree and Distribution, Paths and Geodesic Distance, dyad and triad census, reciprocity, Transitivity, Closure, Centrality, Cliques, Components, Clusters, Modularity, which will be referred for the dataset analysis along with the lecture notes.

The studies reviewed here highlight the need for further exploration of network dynamics, sampling methods, and the influence of external factors. By following a structured approach, like the one proposed by Bianchi et al. [6], SNA can unlock a deeper understanding of how connections shape our world. This will not only improve our knowledge of existing network effects but also open doors for groundbreaking research questions, driving the field's advancement.

# 3. Network Analysis

## 3.1 Dataset Description

Selected Network – "***Russian Troll Twitter Mention Network***" (11)

**Background**

In February 2018, the U.S. Justice Department charged 13 Russian citizens with interfering in the 2016 U.S. Presidential election (Barrett, Horwitz, & Helderman, 2018). The indictment implicated the Internet Research Agency (IRA), headquartered in St. Petersburg, in a Russian campaign, commencing in 2014, aimed at fostering unrest within the U.S. political framework, predominantly through social media channels. Instances of IRA-associated profiles were discovered on various platforms such as Facebook, Twitter, Instagram, and Google. These profiles impersonated American individuals and were utilised to fragment voters across a spectrum of issues, affecting the whole campaign and in turn affecting the elections. (9)

The chosen dataset meets the set criteria and is a shortened version of the dataset from the publication named 'Troll Factories: The Internet Research Agency and State-Sponsored Agenda Building'. The said report was meant to document the methods employed by Internet Research Agency (IRA) to influence the political agenda of the United States from June 19, 2015 to December 31, 2017.

**Selection Criteria for the Dataset**

For the network analysis part, we had devised the below criteria by which this dataset was selected.

1. Social Network and solve a real-world issue
2. Possesses the characteristics of a graph network
3. Has the scope to apply as many concepts of network analysis learned through the course

**Overview of Dataset**

This data is a collection of Twitter mentions as a network model in the period between 19 June 2015 to 31 December 2017. Each account handler corresponds to a node and directed edge is a mention of 'node a' by 'node b' in a tweet, or a reply or a retweet.

| | |
|---|---|
| **Edges represent:** | Mentions |
| **Nodes represent:** | Twitter Accounts |
| **Number of Edges:** | 2974 |
| **Number of Nodes:** | 1245 |
| **Graph/network type:** | Directed |
| **Name of the file:** | Trolls |
| **File Type:** | R data |
| **Format:** | Edgelist |

***Table 1: Overview of Dataset***

This report was used as our reference for this review. Our report contains the same basic network structure, however, we limited ourselves to 'Mel' data. This data contains tweets downloaded from handles identified by Twitter as linked to the IRA.

**Dataset information: Column description**

1. **Names:** names of account holders
2. **Activity:** unique identifiers of the account holders
3. **Account type and account category:**

| Account type | Frequency by account type | Account category | Frequency by account category |
|---|---|---|---|
| Russian | 43 | NonEnglish | 44 |
| Right | 22 | RightTroll | 22 |
| Hashtager | 18 | HashtagGamer | 18 |
| news | 7 | NewsFeed | 14 |
| local | 7 | LeftTroll | 7 |
| left | 7 | Unknown | 1 |

***Table 2: Account type and Account Category***

4. **region: the accounts have posted the tweets from 22 geographic locations including the following countries:** US, UAE, Azerbaijan, Russian Federation, Germany, Belarus, Iraq, UK, Japan, Samoa, Malaysia, Greece, Islamic republic of Israel, Saudi Arabia, Ukraine, Afghanistan, France, Canada, Egypt, Austria. 593 of the 1245 tweets were sent out by accounts exclusively located in the USA.
5. **language:** It has been difficult to get the exact list of languages which requires the application of text analytics. However, they were noted to be from multiple languages, the vast majority of which were non-english. Arabic, Bulgarian, Russian, Macedonian, Serbian were some of them to name a few.
6. **maxfollowing:** maximum number of fellow tweeters each of these account holders had followed in this duration of study
7. **minfollowing:** Minimum number of fellow tweeters each of these account holders had followed in this duration of the study.
8. **maxfollowers:** Maximum number of followers for each of these account holders in this study duration
9. **minfollowers:** min number of followers for each of these account holders in this study duration
10. **maxpostdate:** date of the latest tweet posted by each of these accounts during this period
11. **minpostdate:** date of the earliest tweet posted by each of these accounts during this period

## 3.2 Analysis & Interpretation

As discussed in the previous section, the dataset consists of troll data. This dataset consists of around 1245 nodes, with directed edges. We will be using the python framework to analyse the network and will be using libraries such as:

- Networkx
- networkx.algorithms.community
- rdata
- matplotlib
- pandas
- Counter, etc.

Let us first analyse and view the data using networkx library, Fig 2 shows the network formed when we extract the edge list from the data.

*Refer appendix for code

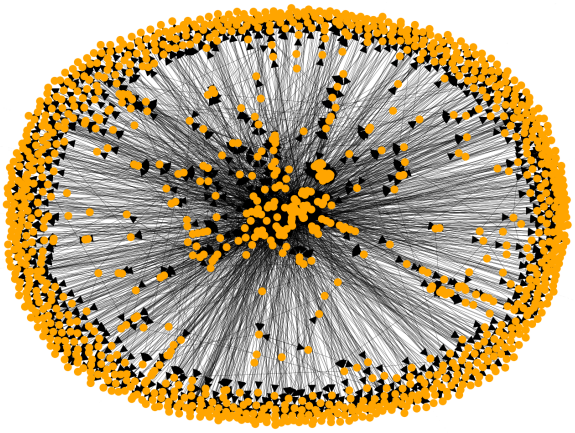


***Fig.2 - Network of Twitter Russian troll mentions (2015-2017)***

For analysis purposes and to find the communities in the network, we have converted directed graph into undirected graph.

Fig 3 highlights the top ten nodes with maximum degree in the complete network. These nodes for the maximum edges which correspond to the most active trolls. They are equally distributed amongst all the different communities/clusters and act as the leaders. These are the users with the maximum number of mentions and are the most influential. These influential nodes are well connected amongst themselves within the communities and the connections are highlighted with red edges.

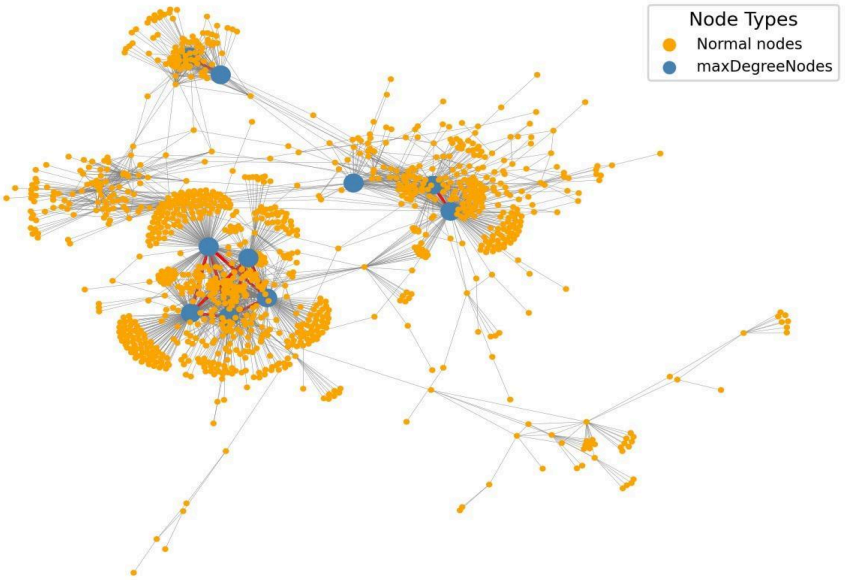



***Fig.3 - Maximum Degree Nodes (top 10)***

Another aspect we can look into is the edge betweenness centrality. Fig 4 highlights the top 10 edges with the highest edge betweenness centrality in red colour. Majority of these edges originate or terminate from the most influential nodes. And these edges demonstrate the role which influential nodes are playing in starting a conversation or a trend. Although there are no direct edges with high betweenness centrality amongst the influential nodes of different communities, they either lie on the same paths or they influence other nodes as a bridge between different communities.

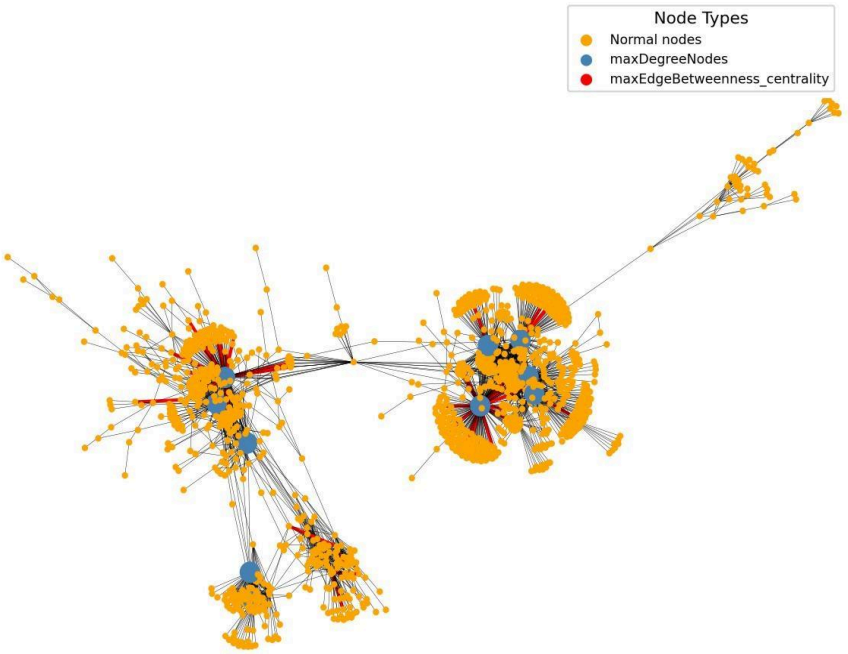



***Fig. 4 - Maximum Edge Betweenness Centrality (top 10 edges are highlighted)***

If we look at the network, we can see that there are groups of users who mention other Twitter accounts and share information. So, That means there can be formation of communities or clusters in the network. As seen in literature review, there are various methods such as, degree, closeness, centrality, etc, But we cannot just directly apply these methods on the whole network, as it won't give us any meaningful insights. The strategy here is to find communities formed in the network and analyse them individually.

Let us use greedy search algorithms based on modularity using networks (greedy_modularity_communities). We will be analysing only the top 10 communities for this research.

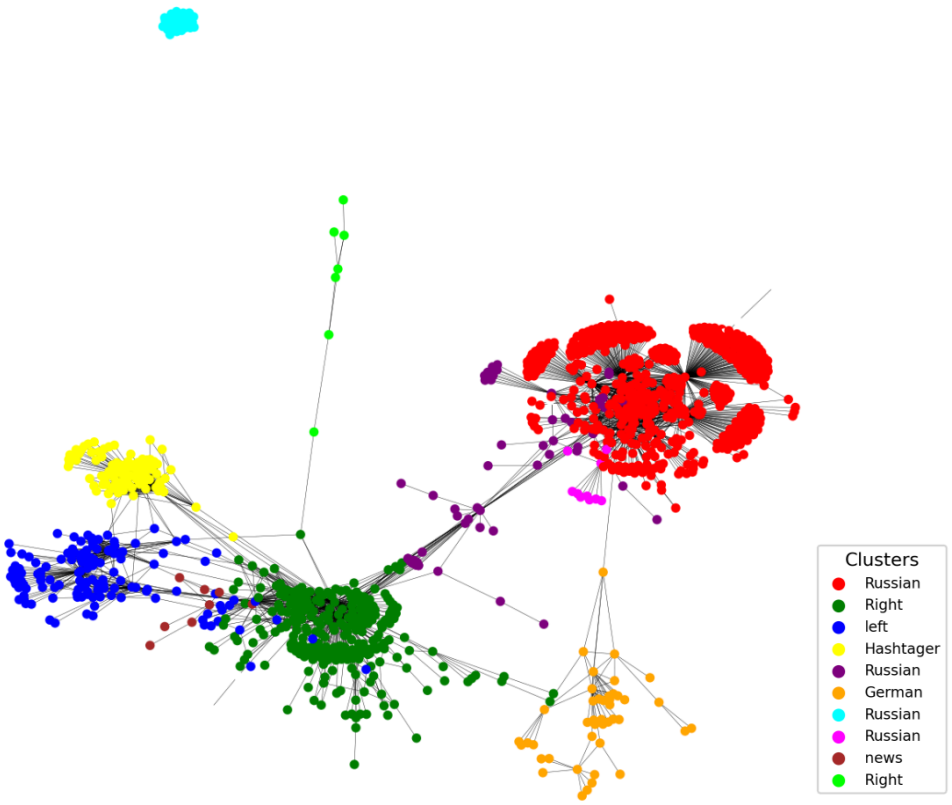



***Fig. 5 - Communities discovered by using greedy modularity algorithm***

Now that we have the communities figured out, Let's analyse the community with highest modularity, as it may be one of the influential communities. Fig 6 below is the graphical representation of the community with highest modularity. When the user data of nodes were analysed it was found that the nodes belonged to people from Russia.

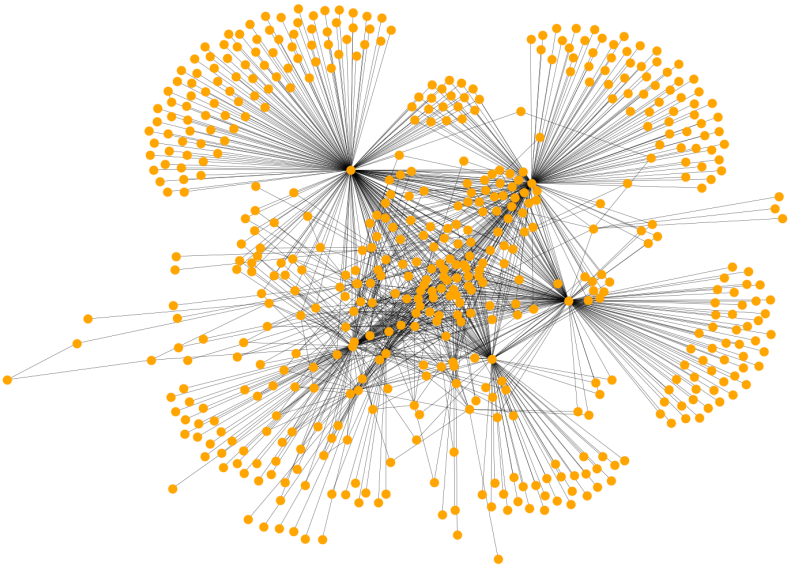


***Fig. 6 - Community with maximum modularity (Russian)***

Below table 3, summarises some of the properties of the Russian trolls community. As evident from the table, this community represents a significant part of the complete network.

| **Property** | **Python function/code** | **Value** |
|---|---|---|
| Number of nodes | len(list(P1.nodes)) | 553 |
| Number of edges | len(list(P1.edges)) | 1122 |
| Density | nx.density(P1) | 0.0073 |
| Diameter | nx.diameter(P1) | 5 |
| Is the partition connected | nx.is_connected(P1) | True |
| Mean Degree | sum(dict(degree).values())/len(list(P1.nods)) | 4.0578 |

***Table 3: Properties of the partition with highest modularity***

Fig. 7 demonstrates the maximum degree nodes, maximum closeness centrality and maximum betweenness centrality amongst the Russian trolls community (community with the highest modularity). All three techniques point to more or less the same group of users. These users are the most active amongst the population, meaning they are the ones with the most mentions or tweets.

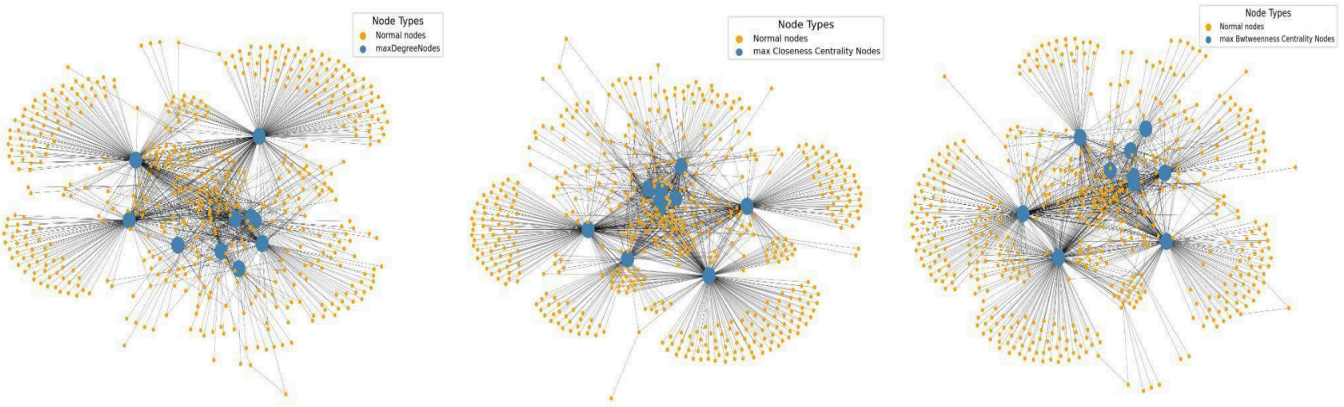



***Fig. 7 - Maximum degree centrality, closeness centrality and betweenness centrality***

Now that we have got the influential nodes, let us analyse k cores of the partition. In Fig 8 we see the nodes who form the highest k core, which signifies the highest connected nodes. Comparing Fig 7 and Fig 8 we see that most of the highlighted nodes among both graphs are common. This outcome might offer insight into the collaborative nature among the influential nodes.

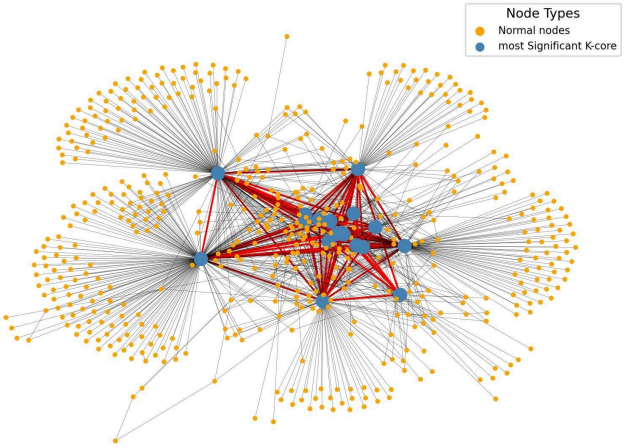



***Fig. 8 - Most significant K-core***

We have now analysed the highest modularity community, But there are other communities as well in this network. Following table shows the amount of nodes/accounts present in each partition. Let us analyse how nodes are distributed among partitions.

Below table 4 shows the results. Other than the Russian community, the Right wing and left-wing communities have a significant number of nodes. The presence of these communities is normal as they belong to the USA. There are around 1245 nodes in the network. Looking at remaining communities we observe that the number of nodes is very less and is the reason we will not be investigating them (Although the smaller communities may have some influence, we will limit ourselves due to the scope of this report).

| Community category | Nodes |
|---|---|
| Russian | 523 |
| Right | 267 |
| Left | 128 |
| Hashtager | 92 |
| Russian | 56 |
| German | 43 |
| Russian | 35 |
| Russian | 10 |
| news | 8 |
| Right | 7 |

***Table 4 - Node distribution among communities***

**Time series analysis:**

In this analysis we will observe how the network grows and try to get some insight from this data.

Here we will be filtering out nodes based on the minimum post date. Once we have the nodes, we will get the corresponding edge list and create the graph, and will use it to visualise to get some key insights.

From Fig. 9, we can see that 75% of the accounts started posting (were created) before 2015-07-15, and based on these results the graphs are plotted.

```
userData['minDate'].describe()
```

```
count                             1245
mean     2015-07-15 06:50:36.144578304
min                2012-02-02 00:00:00
25%                2015-05-17 00:00:00
50%                2015-06-22 00:00:00
75%                2015-07-15 00:00:00
max                2017-12-08 00:00:00
Name: minDate, dtype: object
```

***Fig. 9 - Distribution of minimum post date***

From Fig 10 , We can see the growth of the network from 2015 to 2017. One key point to note here is: US presidential elections were held in 2016. Following are the insights drawn from the visualisation:

- There are some pre-existing communities in subplot: 2015-05-17
- Communities are beginning to engage with one another based on subplot: 2015-06-22
- Based on subplots: 2015-07-15, 2017-12-08, we see that there is no more formation of any new communities, but the density of the network is increasing.
- The rise in density, indicating an increase in communication, began occurring as the US elections came nearer, around 2016.

From previous discussion, we see that the community with highest modularity was ‘Russian’, So there is some possibility of them influencing elections.

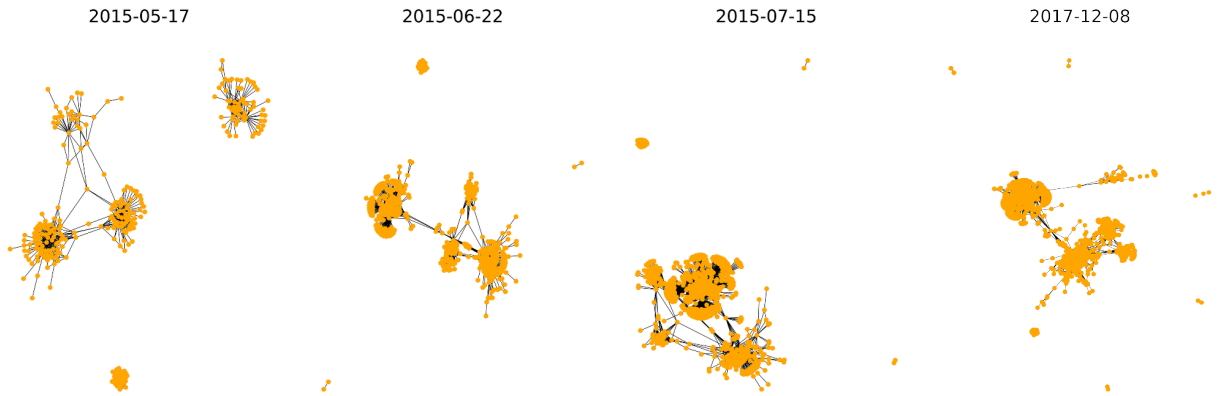



***Fig. 10 - Network growth from 2015 - 2017***

## 3.3 Conclusion

Social network analysis takes the concept that interactions build a network of connections and turns it into a formal representation to model social relationship structures. Analysing the data and the network with the help of centrality, modularity, time series, and other techniques gave us a lot of important information on the Russian internet research agency and their influence on US politics.

Given the network is large, we have conducted analysis on each community individually. We have sorted the communities based on modularity and picked the top 10 communities. Russian being the largest community is picked up for further analysis. In the Russian community the influential nodes are found using maximum degree centrality, closeness centrality and betweenness centrality. When comparing these nodes to the highest k-core in the community, we found out that most of the nodes are common. That leads us to conclude that these nodes/twitter accounts are working together in groups to further spread their agenda during US elections. The influential nodes are highlighted in Fig 7 and Fig 8.

We can also confirm with the help of time series analysis that the growth of these networks and trolls started in 2015 and became stronger and denser in 2016, as the elections were planned for November 8, 2016. The Russian community, which gained the highest modularity, aims to distract, divide, and demoralise with the help of Twitter, which affects the US elections.

Many authors also conclude that the research emphasises the significance of group norms and regulations in developing effective mechanisms to prevent trolling online. Simply identifying and banning trolls is insufficient; actively promoting group norms like "don't feed the troll," especially among well-connected members, is crucial for safeguarding online community well-being. (10)

## 3.4 Future Research

To assess the dissemination and its effects, future studies can also incorporate tweets from languages other than English. In this study post partition, only the Russian community is analysed.

Additional communities within the United States, such as those on the right and left wings, play integral roles in discussions during US elections.. However, smaller communities like those of German or gamer

have not been thoroughly examined due to their relatively limited influence, indicated by their smaller network nodes (30-50 compared to the top communities' 200-500). Nevertheless, it's important to recognize that these communities could potentially exert influence and with other techniques can be checked out for further studies.

Furthermore, another issue lies in the time-consuming nature of studying tweets. By the time any action is proposed, the information has already been widely disseminated. Therefore, developing efficient automatic classification tools to categorise tweets and grasp their content would empower us to swiftly intervene and restrict the spread.

# 4. Bibliography


1. O'Malley AJ, Marsden PV. The analysis of Social Networks. Health Services and Outcomes Research Methodology. 2008 Oct 21;8(4):222–69. doi:10.1007/s10742-008-0041-z
2. Scott J. Social network analysis. Thousand Oaks, CA: Sage; 2017.
3. Popovych A. How to build a social network website from scratch [Internet]. 2023 [cited 2024 Mar 7]. Available from: https://clockwise.software/blog/how-to-build-a-social-network-website/
4. Freeman LC. The development of social network analysis a study in the sociology of Science. Vancouver, BC: Empirical Press; 2011.
5. Peng S, Zhou Y, Cao L, Yu S, Niu J, Jia W. Influence Analysis in social networks: A survey. Journal of Network and Computer Applications. 2018 Mar;106:17–32. doi:10.1016/j.jnca.2018.01.005
6. Bianchi PA, Causholli M, Minutti-Meza M, Sulcaj V. Social Networks Analysis in Accounting and Finance*. Contemporary Accounting Research. 2022 Dec 21;40(1):577–623. doi:10.1111/1911-3846.12826
7. Kong X, Shi Y, Yu S, Liu J, Xia F. Academic Social Networks: Modelling, analysis, mining and applications. Journal of Network and Computer Applications. 2019 Apr;132:86–103. doi:10.1016/j.jnca.2019.01.029
8. Castles M, Heinsohn R, Marshall HH, Lee AEG, Cowlishaw G, Carter AJ. . Animal Behaviour. 2014 Oct;96:59–67. doi:10.1016/j.anbehav.2014.07.023
9. Linvill D, Warren P [Internet]. The Social Media Listen Center, Clemson University; [cited 2024 Mar 12]. Available from: https://pwarren.people.clemson.edu/Linvill_Warren_TrollFactory.pdf
10. Sun Q, Shen C. Who would respond to a troll? A social network analysis of reactions to trolls in online communities. Computers in Human Behavior. 2021 Aug;121:106786. doi:10.1016/j.chb.2021.106786
11. Marcum CS. Russian troll Twitter mention network [Internet]. [cited 2024 Mar 12]. Available from: https://github.com/cmarcum/RussianTrollNetwork/blob/master/README.md
12. Tutorial# [Internet]. [cited 2024 Mar 13]. Available from: https://networkx.org/documentation/latest/tutorial.html
13. Matplotlib cheatsheets and handouts# [Internet]. [cited 2024 Mar 14]. Available from: https://matplotlib.org/cheatsheets/
14. Collections - container datatypes [Internet]. [cited 2024 Mar 14]. Available from: https://docs.python.org/3/library/collections.html

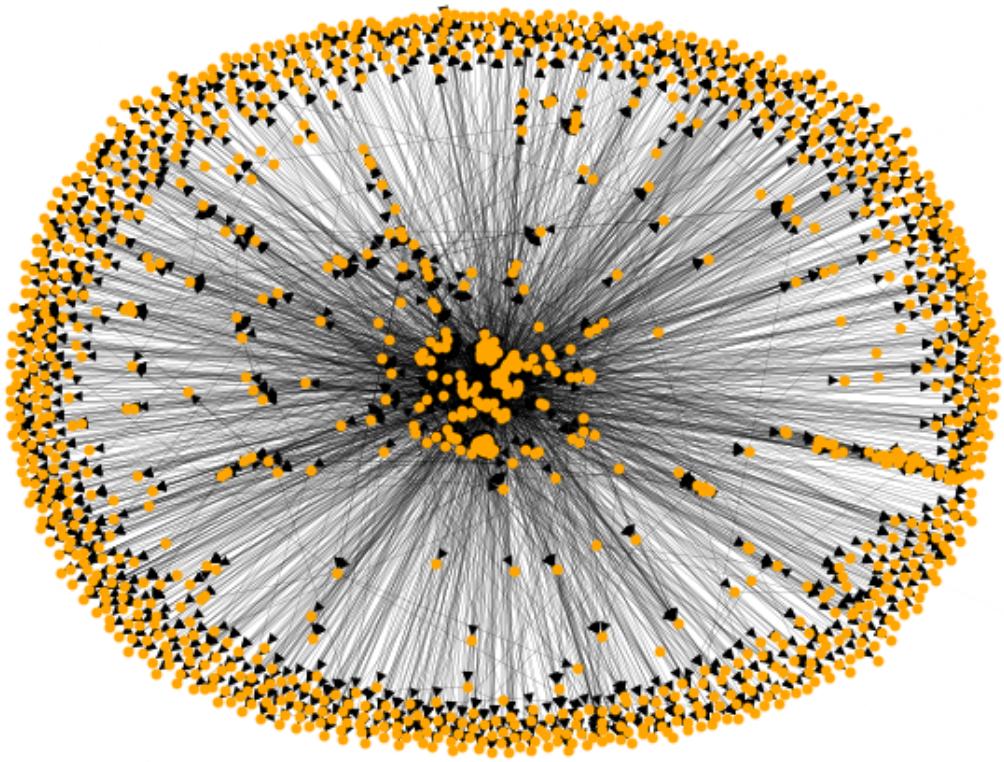

# Appendix

In [1]:
```
#Import libraries
import rdata
import os
import networkx as nx
import pandas as pd
import networkx.algorithms.community as nx_comm
from pyvis.network import Network
import numpy as np
import matplotlib.pyplot as plt
import random
from datetime import datetime
from collections import Counter

# Enable multiple outputs of a cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

In [2]:
```
#read data and extract edgeList and twitter account data.
#Please Map the location of Dataset present in your system to parse.
result = rdata.parser.parse_file("Datasets/trolls.Rdata")
data = rdata.conversion.convert(result)

mel = data['trolls']['mel']
userData = pd.DataFrame(data['trolls']['val'])

edgeList = []
for edge in mel:
    p1 = edge['inl'][0]
    p2 = edge['outl'][0]
    edgeList.append([p1,p2])
```

```
/Users/rohanjadhav/anaconda3/lib/python3.11/site-packages/rdata/conversio
n/_conversion.py:887: UserWarning: Missing constructor for R class "netwo
rk". The underlying R object is returned instead.
  warnings.warn(
```

In [3]:
```
#Plot the whole graph using edgeList. Adjust node color & size, edge widt
G = nx.DiGraph()
G.add_edges_from(edgeList)
nx.draw(G, with_labels=False, node_size=10, width=0.1, node_color='orange
```



In [4]:
```
#Finiding top 10 partitions/communities who mention amoung themselves on
partition = list(nx_comm.greedy_modularity_communities(G))[0:10]

#Note: The partitions except top 10 are very small, which contain 2-10 no
#50-600 nodes, so they are ignored for study here.
```

In [5]:
```
#Get the degree of nodes, sort them from highest degree to lowest degree.
degree_list = list(G.degree())
sorted_degree_list = sorted(degree_list, key=lambda x: x[1], reverse=True
maxDegreeNodes = [node[0] for node in sorted_degree_list]

#From user data pick the category of the top 10 partitions for analysis.
def getpartitionUserCategory(part):
    count = 0
    top_10_nodes_from_part = []
    for node in part:
        # Check if the node is in maxDegreeNodes
        if node in maxDegreeNodes:
            top_10_nodes_from_part.append(node-1)
            count += 1
        # If we have found the top 10 nodes, break the loop
        if count == 10:
            break
    return userData.iloc[top_10_nodes_from_part]
```

In [6]:
```
#Here we are creating a dictionary which maps the partition to correspond
categoryList = []
for i in range(0,10):
    categoryList.append(getpartitionUserCategory(partition[i])['accountty

part = []
for p in partition:
    part.append(len(p))

partitionCategoryMap = {}
for i in range(0,10):
    partitionCategoryMap[part[i]] = categoryList[i][0]
partitionCategoryMap
```

Out[6]:
```
{523: 'Russian',
 267: 'Right',
 128: 'left',
 92: 'Hashtager',
 56: 'Russian',
 43: 'German',
 35: 'Russian',
 10: 'Russian',
 8: 'news',
 7: 'Right'}
```

In [7]:
```
#Since we have edgeList which has all the edges of the network. For analy
#edgelist, so this method returns edgelist of partitions based on their i
#Basically it gets edges for nodes present in parition.
def GetEdgeListForPartition(n):
    partitionEdgeList = []
    for node in partition[n]:
        for edge in edgeList:
            if((edge[0]==node) or (edge[1]==node)):
                partitionEdgeList.append(edge)
    return partitionEdgeList
```

In [8]:
```
#For plotting purpose few distinct colors are chones and each color here
colList = ['red', 'green', 'blue', 'yellow', 'purple', 'orange', 'cyan',
partitionDict = {}

count=0
for partNodes in partition[0:10]:
    partitionDict[colList[count]] = partNodes
    count = count+1

#This method is defined to return the color of the node which will be use
def get_key_by_value(my_dict, target_value):
    for key, value in my_dict.items():
        for n in value:
            if n == target_value:
                return key
    return 'white'  # Return White if the value is not found in the dicti
```

In [9]:
```
partitionNodes = []
for part in partition:
    for node in part:
        partitionNodes.append(node)

partitionedEdgeList = edgeList

#since we are plotting only top 10 nodes, we will be removing edges from
for edge in partitionedEdgeList:
    if edge[0] not in partitionNodes and edge[1] not in partitionNodes:
        partitionedEdgeList.remove(edge)

#Create graph with top 10 partitions to plot.
G_parts = nx.Graph()
G_parts.add_edges_from(partitionedEdgeList)

#Create list of node colors based on paritions.
nodeColor = []
for i in G_parts.nodes():
    nodeColor.append(get_key_by_value(partitionDict, i))
```

In [10]:
```
#Plotting the graph with partitions denated by different colors.
legend_dict = {}
color_index = 0
fig, axes = plt.subplots(1, 1, figsize=(10, 8))

labelsDict = {}
for i in range(0,10):
    labelsDict[colList[i]] = categoryList[i][0]

for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)

nx.draw(G_parts, with_labels=False, node_color=nodeColor, font_size=8, no
#Adding legend of Partition categories.
plt.legend(title='Clusters', bbox_to_anchor=(1.05, 1), loc='upper left',
plt.show()
```

Out[10]: <matplotlib.collections.PathCollection at 0x11dcd0e10>

Out[10]: <matplotlib.collections.PathCollection at 0x1233d81d0>

Out[10]: <matplotlib.collections.PathCollection at 0x120b8f9d0>

Out[10]: <matplotlib.collections.PathCollection at 0x1233d3c50>

Out[10]: <matplotlib.collections.PathCollection at 0x1233d9a90>

Out[10]: <matplotlib.collections.PathCollection at 0x11db1e150>

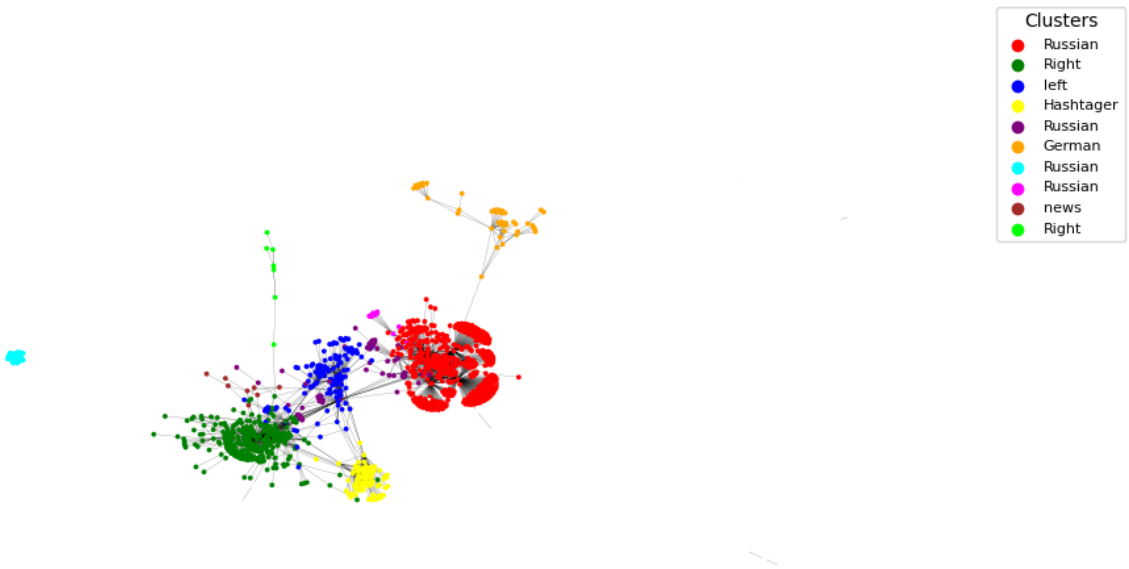
Out[10]: <matplotlib.collections.PathCollection at 0x123416ed0>

Out[10]: <matplotlib.collections.PathCollection at 0x11da64e50>

Out[10]: <matplotlib.collections.PathCollection at 0x1234151d0>

Out[10]: <matplotlib.collections.PathCollection at 0x1233f8c90>

Out[10]: <matplotlib.legend.Legend at 0x12339e410>



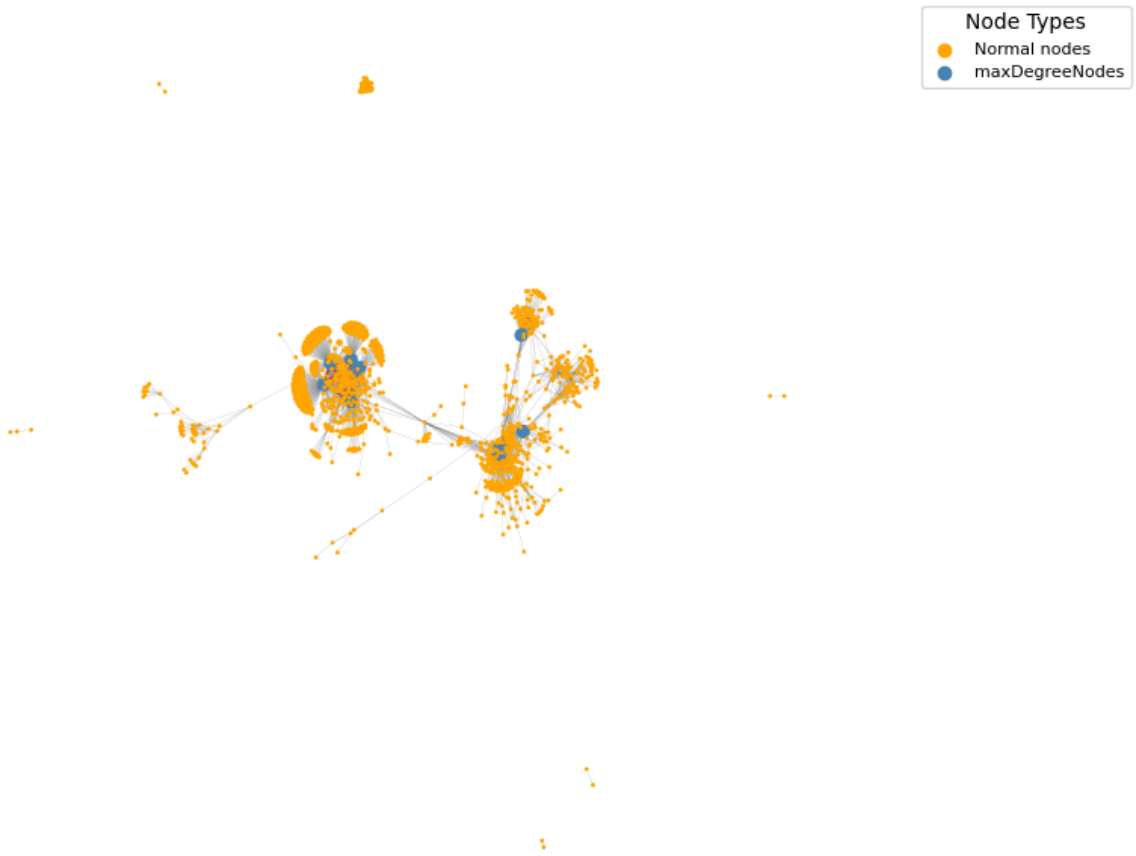

In [11]:
```
# Maxdegree nodes in complete network

degree_list = list(G_parts.degree()) #Degree of all the nodes
sorted_degree_list = sorted(degree_list, key=lambda x: x[1], reverse=True
maxDegreeNodes = [node[0] for node in sorted_degree_list] #Taking the nod

md_node_color = ['steelblue' if i in maxDegreeNodes else 'orange' for i i
md_node_size = [30 if i in maxDegreeNodes else 1 for i in G_parts.nodes()
highlight_edges = [(u, v) for u, v in G_parts.edges() if u in maxDegreeNo
md_edge_colors = ['red' if edge in highlight_edges else 'grey' for edge i
md_edge_width = [1 if edge in highlight_edges else 0.1 for edge in G_part
```

In [12]:
```
# Plotting the graph
fig, axes = plt.subplots(1, 1, figsize=(8, 8))
labelsDict = {'orange':'Normal nodes', 'steelblue':'maxDegreeNodes'}

# Create a scatter plot with fake data to display the legend
for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)

nx.draw(G_parts, with_labels=False, node_color=md_node_color, node_size=m
plt.legend(title='Node Types', bbox_to_anchor=(1.05,1), loc='upper left',
plt.savefig('maxDegreeNodes.png', dpi=300, bbox_inches='tight')
```

Out[12]: <matplotlib.collections.PathCollection at 0x1234cdf90>

Out[12]: <matplotlib.collections.PathCollection at 0x1234cef50>

Out[12]: <matplotlib.legend.Legend at 0x12354c910>




In [13]:
```
#Max edge_betweenness_centrality in complete network

#Getting the top 10 edges with highest edge betweenness centrality
edgeBetweenness_centrality = sorted(nx.edge_betweenness_centrality(G_part

node_list = list(G_parts.nodes())
#setting the color for edges with highest edge betweenness centrality
eb_edge_colors = ['red' if any(node == elem[0] or node == elem[1] for ele
#setting the width for edges with highest edge betweenness centrality
eb_edge_width = [1 if any(node == elem[0] or node == elem[1] for elem in
```

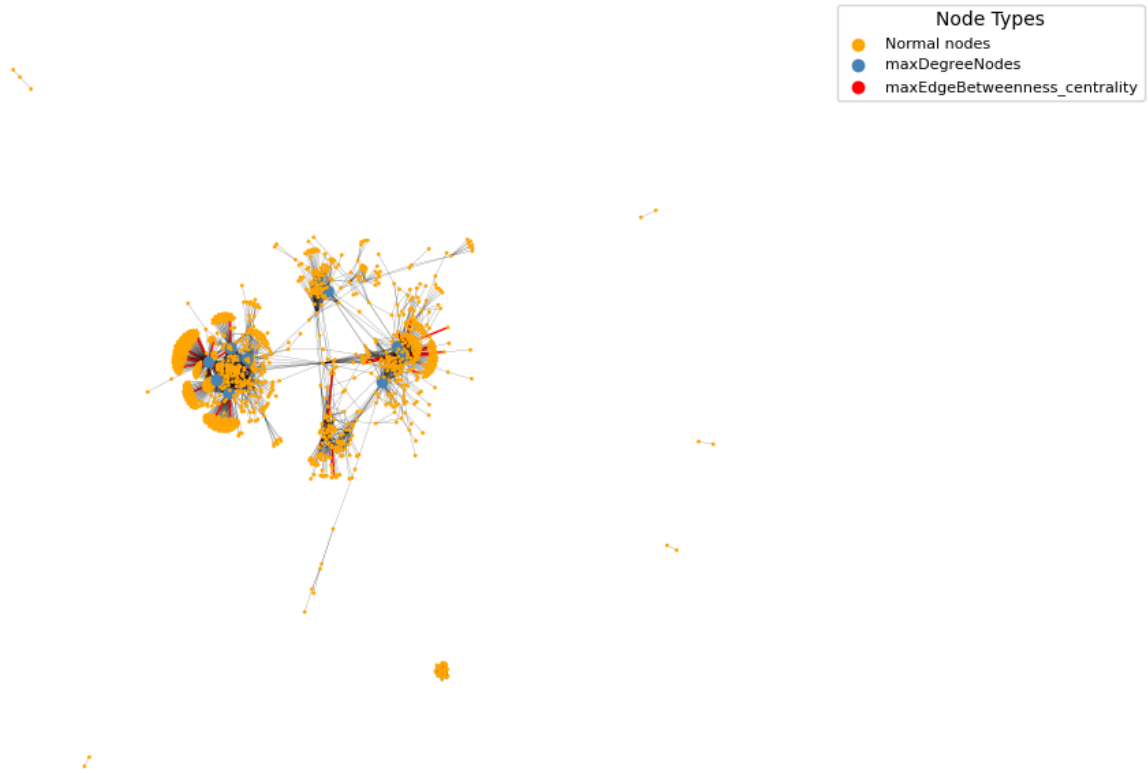
In [14]:
```
fig, axes = plt.subplots(1, 1, figsize=(8, 8))
labelsDict = {'orange':'Normal nodes', 'steelblue':'maxDegreeNodes','red'

# Create a scatter plot with fake data to display the legend
for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)
nx.draw(G_parts, with_labels=False, node_color=md_node_color, node_size=m
plt.legend(title='Node Types', bbox_to_anchor=(1.05,1), loc='upper left',
plt.savefig('edgeBetweenness_centrality.png', dpi=300, bbox_inches='tight
```

Out[14]: `<matplotlib.collections.PathCollection at 0x12344dd90>`

Out[14]: `<matplotlib.collections.PathCollection at 0x12344c0d0>`

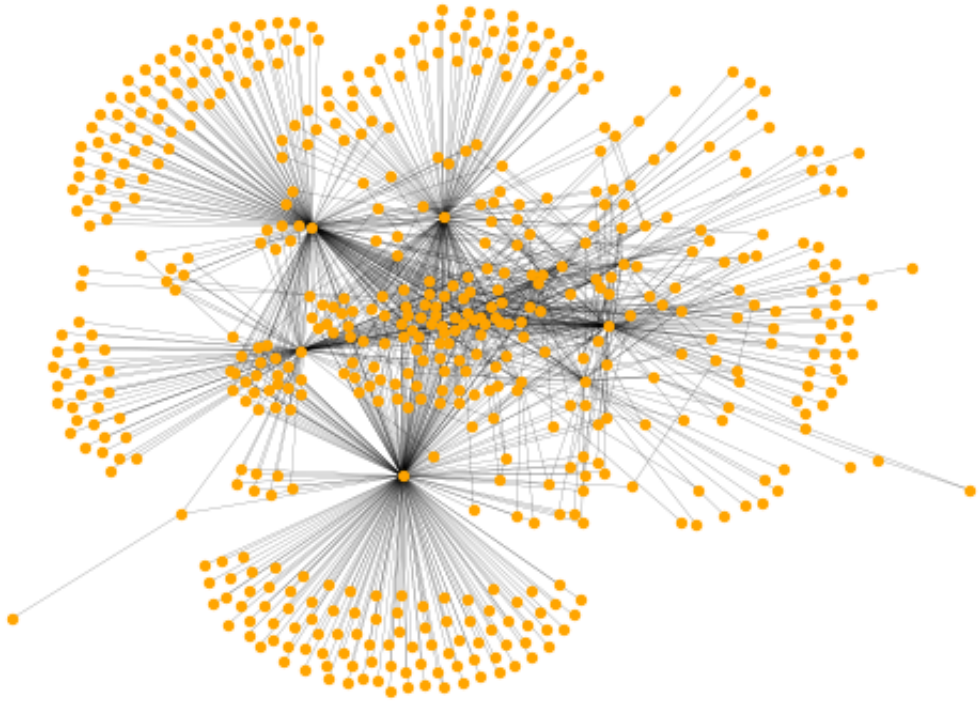
Out[14]: `<matplotlib.collections.PathCollection at 0x123555b50>`

Out[14]: <matplotlib.legend.Legend at 0x120ab2a90>




In [15]:
```
#Most significant partition P1 (highest modularity)

P1_edges = GetEdgeListForPartition(0) #partition with the highest modular
P1 = nx.Graph()
P1.add_edges_from(P1_edges)
```

In [16]:
```
#Visualising most significant partition P1
fig, axes = plt.subplots(1, 1, figsize=(7, 5)) #setting the figure size
nx.draw(P1, with_labels=False, node_size=8,node_color='orange', width=0.1
plt.savefig('MaxModularity_Partition.png', dpi=300, bbox_inches='tight')
```



In [17]:
```
#Some mathematical properties of P1

print("Number of nodes in P1 (partition with highest modularity)", len(li
print("Number of edges in P1 (partition with highest modularity)", len(li
print("Density of P1 is:", nx.density(P1)) #density of P1
print("Diameter of P1 is",nx.diameter(P1)) #Diameter of P1
print("Is the P1 connected:",nx.is_connected(P1)) #Checking if P1 is conn
degree = P1.degree() #Node degree of P1
P1_mean_degree = sum(dict(degree).values())/len(list(P1.nodes)) #Mean deg
print("Mean Degree of P1 is",P1_mean_degree)
```

```
Number of nodes in P1 (partition with highest modularity) 553
Number of edges in P1 (partition with highest modularity) 1122
Density of P1 is: 0.007351206855884897
Diameter of P1 is 5
Is the P1 connected: True
Mean Degree of P1 is 4.057866184448463
```

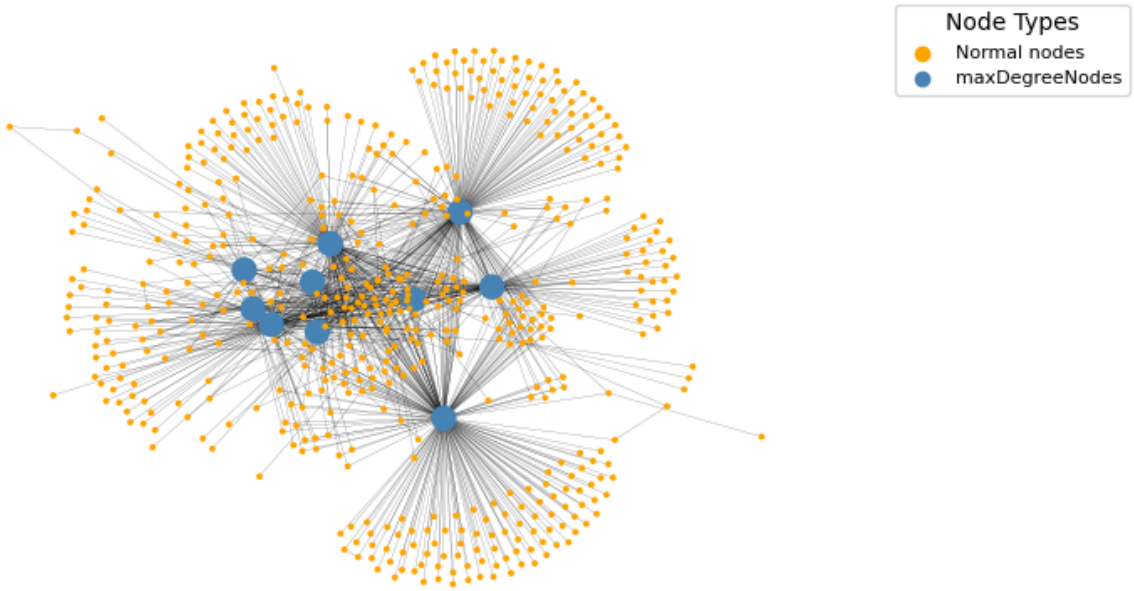
In [18]:
```
# Max degree nodes in P1

P1_max_degree_list = list(nx.degree(P1)) #Degree of nodes in P1
P1_sorted_degree_list = sorted(P1_max_degree_list, key=lambda x: x[1], re
P1_maxDegreeNodes = [node[0] for node in P1_sorted_degree_list] #Node lis

P1_node_color = ['steelblue' if i in P1_maxDegreeNodes else 'orange' for
P1_node_size = [100 if i in P1_maxDegreeNodes else 3 for i in P1.nodes()]
```

In [19]:
```
fig, axes = plt.subplots(1, 1, figsize=(7, 5)) #setting the figure size
labelsDict = {'orange':'Normal nodes', 'steelblue':'maxDegreeNodes'} #cre

# Create a scatter plot with fake data to display the legend
for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)
nx.draw(P1, with_labels=False, node_color=P1_node_color, node_size=P1_nod
plt.legend(title='Node Types', bbox_to_anchor=(1.05,1), loc='upper left',
plt.savefig('MaxDegeeNodes_P1.png', dpi=300, bbox_inches='tight')
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x12066ae90>
```

Out[19]:
```
<matplotlib.collections.PathCollection at 0x1233c9310>
```

Out[19]:
```
<matplotlib.legend.Legend at 0x1234a7ad0>
```



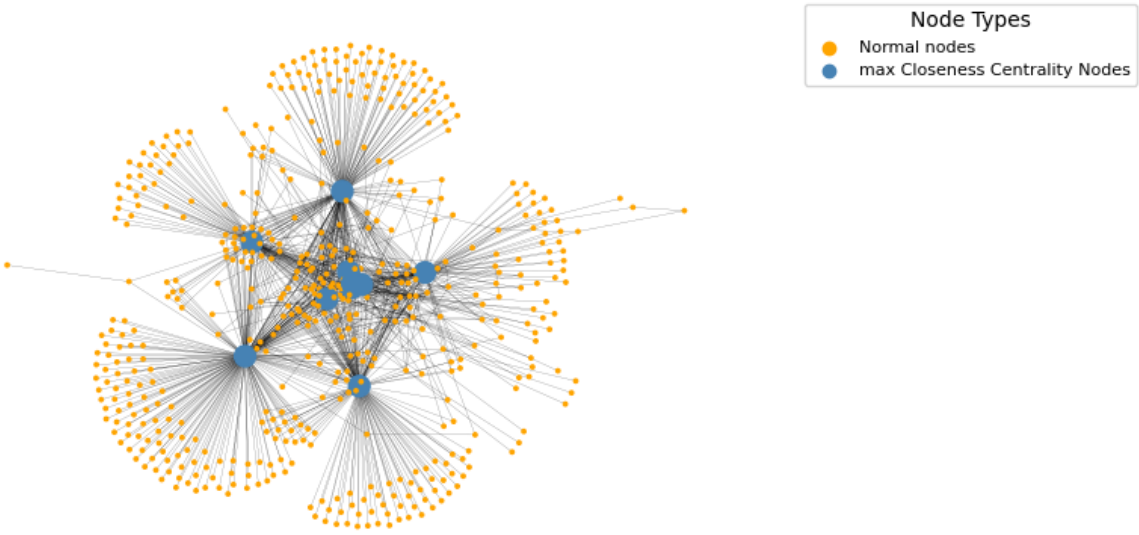

In [20]:
```
#Closeness Centrality of P1
P1_closeness_centrality = nx.closeness_centrality(P1) #Closeness centrali
P1_sorted_closeness_centrality = dict(sorted(P1_closeness_centrality.item
P1_maxClosenessNodes = list(P1_sorted_closeness_centrality.keys()) #Node

P1_cnode_color = ['steelblue' if i in P1_maxClosenessNodes else 'orange'
P1_cnode_size = [100 if i in P1_maxClosenessNodes else 3 for i in P1.node
```

In [21]:
```
fig, axes = plt.subplots(1, 1, figsize=(7, 5)) #setting the figure size
labelsDict = {'orange':'Normal nodes', 'steelblue':'max Closeness Central

# Create a scatter plot with fake data to display the legend
for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)
nx.draw(P1, with_labels=False, node_color=P1_cnode_color, node_size=P1_cn
plt.legend(title='Node Types', bbox_to_anchor=(1.05,1), loc='upper left',
plt.savefig('MaxClosenessCentralityNodes_P1.png', dpi=300, bbox_inches='t
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0x1223a57d0>
```

Out[21]: <matplotlib.collections.PathCollection at 0x1208fb050>

Out[21]: <matplotlib.legend.Legend at 0x1205d0150>



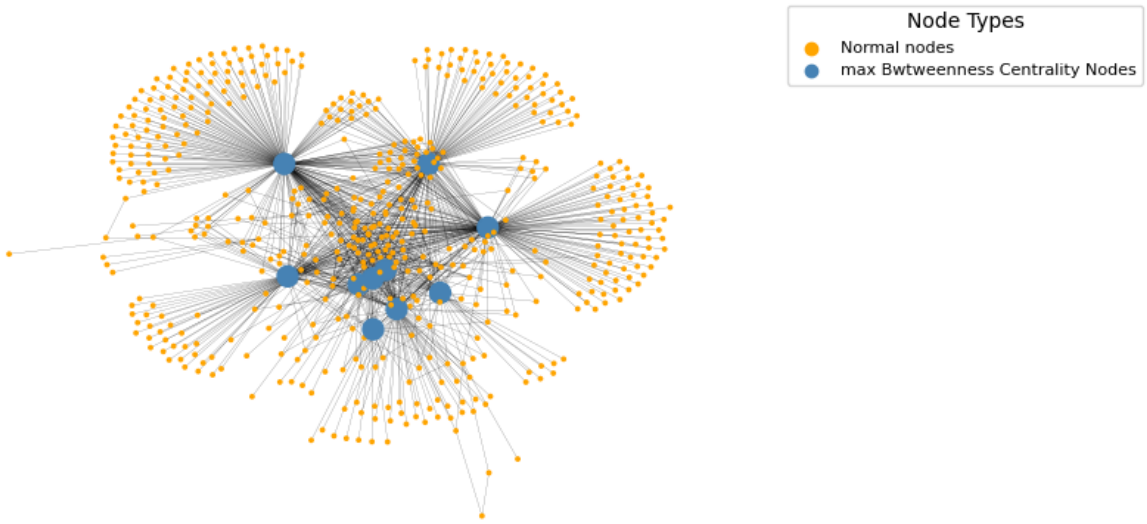

In [22]:
```
#Betweenness Centrality of P1

P1_betweenness_centrality = nx.betweenness_centrality(P1) #Betweenness ce
P1_sorted_betweenness_centrality = dict(sorted(P1_betweenness_centrality.
P1_maxBetweennessNodes = list(P1_sorted_betweenness_centrality.keys()) #N

P1_bnode_color = ['steelblue' if i in P1_maxBetweennessNodes else 'orange
P1_bnode_size = [100 if i in P1_maxBetweennessNodes else 3 for i in P1.no
```

In [23]:
```
fig, axes = plt.subplots(1, 1, figsize=(7, 5)) #setting the figure size
labelsDict = {'orange':'Normal nodes', 'steelblue':'max Bwtweenness Centr

# Create a scatter plot with fake data to display the legend
for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)
nx.draw(P1, with_labels=False, node_color=P1_bnode_color, node_size=P1_bn
plt.legend(title='Node Types', bbox_to_anchor=(1.05,1), loc='upper left',
plt.savefig('MaxBwtweennessCentrality_Nodes_P1.png', dpi=300, bbox_inches
```

Out[23]: <matplotlib.collections.PathCollection at 0x1204805d0>

Out[23]: <matplotlib.collections.PathCollection at 0x120481350>

Out[23]: <matplotlib.legend.Legend at 0x1208afad0>



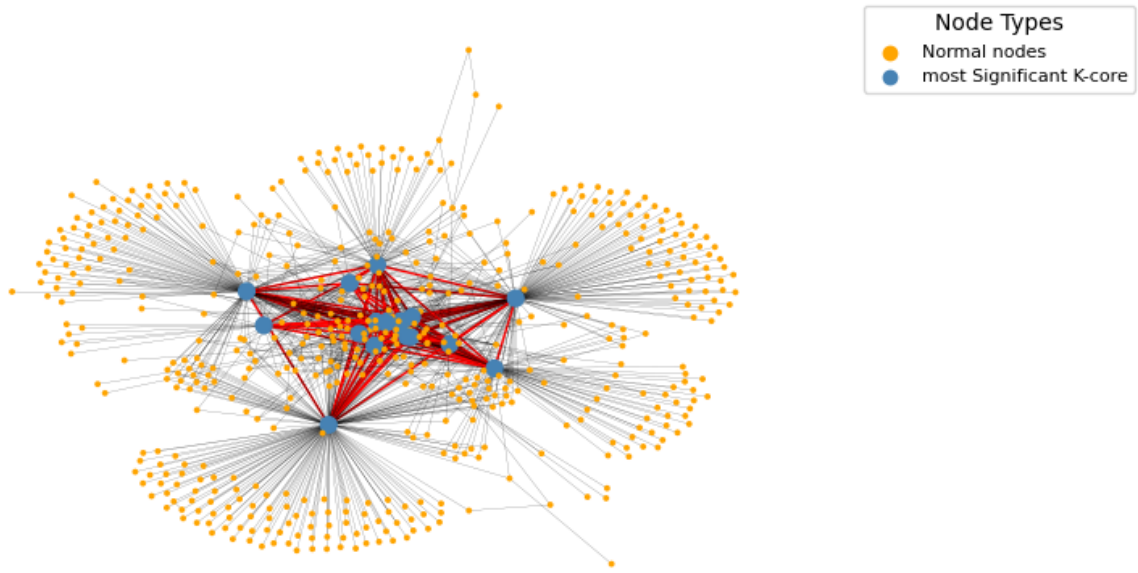

In [24]:
```
#Most significant K-core of P1

P1_kcore = list(nx.k_core(P1)) #most significant k-core

P1_knode_color = ['steelblue' if i in P1_kcore else 'orange' for i in P1.
P1_knode_size = [50 if i in P1_kcore else 3 for i in P1.nodes()] #setting

P1_highlight_edges = [(u, v) for u, v in P1.edges() if u in P1_kcore and
P1_edge_colors = ['red' if edge in P1_highlight_edges else 'black' for ed
P1_edge_width = [1 if edge in P1_highlight_edges else 0.1 for edge in P1.
```

In [25]:
```
fig, axes = plt.subplots(1, 1, figsize=(7, 5)) #setting the figure size
labelsDict = {'orange':'Normal nodes', 'steelblue':'most Significant K-co

# Create a scatter plot with fake data to display the legend
for color, category in labelsDict.items():
    plt.scatter([], [], color=color, label=category)
nx.draw(P1, with_labels=False, node_color=P1_knode_color, font_size=8, no
plt.legend(title='Node Types', bbox_to_anchor=(1.05,1), loc='upper left',
plt.savefig('most Significant K-core_P1.png', dpi=300, bbox_inches='tight
```

Out[25]:
```
<matplotlib.collections.PathCollection at 0x122759850>
```

Out[25]:
```
<matplotlib.collections.PathCollection at 0x12275a590>
```

Out[25]:
```
<matplotlib.legend.Legend at 0x1208f8150>
```




In [26]:
```
#Preprocess the date data. We are looking for range of 2015 to 2017.
#Example of data from data: [2015-12-28 18:16:00], We do not need exact t
#extract the date for analysis purpose.
mindate = []
for i in userData['minpostdate']:
    mindate.append(pd.to_datetime(str(i).replace("[", "").replace("\'", "
```

In [27]:
```
#Analyse the date column.
userData['minDate'] = mindate
userData['minDate'].describe()
```

Out[27]:
```
count                            1245
mean     2015-07-15 06:50:36.144578304
min                2012-02-02 00:00:00
25%                2015-05-17 00:00:00
50%                2015-06-22 00:00:00
75%                2015-07-15 00:00:00
max                2017-12-08 00:00:00
Name: minDate, dtype: object
```

In [28]:
```
#Since we have 4 quartiles, we will be plotting 4 graphs to check based o
#Here we are creating the Graph based on the filtered node based on min p
def GetGraphsBasedOnMinPostDate(date, data, edgeListToFilter, dateCategor
    threshold_date = pd.to_datetime(date)
    index_list = data[data[dateCategory] < threshold_date].index.tolist()
    node_list = [x + 1 for x in index_list]
    filtered_edge_list = [edge for edge in edgeListToFilter if edge[0] in
    G = nx.Graph()
    G.add_edges_from(filtered_edge_list)
    return G
```

In [29]:
```
#Create graphs
G1 = GetGraphsBasedOnMinPostDate('2015-05-17', userData, edgeList, 'minDa
G2 = GetGraphsBasedOnMinPostDate('2015-06-22', userData, edgeList, 'minDa
G3 = GetGraphsBasedOnMinPostDate('2015-07-15', userData, edgeList, 'minDa
G4 = GetGraphsBasedOnMinPostDate('2017-12-08', userData, edgeList, 'minDa
```

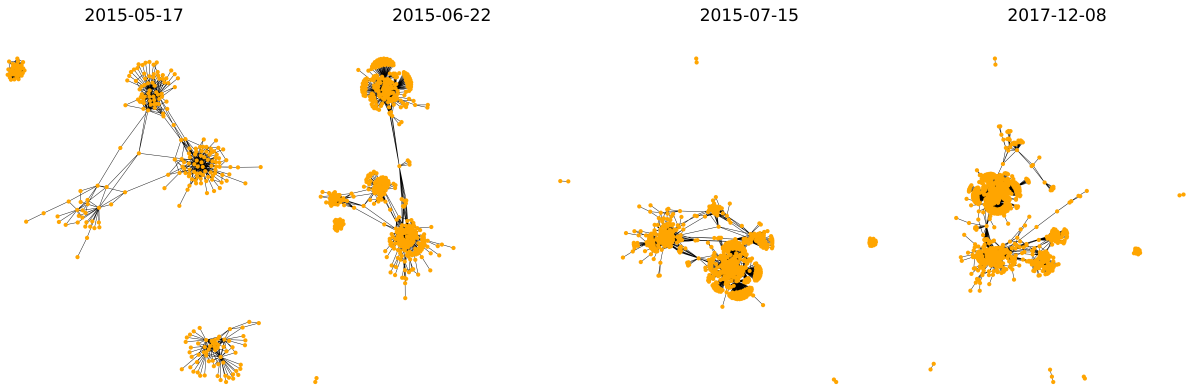
In [30]:
```
#plot graphs
fig, axes = plt.subplots(nrows=1, ncols=4, figsize=(120, 40))

nx.draw(G1, ax=axes[0], node_size=800, width = 3.0, node_color='orange')
nx.draw(G2, ax=axes[1], node_size=800, width = 3.0, node_color='orange')
nx.draw(G3, ax=axes[2], node_size=800, width = 3.0, node_color='orange')
nx.draw(G4, ax=axes[3], node_size=800, width = 3.0, node_color='orange')

axes[0].set_title('2015-05-17', fontsize=120)
axes[1].set_title('2015-06-22', fontsize=120)
axes[2].set_title('2015-07-15', fontsize=120)
axes[3].set_title('2017-12-08', fontsize=120)

plt.tight_layout()
plt.show()
```

Out[30]:
```
Text(0.5, 1.0, '2015-05-17')
```

Out[30]:
```
Text(0.5, 1.0, '2015-06-22')
```

Out[30]:
```
Text(0.5, 1.0, '2015-07-15')
```

Out[30]:
```
Text(0.5, 1.0, '2017-12-08')
```




In [31]:
```
#save the graph
fig.savefig('TimeSeriesAnalysis.png', dpi=300, bbox_inches='tight')
```

In [ ]:
```
#References:
#https://networkx.org/documentation/latest/tutorial.html
#https://matplotlib.org/cheatsheets/
#https://pandas.pydata.org/docs/user_guide/index.html
#https://docs.python.org/3/library/collections.html
```