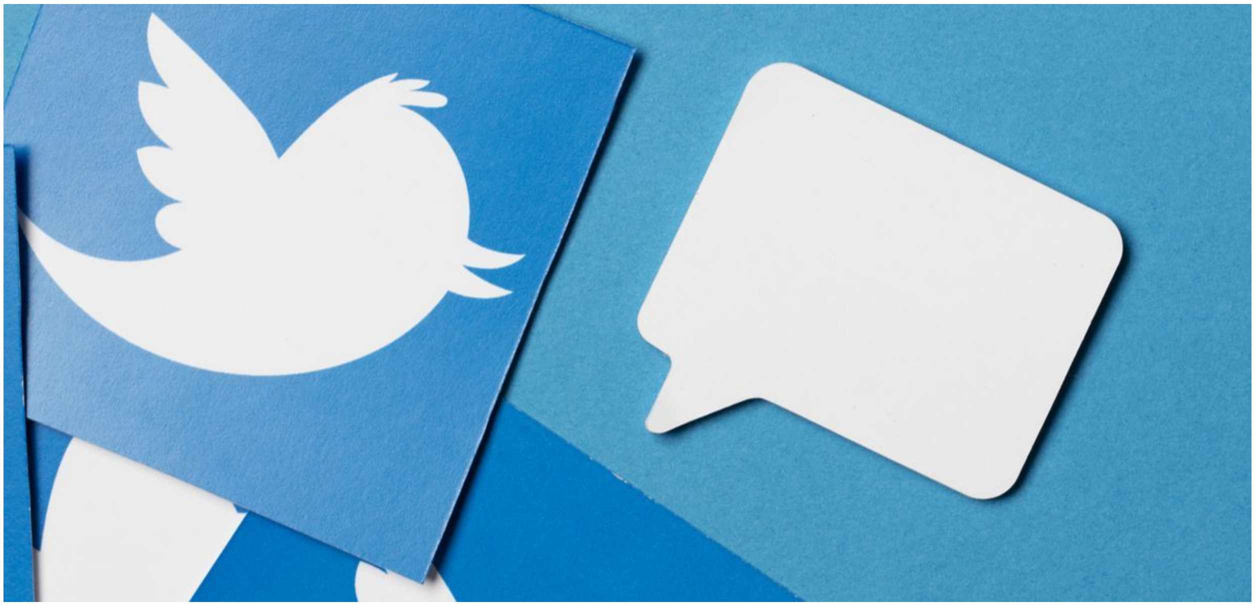


# Analyzing a Twitter conversation


Submitted By:

Aujasvi ( MS Data Science)

Akshi Kathait (MS Data Science)

Polytech Nantes, University of Nantes

Submitted To:

Prof. Pascale Kuntz

Dept. of Computer Science

Polytech Nantes, University of Nantes

# Introduction:

Twitter data is the information collected by the user, the access point, what's in the post and how users view or use the post. Twitter launched in 2006, has several hundred million users and is primarily categorized as a microblogging social network whose posts (tweets) can contain a maximum of 140 characters. Users write tweets on different topics that include their current activities, tweet, retweet and share links to interesting media. Once a user has posted a tweet, all her followers will receive the tweet. Tweets posted by a public profile user can be viewed online by anyone. In particular, we would like to understand the conversation graphs that are formed when users in the network discuss a particular topic.

# Problem Statement:

We wish to examine the underlying graph structure in twitter social network and to understand how to get data from Twitter using API's, information propagating in the network, analysing the important nodes in the network. Here, we are analyzing conversational graphs with **Gephi** on the topic #DataScience on Twitter that represents how users are interacting within the network in the form of nodes and edges.

# Process of analysis:

## 1.Getting the Twitter API keys:

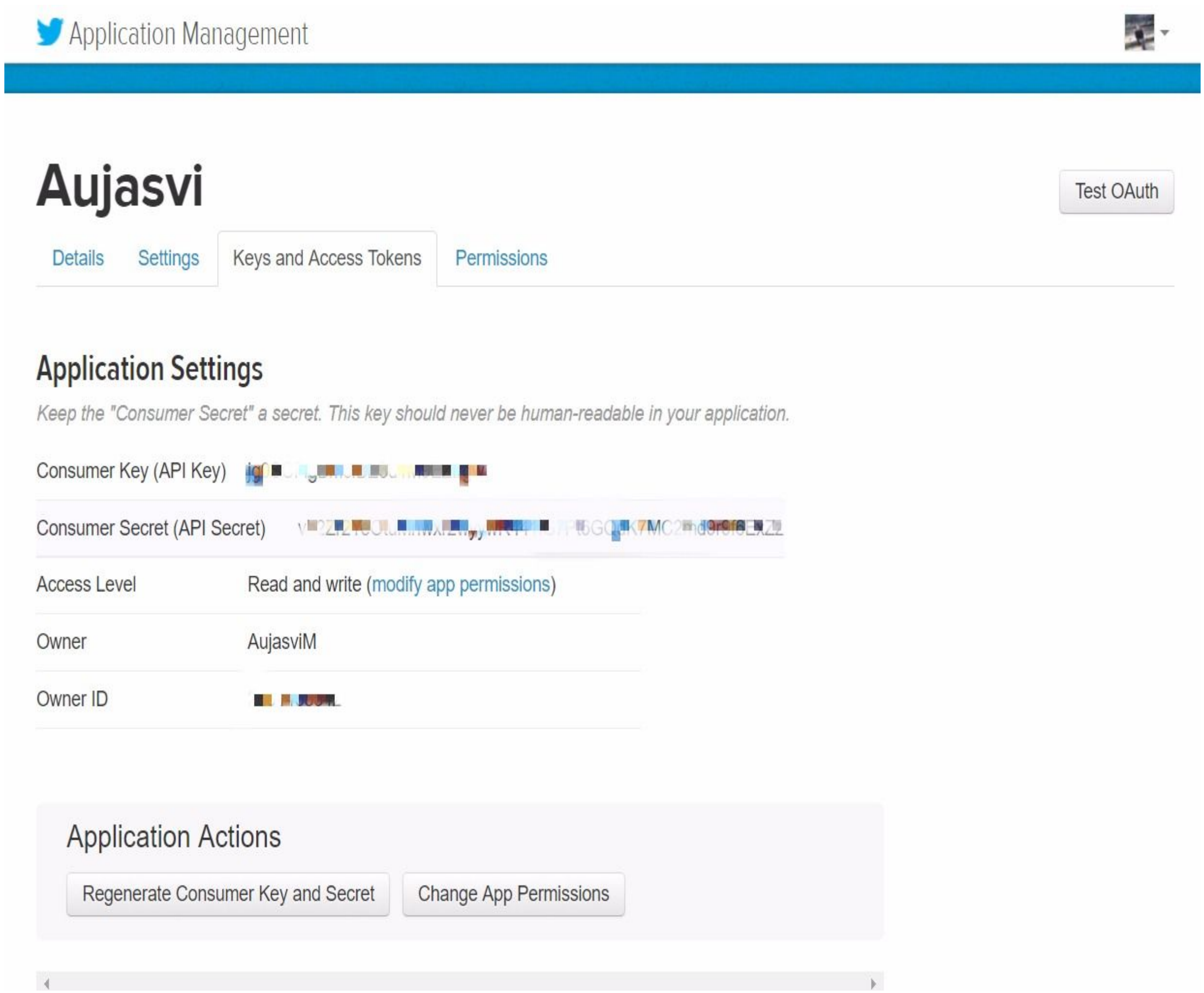
The The Twitter API is the programming interface script will use to get data from Twitter. To use the Twitter API the client needs to be authenticated. To do that we need to register an app first.

1. Go to apps.twitter.com

2. Create new app (top right)

3. Fill out Name and description

4. Switch to the 'Keys and Access Tokens' Tab, We will need the Consumer Key and the Consumer Secret later.

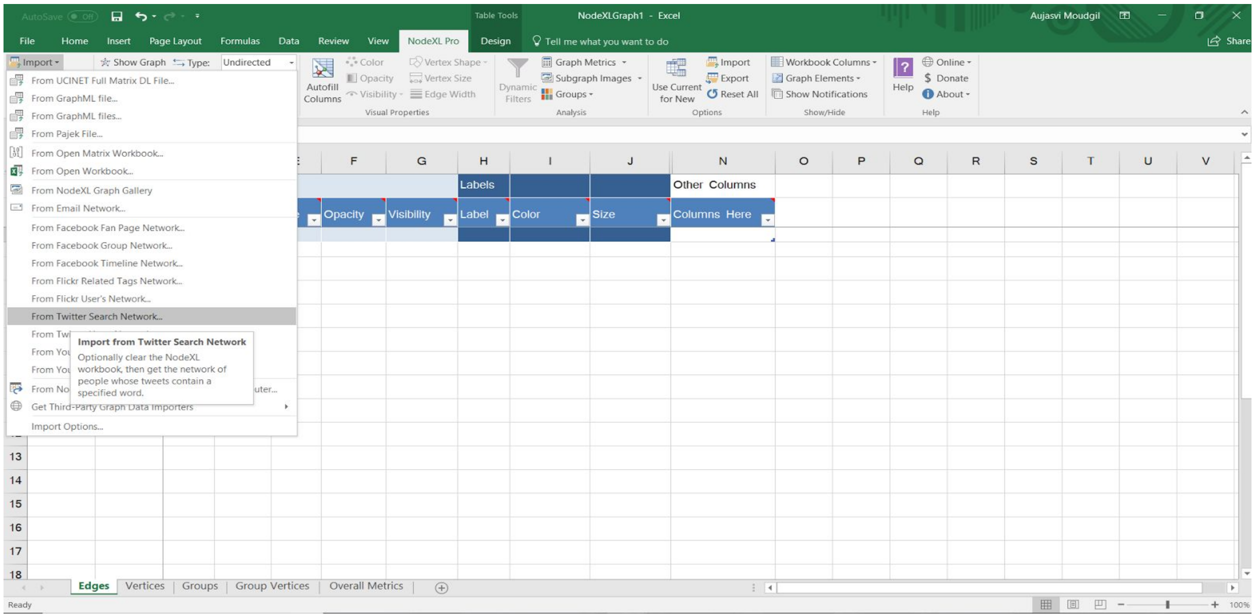



2. We, download the software NodeXL and open a blank nodeXL template. NodeXL is a general purpose network analysis application that supports network overview, discovery and exploration. The tool enables the automation of a data flow that starts with the collection of network data and moves through multiple steps. Data can be imported through text, CSV, or GraphML files. Here, we are importing the in GraphML file format.

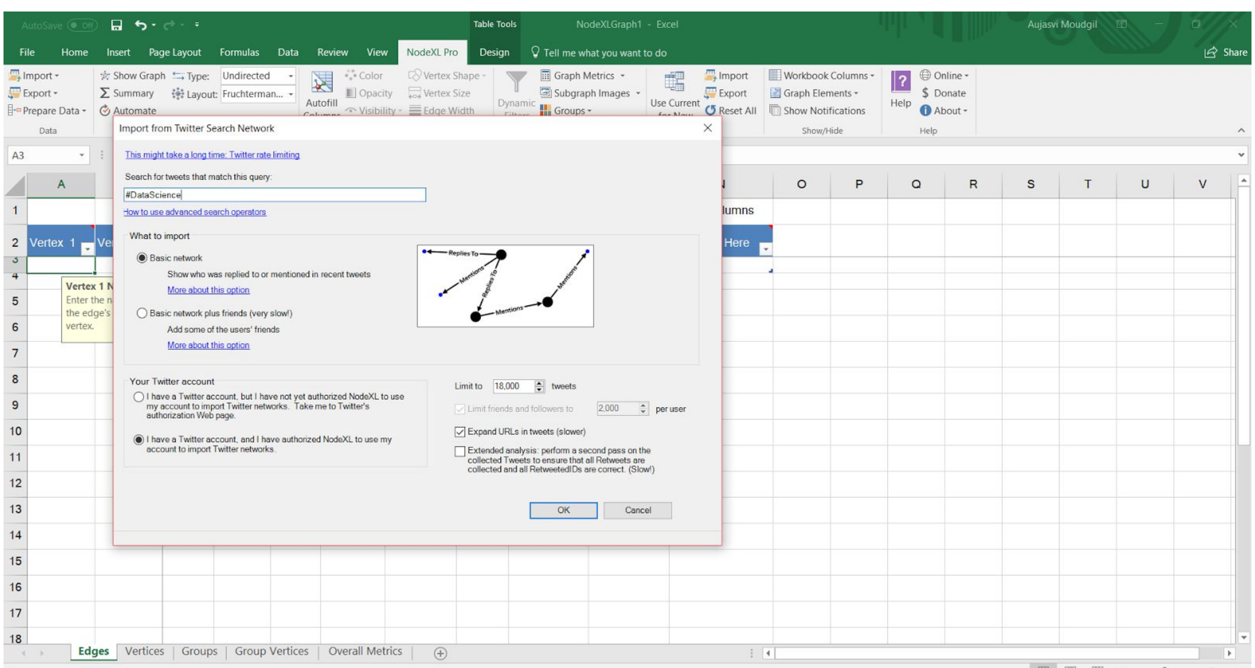
3. A network graph basically consists of **nodes** (in this case Twitter users) and connections between them, which are called **edges**.

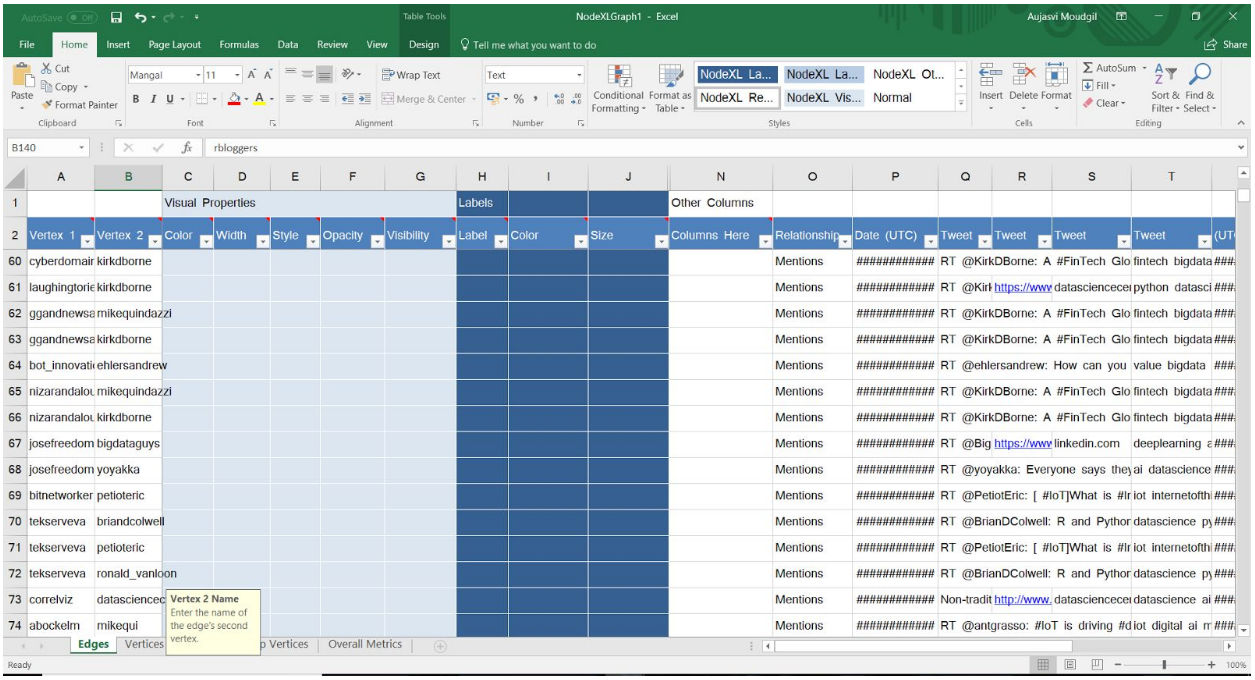
4. We're going to import nodes and edges for all the users who've tweeted using the **#Data Science** hashtag.

5. Click on Import and choose **“Import from Twitter search network”.**




6. We filled in the hashtag #Data Science network we want to analyse. We can import 18,000 tweets on this hashtag but in practice, Twitter rarely returns 18,000 tweets. Data is assembled from the results of many queries to Twitter about the connections among the authors in the data set. The results are displayed in a NodeXL worksheet labeled “Edges” in the workbook.

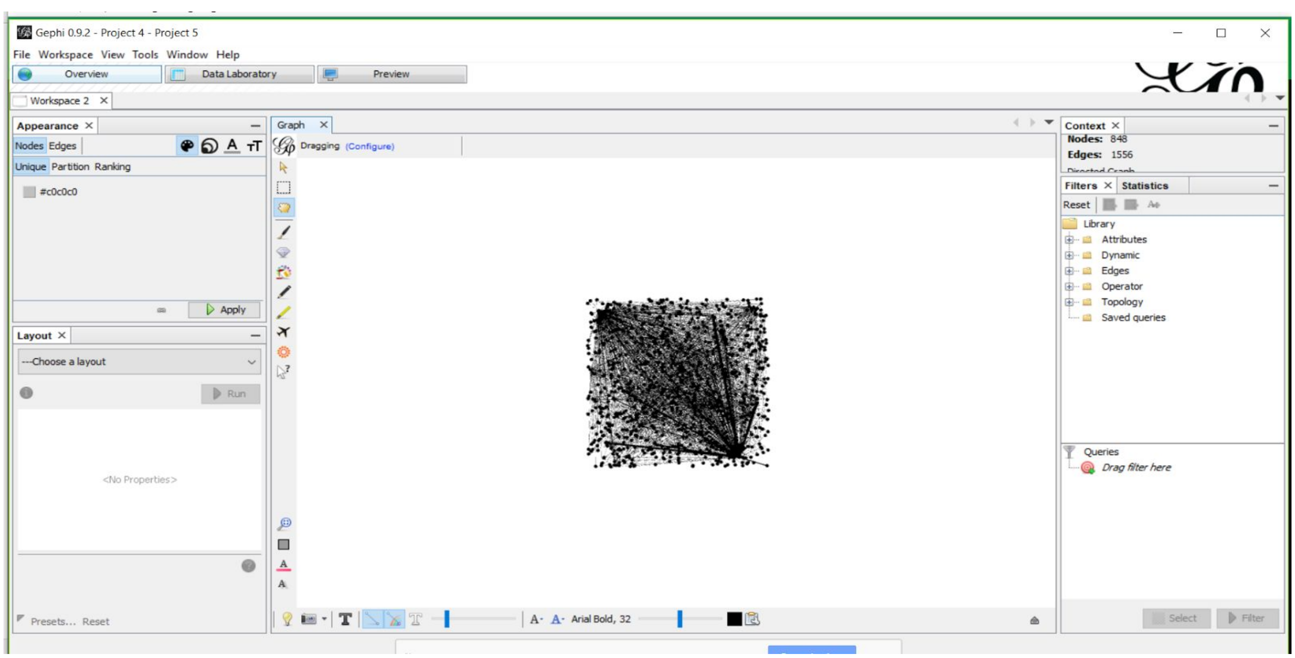




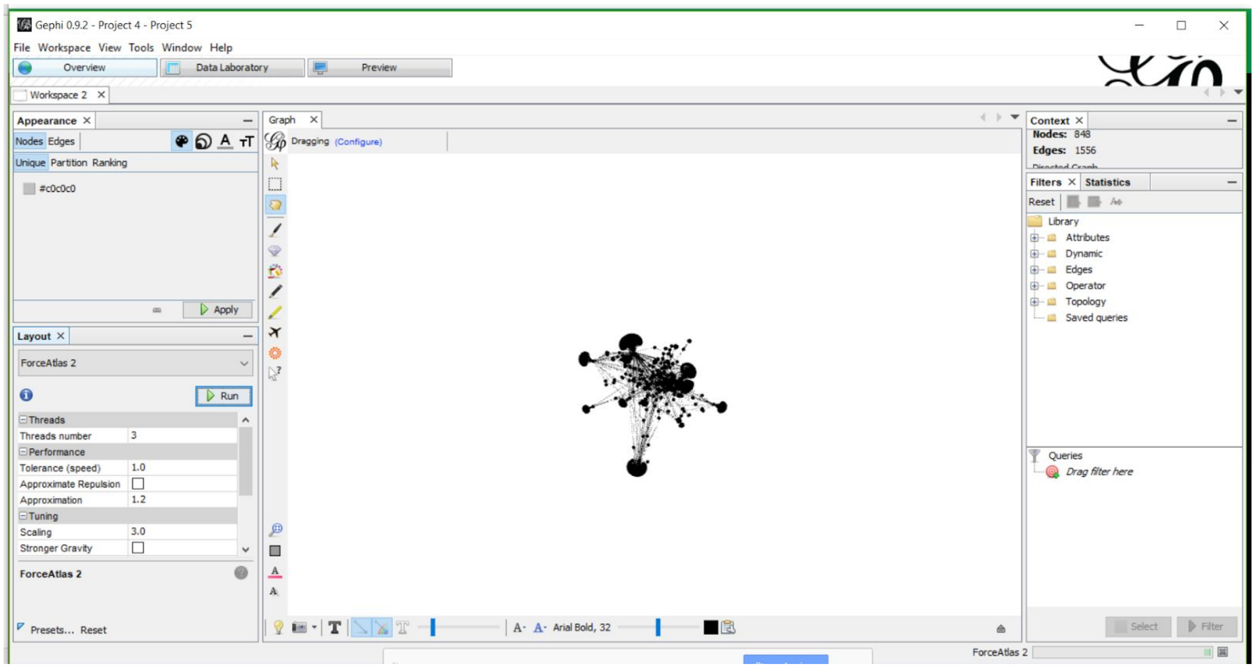
Every **vertex**, or **node**, is a Twitter user, and NodeXL has logged all the ways in which it has connected with other vertices using the #DataScience hashtag. The tool fetches plenty of useful information about the tweets and the tweeters from number of followers and bios to whether something is a retweet.

7. **Export as GraphML file in Gephi**.

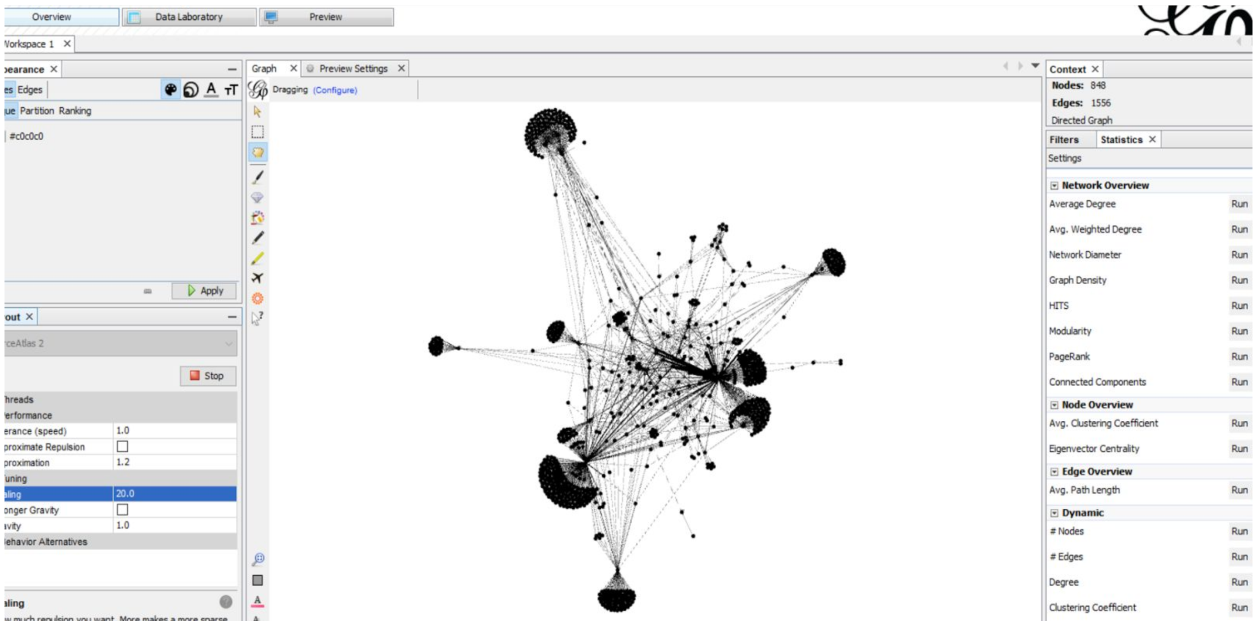
8. As we import a file to **Gephi,** we got a graph that looks like this:



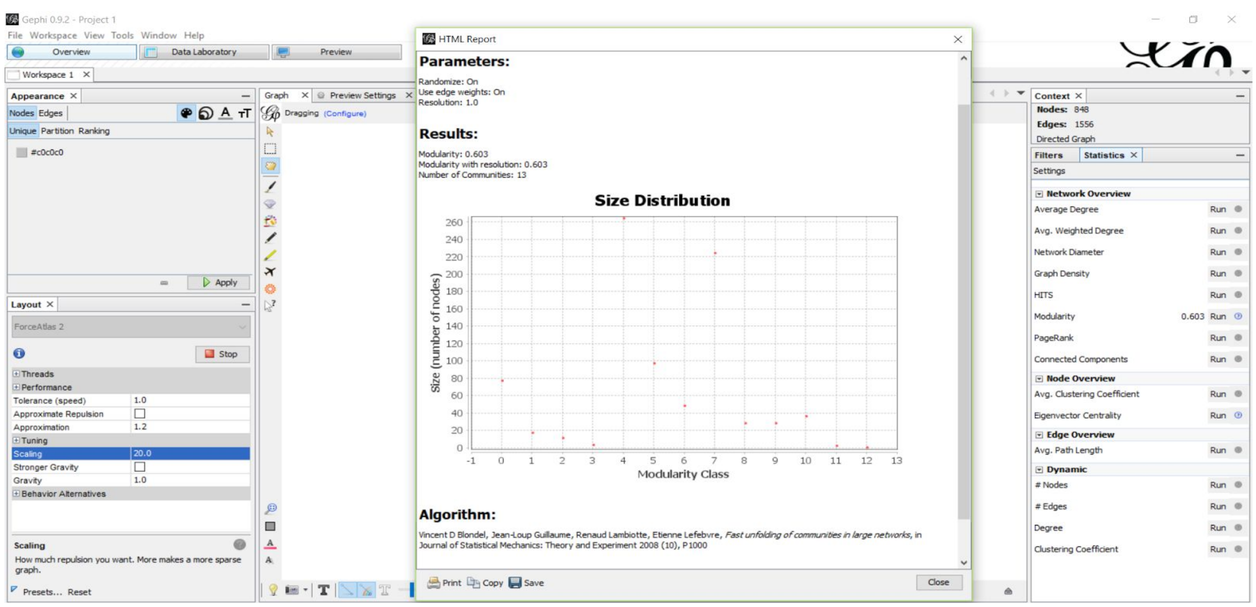
9. Then from layout we select, **Force Atlas 2.** (It's force-directed algorithms available in Gephi which attempts to resolve the shortcomings of the Force Atlas algorithm by making a balance between the quality of the final layout and the speed of the computation algorithm. Its performance for large networks is much better when compared to the Force Atlas layout algorithm.)



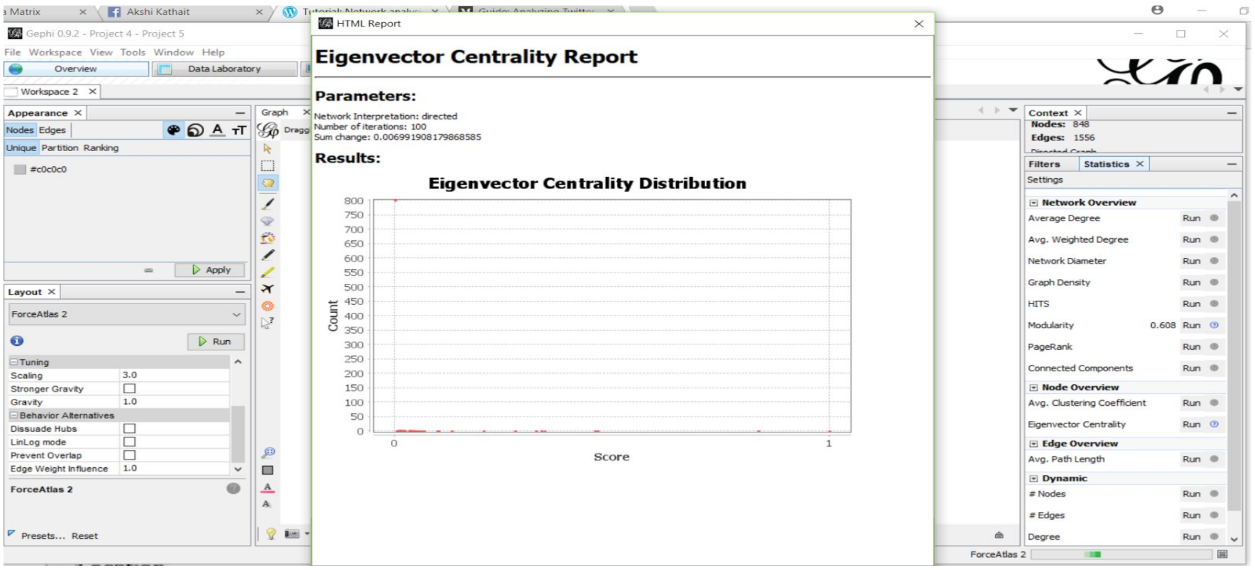

Since the graph is not visible properly, we set the **scaling** to something like 20.0 (The scaling will affect how closely the nodes in the network are drawn to each other). Click on Run and the change appeared in the graph.



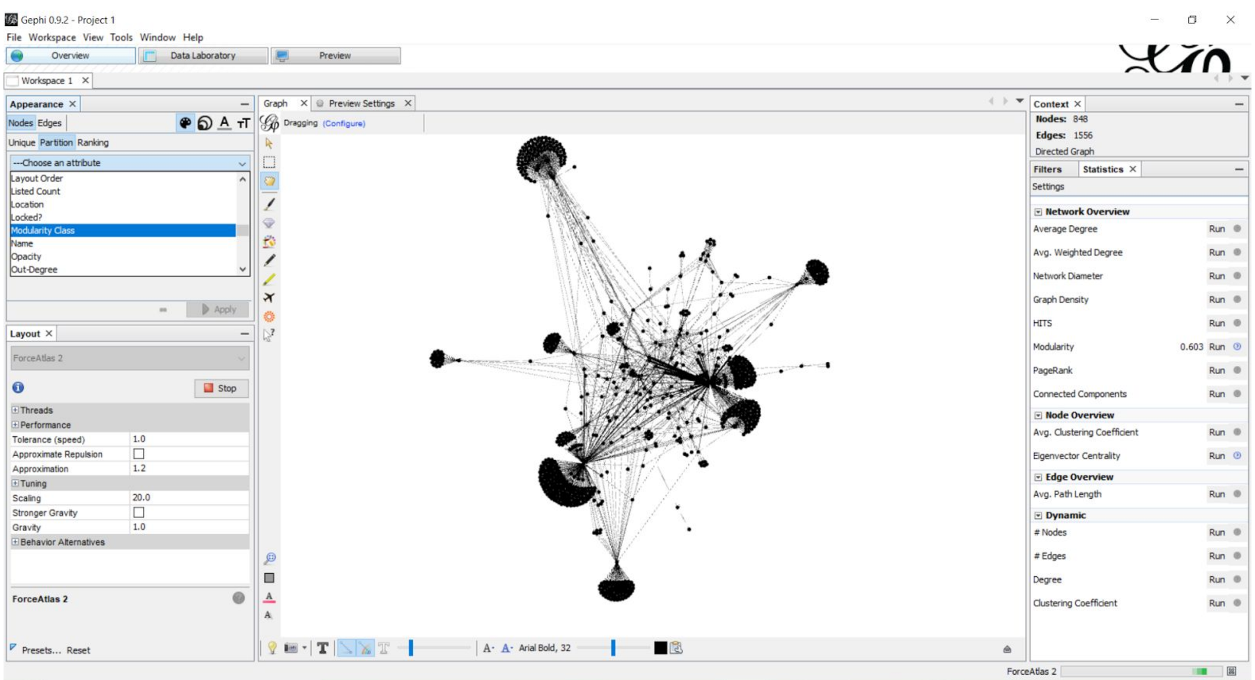

3. On the right-hand side, there are number of settings in which **Modularity** is one of the measure to structure the networks or graphs that are designed to measure the strength of division of a network into modules (also called groups, clusters or communities). Networks with high modularity have dense connections between the nodes within the modules while sparse connections between nodes in different modules. By clicking Run on **Modularity** we can see that network has 13 communities.



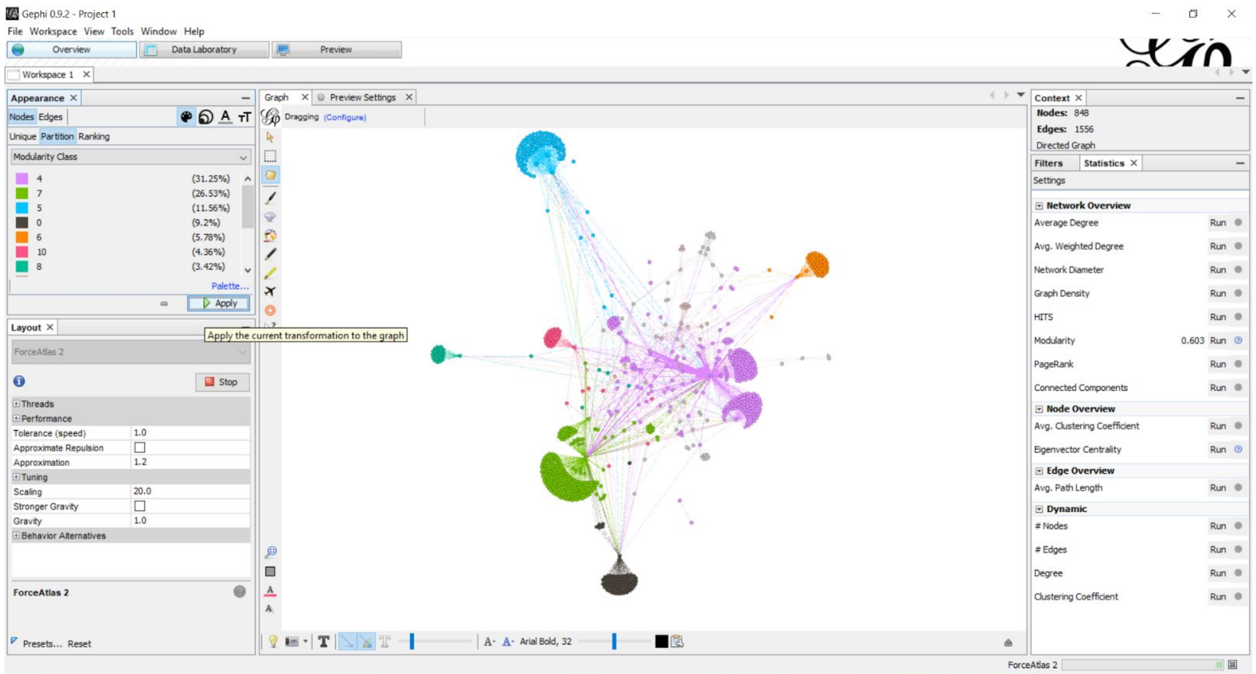
And another measure is **Eigenvector centrality.** It's  a useful way of measuring how important a node is within its network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.



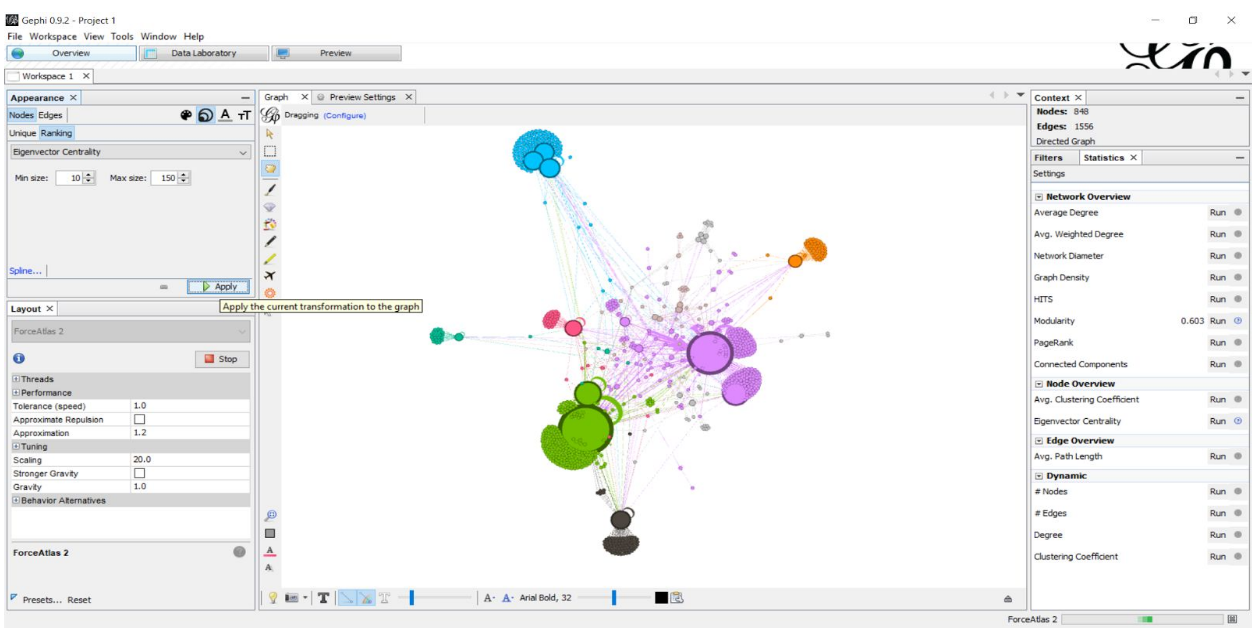

4. To get a more useful or visible layout, we will color the nodes based on their groupings. To do this, click on **Partition** in the top left and choose **Modularity** as the partition parameter. Click Apply.



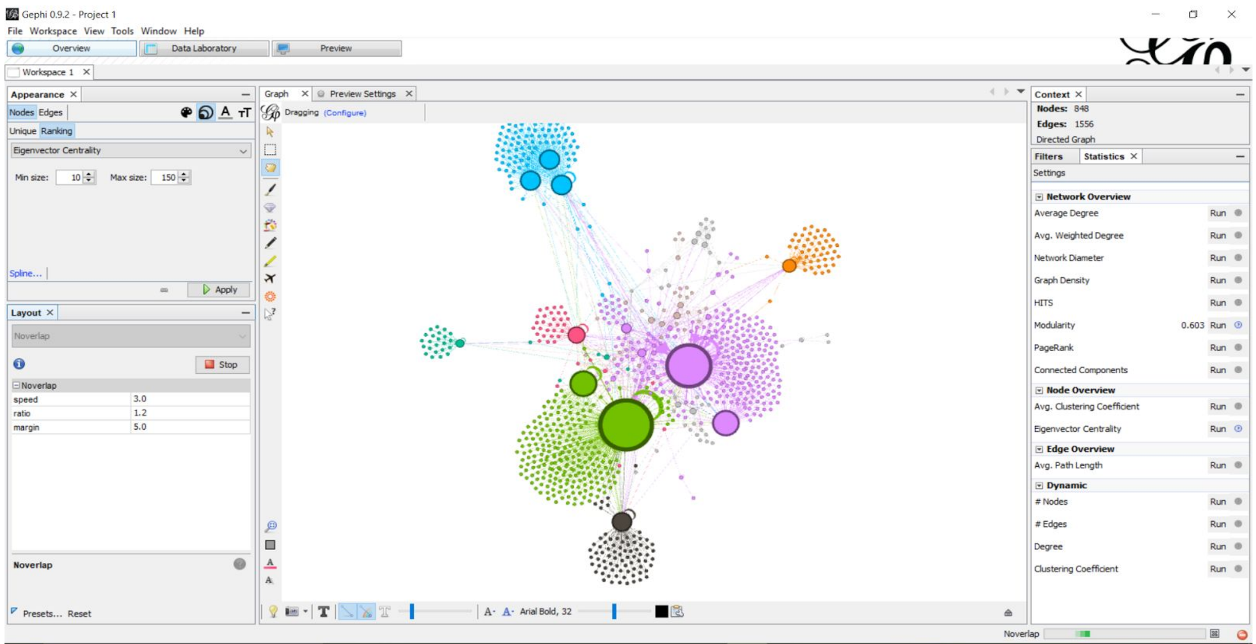
After coloring the nodes, the graph looks something like this:



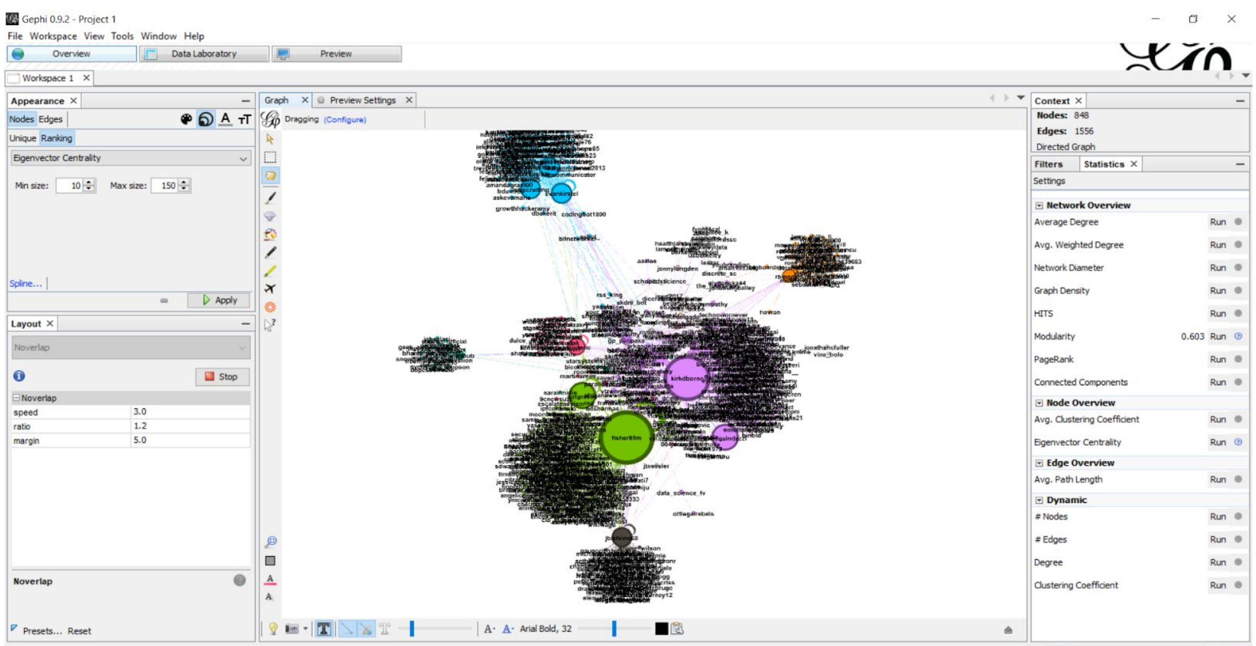

5. Next, we want the nodes' size to reflect how important they are to the network. To do this, Click on **Ranking** and select **Size by Eigenvector centrality**. We kept it in a range of minimum of 10 and maximum of 150.



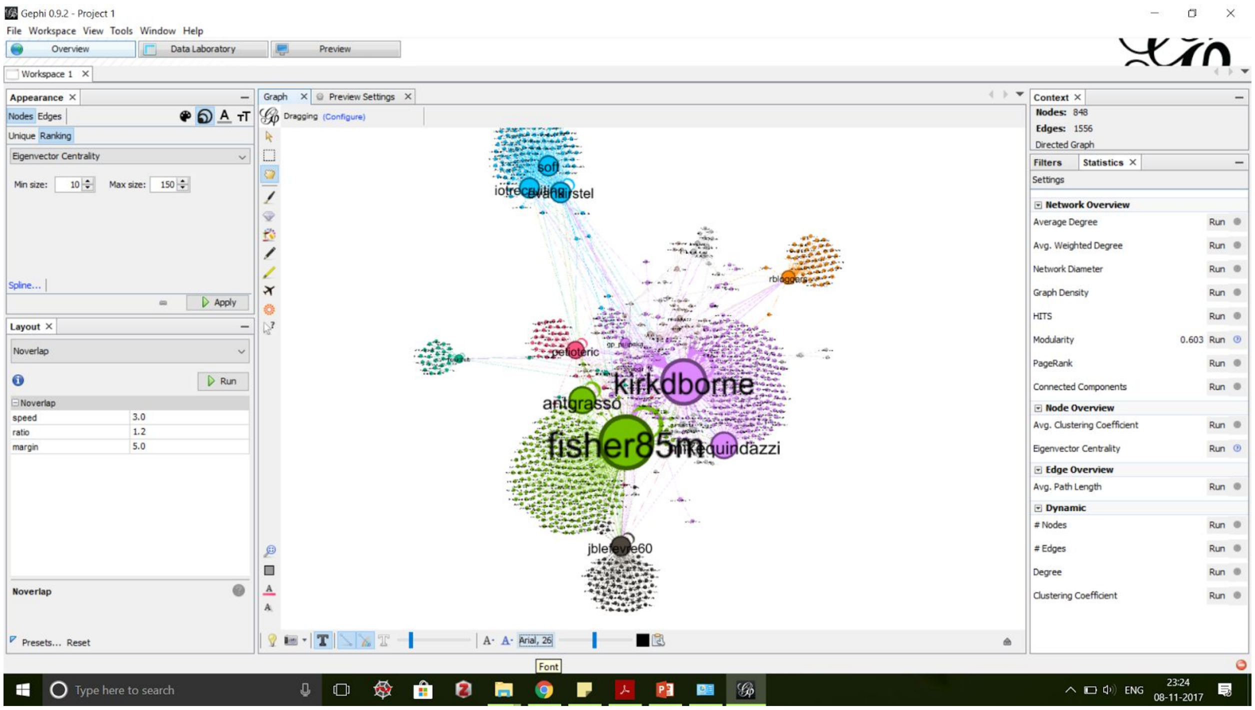
6. We've now got differently sized nodes depending on how central they are – but they're all jumbled up. So, to make them more clear we go back to **layout tab** and we select the **No overlap layout** (as we see in above graph the nodes and labels are overlapped which make them harder to view, this can be solved by using **No overlap layout**). Run that again until the graph is more evenly spaced out.



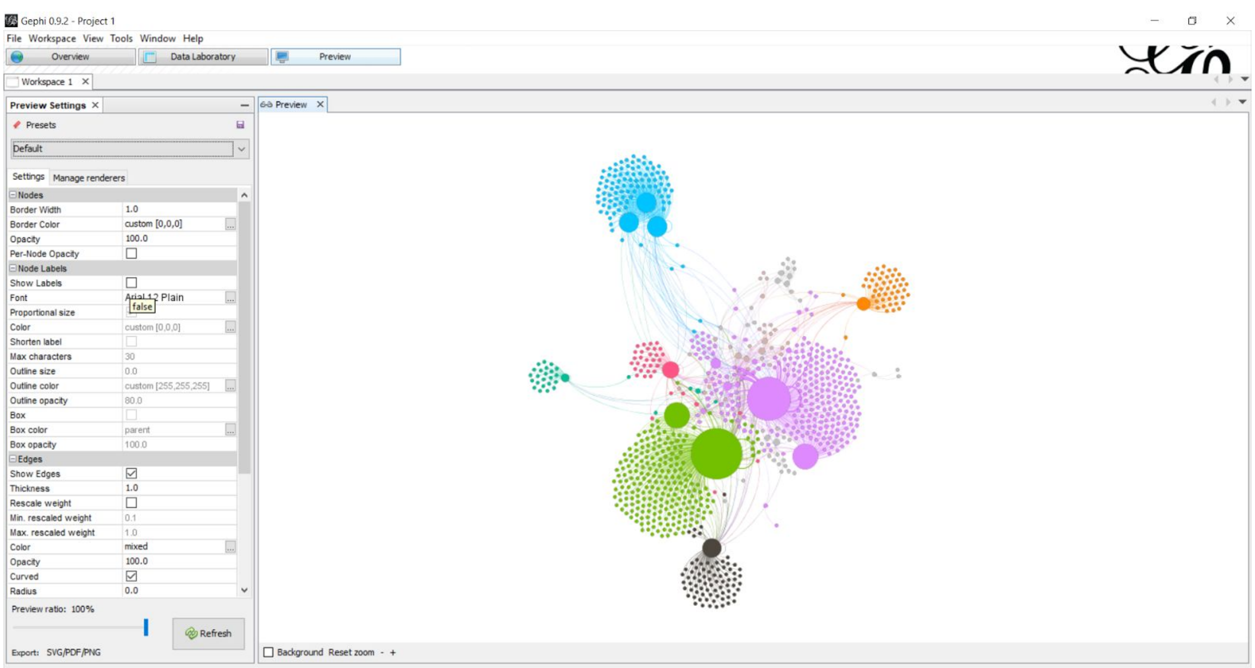

7. After this, To make Labels visible in the graph, Click on **T.**



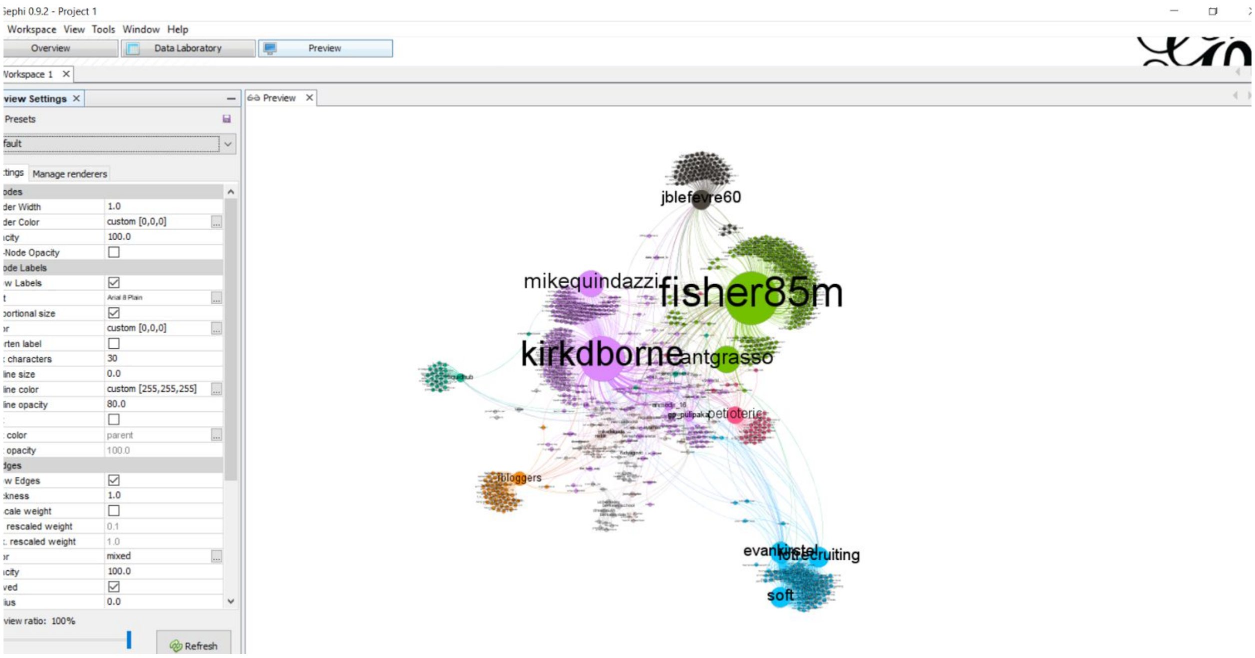
& then Click on **A** next to it and select Scale labels by node size.

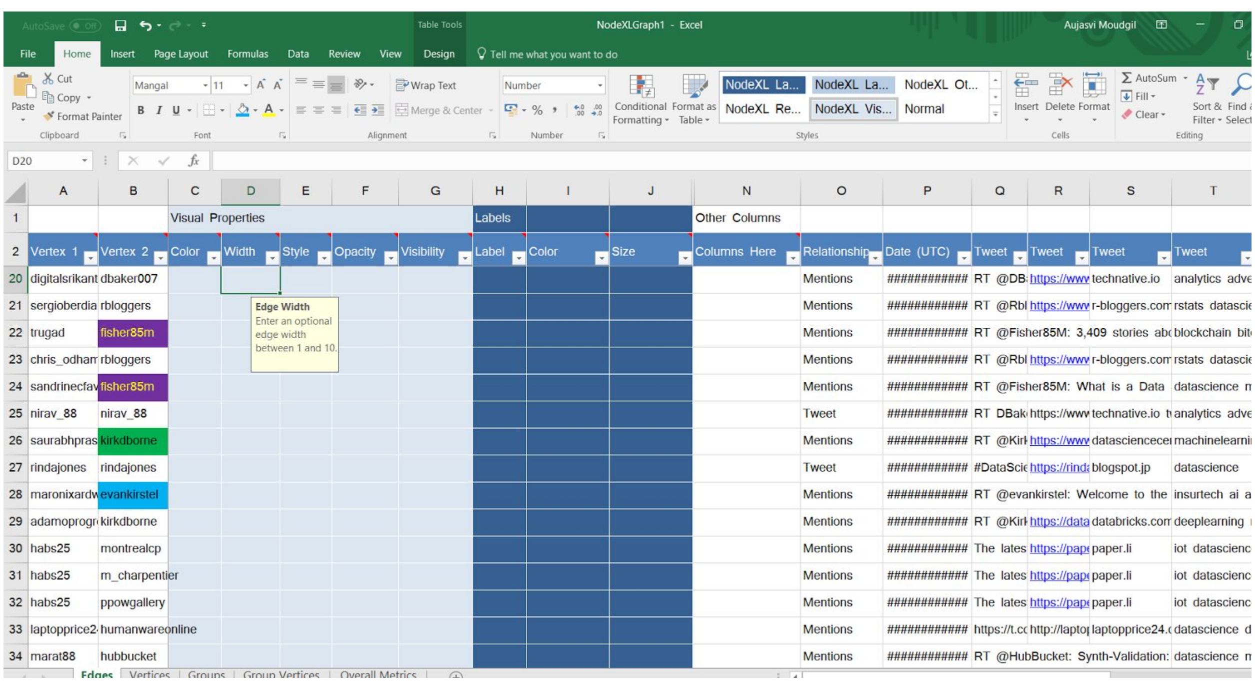



8. For the creation of our final product, click on the ‘Preview’ tab at the top of the window. After clicking ‘refresh’ in this window, our graph appeared in its final form.



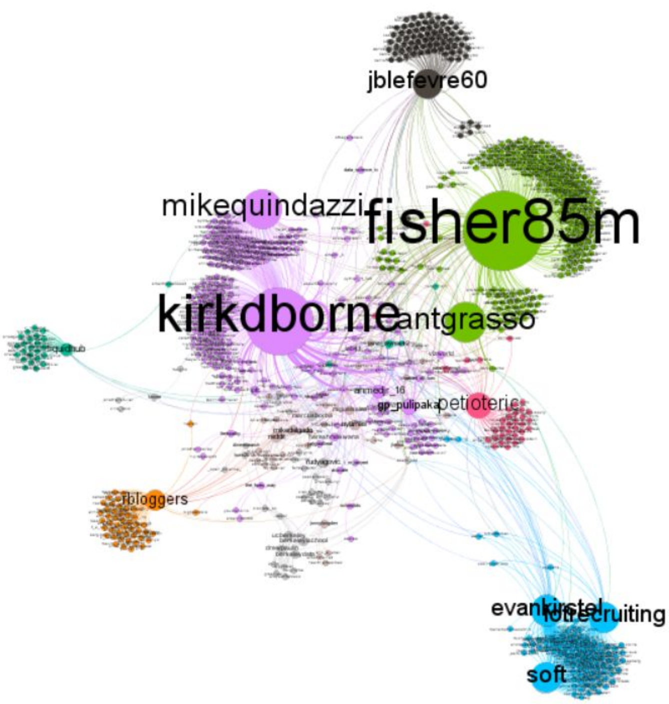
We can fine tune the actual appearance some more, like check the button of **show labels** on the left hand side of the graph.



# Analysis of Result:

One natural feature of any graph is the fact it evolves over time. Here, the graph follows **power law distribution (scale free)** where a relative change in one quantity results in a proportional relative change in the other quantity, independent of the initial size of those quantities: one quantity varies as a power of another. This graph also features the **Matthew effect** ie. "Richer gets Richer" like ***fisher85m, kirkdborne, evankristel*** who are tweeting about #DataScience on Twitter are represented by significant and main nodes in the graph. So, all the new incoming nodes have tendency to join with these nodes making more bigger community.




# References:

[1] : twitter account

[2]. P. Cogan, M. Andrews, M. Bradonijc, W.S. Kennedy, A. Sala, G. Tucci (2012). Reconstruction and analysis of Twitter conversation graphs. In Proc. *of the 1st ACM Int. workshop on Hot Topics on Interdisciplinary Social Networks Research, pp. 25-31.*

[3]. B.A. Mathew, T. Chakraborty, N. Ganguly, S. Datta (2016). Mining twitter conversation around e-commerce Promotional events, *In Proc. of the 19th ACM Conf. on Computer Supported Cooperative Work and Social Computing Companion, pp. 345-348.*

[4]. Tutorial :: Blog