University of Central Florida

Department of Statistics

Report on

Social Network Analysis

Kanak Choudhury

Supervised by
Dr. Alexander V. Mantzaris

May 4, 2017

Contents

Chapter 1

Review of "A New Status Index Derived From Sociometric Analysis by Leo Katz"

1.1 Introduction

The ordinary indices of *status* for investigating inter-personal and inter-group relations are based on the number of direct *votes* received by each individual and most of the researchers were not satisfied with these indices. In this article, author proposed a new method of computing status based on the number of direct *votes* received by each individual and the steps of choosing the status. In other words, the new index considered who chooses and how many choose.

1.2 The New Status Index

To illustrate the new status index, a matrix representation for sociometric data (Page 40) taken from [1] was used. The data matrix should be a square matrix with principal diagonal elements are zeroes. The data matrix was represented as C, with element c_{ij} is the response of individual i to individual j and the response can be either 0 or 1. From this data matrix, if we want to find the higher order chooses, i.e. i chooses k and k chooses j that is chain of length two from i to j, can be obtained by $C^2 = (c_{ij}^{(2)})$, where $c_{ij}^{(2)} = \sum_k c_{ik}c_{kj}$; $c_{ik}c_{kj}$ is equal to one if and only if i chooses k and k chooses j, otherwise zero. This representation is exactly equal to Markov chain. We can find this higher order Markov chain by simply multiplying C matrix.

The column sums of C give the numbers of direct choices made by members of the group to the individual corresponding to each column. Also, the column sums of higher order matrix i.e. C^2, give the numbers of two-step choices from the group to individuals and so on.

To find the index, it was proposed by weighted sum of all higher order matrices. The weighted sum was used to reduce the effect of longer chains and finding appropriate weight method called *attenuation* were described.

The new index calculation was based on two assumptions. (i) the information is accurate that is certain links between individuals exist and if the information indicates no link that means there is no communication, influence, or whatever else it is measured. (ii) Each link independently has the same probability of being effective [2].

Based on the second assumption, a constant a indicated as the force of a probability of effectiveness of a single link was considered which measures the non-attenuation in a link, i.e. $a = 0$ corresponding to complete attenuation and $a = 1$ to absence of any attenuation. This constant a (for C^2, C^3, higher order a^2, a^3, respectively) was considered as the weight for the index calculation.

Let s_i be the sum of jth column of the matrix C and s be a column vector with elements s_i. Then, the column sums of the matrix

$$T = aC + a^2C^2 + \cdots + a^kC^k + \cdots = (I - aC)^{-1} - I$$

$$(\frac{1}{a}I - C^{'})t = s \tag{1.1}$$

where, t be a column vector with $t^{'} = u^{'}[(I-aC)^{-1} - I]$ and u be a column vector with unit elements. So, to obtain t, it is needed to solve the system of linear equations with given a, C, and s. It is mentioned that the general-purpose values of $1/a$ should be between the largest root and twice that root.

Dividing the column sum s_i by $n-1$ $\big((n-1)^{(k)} = (n-1)(n-2)\cdots(n-k)\big)$ that is the number of possible choices for C $(C^{(k)})$, index of status can be obtained. So

$$\begin{aligned} m &= a(n-1) + a^2(n-1)^{(2)} + a^3(n-1)^{(3)} + \cdots \\ &\approx (n-1)! \cdot a^{n-1} e^{1/a} \end{aligned}$$

Finally, the new status index vector is given by t/m, where t is the vector solution to the system of equations (1.1).

1.3 A Numerical Example

Based on the above index calculation, a numerical example was illustrated and found that the new status indices were able to capture the relative position based on the objectives of this article than the ordinary indices.

Chapter 2

Review of "A Model for Dynamic Communicators by Alexander V. Mantzaris and Desmond J. Higham"

2.1 Introduction

For a single static network model, random graph models and centrality measures are very useful tools. However, for dynamic network model where links between nodes may appear and disappear in a time-dependent manner, a new models and algorithms are needed. Examples of dynamic network include email activity, voice calls, online social interaction, geographical proximity of mobile device users, dynamic transportation infrastructure, voting and trading patterns and neural activity etc. For a dynamic network model, final state of an iterative process is not the main concern rather continual change in topological structure is the main consideration in this article.

2.2 Background And Motivation

In this section, authors tried to present the problems of static network model for dynamic scenario of communication with some hypothetical examples. Figure 1 [3] showed undirected, unweighted network communication between a set of 21 nodes over three days. From three different subfigures of Figure 1, we might consider that node 21 was not unusually important. However, node 21 was a special node based on *timing*. Because, at day one node 21 reached to nodes 19 and 20 and then the network spread on subsequent day, that is, nodes 16 to 18 on day two and nodes 12 to 15 on day three. It was explained node 21 as an *influential* player because whenever other individuals receive a message that can be traced back to node 21 they burst into action and pass the message on. For example, when a terrorist group get a message from the leader, they tried to spread it into other networks. So, node 21 can be responsible for future network structure. For example, Donald Trump, precedent of USA, might post a tweet and after that it can spread over other networks. For all the network based on that tweet, he is the most influential on the whole network although his direct communication is just a tweet. Because of this reason authors introduced a new, general dynamic network model based on simple but intuitively reasonable principles that can captures the effect.

For a xed set of N nodes and time points $t_0 \leq t_1 \leq \quad \leq t_M$, it was considered an ordered sequence of unweighted graph adjacency matrices $A^{[k]} \in \mathbb{R}^{N\times N}$, so that $(A^{[k]})_{ij} = 1$ if there was a link from node i to node j at time t_k and $(A^{[k]})_{ij} = 0$ otherwise. It was found that even in the case of undirected networks, where each $A^{[k]}$ is symmetric, dynamic walks lack symmetry with dynamic walk of length w. The computation of the matrix $Q \in \mathbb{R}^{N\times N}$, for which $(Q)_{ij}$ is a weighted count of the number of dynamic walks of length w from node i to node j, was shown in [4] where walks of length w are scaled by a factor a^w. To find effectiveness of broadcast and receive dynamic messages, the following row and column sums were used for centrality measures.

$$C_n^{\text{broadcast}} := \sum_{k=1}^{N} Q_{nk} \text{ and } C_n^{\text{receive}} := \sum_{k=1}^{N} Q_{kn}$$

A normalized matrix $Q/\|Q\|$ were used in order to avoid numerical under or overflow [3].

Using new method, authors found largest dynamic broadcast centrality for the node 21 in Figure 1 although it was ranking much lower based on static measures. They used the term *dynamic communicator* to describe a node that has excellent centrality in the dynamic sense while it was not represented by snapshot or aggregate views of the network sequence.

2.3 Practical Observations

In this section, authors presented a real example covers dynamic broadcasters. They used two weeks of Enron email data [4] and presented the solution in Figure 2. For one-day time interval, it was found in Figure 2 (a) that nodes 1 and 2, an executive and the vice president, had the highest broadcast centrality but modest total out degree which are considered as dynamic communicators. On the other hand, nodes 3, 4 (both correspond to traders) and 5 (unknown role) had high bandwidth but relatively poor broadcast centrality. It was mentioned that for the time-dependent setting, nodes having very high out degree was neither necessary nor sufficient to guarantee influence amongst other nodes. Using two-day period (Figure 2 (b)), the same nodes (1 and 2) were observed as dynamic communicators while an extra node, labeled number 6 (an employee), was found with other 3, 4, and 5 nodes with high bandwidth but relatively poor broadcast centrality.

Similar results were found (Figure 3) using 30 days of voice call data between academics [5].

2.4 New Model

The new model that the authors proposed was considered as the generalizing the concept of network hierarchy from the static case [6]. It was assumed for the new model that there was an underlying hierarchy which can be explained that some nodes have high importance to expand the future network. This type of hierarchy might arise through an imposed chain of command, i.e. business or military organizations, criminal networks, completion of tasks as in on-line gaming.

This new model was based on two objectives, "(i) identifying a key feature in dynamic human inter action data sets and (ii) oering a simple, intuitively reasonable, explanatory mechanism" [3]. It was considered that where was the particular pairs of nodes involved in communication not the particular time of the events for a single node.

The proposed model was started by assigning a fixed level of importance,l_n, to each node n so that $0 < l_1 \leq l_2 \leq \quad \leq l_n$. The concept behind this was that low ranked nodes might generate high communication when received messages from highly ranked nodes. Mathematically, given the time t_k network, $A^{[k]}$, $A^{[k+1]}$ was generated based on basal and responsive.

Basel: with probability b node n generates a xed number c_b of links, with the new neighbors chosen uniformly and independently at random. Otherwise no basal links are generated from node n [3].

Responsive: with probability

$$r_n^{[k]} := \frac{\sum_{i=1}^{N} l_i(A^{[k]})_{in}}{1 + l_N \sum_{i=1}^{N} l_i(A^{[k]})_{in}}$$

node n generates a xed number c_r of links, with the new neighbors chosen uniformly and independently at random. Otherwise no responsive links are generated from node n [3].

Results of this model were shown in Figure 4 and found consistent result for different scaling parameters of $a = 0.75, 0.5, 0.25$. For every case, model found the same node as a dynamic broadcaster while shoed ranking 26th in terms of aggregate degree.

The authors added that the dynamic broadcasters bear higher importance because links from the broadcasters and links from the links might become active and create the future network. So, dynamic receivers might have added global, historical knowledge that they know which nodes are currently most informative and deliberately form links with them.

2.5 Conclusion

In this article, authors presented a new model that can describes the dynamic presence or absence of connections in an evolving network.

Chapter 3

Review of "Communicability Across Evolving Networks by Grindrod, P., Higham, D. J., Parsons M. C. and Estrada, E."

3.1 Introduction

Connectedness, path length, diameter, degree and clique play the main important factors to establish the network science. In this article, authors tried to establish a new type of time-dependent network model. In Figure 1, it was represented the asymmetry network caused by the arrow of time although each individual network is symmetric. That was the motivation to distinguish between static and dynamic networks and pointed out the need for a theory that can deal with-

1. the time ordering inherent in the edge lists when considering communication around the network,

2. Respects the inherent asymmetry imposed by the arrow of time, even when each individual snapshot consists of an undirected network [7].

There are many applications where the network connectivity patterns changes over time. Examples include network of instant messaging systems (Facebook, MSN) [8, 9]; networks of travelers, vehicles dened over a dynamic transportation infrastructure [10–12]; correlated neural activity in response to a functional task [13].

In this article, authors implemented a new extended centrality concept, that was related to [14–16], to address dynamic network. Authors tried to emphasize in this article that in the time-dependent network, the population of nodes remained constant from the outset and the graph evolves through the appearance (birth) or the deletion (death) of edges.

3.2 Katz Centrality

In this section, authors described Katz centrality [2] that works with a single, static network. Let A denote N-by-N binary adjacency matrix for a directed graph G defined over N nodes, where A_{ij} represents one if there is a link from node i to node j and $A_{ij} \neq A_{ji}$ so that the adjacency matrix may be asymmetric.

To find the propensity for node i to communicate, or interact, with node j, it was counted the number of walks of length w from i to j and combined these counts into a single, cumulative total over all w. The walks of length w were scaled by a factor a^w to find the shorter walks, where a is a suitably chosen scalar. The kth power of the adjacency matrix has i, j element that counts the number of walks of length w from node i to node j and the expansion of $I + aA + a^2A^2 + a^3A^3 + \cdots$ converges to $(I - aA)^{-1}$ where $I \in \mathbb{R}^{N\times N}$, $a < {}^1\!/_{\rho(A)}$, and $\rho(\cdot)$ denotes the largest absolute eigenvalue. Since $((I-aA)^{-1})_{ij}$ summarizes how well information can pass from node i to node j and the nth row sum

$$\sum_{k=1}^{N} \left((I - aA)^{-1}\right)_{nk} \tag{3.1}$$

represents the Katz centrality. This centrality measure is based on the combinatorics of walks, which allow nodes and edges to be reused during a traversal, rather than paths or shortest paths. The Katz centrality (3.1) measures the ability of node n to send out information along the directed links. On the other hand, to measure the ability of node n to acquire information can be found by column sum of $((I - aA)^{-1})_{ij}$:

$$\sum_{k=1}^{N} \left((I - aA)^{-1}\right)_{kn}.$$

3.3 Dynamic Centralities

In this section, authors described their proposed method of dynamic centrality for network. It was considered, for a given set of N nodes, an ordered sequence $G^{[k]}$ for $k = 0, 1, 2, \cdots, M$ with time points $t_0 \leq t_1 \leq t_2 \leq \cdots \leq t_M$, where each $G^{[k]}$ is an unweighted graph defined over those nodes and time t_k with corresponding adjacency matrix, $A^{[k]}$. The generalization of adjacency matrix, $A^{[k]}$ for the static graph concept of a walk was defined to explain how well information can be passed between pairs of nodes as following.

Definition 1: A dynamic walk of length w from node i_1 to node i_{w+1} consists of a sequence of edges $i_1 \to i_2, i_2 \to i_3, \cdots, i_w \to i_{w+1}$ and a non-decreasing sequence of times $t_{r_1} \leq t_{r_2} \leq t_{r_3} \leq \cdots \leq t_{r_w}$ such that $A^{[r_m]}_{i_m, i_{m+1}} \neq 0$ and the lifetime of this walk to be $t_{r_w} - t_{r_1}$ [7].

Note that, the sequence of times $t_{r_1}, t_{r_2}, \cdots, t_{r_w}$ must be non-decreasing but it is possible to consider repeated times, i.e. $r_1 < r_2 = r_3 < r_4$ meaning two edges are followed at time t_{r_2}, and not required consecutive times, i.e. if $r_2 > r_1 + 1$ then there is no walk between t_{r_1} and t_{r_2} times.

In this dynamic setting, authors used the same concept that were used to derive the Katz centrality measure (3.1). Using down-weighting walks of length w by a factor a^w on the dynamic sequence, the following matrix product was obtained.

$$Q := (I - aA^{[0]})^{-1}(I - aA^{[1]})^{-1} \cdots (I - aA^{[M]})^{-1} \tag{3.2}$$

where, $a < 1/\max_s \rho(A^{[s]})$. Note that, the identity matrices in (3.2) explained the concept of waiting time of a message at a node until a suitable connection appears at a later time.

So, the centrality measures for the nth node were obtained from Q_{ij}, which explains how well information can be passed from node i to node j, by row and column sums

$$C_n^(broadcast) := \sum_{k=1}^{N} Q_{nk} \text{ and } C_n^(receive) := \sum_{k=1}^{N} Q_{kn} \tag{3.3}$$

and represented by how effectively node n can broadcast and receive messages, respectively.

The normalized Q, to avoid under or overflow and to have relative values of the centrality measures across all nodes, can be obtained by

$$\hat{Q}^{[k]} = \frac{\hat{Q}^{[k-1]}(I - aA^{[k]})^{-1})}{\left\| \hat{Q}^{[k-1]}(I - aA^{[k]})^{-1} \right\|}, k = 0, 1, 2, , M$$

where, $\hat{Q}^{[-1]} = I$. This technique bears two features- (i) computations are solution of linear system equations which is convenient and efficient for large, sparse networks, and (ii) the inherent asymmetry caused by the dynamics is captured directly through the non-commutativity of matrix multiplication.

3.4 Computational Tests

In this section, authors illustrated the new dynamic centralities with some simulated and real data.

3.4.1 Synthetic Data

To compare the dynamic centrality (3.3) with Katz centrality (3.1), simulated networks were generated for $N = 1001$ nodes with 31 time points. In this simulated networks 1000 nodes were constructed independently and node 1001 was connected to the two nodes with largest degree. Figure 2 showed that dynamic centrality correctly identified the importance of node 1001 that was absent for the Katz centrality for different parameters.

3.4.2 Telecommunication Data

Here telecommunication data [17] was used to illustrate the new dynamic centrality technique. Figure 3 showed a summary of the adjacency matrices aggregated into 28 day intervals. Figure 4 showed the daily edge count, and centrality measures with respect to broadcast, receiver and total degree. Figure 5 examined how the centralities changes to the change of parameter a.

3.4.3 Email Data

The new dynamic centrality technique was also illustrated on a public domain data set concerning email activities of Enron employees [18]. Figure 6 showed the daily edge count, scatter plots with respect to broadcast vs receiver centralities, broadcast vs total out degree and receive vs total in degree. It was found that the two new centrality measures were distinct; in particular, only two nodes appeared in the overlap of top twenty broadcast and receive and it was clear that some top receivers were very poor broadcasters. Figure 7 showed how the new centralities change with changes of a, indicated robustness in this parameter regime and also showed the eect of symmetrizing the data.

3.5 Discussion

The new centrality measures proposed in this article can be used to monitor network behavior dynamically. Based on the previous data, it is possible to obtain the expected future communicability using Markovian concept. The expected value of the future adjacency matrix for a given H_p, where $H_p = A^{[p]}, A^{[p-1]}, \cdots$, can be obtained by $E(A^{[p']} \mid H_p)$, where $p' > p$. Then the estimated expectation of the communicability over the current and future time steps can be obtained by-

$$E(S(Q) \mid H_0) = I + a\sum_{p=0}^{M} E(A^{[p']} \mid H_0) + O(a^2)$$

and

$$E(AS(Q) \mid H_0) = a^2 \sum_{p=0}^{M} \sum_{p'=p+1}^{M} E([A^{[p]}, A^{[p']}] \mid H_0) + O(a^3)$$

For $p \to \infty$, it is found that $A_{ij}^{[\infty]} = {\alpha_{ij}}/{(\alpha_{ij}+w_{ij})}$, where α_{ij} and w_{ij} denoted as the stepwise *birth* and *death* rates. Then, for considering time steps 0 to M, it was found that

$$E(Q \mid A^{[0]}) = I + a(R_M \circ (A^{[0]} - A^{[\infty]}) + (M+1)A^{[\infty]} + O(a^2)$$

where R_M is the symmetric matrix given by $(R_p)_{ij} = {(1-(1-\alpha_{ij}+w_{ij})^{M+1})}/{(\alpha_{ij}+w_{ij})}$ and $\circ$ denotes component-wise multiplication. This is nothing but the limiting value of the Markov chain that explains the relative contributions to Q made by the initial condition and the long term expected equilibrium value for each edge.

So if the observer wants to take some action on the future state, it is possible to predict long term expected communication based on the current state of network.

Chapter 4

Review of "Temporal Networks by Holm, P., and Saramäki J."

A graph consists of set of vertices, the units of the system, and a set of edges, the pairs of vertices that are interacting with each other that represents the network for example disease spreading on social networks. This type of networks helps us to understand how much one part of the network influences another or which vertices play similar roles in the systems operation etc. Now, if we can integrate another dimension, "Time", then this type of networks called temporal networks. In static networks, whether directed or not, it is considered that if the nodes follow transitivity property i.e. $A \to B$ and $B \to C$, implies $A \to C$ (indirectly through path B). However, in temporal networks, if the edge (A, B) is active only at a later point in time than the edge (B, C), then A and C are disconnected. In traditional network modeling one separates the underlying static network and the dynamical system on the network. On the other hand, temporal network methods consider when things happen from the dynamical system to the network. There are different types of temporal networks including Person-to-person communication, One-to-many information dissemination, Physical proximity, Cell biology, Distributed computing, Infrastructural networks, Neural and brain networks, Ecological networks etc.

There are two classes of representations for the temporal networks. In the first class includes vertices interacting with each other at certain times and the durations of the interactions are negligible e.g. emails, phone calls, text messages etc. In the second class, edges are active over a set of intervals instead of set of times e.g. proximity networks, seasonal food webs, infrastructural systems like internet.

There are lot of measures for topological structure of a static networks based on connections between neighboring nodes or between larger sets of nodes. However, for the temporal networks, it is important to incorporate the concept of time. These topological structure measures can be grouped based on as (i) Time-respecting paths and reachability, (ii) Time-respecting paths with limits on waiting times, (iii) connectivity and components, (iv) Distances, latencies and fastest paths, (iv) Average latency, (v) Diameter and network efficiency, (vi) minimum spanning tree, (vii) Centrality measures. The models that is used for the temporal networks broadly classified as temporal exponential random graphs, models of social group dynamics and randomized reference models [19].

Bibliography

[1] E. Forsyth and L. Katz, "A matrix approach to the analysis of sociometric data: preliminary report," *Sociometry*, vol. 9, no. 4, pp. 340–347, 1946.

[2] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.

[3] A. V. Mantzaris and D. J. Higham, "A model for dynamic communicators," *European Journal of Applied Mathematics*, vol. 23, no. 06, pp. 659–668, 2012.

[4] P. Grindrod, M. C. Parsons, D. J. Higham, and E. Estrada, "Communicability across evolving networks," *Physical Review E*, vol. 83, no. 4, p. 046120, 2011.

[5] N. Eagle, A. S. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the national academy of sciences*, vol. 106, no. 36, pp. 15274–15278, 2009.

[6] L. Muchnik, R. Itzhack, S. Solomon, and Y. Louzoun, "Self-emergence of knowledge trees: Extraction of the wikipedia hierarchies," *Physical Review E*, vol. 76, no. 1, p. 016106, 2007.

[7] P. Grindrod, M. C. Parsons, D. J. Higham, and E. Estrada, "Communicability across evolving networks," *Physical Review E*, vol. 83, no. 4, p. 046120, 2011.

[8] J. Tang, S. Scellato, M. Musolesi, C. Mascolo, and V. Latora, "Small-world behavior in time-varying graphs," *Physical Review E*, vol. 81, no. 5, p. 055101, 2010.

[9] J. Leskovec and E. Horvitz, "Planetary-scale views on a large instant-messaging network," in *Proceedings of the 17th international conference on World Wide Web*, pp. 915–924, ACM, 2008.

[10] A. Gautreau, A. Barrat, and M. Barthélemy, "Microdynamics in stationary complex networks," *Proceedings of the National Academy of Sciences*, vol. 106, no. 22, pp. 8847–8852, 2009.

[11] K. A. Berman, "Vulnerability of scheduled networks and a generalization of menger's theorem," *Networks*, vol. 28, no. 3, pp. 125–134, 1996.

[12] L. McNamara, C. Mascolo, and L. Capra, "Media sharing based on colocation prediction in urban transport," in *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pp. 58–69, ACM, 2008.

[13] P. Grindrod and D. J. Higham, "Evolving graphs: dynamical models, inverse problems and propagation," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 466, pp. 753–770, The Royal Society, 2010.

[14] J. Tang, M. Musolesi, C. Mascolo, and V. Latora, "Characterising temporal distance and reachability in mobile and online social networks," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 118–124, 2010.

[15] J. Tang, M. Musolesi, C. Mascolo, and V. Latora, "Temporal distance metrics for social network analysis," in *Proceedings of the 2nd ACM workshop on Online social networks*, pp. 31–36, ACM, 2009.

[16] J. Tang, M. Musolesi, C. Mascolo, V. Latora, and V. Nicosia, "Analysing information flows and key mediators through temporal centrality metrics," in *Proceedings of the 3rd Workshop on Social Network Systems*, p. 3, ACM, 2010.

[17] N. Eagle, A. S. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the national academy of sciences*, vol. 106, no. 36, pp. 15274–15278, 2009.

[18] A. Chapanond, M. S. Krishnamoorthy, and B. Yener, "Graph theoretic and spectral analysis of enron email data," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 265–281, 2005.

[19] P. Holme and J. Saramäki, “Temporal networks,” *Physics reports*, vol. 519, no. 3, pp. 97–125, 2012.

Appendix: Python Code

```
\fancyhead[CO,CE]{---Draft---}
############################################
# -*- coding: utf-8 -*-
"""
Created on Wed Jan 25 20:30:45 2017

@author: Kanak
"""
import numpy as np
data = np.loadtxt('D:/UCF/STA␣6908␣-␣Alexander␣V.␣Mantzaris/Data/3␣days␣
    network␣data␣from␣Alexander␣V.␣Mantzaris.txt')

print(data.shape)

# convert into 3D array
data = data.reshape((3,21,21))

print(data[2][9])


############################################
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 19 16:28:08 2017

@author: ka746940
"""

import numpy as np

size_n = 5 # sample size of each timestamp
high_n = 3 # number of node
time_n = 4 # number of timestamp

# Generating data contains timestamp, sender number, receiver number

dt = []
for j in range(time_n):
    time = np.repeat(j, size_n)
    sender = np.random.randint(low = 0, high = high_n, size = size_n)
    receiver = np.random.randint(low = 0, high = high_n, size = size_n)

    for i in range(size_n):
        if sender[i] != receiver[i]:
            dt.append([time[i], sender[i], receiver[i]])
data = np.array(dt)


# generating #node x #node zero matrix for A for each timestamp (time_n)
A = np.zeros((time_n, high_n, high_n))

# if sender sends a message to receiver, assign 1 otherwise 0 for each
```

```
    timestamp

for row in data:
     A[row[0], row[1], row[2]] = 1

print(data)
print(A)

############################################

# -*- coding: utf-8 -*-
"""
Created on Sun Jan 22 21:51:15 2017

@author: Kanak
"""
import numpy as np
import scipy.io
mat = scipy.io.loadmat('D:/UCF/STA␣6908␣-␣Alexander␣V.␣Mantzaris/Data/
    enronAtensor.mat')
data = mat.get('enronAtensor')
with open('D:/UCF/STA␣6908␣-␣Alexander␣V.␣Mantzaris/Data/enronAtensor.txt', 'w
    ') as outfile:

     outfile.write('#␣Array␣shape:␣{0}\n'.format(data.shape))


     for data_slice in data:

          for row in data_slice:
               j = 0
               lengthr = len(row)
               for i in row:
                    j +=1
                    if j < lengthr:
                         outfile.write('%s,'%i)
                    else:
                         outfile.write('%s\n'%i)
               #outfile.write('\n')

          # Writing out a break to indicate different slices...
          outfile.write('#␣New␣slice\n')


# Read the array from disk
new_data = np.loadtxt('D:/UCF/STA␣6908␣-␣Alexander␣V.␣Mantzaris/Data/
    enronAtensor.txt')

print(new_data.shape)

# convert into 3D array
new_data = new_data.reshape((151,151,1137))

# compare two data set
```

```
assert np.all(new_data == data)

##########################################

# -*- coding: utf-8 -*-
"""
Created on Fri Feb 10 12:25:39 2017

@author: Kanak
"""

import numpy as np
import sys
sys.path.insert(0, 'D:/UCF/STA_6908_-_Alexander_V._Mantzaris/Python_Code/')
import Network_Centrality as cent

#
new_data = np.loadtxt('D:/UCF/STA_6908_-_Alexander_V._Mantzaris/Data/
    enronAtensor.txt', delimiter = ',')
nnode = 151
nday = 1137

# convert into 3D array
new_data = new_data.reshape((nnode, nnode, nday))

Atensor = np.zeros((nday, nnode,nnode), dtype=int)

dayspec = np.zeros(nday, dtype=float)
for k in range(nday):
    A = new_data[:,:, k]
    #ensure symmetric, binary, zero diag
    A = np.sign(A+np.transpose(A))
    A = A - np.diag(np.diag(A))
    Atensor[k,:,:] = A
    w, v = np.linalg.eig(A)
    dayspec[k] = np.max(np.abs(w))
del(w,v,k,A)

##################################################################

new_data = np.loadtxt('D:/UCF/STA_6908_-_Alexander_V._Mantzaris/Data/3_days_
    network_data_from_Alexander_V._Mantzaris.txt')
nnode = 21
nday = 3
new_data.shape

# convert into 3D array
new_data = new_data.reshape((nday, nnode, nnode))

Atensor = np.zeros((nday, nnode,nnode), dtype=int)

dayspec = np.zeros(nday, dtype=float)
for k in range(nday):
    A = new_data[k, :,:]
    #ensure symmetric, binary, zero diag
    A = np.sign(A+np.transpose(A))
    A = A - np.diag(np.diag(A))
    Atensor[k,:,:] = A
    w, v = np.linalg.eig(A)
    dayspec[k] = np.max(np.abs(w))
del(w,v,k,A)

cent.test_symmetric(Atensor)
xx = cent.create_sym(Atensor)

B = cent.grindrod(Atensor)
rankind = cent.dy_rank_degree(B)
print(rankind['sort_node_index'])
print(rankind['sort_node_ln_index'])


kz = cent.katz(np.sum(Atensor, 0), .015)
rkz = cent.kz_rank_degree(kz)
rkz

from matplotlib.dates import num2date
import datetime
convert = lambda x: datetime.datetime.fromtimestamp((x))
new_data = np.loadtxt('D:/UCF/STA 6908 - Alexander V. Mantzaris/Data/
    CollegeMsg.txt',
                        delimiter = ' ')
new_data[:,2]= []
nnode = 151
nday = 1137


##########################################

# -*- coding: utf-8 -*-
"""
Created on Mon Feb  6 15:41:58 2017

@author: Kanak
"""
import numpy as np

#####################################################
# Test symmetric and 0 diagonal matrix
#####################################################

def test_symmetric(x):
    if not isinstance(x, np.ndarray):
        try:
            x = np.array(x)
        except:
            print ("x is not possible to convert as array")
            return
    dimsize = x.shape
    if len(dimsize) == 3:
        nday = dimsize[0]
    else:
        if (x.transpose() == x).all():
            if (np.diag(x) == 0).all():
                print('Input is a symmetric matrix and diagonal elements equal
                    0')
                return(True)
            print('Input is a symmetric matrix but diagonal elements not equal
                0')
            return(False)
        else:
            print('Input is not a symmetric matrix')
            return(False)
    i = 0
    symm = True
    diagt = True
    while i < nday and symm:
        a = x[i,:,:]
        if (a.transpose() == a).all():
            if (np.diag(a) == 0).all() and diagt:
                diagt = True
            else:
                diagt = False
            symm = True
            i += 1
        else:
            symm = False
            i += 1
    if symm and diagt:
        print('Input is a symmetric matrix and diagonal elements equal 0')
        return(True)
    elif symm and not diagt:
        print('Input is a symmetric matrix but diagonal elements not equal 0')
        return(False)
    else:
        print('Input is not a symmetric matrix')
        return(False)

######################################################
# create binary symmetric and 0 diagonal matrix
######################################################


def create_sym(x):
    if not isinstance(x, np.ndarray):
        try:
            x = np.array(x)
        except:
            print ("x is not possible to convert as array")
            return

    dimsize = x.shape
    if len(dimsize) == 3:
        nday = dimsize[0]
        nnode = dimsize[1]
    else:
        #ensure symmetric, binary, zero diag
        x = np.sign(x + np.transpose(x))
        np.fill_diagonal(x, 0)
#         w, v = np.linalg.eig(x)
#         dayspec = np.max(np.abs(w))
#         return({'data': x, 'max_eigen':dayspec})
        return(x)


    Atensor = np.zeros((nday, nnode,nnode), dtype=int)
#     dayspec = np.zeros(nday, dtype=float)
    for k in range(nday):
        A = x[k,:,:]
        A = np.sign(A+np.transpose(A))
        np.fill_diagonal(A, 0)
        Atensor[k,:,:] = A
#         w, v = np.linalg.eig(A)
#         dayspec[k] = np.max(np.abs(w))

#     return({'data': Atensor, 'max_eigen':dayspec})
    return(Atensor)

##########################################################
# Dynamic centrality matrix
##########################################################

def grindrod(x,
             alpha = 0.1
             ):
    if not isinstance(x, np.ndarray):
        try:
            x = np.array(x)
        except:
            print ("x is not possible to convert as array")
            return
    dimention = x.shape
    if dimention[1] == dimention[2]:
        nday = dimention[0]
        nnode = dimention[1]
    else:
        print ("Input data is not Symmetric")
        return

    Adeg = np.zeros(nday, dtype=float)
    B = np.identity(nnode)
    I = np.identity(nnode)
    for k in reversed(range(nday)):
        Aday = x[k,:,:]
        Adeg[k] = np.sum(Aday, axis=(0,1))/2.0
        B = np.dot(np.linalg.inv(I - alpha*Aday),B)
        B = abs(B)
        B = B/np.linalg.norm(B)
    print('A. V. Mantzaris and D. J. Higham,   A model for dynamic 
        communicators,      European Journal of Applied Mathematics, vol. 23, no
        . 06, pp. 659  668 , 2012.')
    return(B)



######################################################
# Rank based on dynamic centrality matrix
######################################################

def dy_rank_degree(x, ninfnode = None):
    broadcast = np.sum(x, axis = 1)
    receive = np.sum(x, axis = 0)
    lnbroadcast = np.log(broadcast)
    lnreceive = np.log(receive)
    sortedQrankindices = (broadcast - receive).argsort()[::-1]
    sortedQrankLNindices = (lnbroadcast - lnreceive).argsort()[::-1]
    s_broadcast = broadcast[sortedQrankindices]
    s_receive = receive[sortedQrankindices]
    s_lnbroadcast = lnbroadcast[sortedQrankLNindices]
    s_lnreceive = lnreceive[sortedQrankLNindices]
    if ninfnode == None:
        ninfnode = x.shape[0]
    return{'broadcase': broadcast, 'receive': receive,
           'ln_broadcase': lnbroadcast, 'ln_receive': lnreceive,
           'sort_broadcase': s_broadcast[:ninfnode],
           'sort_receive': s_receive[:ninfnode],
           'sort_ln_broadcase': s_lnbroadcast[:ninfnode],
           'sort_ln_receive': s_lnreceive[:ninfnode],
           'sort_node_index': sortedQrankindices[:ninfnode],
           'sort_node_ln_index': sortedQrankLNindices[:ninfnode]}


######################################################
# Katz centrality matrix
######################################################

def katz(x,
         alpha = None
         ):
    if not isinstance(x, np.ndarray):
        try:
            x = np.array(x)
        except:
            print ("x is not possible to convert as array")
            return
    dimention = x.shape
    if dimention[0] == dimention[1]:
        N = dimention[0]
    else:
        print ("Input data is not Symmetric")
        return
    if alpha == None:
        w,v = np.linalg.eig(x)
        alpha = np.round_(1/max(np.abs(w)), decimals = 3) #%gives 0.0196
    return(np.linalg.inv(np.identity(N) - alpha*x))


##############################################################
# Rank based on katz centrality matrix
##############################################################


def kz_rank_degree(x, ninfnode = None):
    centrality = np.sum(x,axis = 0)
    ln_centrality = np.log(centrality)
    sortedQrankindices_k = centrality.argsort()[::-1]
    sortedQrankLNindices_k = (ln_centrality).argsort()[::-1]
    Centrality_k = centrality[sortedQrankindices_k]
    Centrality_ln_k = ln_centrality[sortedQrankLNindices_k]
    if ninfnode == None:
        ninfnode = x.shape[0]

    return{'centrality': centrality, 'ln_centrality': ln_centrality,
           'sort_centrality': Centrality_k[:ninfnode],
           'sort_ln_centrality': Centrality_ln_k[:ninfnode],
           'sort_node_index': sortedQrankindices_k[:ninfnode],
           'sort_node_ln_index': sortedQrankLNindices_k[:ninfnode]}


##############################################################
# time weighted version of the temporal centrality
##############################################################

# %now compute the dynamic matrix itereation

Acells = np.loadtxt('D:/UCF/STA_6908_-_Alexander_V._Mantzaris/Data/3_days_
    network_data_from_Alexander_V._Mantzaris.txt')
node_num = 21
nday = 3
alpha = 0.1

# convert into 3D array
Acells = Acells.reshape((nday, node_num, node_num))


eyeM = np.identity(node_num, dtype=float)
S_iters = np.zeros((nday, node_num, node_num), dtype=float)
S_iters_temp = np.dot((eyeM + np.exp(-0.2)*0),(np.linalg.inv(eyeM - alpha*0)))
    - eyeM
S_iters_Broadcast = np.zeros((nday, node_num), dtype=float);
for ii in range(nday):
    Aday = Acells[ii,:,:];
    S_iters[ii,:,:] = np.dot((eyeM + np.exp(-0.2)*S_iters_temp),(np.linalg.inv
        (eyeM - alpha*Aday))) - eyeM
    S_iters_temp = S_iters[ii,:,:];
    S_iters_Broadcast[ii,:] = np.sum(S_iters[ii, :,:],axis = 1);



S_last = S_iters[ii,:,:];
S_last_Broadcast = np.sum(S_last,axis = 1);
rankIndexesS_last_Broadcast = S_last_Broadcast.argsort()[::-1]
S_last_Broadcast_sort = S_last_Broadcast[rankIndexesS_last_Broadcast]
S_last_Broadcast_rank = rankIndexesS_last_Broadcast[0:4]



##########################################

# -*- coding: utf-8 -*-
"""
Created on Sun Jan 29 19:28:20 2017

@author: Kanak
"""
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
nnode = 151
nday = 1137
new_data = np.loadtxt('D:/UCF/STA_6908 - Alexander V. Mantzaris/Data/
    enronAtensor.txt')
new_data.shape
new_data = new_data.reshape((nday, nnode, nnode))
new_data = new_data.astype(int)


Atensor = np.zeros((nday, nnode,nnode), dtype=int)

dayspec = np.zeros(nday, dtype=float)
for k in range(nday):
    A = new_data[:,:,k]
    #ensure symmetric, binary, zero diag
    A = np.sign(A+np.transpose(A))
    A = A - np.diag(np.diag(A))
    Atensor[k,:,:] = A
    w, v = np.linalg.eig(A)
    dayspec[k] = np.max(np.abs(w))

######################################
# Dynamic Centrality
######################################


alpha = 0.1

Adeg = np.zeros(nday, dtype=float)
```

```
B = np.identity(nnode)
I = np.identity(nnode)
for k in reversed(range(nday)):
    Aday = Atensor[k,:,:]
    Adeg[k] = sum(sum(Aday))/2.0
    B = np.dot(np.linalg.inv(I - alpha*Aday),B)
    B = abs(B)
    B = B/np.linalg.norm(B)

Cbroad = np.sum(B, axis = 1)
Crec = np.sum(B, axis = 0)


#%!!!Alex
#%now do the ranking in terms of Hierarchy
sortedQrankindices = (Cbroad - Crec).argsort()[::-1]
sortedQrankLNindices = (np.log(Cbroad) - np.log(Crec)).argsort()[::-1]
Cbroad = Cbroad[sortedQrankLNindices]
Crec = Crec[sortedQrankLNindices]
print(Cbroad[1:10])
print(Crec[1:10])
print(sortedQrankindices +1, '\n', sortedQrankLNindices + 1)
#%end Alex!!!

Degtemp = np.sum(Atensor,axis = 2)
Degall = sum(Degtemp)

specmax = max(dayspec)
amax = 1/specmax

########################################
# %Do KATZ on binarized aggregate
########################################

Astat = np.real(np.sum(Atensor,0)>0)
N = len(Astat)
w,v = np.linalg.eig(Astat)
amaxstar = 1/max(np.abs(w)) #%gives 0.0196
alph = 0.015
katz = np.linalg.inv(np.identity(N) - alph*Astat)
katz_sum = np.sum(katz,axis = 0)

katz_sum.shape
cbk = np.corrcoef(np.transpose(Cbroad), katz_sum)[0,1]
kenbk = stats.kendalltau(np.transpose(Cbroad),katz_sum)[0]
crk = np.corrcoef(np.transpose(Crec), katz_sum)[0,1]
kenrk = stats.kendalltau(np.transpose(Crec),katz_sum)[0]



sortedQrankindices_k = katz_sum.argsort()[::-1]
sortedQrankLNindices_k = np.log(katz_sum).argsort()[::-1]
Centrality_k = katz_sum[sortedQrankLNindices_k]
print(Centrality_k[1:10])
print(sortedQrankindices_k + 1,'\n', sortedQrankLNindices_k + 1)
# end
########################################



########################################
# Figures
########################################


#figure(1)
#subplot(221)
Act1 = np.sum(Atensor, axis = 2)
Actall = np.sum(Act1, axis = 2)
plt.plot(Actall,'.')
plt.xlabel('Day')
plt.ylabel('Total Activity')
plt.title('gca')
plt.show()

#subplot(222)
plt.loglog(Cbroad, Crec, '.', basex=np.e, basey=np.e)
plt.xlim([1e-10,1e3])
plt.ylim([1e-15,1e3])
plt.xlabel('Broadcast')
plt.ylabel('Receive')
plt.title('gca')
plt.show()

#subplot(223)
plt.loglog(Cbroad, Degall, '.', basex=np.e, basey=np.e)
plt.xlim([1e-10,1e3])
plt.ylim([1e0,1e4])
plt.xlabel('Broadcast')
plt.ylabel('Total Degree')
plt.title('gca')
plt.show()

#subplot(224)

plt.loglog(Crec, Degall, '.', basex=np.e, basey=np.e)
plt.xlim([1e-10,1e3])
plt.ylim([1e0,1e4])
plt.xlabel('Receive')
plt.ylabel('Total Degree')
plt.title('gca')
plt.show()

###########################################

# -*- coding: utf-8 -*-
"""
```

```
Created on Sun Feb 26 11:32:15 2017

@author: Kanak
"""
import numpy as np

Acells = np.loadtxt('D:/UCF/STA␣6908␣-␣Alexander␣V.␣Mantzaris/Data/3␣days␣
    network␣data␣from␣Alexander␣V.␣Mantzaris.txt')
node_num = 21
nday = 3
alpha = 0.1

# convert into 3D array
Acells = Acells.reshape((nday, node_num, node_num))


eyeM = np.identity(node_num, dtype=float)
S_iters = np.zeros((nday, node_num, node_num), dtype=float)
S_iters_temp = np.dot((eyeM + np.exp(-0.2)*0),(np.linalg.inv(eyeM - alpha*0)))
     - eyeM
S_iters_Broadcast = np.zeros((nday, node_num), dtype=float);
for ii in range(nday):
    Aday = Acells[ii,:,:];
    S_iters[ii,:,:] = np.dot((eyeM + np.exp(-0.2)*S_iters_temp),(np.linalg.inv
        (eyeM - alpha*Aday))) - eyeM
    S_iters_temp = S_iters[ii,:,:];
    S_iters_Broadcast[ii,:] = np.sum(S_iters[ii, :,:],axis = 1);



S_last = S_iters[ii,:,:];
S_last_Broadcast = np.sum(S_last,axis = 1);
rankIndexesS_last_Broadcast = S_last_Broadcast.argsort()[::-1]
S_last_Broadcast_sort = S_last_Broadcast[rankIndexesS_last_Broadcast]
S_last_Broadcast_rank = rankIndexesS_last_Broadcast[0:4]

###########################################
```