

Social Networks

Crash Course



Giovanna Miritello

 @gmiritello



1. Introduction on Social Networks and SNA

- what are Social Networks
- what is Social Network Analysis (SNA)
- what do we want to know?
- why and when to use SNA

2. Social Networks: from properties to information

- heterogeneity
- 6 degrees of separation
- clustering and density
- assortativity. social affinity..or social contagion?
- groups and communities, the strength of weak ties and Dunbar`s number
- two more dimensions: time and space
- from Social Networks to Social Network Analysis

3. Practical Examples and Use Cases (IP[y]: IPython Interactive Computing)

3.1 Identify key players

3.2 Link Persistence

3.3 Community Detection

*each session organized as follows:

- a) Question/Problem
- b) Analysis and Insights
- c) Interpretation of results and Applications

4. Tools for social network analysis and visualization

5. Interesting references and sources

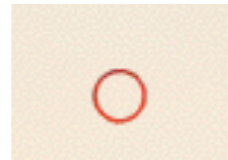


What are Social Networks

Components

nodes

(people, organizations)



What are Social Networks

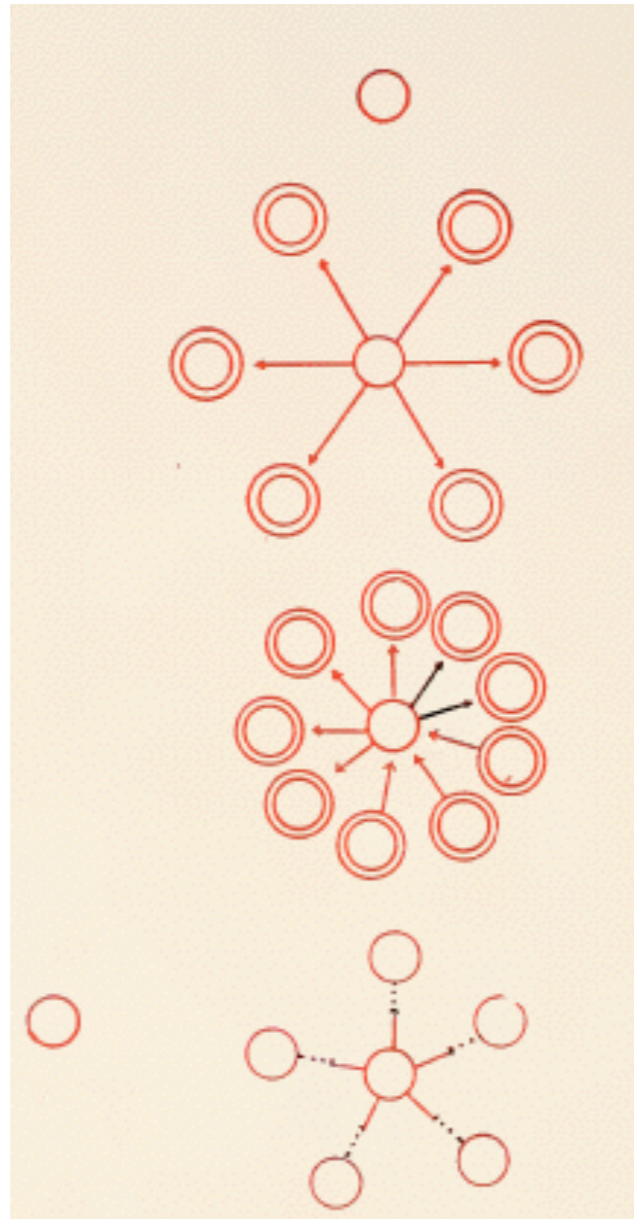
Components

nodes

(people, organizations)

links

- relationships (friend, family, work,..)
- communication (f2f, mobile phone, email,..)
- sexual relationships



What are Social Networks

Components

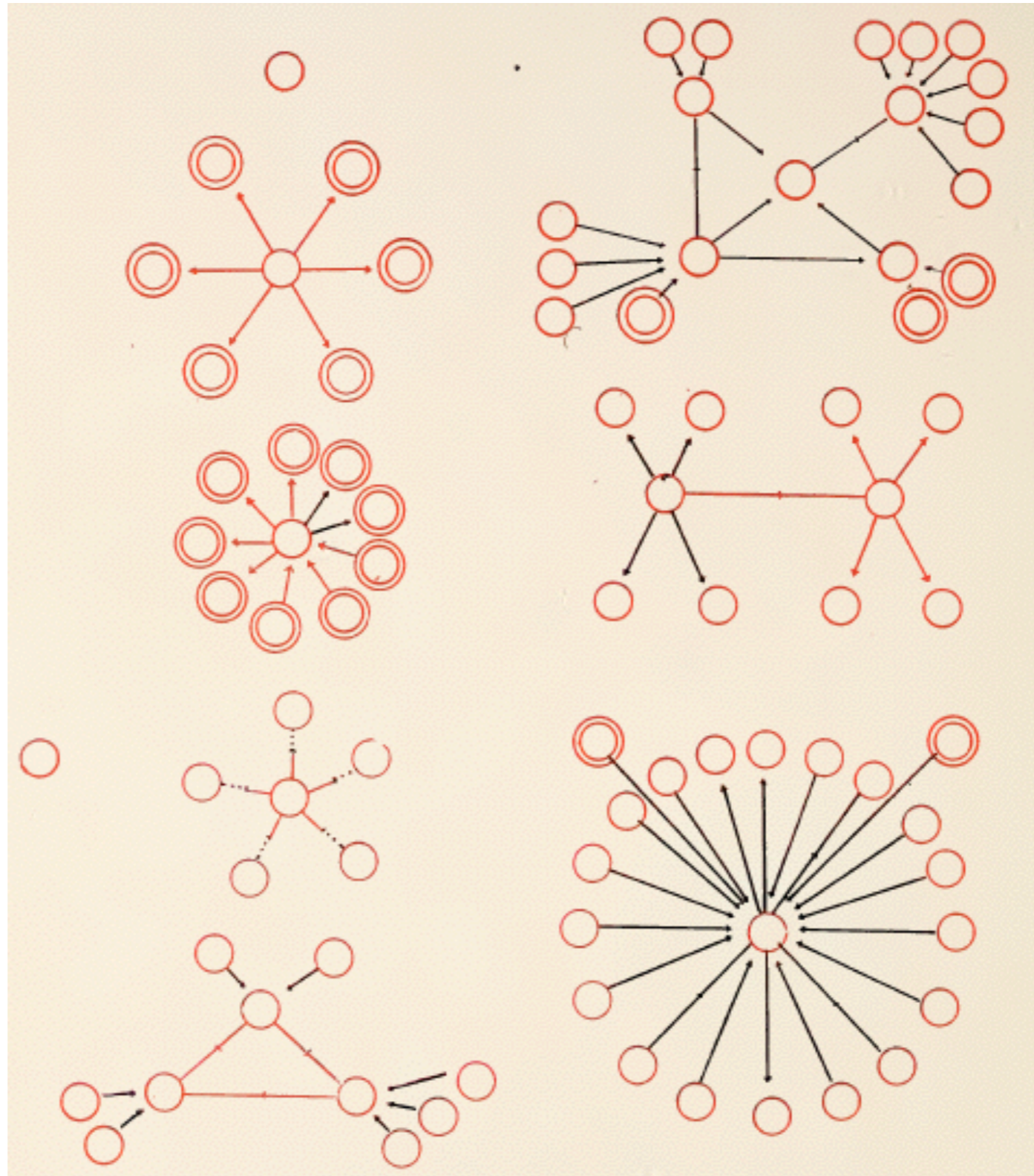
nodes

(people, organizations)

links

- relationships (friend, family, work,...)
- communication (f2f, mobile phone, email,..)
- sexual relationships

triangles communities



What is Social Network Analysis (SNA)

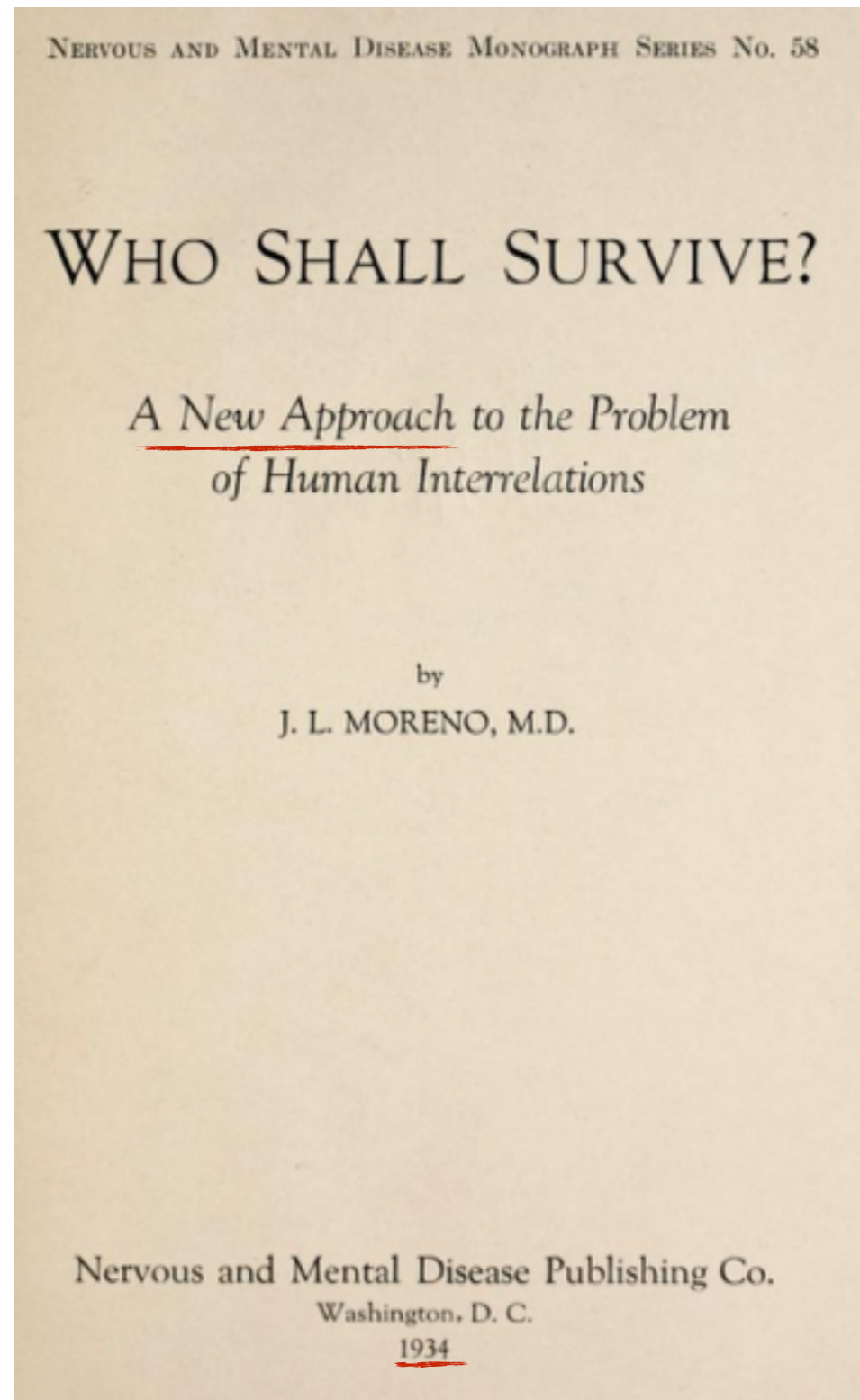
Durkheim and Tönnies, 1890s. Theory of social groups

Moreno, 1930. First sociogram to study personal relationships group of elementary school students

“The boys were friends of boys and the girls were friends of girls with the exception of one boy who said he liked a single girl. The feeling was not reciprocated

1950s. Max Gluckman et al. community networks in south Africa, India and UK

1960 Peter Blau. Analysis of relational ties



1963 Stanley Milgram Six degrees of separation and “small world experiment”

1970s Harrison White Models of social structure based on patterns of relations instead than attributes

1970s Charles Tilly Networks in politics and social movements

(1969) 1973 Mark Granovetter The strength of weak ties

What is Social Network Analysis (SNA)

Today The social network of superheroes



Marvel Universe looks almost like a real social network

R. Alberich, J. Miro-Julia, F. Rosselló

*Departament de Matemàtiques i Informàtica, Universitat de les Illes Balears,
07071 Palma de Mallorca (Spain)*

{ricardo,joe,cesc}@ipc4.uib.es

Abstract

We investigate the structure of the Marvel Universe collaboration network, where two Marvel characters are considered linked if they jointly appear in the same Marvel comic book. We show that this network is clearly not a random network, and that it has most, but not all, characteristics of “real-life” collaboration networks, such as movie actors or scientific collaboration networks. The study of this artificial universe that tries to look like a real one, helps to understand that there are underlying principles that make real-life networks have definite characteristics.

What is Social Network Analysis (SNA)



What is Social Network Analysis (SNA)

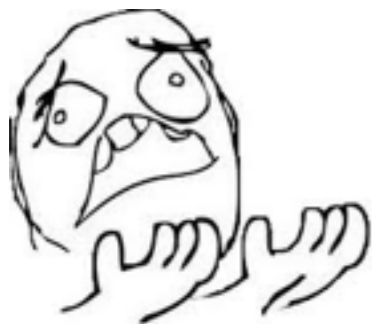


SNA is the study of human interaction using network theory. Although the origin of SNA dates back at the early 20th century, the interest in this field is growing in the last years also thanks to online social networks and data availability.



SNA consists in the formulation and solution of problems that can be represented with a network structure.

SNA has its origins in both social science and in the fields of network analysis and graph theory.

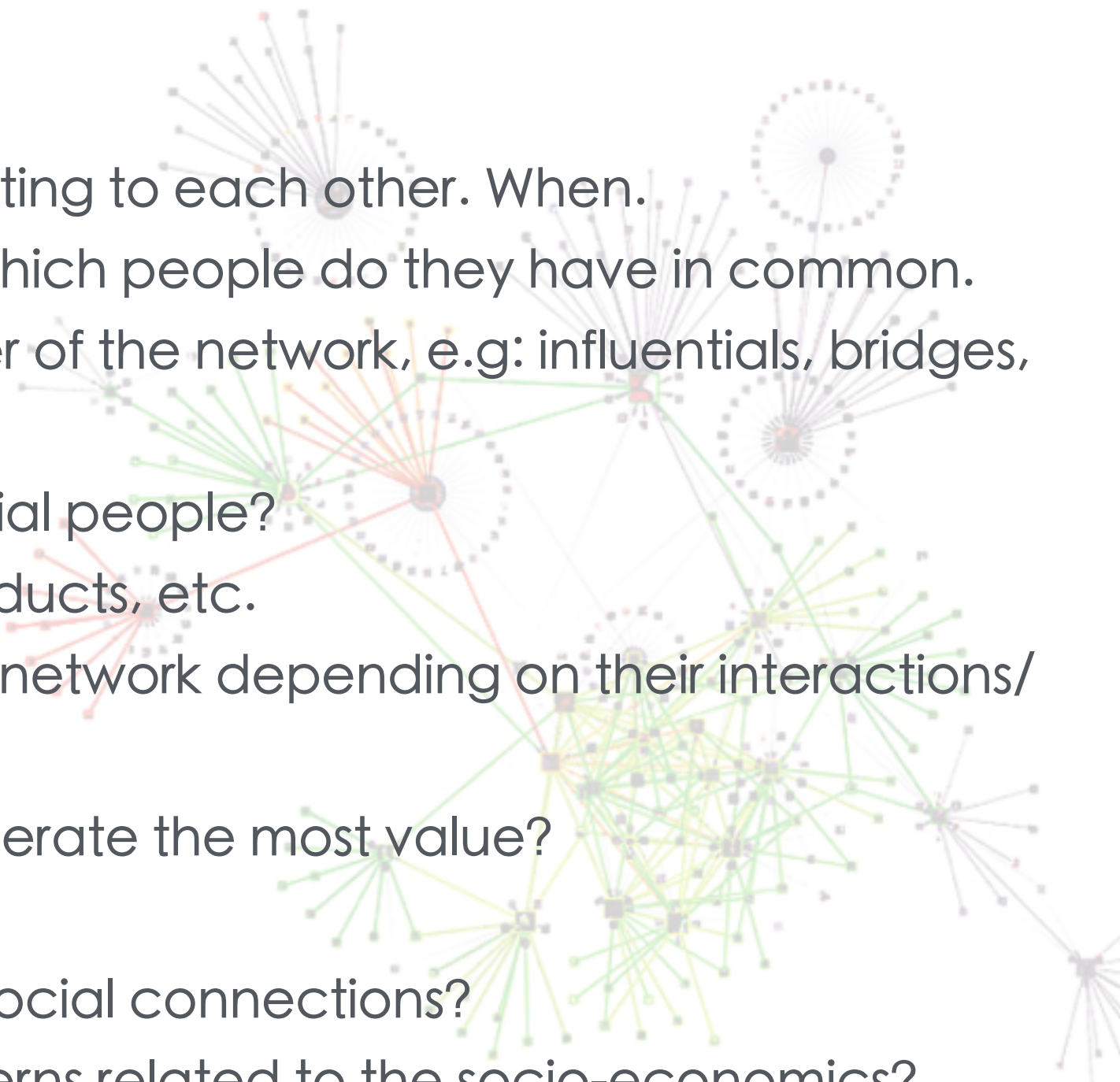


SNA is not just a methodology, it is a perspective on how society functions. Instead of focusing on individuals and their properties, it centers on relations between individuals, groups and the society as a whole.

Networks have always existed but the emergence of data, information and technology has made them evident and traceable.

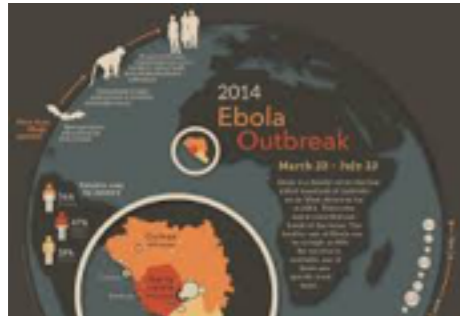


What do we want to know ?

- 
- ✓ How frequently people are interacting to each other. When.
 - ✓ Who is connected to whom and which people do they have in common.
 - ✓ Role and function of each member of the network, e.g: influentials, bridges, early adopters.
 - ✓ Who are the most popular/influential people?
 - ✓ Opinion and interest in events, products, etc.
 - ✓ How can we classify people in the network depending on their interactions/behavior/habits?
 - ✓ Which social connections can generate the most value?
 - ✓ How do people move and travel?
 - ✓ How geographic distance affect social connections?
 - ✓ Are temporal and/or mobility patterns related to the socio-economics?

Why and when to use SNA

Health/ Epidemic spreading



Communication



Smart cities



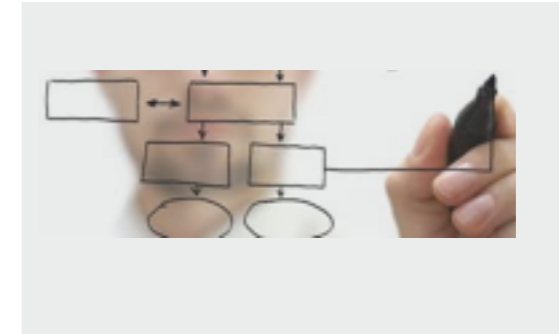
Fraud and risk



Marketing/Opinions



Management Research



Logistics and Optimization



Recommendation Systems



Cybersecurity



The properties of social networks

Member properties are very heterogeneous

Emergence of Scaling in Random Networks

Albert-László Barabási* and Réka Albert

Systems as diverse as genetic networks or the World Wide Web are best described as networks with complex topology. A common property of many large networks is that the vertex connectivities follow a scale-free power-law distribution. This feature was found to be a consequence of two generic mechanisms: (i) networks expand continuously by the addition of new vertices, and (ii) new vertices attach preferentially to sites that are already well connected. A model based on these two ingredients reproduces the observed stationary scale-free distributions, which indicates that the development of large networks is governed by robust self-organizing phenomena that go beyond the particulars of the individual systems.

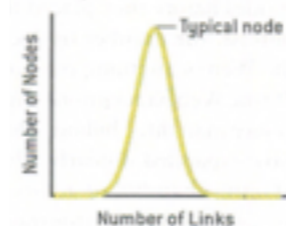
Random Network



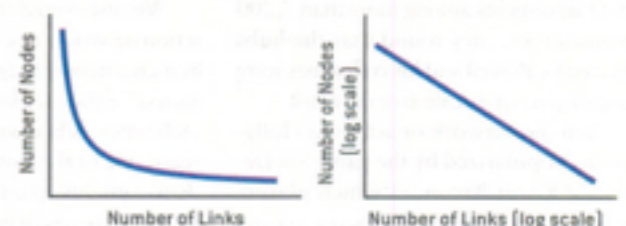
Scale-Free Network



Bell Curve Distribution of Node Linkages



Power Law Distribution of Node Linkages



The properties of social networks

Small-world (short path between any two nodes)

The Small-World Problem

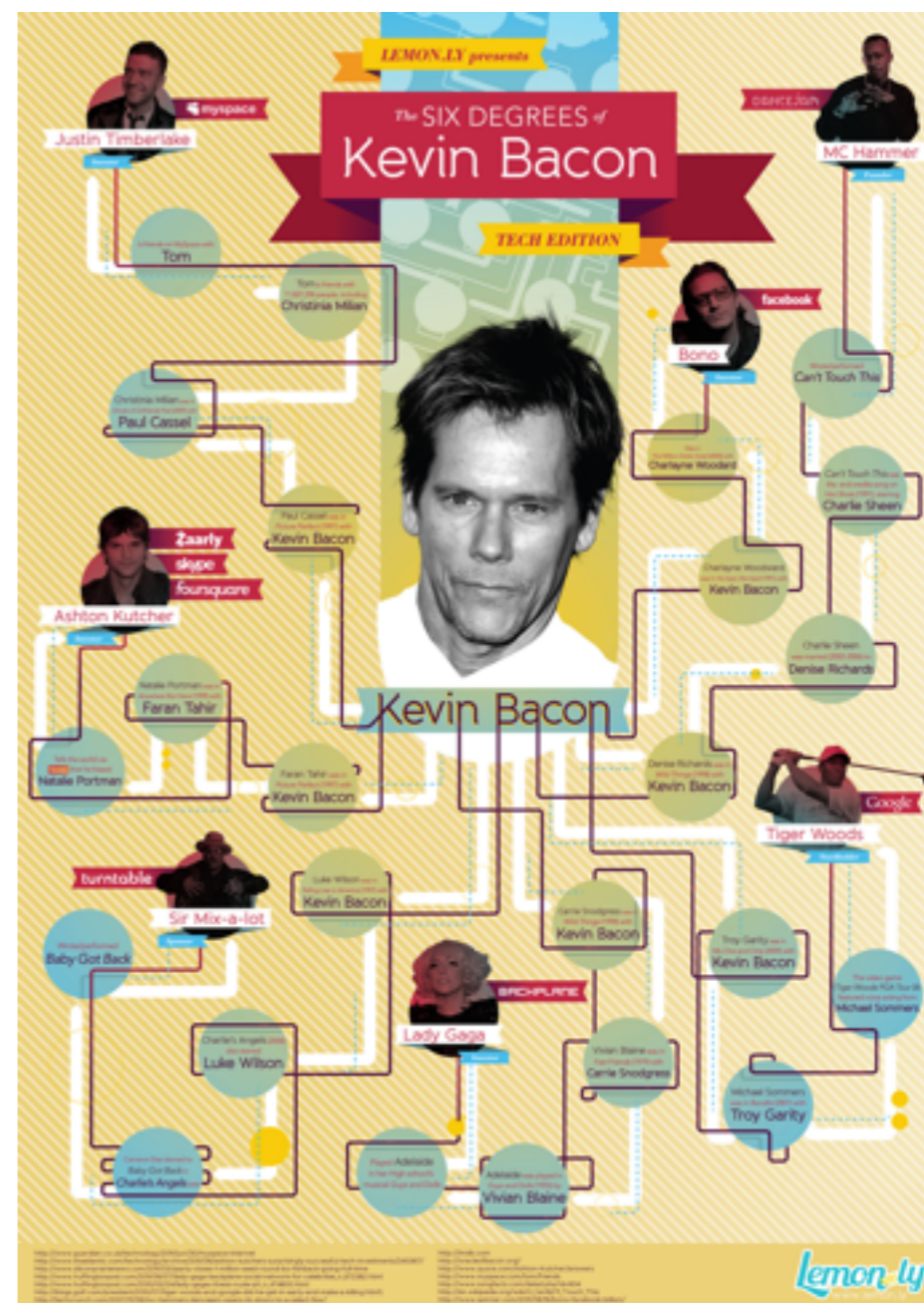
By Stanley Milgram

Fred Jones of Peoria, sitting in a sidewalk cafe in Tunis, and needing a light for his cigarette, asks the man at the next table for a match. They fall into conversation; the stranger is an Englishman who, it turns out, spent several months in Detroit studying the operation of an interchangeable-bottlecap-factory. "I know it's a foolish question," says Jones, "but did you ever by any chance run into a fellow named Ben Arkadian? He's an old friend of mine, manages a chain of supermarkets in Detroit . . ."

"Arkadian, Arkadian," the Englishman mutters. "Why, upon my soul, I believe I do! Small chap, very energetic, raised merry hell with the factory over a shipment of defective bottlecaps."

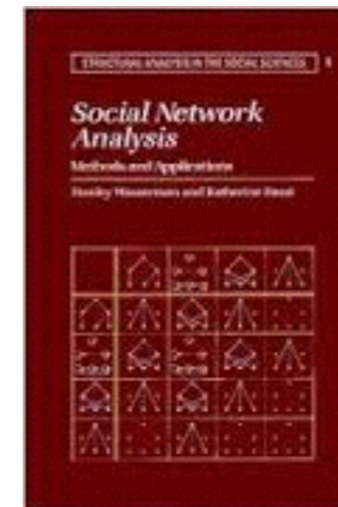
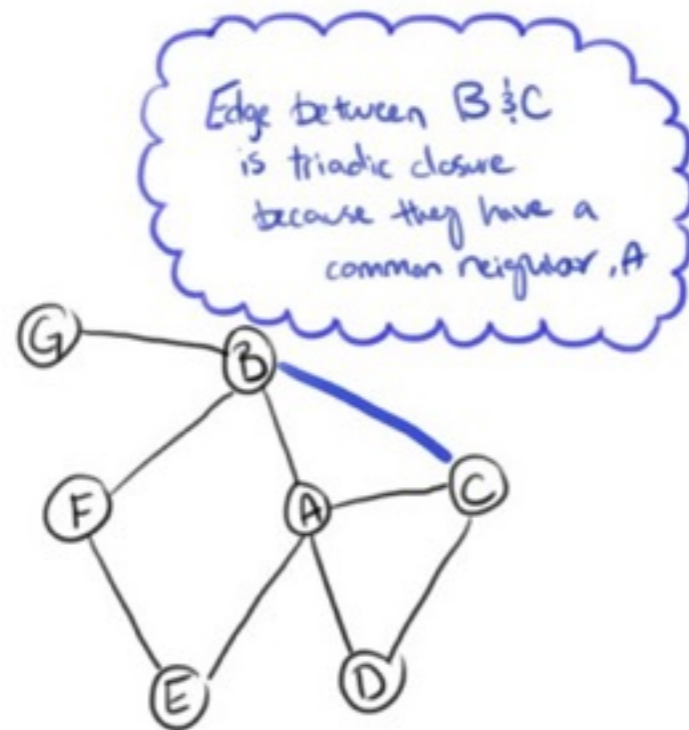
"No kidding!" Jones exclaims in amazement.

"Good lord, it's a small world, isn't it?"



The properties of social networks

High clustering (very connected groups with high density of ties)



Collective dynamics of 'small-world' networks

Duncan J. Watts² & Steven H. Strogatz¹

1. Department of Theoretical and Applied Mechanics, Kimball Hall, Cornell University, Ithaca, New York 14853, USA

2. Present address: Paul F. Lazarsfeld Center for the Social Sciences, Columbia University, 812 SIPA Building, 420 W118 St, New York, New York 10027, USA.

The properties of social networks

Social networks are assortative: homophily/affinity..

BIRDS OF A FEATHER: Homophily in Social Networks

Miller McPherson¹, Lynn Smith-Lovin¹, and
James M Cook²

¹Department of Sociology, University of Arizona, Tucson, Arizona 85721;
e-mail: mcperson@u.arizona.edu; smithlov@u.arizona.edu

²Department of Sociology, Duke University, Durham, North Carolina 27708;
e-mail: jcook@soc.duke.edu

- ✓ degree
- ✓ habits
- ✓ loneliness/happiness
- ✓ interests
- ✓ age/gender/class
- ✓ organizational roles
- ✓ values/education



The NEW ENGLAND
JOURNAL of MEDICINE

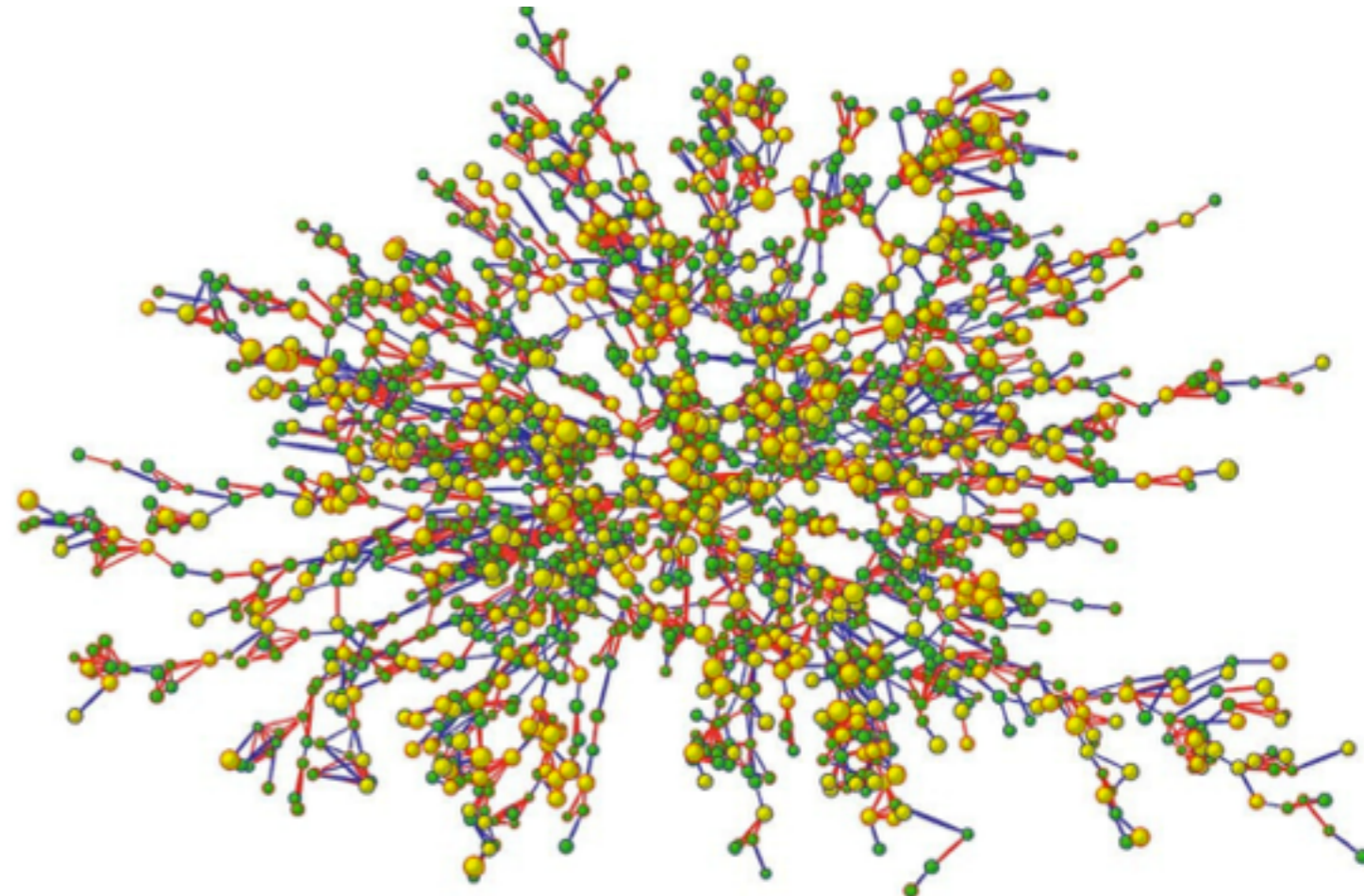
HOME ARTICLES & MULTIMEDIA ▾ ISSUES ▾ SPECIALTIES & TOPICS ▾ FOR AUTHORS ▾ CME ▾

SPECIAL ARTICLE

The Spread of Obesity in a Large Social Network over 32 Years

Nicholas A. Christakis, M.D., Ph.D., M.P.H., and James H. Fowler, Ph.D.

N Engl J Med 2007; 357:370-379 | July 26, 2007 | DOI: 10.1056/NEJMsa066082



The properties of social networks

..or social contagion?

Distinguishing influence-based contagion from homophily-driven diffusion in dynamic networks

Sinan Aral^{a,b,1}, Lev Muchnik^a, and Arun Sundararajan^a

^aInformation, Operations and Management Sciences Department, Stern School of Business, New York University, Kaufmann Management Center, 44 West 4th Street, New York, NY 10012; and ^bCenter for Digital Business, Sloan School of Management, Massachusetts Institute of Technology, 5 Cambridge Center-NE25, Cambridge, MA 02142

- ✓ degree
- ✓ habits
- ✓ loneliness/happiness
- ✓ interests
- ✓ age/gender/class
- ✓ organizational roles
- ✓ values/education



The properties of social networks

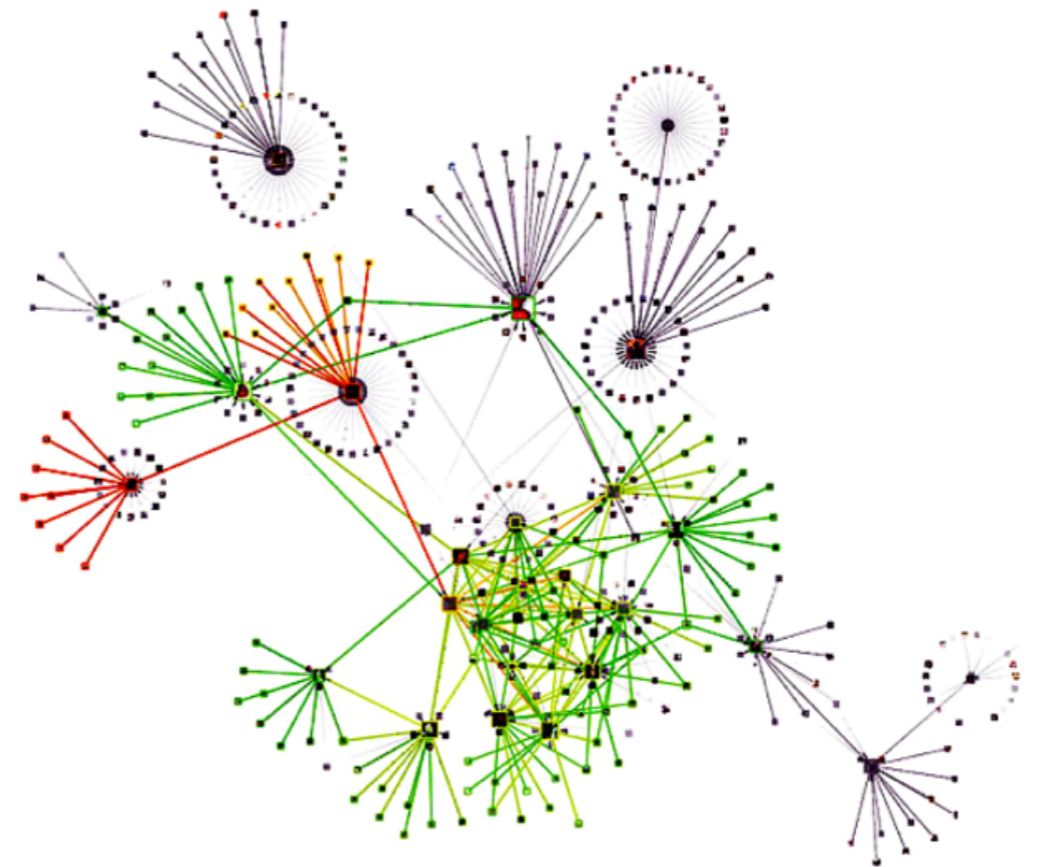
Well organized structure (high modularity)

Detecting community structure in networks

M. E. J. Newman

*Department of Physics and Center for the Study of Complex Systems,
University of Michigan, Ann Arbor, MI 48109-1120*

There has been considerable recent interest in algorithms for finding communities in networks—groups of vertices within which connections are dense, but between which connections are sparser. Here we review the progress that has been made towards this end. We begin by describing some traditional methods of community detection, such as spectral bisection, the Kernighan–Lin algorithm and hierarchical clustering based on similarity measures. None of these methods, however, is ideal for the types of real-world network data with which current research is concerned, such as Internet and web data and biological and social networks. We describe a number of more recent algorithms that appear to work well with these data, including algorithms based on edge betweenness scores, on counts of short loops in networks and on voltage differences in resistor networks.



The properties of social networks

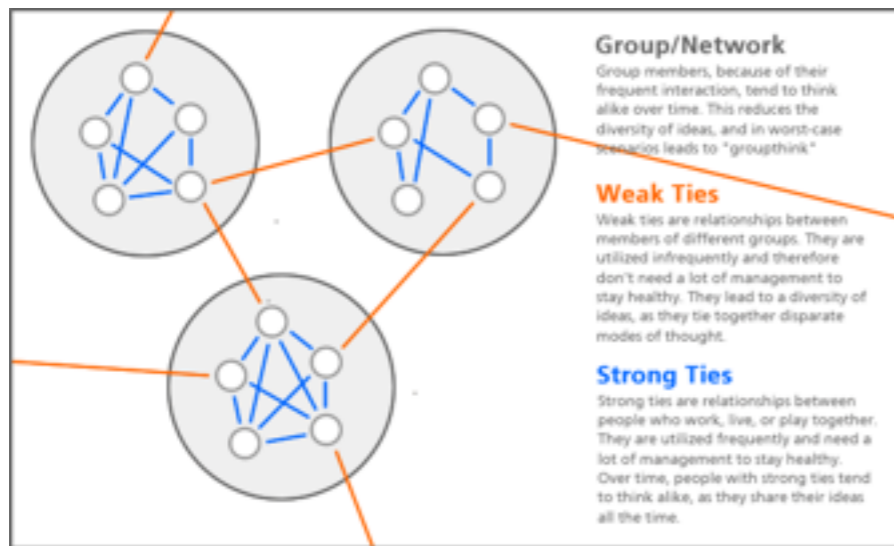
The strength of weak ties (Granovetter thesis)

The Strength of Weak Ties¹

Mark S. Granovetter
Johns Hopkins University

Analysis of social networks is suggested as a tool for linking micro and macro levels of sociological theory. The procedure is illustrated by elaboration of the macro implications of one aspect of small-scale interaction: the strength of dyadic ties. It is argued that the degree of overlap of two individuals' friendship networks varies directly with the strength of their tie to one another. The impact of this principle on diffusion of influence and information, mobility opportunity, and community organization is explored. Stress is laid on the cohesive power of weak ties. Most network models deal, implicitly, with strong ties, thus confining their applicability to small, well-defined groups. Emphasis on weak ties lends itself to discussion of relations *between* groups and to analysis of segments of social structure not easily defined in terms of primary groups.

American Journal of Sociology © 1973

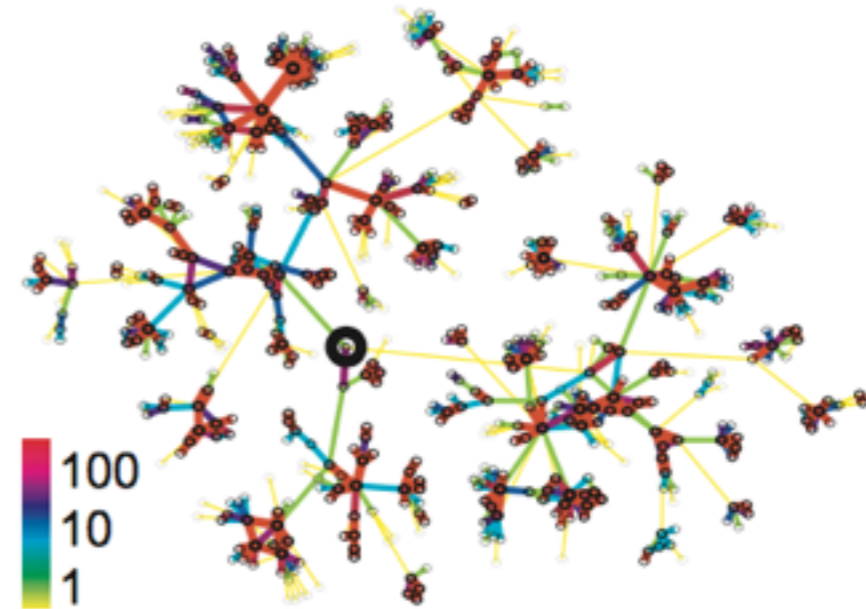


Structure and tie strengths in mobile communication networks

J.-P. Onnela^{*†‡}, J. Saramäki^{*}, J. Hyvönen^{*}, G. Szabó^{§¶}, D. Lazer^{||}, K. Kaski^{*}, J. Kertész^{*••}, and A.-L. Barabási^{§¶}

^{*}Laboratory of Computational Engineering, Helsinki University of Technology, P.O. Box 9203, FI-02015 TKK, Helsinki, Finland; [†]Physics Department, Clarendon Laboratory, Oxford University, Oxford OX1 3PU, United Kingdom; [‡]Department of Physics and Center for Complex Networks Research, University of Notre Dame, South Bend, IN 46556; [§]Center for Cancer Systems Biology, Dana-Farber Cancer Institute, Harvard University, Boston, MA 02115; ^{||}John F. Kennedy School of Government, Harvard University, Cambridge, MA 02138; and ^{••}Department of Theoretical Physics, Budapest University of Technology and Economics, H1111, Budapest, Hungary

Edited by H. Eugene Stanley, Boston University, Boston, MA, and approved January 27, 2007 (received for review November 18, 2006)



The properties of social networks

Dunbar Number

The Social Brain Hypothesis

Robin I.M. Dunbar

Conventional wisdom over the past 160 years in the cognitive and neurosciences has assumed that brains evolved to process factual information about the world. Most attention has therefore been focused on such features as pattern recognition, color vision, and speech perception. By extension, it was assumed that brains evolved to deal with essentially ecological problem-solving tasks.¹

[..people with many connections dedicate on average less time to each of them]

PLoS One. 2011; 6(8): e22656.

PMCID: PMC3149601

Published online 2011 Aug 3. doi: [10.1371/journal.pone.0022656](https://doi.org/10.1371/journal.pone.0022656)

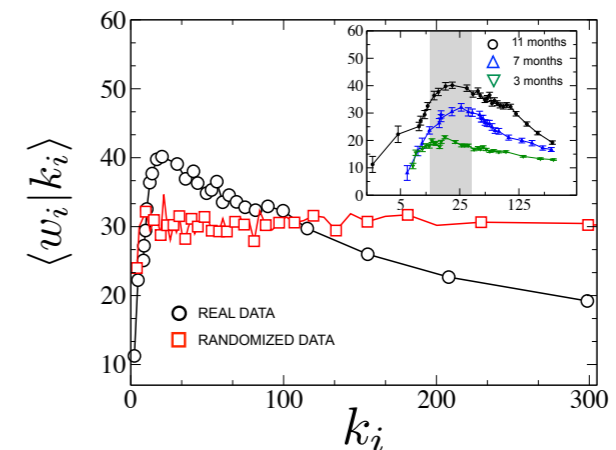
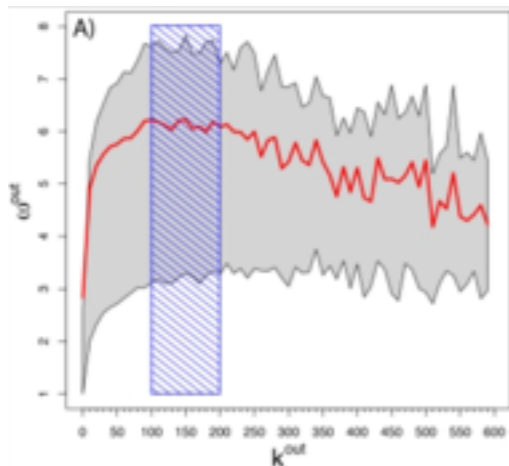
Modeling Users' Activity on Twitter Networks: Validation of Dunbar's Number

Bruno Gonçalves,^{1,2} Nicola Perra,^{1,3,*} and Alessandro Vespignani^{1,2,4}

Matjaz Perc, Editor

Time as a limited resource: Communication strategy in mobile phone networks

Giovanna Miritello^{a, b,} Esteban Moro^{b, c, d,} Rubén Lara^{a,} Rocío Martínez-López^{a,} John Belchamber^{a,} Sam G.B. Roberts^{e,} Robin I.M. Dunbar^f



Other dimensions of Social Networks

Social Structure



Dynamical processes on social networks

bursty behavior

Nature **435**, 207-211 (12 May 2005) | doi:10.1038/nature03459; Received 7 November 2004; Accepted 15 February 2005

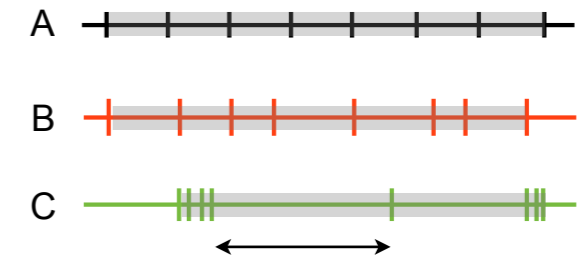
The origin of bursts and heavy tails in human dynamics

Albert-László Barabási¹

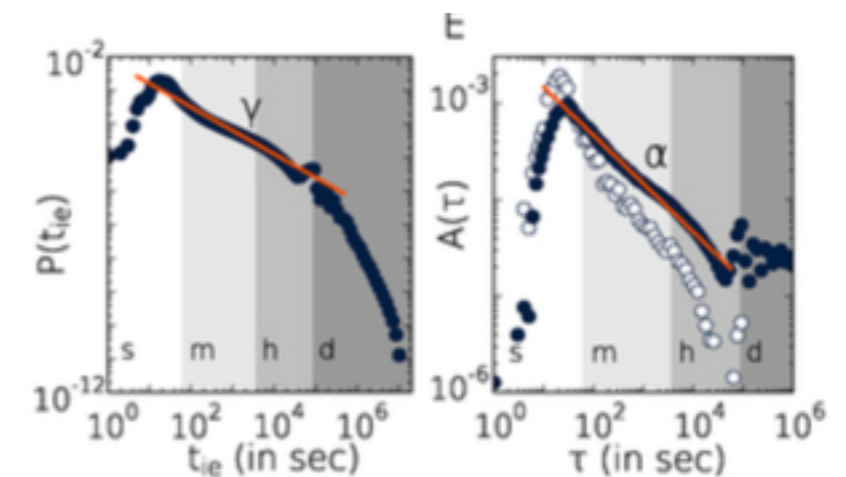
1. Center for Complex Networks Research and Department of Physics, University of Notre Dame, Indiana 46556, USA

Correspondence to: Albert-László Barabási¹ Correspondence and requests for materials should be addressed to A.-L. B. (Email: alb@nd.edu).

The dynamics of many social, technological and economic phenomena are driven by individual human actions, turning the quantitative understanding of human behaviour into a central question of modern science. Current models of human dynamics, used from risk assessment to communications, assume that human actions are randomly distributed in time and thus well approximated by Poisson processes^{1, 2, 3}. In



$$\Delta_{ij}$$

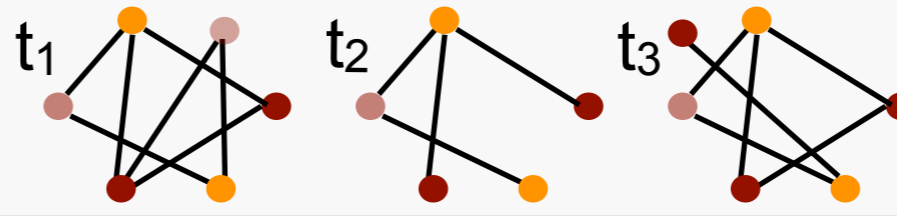


From
Universal features of correlated bursty behaviour
 Márton Karsai, Kimmo Kaski, Albert-László Barabási & János Kertész
Scientific Reports **2**, Article number: 397 | doi:10.1038/srep00397
 Received 02 January 2012 | Accepted 23 April 2012 | Published 04 May 2012

Dynamical processes on social networks

Nodes

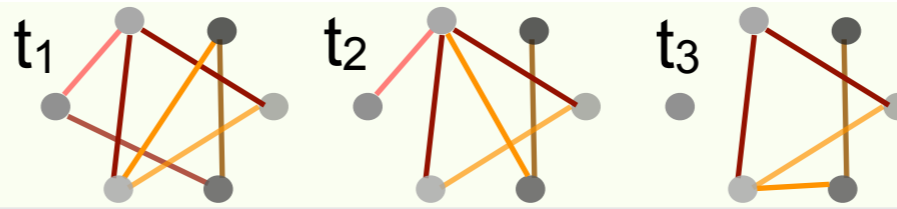
appear/disappear



Barabasi et al., Physica A (2002), Holme et al. Soc.Net.(2004)

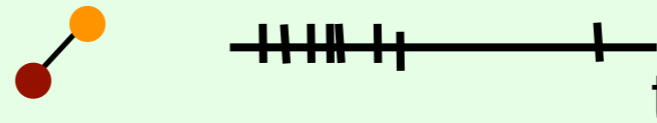
Ties

form/decay



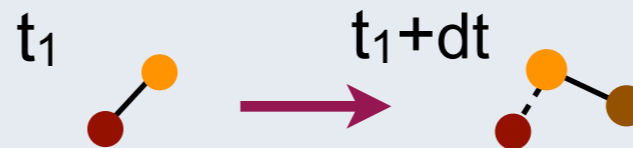
Hidalgo et al., Physica A (2008), Burt, Soc.Net.(2000)

Tie activity is bursty



Barabasi, Nature (2005)

Groups of conversation



Kovanen et al. J.Stat.Mech (2011), Zhao et al. NetMob (2011)

Communities

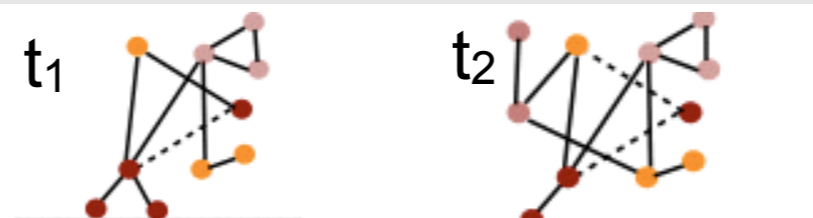
Communities form/change/decay



Palla et al. Proc.of SPIE (2007)

Network

Networks form/change/decay



Kossinets and Watts, Science (2006)

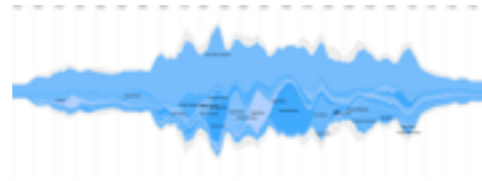
Other dimensions of Social Networks

Social Structure



+

Dynamics



Spatial processes on social networks



Spatial networks

Marc Barthélemy

[Show more](#)

Abstract

Complex systems are very often organized under the form of networks where nodes and edges are embedded in space. Transportation and mobility networks, Internet, mobile phone networks, power grids, social and contact networks, and neural networks, are all examples where space is relevant and where topology alone does not contain all the information. Characterizing and understanding the structure and the evolution of spatial networks is thus crucial for many different fields, ranging from urbanism to epidemiology. An important consequence of space on networks is that there is a cost associated with the length of edges which in turn has dramatic effects on the topological structure of these networks. We will thoroughly explain the current state of our understanding of how the spatial constraints affect the structure and properties of these networks. We will review the most recent empirical observations and the most important models of spatial networks. We will also discuss various processes which take place on these spatial networks, such as phase transitions, random walks, synchronization, navigation, resilience, and disease spread.

Friendship and Mobility: User Movement In Location-Based Social Networks

Eunjoon Cho *
Stanford University
ejcho@stanford.edu

Seth A. Myers *
Stanford University
samyers@stanford.edu

Jure Leskovec
Stanford University
jure@cs.stanford.edu



(a) Spatial model



(b) Temporal model

Figure 6: (a) Check-ins of a user in San Francisco: geographic distribution of check-ins when in home/work state. (b) Temporal model: distance to the red/blue line from the center is proportional the prob. of user being in home/work state.

Socio-Spatial Properties of Online Location-Based Social Networks

Salvatore Scellato

Computer Laboratory
University of Cambridge

salvatore.scellato@cam.ac.uk

Anastasios Noulas

Computer Laboratory
University of Cambridge

anastasios.noulas@cl.cam.ac.uk

Renaud Lambiotte

Department of Mathematics
Imperial College London

r.lambiotte@imperial.ac.uk

Cecilia Mascolo

Computer Laboratory
University of Cambridge

cecilia.mascolo@cl.cam.ac.uk

A universal model for mobility and migration patterns

Filippo Simini, Marta C. González, Amos Maritan & Albert-László Barabási

[Affiliations](#) | [Contributions](#) | [Corresponding authors](#)

Nature 484, 96–100 (05 April 2012) | doi:10.1038/nature10856

Received 08 August 2011 | Accepted 13 January 2012 | Published online 26 February 2012

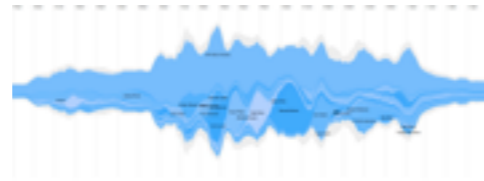
Other dimensions of Social Networks

Social Structure



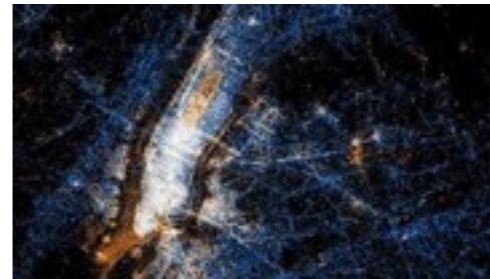
+

Dynamics



+

Mobility



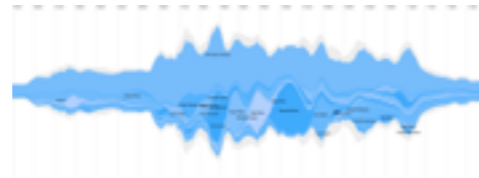
Other dimensions of Social Networks

Social Structure



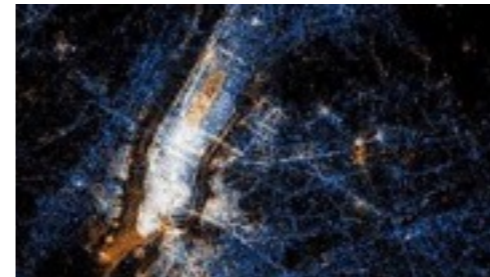
+

Dynamics



+

Mobility



+

Content



+

Other

- demographic
- economics
- relationships



From properties to information

Social variables

- Degree
- Clustering
- Centrality
- Tie strength

Social networks and social roles

- Community analysis
- Affinity
- Social roles
(central users/influential/early adopters)

Activity

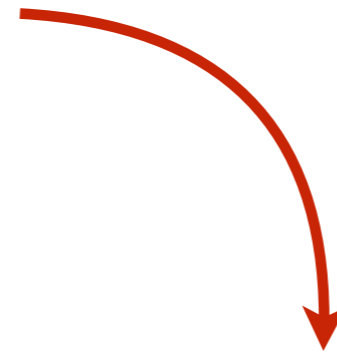
- Temporal patterns (hourly/weekly/..)
- Anomalies
- Correlations and cascades

Spatial

- Mobility
- Points of interest
- Hot spots
- Relation with socio-economic factors

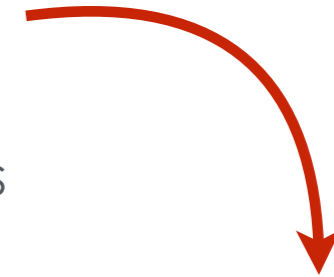
Content

- Opinion analysis
- Sentiment analysis
- Interests
- Trends



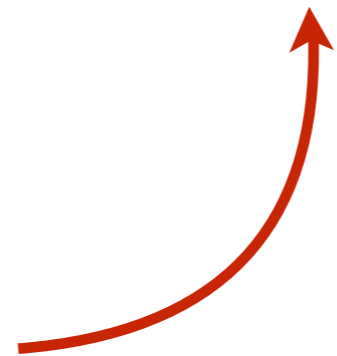
Analysis and Models

- Social Network Analysis
- User segmentation
- Recommendation systems
- Prediction models
- Information diffusion models



Applications

- Social habits and needs
- Opinion analysis
- Products/brands penetration and diffusion
- Resource optimization
- Smart cities (health, public transport)
- Fraud and risk
- Criminal and terrorist networks
- Communication flows in companies



From properties to information

Social variables

- Degree
- Clustering
- Centrality
- Tie strength

Social networks and social roles

- Community analysis
- Affinity
- Social roles
(central users/influential/early adopters)

Activity

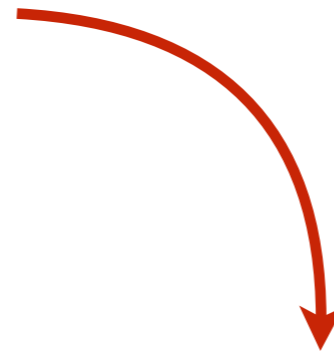
- Temporal patterns (hourly/weekly/..)
- Anomalies
- Correlations and cascades

Spatial

- Mobility
- Points of interest
- Hot spots
- Relation with socio-economic factors

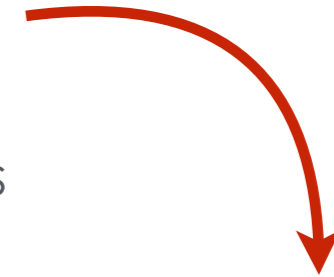
Content

- Opinion analysis
- Sentiment analysis
- Interests
- Trends



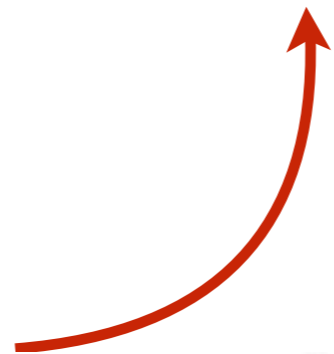
Analysis and Models

- Social Network Analysis
- User segmentation
- Recommendation systems
- Prediction models
- Information diffusion models

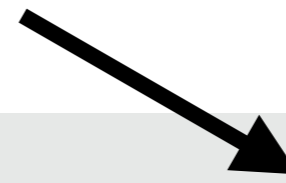


Applications

- Social habits and needs
- Opinion analysis
- Products/brands penetration and diffusion
- Resource optimization
- Smart cities (health, public transport)
- Fraud and risk
- Criminal and terrorist networks
- Communication flows in companies



it's not the end.



1: Identify key players

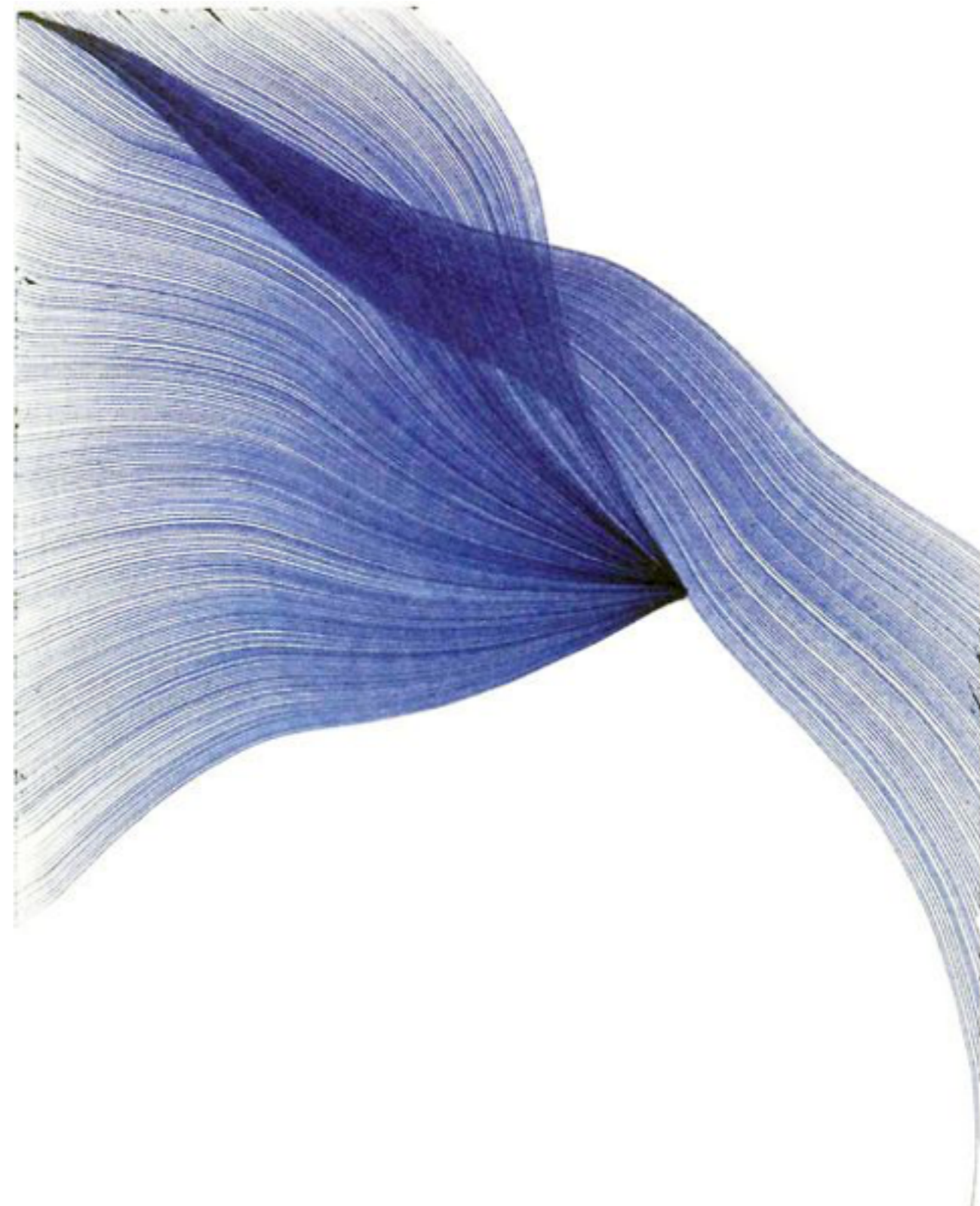
In viral marketing, it is important to identify and target the right set of key players in a population to spread information efficiently and effectively.

In disease and epidemic spreading, immunizing important nodes is an effective solution to suppress the spreading.

In management, it is critical to identify and strategically place the key players to improve the productivity of colleagues.

In politics, it is necessary to identify key players to gain advantage in political campaigns.

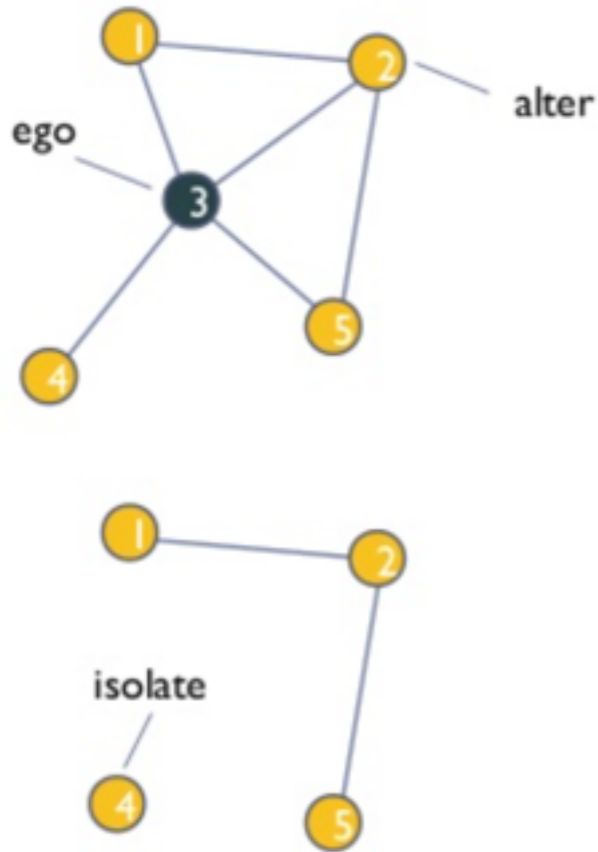
In everyday life, it is useful to identify the right people to strategically find information/resource/etc.



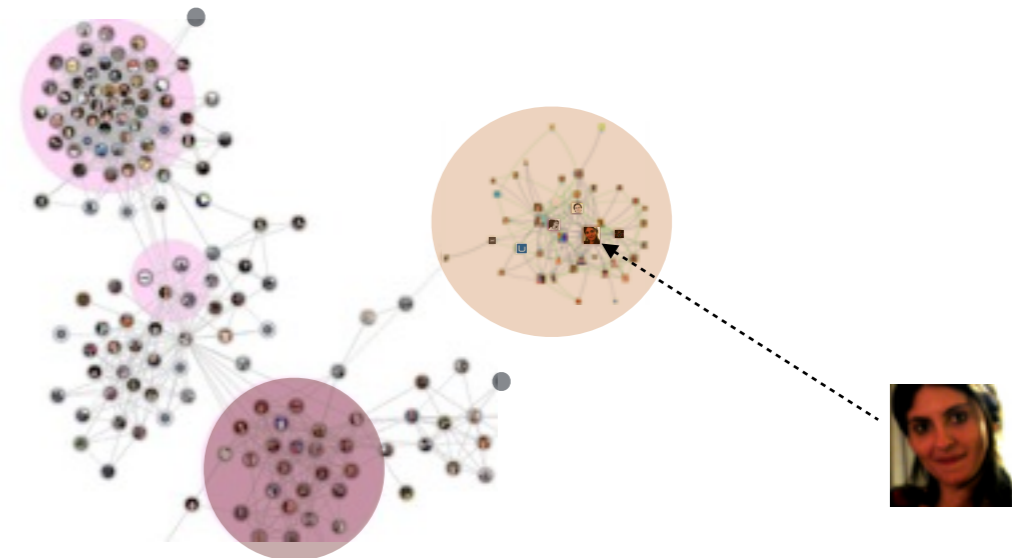
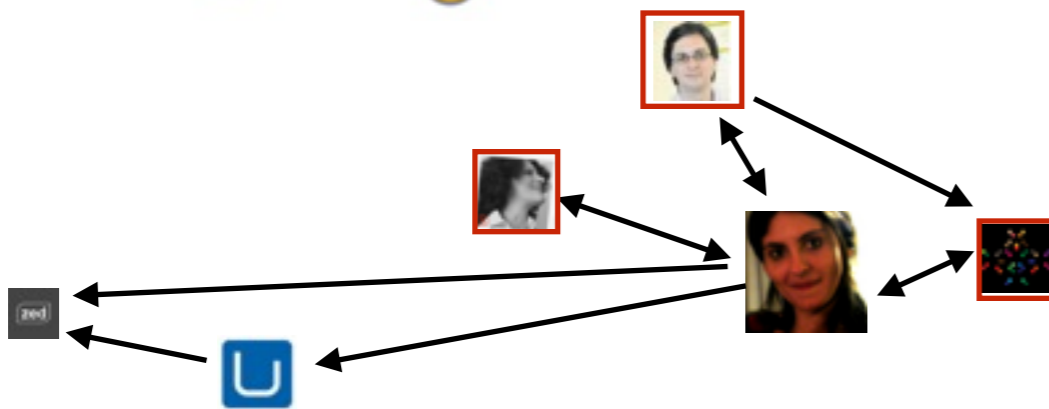
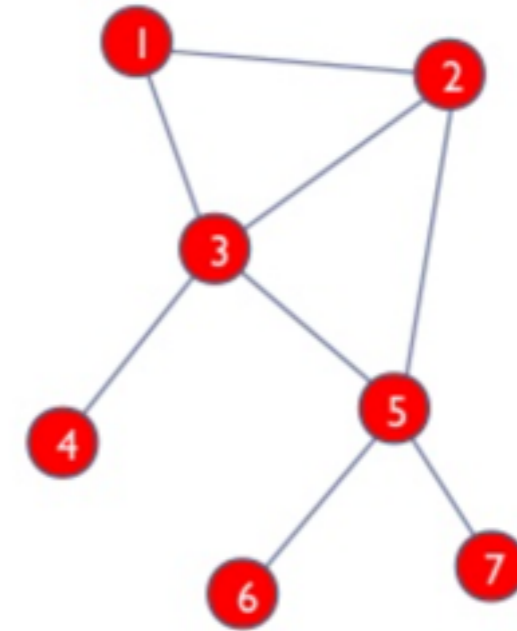
1: Identify key players

What we want to measure

Local, ego-network



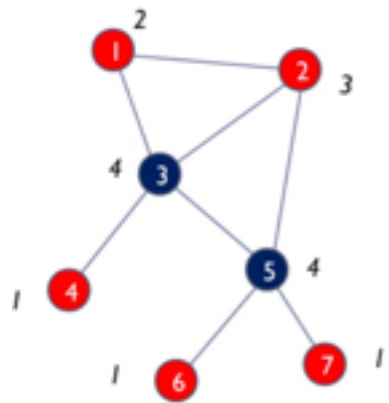
Global, whole network



1: Identify key players

What we measure

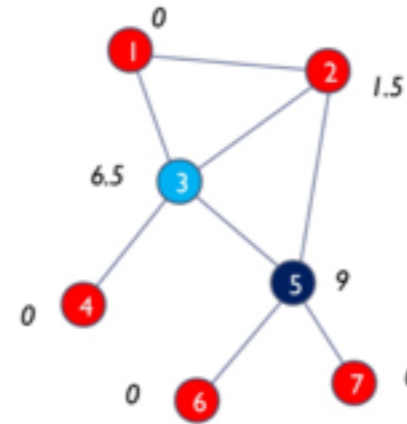
Degree centrality



Number of links of a node. Often used as measure of connectedness and therefore of popularity. It is quantity over quality

Nodes 3 and 5 have highest degree

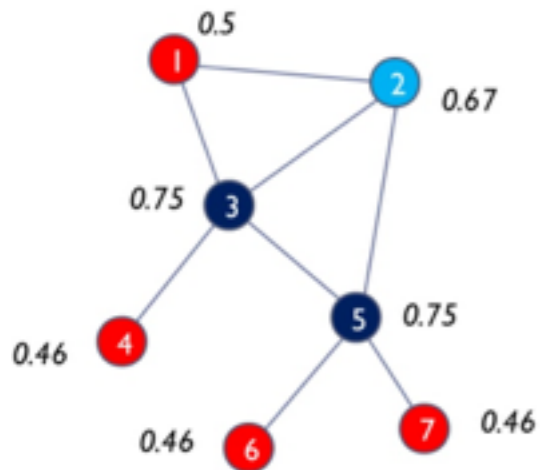
Betweenness centrality



Number of shortest paths between any 2 nodes that pass through a given node. It can be used to detect nodes that act as bridges between different zones of the network.

Node 5 has highest betweenness

Closeness centrality



How many hops on average it takes to reach every other node? The less, the better. It is a measure of reach.

Nodes 3 and 5 have highest closeness

Eigenvector centrality



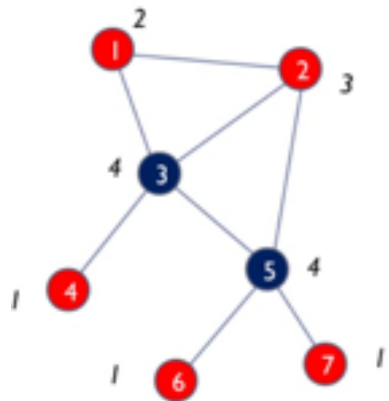
Similar to how Google ranks web pages: links from high linked-to pages count more. Useful in determining who is connected to the most connected nodes.

Node 3 has the highest eigenvector centrality

1: Identify key players

What we get

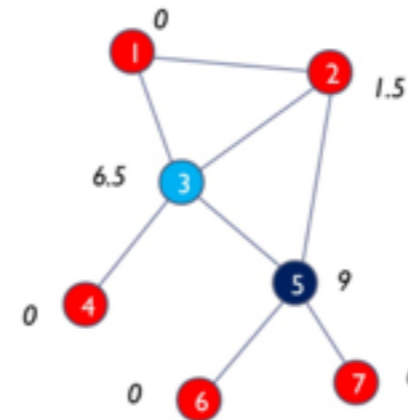
Degree centrality



How many people can this person reach directly?

Nodes 3 and 5 have highest degree

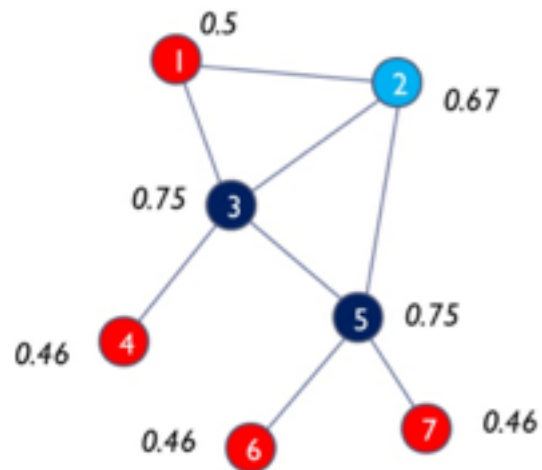
Betweenness centrality



How likely is this person to be the most direct route between two people in the network?

Node 5 has highest betweenness

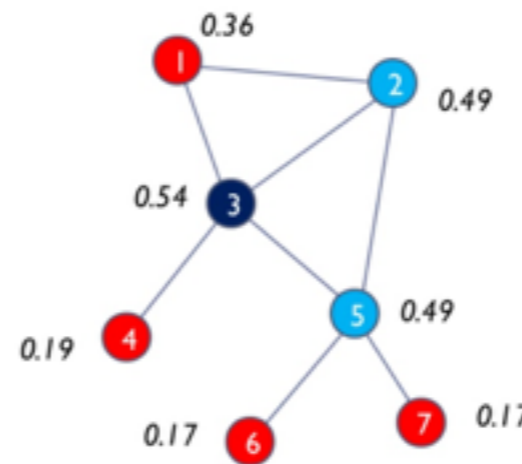
Closeness centrality



How fast can this person reach everyone in the network?

Nodes 3 and 5 have highest closeness

Eigenvector centrality



How well is this person connected to other well-connected people?

Node 3 has the highest eigenvector centrality

1: Identify key players

IP[y]: Notebook

keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Data and Libraries

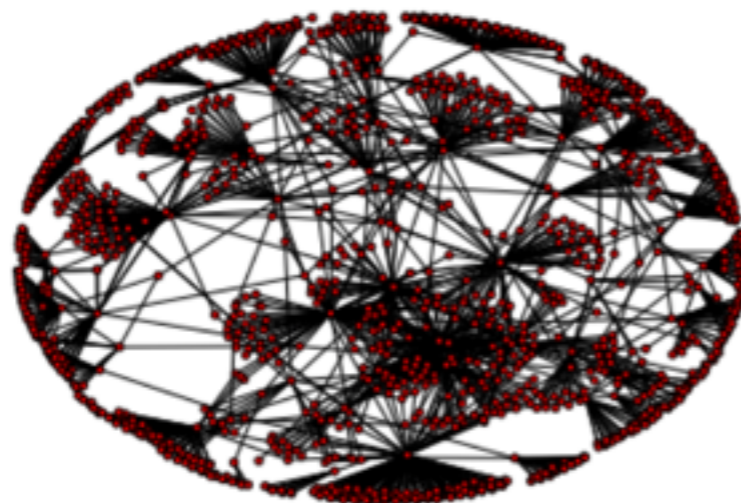
In this session we will use a simplified version of the facebook-links.txt file which is courtesy of the Max Planck Institute for Software Systems: <http://socialnetworks.mpi-sws.org/data-wosn2009.html> File facebook-links.txt contains a list of all of the user-to-user links from the Facebook New Orleans network. These links are undirected on Facebook.

File facebook-links.txt contains a list of all of the user-to-user links from the Facebook New Orleans network. These links are undirected on Facebook.

```
In [1]: import networkx as nx
import community
import pylab as plt
import math
import operator
import pandas as pd
from numpy import *
# show plots in the notebook
%matplotlib inline
```

```
In [2]: g = nx.read_edgelist('/Users/giovanna/Dropbox/CONFERENCES/Netmob2015/data_facebook/facebook-links-sample2000.dat',
delimiter=None, create_using=None,
nodetype=int, data=True, edgetype=None,
encoding='utf-8')
```

```
In [3]: nx.draw(g, node_size=10)
```



1: Identify key players

IP[y]: Notebook

keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Degree Centrality

The degree centrality of a node is just its degree, that is, the number of neighbors it has. The more connected to other nodes a node is, the higher its degree centrality. It's somehow measuring the "celebrities" of the graph.

```
In [4]: degree = nx.degree_centrality(g) # returns a dictionary
# The degree centrality values are normalized by dividing by the /
# maximum possible degree in a simple graph n-1 where n is the number of nodes in G.
```

```
In [5]: sorted(degree.items(), key=lambda x: x[1], reverse = True)[:10]
```

```
Out[5]: [(510, 0.10333551340745586),
(89, 0.06998037933289733),
(858, 0.06344015696533682),
(571, 0.05428384565075213),
(86, 0.04839764551994768),
(1284, 0.04447351209941138),
(90, 0.040549378678875085),
(509, 0.039241334205362986),
(373, 0.03466317854807063),
(303, 0.034009156311314584)]
```

```
In [6]: # degree_hist = plt.hist(list(degree.values()), 100)
# plt.loglog(degree_hist[1][1:], degree_hist[0])
# plt.title("Degree histogram (loglog)")
```

```
In [7]: def get_key_players_degree(G, top=10):
prk = nx.degree_centrality(G)
sprk = sorted(prk.items(), key=operator.itemgetter(1), reverse=True)
toplabeles = [x[0] for x in sprk[:top]]
return toplabeles
```

```
In [8]: # print top 10 nodes ids for degree

get_key_players_degree(g)
```

```
Out[8]: [510, 89, 858, 571, 86, 1284, 90, 509, 373, 303]
```

1: Identify key players

IP[y]: Notebook keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
get_key_players_degree(g)
```

```
Out[8]: [510, 89, 858, 571, 86, 1284, 90, 509, 373, 303]
```

Betweenness centrality

The betweenness centrality of a node is defined as the proportion of best paths between any other pairs of nodes which pass through it. By definition the betweenness centrality of a node depends on the connection properties of every pair of nodes in the graph, except pairs with it, whereas degree centrality depends only on the node's neighbors.

```
In [9]: def get_key_players_betweenness(G, top=10):
prk = nx.betweenness_centrality(G)
sprk = sorted(prk.items(), key=operator.itemgetter(1), reverse=True)
toplabeles = [x[0] for x in sprk[:top]]
return toplabeles
```

```
In [10]: # print top 10 nodes ids for betweenness
top_bet = get_key_players_betweenness(g)
print top_bet

[510, 232, 571, 639, 858, 711, 86, 275, 161, 446]
```

Closeness centrality

Closeness centrality is based on distance calculation and tries to quantify the intuitive notion of what position a node occupies, if central or peripheral.

```
In [11]: def get_key_players_closeness(G, top=10):
prk = nx.closeness_centrality(G)
sprk = sorted(prk.items(), key=operator.itemgetter(1), reverse=True)
toplabeles = [x[0] for x in sprk[:top]]
return toplabeles
```

```
In [12]: # print top 10 nodes ids for closeness
top_clos = get_key_players_closeness(g)
print top_clos

[90, 510, 571, 639, 89, 88, 509, 111, 9, 86]
```


1: Identify key players

IP[y]: Notebook **keyPlayers** Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
top_clos = get_key_players_closeness(g)
print top_clos

[90, 510, 571, 639, 89, 88, 509, 111, 9, 86]
```

Eigenvector Centrality

For this measure, an important node is connected to important neighbors. Which is very similar to the idea of the Google's PageRank. PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

```
In [13]: def get_key_players_eigenvector(G, top=10):
         prk = nx.eigenvector_centrality(G)
         sprk = sorted(prk.items(), key=operator.itemgetter(1), reverse=True)
         toplabels = [x[0] for x in sprk[:top]]
         return toplabels
```

```
In [14]: # print top 10 nodes ids for eigenvector centrality
top_eigenvector = get_key_players_eigenvector(g)
print top_eigenvector

[89, 86, 90, 571, 87, 509, 88, 111, 96, 103]
```

Visualization

```
In [15]: # compute centrality measures for all nodes in the graph
res_degree = nx.degree(g)
res_bet = nx.betweenness_centrality(g)
res_clos = nx.closeness_centrality(g)
res_eigen = nx.eigenvector_centrality(g)
```

1: Identify key players

IP[y]: Notebook

keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
print top_eigenvector
```

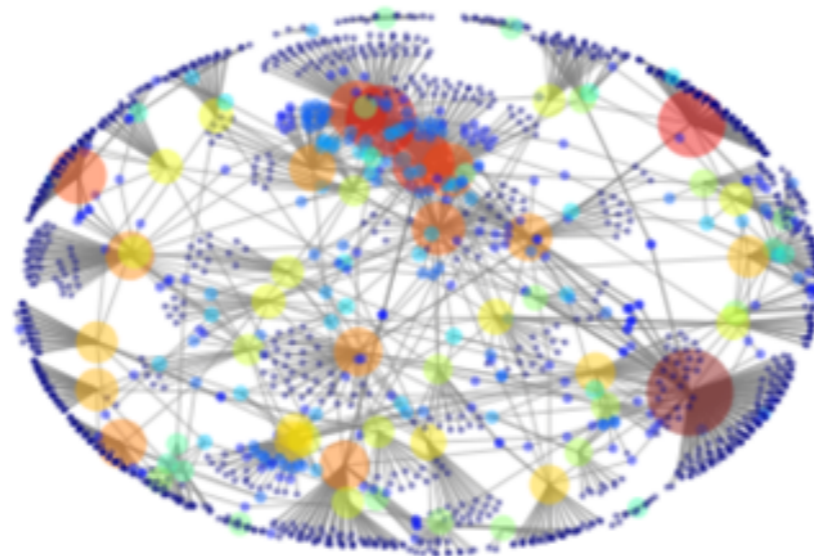
```
[89, 86, 90, 571, 87, 509, 88, 111, 96, 103]
```

Visualization

```
In [15]: # compute centrality measures for all nodes in the graph
res_degree = nx.degree(g)
res_bet = nx.betweenness centrality(g)
res_clos = nx.closeness centrality(g)
res_eigen = nx.eigenvector centrality(g)
```

Size and color proportional to degree

```
In [16]: node_list = res_degree.keys()
node_size = [7*v for v in res_degree.values()]
node_col = [log(v)+1 for v in res_degree.values()]
# pos = nx.shell_layout(g)
nx.draw(g, node_list=node_list, node_size=node_size,
        node_color=node_col, linewidths=0, alpha=.45,
        edge_color="gray")
```



1: Identify key players

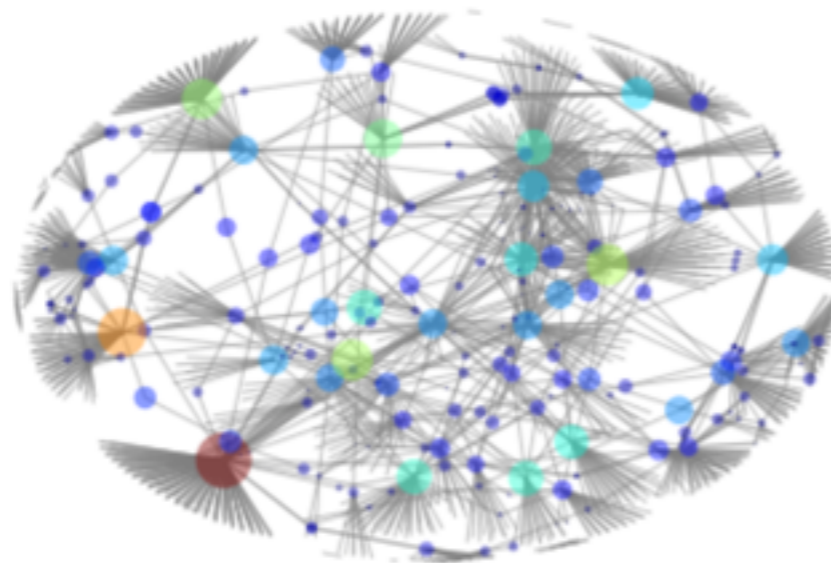
IP[y]: Notebook keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Size and color proportional to betweenness

```
In [17]: node_list = res_bet.keys()
node_size = [1800 * v for v in res_bet.values()]
node_col = [v + 6 for v in res_bet.values()]
# pos = nx.shell_layout(g)
nx.draw(g, node_list=node_list, node_size=node_size,
        node_color=node_col, linewidths=0, alpha=.45,
        edge_color="gray")
```



1: Identify key players

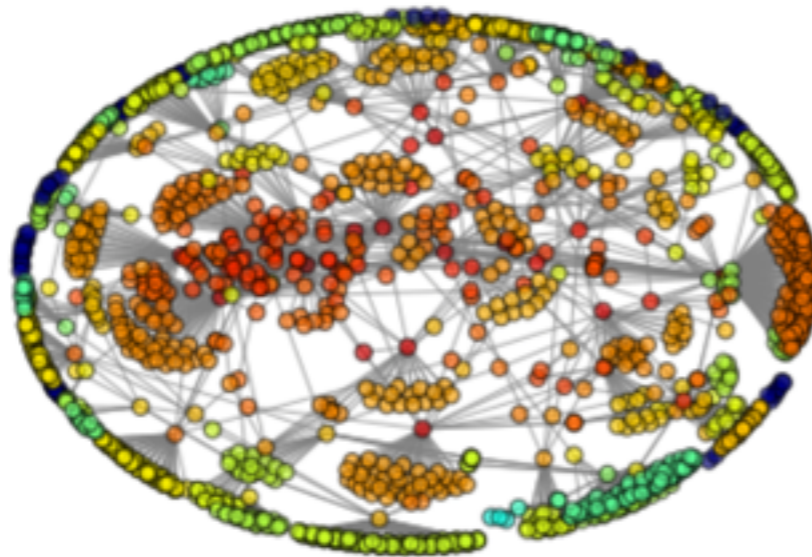
IP[y]: Notebook keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Size and color proportional to closeness

```
In [18]: node_list = res_clos.keys()
node_size = [sqrt(v) + 40 for v in res_clos.values()]
node_col = [v + 1000 for v in res_clos.values()]
nx.draw(g, node_list=node_list, node_size=node_size,
        node_color=node_col, alpha=.65,
        edge_color="gray")
```



1: Identify key players

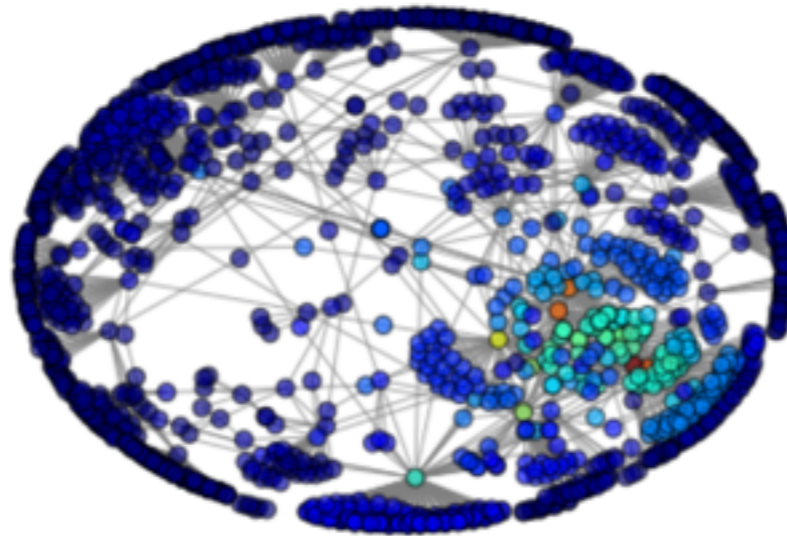
IP[y]: Notebook keyPlayers Last Checkpoint: Mar 24 17:26 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Size and color proportional to eigenvector centrality

```
In [19]: node_list = res_eigen.keys()
node_size = [sqrt(v) + 50 for v in res_eigen.values()]
node_col = [sqrt(v) + 100 for v in res_eigen.values()]
nx.draw(g, node_list=node_list, node_size=node_size,
        node_color=node_col, alpha=.65,
        edge_color="gray")
```



2: Link Persistence

Social networks are highly dynamic objects

To what extent can the evolution of a network be modeled using features intrinsic of the network itself?

Link persistence/decay in organizational networks; interactions or collaborations that have not been utilized

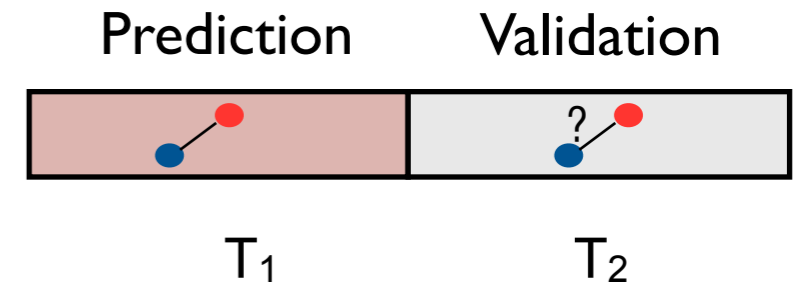
Terrorist networks; link prediction in this context allows one to conjecture that particular individuals are working together even though their interaction has not been directly observed



2: Link Persistence

Problem

Given the properties of links at time t predict which ones persist or decay at time $t + \Delta t$



Approach and Model

Logistic Regression with following predictive variables:

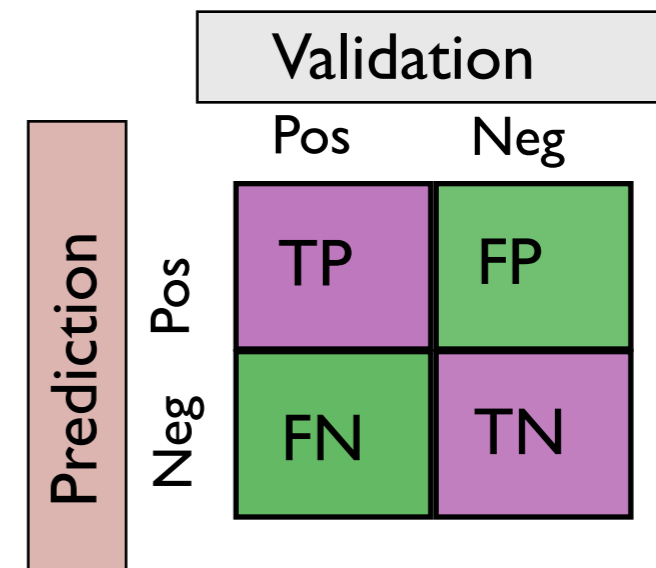
- Topological properties
 - volume of interaction
 - topological overlap
 - Adamic-Adar coefficient

- Link lifetime
 - stability

- Temporal Features
 - coefficient of variation
 - time since last interaction (freshness)

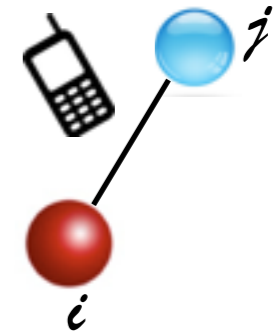
Validation and Performance

- Comparing the prediction with the observation of link in the future time window.
- Test/Training set
- Cross Validation



2: Link Persistence

Data: anonymized mobile phone logs of communication between people.



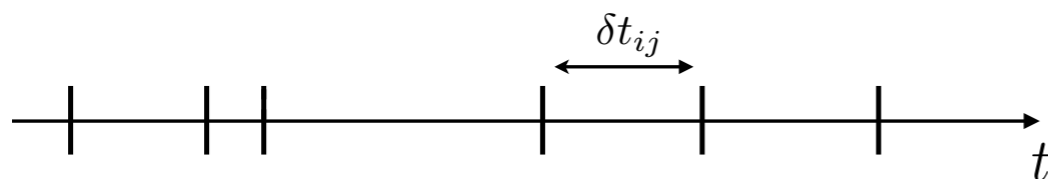
A quick remind of some variables..

Volume of interaction: total number of calls between nodes i and j

Temporal stability: time difference between the last and first communication between nodes i and j Δ_{ij}

Stable $\Delta_{ij} \simeq T$
 Unstable $\Delta_{ij} \simeq 0$

Coefficient of variation of inter-event time series



$$cv_{\delta t_{ij}} = \sigma_{\delta t_{ij}} / |\mu_{\delta t_{ij}}|$$

σ stand.deviation
 μ mean

Poisson-like $cv_{\delta t_{ij}} \simeq 1$
 Bursty $cv_{\delta t_{ij}} > 1$

Topological Overlap and Adamic Adar coefficient

$$O_{ij} = \frac{n_{ij}}{(k_i - 1) + (k_j - 1) - n_{ij}}$$

(almost) all friends in common $O_{ij} \simeq 1$
 (almost)no friends in common $O_{ij} \simeq 0$

2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (unsaved changes)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Link Prediction/Decay

Import Data and Libraries

```
In [1]: import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
from patsy import dmatrices
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import train_test_split
from sklearn import metrics
from sklearn.cross_validation import cross_val_score
from sklearn import preprocessing
```

We will use a dataset derived from mobile communication between anonymized users. The dataset contains:

- the id of each link
- number of calls in T1=4 months
- stability in T1=4 months
- overlap in T1=4 months
- Adamic-Adar index in T1=4 months
- coefficient of variation in T1=4 months
- freshness in T1=4 months
- decay (that can take the values 1 or 0 depending if the link persists or not respectively in the next T2=6 months)

All the properties of links in T1 will be used as predictor variables to predict whether the link decays or not in the next T2=6 months (1 = persist = at least 1 calls observed; 0 = decay = no calls observed).

```
In [2]: dta = pd.DataFrame.from_csv("/Users/giovanna/Dropbox/CONFERENCES/Netmob2015/link_prediction/data_links_sample.dat",
                                header=0, sep=' ')
#("/Users/giovanna/Dropbox/CONFERENCES/Netmob2015/link_prediction/datos300-variT2-simplified-sample.dat",
#header=0, sep=' ')
```

```
In [3]: # show the names of columns
list(dta.columns.values)
```

```
Out[3]: ['cij', 'dtij', 'oij', 'aaij', 'cvij', 'friij', 'decay']
```

```
In [4]: # show plots in the notebook
%matplotlib inline
```


2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (unsaved changes)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Data overview

All the predictors we are considered are continuous. Before to use a continuous predictor in a linear regression it should be approximately symmetric. This helps ensure the numerical stability of the solution. So, let's first examine the the distribution of the predictors.

In [5]: `dta.describe()`

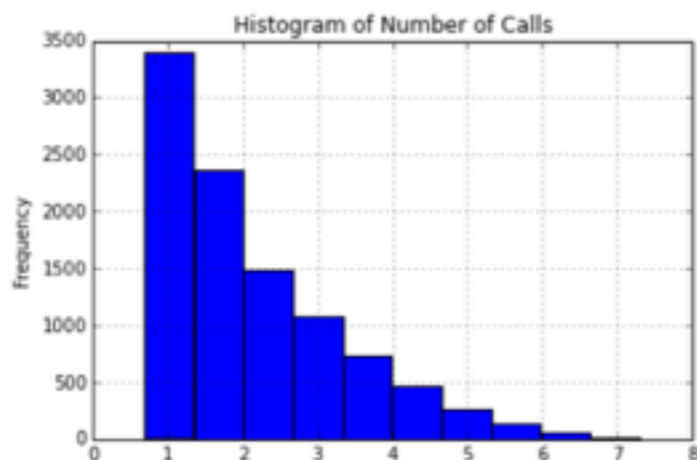
Out[5]:

	cij	dtij	oij	aaaj	cvij	frij	decay
count	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000
mean	2.063487	6.077203	0.035328	0.949848	1.547560	87.475767	0.627363
std	1.280885	3.751681	0.043993	1.441987	0.847347	32.692191	0.483531
min	0.693147	0.010758	0.000000	0.000000	0.016351	0.557465	0.000000
25%	1.098612	2.627167	0.000000	0.000000	1.067933	66.594826	0.000000
50%	1.791759	6.762886	0.020833	0.465295	1.392006	98.783218	1.000000
75%	2.833213	9.583361	0.050633	1.238666	1.791500	115.495891	1.000000
max	7.295056	10.997250	0.500000	19.661354	13.236598	121.159109	1.000000

Before to use a continuous predictor in a linear regression it should be approximately symmetric. This helps ensure the numerical stability of the solution. This is the reason why both number of calls and stability have been transformed by using sqrt/log.

In [6]: `# histogram of number of calls`
`dta.cij.hist()`
`plt.title('Histogram of Number of Calls')`
`plt.xlabel('Number of calls')`
`plt.ylabel('Frequency')`

Out[6]: `<matplotlib.text.Text at 0x10875d9d0>`



2: Link Persistence

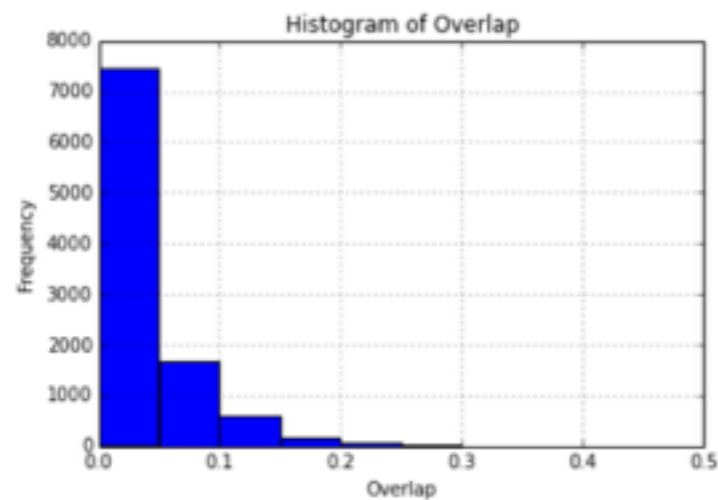
IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (unsaved changes)

File Edit View Insert Cell Kernel Help

Cell Toolbar: None

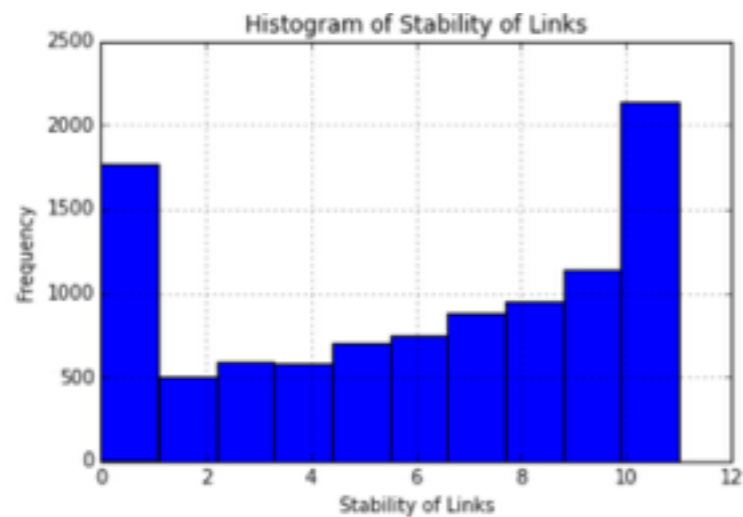
```
In [7]: # histogram of number of calls
dta.oij.hist()
plt.title('Histogram of Overlap')
plt.xlabel('Overlap')
plt.ylabel('Frequency')
```

Out[7]: <matplotlib.text.Text at 0x1031707d0>



```
In [8]: # histogram of number of stability
dta.dtij.hist()
plt.title('Histogram of Stability of Links')
plt.xlabel('Stability of Links')
plt.ylabel('Frequency')
```

Out[8]: <matplotlib.text.Text at 0x1031e8f90>



2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Logistic Regression

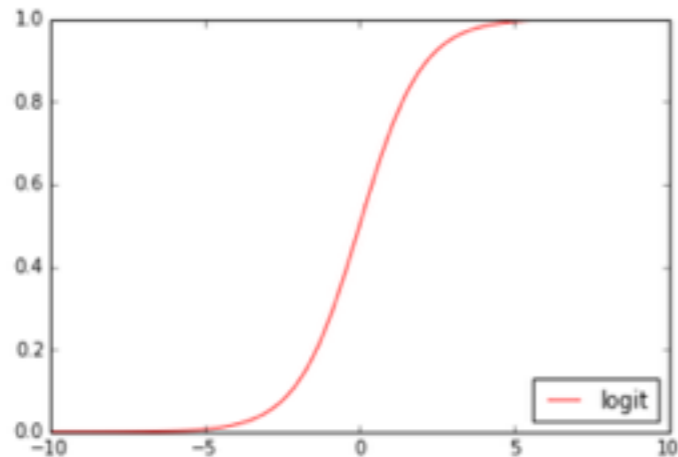
Let's go ahead and run logistic regression on the entire data set, and see how accurate it is!

Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

```
In [9]: # Logistic function
x = np.linspace(-10, 10, 100)
y = 1.0 / (1.0 + np.exp(-x))

plt.plot(x, y, 'r-', label='logit')
plt.legend(loc='lower right')
```

Out[9]: <matplotlib.legend.Legend at 0x1087a9550>



Prepare data and apply logistic regression

```
In [10]: list(dta.columns.values)
```

Out[10]: ['cij', 'dtij', 'oij', 'aaij', 'cvij', 'friij', 'decay']

2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Logistic Regression from one single predictor

```
In [11]: # create dataframes with an intercept column and variables for
# occupation and occupation_husb
y, X = dmatrices('decay ~ cij',
                dta, return_type="dataframe")
print X.columns

Index([u'Intercept', u'cij'], dtype='object')
```

```
In [12]: # flatten y into a 1-D array, so that scikit-learn will properly understand it as the response variable.
y = np.ravel(y)
```

```
In [13]: # instantiate a logistic regression model, and fit with X and y
model = LogisticRegression()
model = model.fit(X, y)
```

```
In [14]: # check the accuracy
model.score(X, y)
```

```
Out[14]: 0.68836883688368833
```

```
In [15]: # examine the coefficients
pd.DataFrame(zip(X.columns, np.transpose(model.coef_)))
```

```
Out[15]:
```

	0	1
0 Intercept		[-0.483694575779]
1 cij		[0.798099318659]

The coefficient for cij (0.7980) is the difference in the log odds; for a one-unit increase in the number of calls, the expected change in log-odds is 0.7980. By taking the exp of this quantity we recover the odds: $\exp(0.7980) = 2.22$. So for one-unit increase in number of calls, we expect to see about the 1.22% increase in the odds of appearance of the link.

2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Logistic Regression from multiple predictors

```
In [16]: # create dataframes with an intercept column and variables for
# occupation and occupation_husb
y, X = dmatrices('decay ~ cij + dtij + oij + aaij + \
                cvij + frij',
                dta, return_type="dataframe")
print X.columns

Index([u'Intercept', u'cij', u'dtij', u'oij', u'aaij', u'cvij', u'frij'], dtype='object')
```

```
In [17]: # scale all the variables
X_scaled = preprocessing.scale(X)
```

```
In [18]: # flatten y into a 1-D array
y = np.ravel(y)
```

```
In [19]: # instantiate a logistic regression model, and fit with X and y
model = LogisticRegression()
model = model.fit(X, y)
```

```
In [20]: # check the accuracy
model.score(X, y)
```

Out[20]: 0.75907590759075905

```
In [21]: # examine the coefficients
pd.DataFrame(zip(X.columns, np.transpose(model.coef_)))
```

Out[21]:

	0	1
0 Intercept	[-1.15478701686]	
1 cij	[0.304301518539]	
2 dtij	[0.174326842374]	
3 oij	[2.07575408641]	
4 aaij	[0.0670931665157]	
5 cvij	[-0.264854446812]	
6 frij	[0.0185822196228]	

2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

```
In [21]: # examine the coefficients
pd.DataFrame(zip(X.columns, np.transpose(model.coef_)))
```

```
Out[21]:
```

	0	1
0 Intercept		[-1.15478701686]
1 cij		[0.304301518539]
2 dtij		[0.174326842374]
3 oij		[2.07575408641]
4 aaij		[0.0670931665157]
5 cvij		[-0.264854446812]
6 frij		[0.0185822196228]

PREDICTED PROBABILITIES $p = \text{ODDS} / (1 + \text{ODDS})$

The predicted probabilities tell us that fixed all the others parameters, we will see the 30% increase of odds to observe the link in T2 for unit increase of cij ($p=1.30$), the 18% increase of odds for unit increase of dtij, etc.

The sign of the value of the coefficients is related to the correlation with the probability for the link to persist (outcome=1) in the following 6 months. If the sign is positive such probability will increase for larger values of the corresponding parameter, otherwise it will decrease. So for example, the larger the number of calls the higher the probability to observe the link in time window T2. On the contrary, the negative sign for the coefficient of variation indicates that the larger the cv (the more heterogeneous the temporal pattern of communication), the smaller the probability that the link will be observed in T2.

Increases in number of calls and common friends correspond to a decrease in the likelihood of decay in the next 6 months. The coefficient for cij (0.304) is the difference in the log odds; for a one-unit increase in the number of calls, the expected change in log-odds is 0.304. By taking the exp of this quantity we recover the odds: $\exp(0.304) = 1.355269$. So for one-unit increase in number of calls, we expect to see about the 35% of decrease in the odds of decay of the link.

```
In [22]: np.exp(model.coef_)
```

```
Out[22]: array([[ 0.31512465,  1.35567776,  1.19044459,  7.97055468,  1.06939511,
  0.76731763,  1.01875594]])
```


IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

Model Evaluation

Train and Test Set

So far, we have trained and tested on the same set. Let's instead split the data into a training set and a testing set.

```
In [23]: # evaluate the model by splitting into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
model2 = LogisticRegression()
model2.fit(X_train, y_train)
```

```
Out[23]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, penalty='l2', random_state=None, tol=0.0001)
```

```
In [24]: # predict class labels for the test set
predicted = model2.predict(X_test)
print predicted
```

```
[ 0.  0.  1. ...,  1.  1.  1.]
```

```
In [25]: # generate class probabilities
probs = model2.predict_proba(X_test)
print probs
```

```
[[ 0.57168653  0.42831347]
 [ 0.71803643  0.28196357]
 [ 0.03204813  0.96795187]
 ...,
 [ 0.18895381  0.81104619]
 [ 0.21808514  0.78191486]
 [ 0.22012286  0.77987714]]
```

These are the predicting probabilities for each link to decay. Let's generate some evaluation metrics.

```
In [26]: # generate evaluation metrics (accuracy and AUC)
print metrics.accuracy_score(y_test, predicted)
print metrics.roc_auc_score(y_test, probs[:, 1])
print metrics.mean_absolute_error(y_test, predicted)
print metrics.f1_score(y_test, predicted)
```

```
0.762666666667
0.834180729774
0.237333333333
0.817716333845
```

2: Link Persistence

IP[y]: Notebook linkPrediction Last Checkpoint: Mar 24 17:55 (autosaved)

File Edit View Insert Cell Kernel Help

Markdown Cell Toolbar: None

```
In [27]: print metrics.confusion_matrix(y_test, predicted)
```

```
[[ 691  405]
 [ 307 1597]]
```

```
In [28]: print metrics.classification_report(y_test, predicted)
# see here for all metrics: http://scikit-learn.org/stable/modules/model_evaluation.html
```

	precision	recall	f1-score	support
0.0	0.69	0.63	0.66	1096
1.0	0.80	0.84	0.82	1904
avg / total	0.76	0.76	0.76	3000

Cross Validation

We will try a 10-fold cross-validation, to see if the accuracy holds up more rigorously.

```
In [29]: # evaluate the model using 10-fold cross-validation
scores = cross_val_score(LogisticRegression(), X, y, scoring='accuracy', cv=10)
print scores
print scores.mean()
```

```
[ 0.77422577  0.76523477  0.75624376  0.757          0.753          0.771
  0.76776777  0.75475475  0.74874875  0.73373373]
0.758170930071
```

Accuracy is still around 75-75%

```
In [30]: from sklearn.metrics import confusion_matrix
```

Predicting the probability for a link to persist

Let's predict now the probability to decay for a random link not present in the dataset. In the last 4 month the link had: 0.1 cij + dtij + oij + aaij + cvij + frij

```
In [31]: model.predict_proba(np.array([0.1, 0.2, 0.1, 0.5, 0.1, 0.04, 0.3]))
```

```
Out[31]: array([[ 0.53802789,  0.46197211]])
```

The predicted probability to persist is 46%.

3: Communities Detection

Understand the nature of a given network

Find individuals with common interests and/or behavior

Characterize the role of a given individual: does he/she connects different part of a network?

Predictive models, e.g. churn. Our behavior also depend on the behavior of our neighbors

Spreading of information, identify key individuals in a particular community



3: Communities Detection

How do we detect communities in a network?

The hypothesis

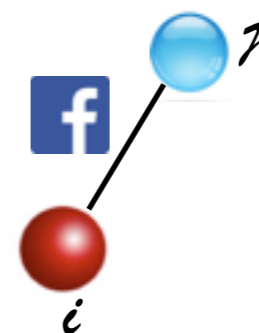
randomly wired networks are not expected to have a community structure

The idea

Modularity is a measure associated to a partition. The partition with the maximum modularity M for a given network offers the optimal community structure!

Method (Louvain) First, the method looks for "small" communities by optimizing modularity locally. Second, it aggregates nodes belonging to the same community and builds a new network whose nodes are the communities. These steps are repeated iteratively until a maximum of modularity is attained and a hierarchy of communities is produced.

Data: user connections from the Facebook
New Orleans network



3: Communities Detection

IP[y]: Notebook communities Last Checkpoint: Mar 24 16:30 (unsaved changes)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Data and Libraries

In this session we will use a simplified version of the facebook-links.txt file which is courtesy of the Max Planck Institute for Software Systems:
<http://socialnetworks.mpi-sws.org/data-wosn2009.html>

File facebook-links.txt contains a list of all of the user-to-user links from the Facebook New Orleans network. These links are undirected on Facebook.

```
In [19]: import networkx as nx
import community
import pylab as plt
import math
import operator
import pandas as pd
# show plots in the notebook
%matplotlib inline
```

```
In [20]: g = nx.read_edgelist('/Users/giovanna/Dropbox/CONFERENCES/Netmob2015/data_facebook/facebook-links-sample20000.dat',
delimter=None, create_using=None,
nodetype=int, data=True, edgetype=None,
encoding='utf-8')
# since we want an undirected graph, we use create_using=None (otherwise use create_using=nx.DiGraph())
```

```
In [21]: type(g)
```

```
Out[21]: networkx.classes.graph.Graph
```

Simple Metrics

```
In [22]: # compute node degrees
degrees = g.degree() # dictionary node:degree
```

```
In [23]: #nx.degree_histogram(g)
#figure(figsize=(6,5))
# uses numpy and matplotlib
# plt.figure()
plt.plot(nx.degree_histogram(g), 'ro-', markersize=8)
plt.legend(['Degree'])
plt.xlabel('Degree')
plt.ylabel('Number of nodes')
plt.title('Facebook New Orleans network')
plt.xscale('log')
plt.yscale('log')
# plt.savefig('Facebook_degree_distribution.pdf')
plt.show()
```

3: Communities Detection

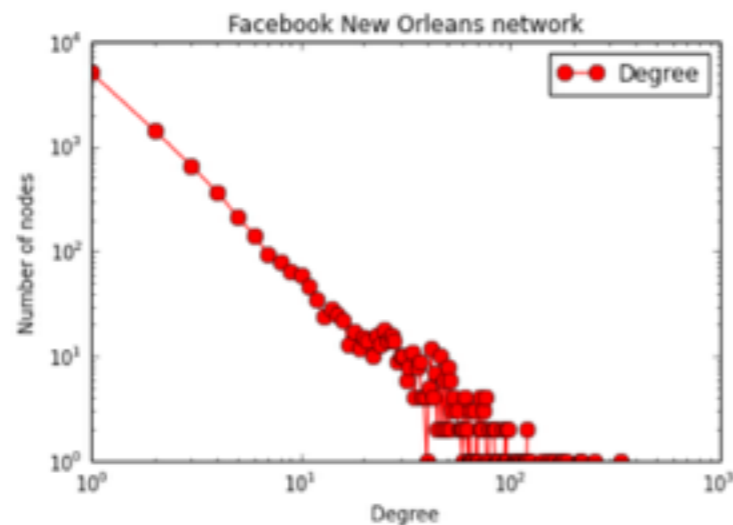
IP[y]: Notebook

communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [23]: #nx.degree_histogram(g)
#figure(figsize=(6,5))
# uses numpy and matplotlib
# plt.figure()
plt.plot(nx.degree_histogram(g), 'ro-', markersize=8)
plt.legend(['Degree'])
plt.xlabel('Degree')
plt.ylabel('Number of nodes')
plt.title('Facebook New Orleans network')
plt.xscale('log')
plt.yscale('log')
# plt.savefig('Facebook_degree_distribution.pdf')
plt.show()
```



```
In [24]: # Clustering coefficient
```

```
In [25]: # Clustering coefficient of nodes 1 and 22
print nx.clustering(g, 1)
print nx.clustering(g, 22)
```

```
0.0276923076923
0.254545454545
```

```
In [26]: # Clustering coefficient of all nodes (dictionary)
clust_coefficients = nx.clustering(g)
```


3: Communities Detection

IP[y]: Notebook

communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

`clust_coefficients = nx.clustering(g)`In [27]: *# Clustering coefficient of each node can be recovered*

```
print clust_coefficients.values()[0]
print clust_coefficients.values()[21]
```

```
0.0276923076923
0.254545454545
```

In [28]: *# Average clustering coefficient*

```
avg_clust = sum(clust_coefficients.values()) / len(clust_coefficients)
print avg_clust
```

```
0.1155548385
```

In [29]: **def** highest centrality(centr_dict):

```
    centr_items = centr_dict.items()
    centr_items = [(b,a) in centr_items]
    centr_items.sort()
    centr_items.reverse()
    return centr_items[0][1]
```

In [30]: *# a bit slow with this graph*

```
# betw_cent = nx.betweenness centrality(g)
# print highest centrality(betw_cent) # print the node with highest betweenness centrality
```

General properties

Other measures available:

- Diameter
- Radius
- Average Shortest Path Length
- Average Clustering Coefficient
- Average Degree
- Degree Distribution
- Degree Centrality
- Closeness Centrality
- Node Betweenness Centrality
- Edge Betweenness Centrality

3: Communities Detection

IP[y]: Notebook communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Largest Connected Component

A connected component is a set of nodes in which each pair of nodes is connected by a path.

```
In [31]: # get the largest connected component of G
cc = [(c, len(c)) for c in nx.connected_components(g)]
maxlenc = max([x[1] for x in cc])
for x in cc:
    if x[1] == maxlenc:
        maxcc = x[0]
h = g.subgraph(maxcc)
```

```
In [32]: print len(g.nodes())
print len(g.edges())

8771
18054
```

```
In [33]: print len(h.nodes())
print len(h.edges())

8713
18010
```

```
In [34]: g_components = nx.connected_components(g)
[ len(c) for c in g_components ]
```

```
Out[34]: [8713, 3, 7, 2, 3, 3, 5, 6, 3, 7, 5, 3, 2, 4, 2, 3]
```

In this case we can say that the network has one giant component with 8713 nodes.

Community Detection

```
In [35]: # Font: http://nbviewer.ipython.org/github/BrusselsDataScienceCommunity/SNA/blob/master/Using_NetworkX_for_graphs.ipynb
# Use the Louvain method on h
partition = community.best_partition(h) # the output is a dictionary {node:community_number}
```

```
In [36]: # transform this output to a list of list of nodes
commlist = list(set([partition[x] for x in partition]))
comm = [[] for c in commlist]
for n in h.nodes():
    comm[partition[n]].append(n)
```

Have a look at the size of each community

```
In [37]: 'Sizes of the communities: ' + ' '.join(['comm'+ str(i) + ':' + str(len(comm[i])) for i in commlist])
```

```
Out[37]: 'Sizes of the communities: comm0:454 comm1:508 comm2:360 comm3:485 comm4:313 comm5:420 comm6:152 comm7:307 comm8:428 comm9:
311 comm10:320 comm11:191 comm12:178 comm13:97 comm14:297 comm15:198 comm16:790 comm17:77 comm18:874 comm19:335 comm20:159
comm21:67 comm22:146 comm23:232 comm24:580 comm25:7 comm26:168 comm27:89 comm28:164 comm29:6'
```

3: Communities Detection

IP[y]: Notebook

communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

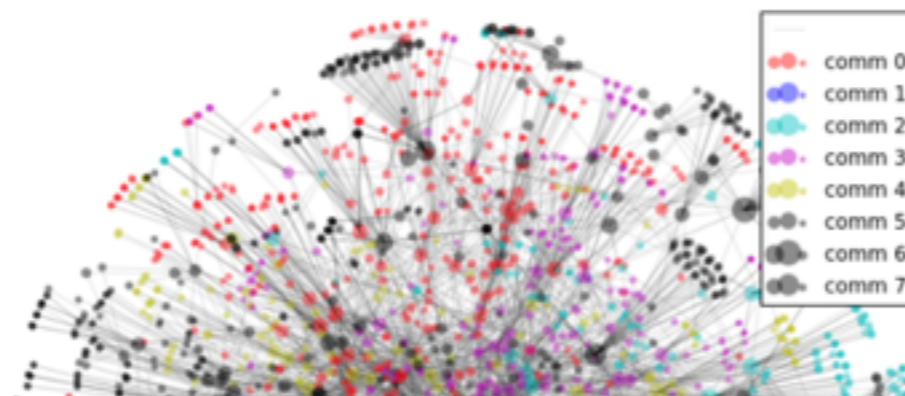
Visualize the graph and the communities

```
In [38]: # Combine the pageranks from the different communities into a single dictionary:
comm_prk = {}
for c in comm:
    comm_prk.update(nx.pagerank(h.subgraph(c)))
```

```
In [39]: #Compute graph layout : TIME CONSUMING (if selecting all nodes)
subset = comm[0]+comm[1]+comm[2]+comm[3]+comm[4]+comm[5]+comm[6]+comm[7]
h2 = h.subgraph(subset)
pos = nx.spring_layout(h2)
```

```
In [40]: showcomm = [0,1,2,3,4,5,6,7]
nodelist = []
comm_to_col = {0:'r',1:'b',2:'c',3:'m',4:'y',5:'k',6:'k',7:'k'}

# create plot
plt.figure(figsize=(10,9))
nx.draw_networkx_edges(h2,pos,alpha=.1)
for i in showcomm:
    nodelist = comm[i]
    colors = [comm_to_col[partition[i]] for i in nodelist]
    sizes = [600*math.sqrt(comm_prk[i]) for i in nodelist]
    node_comm = [partition[i] for i in nodelist]
    nx.draw_networkx_nodes(h2,pos,nodelist = nodelist,linewidths=0,node_size=sizes,node_color=colors,alpha=.45)
plt.legend(['']+['comm '+ str(i) for i in showcomm])
plt.xlim(0,1)
plt.ylim(0,1)
plt.axis('off')
plt.show()
```



3: Communities Detection

IP[y]: Notebook

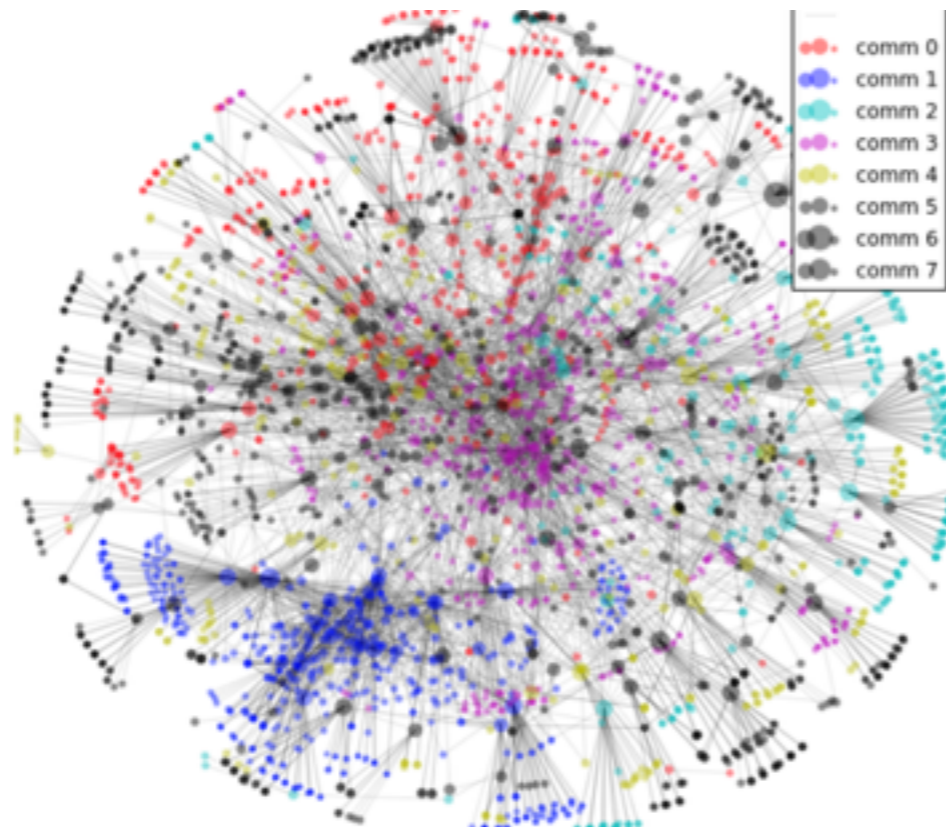
communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [40]: showcomm = [0,1,2,3,4,5,6,7]
nodelist = []
comm_to_col = {0:'r',1:'b',2:'c',3:'m',4:'y',5:'k',6:'k',7:'k'}

# create plot
plt.figure(figsize=(10,9))
nx.draw_networkx_edges(h2,pos,alpha=.1)
for i in showcomm:
    nodelist = comm[i]
    colors = [comm_to_col[partition[i]] for i in nodelist]
    sizes = [600*math.sqrt(comm_prk[i]) for i in nodelist]
    node_comm = [partition[i] for i in nodelist]
    nx.draw_networkx_nodes(h2,pos,nodelist = nodelist,linewidths=0,node_size=sizes,node_color=colors,alpha=.45)
plt.legend(['']+[ 'comm ' + str(i) for i in showcomm])
plt.xlim(0,1)
plt.ylim(0,1)
plt.axis('off')
plt.show()
```



3: Communities Detection

IP[y]: Notebook communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

Use node metrics to compute the "top influencers"

```
In [41]: def get_top_influencers(G, top=10):
         prk = nx.pagerank(G)
         sprk = sorted(prk.items(), key=operator.itemgetter(1), reverse=True)
         toplabels = [x[0] for x in sprk[:top]]
         return toplabels
```

```
In [42]: get_top_influencers(h) # top 10 influencers in the entire h
```

```
Out[42]: [5030, 3580, 4956, 3131, 2294, 94, 2186, 4836, 682, 4905]
```

```
In [43]: data = {'community_id': [i for i in commlist],
                'size': [len(comm[i]) for i in commlist],
                }
         communities = pd.DataFrame(data, columns=['community_id', 'size'])
         #print communities
```

```
In [44]: # order communities by their size
         top_communities = communities.sort(['size'], ascending=0).head(10)
         print top_communities
```

	community_id	size
18	18	874
16	16	790
24	24	580
1	1	508
3	3	485
0	0	454
8	8	428
5	5	420
2	2	360
19	19	335

```
In [45]: get_top_influencers(h.subgraph(18))
```

```
Out[45]: [18]
```

3: Communities Detection

IP[y]: Notebook communities Last Checkpoint: Mar 24 16:30 (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

```
In [46]: for i in top_communities:
         print 'Community', i, 'has', len(c), 'members and its top influencers are:'
```

Community community_id has 6 members and its top influencers are:
Community size has 6 members and its top influencers are:

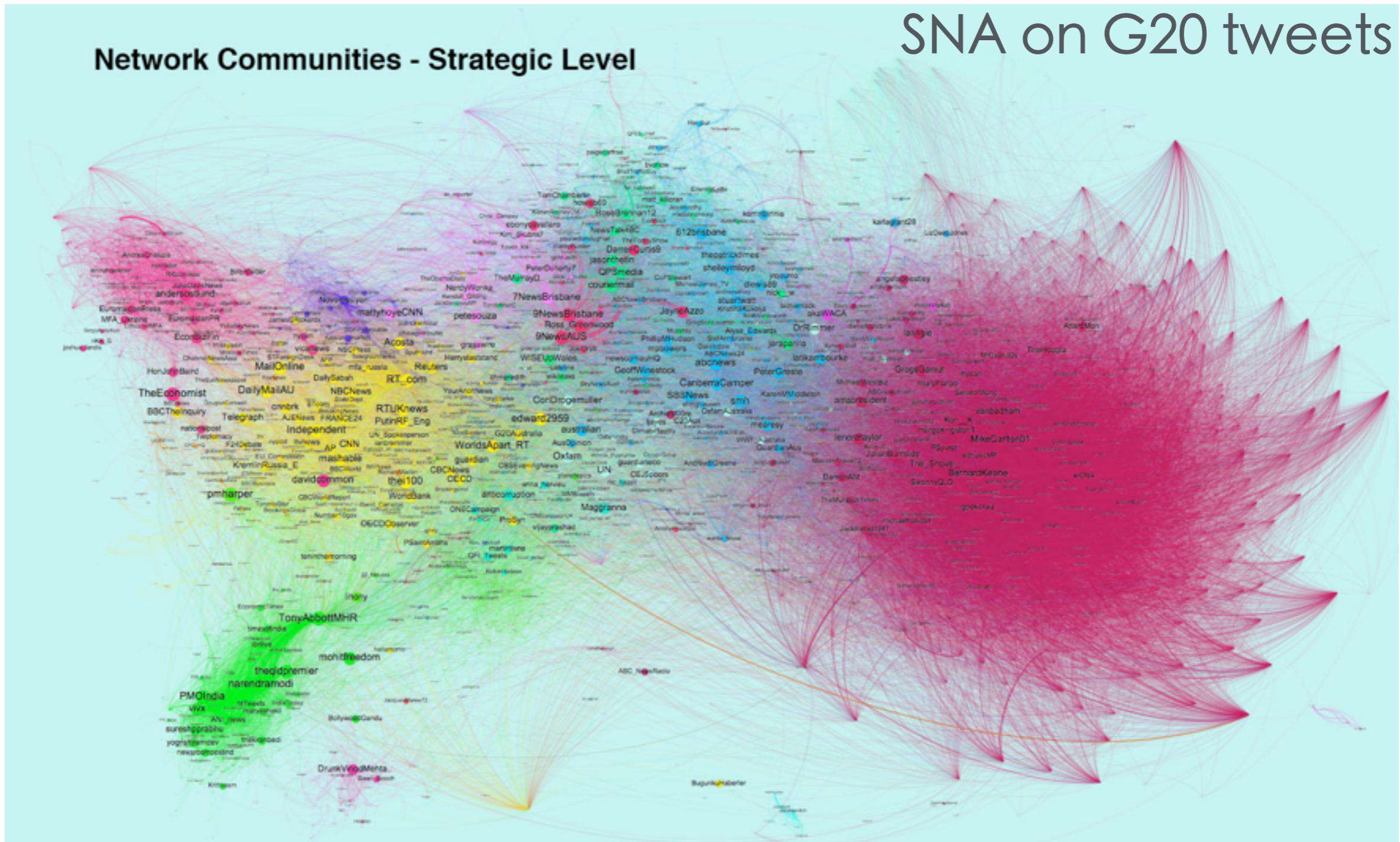
```
In [47]: for idx, c in enumerate(comm):
         print 'Community', idx, 'has', len(c), 'members and its top influencers are:'
         print get_top_influencers(h.subgraph(c))
```

Community 0 has 454 members and its top influencers are:
[165, 161, 1640, 1234, 39, 814, 2829, 812, 813, 2397]
Community 1 has 508 members and its top influencers are:
[94, 2296, 89, 1749, 513, 93, 86, 571, 1983, 1201]
Community 2 has 360 members and its top influencers are:
[858, 237, 4547, 4235, 4782, 2757, 2521, 3151, 2017, 3152]
Community 3 has 485 members and its top influencers are:
[738, 242, 245, 243, 1150, 232, 4325, 740, 1040, 3780]
Community 4 has 313 members and its top influencers are:
[1284, 1286, 1407, 1173, 551, 4483, 4503, 4775, 2071, 2551]
Community 5 has 420 members and its top influencers are:
[2072, 2512, 3296, 3816, 4829, 7935, 2508, 6527, 4495, 2509]
Community 6 has 152 members and its top influencers are:
[2882, 454, 4407, 453, 2905, 4076, 998, 4197, 455, 2636]
Community 7 has 307 members and its top influencers are:
[4679, 286, 4646, 4957, 4651, 210, 5247, 4677, 4725, 4497]
Community 8 has 428 members and its top influencers are:
[2074, 4616, 477, 3153, 2095, 1413, 292, 170, 1278, 2988]
Community 9 has 311 members and its top influencers are:
[682, 4901, 5546, 991, 1808, 3806, 4581, 61, 2722, 758]
Community 10 has 320 members and its top influencers are:
[303, 1686, 5316, 2476, 3070, 2117, 1412, 1689, 2398, 3103]
Community 11 has 191 members and its top influencers are:
[6836, 4814, 1660, 1659, 5707, 2053, 3350, 6281, 1677, 67]
Community 12 has 178 members and its top influencers are:
[4578, 1015, 6330, 3333, 4721, 5589, 4462, 5774, 6334, 4972]
Community 13 has 97 members and its top influencers are:
[8020, 74, 73, 4060, 4061, 207, 2702, 85, 4062, 4063]
Community 14 has 297 members and its top influencers are:
[476, 1590, 2014, 1709, 1708, 3450, 2016, 5738, 1553, 2015]
Community 15 has 198 members and its top influencers are:
[4549, 6141, 6140, 6142, 285, 5009, 162, 6164, 6149, 6157]
Community 16 has 790 members and its top influencers are:
[4591, 1082, 1080, 5704, 4424, 2399, 750, 4834, 1081, 4423]
Community 17 has 77 members and its top influencers are:
[3972, 3973, 284, 3978, 3983, 4002, 3423, 3977, 4000, 4003]
Community 18 has 874 members and its top influencers are:

3: Communities Detection

SNA on G20 tweets

Network Communities - Strategic Level



Colour - indicates community as detected by the community detection algorithm.

Node Size - determined by the Page Rank of the User. **Page Rank** is an algorithm that calculates (recursively) how important a node is by reference to how important the nodes it is connected to are.

Connecting Lines - each line indicates that the User has Retweeted another User's Tweet. The thicker the line the more times this has happened between the two Users. The curve of the line indicates which way the Retweet occurred; the curve is anti-clockwise from Source Tweeter node to Retweeter node.

@swainjo

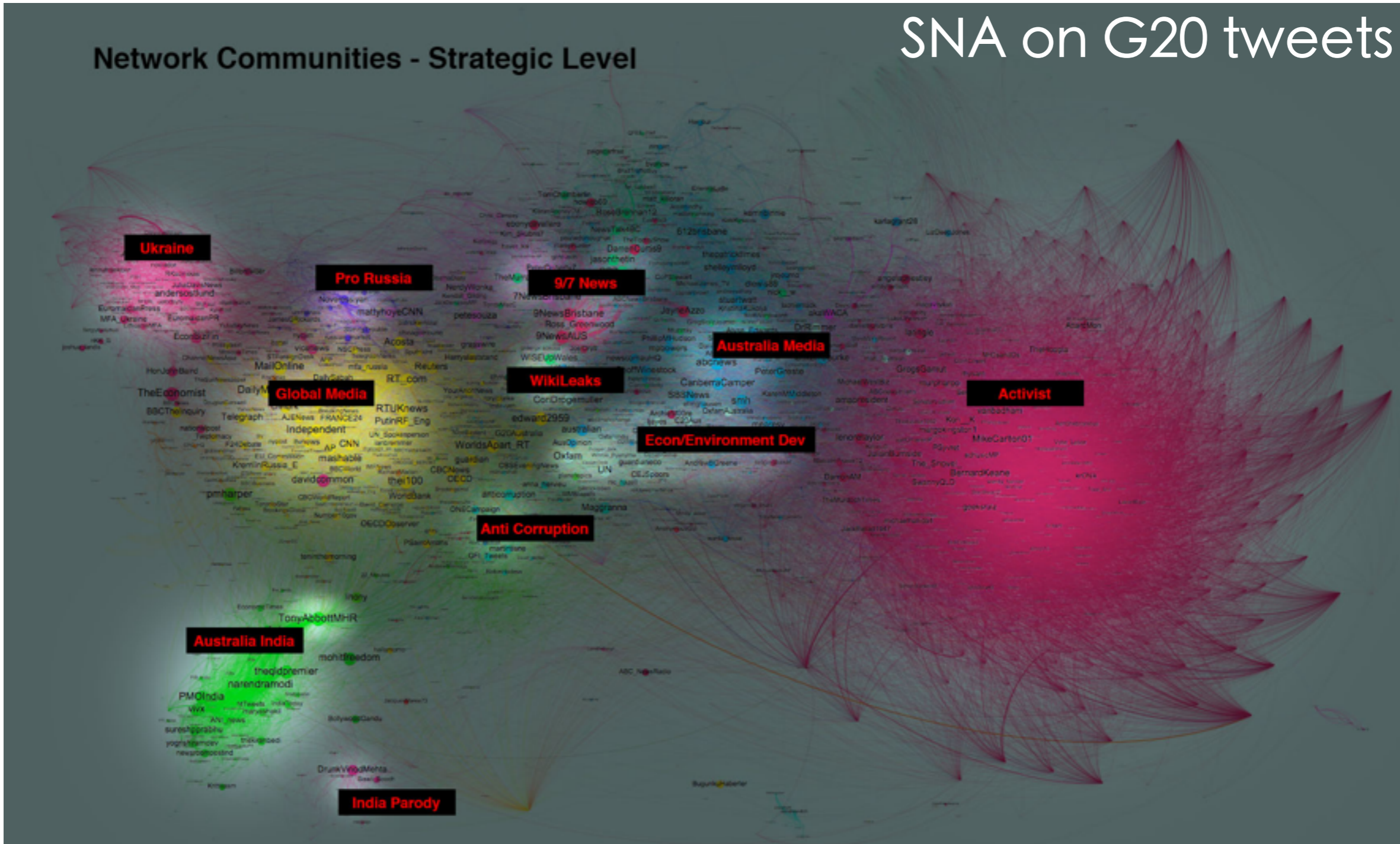
<http://swainworld.ghost.io>

@gmitello

3: Communities Detection

SNA on G20 tweets

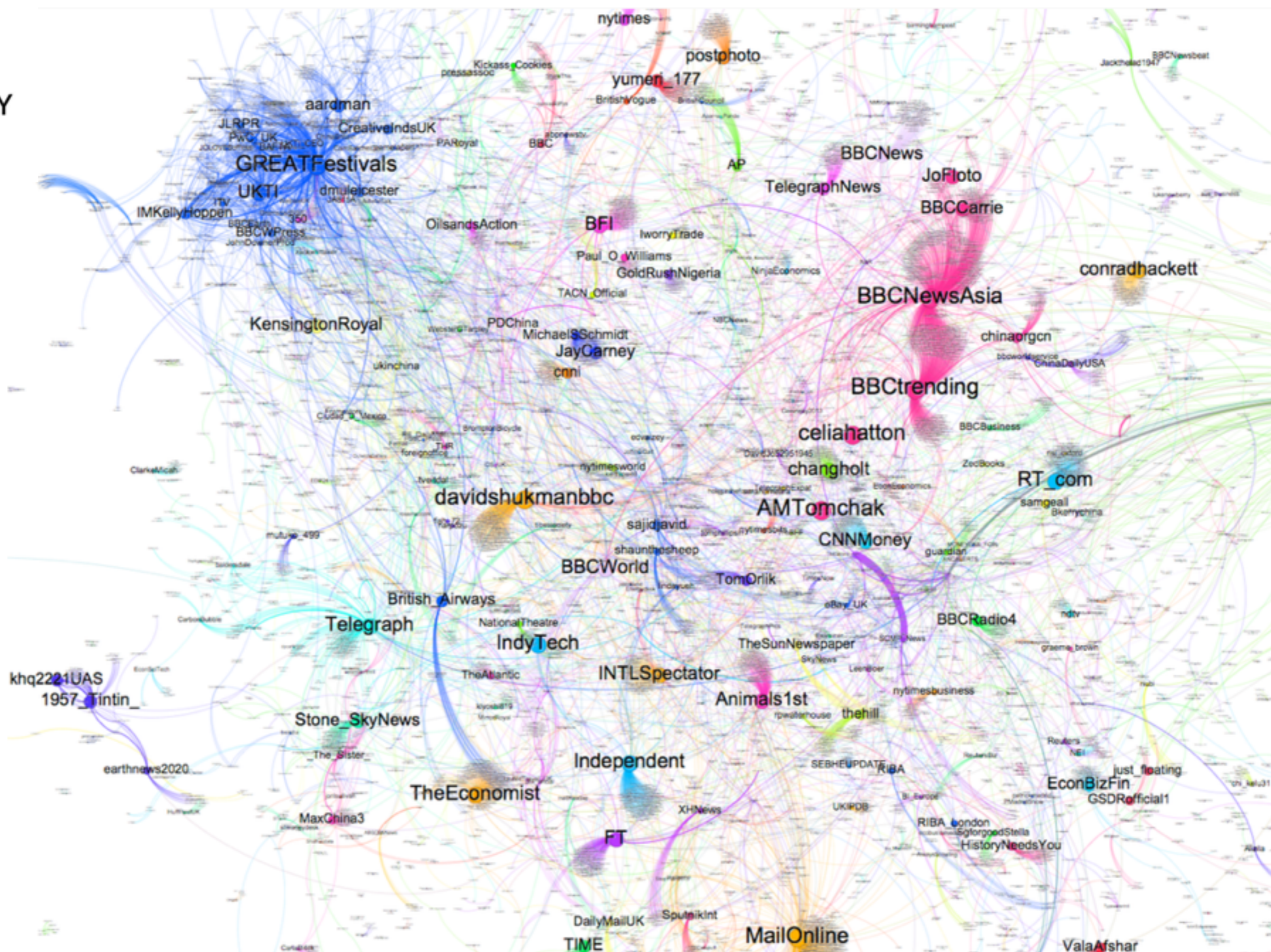
Network Communities - Strategic Level



- Influence and opinion phenomena
- Interests
- Information spreading
- Key connectors

@swainjo
<http://swainsworld.ghost.io>

**GREAT
FESTIVAL
OF
CREATIVITY**
Shanghai
2-4 March 2015



@swainjo

<http://swainsworld.ghost.io>

@gmitello

Tools for SNA and Visualization

IP[y]: IPython Interactive Computing

- SciPy
- NumPy
- NetworkX
- Communities
- scikit-learn

```

IP[y]: Notebook communities Last Checkpoint: Mar 24 16:30 (unsaved changes)
File Edit View Insert Cell Kernel Help
[Navigation icons] Code

Data and Libraries
In this session we will use a simplified version of the facebook-links.txt file which is courtesy of the Max Planck Institute for Software Systems:
https://socialnetworks.mpi-sws.org/data-wos2009.html
File facebook-links.txt contains a list of all of the user-to-user links from the Facebook New Orleans network. These links are undirected on Facebook.

In [19]: import networkx as nx
import community
import pylab as plt
import math
import operator
import pandas as pd
# show plots in the notebook
matplotlib inline

In [20]: g = nx.read_edgelist('/Users/giovanna/Dropbox/CONFERENCES/Netmob2015/data_facebook/facebook-links-sample20000.dat',
# since we want an undirected graph, we use create_using=None (otherwise use create_using=nx.DiGraph))
encoding='utf-8')

In [21]: type(g)
Out[21]: networkx.classes.graph.Graph

Simple Metrics
In [22]: # compute node degrees
degrees = g.degree() # dictionary node:degree

In [23]: # nx.degree_histogram(g)
# figure(figsize=(6,5))
# uses numpy and matplotlib
# plt.figure()
plt.plot(nx.degree_histogram(g), 'ro-', markersize=8)
plt.legend(['degree'])
plt.xlabel('degree')
plt.ylabel('Number of nodes')
plt.title('Facebook New Orleans network')
plt.yscale('log')
# plt.savefig('Facebook_degree_distribution.pdf')
plt.show()
    
```

+ ggplot for python

R Studio

- igraph
- sna
- network
- statnet
- plyr

```

chunks.Rmd
1 R Code Chunks
2
3 With R Markdown, you can insert R code
4 chunks including plots:
5
6 ""[<= rplot, fig.width=4, fig.height=3,
7 message=FALSE]
8 # quick summary and plot
9 library(ggplot2)
10 summary(cars)
11 qplot(speed, dist, data=cars) +
12 geom_smooth()
13
RStudio - Preview HTML
Preview: ./chunks.html Save As Publish
R Code Chunks
With R Markdown, you can insert R code chunks including plots:
# quick summary and plot
library(ggplot2)
summary(cars)
##      speed      dist
## Min.   : 4.8   Min.   :  2
## 1st Qu.:12.0   1st Qu.: 26
## Median:15.0   Median: 36
## Mean   :15.4   Mean   : 43
## 3rd Qu.:19.0   3rd Qu.: 56
## Max.   :25.0   Max.   :129
qplot(speed, dist, data = cars) + geom_smooth()
    
```

+ ggplot2
googleVis, gmap



References



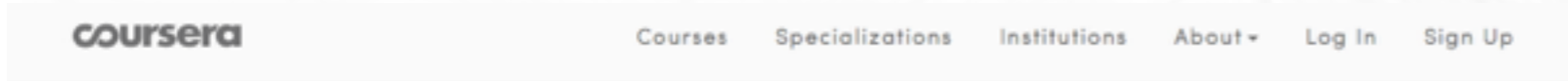
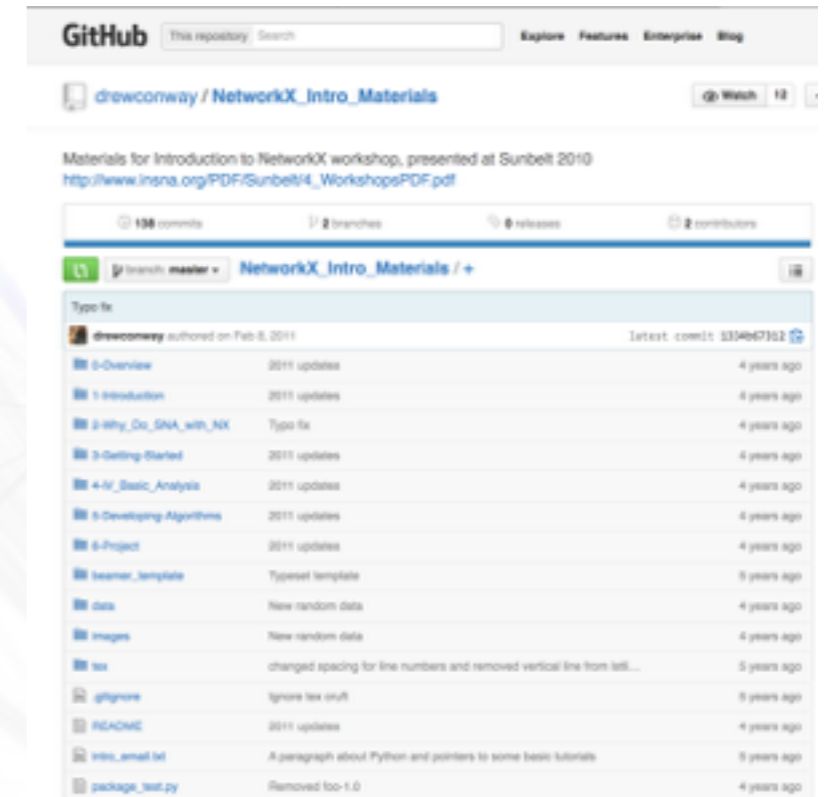
Welcome to the Network Science course website. Please tune in for updates, messages, homework assignments and useful links.

The lectures will take place on Mondays, Wednesdays, and occasionally on Fridays, 5:30 – 7:20 PM. Check the agenda weekly for Friday classes!

At the Center for Complex Network Research
 Northeastern University Physics Department
 810 Forsyth Street, Boston, MA 02115
 Hint: It's on the 5th floor of the Charles A. Dana building.
 Use the left elevator for 5th floor access.

The course will be presented by Prof. Albert-László Barabási and assisted by Dr. Roberta Sinatra

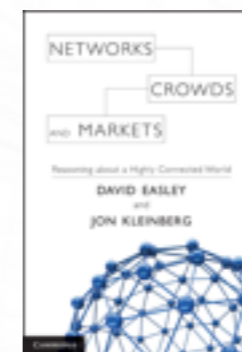
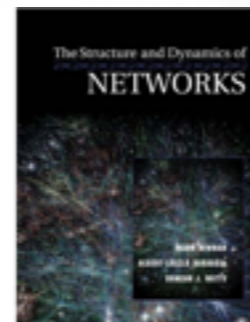
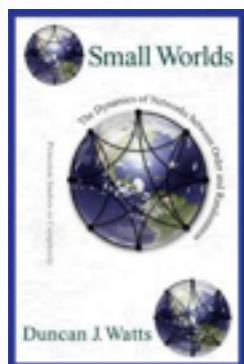
[Syllabus](#) [Class Agenda](#) [OpenRev Portal](#)



Social Network Analysis

This course will use social network analysis, both its theory and computational tools, to make sense of the social and information networks that have been fueled and rendered accessible by the internet.

Instructors



Shameless Plug



Everything you always wanted to know about network science but were afraid to ask

NetSci 2015 Satellite, Jun 2, Zaragoza

[Learn more](#) The NetSci Backstage satellite at **NetSci 2015** will gather leading experts from network and data sciences to explore the means to conduct good data-driven science, presenting and discussing scientific results, applications and hands-on best practices on how to deal with statistics, tools, data, and how to correctly reach and present conclusions, bridging academia and industry. Got a question? Contact us at: netscibackstage@gmail.com

[Register now!](#) [Submit now!](#)

Organizers

For any questions about the satellite, please contact us at: netscibackstage@gmail.com



Paul Expert
King's College



Giovanna Miritello
[@gmiritello](#)
Zed Worldwide



Giovanni Petri
[@lordgrilo](#)
ISI Foundation



Claudia Sinatra
Space Agency



Roberta Sinatra
[@robysinatra](#)
Northeastern University



Michael Szell
[@mszell](#)
Northeastern University

Submit your abstract until April 14th!

Social Networks

Crash Course



Giovanna Miritello

 @gmiritello

