Computational Social Network Analysis

Social Network Analysis: Essentials from Graph Theory

Part 1/3

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

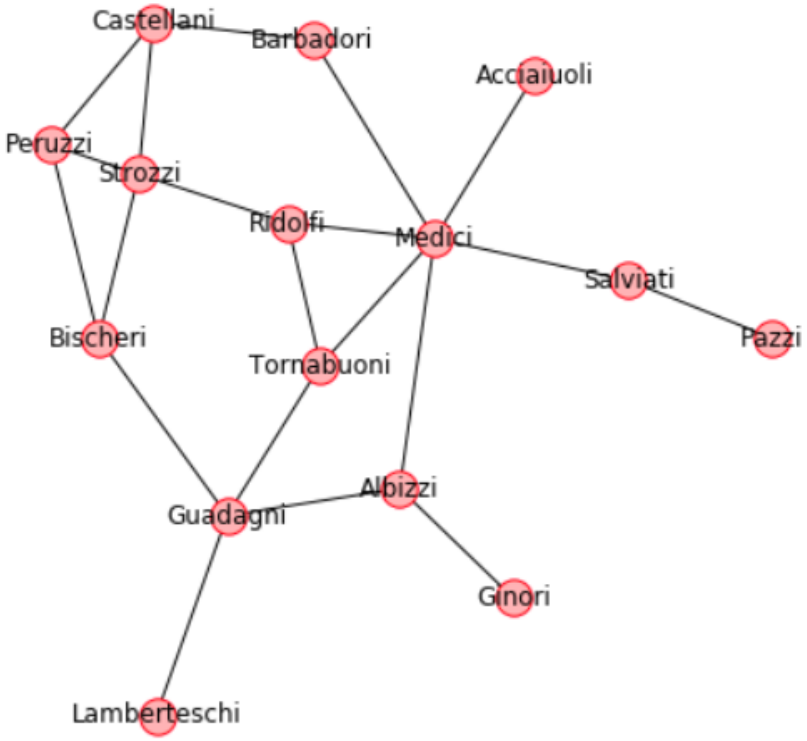
Linköping University, Norrköping, Sweden

Spring 2019

What is a Graph/Network?

- **Graph/Network** is an object of *Graph Theory* consisting of:
  - a finite number of **nodes** (or **points** or **vertices** or **actors**) and
  - a finite number of **edges** (or **links** or **connections** or **ties**) which join pairs of nodes
- Nodes are denoted (labelled) by *numbers* or *names*, as e.g.:
  - $V = \{0, 1, 2, 3, 4\}$ or
  - $V = \{\text{Mary}, \text{Jack}, \text{Helen}\}$
- Edges are denoted by *pairs of nodes* (in round brackets). For an edge $e = (u, v)$, nodes u, v form the *end points* of e. An edge $e = (u, u)$ is a **self–loop**.
  - $E = \{(0, 1), (0, 3), (1, 4), (2, 3)\}$ or
  - $E = \{(\text{Mary}, \text{Jack}), (\text{Jack}, \text{Helen}), (\text{Helen}, \text{Mary})\}$
- Nodes might represent:
  - persons or things
  - groups of various entities (like organizations etc.)
  - events or any type of matters
- Edges might represent:
  - relation(ship)s or interactions
  - exchanges, shares or any type of transactions
  - associations, affiliations, attachments, encounters etc.

The graph of Florentine Families



Basic Terminology

- **Undirected** is a graph in which all existing edges are symmetrical in the sense that, if (u, v) is an edge, so is (v, u) and (v, u) is considered to be identical to (u, v).
- **Directed** is a graph in which, if (u, v) is an edge, then (v, u) may or may not be an edge of the graph. If both (u, v) and (v, u) happen to be edges, then they are called **reciprocal**.
- **Simple** is a graph such that between any two nodes u and v only one edge (u, v) can exist (with the same order in the directed graph case).
- **Multigraph** is a graph such that between any two nodes u and v there can exist multiple edges (u, v) (at least one edge).
- **Weigthed** is a graph such that to any edge a non-zero numerical value (called *edge weight*) is assigned.
- **Bipartite** (**2-mode network**) is a graph whenever its nodes are partitioned in two sets (**modes**) such there can only exist edges between nodes across these sets but never inside the same set.

Neighboring

- In a graph, two nodes u and v are **adjacent** if they are the end points of an edge.
- In an undirected graph, if two nodes u and v are adjacent, then each one of them is a **neighbor** of the other one. Furthermore, the sum of the number of neighbors of a node is the **degree** of this node.
- In a directed graph, given two adjacent nodes u and v such that (u, v) is an edge (directed from u to v):
  - node v is called **outgoing–neighbor** of node u and
  - node u is called **incoming–neighbor** of node v.

  Furthermore, the sum of the number of outgoing-neighbors of a node is the **out-degree** of this node and the sum of the number of incoming-neighbors of a node is its **in-degree**.

Adjacency matrix

- If the set of nodes of a simple graph is the set $V = \{1, 2, \ldots, n\}$, the **adjacency matrix** of this graph is a $n \times n$ matrix A of 0's and 1's such that, for any $i, j \in V$, $A_{ij} = 1$, whenever nodes i and j are adjacent, and $A_{ij} = 1$, otherwise. Notice that if the graph is undirected, then its adjacency matrix is symmetric, while, if it is directed, its adjacency matrix could be either symmetric or non-symmetric.
- The adjacency matrix of a weighted (simple) graph on the set $V = \{1, 2, \ldots, n\}$ is defined as a real–valued matrix A such that, for any $i, j \in V$, A_{ij} is the weight of the edge (i, j), whenever nodes i and j are adjacent, while $A_{ij} = 0$, otherwise.
- The adjacency matrix of a (weighted) multigraph is defined in the same way by setting A_{ij} equal to the number of multiple edges (sum of their weights), whenever nodes i and j are adjacent, while $A_{ij} = 0$, otherwise.

Paths and distances

- In a graph, given a sequence of distinct nodes $v_1, v_2, \ldots, v_k$ (with $k \geq 2$) such that $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$ are all edges, then this sequence of edges is said to be a **path** in the graph (starting) from node v_1 (and ending) to node v_k.
- In a directed graph, a **directed path** is again a sequence of edges which connect a sequence of nodes, but with the added restriction that the edges are all directed in the same direction.
- In a graph, the **length** of a path is the number of edges in the sequence of edges connecting a sequence of nodes. A path between two nodes u and v is called **geodesic** if it has the minimum length among all possible paths between these nodes. Moreover, in an undirected (directed) graph, the **distance** $d(u, v)$ between two nodes u and v is the length of the geodesic (directed) path between these nodes.
- The **diameter** of a graph is the greatest distance between any pair of nodes.

Connectedness

- An undirected graph is **connected** when there is a path between every pair of nodes. A graph that is not connected is **disconnected**.
- A directed graph is called **weakly connected** if rescinding the direction of edges produces a connected (undirected) graph. It is **strongly connected** if it contains a directed path from u to v and a directed path from v to u for every pair of nodes u, v.
- In an undirected (directed) graph, a **cycle** is a set of (at least 3 (2)) nodes such that there is a (directed) path between every two of them.
- An undirected graph is a **tree** if between any two nodes there exists *exactly one* path. Every acyclic connected graph is a tree, and vice versa.

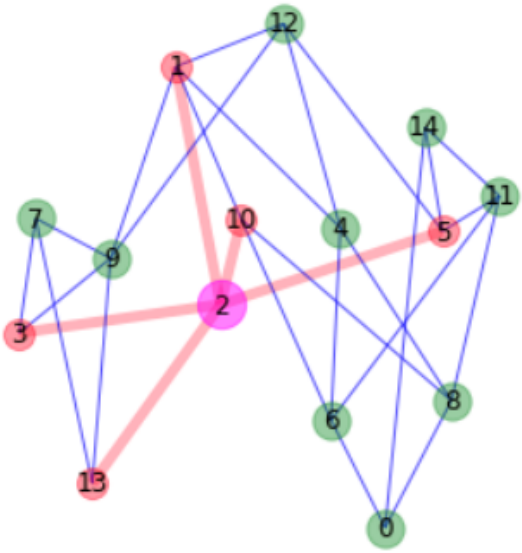
More definitions

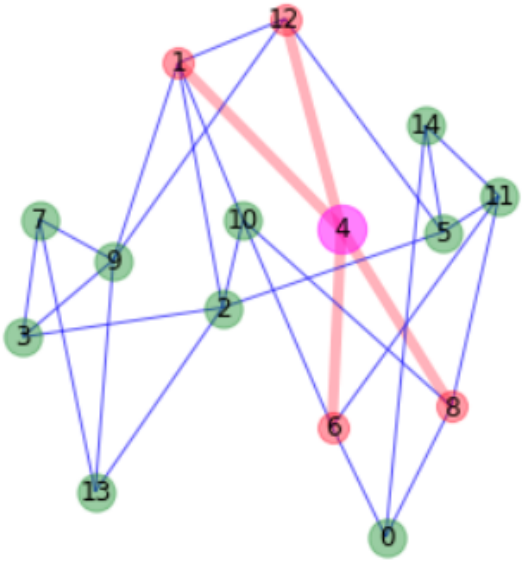
- A graph without edges and with at least one node is called **trivial graph**. A graph without any edges and without any nodes is called **empty graph**.

- A graph is **complete** if it contains all possible edges between any pair of its nodes.
  - In a complete undirected graph without any self–loops and with n nodes, there are $\frac{1}{2}n(n-1)$ edges.
  - In a complete directed graph without any self–loops and with n nodes, there are $n(n-1)$ edges.

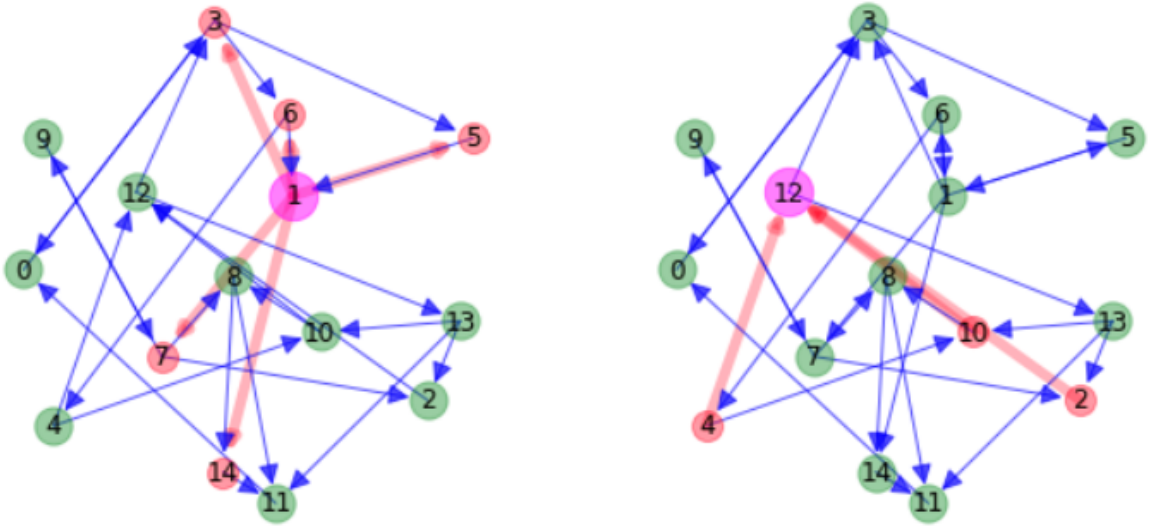
- The **density** of a graph with n (> 1) nodes and m edges is defined as the ratio:
  - $\frac{2m}{n(n-1)}$, when the graph is undirected, and
  - $\frac{m}{n(n-1)}$, when the graph is directed

An undirected graph

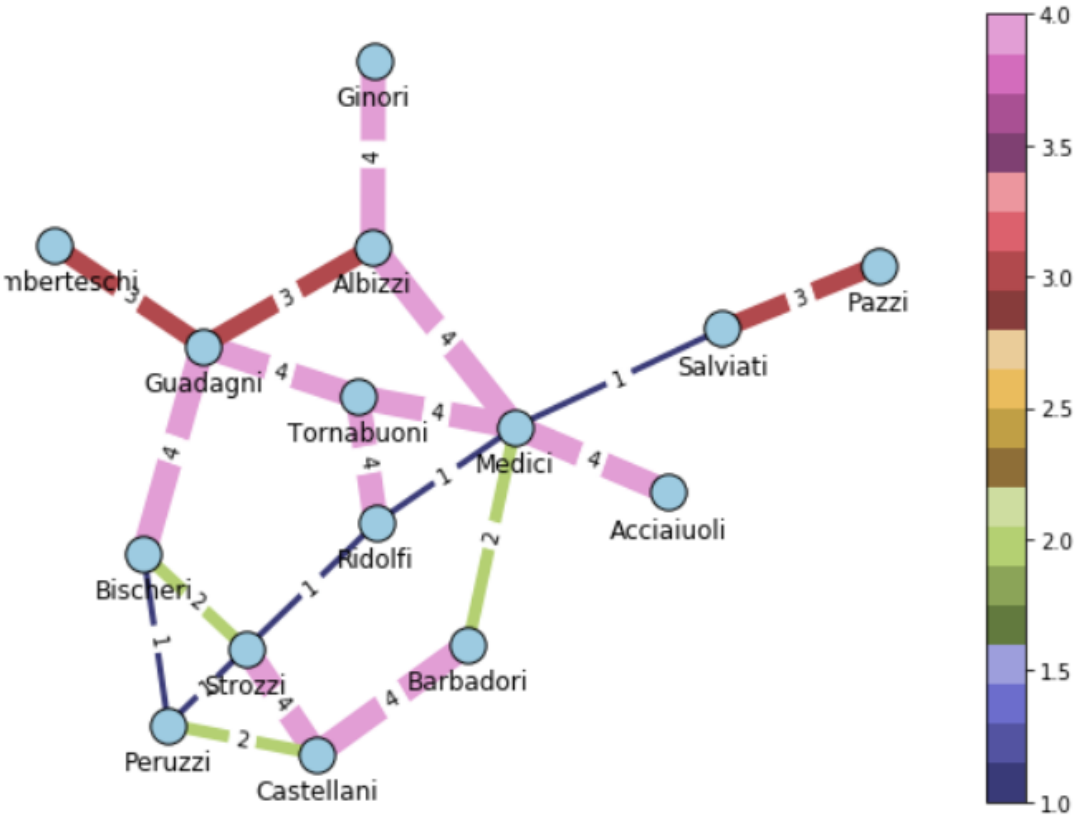




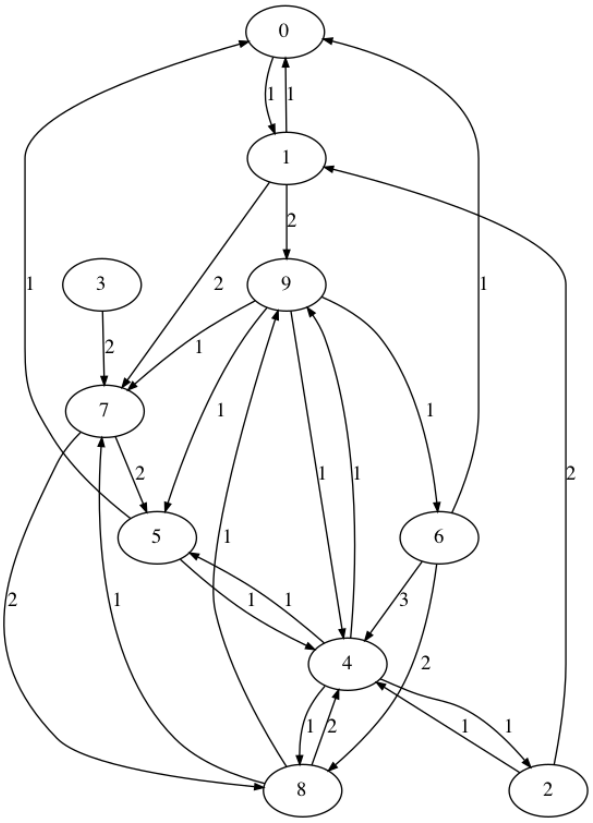
A directed graph



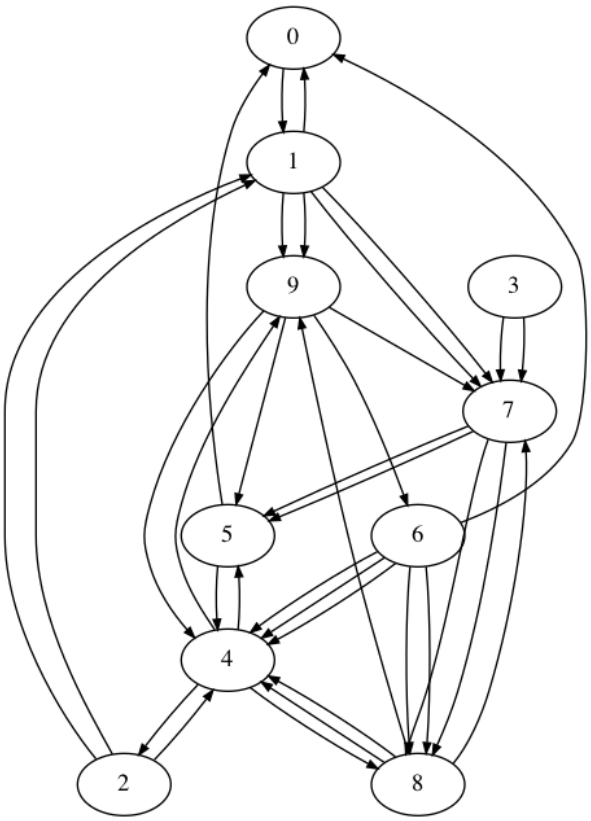
A weighted undirected graph



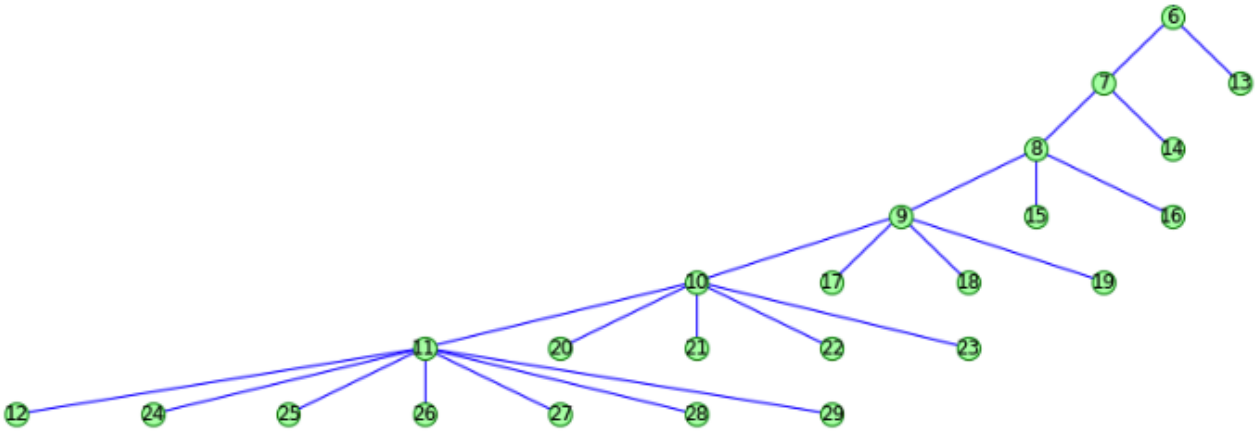
A weighted directed graph



A weighted directed multigraph



A tree



A bipartite graph

Computational Social Network Analysis

Social Network Analysis: Essentials from Graph Theory

Part 2/3

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

Linköping University, Norrköping, Sweden

Spring 2019

Doing network analysis using Python's NetworkX

1. NetworkX Documentation Pages:
   https://networkx.github.io/documentation/stable/
2. Importing main modules:

```
import networkx as nx
import matplotlib.pyplot as plt
```

3. Importing plotting modules (*optional*):

```
import pygraphviz
from networkx.drawing.nx_agraph import graphviz_layout
from networkx.drawing.nx_agraph import to_agraph
```

4. Creating empty undirected/directed graphs/multigraphs:

```
G = nx.Graph()
G = nx.DiGraph()
G = nx.MultiGraph()
G = nx.MultiDiGraph()
```

5. Adding nodes:

```
G.add_node(node) # node is an integer or a string
G.add_nodes_from(node_list) # node_list is a list
```

6. Removing nodes:

```
G.remove_node(node)
G.remove_nodes_from(node_list)
```

7. Adding edges:

```
G.add_edge(i,j) # i,j are nodes, w weight value
G.add_edge(i,j,weight=w) # w is the weight value
G.add_edges_from(edge_list)
# edge_list is a list with elements 2-tuples (i,j)
G.add_edges_from(edge_list)
# edge_list is a list with elements 3-tuples (i,j,w)
```

8. Removing edges:

```
G.remove_edge(i,j)
G.remove_edge(i,j,weight=w)
G.remove_edges_from(edge_list) # node_list is a list
```

9. Setting/getting graph node and edge attributes:

```
nx.set_node_attributes(G, dictionary, 'name')
nx.set_edge_attributes(G, dictionary, 'name')
nx.get_node_attributes(G, 'name')
nx.get_edge_attributes(G, 'name')
```

10. Inspecting graph nodes and edges:

```
G.nodes()
G.nodes(data=True) # to view node attributes too
G.edges()
G.egges(data=True) # to view edge attributes too
```

11. Checking type of graph:

```
nx.is_directed(G)
G.is_multigraph()
nx.is_weighted(G)
nx.is_bipartite(G)
nx.is_tree(G)
nx.is_connected(G)
nx.is_strongly_connected(G)
nx.is_weakly_connected(G)
```

12. Adjacency related:

```
nx.neighbors(G, node)
nx.isolates(G)
nx.predecessors(node)
nx.successors(node)
G.degree()
G.in_degree()
G.out_degree()
nx.adjacency_matrix(G)
```

13. Graph generators:

https://networkx.github.io/documentation/stable/reference/generators.html

14. Various graph substructures:

```
nx.all_simple_paths(G, source=i, target=j)
list(nx.shortest_simple_paths(G, i, j))
nx.subgraph(G, node_list)
nx.edge_subgraph(G, edge_list)
nx.find_cliques(G)
nx.number_connected_components(G)
nx.connected_components(G)
nx.number_strongly_connected_components(G)
nx.strongly_connected_components(G)
nx.number_weakly_connected_components(G)
nx.weakly_connected_components(G)
nx.find_cycle(G)
```

15. Drawing graphs:

https://networkx.github.io/documentation/stable/reference/drawing.html

Computational Social Network Analysis

Social Network Analysis: Essentials from Graph Theory

Part 3/3

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

Linköping University, Norrköping, Sweden

Spring 2019

Assortativity in NetworkX

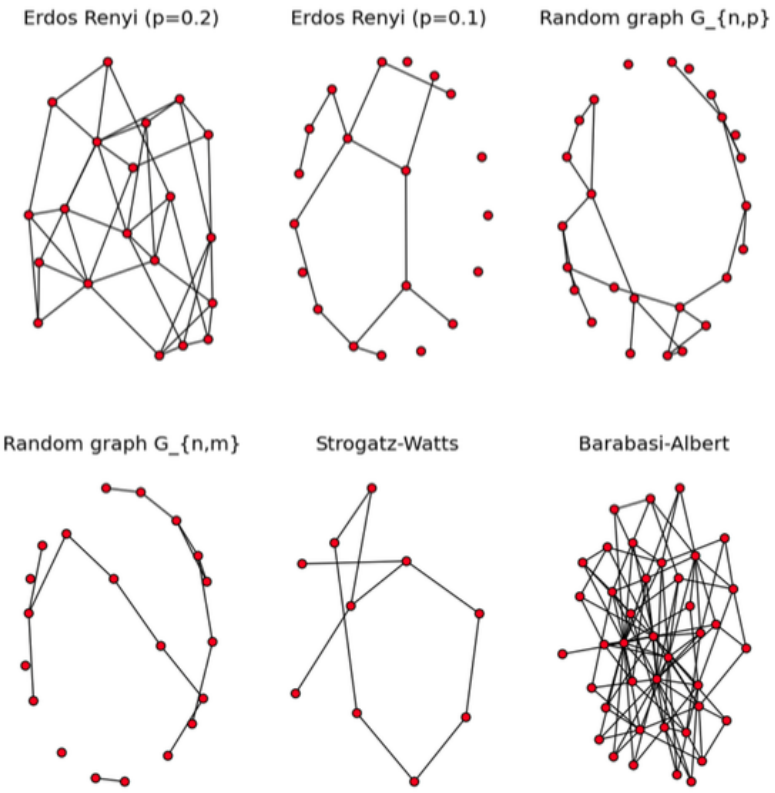
- **Assortativity**, or **assortative mixing**, is a preference for a network's nodes to attach to others that are similar in some way. In social network analysis, this term is usually referred to as **homophily** ("birds of a feather flock together").
- Assortativity coefficients:

```
nx.attribute_assortativity_coefficient(G, attribute)
nx.attribute_mixing_matrix(G, attribute)
nx.attribute_mixing_dict(G, attribute)

nx.numeric_assortativity_coefficient(G, attribute)
nx.numeric_mixing_matrix(G, attribute)
nx.numeric_mixing_dict(G, attribute)

nx.degree_assortativity_coefficient(G, x='out', y='in')
nx.degree_mixing_matrix(G, x='out', y='in')
nx.degree_mixing_dict(G, x='out', y='in')
```

Random Graphs

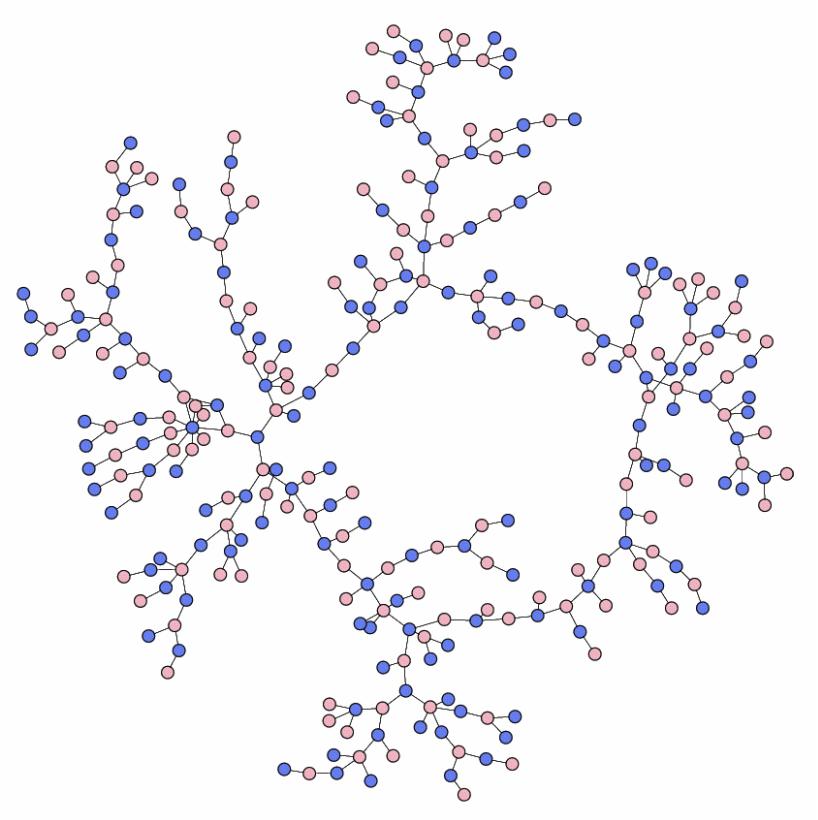
- Some Random Graphs in NetworkX:

```
nx.erdos_renyi_graph(n, p)
nx.gnp_random_graph(n, p)
nx.gnm_random_graph(n, m)
nx.watts_strogatz_graph(n, k, p)
nx.barabasi_albert_graph(n, m)
nx.random_regular_graph(d, n)
nx.random_tree(n)
```


Erdos Renyi (p=0.2)
Erdos Renyi (p=0.1)
Random graph G_{n,p}
Random graph G_{n,m}
Strogatz-Watts
Barabasi-Albert

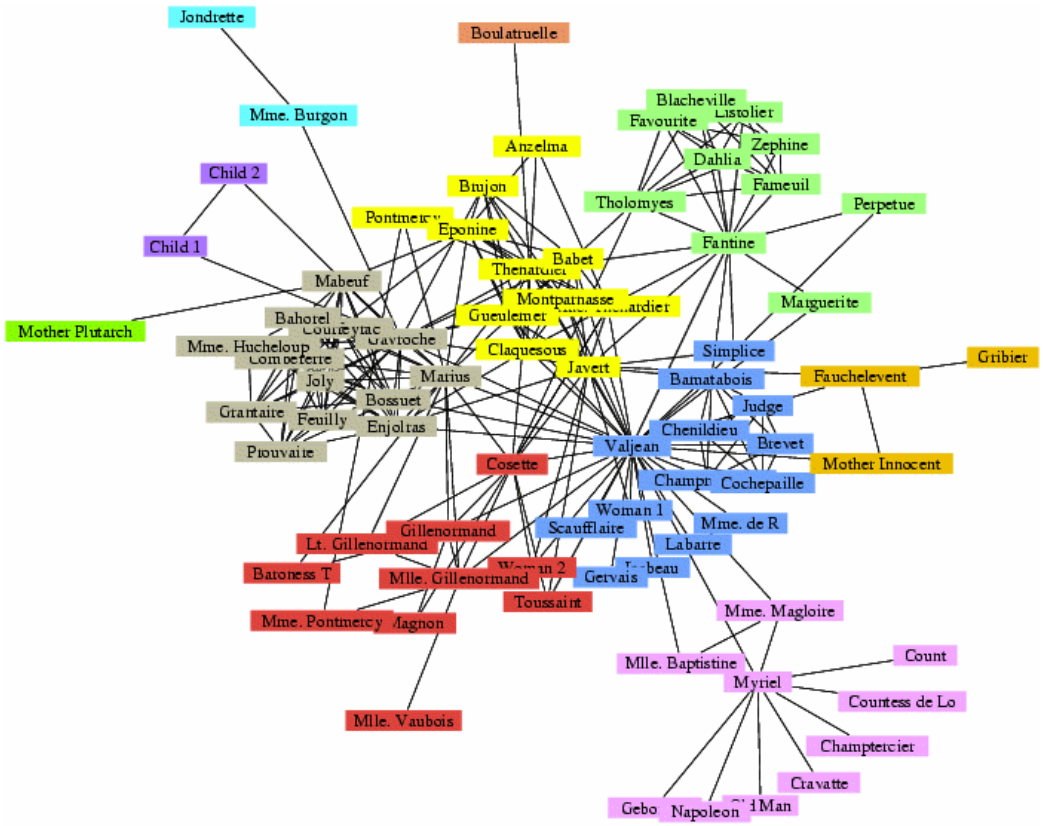
A Gallery of Empirical Networks

Adolescent romantic and sexual networks



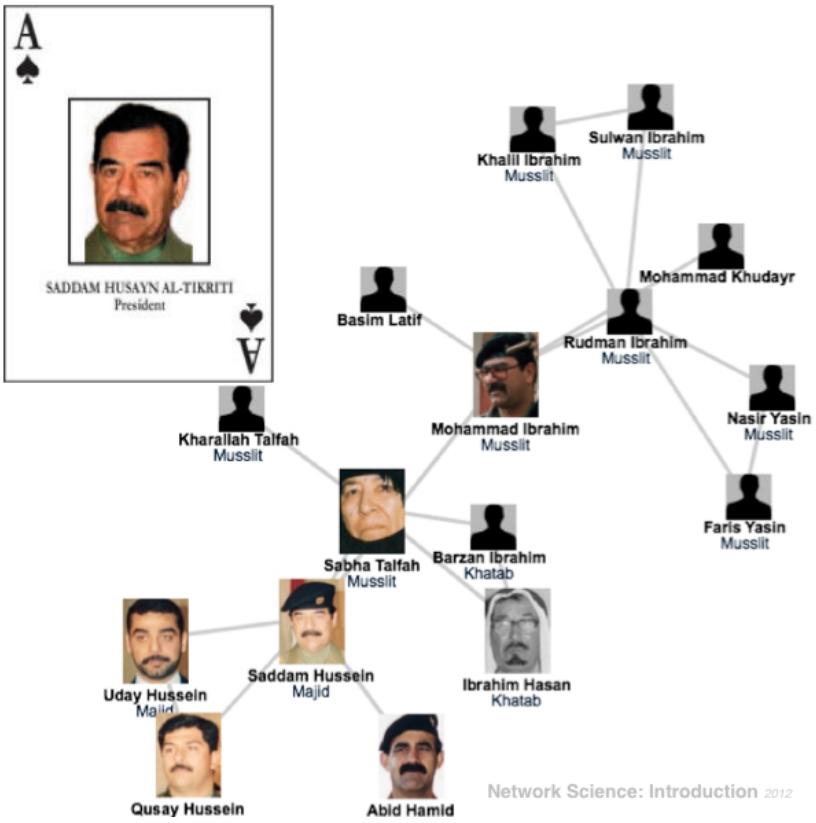
Bearman et al., Chains of affection: The structure of adolescent romantic and sexual networks, *American Journal of Sociology*, 2004, 110: 44-91

Victor Hugo's Les Misérables network



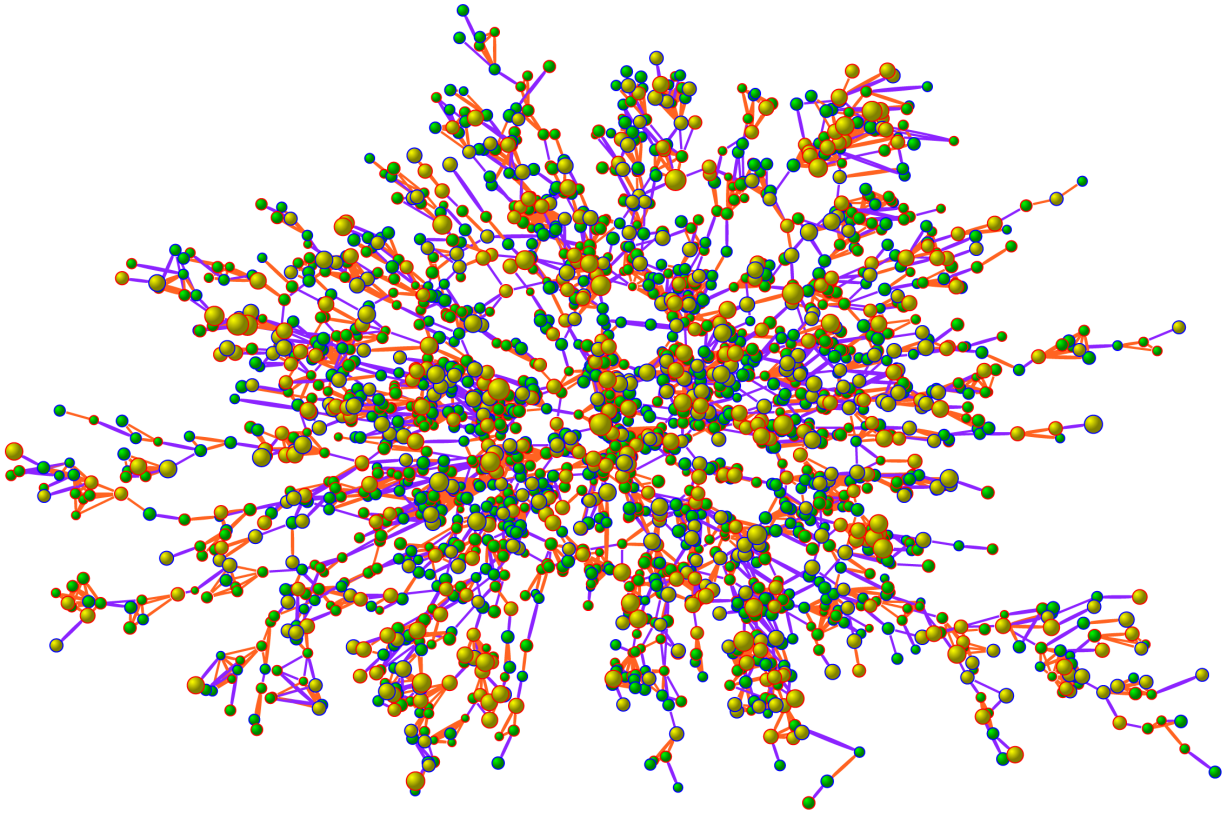
Newman & Girvan, Finding and evaluating community structure in networks, *Physical Review E*, 2004, 69: 026113

Saddam Hussein's networks



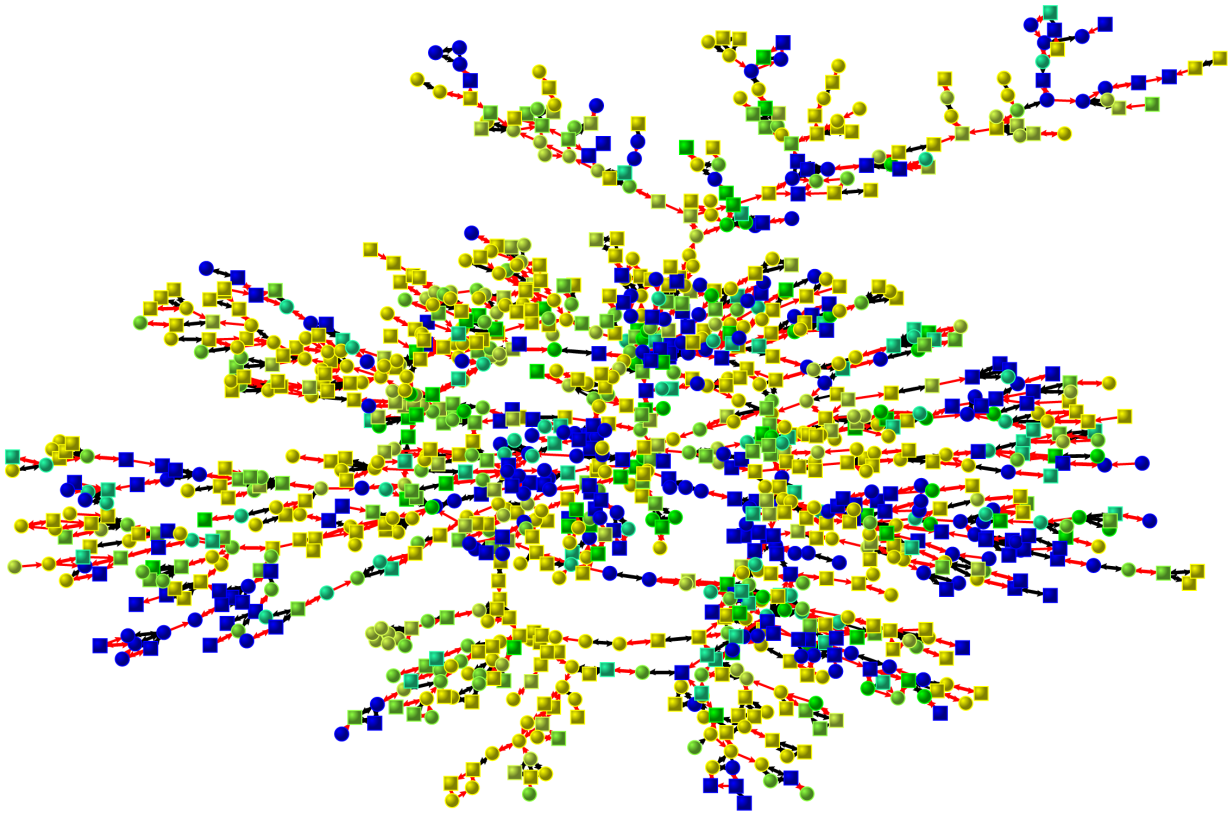
https://slate.com/news-and-politics/2010/02/searching-for-saddam-a-five-part-series-on-how-social-network

Obesity network



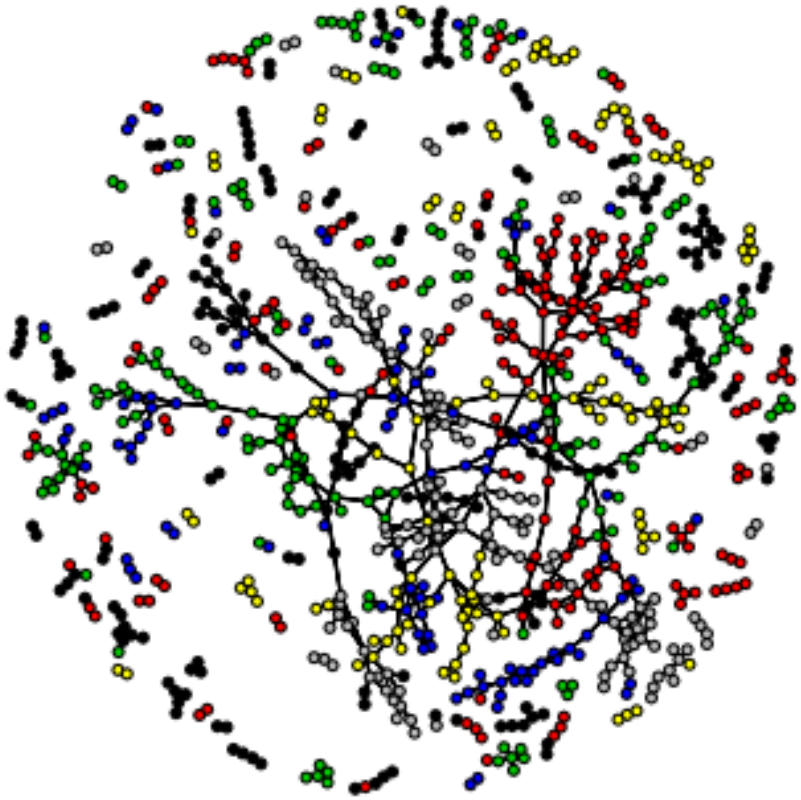
Christakis & Fowler, The spread of obesity in a large social network over 32 years, *New Englnd Journal of Medicine*, 2007, 357(4): 370-379

Happiness network



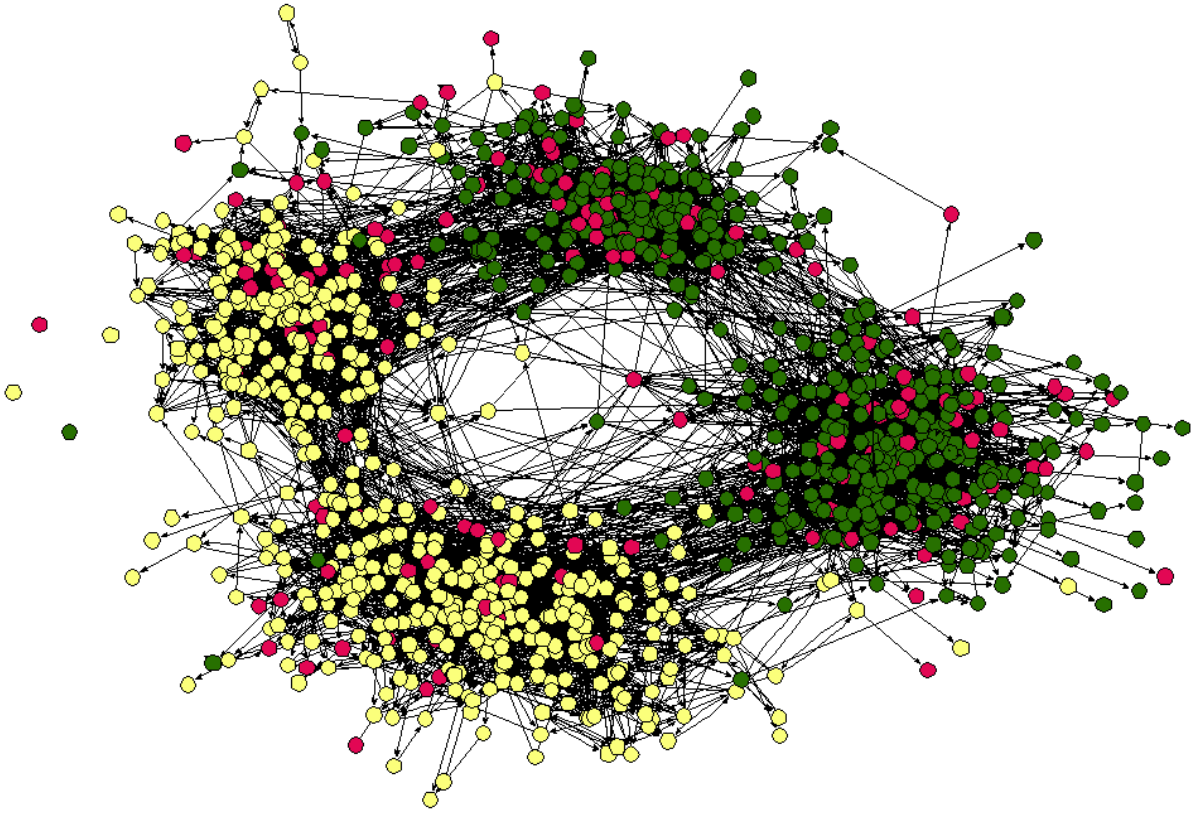
Fowler & Christakis, Dynamic spread of happiness in a large social network: Longitudinal analysis over 20 years in the Framingham Heart Study, *British Medical Journal*, 2008, 337(768): a2338

The Faux Magnolia High School network



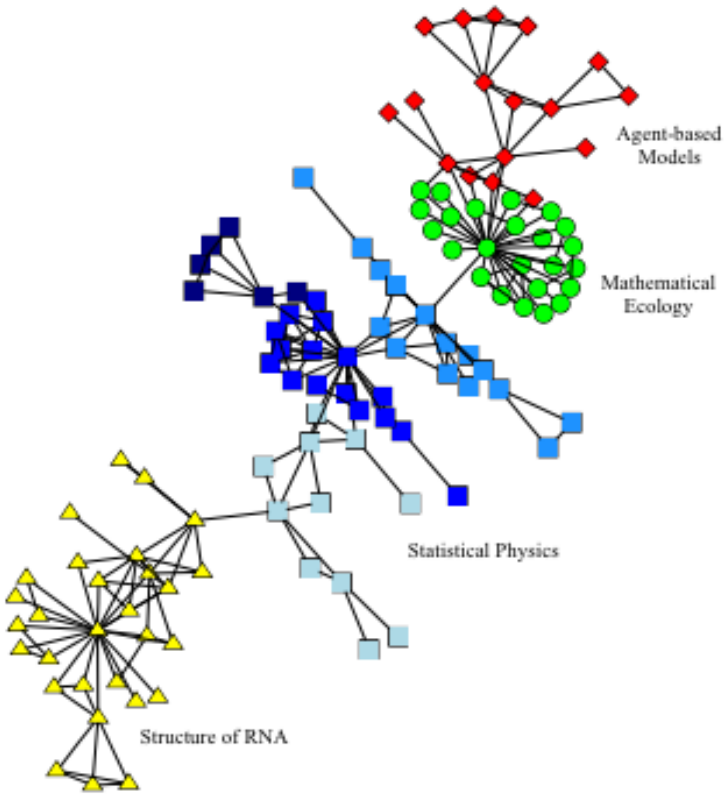
Goudreau et al., A `statnet` Tutorial, *Journal of Statistical Software*

Race and friendship in school network



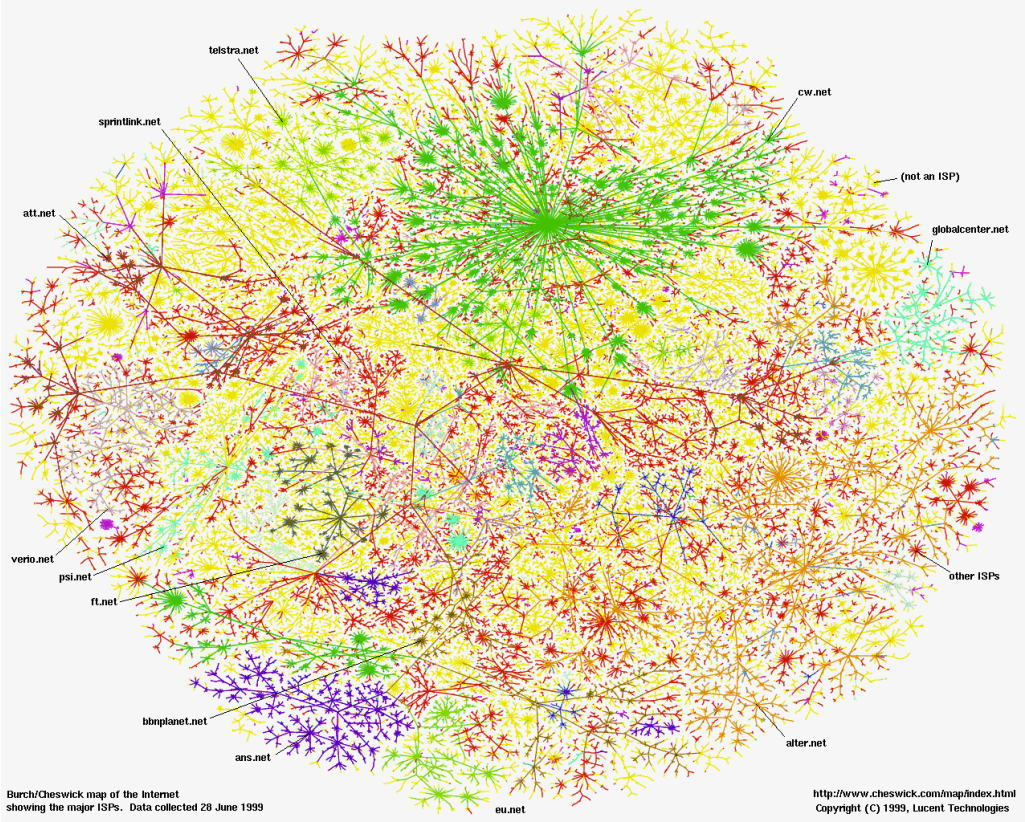
Moody, James. Race, school integration, and friendship segregation in America, *American Journal of Sociology*, 2001, 107: 679-716

Santa Fe collaboration network



Girvan & Newman. Community structure in social and biological networks,
Proceedings of the National Academy of Sciences of the USA, 2002, 99: 8271-8276

Internet ISPs network



Cheswick & Burch. Internet Atlas Gallery

Computational Social Network Analysis

Social Network Analysis: Patterns and Measures

Part 1/2

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

Linköping University, Norrköping, Sweden

Spring 2019

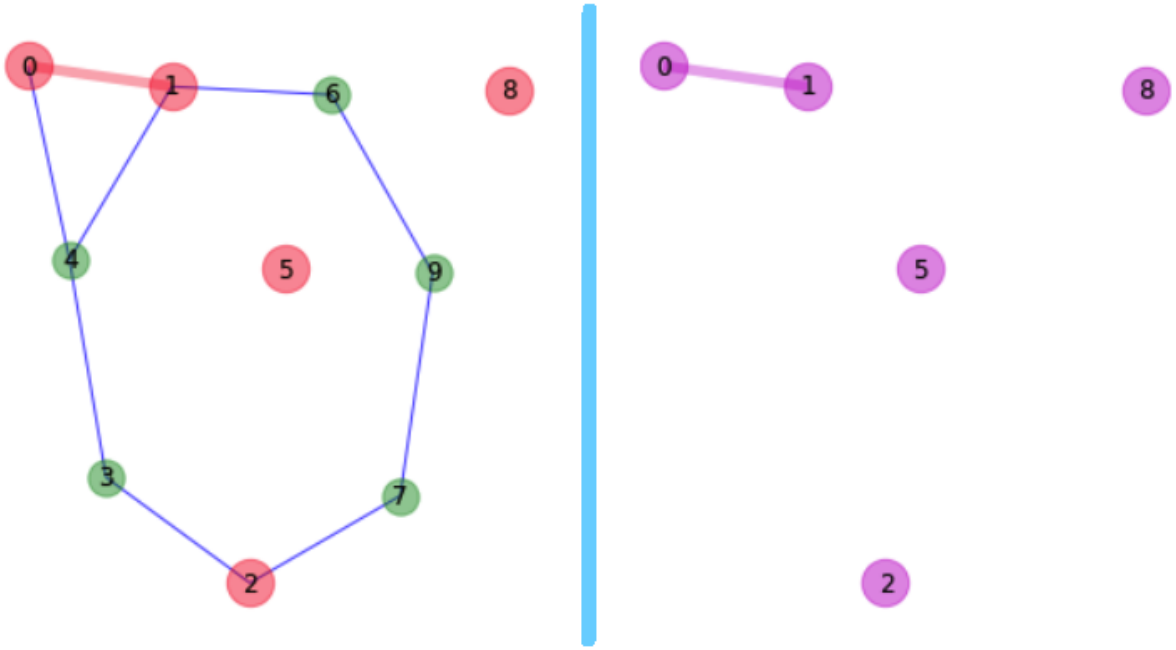
Subgraphs

- Let G be a graph with set of nodes V and set of edges E.
- If X is a subset of nodes of G (i.e., $X \subset V$), the **node–induced subgraph** of X is a graph $G(X)$ with set of nodes X and set of edges $E(X) = \{(u, v) \in E : u, v \in X\}$, i.e., the subgraph $G(X)$ includes only those edges of G having end points in the subset of nodes X.
- If Y is a subset of edges of G (i.e., $Y \subset E$), the **edge–induced subgraph** of Y is a graph $G(Y)$ with set of edges Y and set of nodes $V(X) = \{u \in V : \text{either } (u, v) \in Y \text{or } (v, u) \in Y\}$, i.e., the subgraph $G(Y)$ includes only those nodes of G being end points of the subset of edges Y.
- Here we will be only concerned with node-induced subgraphs that we will be calling (simply) **subgraphs**.
- NetworkX command:

  `G.subgraph(X)`

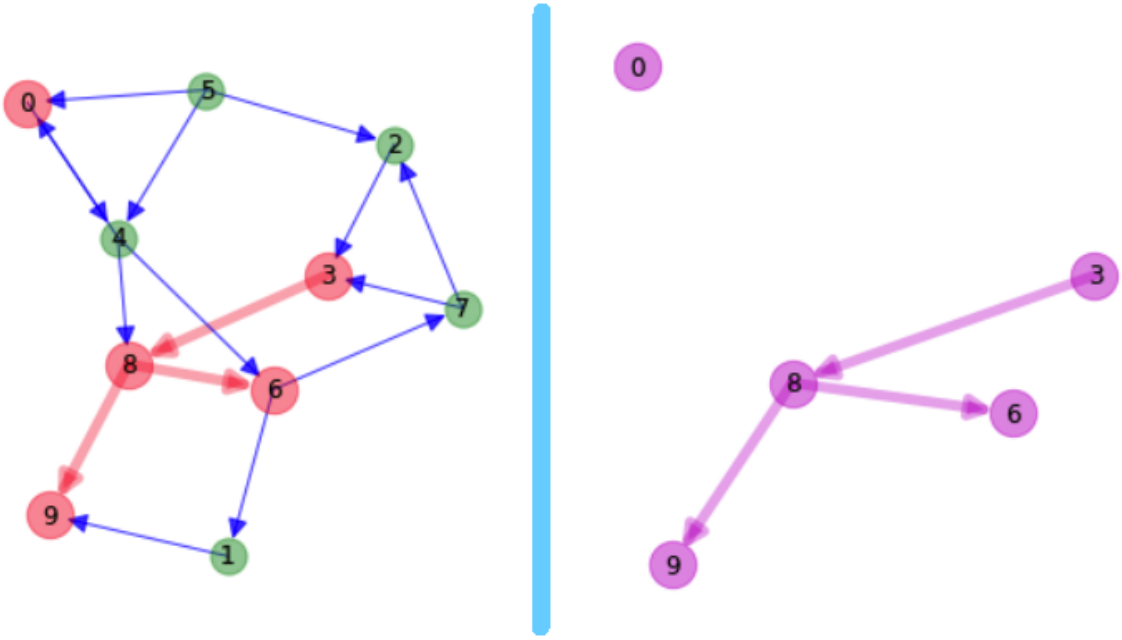
Subgraph of an undirected graph

```
Graph isolates: [5, 8]
Subgraph of [0, 1, 2, 5, 8]
True
Subgraph isolates: [2, 5, 8]
```



Subgraph of a directed graph

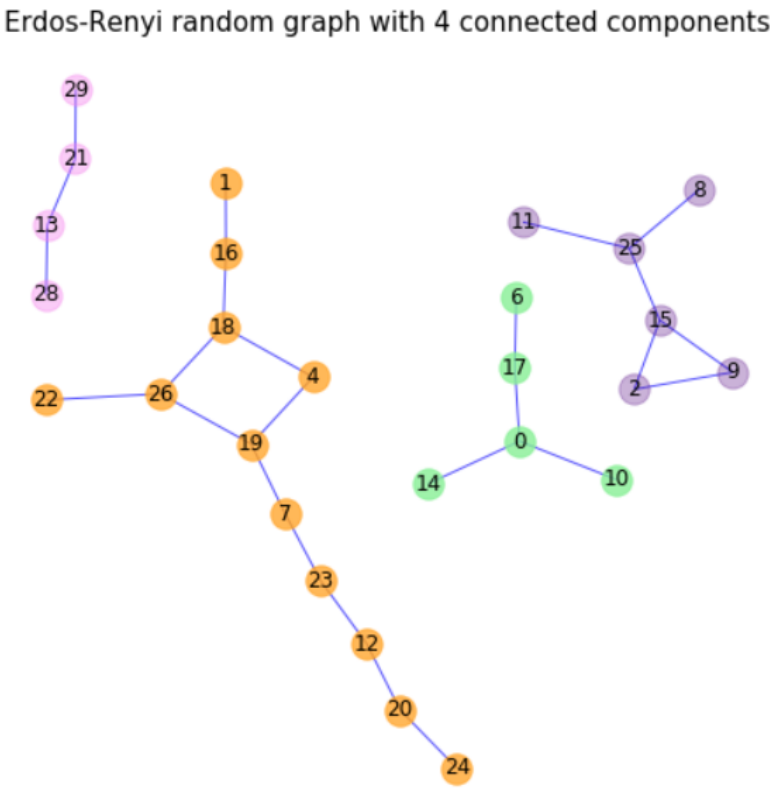
```
Graph isolates: []
Subgraph of [0, 3, 6, 8, 9]
True
Subgraph isolates: [0]
```



Connected components of undirected graphs

- In an undirected graph G, a **connected component** is a maximal connected component, i.e., a connected subgraph such that there is no other subgraph containing it which is connected.
- Clearly, an undirected graph is connected if and only if it contains a single (only one) connected component.
- Networkx commands:
  - The list of all connected components:
    `list(nx.connected_components(G))`
  - The number of all connected components:
    `nx.number_connected_components(G)`

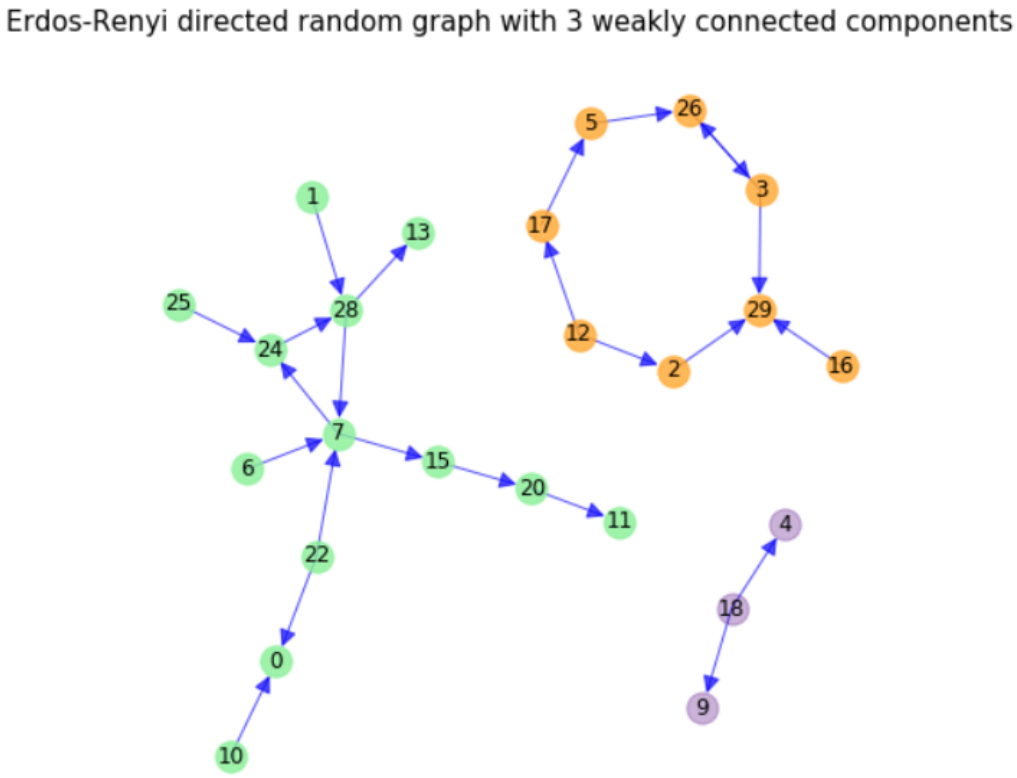
An example of multiple connected components in an undirected graph



Weakly connected components of directed graphs

- In a directed graph G, a **weakly connected component** is a subgraph such that, for every pair of nodes u, v in the subgraph, there is a path from to u to v (no matter what directions its edges have) and there is no other subraph containing it which this property.
- Clearly, a directed graph is weakly connected if and only if it contains a single (only one) weakly connected component.
- Networkx commands:
  - The list of all weakly connected components:
    `list(nx.weakly_connected_components(G))`
  - The number of all weakly connected components:
    `nx.number_weakly_connected_components(G)`

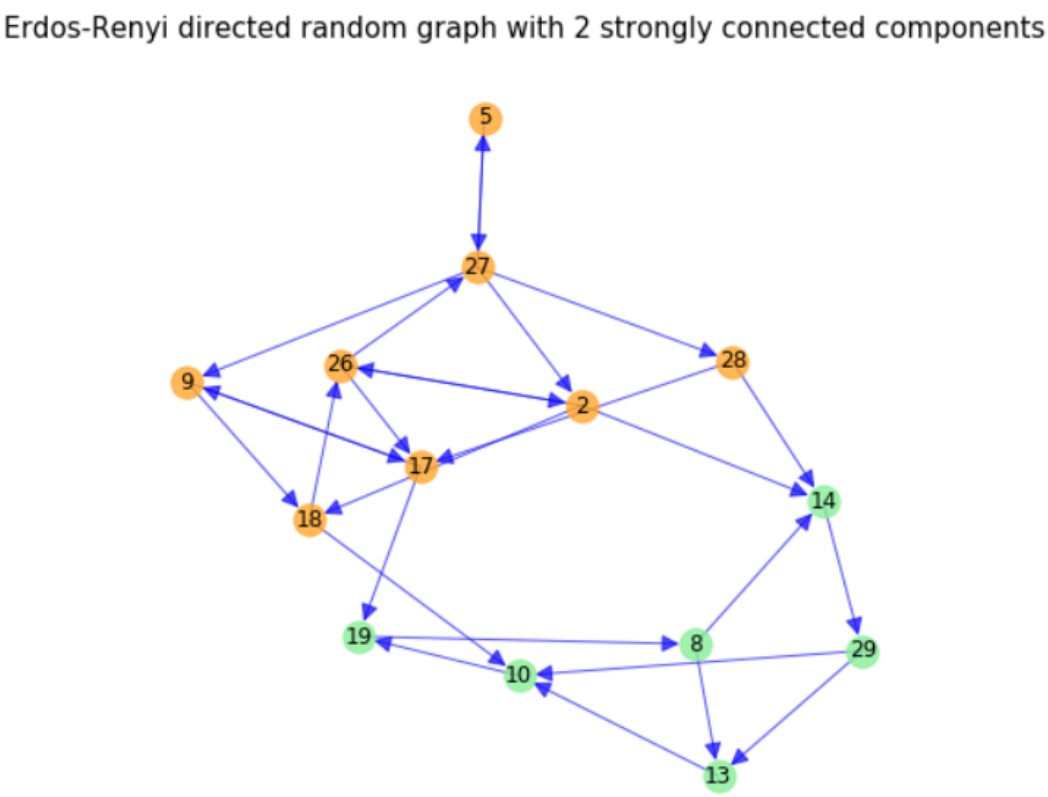
An example of multiple weakly connected components in a directed graph



Strongly connected components of directed graphs

- In a directed graph G, a **strongly connected component** is a subgraph such that, for every pair of nodes u, v in the subgraph, there is a directed path from to u to v and a directed path from to v to u; moreover, there is no other subraph containing it which this property.
- Clearly, a directed graph is strongly connected if and only if it contains a single (only one) strongly connected component.
- Networkx commands:
  - The list of all strongly connected components:
    `list(nx.strongly_connected_components(G))`
  - The number of all strongly connected components:
    `nx.number_strongly_connected_components(G)`

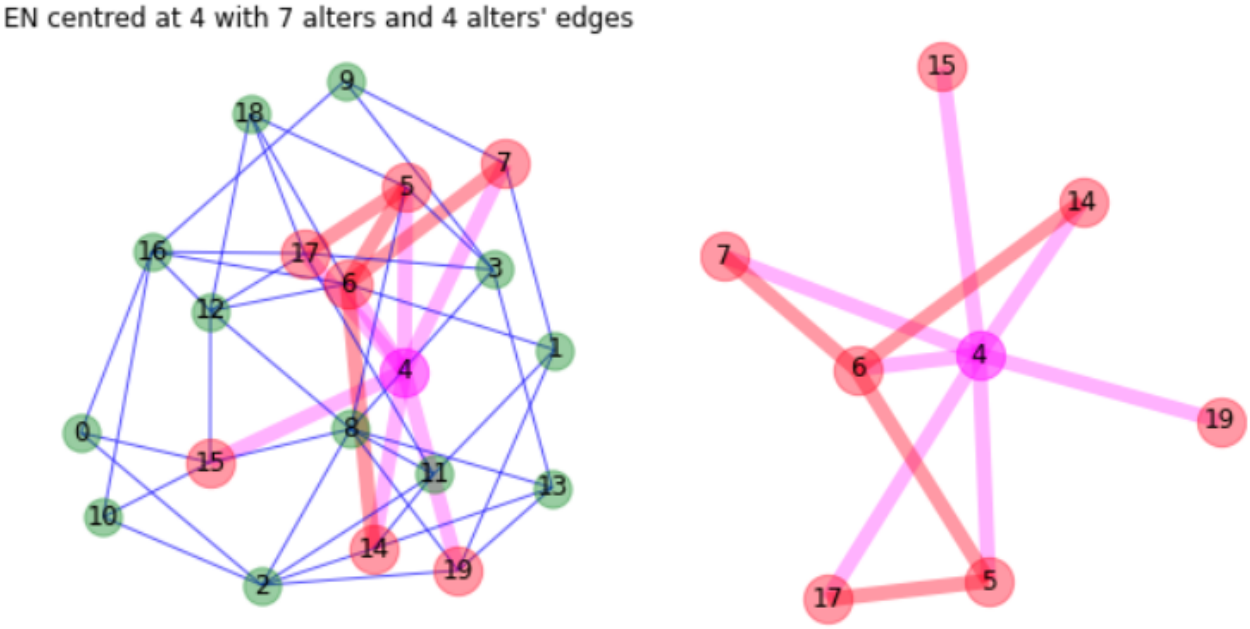
An example of multiple strongly connected components in a directed graph



Ego–centric graphs

- Let G be an undirected graph and u a node which is not isolated.
- The subgraph of the set of all neighbors of u together with u itself is called the **ego–net** of u (or with centre at u).
- In other words, the ego–net of u contains two types of links (edges):
  1. all links joining u to its neighbors and
  2. all links among u's neighbors
- Node u is called the **ego** of this ego–net and u's neighbors are called its **alters**.

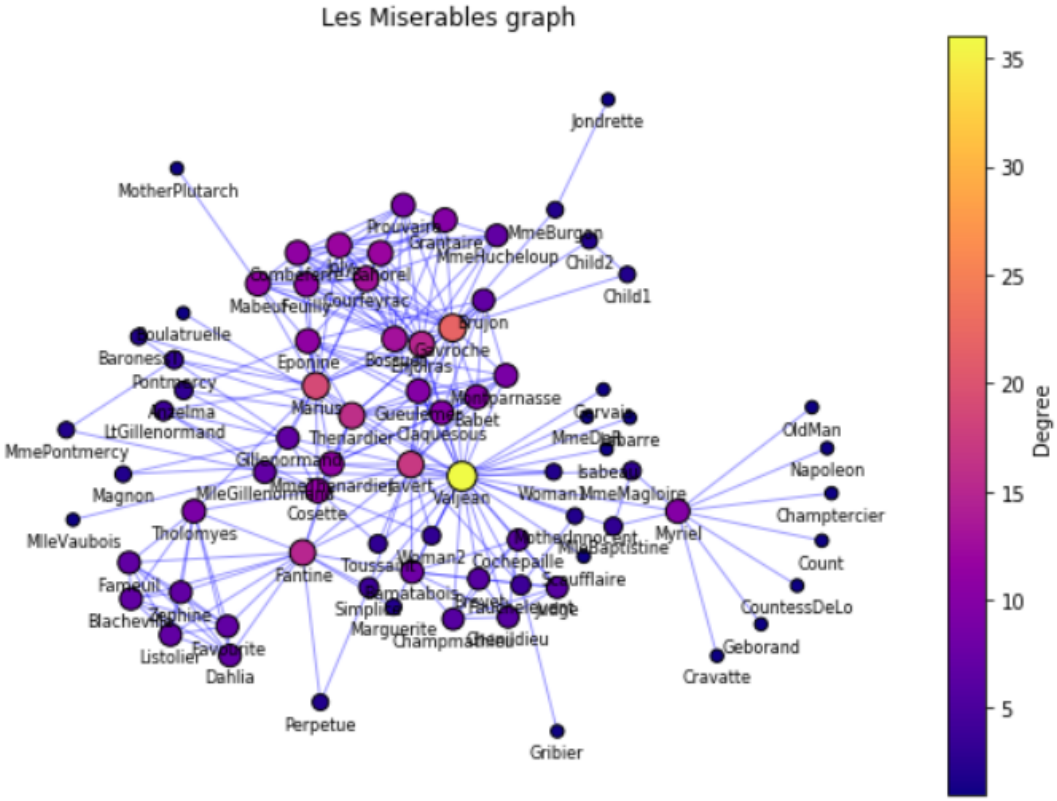
An example of an ego–centric graph (EN)



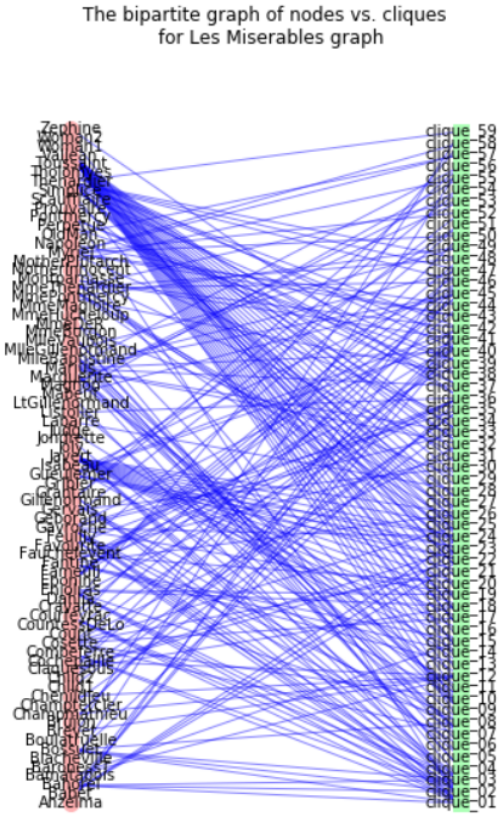
Cliques

- **Clique** is a subset of nodes of an *undirected graph* such that every pair of nodes in the clique are adjacent (i.e., the induced subgraph of a clique is complete).
- The **size** of a clique is the number of its nodes.
- **Maximal clique** is a clique such that there is no ther clique including it. *In our computations here, all cliques considered will be assumed to be maximal.*
- The problem of finding whether there is a clique of a given size in a graph (the clique problem) is NP-complete.
- **Maximum clique** of a graph is a clique with the maximum size. Moreover, the **clique number** $\omega(G)$ of a graph G is the size a maximum clique.
- **Intersection number** of a graph is the smallest number of cliques that together cover all edges of the graph.
- **Clique cover number** of a graph is the smallest number of cliques whose union covers the set of nodes of the graph.

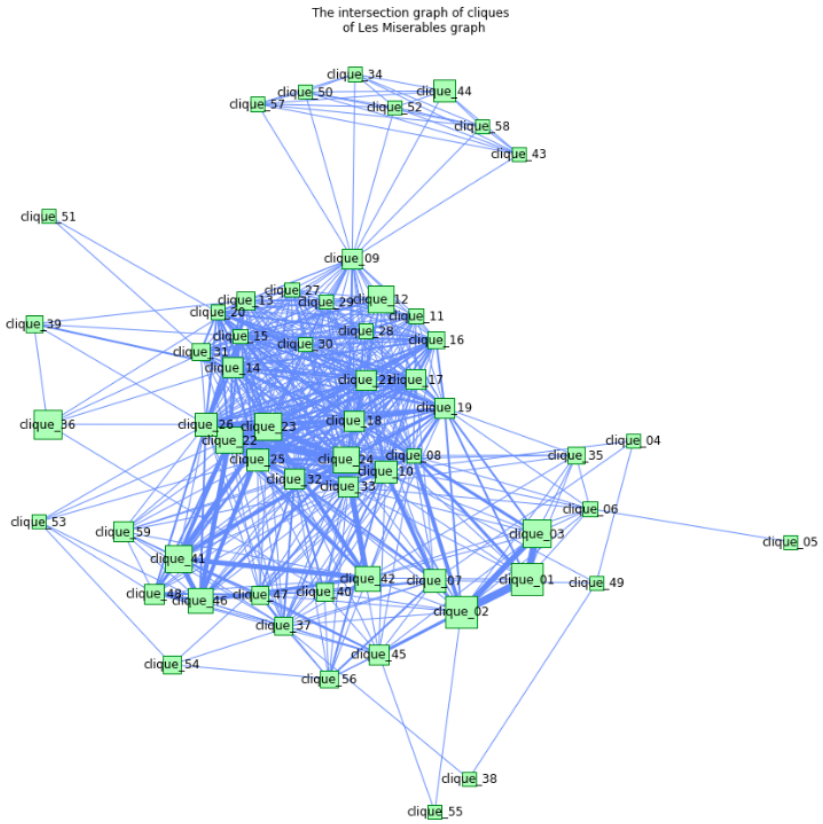
Victor Hugo's Les Misérables network



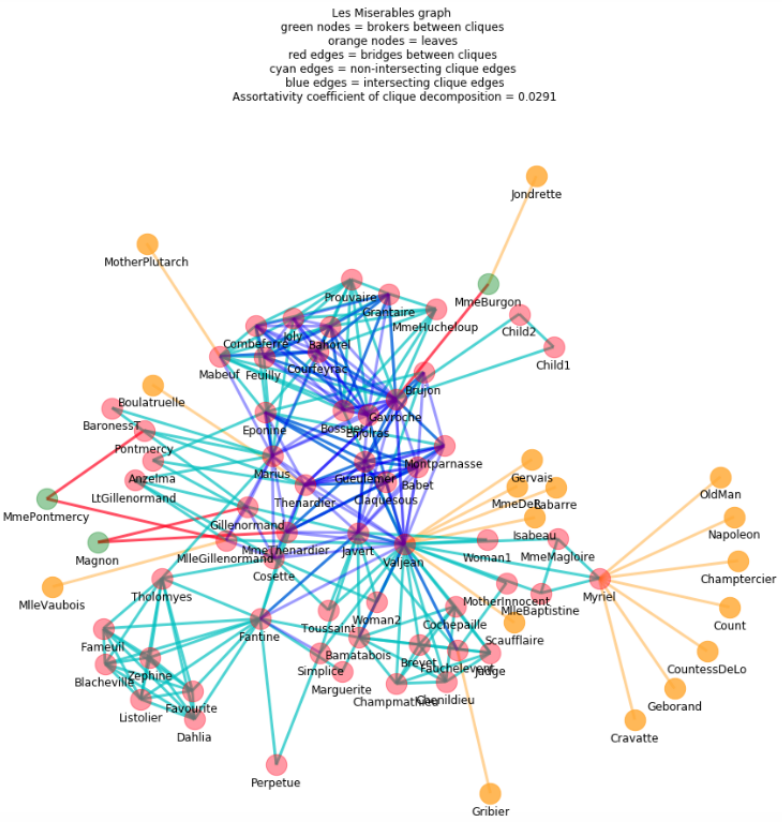
Victor Hugo's Les Misérables network: The bipartite graph of nodes vs. cliques



Victor Hugo's Les Misérables network: The intersection graph of cliques



Victor Hugo's Les Misérables network: Clique assortativity coefficient = 0.0291



Cycles

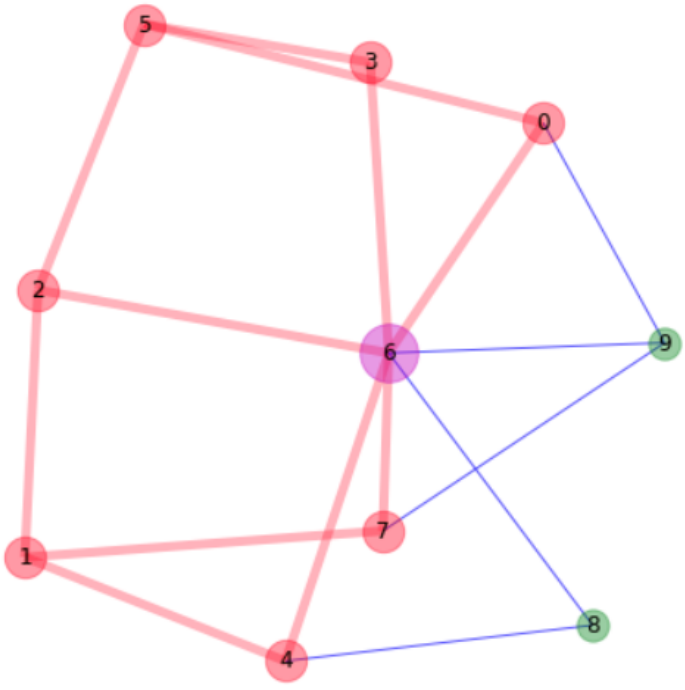
- k–**Cycle** in a graph G is a set of nodes $x_1, x_2, \ldots, x_k$ of G such that $(x_1, x_2), (x_2, x_3), \ldots, (x_{k-1}, x_k)$ are all edges of G. The integer k is called **length** of the k–cycle. If the length k is clear what it is, one simply says **cycles**. Apparently:
  - 3–cycles are *triangles*
  - 4–cycles are *quadrangles*
  - 5–cycles are *pentagons*
  - 6–cycles are *hexagons*, etc.
- NetworkX commands for finding all cycles in a graph G:
  - in an undirected graph:

    `list(nx.cycle_basis(G))`
  - and in a directed graph:

    `list(nx.simple_cycles(G))`

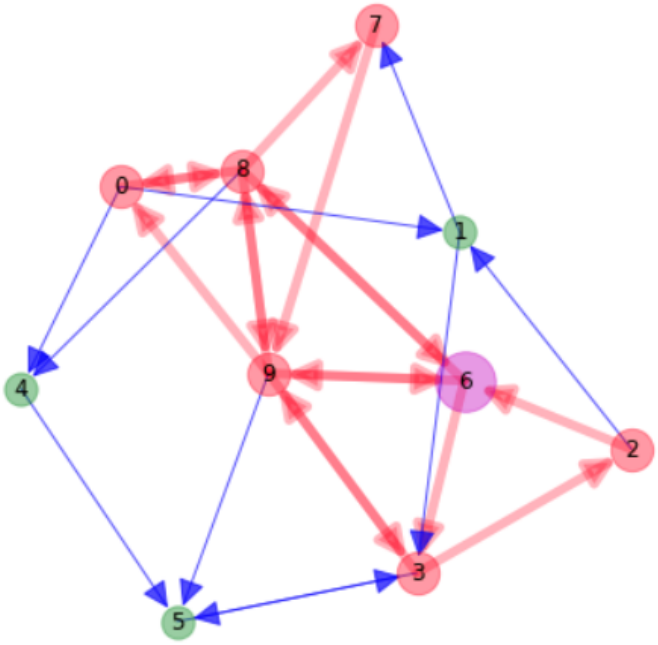
Cycles in an undirected graph

```
Node 6 is in the 4 4-cycles [[2, 1, 7, 6], [4, 1, 7, 6], [5, 3, 6, 0],
[5, 2, 6, 0]]
```



Cycles in a directed graph

```
Node 6 is in the 4 directed 4-cycles [[0, 8, 6, 9], [2, 6, 9, 3], [3,
9, 8, 6], [8, 7, 9, 6]]
```



Computational Social Network Analysis

Social Network Analysis: Patterns and Measures

Part 2/2

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

Linköping University, Norrköping, Sweden

Spring 2019

Node centrality indices

Centrality Index	*Undirected Graph*	*Directed Graph*
Degree	✓	
Out–degree		✓
In–degree		✓
Closeness	✓	✓
Betweenness	✓	✓
Eigenvector	✓	✓
Katz	✓	✓
PageRank	✓	✓
Load	✓	✓
HITS	✓	
HITS–hubs		✓
HITS–auths		✓
Communicability	✓	
Current flow	✓	

- **Degree centrality** of nodes of an undirected graph:

  `nx.degree_centrality(G)`

  The degree centrality for a node is the fraction of nodes it is connected to.
- **Out–degree centrality** of nodes of a directed graph:

  `nx.out_degree_centrality(G)`

  The out-degree centrality for a node is the fraction of nodes its outgoing edges are connected to.
- **In–degree centrality** of nodes of a directed graph:

  `nx.in_degree_centrality(G)`

  The in-degree centrality for a node is the fraction of nodes its incoming edges are connected to.

- **Closeness centrality** of nodes of undirected/directed graph:

  `nx.closeness_centrality(G)`

  Closeness centrality of a node is the reciprocal of the average shortest path distance to this node over all other reachable nodes.
- **Betweenness centrality** of nodes of undirected/directed graph:

  `nx.betweenness_centrality(G)`

  Betweenness centrality of a node is the sum of the fraction of all–pairs shortest paths that pass through this node.
- **Eigenvector centrality** of nodes of undirected/directed graph:

  `nx.eigenvector_centrality(G,max_iter=maxiter)`

  Eigenvector centrality computes iteratively the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is the i-th element of the vector x solving the equation $Ax = \lambda x$, where A is the adjacency matrix of the graph with largest eigenvalue λ.

- **Katz centrality** of nodes of undirected/directed graph:

  `nx.katz_centrality_numpy(G, 0.05)`

  Katz centrality computes the centrality for a node based on the centrality of its neighbors. It is a generalization of the eigenvector centrality.
- **PageRank** of nodes of undirected/directed graph:

  `nx.pagerank_centrality(G)`

  PageRank computes a ranking of the nodes in the graph based on the structure of the incoming links. It was originally designed as an algorithm to rank web pages.
- **Load centrality** of nodes of undirected/directed graph:

  `nx.load_centrality(G)`

  The load centrality of a node is the fraction of all shortest paths that pass through that node.

- **HITS centrality** of nodes of an undirected graph:

  `nx.hits(G, max_iter=maxiter)`

  Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm that rates Web pages, developed by Jon Kleinberg. The idea stemmed from a particular insight into the creation of web pages when the Internet was originally forming, where a good hub represented a page that pointed to many other pages and a good authority represented a page that was linked by many different hubs.
- **HITS–hubs centrality** of nodes of a directed graph:

  `nx.hits(G, max_iter=maxiter)[0]`

  According to the HITS algorithm, HITS–hubs estimates an index for a node based on ougoing links.
- **HITS–auths centrality** of nodes of a directed graph:

  `nx.hits(G, max_iter=maxiter)[1]`

  According to the HITS algorithm, HITS–auths estimates an index for a node based on incoming links.

- **Communicability betweenness centrality** of nodes of an undirected graph:

  `nx.communicability_betweenness_centrality(G)`

  Communicability betweenness centrality uses the number of walks connecting every pair of nodes as the basis of a betweenness centrality measure.
- **Current–flow closeness centrality** of nodes of an undirected graph:

  `nx.current_flow_closeness_centrality(G)`

  Current–flow closeness centrality is variant of closeness centrality based on effective resistance between nodes in a network. This metric is also known as information centrality.

Example of centralities of an undirected graph

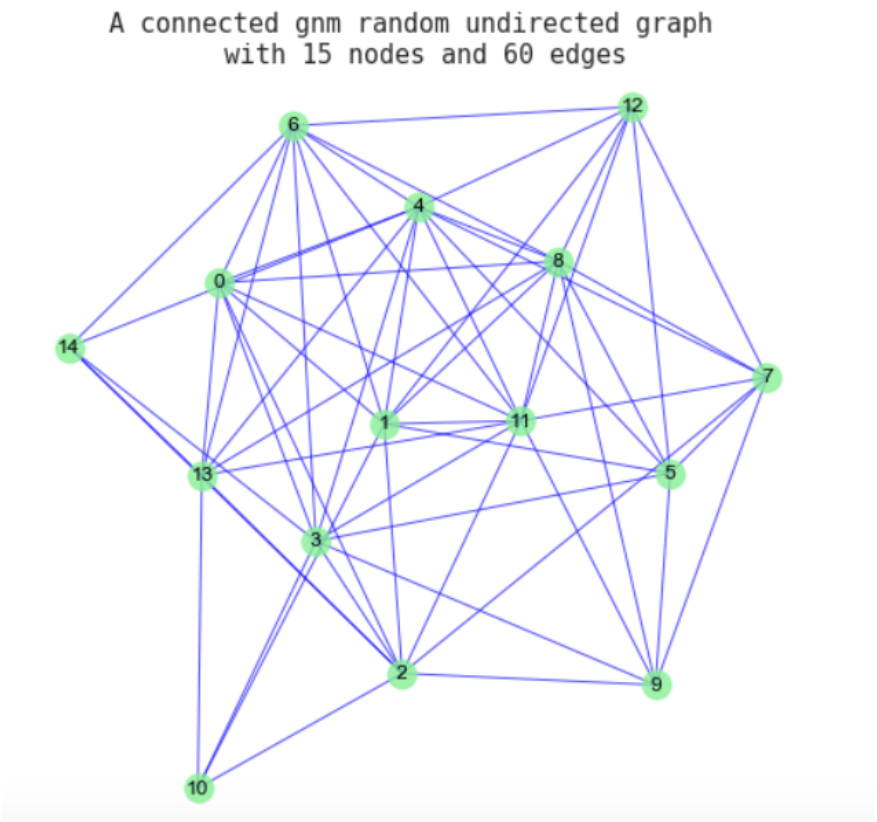


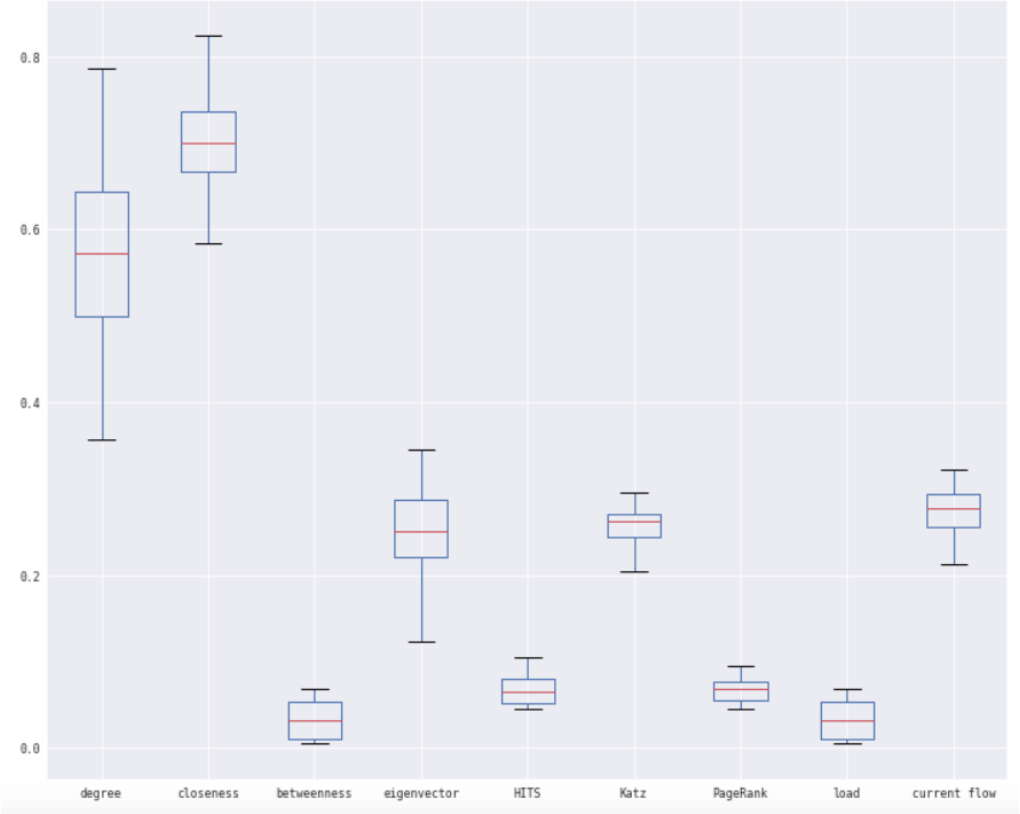
Table of centrality indices

	degree	closeness	betweenness	eigenvector	HITS	Katz	PageRank	load	communicability	current flow
0	0.571429	0.700000	0.011774	0.275861	0.086141	0.261842	0.075298	0.011774	0.433975	0.276866
1	0.642857	0.736842	0.055154	0.283442	0.056056	0.270054	0.061573	0.055154	0.468077	0.293727
2	0.642857	0.736842	0.068080	0.250230	0.078664	0.264296	0.073341	0.068080	0.404873	0.293207
3	0.642857	0.736842	0.064129	0.260082	0.066822	0.266128	0.076649	0.064129	0.426276	0.294129
4	0.785714	0.823529	0.066719	0.338860	0.084223	0.294715	0.085454	0.066719	0.607221	0.320795
5	0.500000	0.666667	0.017870	0.218788	0.045476	0.243579	0.053492	0.017870	0.298508	0.255972
6	0.642857	0.736842	0.031450	0.292004	0.065484	0.271462	0.060511	0.031450	0.479166	0.292240
7	0.500000	0.666667	0.016327	0.224253	0.046829	0.244741	0.050207	0.016327	0.310220	0.255805
8	0.714286	0.777778	0.046363	0.314136	0.063647	0.282624	0.068632	0.046363	0.536660	0.306364
9	0.428571	0.636364	0.010178	0.190701	0.070219	0.231622	0.068059	0.010178	0.235853	0.235903
10	0.285714	0.583333	0.004710	0.123301	0.051683	0.204588	0.052578	0.004710	0.106702	0.183916
11	0.785714	0.823529	0.053166	0.344968	0.105603	0.296284	0.094994	0.053166	0.623042	0.321229
12	0.500000	0.666667	0.008346	0.238479	0.051487	0.246912	0.055817	0.008346	0.332679	0.255682
13	0.571429	0.700000	0.035269	0.248805	0.082190	0.256499	0.077900	0.035269	0.383238	0.275708
14	0.357143	0.608696	0.004971	0.164389	0.045478	0.219394	0.045494	0.004971	0.177862	0.212117

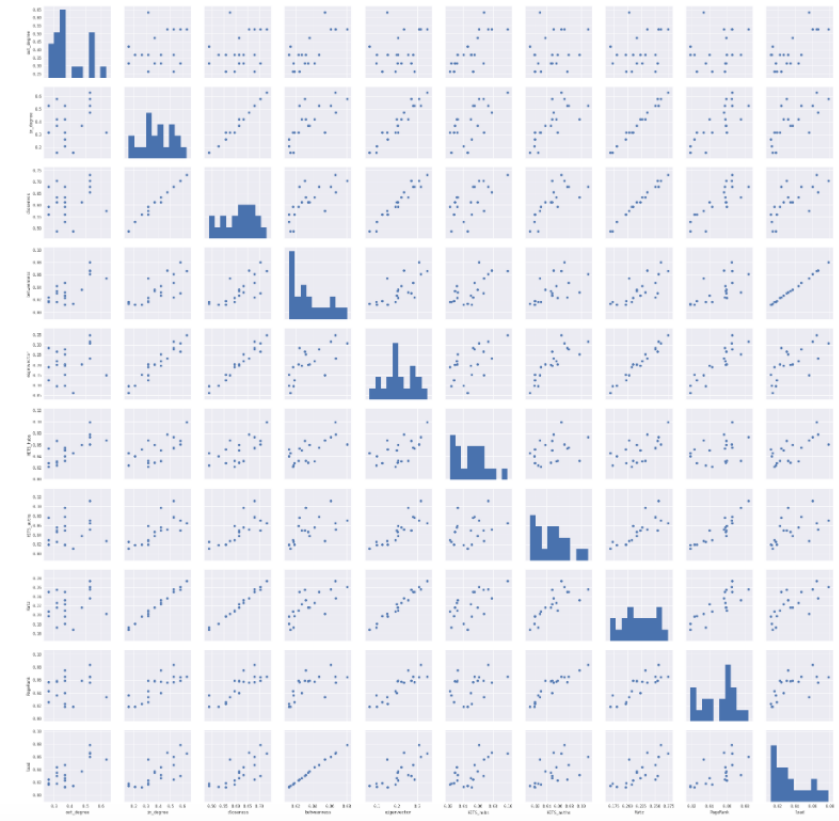
Descriptive statistics of centrality indices

	degree	closeness	betweenness	eigenvector	HITS	Katz	PageRank	load	communicability	current flow
count	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000	15.000000
mean	0.571429	0.706707	0.032967	0.251220	0.066667	0.256983	0.066667	0.032967	0.388290	0.271577
std	0.145386	0.071040	0.024042	0.061719	0.017870	0.025911	0.014089	0.024042	0.148219	0.038982
min	0.285714	0.583333	0.004710	0.123301	0.045476	0.204588	0.045494	0.004710	0.106702	0.183916
25%	0.500000	0.666667	0.010976	0.221521	0.051585	0.244160	0.054655	0.010976	0.304364	0.255743
50%	0.571429	0.700000	0.031450	0.250230	0.065484	0.261842	0.068059	0.031450	0.404873	0.276866
75%	0.642857	0.736842	0.054160	0.287723	0.080427	0.270758	0.075973	0.054160	0.473622	0.293928
max	0.785714	0.823529	0.068080	0.344968	0.105603	0.296284	0.094994	0.068080	0.623042	0.321229

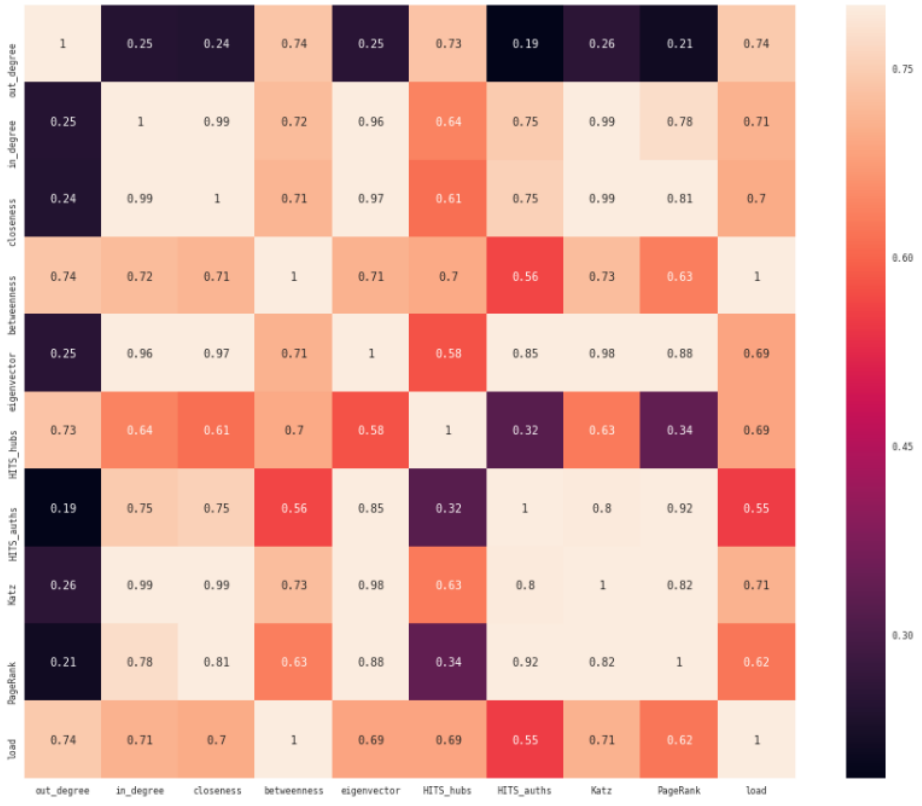
Boxplots of statistics of centrality indices



Pairplot of statistics of centrality indices



Correlation matrix of centrality indices



Example of centralities of a directed graph

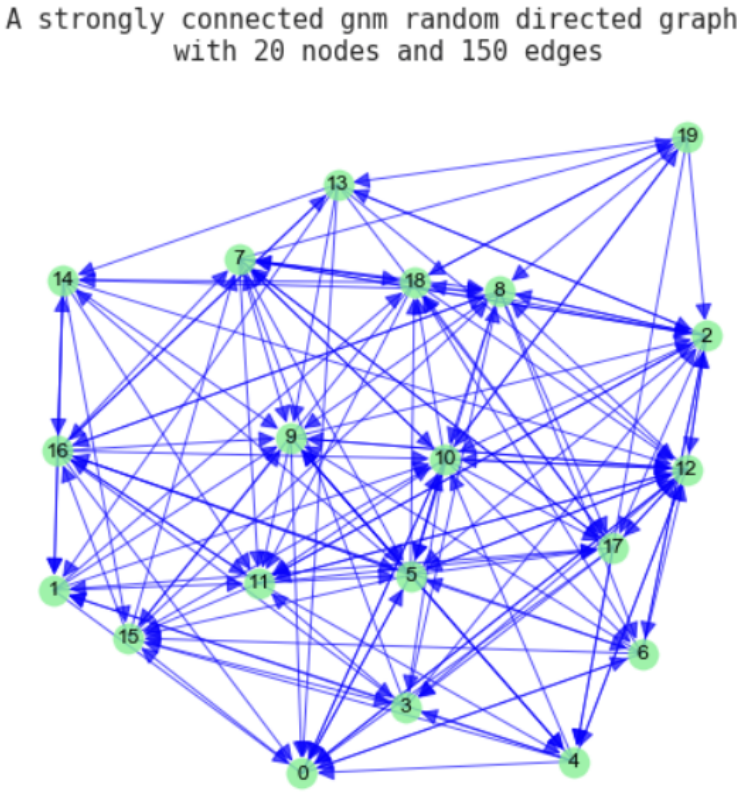


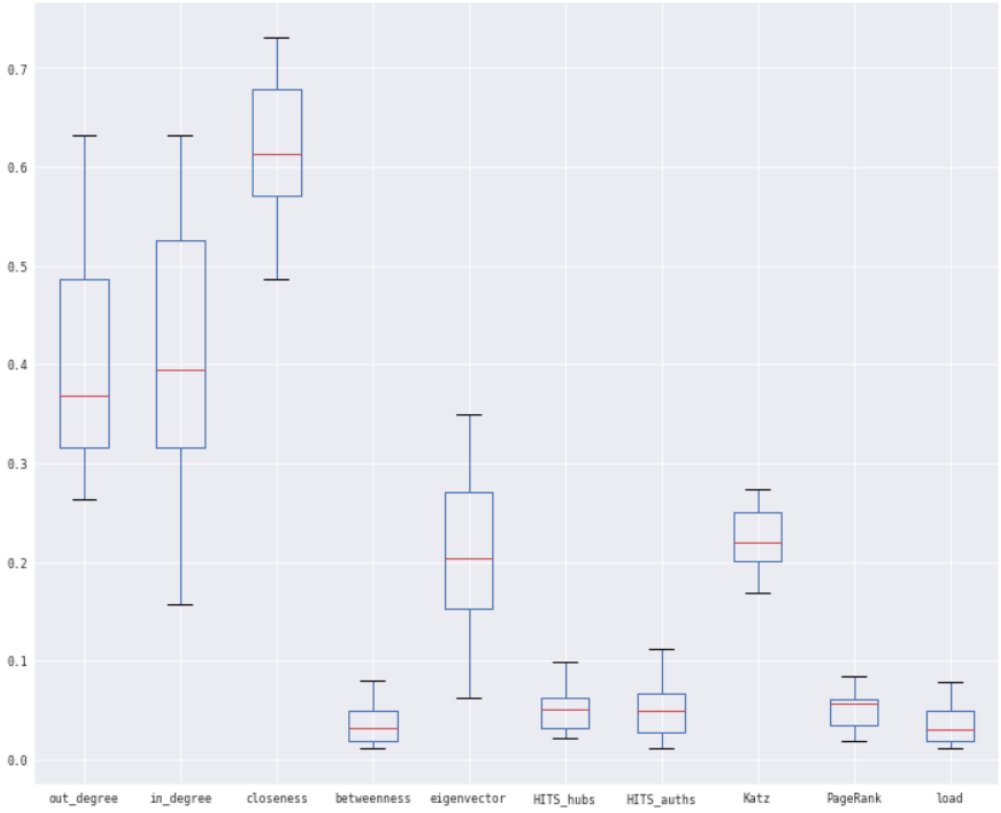
Table of centrality indices

	out_degree	in_degree	closeness	betweenness	eigenvector	HITS_hubs	HITS_auths	Katz
0	0.263158	0.526316	0.678571	0.024299	0.285211	0.053191	0.075977	0.250417
1	0.263158	0.315789	0.558824	0.018591	0.124907	0.027867	0.019535	0.197833
2	0.526316	0.473684	0.655172	0.066846	0.234451	0.077989	0.051721	0.236028
3	0.263158	0.315789	0.593750	0.016603	0.191085	0.021956	0.029508	0.207553
4	0.368421	0.315789	0.593750	0.023044	0.203482	0.031836	0.058938	0.209819
5	0.526316	0.526316	0.678571	0.061056	0.316380	0.073213	0.111718	0.256062
6	0.368421	0.263158	0.558824	0.012235	0.153165	0.051525	0.019669	0.196428
7	0.473684	0.368421	0.612903	0.036235	0.203154	0.059530	0.037486	0.216858
8	0.368421	0.421053	0.612903	0.026984	0.198393	0.050145	0.049564	0.223123
9	0.315789	0.578947	0.703704	0.030843	0.268462	0.066982	0.049759	0.254876
10	0.526316	0.578947	0.703704	0.080376	0.307722	0.060926	0.070818	0.260585
11	0.368421	0.526316	0.678571	0.047213	0.279553	0.055164	0.078232	0.250636
12	0.526316	0.631579	0.730769	0.066356	0.349256	0.099373	0.065012	0.273892
13	0.421053	0.157895	0.487179	0.013581	0.062524	0.045741	0.010792	0.168370
14	0.368421	0.210526	0.527778	0.012221	0.097575	0.040305	0.018423	0.180453
15	0.315789	0.368421	0.612903	0.034216	0.202493	0.028888	0.046809	0.216295
16	0.631579	0.315789	0.575758	0.054482	0.150628	0.067553	0.027821	0.202540
17	0.368421	0.421053	0.633333	0.031965	0.254072	0.032417	0.096951	0.232071
18	0.315789	0.421053	0.633333	0.042139	0.219903	0.031311	0.056253	0.226235
19	0.315789	0.157895	0.487179	0.017088	0.095145	0.024088	0.025014	0.172977

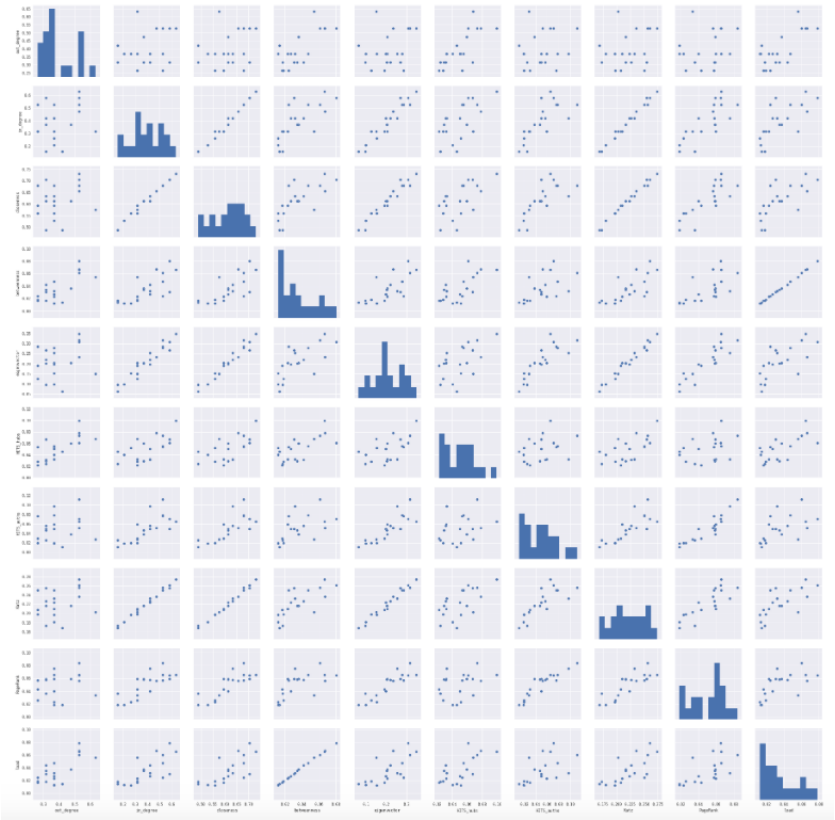
Descriptive statistics of centrality indices

	out_degree	in_degree	closeness	betweenness	eigenvector	HITS_hubs	HITS_auths	Katz	PageRank	load
count	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000
mean	0.394737	0.394737	0.615874	0.035819	0.209878	0.050000	0.050000	0.221652	0.050000	0.035819
std	0.105953	0.140293	0.069806	0.020617	0.079148	0.020647	0.027484	0.030265	0.018924	0.020201
min	0.263158	0.157895	0.487179	0.012221	0.062524	0.021956	0.010792	0.168370	0.018695	0.012138
25%	0.315789	0.315789	0.571524	0.018215	0.152531	0.031705	0.027119	0.201363	0.035395	0.018690
50%	0.368421	0.394737	0.612903	0.031404	0.203318	0.050835	0.049662	0.219990	0.057397	0.030981
75%	0.486842	0.526316	0.678571	0.049031	0.271235	0.062440	0.066463	0.250471	0.060642	0.049353
max	0.631579	0.631579	0.730769	0.080376	0.349256	0.099373	0.111718	0.273892	0.083634	0.078390

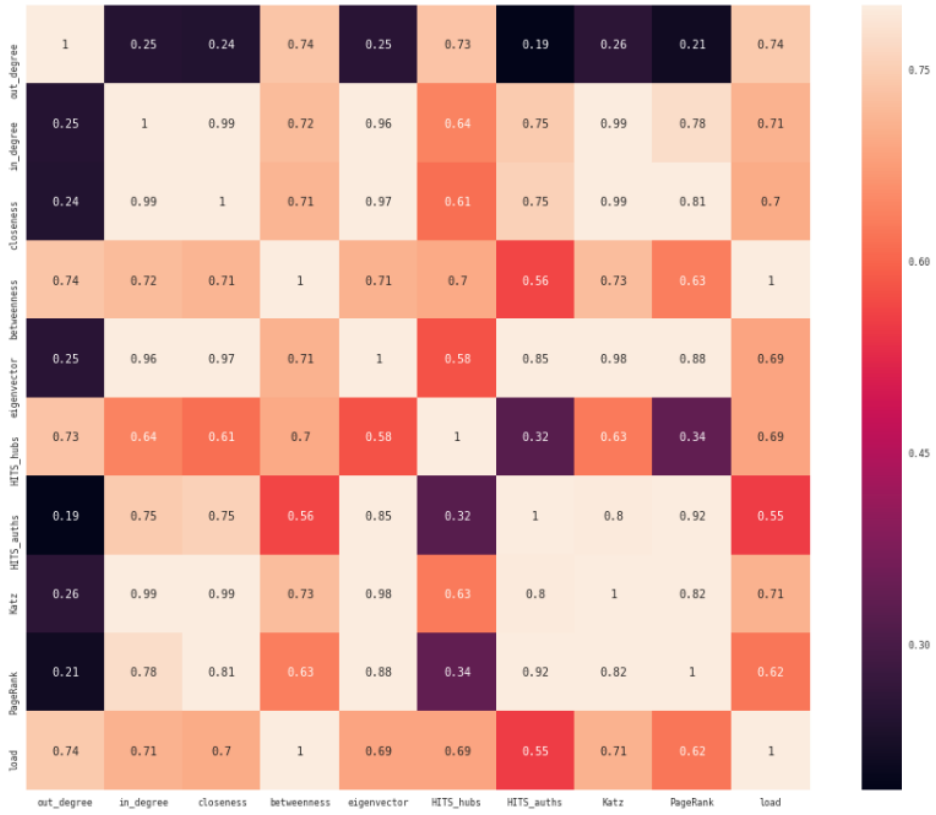
Boxplots of statistics of centrality indices



Pairplot of statistics of centrality indices



Correlation matrix of centrality indices



Computational Social Network Analysis

Social Network Analysis: Algorithms and Models

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

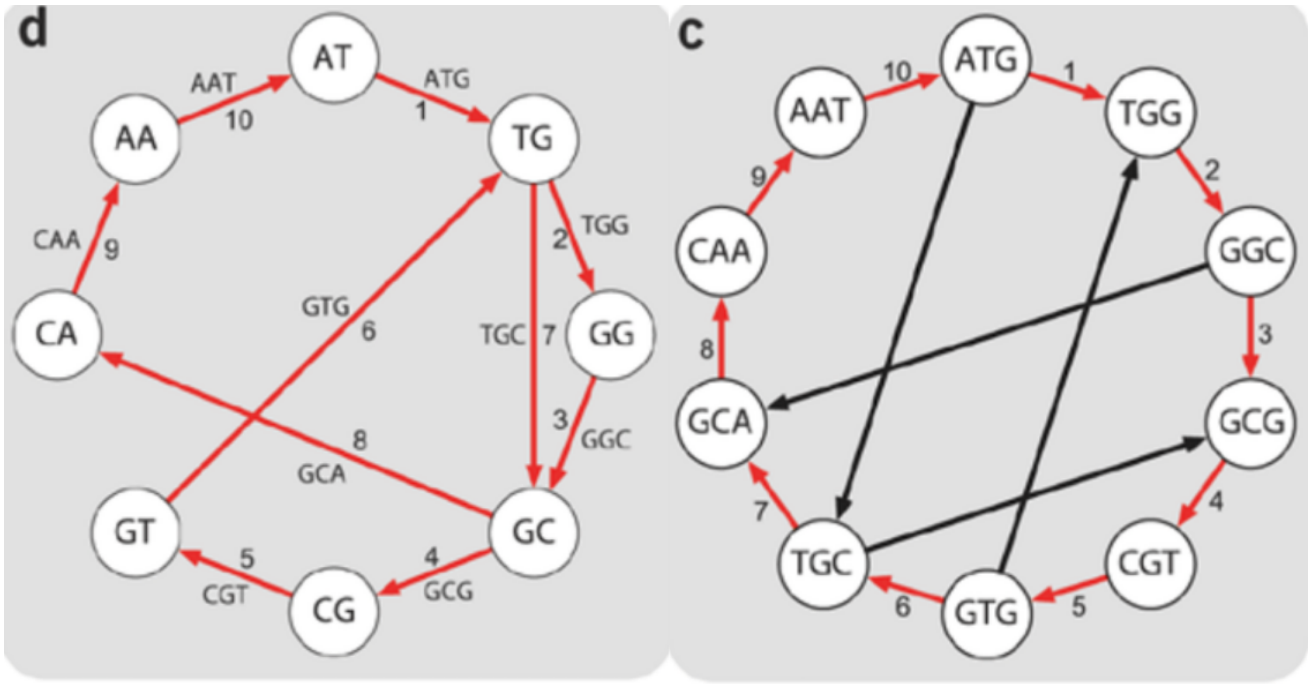
[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

Linköping University, Norrköping, Sweden

Spring 2019

Graph traversals



Eulerian cycle
Visit each edge once

Hamiltonian cycle
Visit each vertex once

Graph community analysis (Newman & Girvan, 2004)

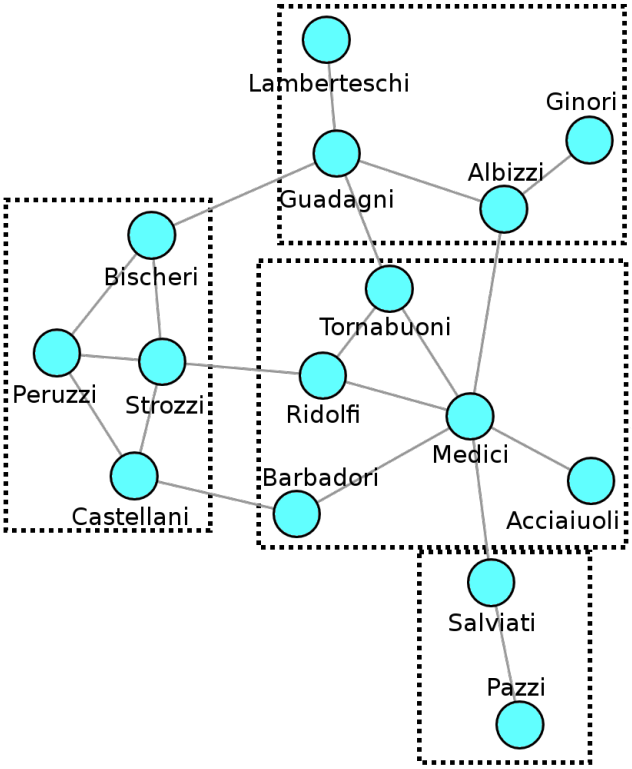
- Let $G = (V, E)$ be a graph. A **clustering** of vertices of G is a partition of the set of verticies V into a (finite) family $\mathcal{C} \subset 2^V$ of subsets of vertices, often called **modules**, such that $\bigcup_{C \in \mathcal{C}} C = V$ and $C \cap C' = \varnothing$, for each $C, C' \in \mathcal{C}, C \neq C'$.
- A clustering $\mathcal{C}$ may be assessed by a quality function $Q = Q(\mathcal{C})$, called **modularity**, which is defined as:

$$
\begin{aligned}
Q &= \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{|E|} - \left(\frac{2|E(C)| + \sum_{C' \in \mathcal{C}, C \neq C'} |E(C, C')|}{2|E|} \right)^2 \right] \\
&= \sum_{C \in \mathcal{C}} \left[\frac{|E(C)|}{|E|} - \left(\frac{\sum_{v \in C} \text{degree}(v)}{2|E|} \right)^2 \right],
\end{aligned}
$$

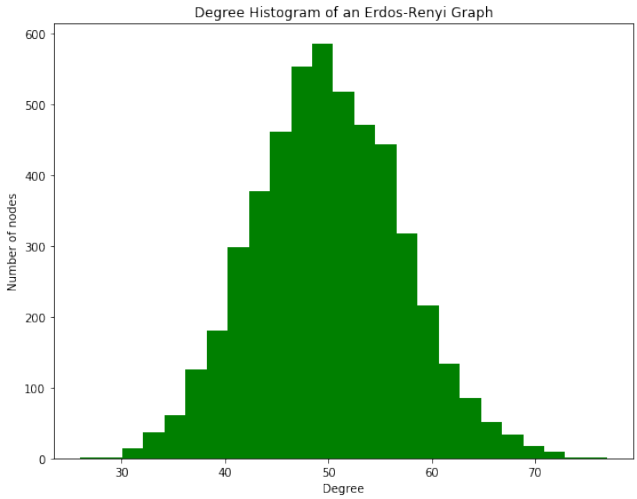
where $E(C)$ is the number of edges inside module C and $E(C, C')$ is the number of edges among modules C and C'. Essentially, modularity compares the number of edges inside a given module with the expected value for a randomized graph of the same size and same degree sequence.

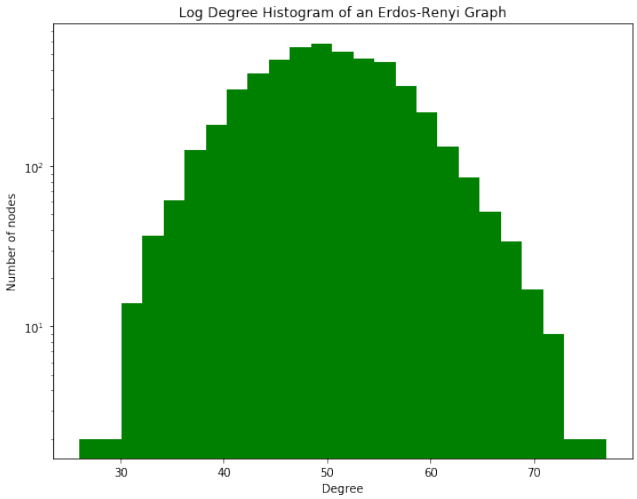
- A clustering that maximizes modularity is usually called **community partition** and the corresponding modules are called **communities**.

Communities of the graph of Florentine Families

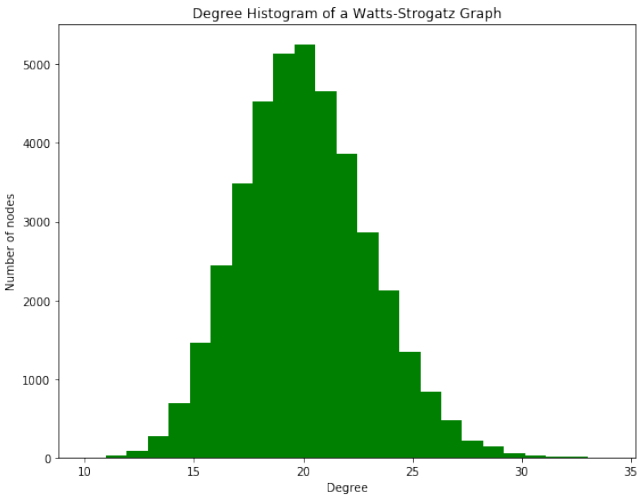


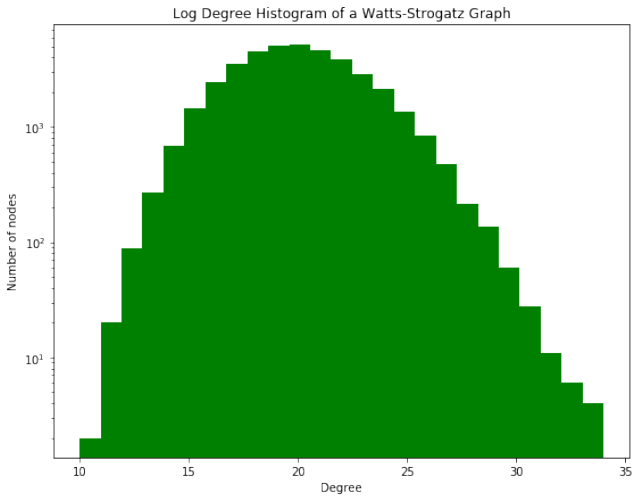
Histogram of an Erdos-Renyi random graph (n=5000, p=0.1)



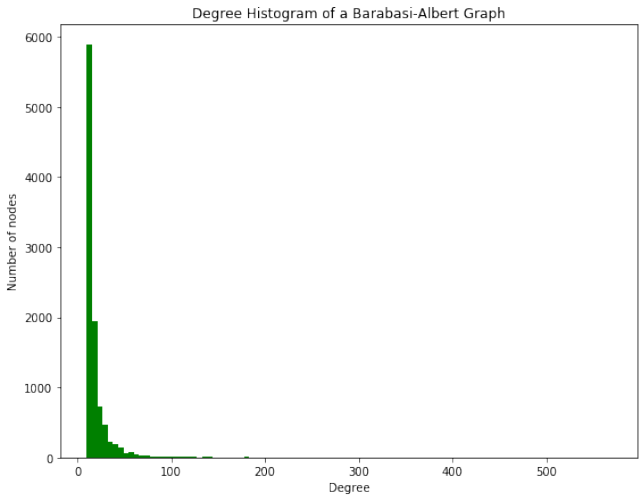


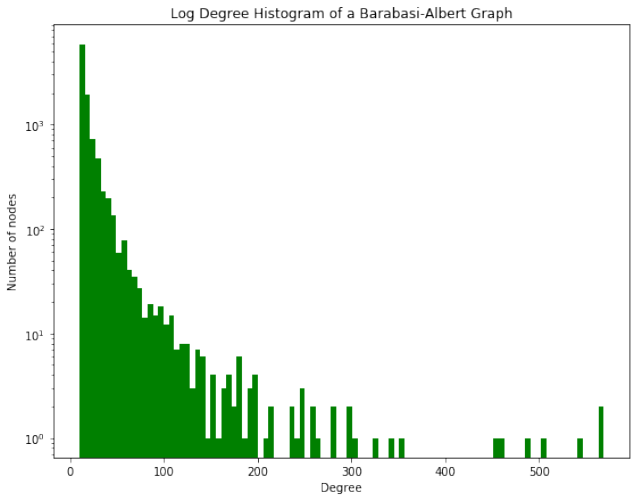
Histogram of a Watts-Strogatz Small-Worlds random graph (n=40000, k=20, p=0.8)



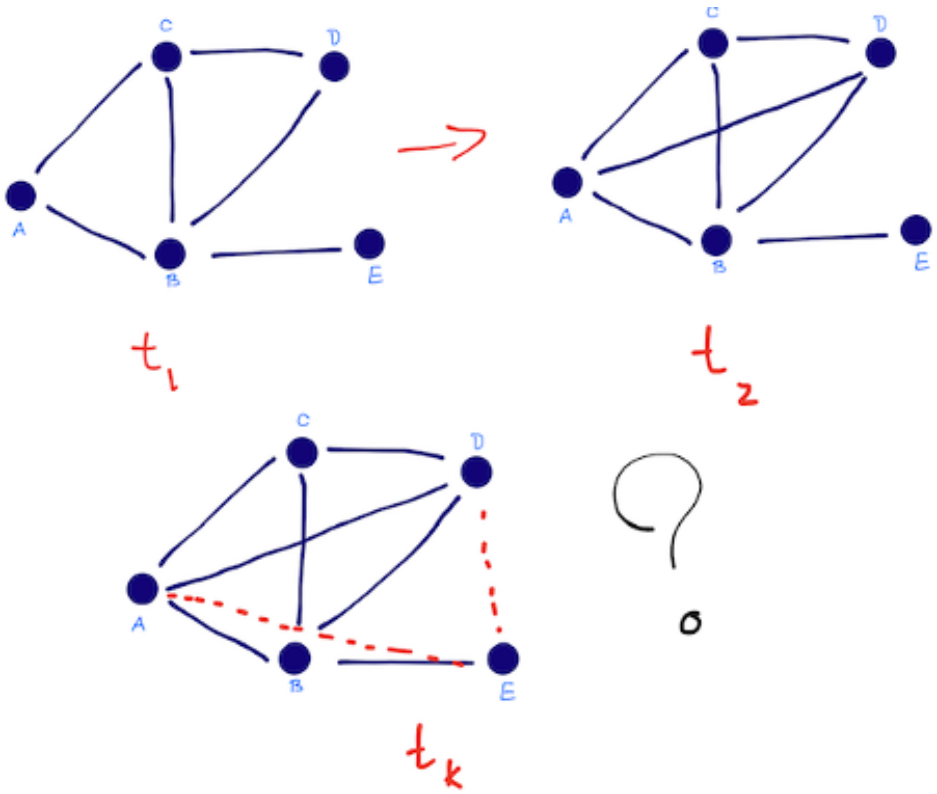


Histogram of a Barabasi-Albert Preferential Attachment random graph (n=10000, m=10): Scale-Free Network





Statistical network analysis: Link prediction algorithms



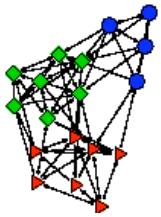
Statistical network analysis: Exponential Random Graph Models (ERGM)

- MLE approximation
- MCMC simulations
- Goodness of fit

ERGM class $\exp\{\boldsymbol{\theta}^t \mathbf{g}(\mathbf{y})\}$ $\longrightarrow$ (approx) MLE $\hat{\boldsymbol{\theta}}$ $\longrightarrow$ Fitted ERGM $\exp\{\hat{\boldsymbol{\theta}}^t \mathbf{g}(\mathbf{y})\}$

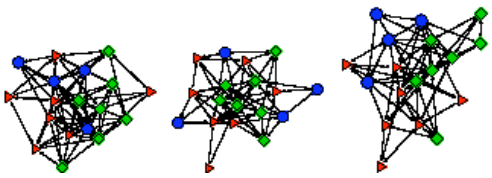
$\uparrow$

$\mathbf{y}_{\text{obs}}$



$\downarrow$

Randomly generated networks $\tilde{\mathbf{Y}}_1, \tilde{\mathbf{Y}}_2, \tilde{\mathbf{Y}}_3, \ldots$



Computational Social Network Analysis

Network Science: Applications and Processes

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

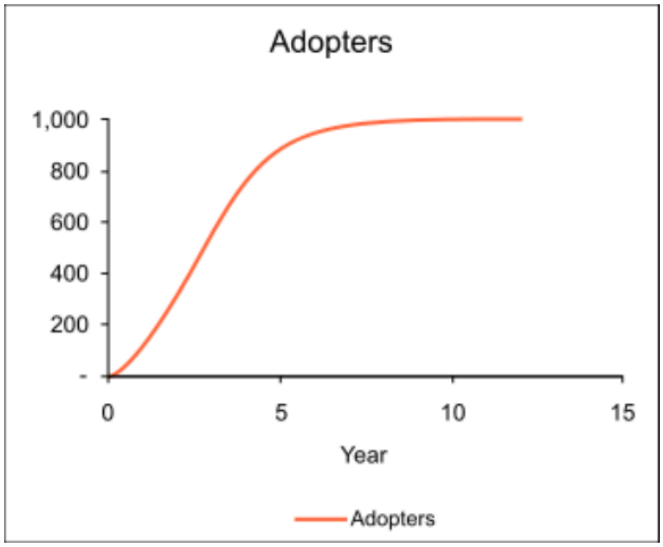
Linköping University, Norrköping, Sweden

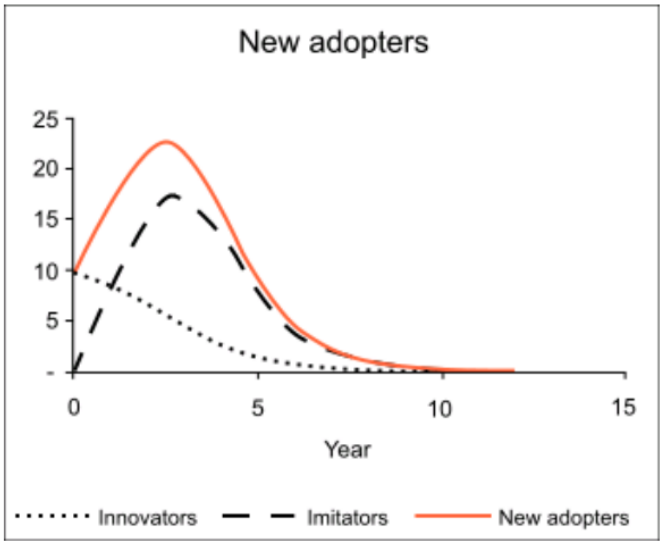
Spring 2019

Diffusion

- Ideas, opinions, beliefs
- Habits, consumption patterns
- Adoption of innovations or new technologies etc.
- Contagious diseases
- Behavioral contagion and imitation

Innovation and adoption

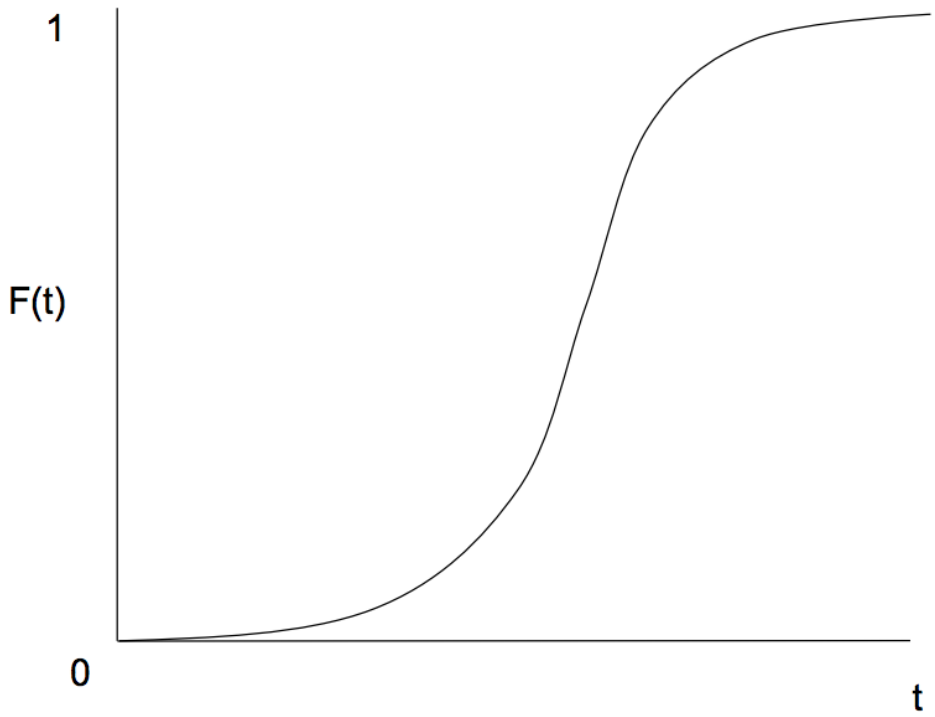




The Bass Model

- $\boxed{\frac{d}{dt}F(t) = (p + qF(t))(1 - F(t))}$
- where:
  - $F(t)$ is the fraction of the population who have adopted some action at time t
  - p is the coefficient of innovation
  - q is the coefficient of imitation
- $\boxed{F(t) = \frac{1-e^{-(p+q)t}}{1+\frac{q}{p}e^{-(p+q)t}}}$

The S-shaped diffusion curve



The SIR epidemic model

- S (Susceptible) $\longrightarrow$ I (Infected) $\longrightarrow$ R (Recovered)

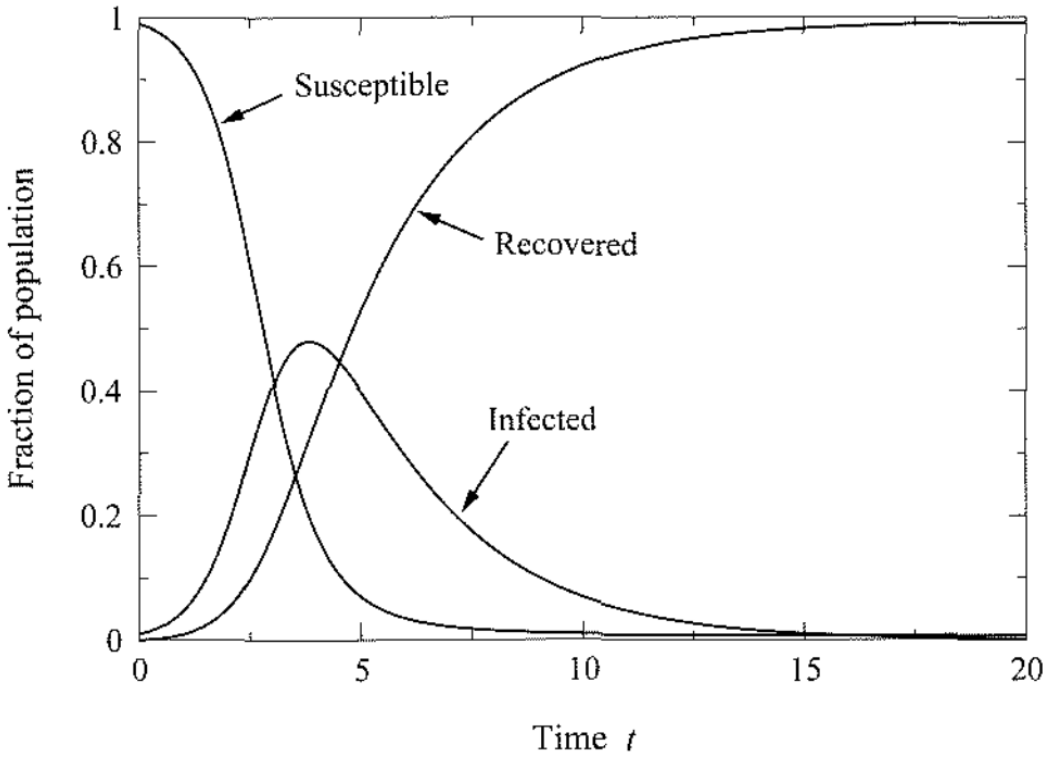
equations for the SIR model are

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\beta s x,$$
$$\frac{\mathrm{d}x}{\mathrm{d}t} = \beta s x - \gamma x,$$
$$\frac{\mathrm{d}r}{\mathrm{d}t} = \gamma x,$$

and in addition the three variables necessarily satisfy

$$s + x + r = 1.$$

Time evolution of the SIR model



The Friedkin–Johnsen Model of Network Influence

- Let $G = (V, E)$ be a **graph** with n nodes, identified as **persons** in the **node**–set V, and interacting with each other along the **edge**-set E.
- For each node $i \in V$ and each time step $k = 0, 1, 2, \ldots$, the **opinion** of i at time k is denoted by $x_i^{(k)} \in \mathbb{R}$.
- The opinion of i at time k is updated at the subsequent time step $k + 1$ according to the following iterative scheme of the **Friedkin–Johnsen Social Influence Model**:

$$x_i^{(k+1)} = s_i \, N \, x_i^{(k)} + (1 - s_i) \, x_i^{(0)},$$

  - where $N \, x_i^{(k)}$ is the *average* opinion held by i's neighbors at time step k and
  - the scalar parameter $s_i \in [0, 1]$ is called **susceptibility coefficient** of person/node i.

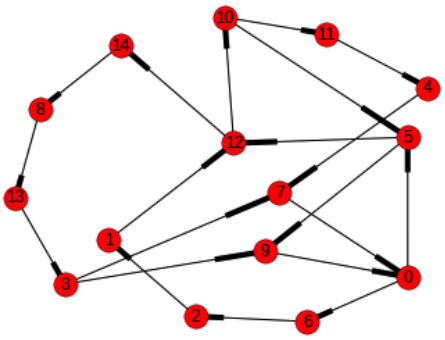
Remarks on Influence Susceptibility Coefficients

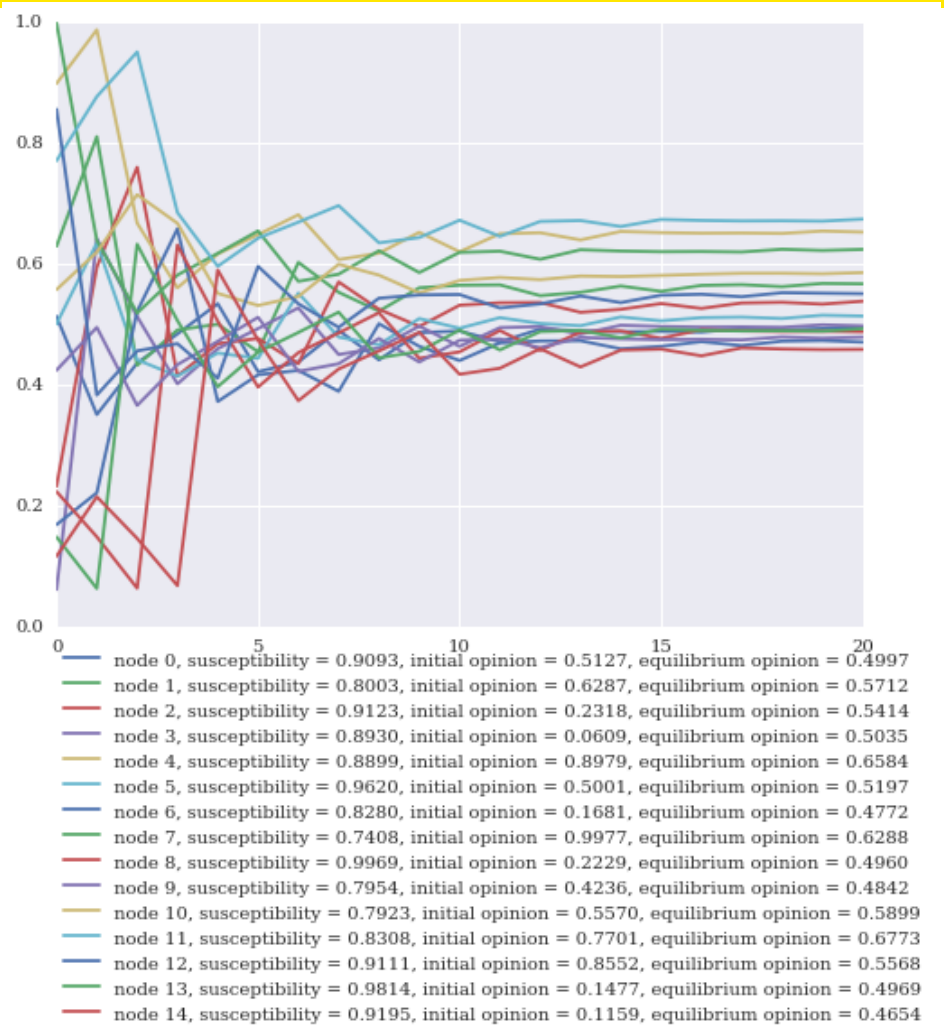
- When $s_i = 0$, i's opinion does not change ($x_i^{(k)} = x_i^{(0)}$, for every time iteration $k = 1, 2, \ldots$). Such a person/node is called **persistent** or **stubborn** in holding the same opinion without being influenced by anybody else.

- When $s_i = 1$, i is always adopting the average neighbors' opinion $N\,x_i^{(k)}$ without taking into account the initial opinion $x_i^{(0)}$. Such a person/node is called **malleable** or fully **conforming** to the neighbors' influences.

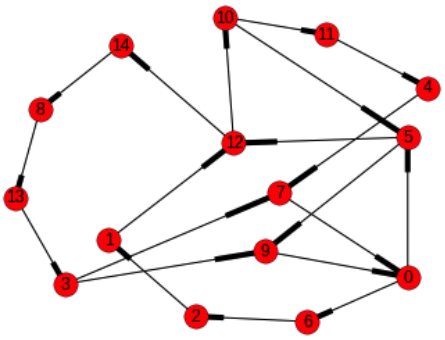
- When $0 < s_i < 1$, i's opinion is interpolated in between the average neighbors' opinion $N\,x_i^{(k)}$ and the initial opinion $x_i^{(0)}$, in such a way that the exact position of the resulting i's opinion is weighted as a convex combination through s_i.

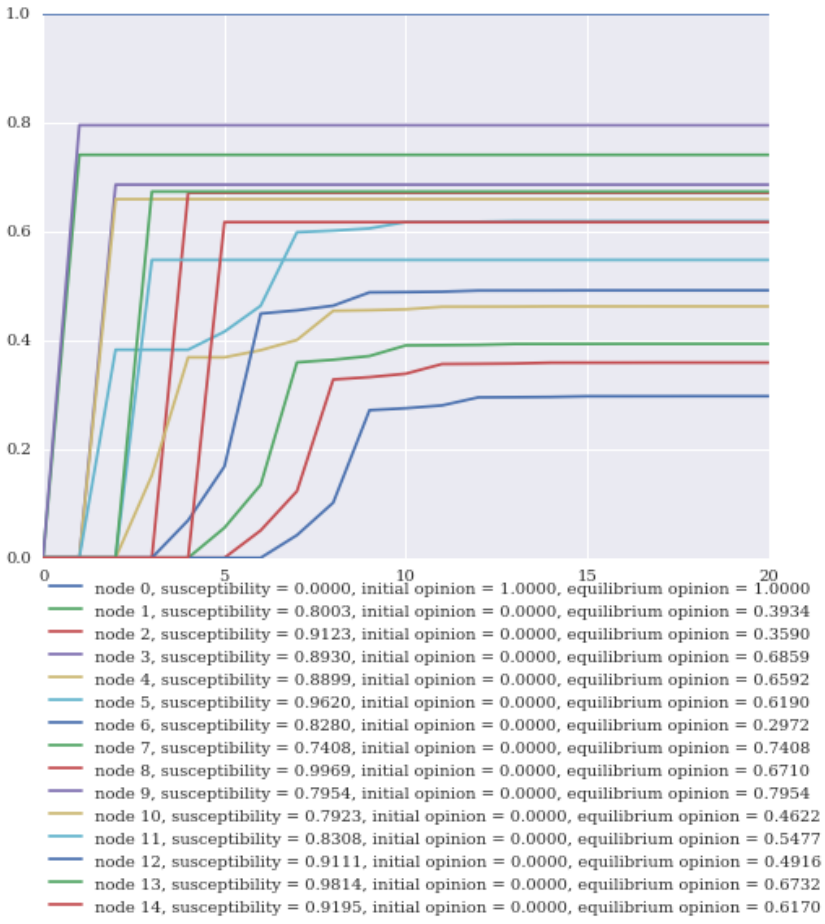
An Example of Friedkin–Johnsen Network Influence



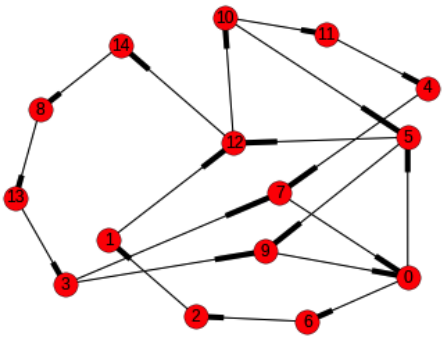


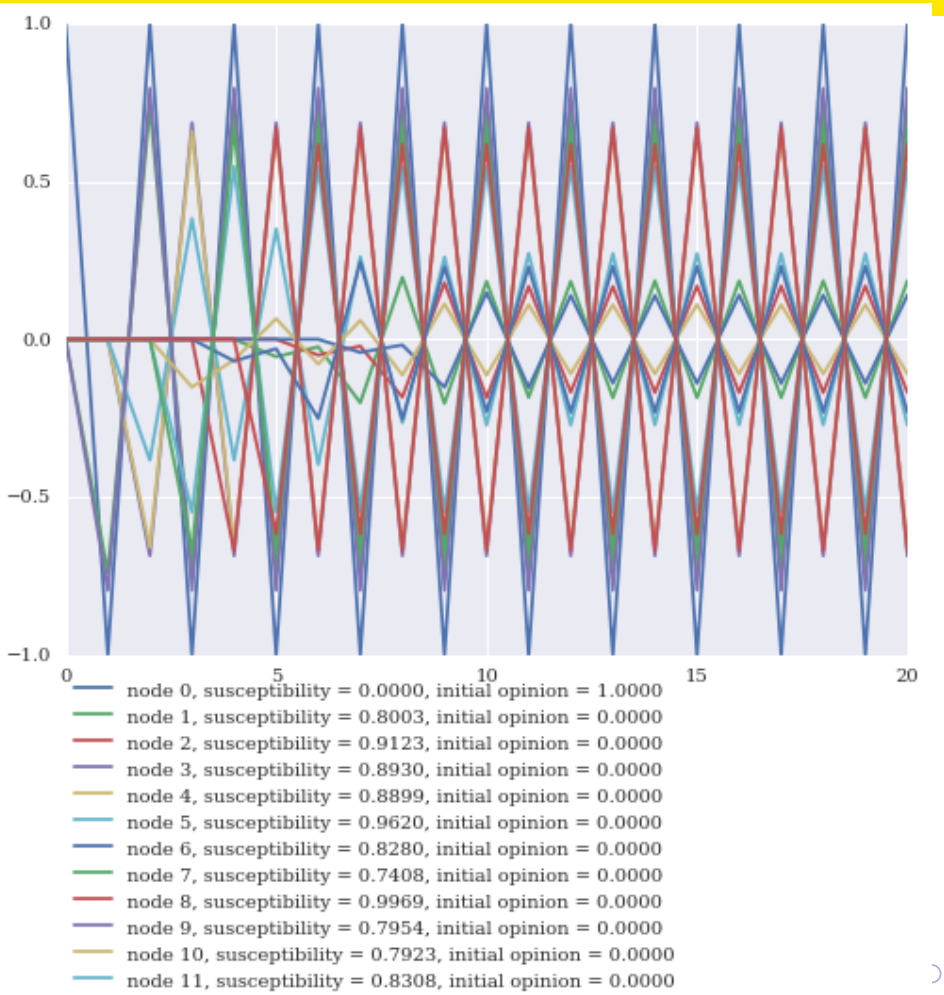
An Example with One Source of Persistent Opinion Stimulation



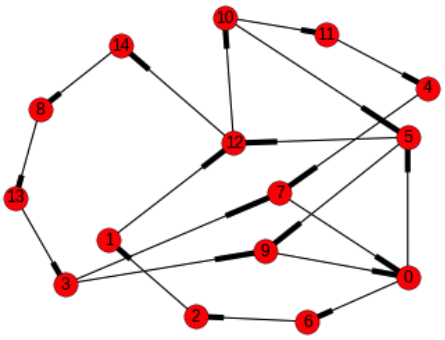


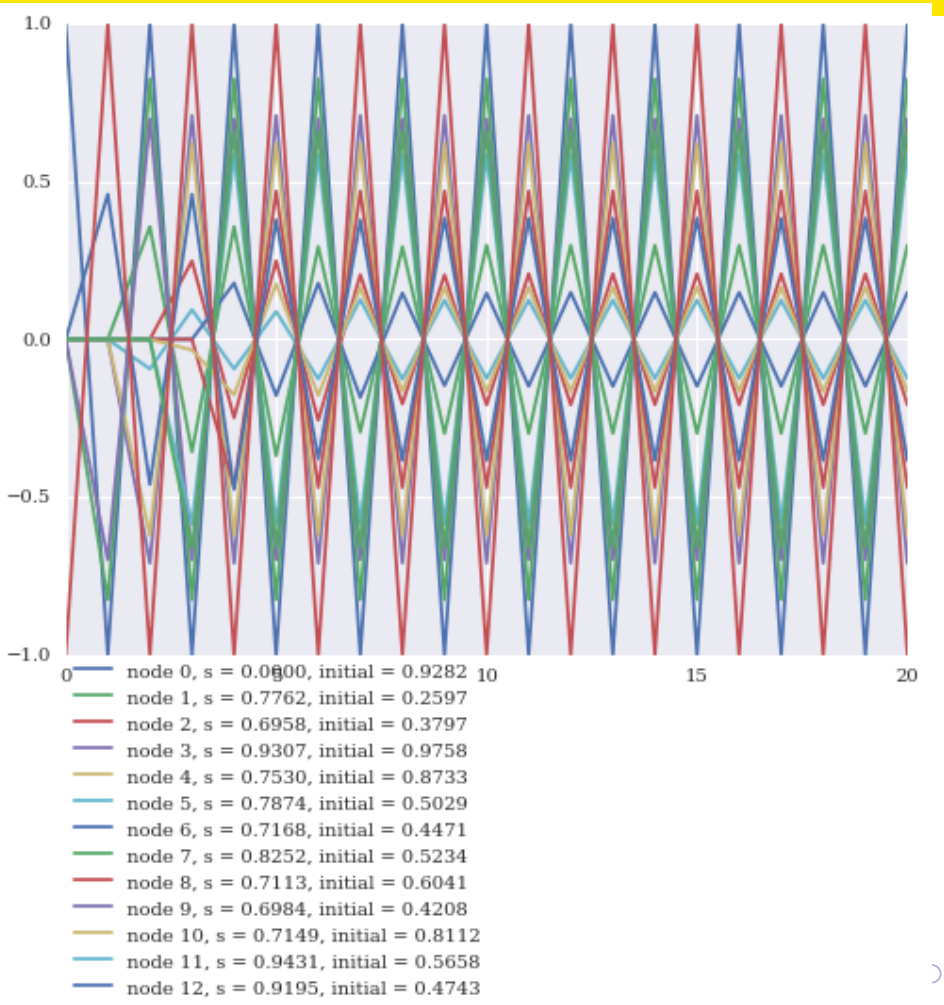
Synchronization from One Source of Alternating Persistent Opinion Stimulation





Synchronization from Two Sources of Alternating Persistent Opinion Stimulation





Computational Social Network Analysis

Social Network Analysis of Data Extracted from the Web

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

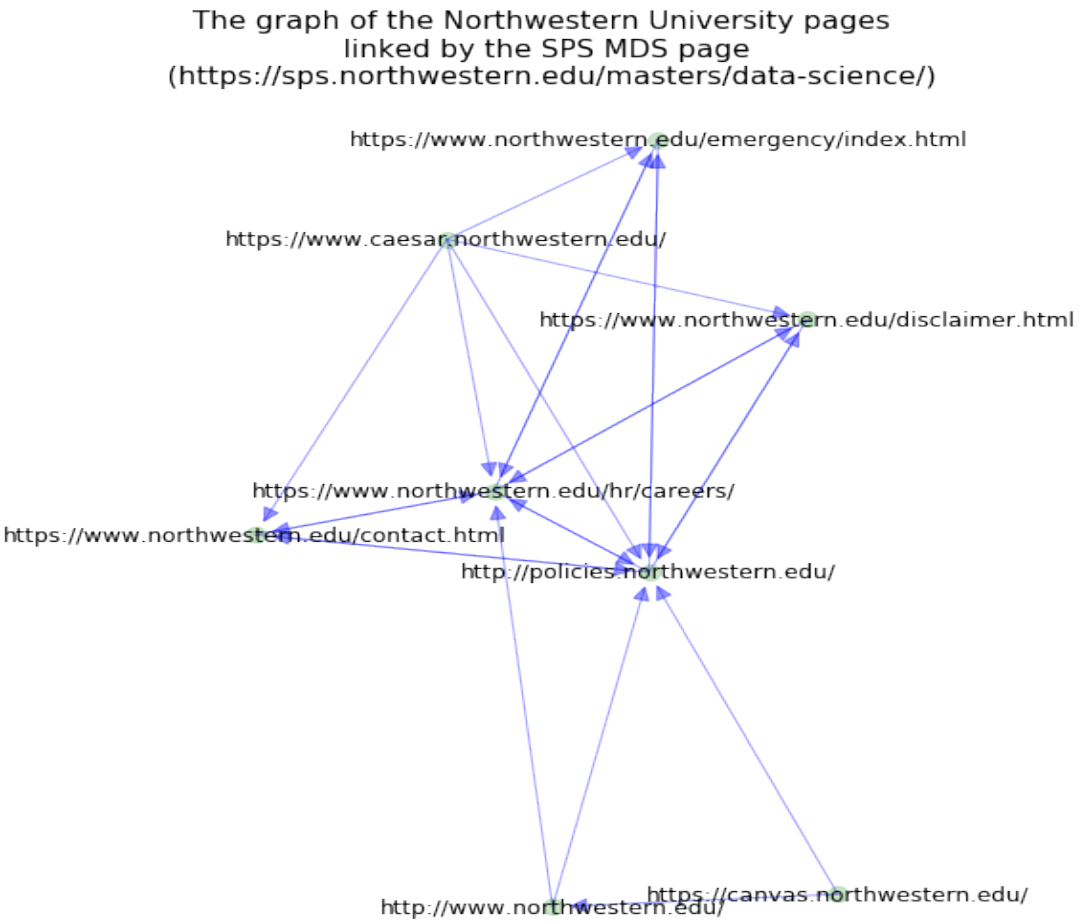
[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

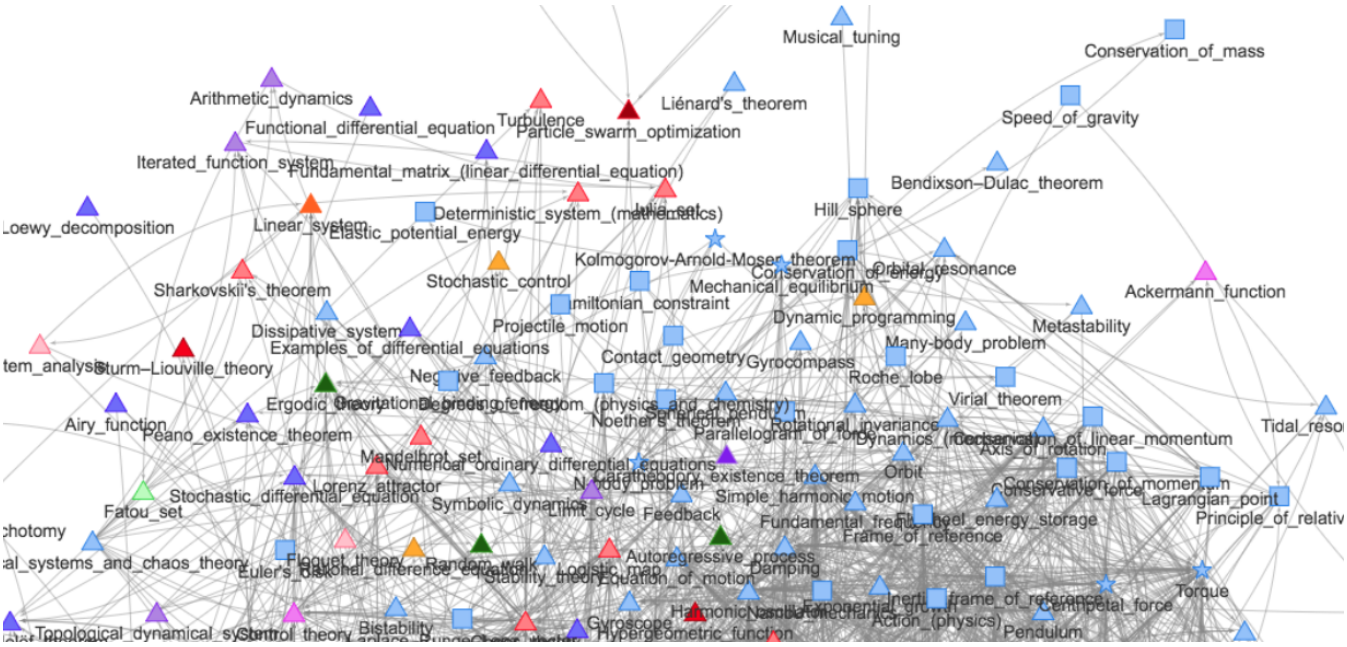
Linköping University, Norrköping, Sweden

Spring 2019

Networks of hyperlinks among web pages



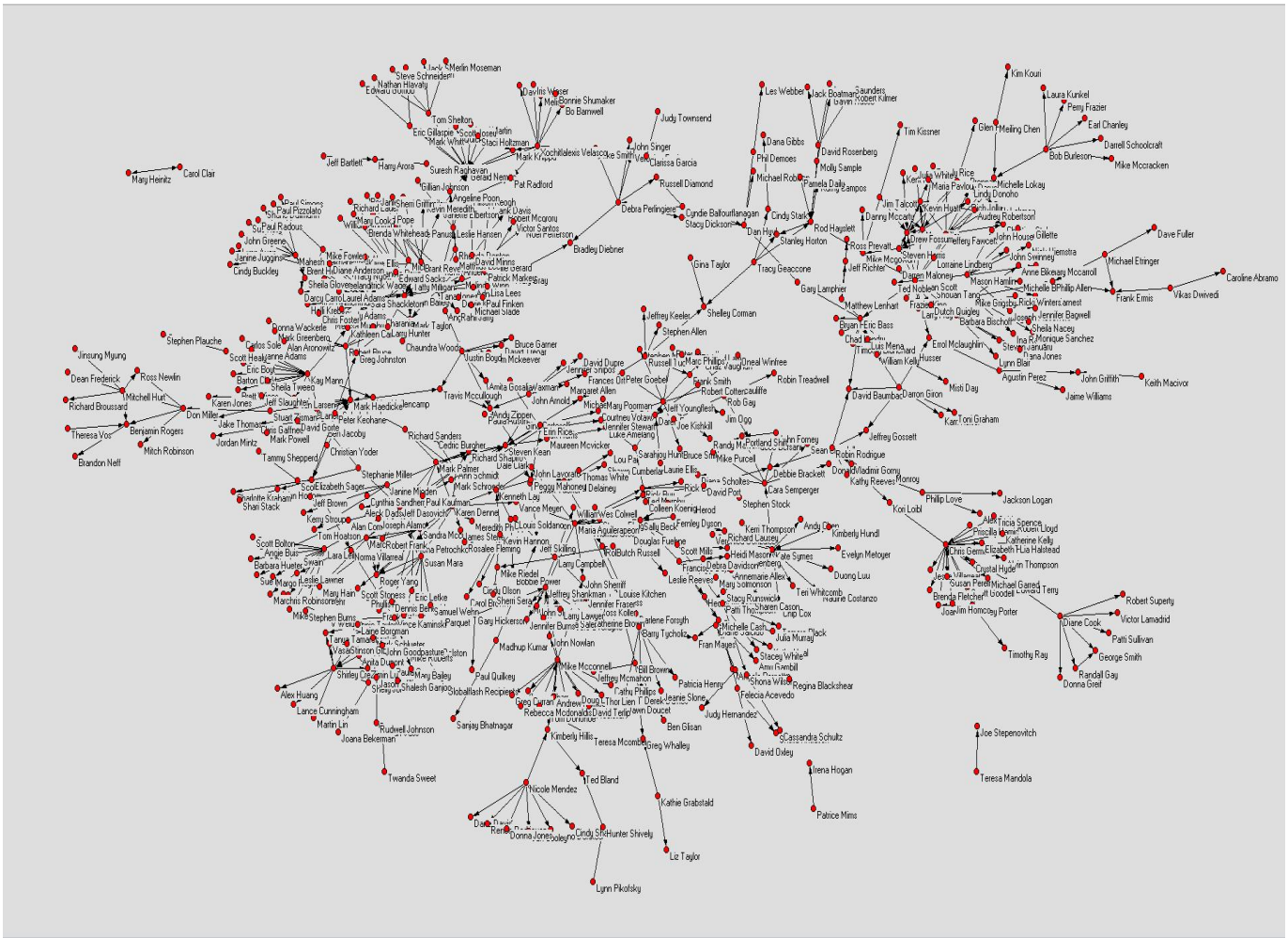
Networks of Wikipedia pages of topics



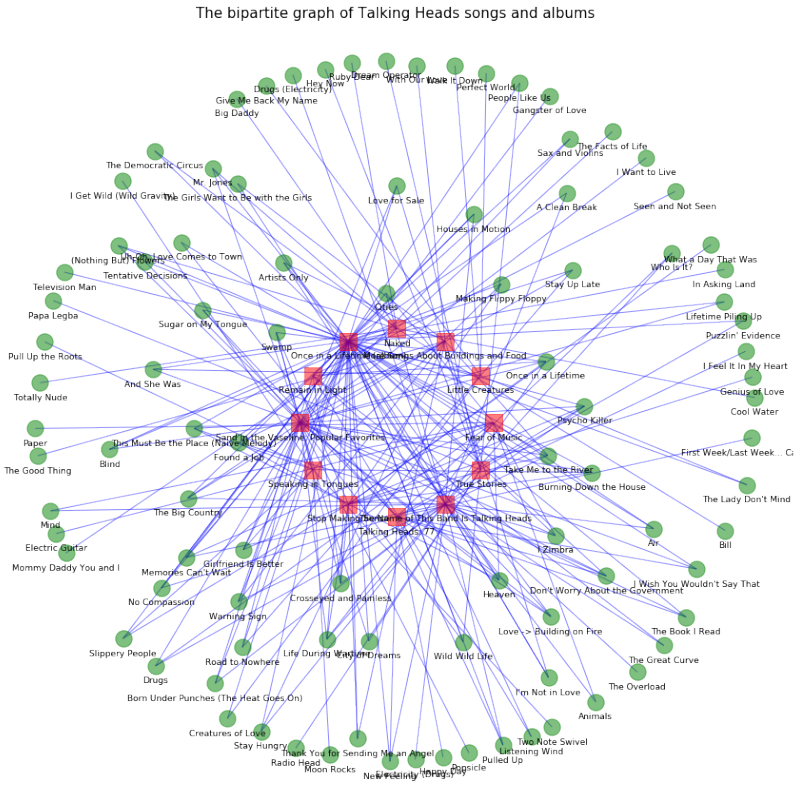
The citation network among Wikipedia pages on Dynamical Systems and Mechanics

https://medium.com/@mosabou/the-citation-network-among-wikipedia-pages-on-dynamical-systems-and-mechanics-

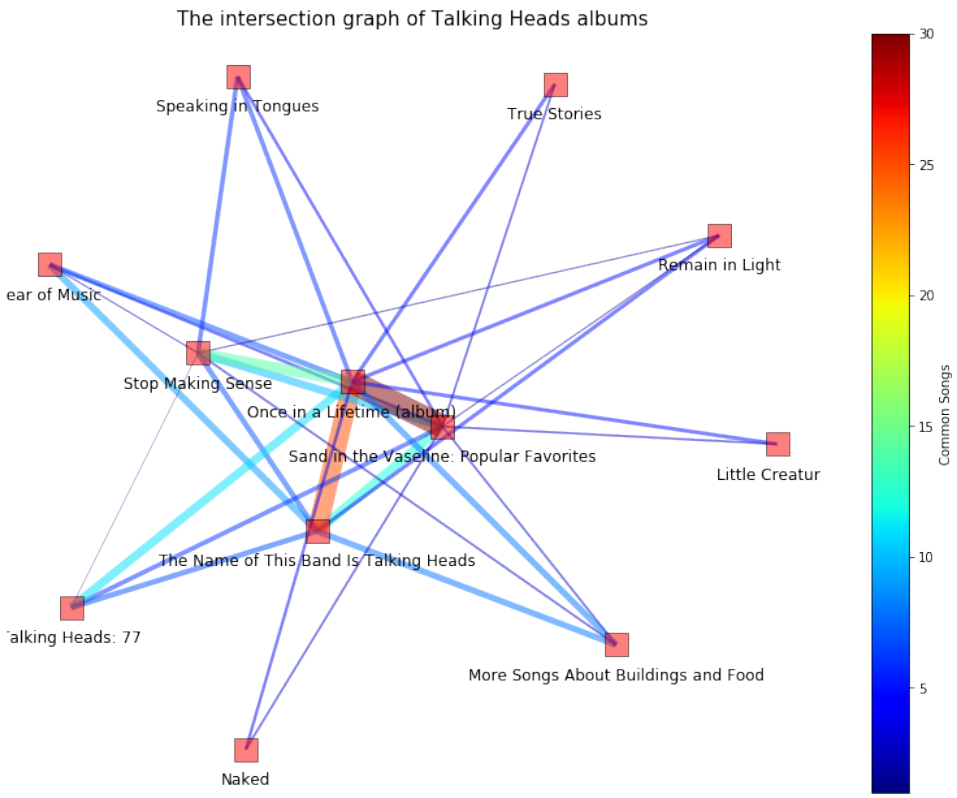
Networks of emails



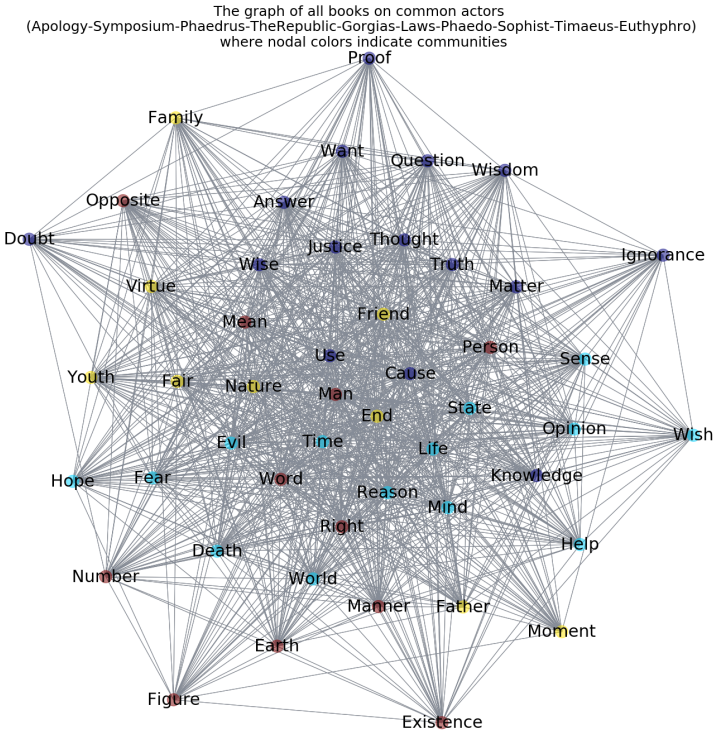
Networks of songs: Talking Heads bipartite graph of albums–songs



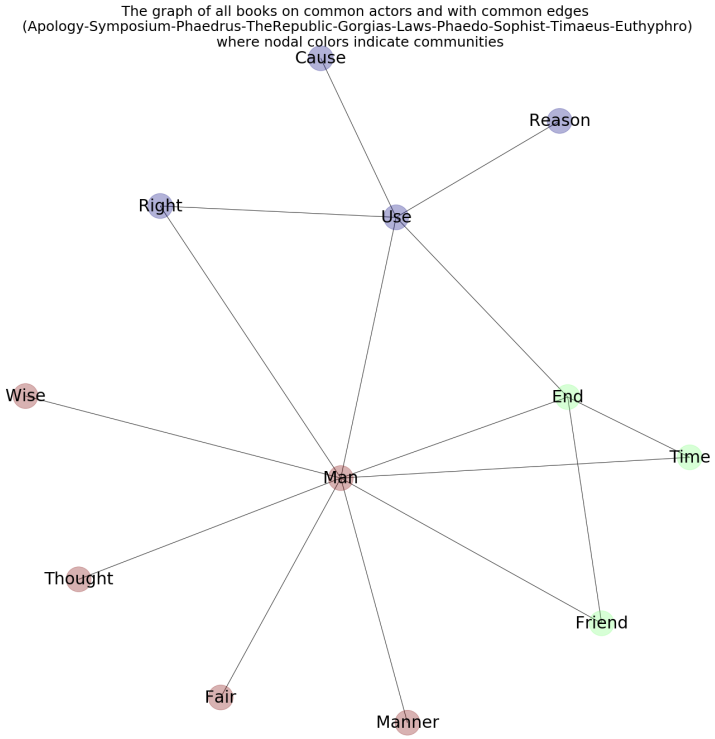
Networks of songs: Talking Heads intersection graph of albums



Wordnets: The sentential co-occurrence graph of common words in all Plato's Ouvre



Wordnets: The subgraph of common sentential co-occurrences in all Plato's Ouvre



Computational Social Network Analysis

Social Network Analysis of Data Extracted from Social Media

Moses A. Boudourides[1]

Robert K. Merton Visiting Research Fellow 2019, IAS, LiU
Northwestern University School of Professional Studies

[1] Moses.Boudourides@northwestern.edu

The Institute for Analytical Sociology

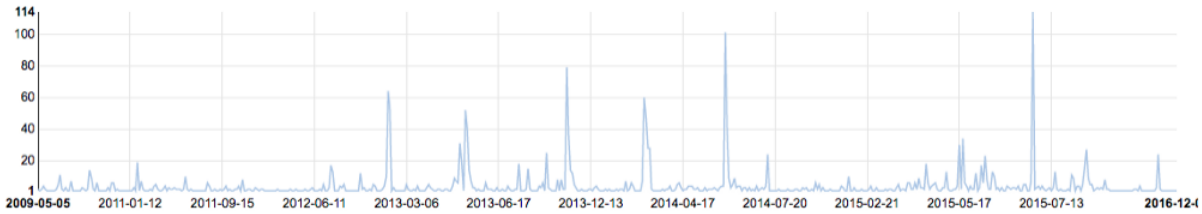
Linköping University, Norrköping, Sweden

Spring 2019

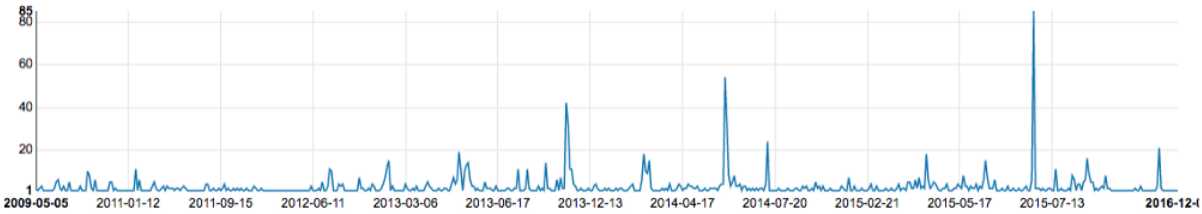
A case study of Twitter data & network analysis

- **#EUSealBan**
  - About 100K tweets based on search terms like #seal* in the period 2009–2016 were dowloaded by the Twitter API (by scraping the results of the TwitterAdvanced Search page).
  - Filtering only the #eu*, #ban, #wto hashtags resulted
  - 2513 tweets written (solely) in English by
  - 763 Twitter users ("tweeple") in the period
  - from January 4, 2009 to December 31, 2016

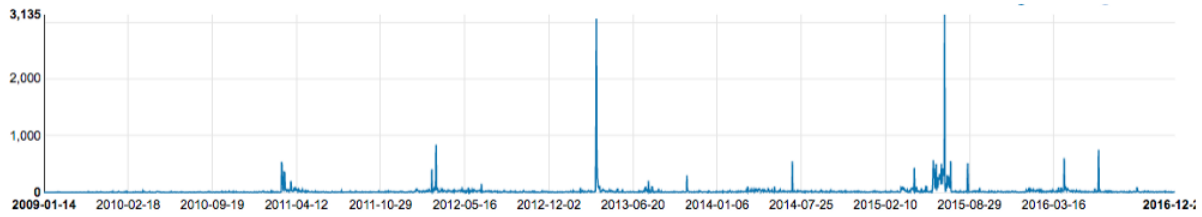
Timeseries of tweets, tweeple and mentions



The time series of the count of tweets.

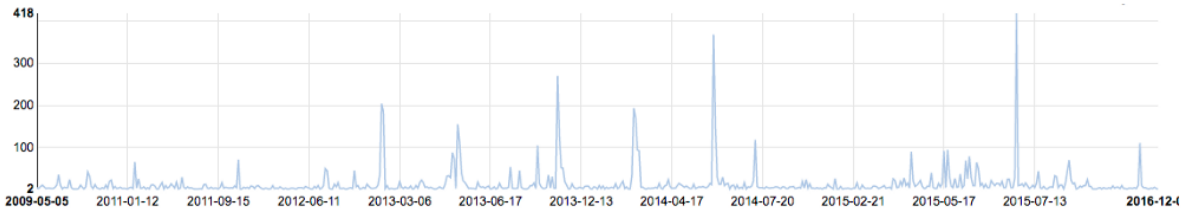


The time series of the count of tweeple.

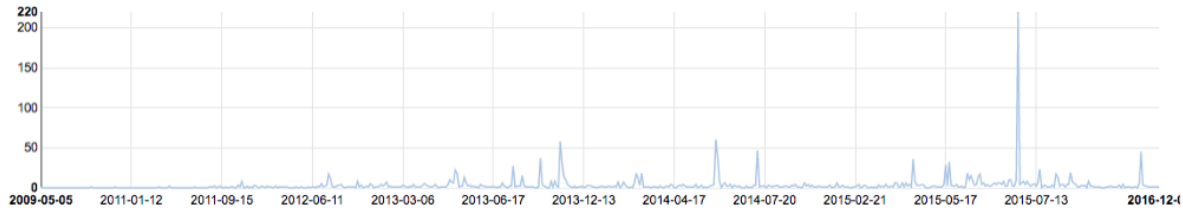


The time series of the count of mentions.

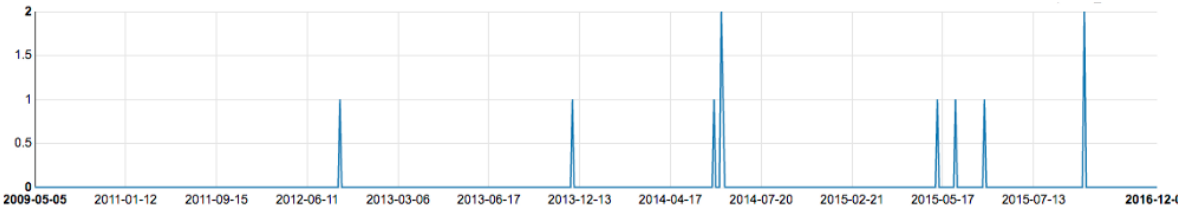
Timeseries of hashtags, embedded photos-URLs and videos



The time series of the count of hashtags.

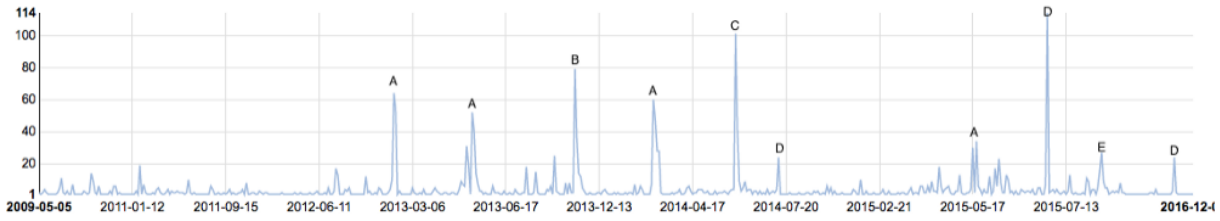


The time series of the count of embedded photos and URLs.



The time series of the count of embedded videos.

Peak detection in the timeline of tweets



The main episodes of the EU seal ban issue.

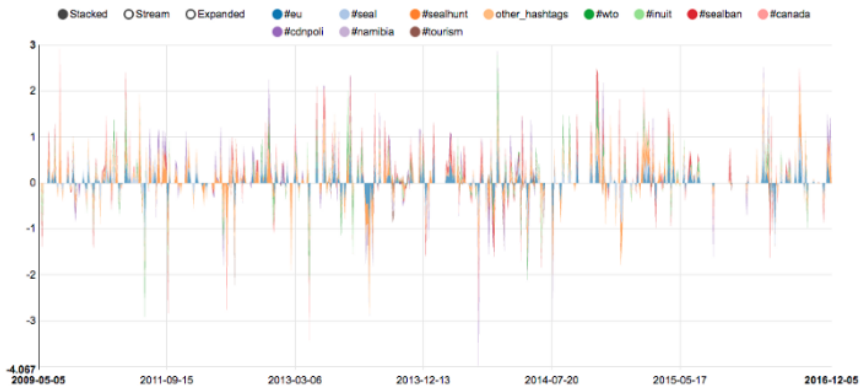
- Main episodes/events:
  - Seal hunting in Canada (February 2013, April 2013, March 2014, May 2015).
  - WTO confirms EU seal trade ban (November 2013).
  - WTO rejects Canada appeal against EU seal import ban (May 2014).
  - Seal hunting in Namibia (July 2015, July 2016).
  - EU toughens the seal ban regulations (September 2015).

Top hashtags

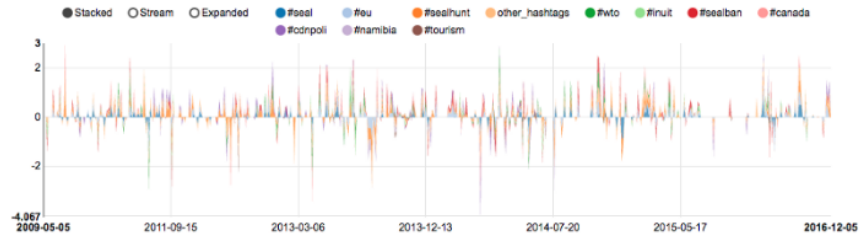
Hashtag	N of Tweets
#sealhunt	1467
#eu	1388
#wto	645
#seal	627
#sealban	597
#namibia	432
#inuit	345
#cdnpoli	291
#canada	261
#tourism	249

The 10 top hashtags.

Sentiment analysis of tweets in top hashtags

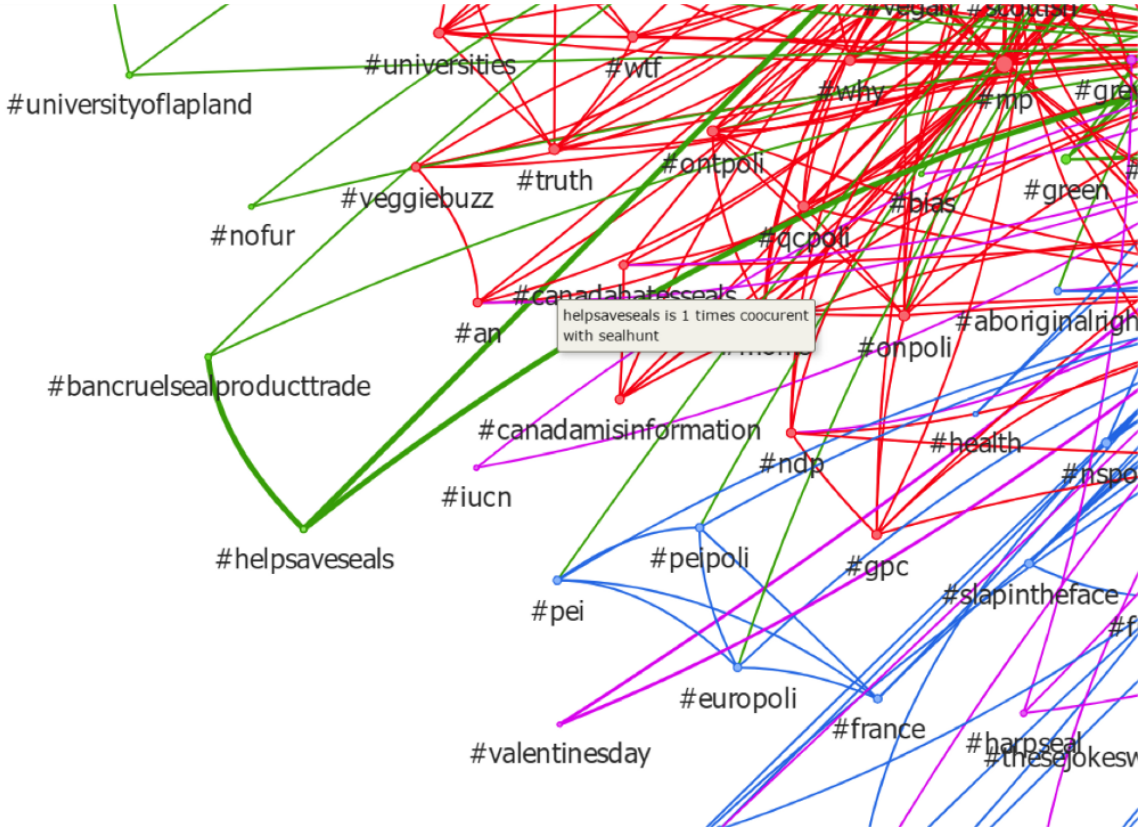


The subjectivity score of the sentiment analysis of tweets grouped in hashtags.

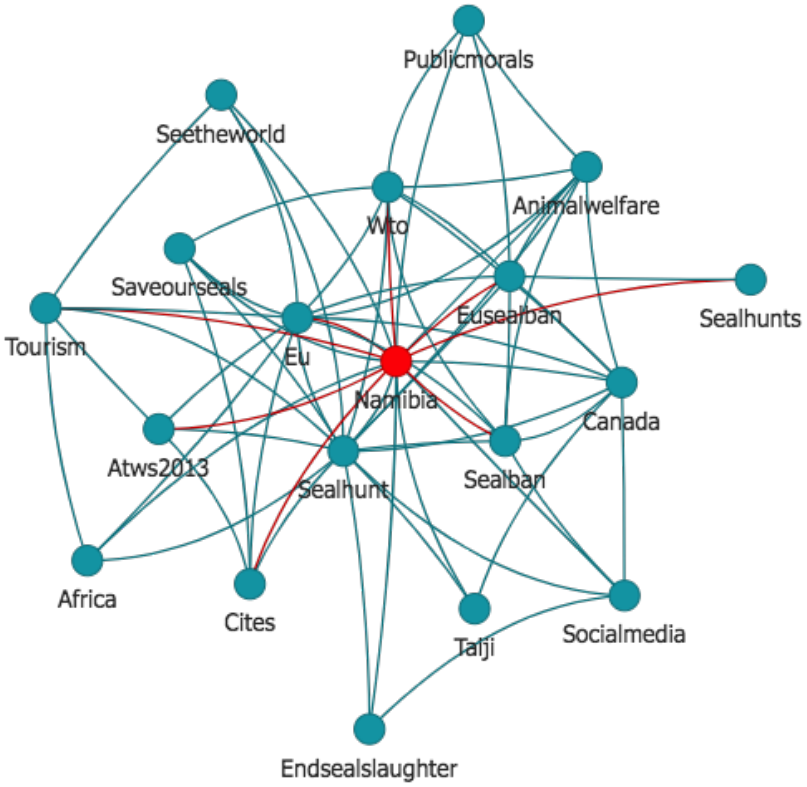


The polarity score of the sentiment analysis of tweets grouped in hashtags.

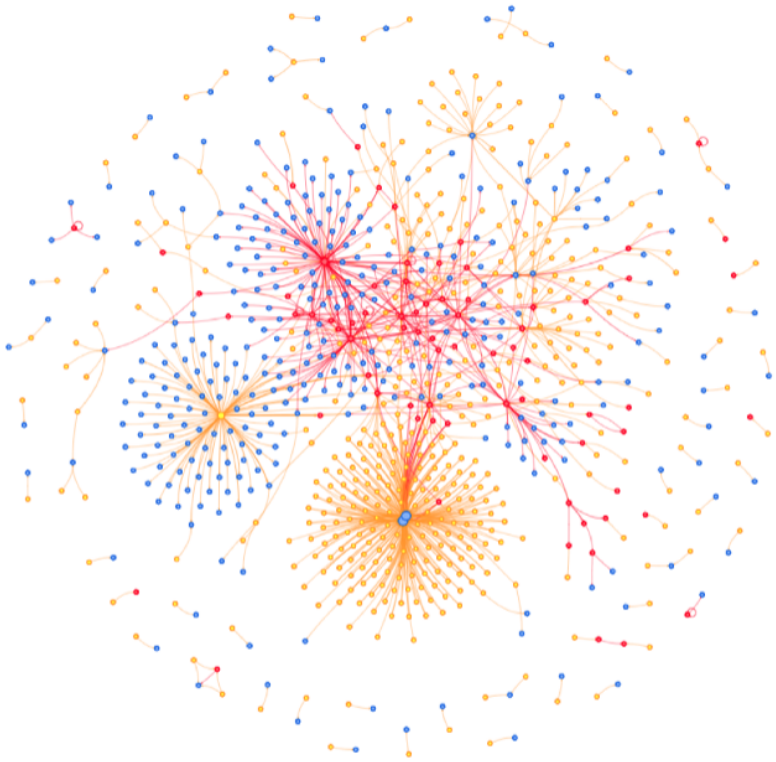
Network of co–occurring hashtags



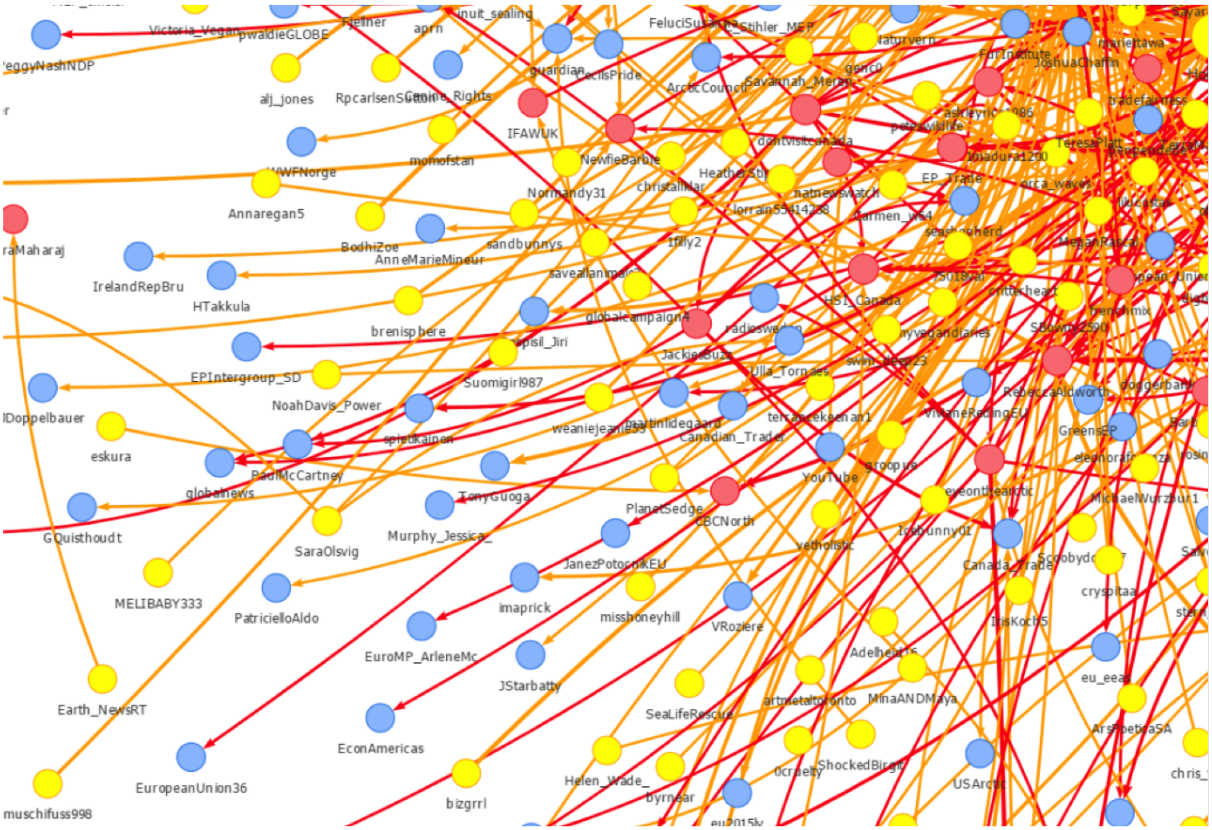
An ego–centric subraph of co–occurring hashtags around #Namibia



Network of mentions among tweeple



A detailed part of the graph of tweeple mentions

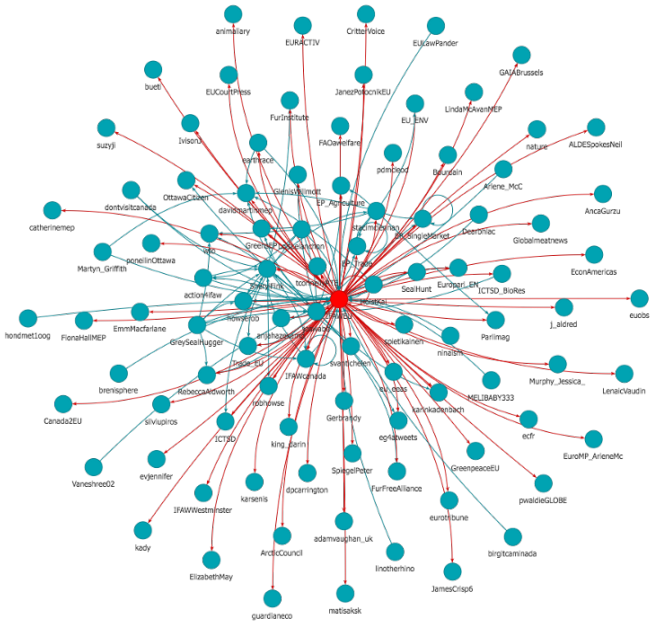


Influential tweeple according to the centralities of their mention-ing/-ed behavior

node	load	Katz	outdegree	indegree	closeness	hits_auths	hits_hub	betweenness	PageRank	eigenvector
HolstKai	0.0000	0.0088	0.1223	0.0000	0.1468	0.0000	0.0000	0.0000	0.0007	0.0000
IFAWEU	0.0101	0.2534	0.0950	0.0273	0.1037	0.0000	0.0003	0.0000	0.0126	0.2765
joswabe	0.0071	0.3373	0.0511	0.0131	0.0864	0.0014	0.0001	0.0000	0.0072	0.3754
madinuk	0.0015	0.0129	0.0333	0.0071	0.0851	0.0000	0.0000	0.0000	0.0024	0.0000
GreySealHugger	0.0019	0.0103	0.0309	0.0036	0.1129	0.0000	0.0001	0.0000	0.0014	0.0000
SherylFink	0.0061	0.5410	0.0238	0.0273	0.0832	0.0011	0.0000	0.0000	0.0167	0.6000
stacimclennan	0.0006	0.0612	0.0190	0.0024	0.0682	0.0000	0.0000	0.0000	0.0013	0.0616
marcelmason	0.0000	0.0088	0.0143	0.0000	0.0830	0.0000	0.0000	0.0000	0.0007	0.0000
dontvisitcanada	0.0002	0.0092	0.0143	0.0012	0.0820	0.0000	0.0000	0.0000	0.0010	0.0000
veganrick	0.0000	0.0092	0.0095	0.0012	0.0096	0.0000	0.0000	0.0000	0.0013	0.0000
eyeonthearctic	0.0010	0.0124	0.0083	0.0024	0.0087	0.0000	0.0000	0.0000	0.0026	0.0034
TheSealsOfNam	0.0037	0.0213	0.0083	0.0261	0.0619	0.0024	0.0396	0.0000	0.0066	0.0000

Centrality indices of the top 10 tweeple in their graph of mentions (sorted by Out-Degree).

An ego–centric subraph of mention-ing/-ed tweeple around IFAWEU

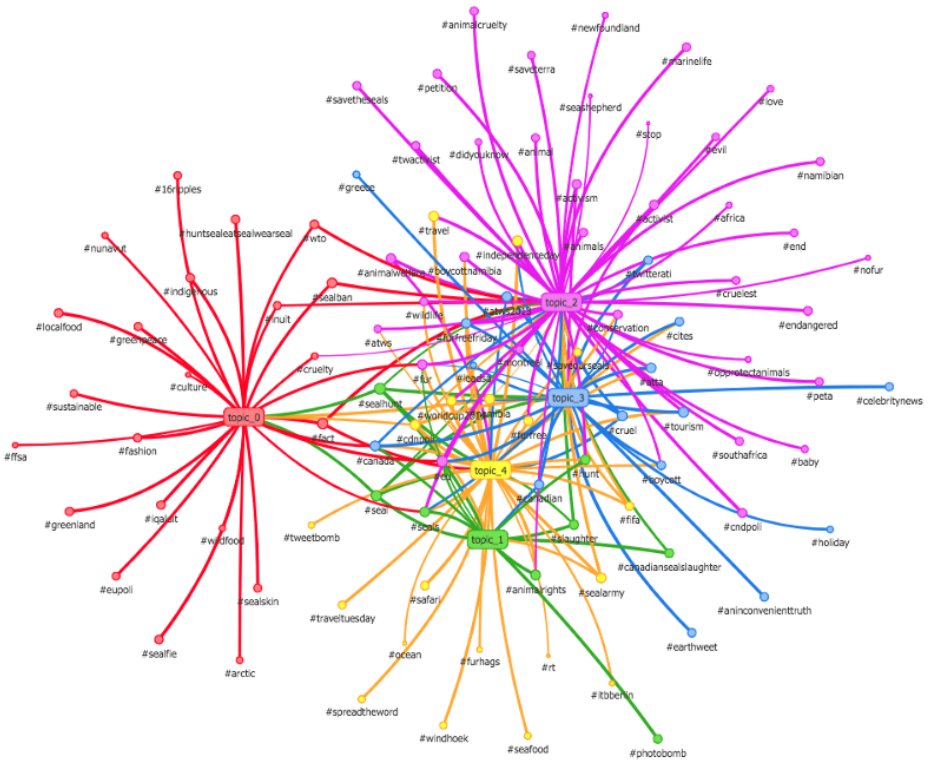


Topic Modeling in the content of tweets

Topic 0: Traditional hunt
sealhunt (4.8%), inuit (3.4%), seal (2.3%), cdnpoli (1.6%), wildfood (1.3%), sealskin (1.2%), sustainable (0.9%), know (0/9%), canada (0.9%), need (0.9%
Topic 1: Seal hunt politics
sealhunt (7.4%), cdnpoli (4.6%), seal (3.2%), namibia (2.4%), canada (2.1%), seals (2.0%), canadian (1.7%), baby (1.6%), end (1.6%),prime (1.2%)
Topic 2: Stop the seal hunt
sealhunt (7.9%), namibia (4.6%), seal (3.9%), slaughter (2.0%), canada (1.6%), hunt (1.3%), end (1.2%), , help (1.1%), stop (1.1%)
Topic 3: Exposure of Namimbia hunt
sealhunt (11.2%), namibia (11.1%), seal (1.4%), footage (1.2%), via (1.1%), baby (1.1%), cruel (1.1%), blood (1.0%), see (0.9%)
Topic 4: Boycott Namimbia
sealhunt (11.4%), namibia (10.3%), boycott (6.5%), via (3.8%), join (3.6%), end (3.1%), help (2.7%), seals(2.5%), pls (2.2%), save (2.2%)

The 5 topics of the #EUSealBan Twitter dataset.

The bipartite graph of topics and hashtags



The bipartite graph of topics and tweeple