Hubs

ivanovitch.silva@ufrn.br
@ivanovitchm

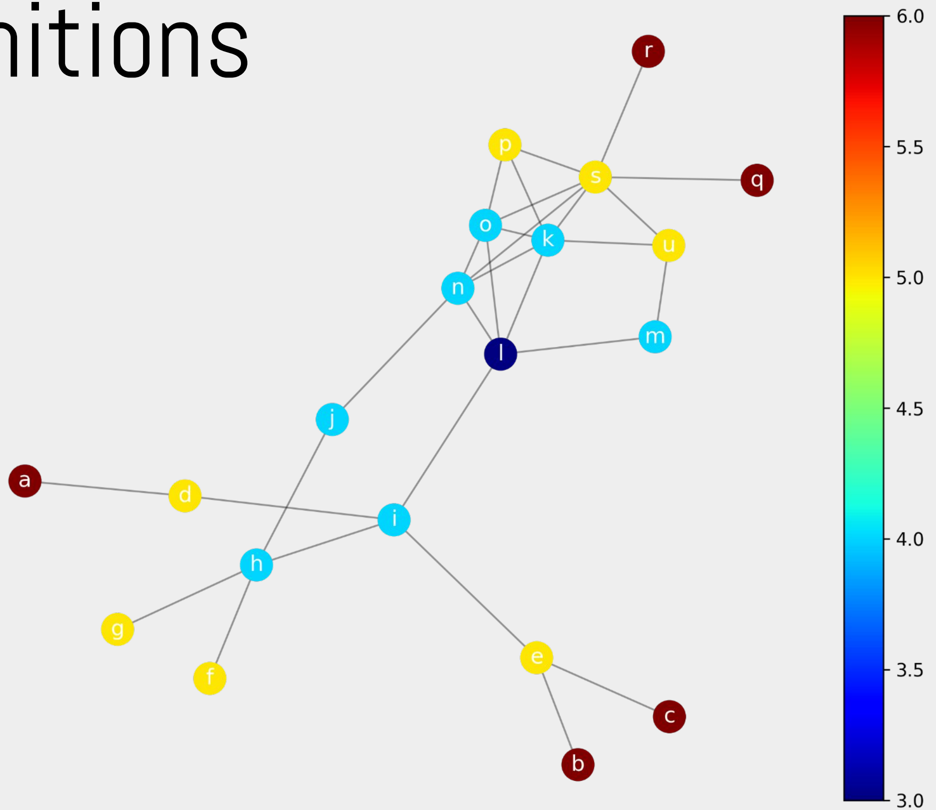
Hub: (n.) a center around which other things revolve or from which they radiate; a focus of activity, authority, commerce, transportation, etc

Some basic definitions

Eccentricity

It is the maximum distance from a node to all other node in the network.

```
nx.eccentricity(g)
{'a': 6, 'b': 6, 'c': 6, 'd': 5,
 'e': 5, 'f': 5, 'g': 5, 'h': 4,
 'i': 4, 'j': 4, 'k': 4, 'l': 3,
 'm': 4, 'n': 4, 'o': 4, 'p': 5,
 'q': 6, 'r': 6, 's': 5, 'u': 5}
```

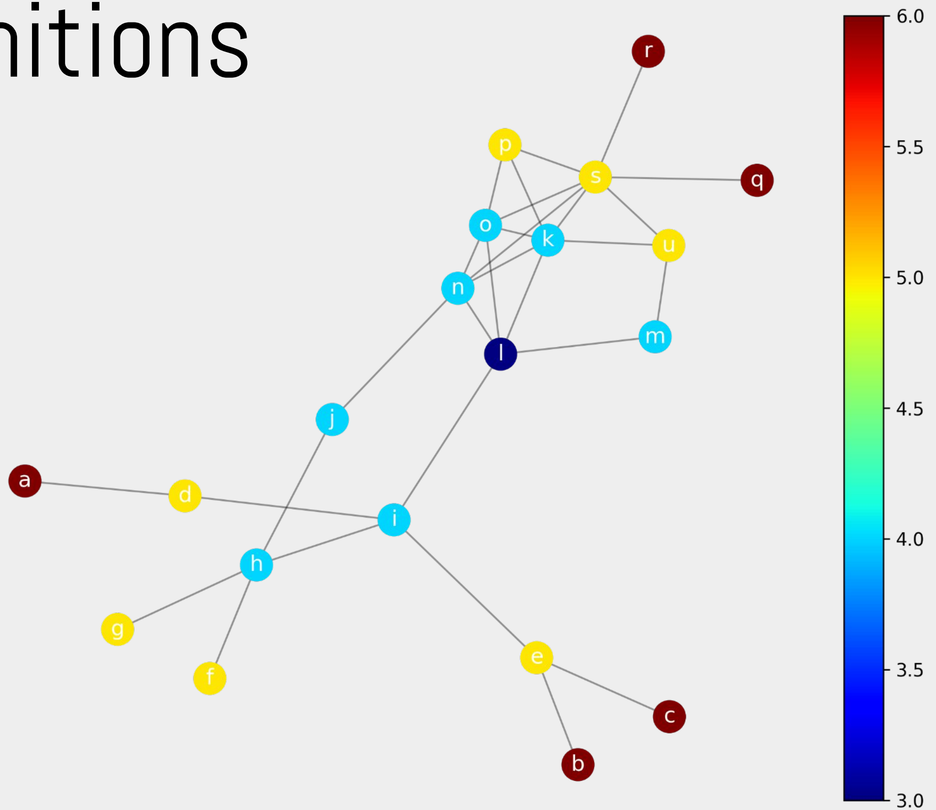


Some basic definitions

Diameter

The diameter of a network is the maximum eccentricity.

```
nx.eccentricity(g)
{'a': 6, 'b': 6, 'c': 6, 'd': 5,
 'e': 5, 'f': 5, 'g': 5, 'h': 4,
 'i': 4, 'j': 4, 'k': 4, 'l': 3,
 'm': 4, 'n': 4, 'o': 4, 'p': 5,
 'q': 6, 'r': 6, 's': 5, 'u': 5}

nx.diameter(g)
6

[k for k,v in nx.eccentricity(g).items()
if v == nx.diameter(g)]
['a', 'b', 'c', 'q', 'r']
```



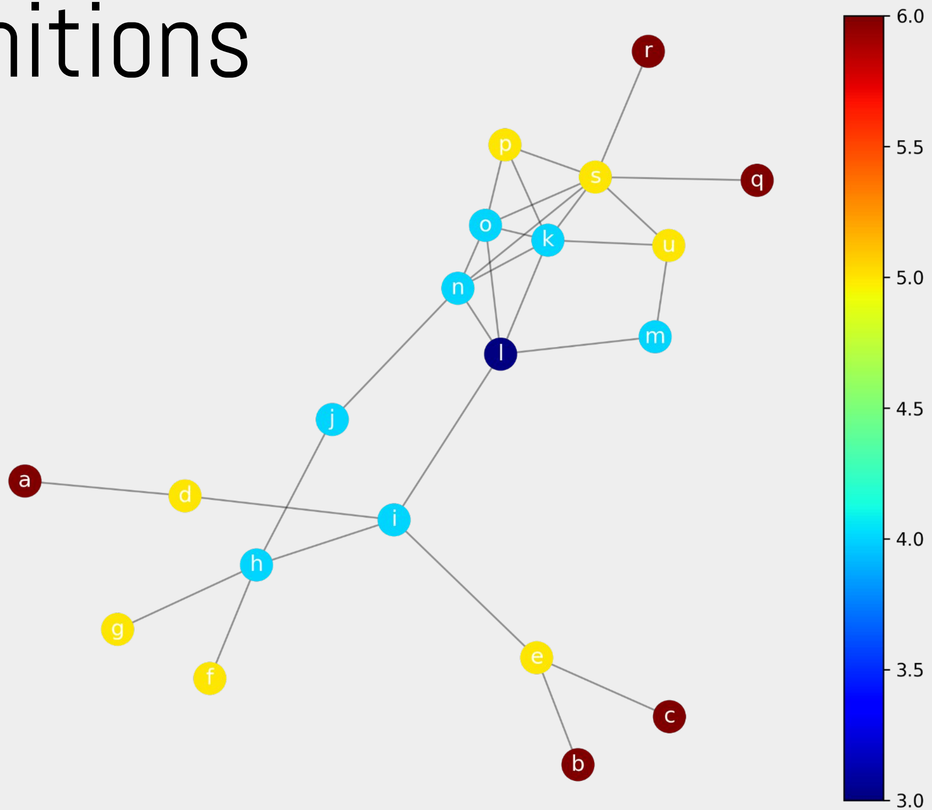
NetworkX
Network Analysis in Python

Some basic definitions

Periphery

The periphery of a network is a set of all nodes whose eccentricity equals the diameter.

```
nx.eccentricity(g)
{'a': 6, 'b': 6, 'c': 6, 'd': 5,
 'e': 5, 'f': 5, 'g': 5, 'h': 4,
 'i': 4, 'j': 4, 'k': 4, 'l': 3,
 'm': 4, 'n': 4, 'o': 4, 'p': 5,
 'q': 6, 'r': 6, 's': 5, 'u': 5}

nx.diameter(g)
6

nx.periphery(g)
['a', 'b', 'c', 'q', 'r']
```



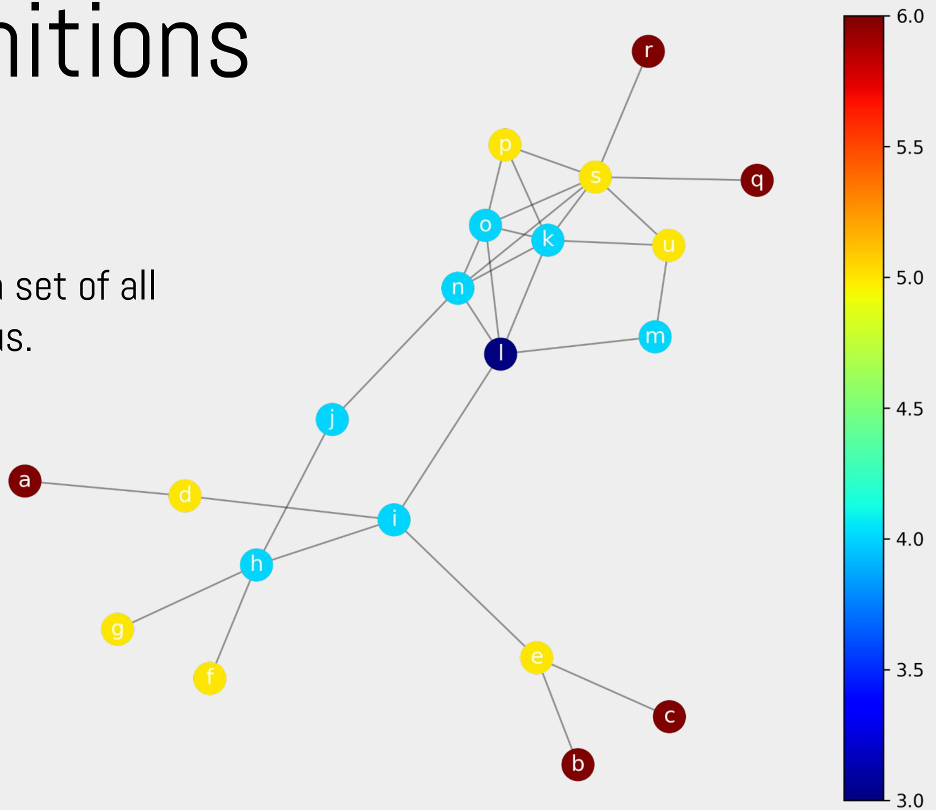
NetworkX
Network Analysis in Python

Some basic definitions

Radius & Center

The radius of a network is the minimum eccentricity. The center of a network is a set of all nodes whose eccentricity equal the radius.

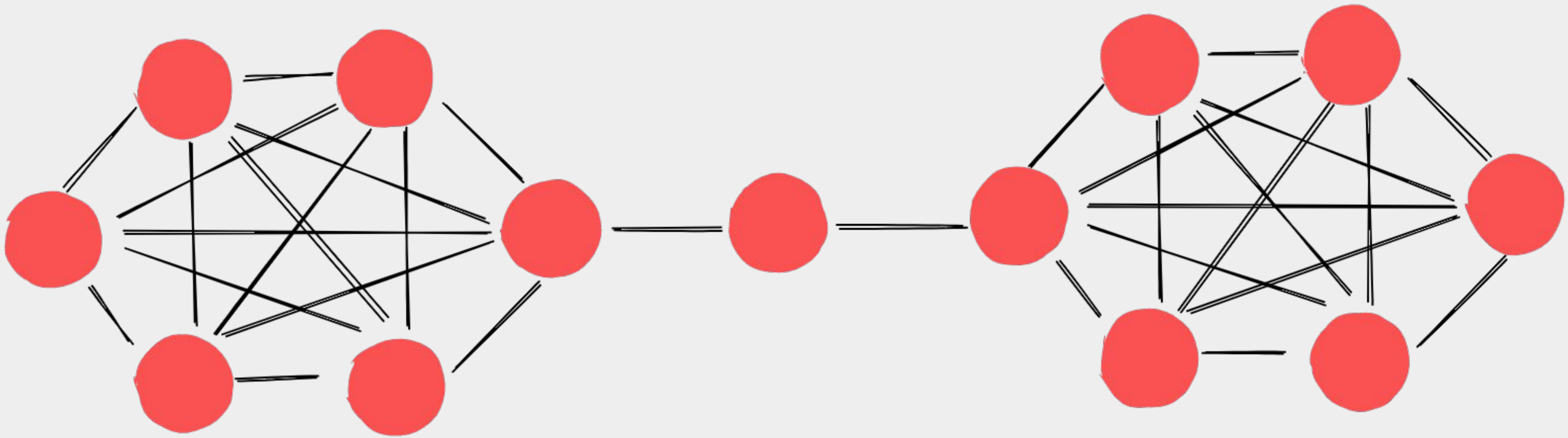
```
nx.eccentricity(g)
{'a': 6, 'b': 6, 'c': 6, 'd': 5,
 'e': 5, 'f': 5, 'g': 5, 'h': 4,
 'i': 4, 'j': 4, 'k': 4, 'l': 3,
 'm': 4, 'n': 4, 'o': 4, 'p': 5,
 'q': 6, 'r': 6, 's': 5, 'u': 5}

nx.radius(g)
3
[k for k,v in nx.eccentricity(g).items()
if v == nx.radius(g)]
['l']
nx.center(g)
['l']
```



NetworkX
Network Analysis in Python

Node Ranking

The more friends a person has, the more important he/she is?
What if there is a person with only few friends, but placed in different communities?

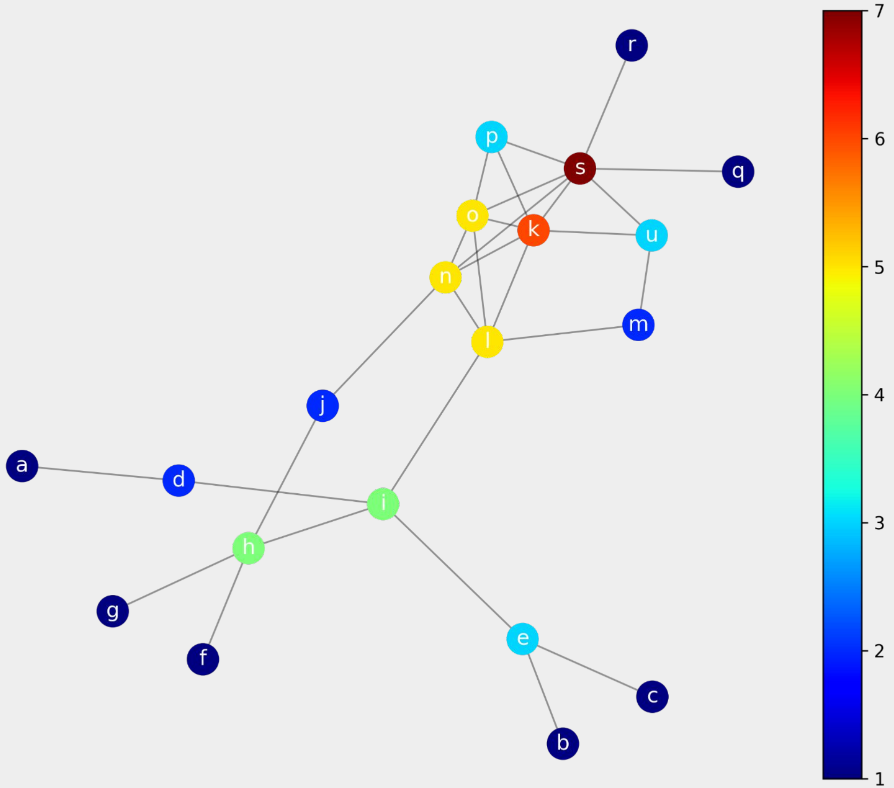


Degree Centrality
Number of connections

Closeness centrality
Average distance to all other vertices

Betweenness Centrality
Position on the shortest path
(intermediação)

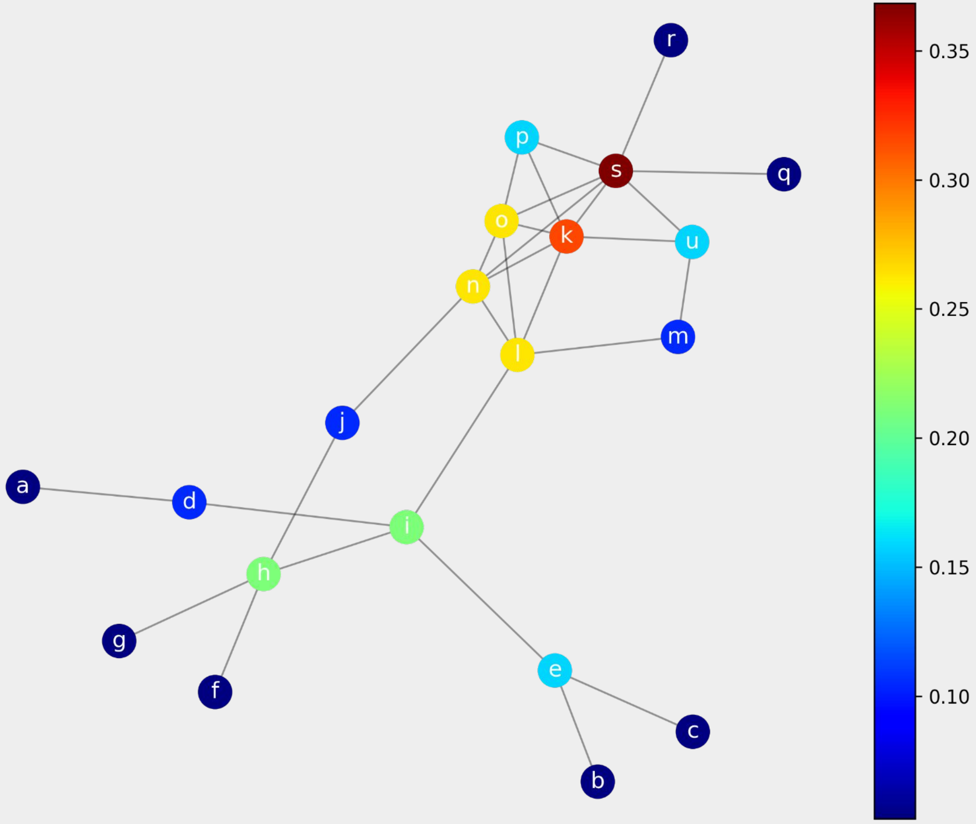
Eigenvector Centrality
Authority score based on the
score of the neighbors



Degree Centrality

Number of connections

```
nx.degree_centrality(g)

{'a': 0.05263157894736842,
 'b': 0.05263157894736842,
 'c': 0.05263157894736842,
 'd': 0.10526315789473684,
 'e': 0.15789473684210525,
 'f': 0.05263157894736842,
 'g': 0.05263157894736842,
 'h': 0.21052631578947367,
 'i': 0.21052631578947367,
 'j': 0.10526315789473684,
 'k': 0.3157894736842105,
 'l': 0.2631578947368421,
 'm': 0.10526315789473684,
 'n': 0.2631578947368421,
 'o': 0.2631578947368421,
 'p': 0.15789473684210525,
 'q': 0.05263157894736842,
 'r': 0.05263157894736842,
 's': 0.3684210526315789,
 'u': 0.15789473684210525}
```



$$degree\_centrality(g) = \frac{\langle k \rangle}{N-1}$$

NetworkX
Network Analysis in Python

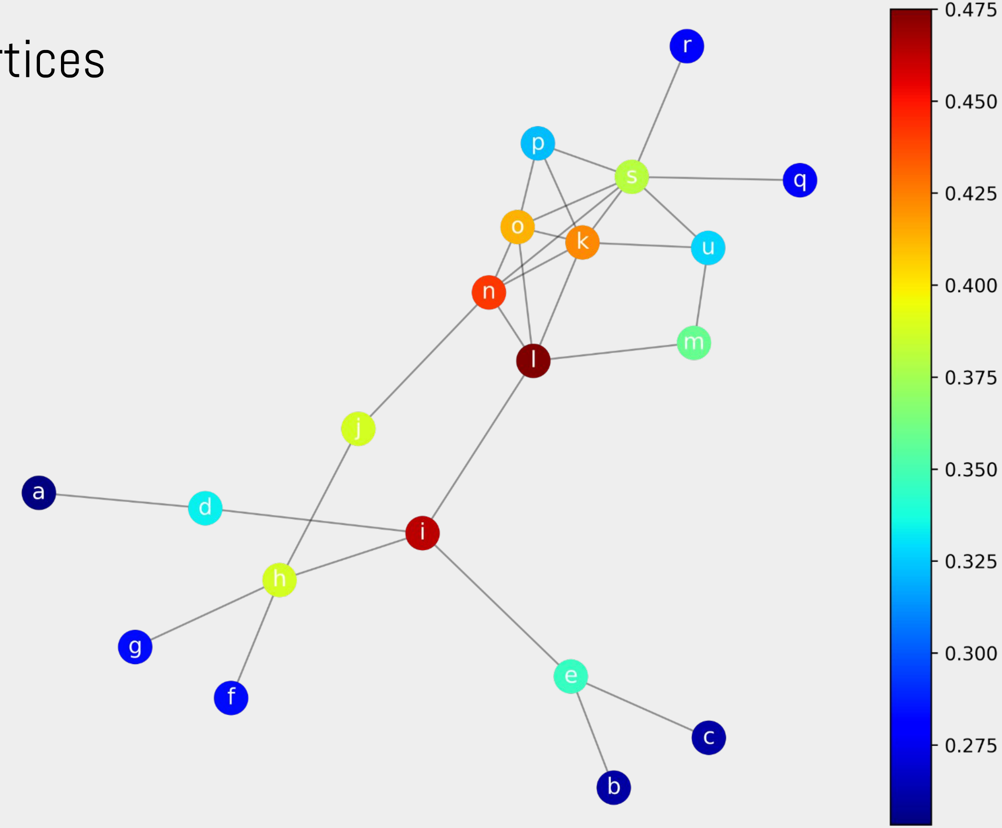
Closeness Centrality

Average distance to all other vertices

Another way to measure the centrality of a node is by determining how "close" it is to the other nodes. This can be done by summing the distances from the node to all others.

$$closeness\_centrality(g, i) = \frac{N-1}{\sum_{j \neq i} l_{ij}}$$

```
nx.closeness_centrality(g)
```

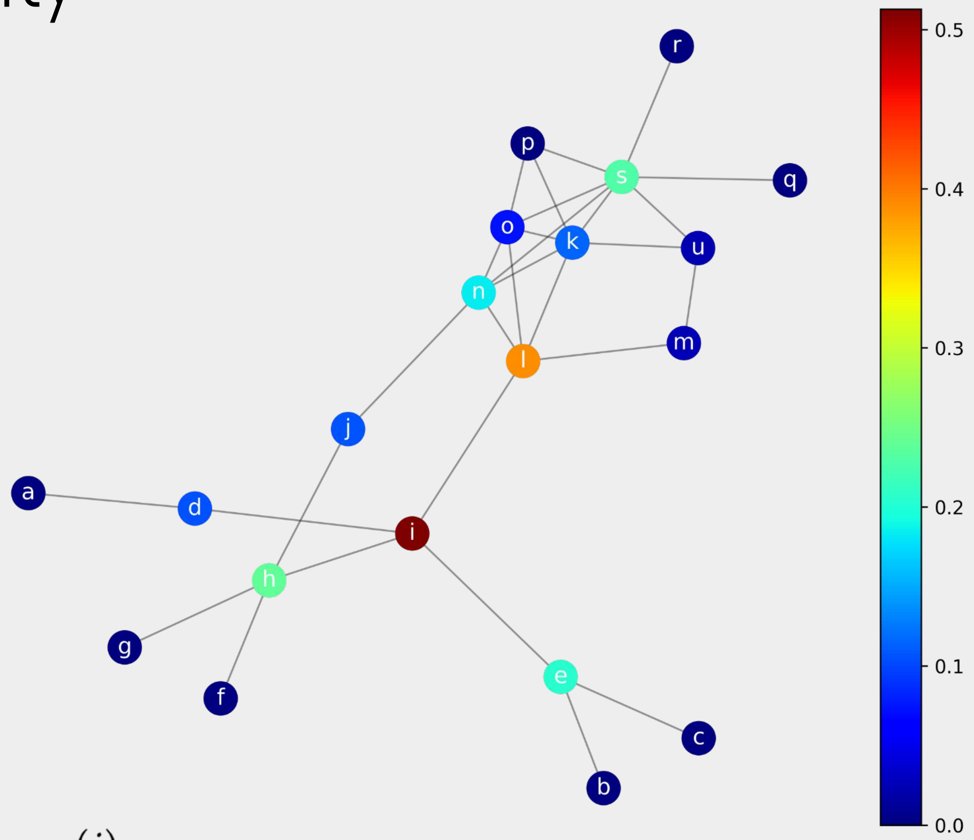


NetworkX
Network Analysis in Python

Betweenness Centrality

Position on the shortest path

Many phenomena taking place in networks are based on diffusion processes. Examples include the transmission of information across a social network, the traffic of goods through a port, and the spread of epidemics in the network of physical contacts between the individuals of a population. This has suggested a third notion of centrality, called **betweenness** : a node is the more central, the more often it is involved in these processes.



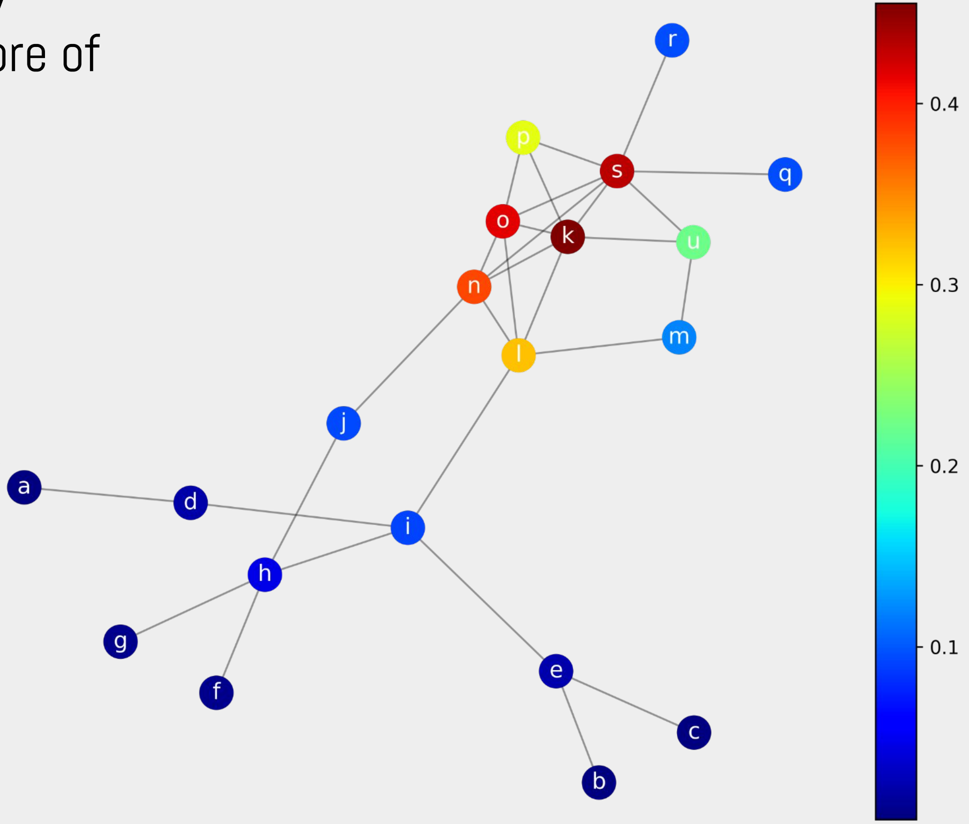
$$betweenness\_centrality(g, i) = \sum_{h \neq j \neq i} \frac{\sigma_{hj}(i)}{\sigma_{hj}}$$

```
nx.betweenness_centrality(g)
```

Eigenvector Centrality

Authority score based on the score of the neighbors

In many circumstances a node's importance in a network is increased by having connections to other nodes that are themselves important. For instance, you might have only one friend in the world, but if that friend is the CR7 then you yourself may be
an important person. Thus centrality is not only about how many people you know but also who you know.

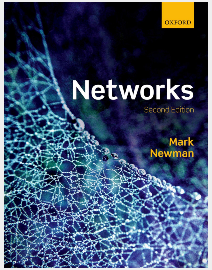



```
nx.eigenvector_centrality(g)
```

NetworkX
Network Analysis in Python

Eigenvector Centrality

Authority score based on the score of the neighbors



Chapter 7

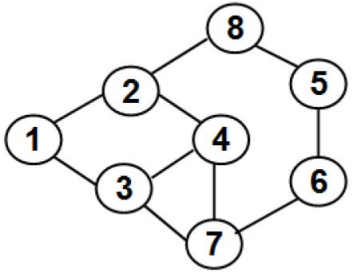
Consider an undirected network of n nodes. The eigenvector centrality x_i of node i is defined to be proportional to the sum of the centralities of i's neighbors, so that:

$$x_i = \kappa^{-1} \sum_{j=1}^{n} A_{ij} x_j$$

$$Ax = \kappa x$$

where x is the vector with elements equal to the centrality scores x_i. In other words, x is an eigenvector of the adjacency matrix. κ is the eigenvalue of the adjacency matrix.

```python
eigenvector_centrality(G, max_iter=100, tol=1.0e-6, nstart=None,weight='weight')
```



$$\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6&7&8\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix} & \begin{bmatrix}0&1&1&0&0&0&0&0\\1&0&0&1&0&0&0&1\\1&0&0&1&0&0&1&0\\0&1&1&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&1&1&0&1&0&0\\0&1&0&0&1&0&0&0\end{bmatrix}\end{array} \times \begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}2\\3\\3\\3\\2\\2\\3\\2\end{bmatrix} \equiv \begin{bmatrix}0.277\\0.416\\0.416\\0.416\\0.277\\0.277\\0.416\\0.277\end{bmatrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix}$$

Iteration 1 Normalized Value = 7.21

$$\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6&7&8\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix} & \begin{bmatrix}0&1&1&0&0&0&0&0\\1&0&0&1&0&0&0&1\\1&0&0&1&0&0&1&0\\0&1&1&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&1&1&0&1&0&0\\0&1&0&0&1&0&0&0\end{bmatrix}\end{array} \times \begin{bmatrix}0.277\\0.416\\0.416\\0.416\\0.277\\0.277\\0.416\\0.277\end{bmatrix} = \begin{bmatrix}0.832\\0.971\\1.109\\1.248\\0.555\\0.693\\1.109\\0.693\end{bmatrix} \equiv \begin{bmatrix}0.316\\0.369\\0.422\\0.474\\0.211\\0.264\\0.422\\0.264\end{bmatrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix}$$

Iteration 2 Normalized Value = 2.63

$$\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6&7&8\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix} & \begin{bmatrix}0&1&1&0&0&0&0&0\\1&0&0&1&0&0&0&1\\1&0&0&1&0&0&1&0\\0&1&1&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&1&1&0&1&0&0\\0&1&0&0&1&0&0&0\end{bmatrix}\end{array} \times \begin{bmatrix}0.316\\0.369\\0.422\\0.474\\0.211\\0.264\\0.422\\0.264\end{bmatrix} = \begin{bmatrix}0.791\\1.054\\1.212\\1.212\\0.527\\0.632\\1.159\\0.579\end{bmatrix} \equiv \begin{bmatrix}0.298\\0.397\\0.457\\0.457\\0.198\\0.238\\0.437\\0.219\end{bmatrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix}$$

Iteration 3 Normalized Value = 2.65

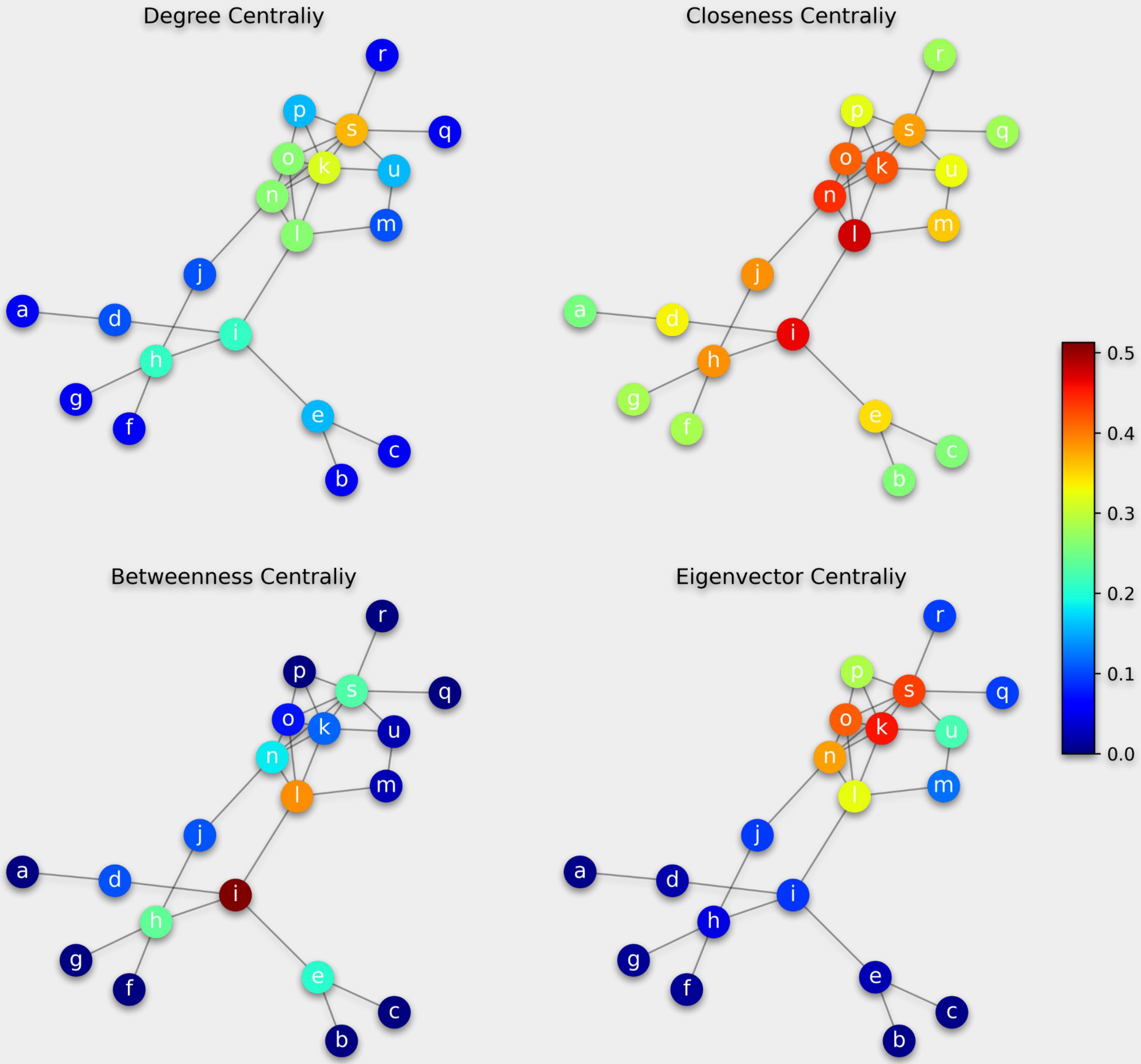
$$\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6&7&8\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix} & \begin{bmatrix}0&1&1&0&0&0&0&0\\1&0&0&1&0&0&0&1\\1&0&0&1&0&0&1&0\\0&1&1&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&1&1&0&1&0&0\\0&1&0&0&1&0&0&0\end{bmatrix}\end{array} \times \begin{bmatrix}0.298\\0.397\\0.457\\0.457\\0.198\\0.238\\0.437\\0.219\end{bmatrix} = \begin{bmatrix}0.855\\0.974\\1.192\\1.292\\0.457\\0.636\\1.153\\0.596\end{bmatrix} \equiv \begin{bmatrix}0.321\\0.366\\0.449\\0.486\\0.172\\0.239\\0.434\\0.224\end{bmatrix}\begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix}$$

Iteration 4 Normalized Value = 2.66

$$\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6&7&8\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\\7\\8\end{matrix} & \begin{bmatrix}0&1&1&0&0&0&0&0\\1&0&0&1&0&0&0&1\\1&0&0&1&0&0&1&0\\0&1&1&0&0&0&1&0\\0&0&0&0&0&1&0&1\\0&0&0&0&1&0&1&0\\0&0&1&1&0&1&0&0\\0&1&0&0&1&0&0&0\end{bmatrix}\end{array} \times \begin{bmatrix}0.321\\0.366\\0.449\\0.486\\0.172\\0.239\\0.434\\0.224\end{bmatrix} = \begin{bmatrix}0.815\\1.032\\1.241\\1.248\\0.434\\0.606\\1.174\\0.538\end{bmatrix} \equiv \begin{bmatrix}0.306\\0.388\\0.467\\0.469\\0.174\\0.228\\0.441\\0.202\end{bmatrix}\begin{matrix}\textbf{Vertex ID}\\1\\2\\3\\4\\5\\6\\7\\8\end{matrix}$$

Iteration 5 Normalized Value = 2.66 **Eigenvector Centrality**

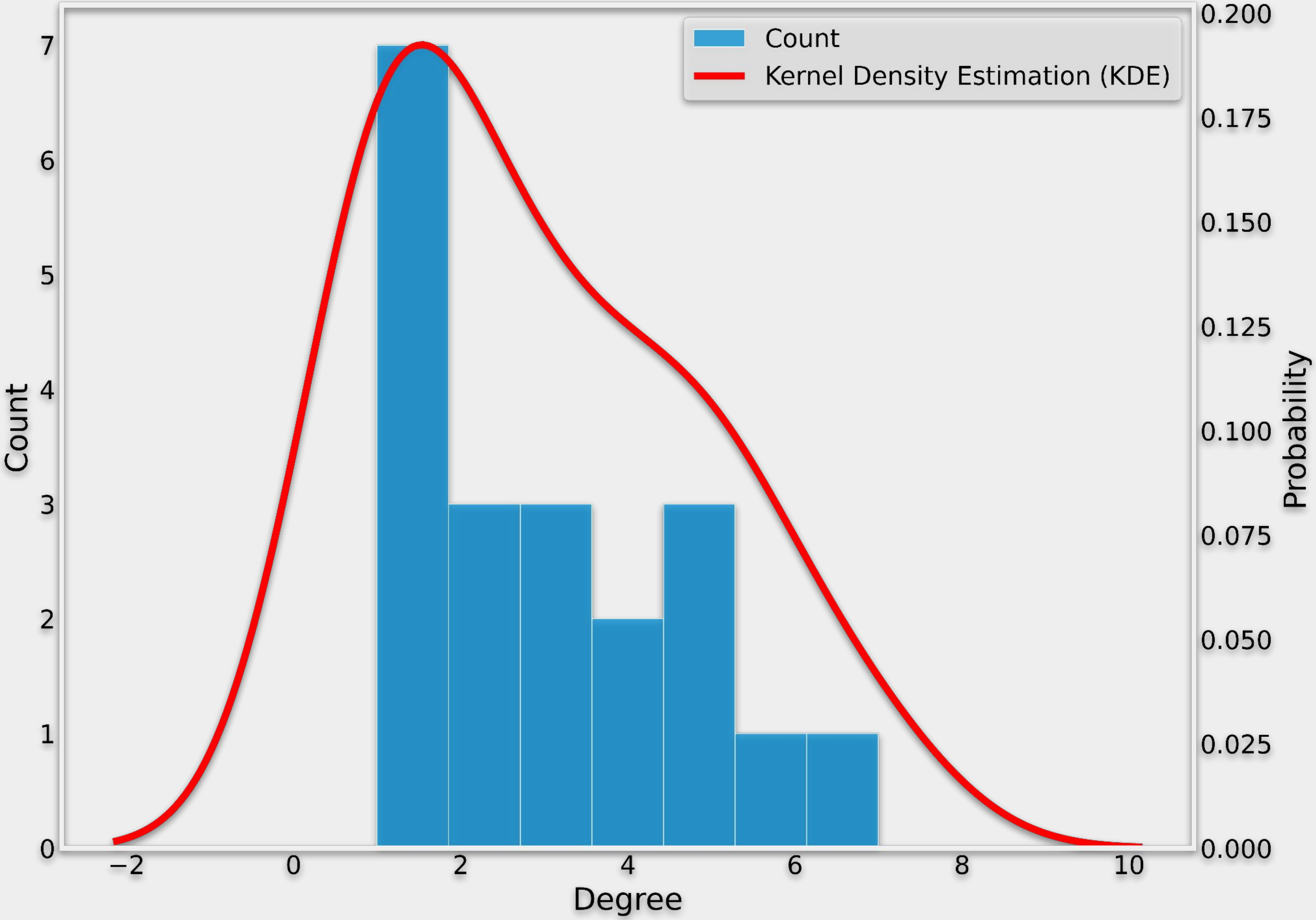
Meghanathan, Natarajan. Use of Eigenvector centrality to detect graph isomorphism, 2015. https://arxiv.org/pdf/1511.06620.pdf

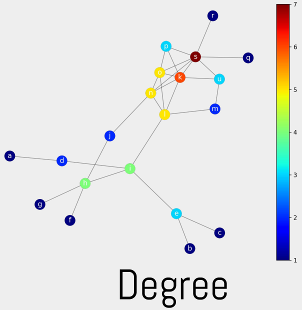


NetworkX
Network Analysis in Python

Centrality Distributions

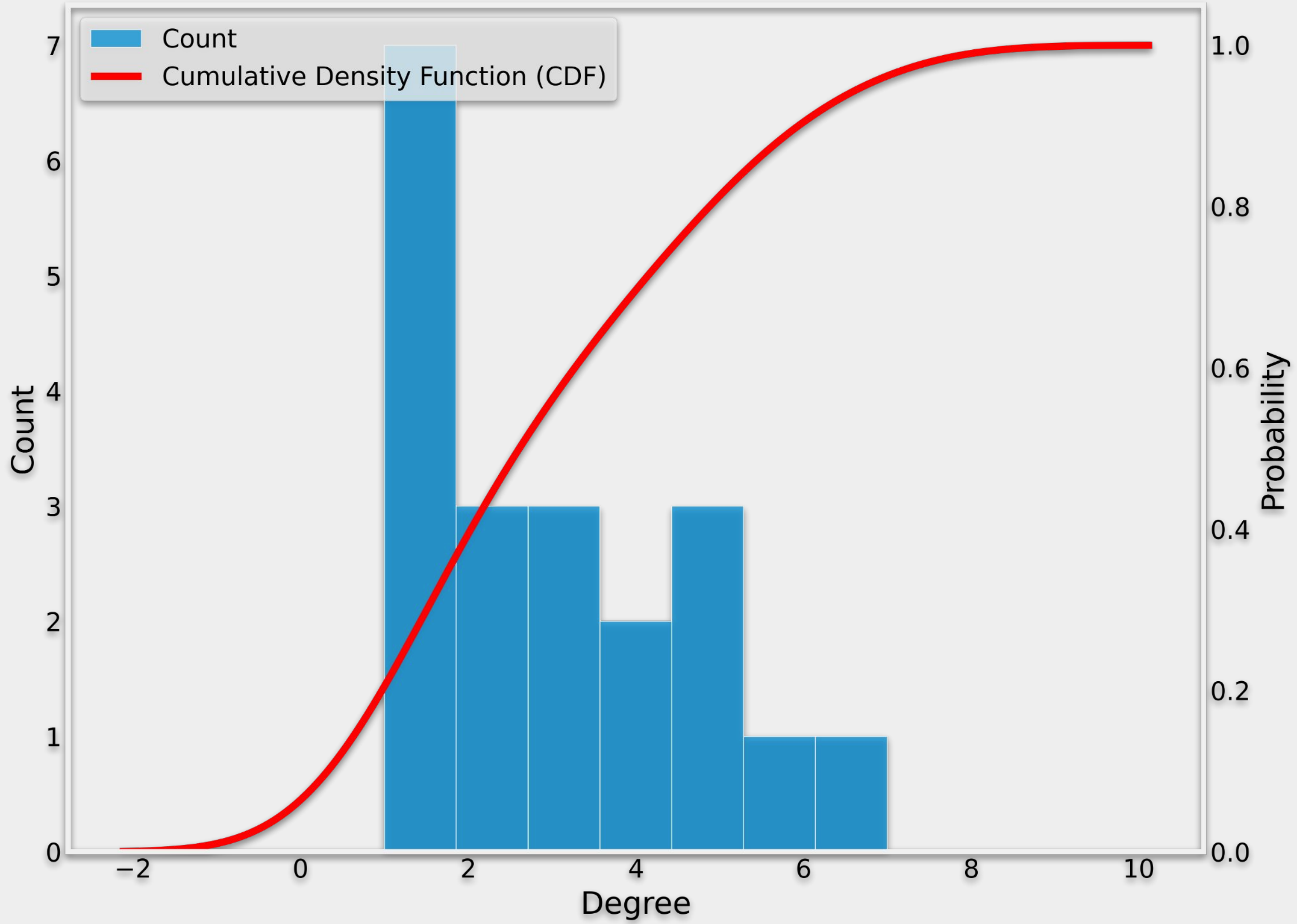
To better understand how centrality is distributed among the many nodes in large networks, we need to take a ***statistical approach***

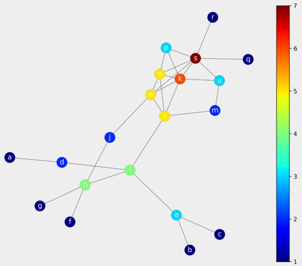




seaborn

NetworkX
Network Analysis in Python





Degree

seaborn

NetworkX
Network Analysis in Python

https://arxiv.org/pdf/1006.1260.pdf

arXiv:1006.1260v3 [physics.soc-ph] 5 Jan 2011

What's in a crowd?
Analysis of face-to-face behavioral networks

Lorenzo Isella, Ciro Cattuto, and Wouter Van den Broeck
Complex Networks and Systems Group, Institute for Scientific Interchange (ISI) Foundation, Turin, Italy

Juliette Stehlé and Alain Barrat
Centre de Physique Théorique, CNRS UMR 6207, Marseille, France

Jean-François Pinton
Laboratoire de Physique de l'ENS Lyon, CNRS UMR 5672, Lyon, France

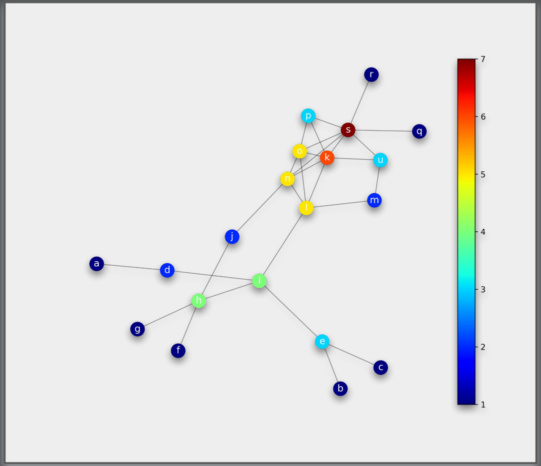
The availability of new data sources on human mobility is opening new avenues for investigating the interplay of social networks, human mobility and dynamical processes such as epidemic spreading. Here we analyze data on the time-resolved face-to-face proximity of individuals in large-scale real-world scenarios. We compare two settings with very different properties, a scientific conference and a long-running museum exhibition. We track the behavioral networks of face-to-face proximity, and characterize them from both a static and a dynamic point of view, exposing differences and similarities. We use our data to investigate the dynamics of a susceptible-infected model for epidemic spreading that unfolds on the dynamical networks of human proximity. The spreading patterns are markedly different for the conference and the museum case, and they are strongly impacted by the causal structure of the network data. A deeper study of the spreading paths shows that the mere knowledge of static aggregated networks would lead to erroneous conclusions about the transmission paths on the dynamical networks.

I. INTRODUCTION

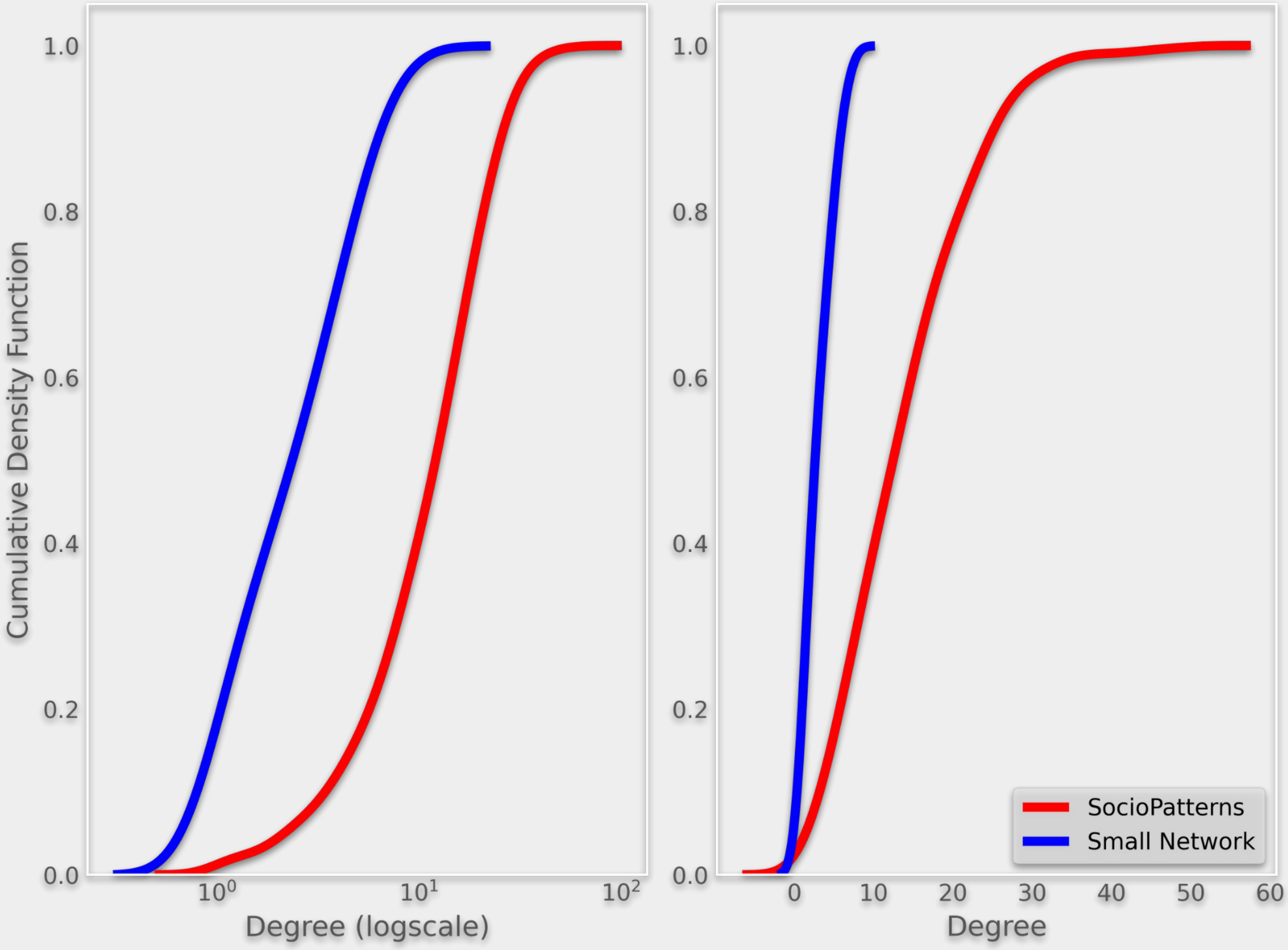
Access to large data sets on human activities and interactions has long been limited by the difficulty and cost of gathering such information. Recently, the ever increasing availability of digital traces of human actions is widely enabling the representation and the analysis of massive amounts of information on human behavior. The representation of this information in terms of complex networks [1–8] has led to many research efforts because of the naturally interlinked nature of these new data sources.

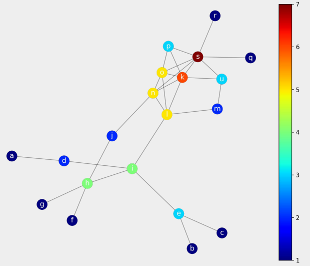
Tracing human behavior in a variety of contexts has become possible at very different spatial and temporal scales: from mobility of individuals inside a city [9] and between cities [10], to mobility and transportation in an entire country [11], all the way to planetary-scale travel [12, 13]. Mobile devices such as cell phones make it possible to investigate mobility patterns and their predictability [14, 15]. On-line interactions occurring between individuals can be monitored by logging instant messaging or email exchange [16–21]. Recent technological advances further support mining real-world interactions by means of mobile devices and wearable sensors, opening up new avenues for gathering data on human and social interactions. Bluetooth and Wifi technologies give access to proximity patterns [22–26], and even face-to-face presence can be resolved with high spatial and temporal resolution [27–30]. The combination of these technological advances and of heterogeneous data sources allow researchers to gather longitudinal data that have been traditionally scarce in social network analysis [31, 32]. A dynamical perspective on interaction networks paves the way to investigating interesting problems such as the interplay of the network dynamics with dynamical processes taking place on these networks.

In this paper, we capitalize on recent efforts [27–30] that made possible to mine behavioral networks of face-to-face interactions between individuals, in a variety of real-world settings and in a time-resolved fashion. We present an in-depth analysis of the data we collected at two widely different events. The first event was the INFECTIOUS exhibition [33] held at the Science Gallery in Dublin, Ireland, from April 17^{th} to July 17^{th}, 2009. The second event was the ACM Hypertext 2009 conference [34] hosted by the Institute for Scientific Interchange Foundation in Turin, Italy, from June 29^{th} to July 1^{st}, 2009. In the following, we will refer to these events as SG and HT09, respectively. Intuitively, interactions among conference participants differ from interactions among museum visitors, and the concerned individuals have very different goals in both settings. The study of the corresponding networks of proximity and interactions, both static and dynamic, reveals indeed strong differences but also interesting similarities. We take advantage of the availability of time-resolved data to show how dynamical processes that can unfold on the close proximity network — such as the propagation of a piece of information or the spreading of an infectious agent — unfold in very different ways in the investigated settings. In the epidemiological literature, traditionally,

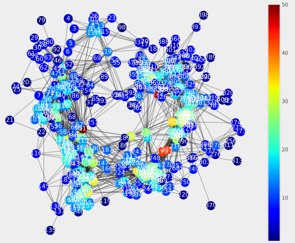


How to compare networks with different scales of network centralities ?

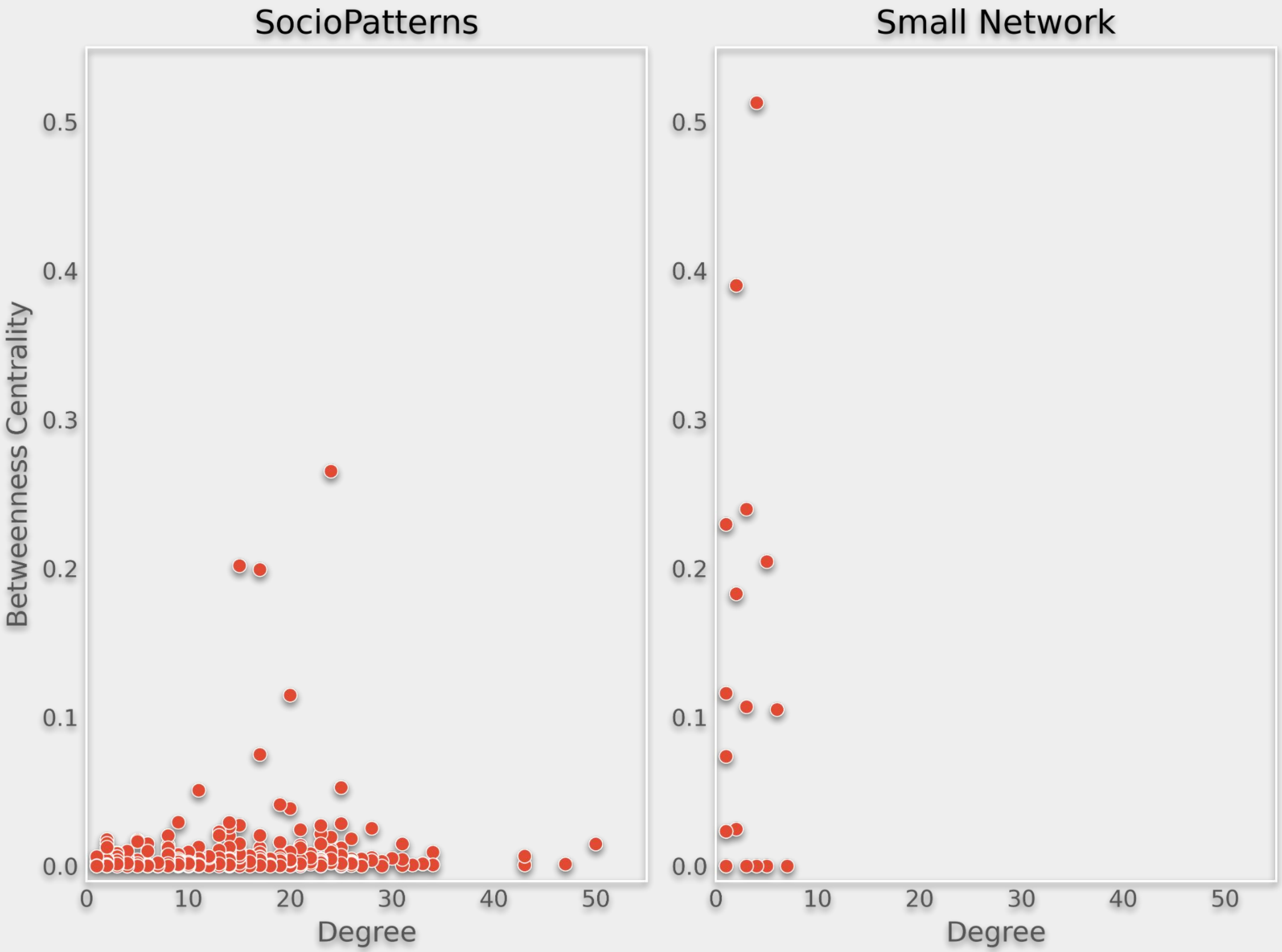


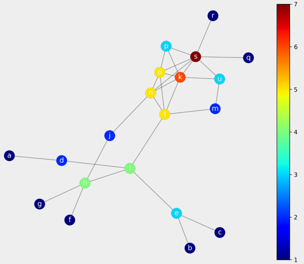


20 vertices, 29 edges

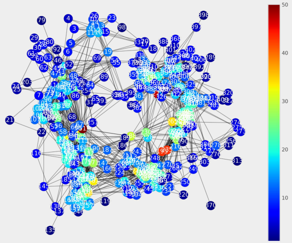


410 vertices, 2765 edges





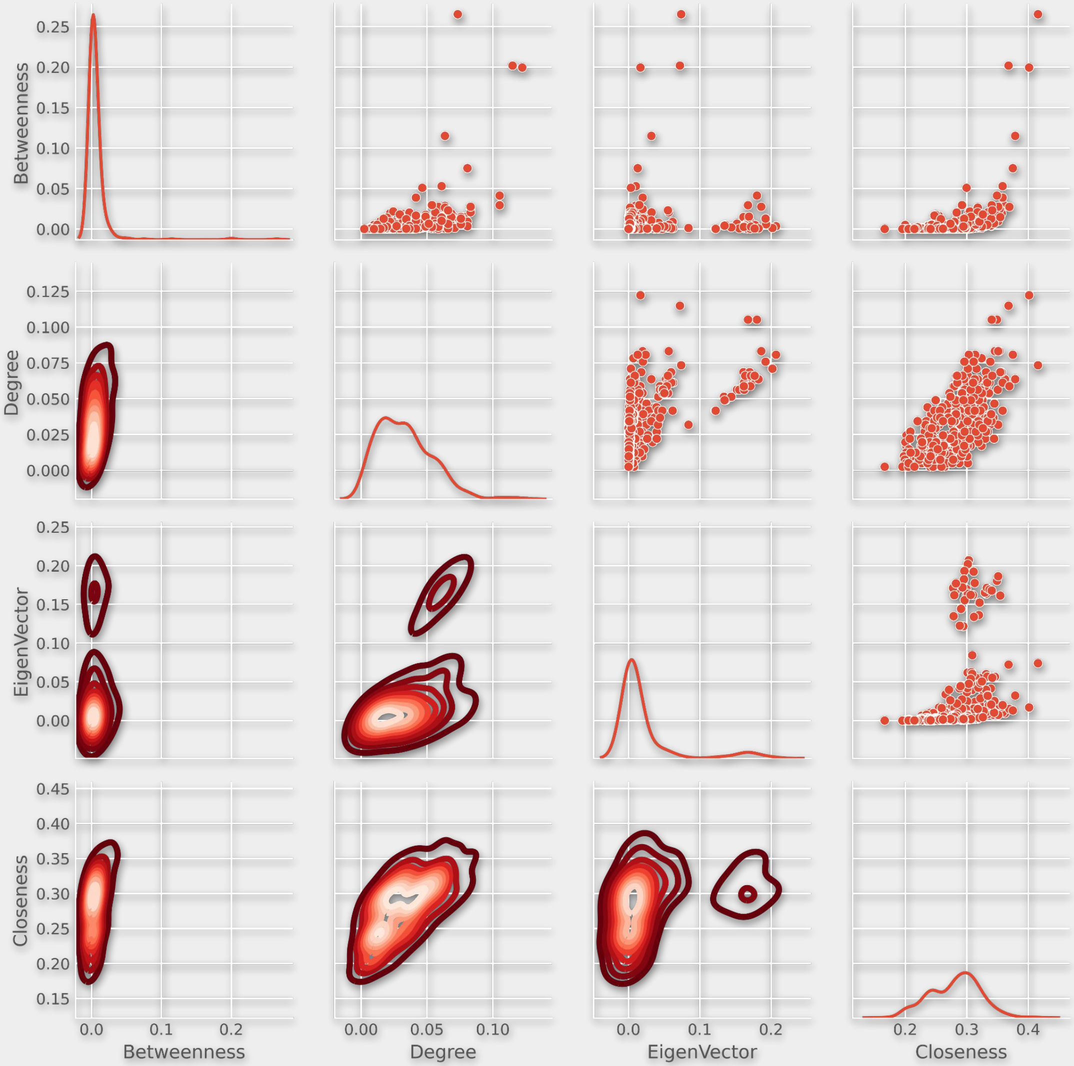
20 vertices, 29 edges



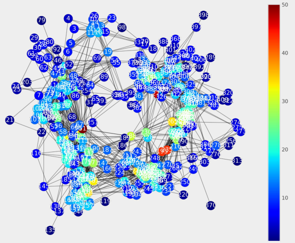
410 vertices, 2765 edges

seaborn

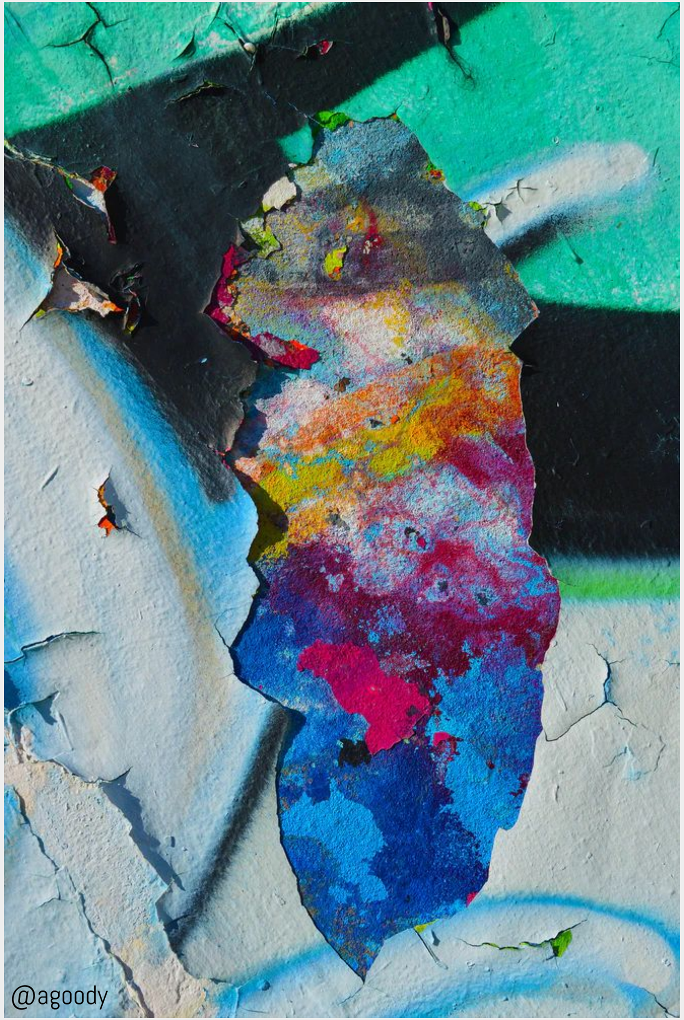
NetworkX
Network Analysis in Python




```python
fig = sns.PairGrid(df)
fig.map_upper(sns.scatterplot)
fig.map_lower(sns.kdeplot, cmap="Reds_r")
fig.map_diag(sns.kdeplot, lw=2, legend=False)
```



410 vertices, 2765 edges



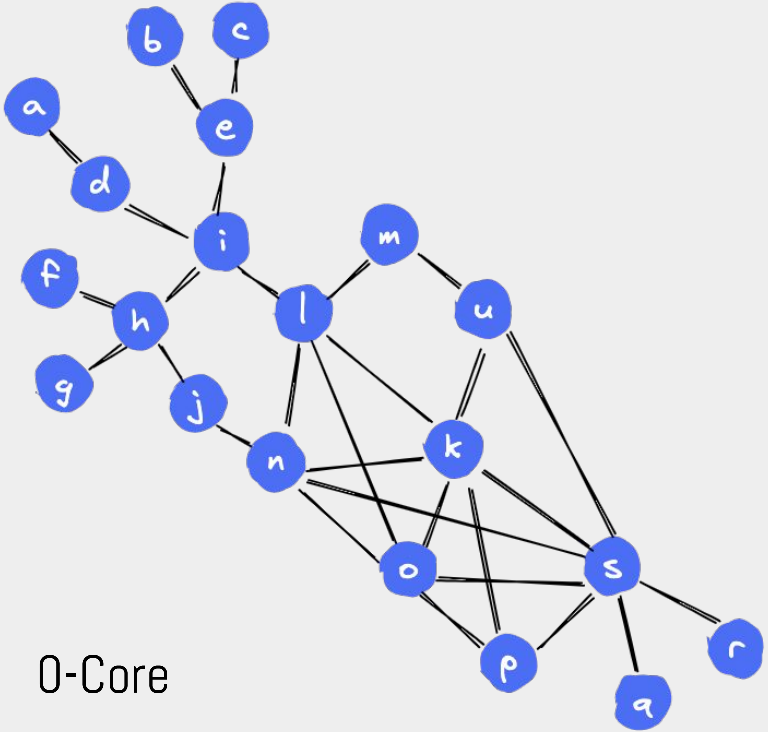
Core Decomposition

K-Core vs K-Shell

When it comes to node centrality, one common term you'll hear thrown around is one of **"core" node**. This is usually a qualitative distinction.

A **k-core** in a network is a subset of its nodes in which all nodes have at least **k connections** to each other. One can easily identify the k-core of a network via the **k-core decomposition algorithm.**

K-Core Decomposition

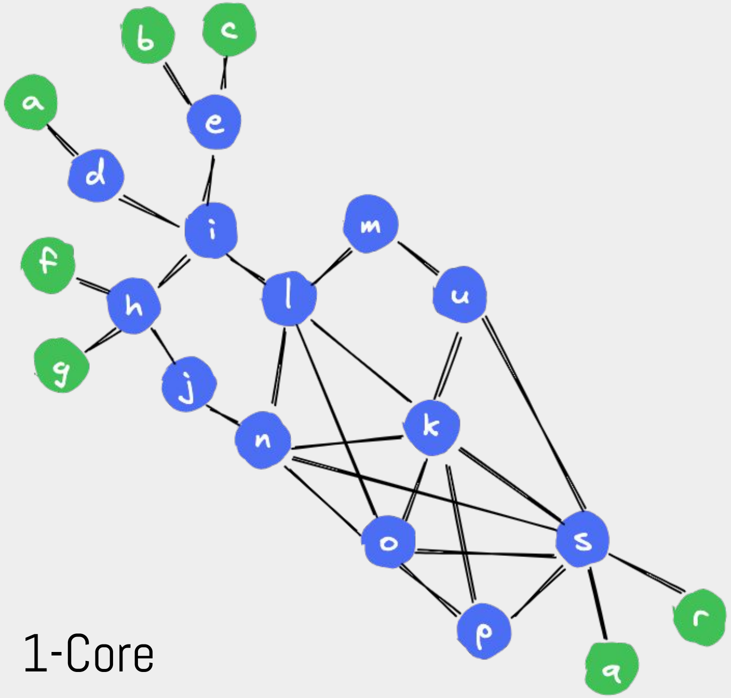


0-Core

```
core = 0
for i in nx.k_core(g,core):
    print(i)


a
d
b
e
c
i
f
h
g
j
l
n
m
k
o
u
s
p
q
r
```

NetworkX
Network Analysis in Python

K-Core Decomposition

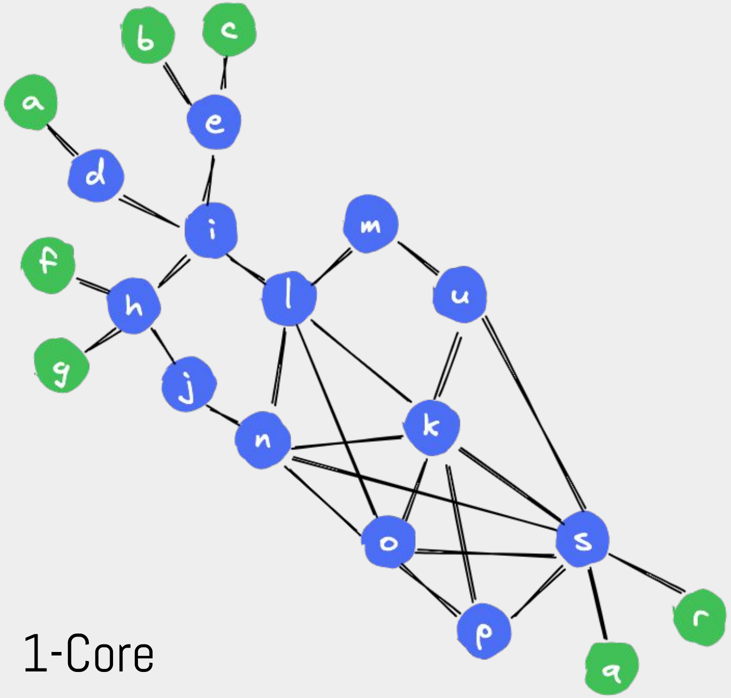


1-Core

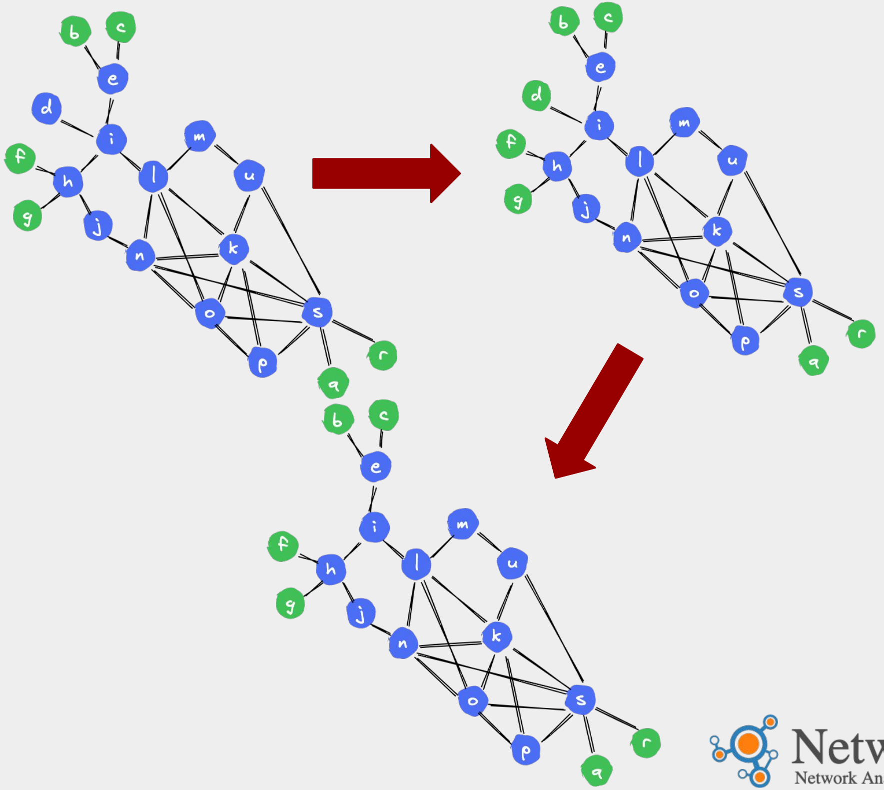
```
core = 1
for i in nx.k_core(g,core):
    print(i)

a
d
b
e
c
i
f
h
g
j
l
n
m
k
o
u
s
p
q
r
```

NetworkX
Network Analysis in Python

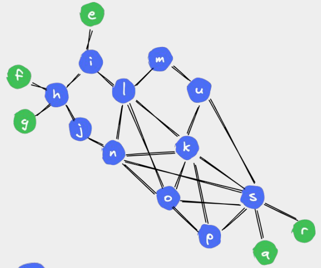
K-Core Decomposition

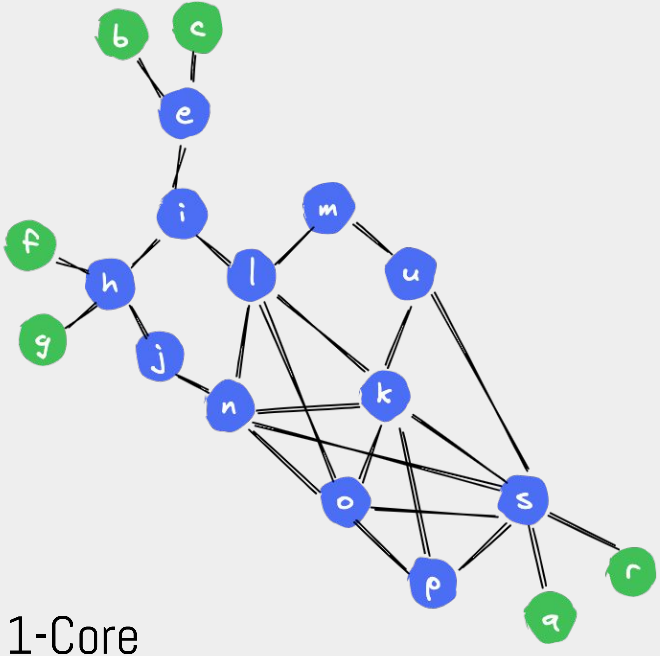


1-Core

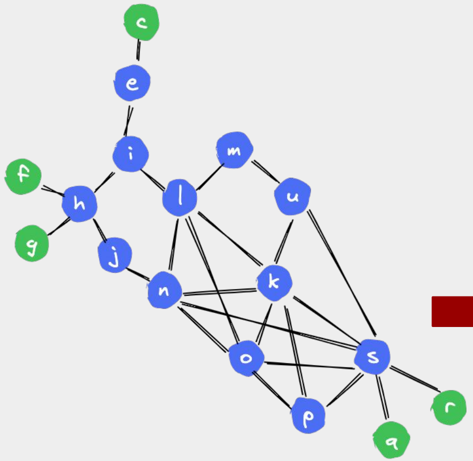


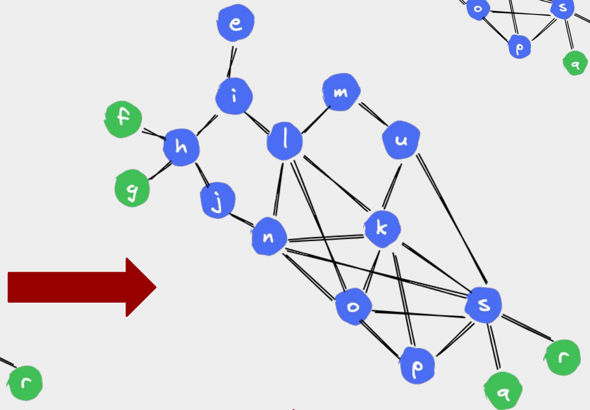
K-Core Decomposition

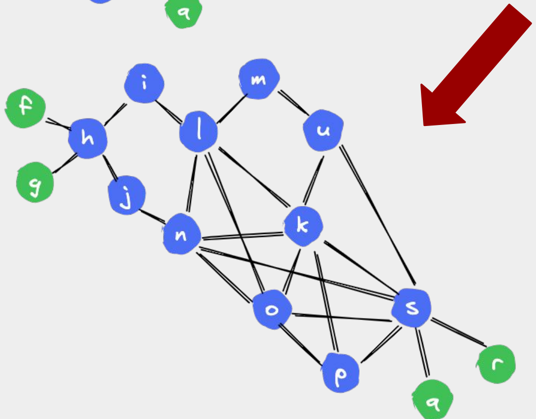




1-Core

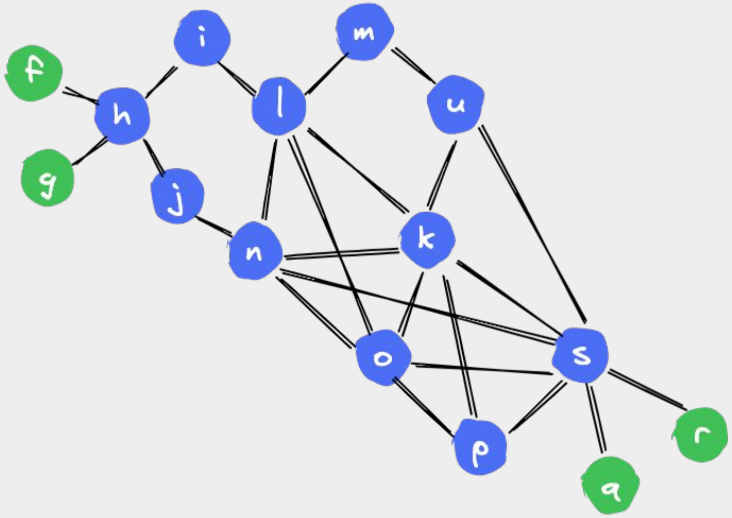




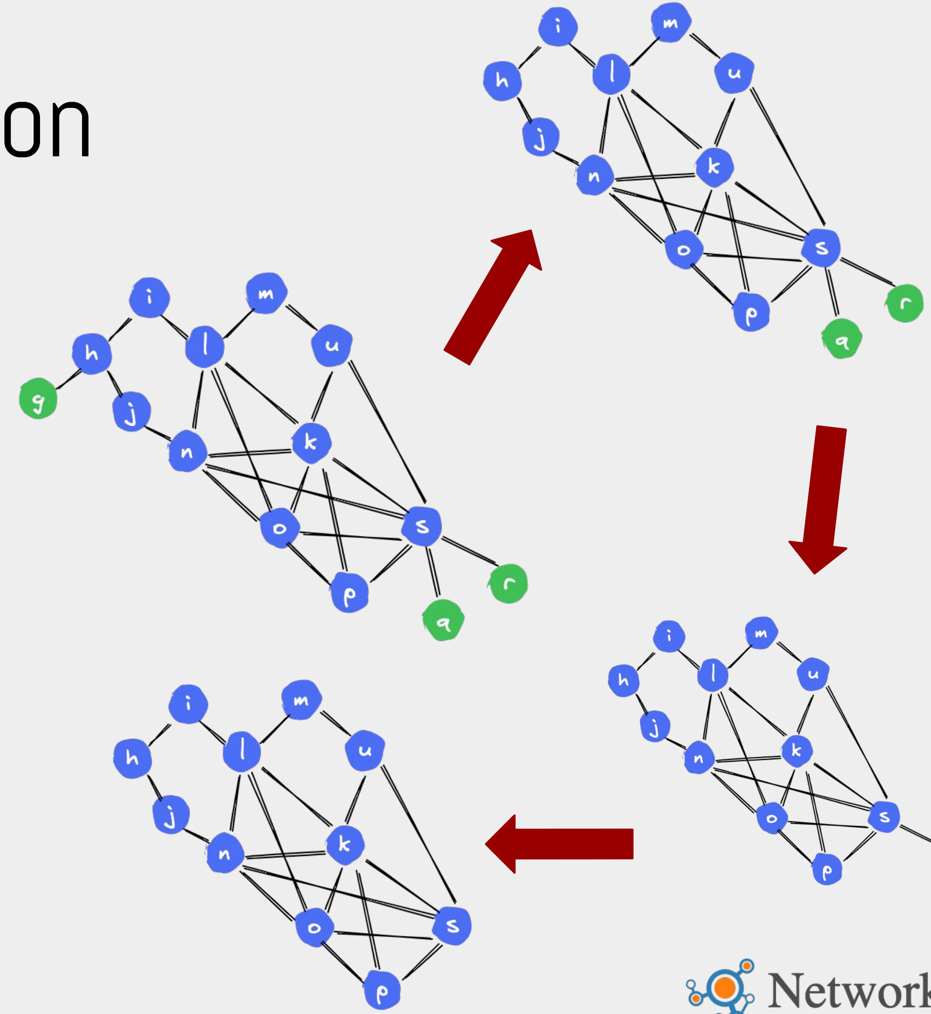


NetworkX
Network Analysis in Python

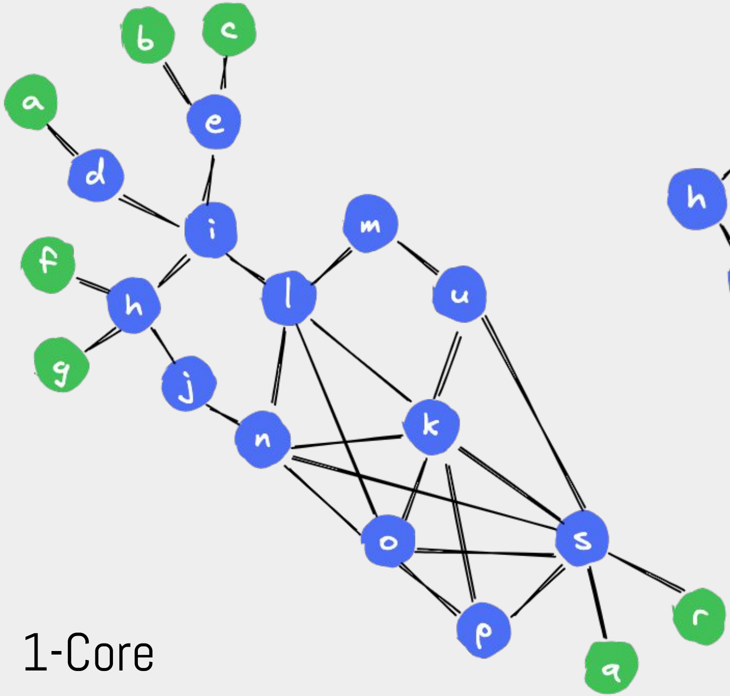
K-Core Decomposition



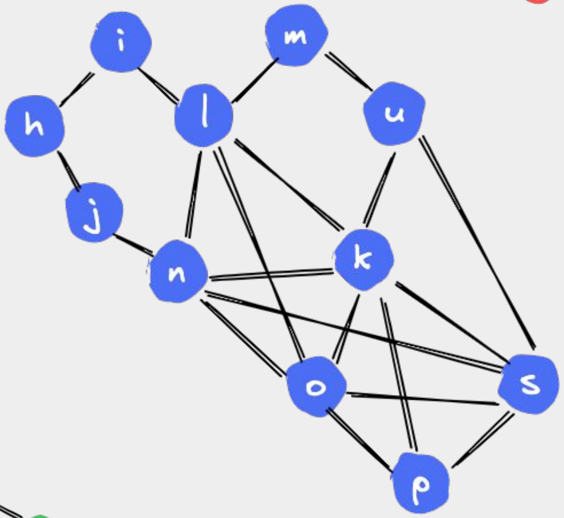
1-Core



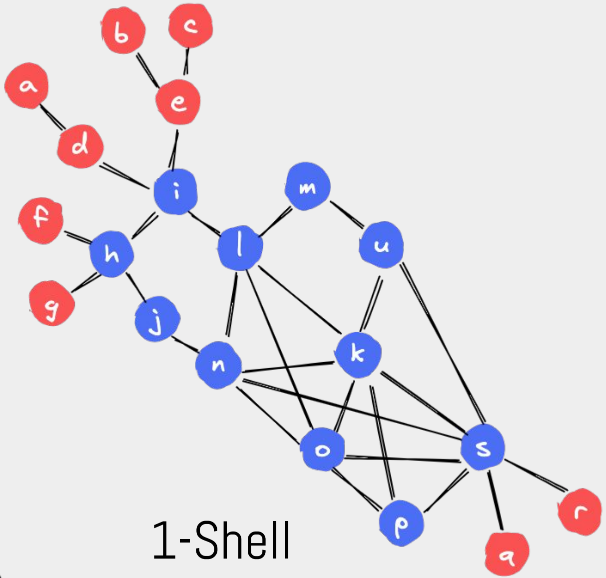
K-Core Decomposition



1-Core



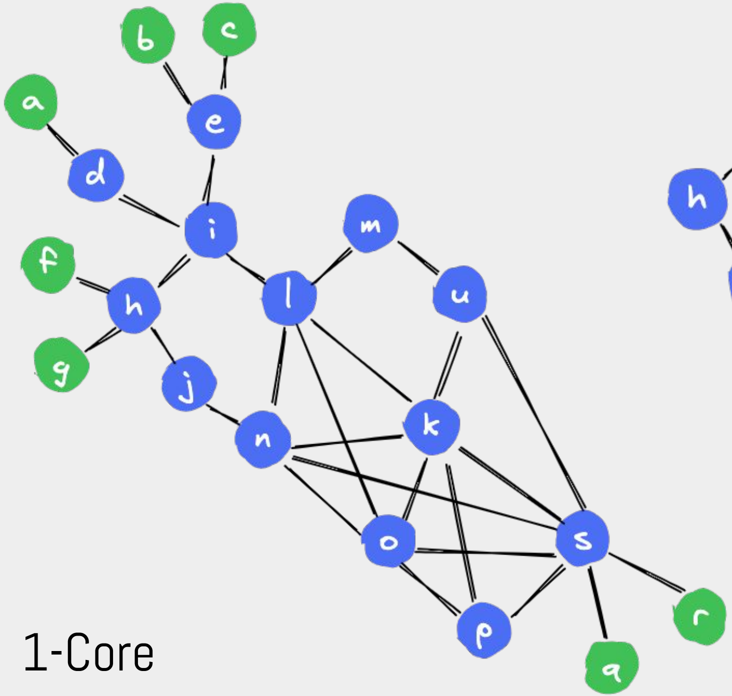
2-Core



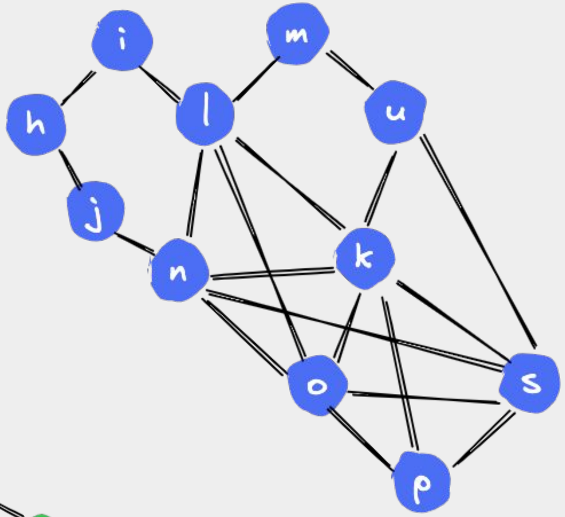
1-Shell

NetworkX
Network Analysis in Python

K-Core Decomposition



1-Core



2-Core

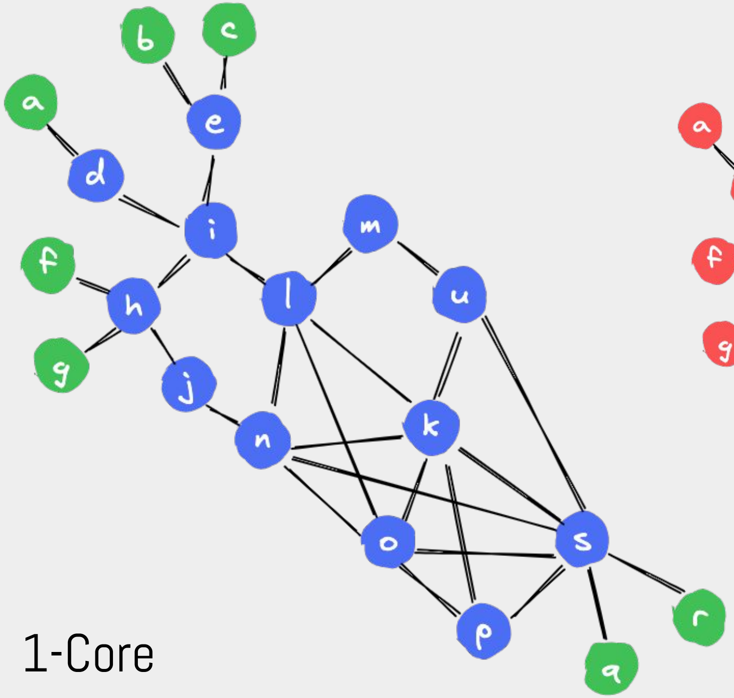
```
core = 2
for i in nx.k_core(g,core):
   print(i)

i
h
j
l
n
m
k
o
u
s
p
```

NetworkX
Network Analysis in Python

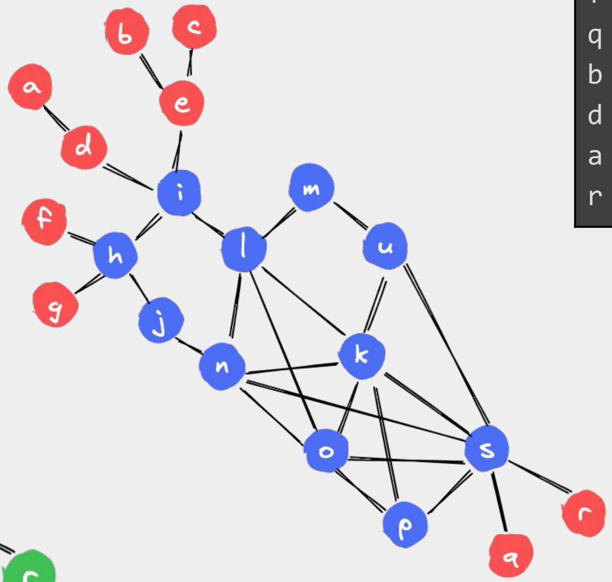
K-Core Decomposition

```
shell = 1
for i in nx.k_shell(g,shell):
 print(i)
```

```
g
e
c
f
q
b
d
a
r
```



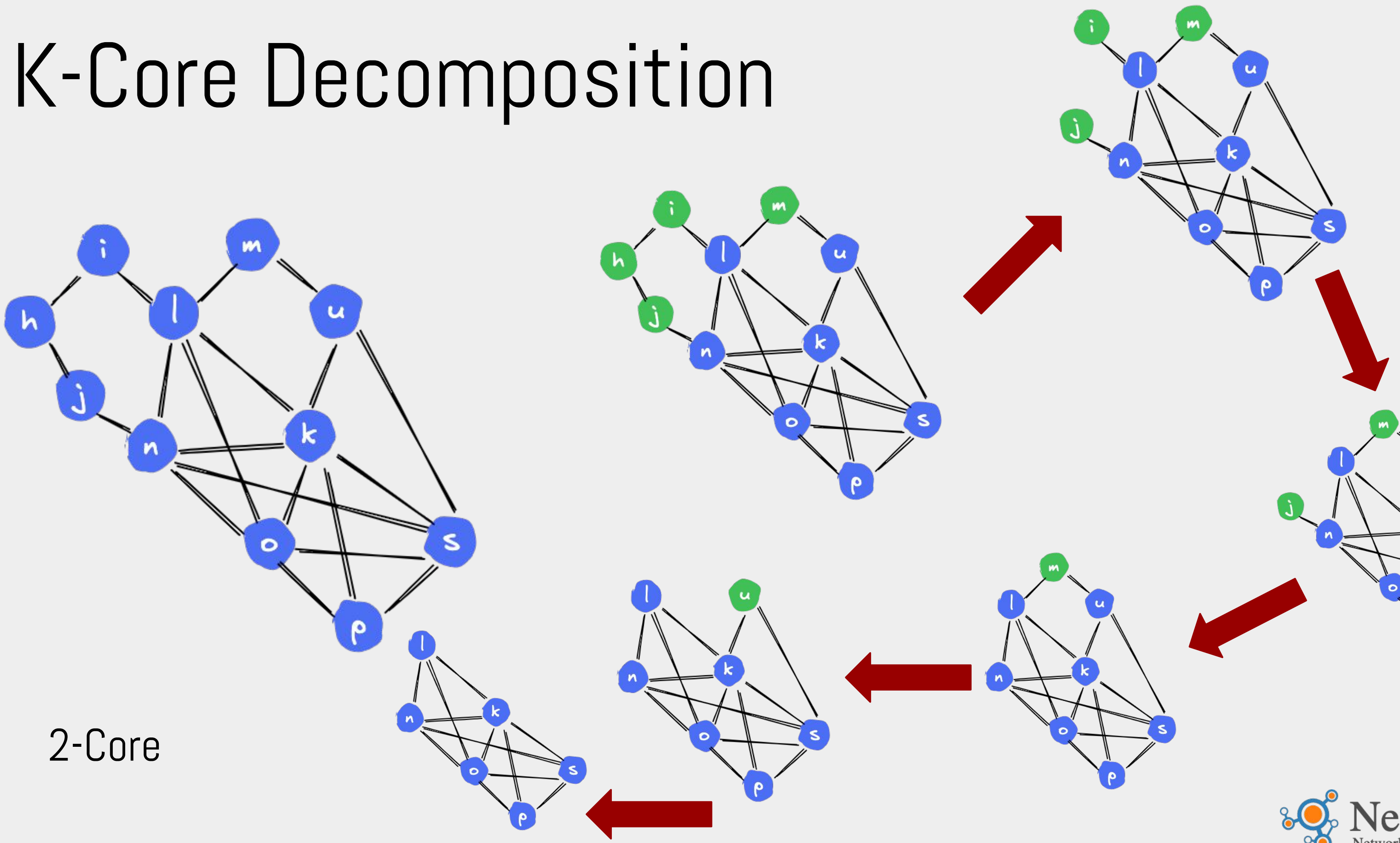
1-Core



1-Shell

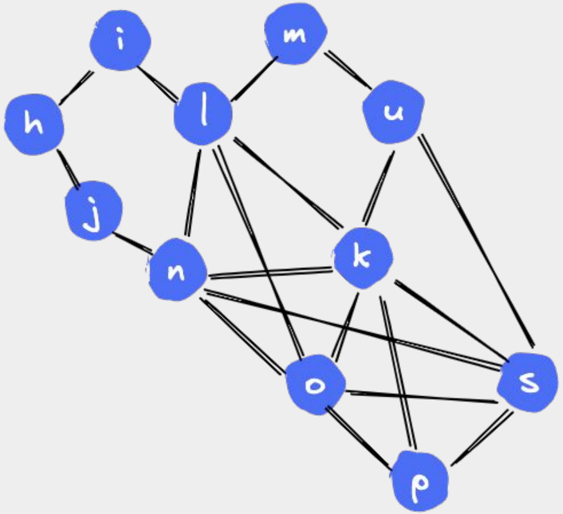
NetworkX
Network Analysis in Python

K-Core Decomposition

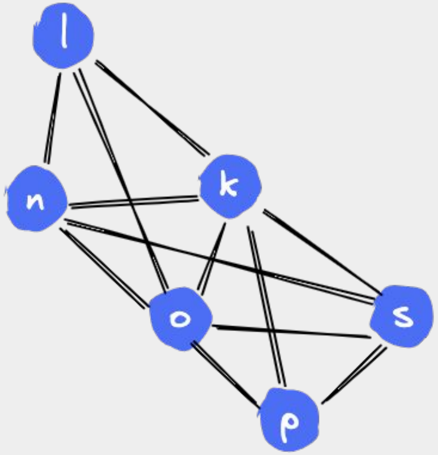


2-Core

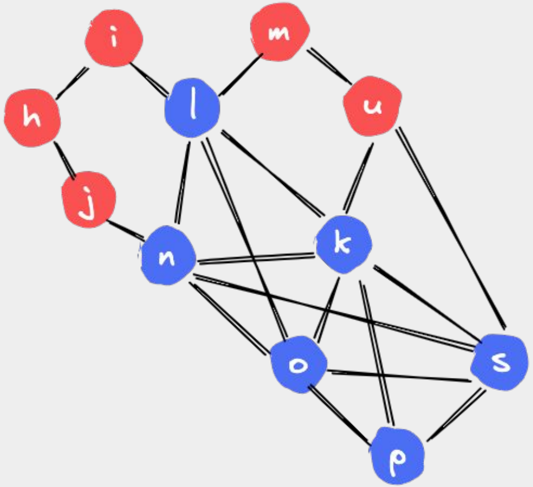
K-Core Decomposition



2-Core



3-Core

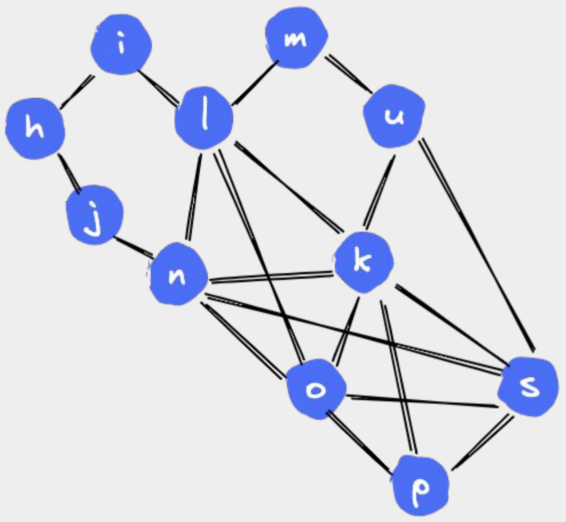


2-Shell

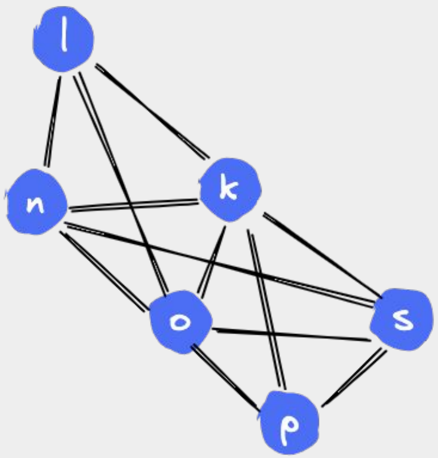
NetworkX
Network Analysis in Python

K-Core Decomposition

```
core = 3
for i in nx.k_core(g,core):
   print(i)


o
l
p
k
n
s
```



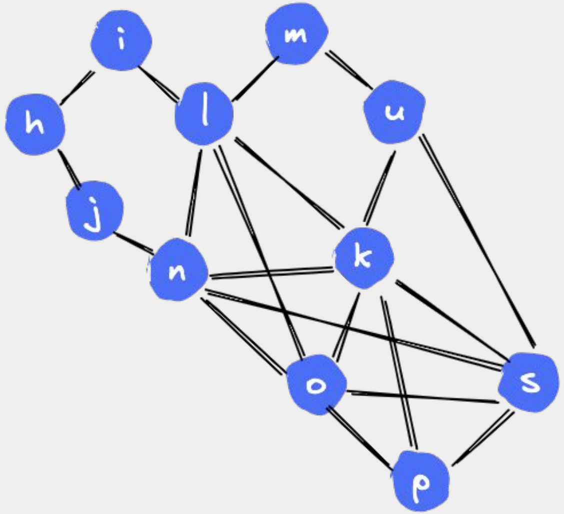
2-Core



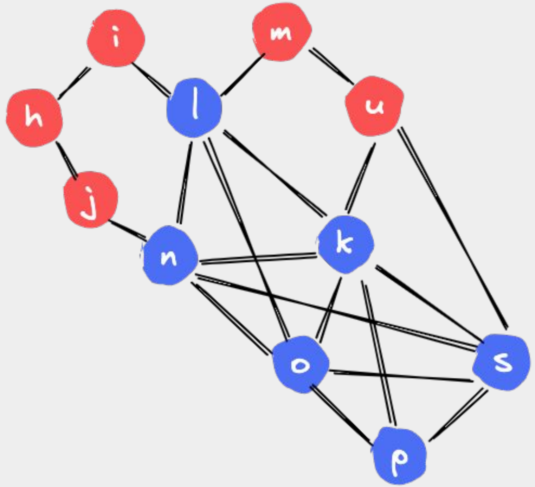
3-Core

K-Core Decomposition

```
shell = 2
for i in nx.k_shell(g,shell):
 print(i)

u
h
m
j
i
```



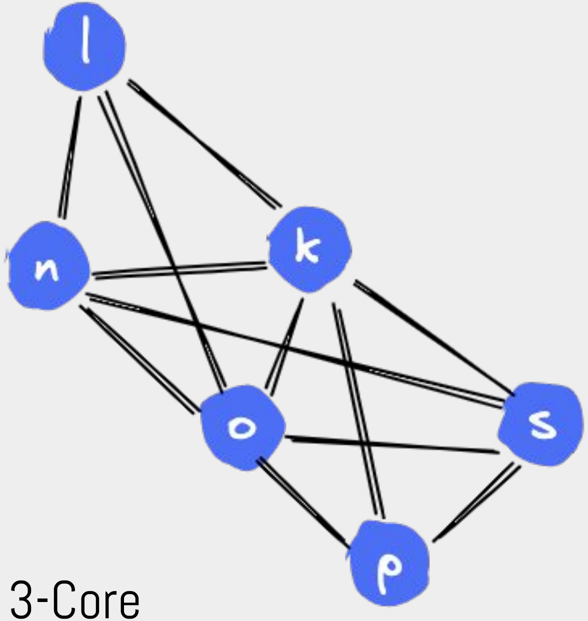
2-Core



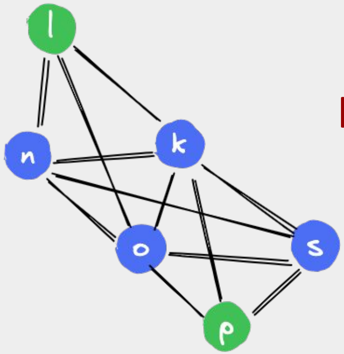
2-Shell

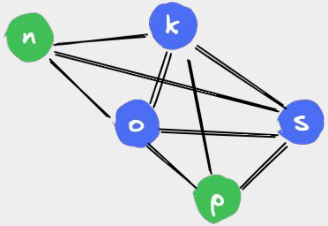
NetworkX
Network Analysis in Python

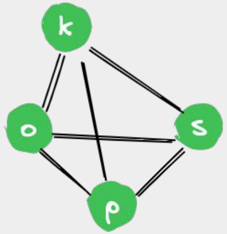
K-Core Decomposition



3-Core







3-Core == 3-Shell

NetworkX
Network Analysis in Python

https://arxiv.org/pdf/1006.1260.pdf

What's in a crowd?
Analysis of face-to-face behavioral networks

Lorenzo Isella, Ciro Cattuto, and Wouter Van den Broeck
Complex Networks and Systems Group, Institute for Scientific Interchange (ISI) Foundation, Turin, Italy

Juliette Stehlé and Alain Barrat
Centre de Physique Théorique, CNRS UMR 6207, Marseille, France

Jean-François Pinton
Laboratoire de Physique de l'ENS Lyon, CNRS UMR 5672, Lyon, France

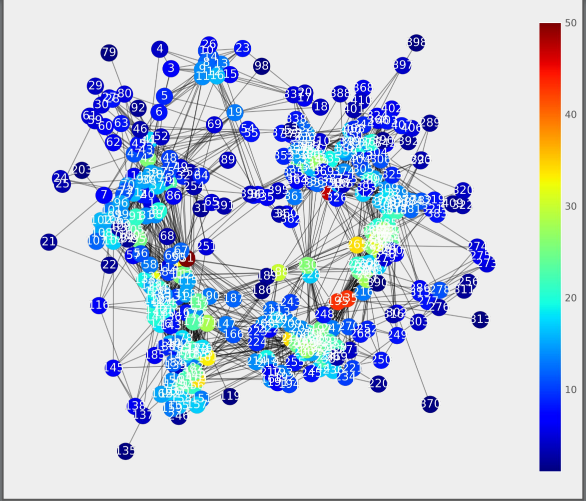
The availability of new data sources on human mobility is opening new avenues for investigating the interplay of social networks, human mobility and dynamical processes such as epidemic spreading. Here we analyze data on the time-resolved face-to-face proximity of individuals in large-scale real-world scenarios. We compare two settings with very different properties, a scientific conference and a long-running museum exhibition. We track the behavioral networks of face-to-face proximity, and characterize them from both a static and a dynamic point of view, exposing differences and similarities. We use our data to investigate the dynamics of a susceptible-infected model for epidemic spreading that unfolds on the dynamical networks of human proximity. The spreading patterns are markedly different for the conference and the museum case, and they are strongly impacted by the causal structure of the network data. A deeper study of the spreading paths shows that the mere knowledge of static aggregated networks would lead to erroneous conclusions about the transmission paths on the dynamical networks.

I. INTRODUCTION

Access to large data sets on human activities and interactions has long been limited by the difficulty and cost of gathering such information. Recently, the ever increasing availability of digital traces of human actions is widely enabling the representation and the analysis of massive amounts of information on human behavior. The representation of this information in terms of complex networks [1–8] has led to many research efforts because of the naturally interlinked nature of these new data sources.

Tracing human behavior in a variety of contexts has become possible at very different spatial and temporal scales: from mobility of individuals inside a city [9] and between cities [10], to mobility and transportation in an entire country [11], all the way to planetary-scale travel [12, 13]. Mobile devices such as cell phones make it possible to investigate mobility patterns and their predictability [14, 15]. On-line interactions occurring between individuals can be monitored by logging instant messaging or email exchange [16–21]. Recent technological advances further support mining real-world interactions by means of mobile devices and wearable sensors, opening up new avenues for gathering data on human and social interactions. Bluetooth and Wifi technologies give access to proximity patterns [22–26], and even face-to-face presence can be resolved with high spatial and temporal resolution [27–30]. The combination of these technological advances and of heterogeneous data sources allow researchers to gather longitudinal data that have been traditionally scarce in social network analysis [31, 32]. A dynamical perspective on interaction networks paves the way to investigating interesting problems such as the interplay of the network dynamics with dynamical processes taking place on these networks.

In this paper, we capitalize on recent efforts [27–30] that made possible to mine behavioral networks of face-to-face interactions between individuals, in a variety of real-world settings and in a time-resolved fashion. We present an in-depth analysis of the data we collected at two widely different events. The first event was the INFECTIOUS exhibition [33] held at the Science Gallery in Dublin, Ireland, from April 17^{th} to July 17^{th}, 2009. The second event was the ACM Hypertext 2009 conference [34] hosted by the Institute for Scientific Interchange Foundation in Turin, Italy, from June 29^{th} to July 1^{st}, 2009. In the following, we will refer to these events as SG and HT09, respectively. Intuitively, interactions among conference participants differ from interactions among museum visitors, and the concerned individuals have very different goals in both settings. The study of the corresponding networks of proximity and interactions, both static and dynamic, reveals indeed strong differences but also interesting similarities. We take advantage of the availability of time-resolved data to show how dynamical processes that can unfold on the close proximity network — such as the propagation of a piece of information or the spreading of an infectious agent — unfold in very different ways in the investigated settings. In the epidemiological literature, traditionally,

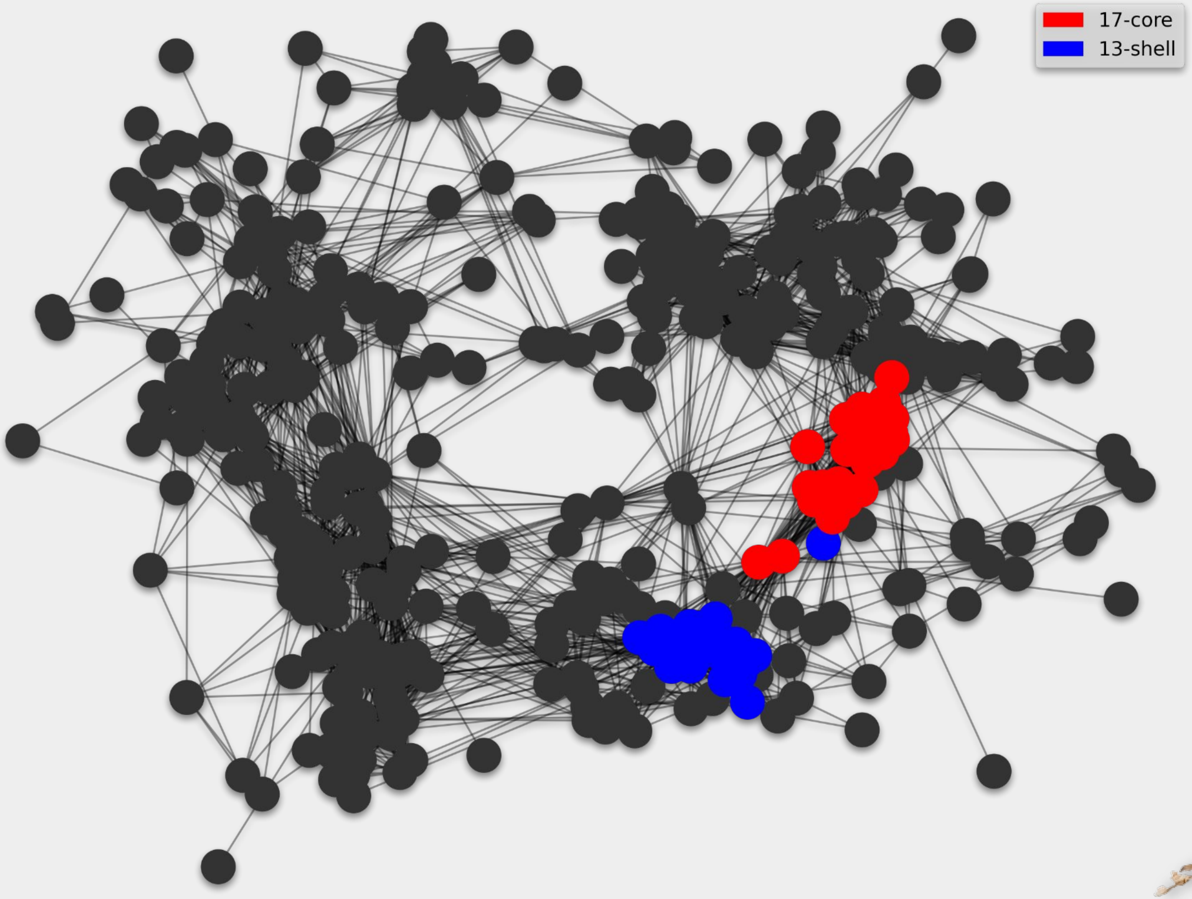


How many k-core does this network have?

```
# how many k-cores does this network have?
set([v for k,v in nx.core_number(g).items()])
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 17}
```

```
# who are in the innermost k-core
list(nx.k_shell(g,17))
[257, 258, 259, 260, 261, 262, 263, 264, 265,
266, 267, 281, 282, 283, 284, 285, 287, 288, 291,
292, 293, 294, 295, 296, 297, 298, 299, 300, 301,
314, 195, 235]
```

NetworkX
Network Analysis in Python

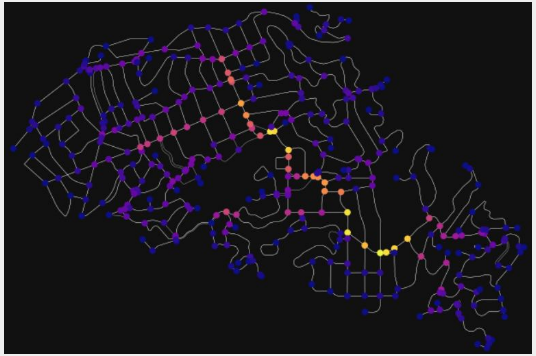


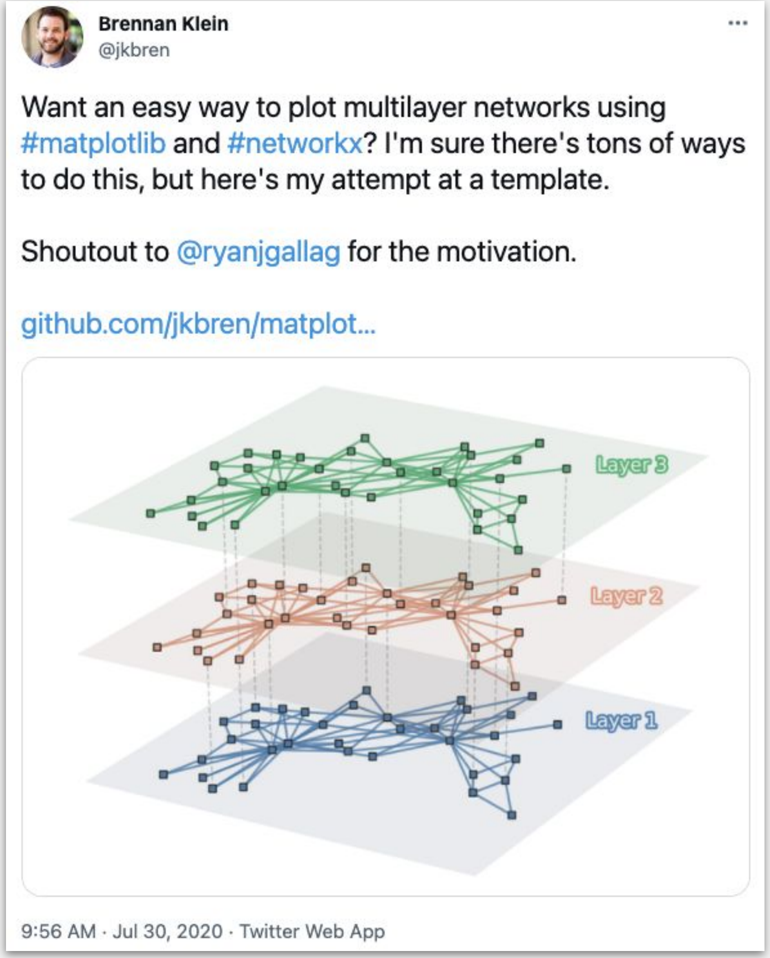

```
# how many k-cores does this network have?
set([v for k,v in nx.core_number(g).items()])
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 17}
```

NetworkX
Network Analysis in Python

Further Analysis

Investigate the PDF, CDF, univariate and bivariate analysis when considering the centrality measurements for each **k-shell** and maximum **k-core** of network.





https://twitter.com/jkbren/status/1288820743963324421