
CN≠CNA
Lesson #18
Measuring Networks
June 2019

- Global measures
- Explore neighborhoods
- Think in terms of paths
- Choose the right centralities

Update from repository

```
git clone https://github.com/ivanovitchm/datascience_one_2019_1
```

Or ....

```
git pull
```

Start with global measures

```
G = nx.read_graphml("cna.graphml")

nx.number_of_nodes(G)   #2995
nx.number_of_edges(G)   #11817
nx.density(G)   #0.00131
```

Explore neighborhoods

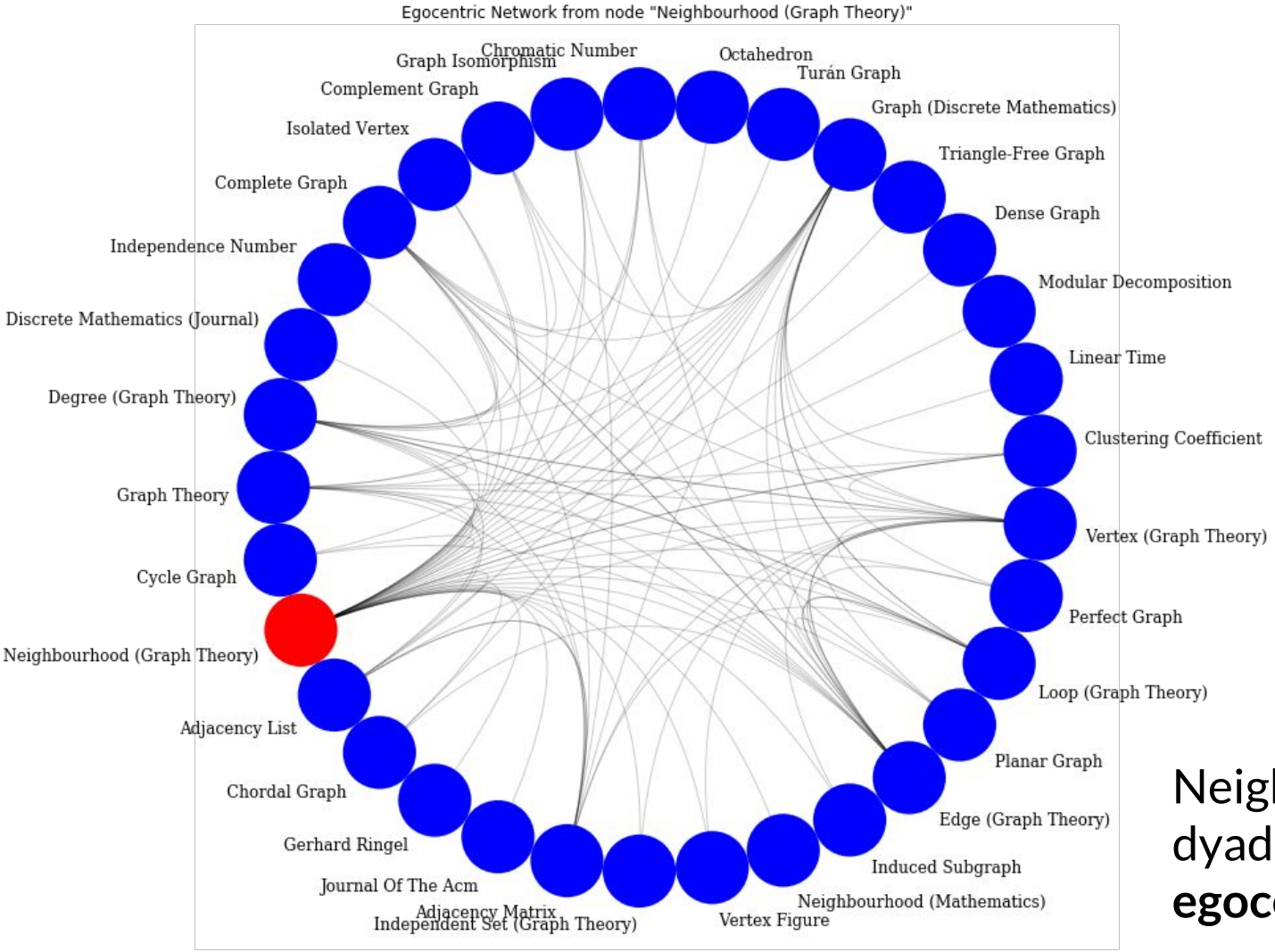
Node and edge counts and density are some of the **macroscopic network properties**.

Neighborhoods are responsible for the local properties of network graphs (**microscopic level**)

- Egocentric network
- Clustering coefficient

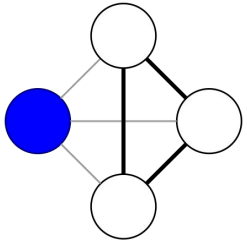
```
ego = "Neighbourhood (Graph Theory)"

alters_1 = G[ego]  #32
alters_2 = list(nx.all_neighbors(G, ego)) # 61
nx.degree(G,ego)  #61
```



Neighborhood as a dyadic structure: **egocentric network**

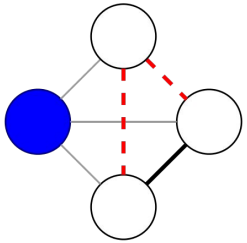
Clustering Coefficient



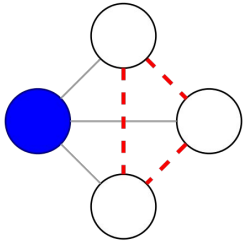
c = 1

Some social theories consider **triads** essential units of social network analysis.

- the **clustering coefficient** is the fraction of possible triangles that contain the ego
- Think of the clustering coefficient as a measure of "**stardom.**"



c = 1/3

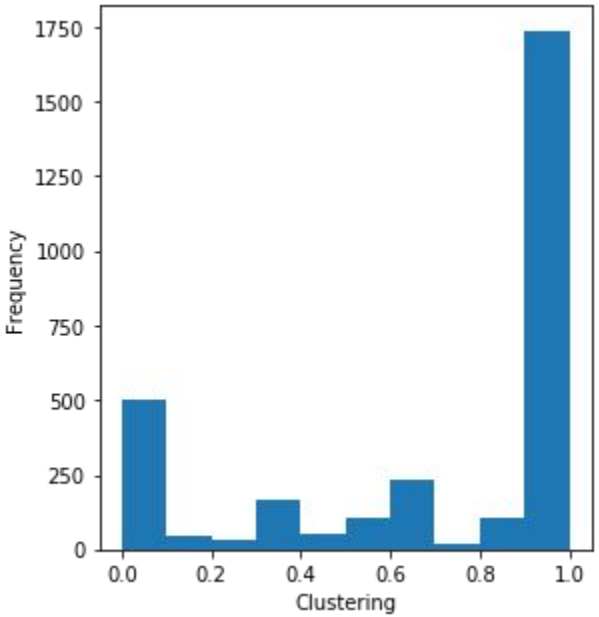


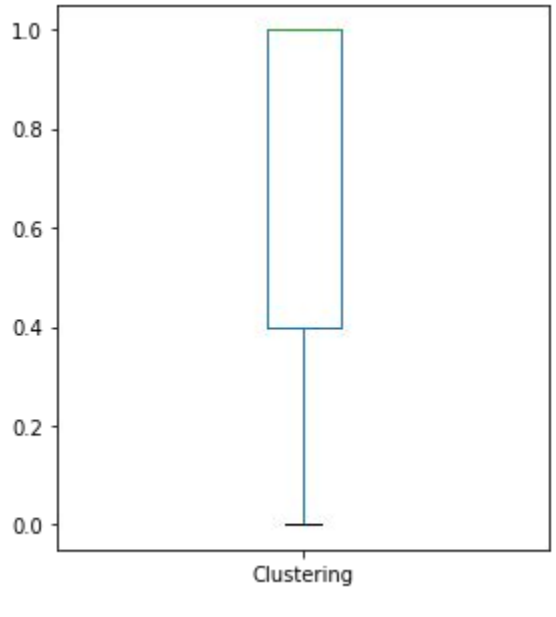
c = 0

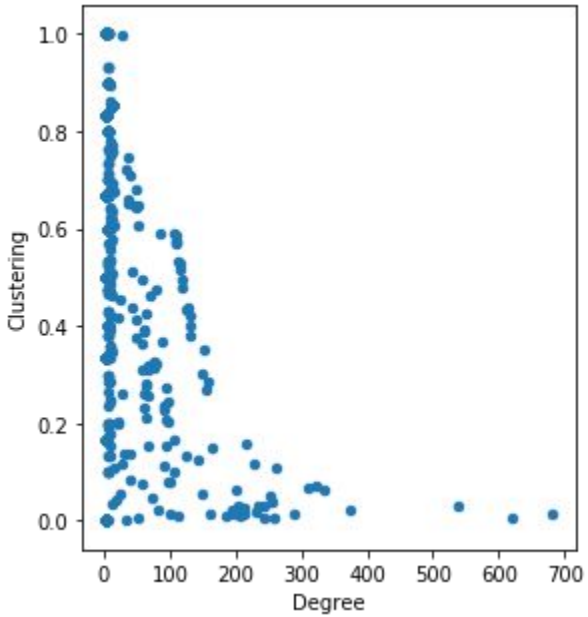
```
cc = nx.clustering(nx.Graph(G), ego)
0.39378
```

Clustering Coefficient

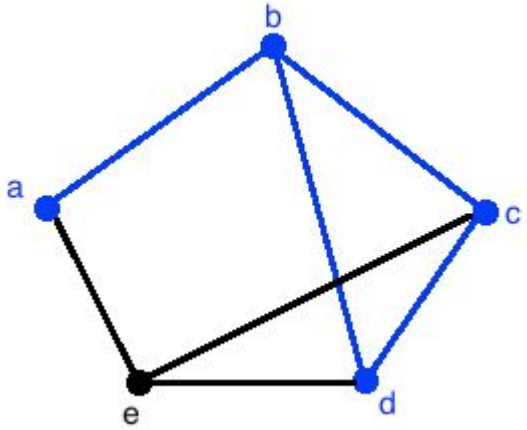
cna.graphml



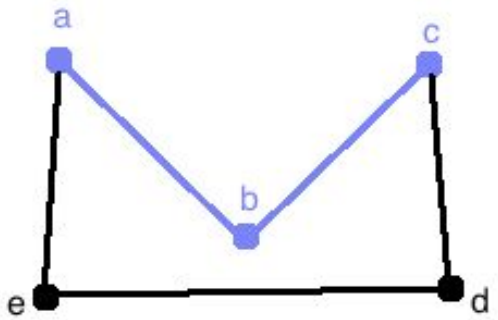




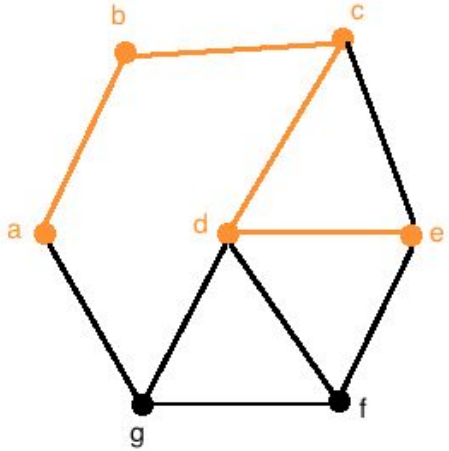
Think in terms of path



Walk
abcdb
abcdcbce
aecbde



Trail
abc
aedc
abcdea



Path
abcde
agdef

The shortest paths are called **geodesics**

```
nx.shortest_path(G,ego,"Webgraph")
['Neighbourhood (Graph Theory)',
 'Degree (Graph Theory)',
 'Degree Distribution',
 'Scale-Free Network',
 'Webgraph']
```

Network as a circle

The **eccentricity** is the maximum distance from a node to all other nodes in the network

```
ecc = nx.eccentricity(nx.Graph(G))
ecc[ego]  #3
```

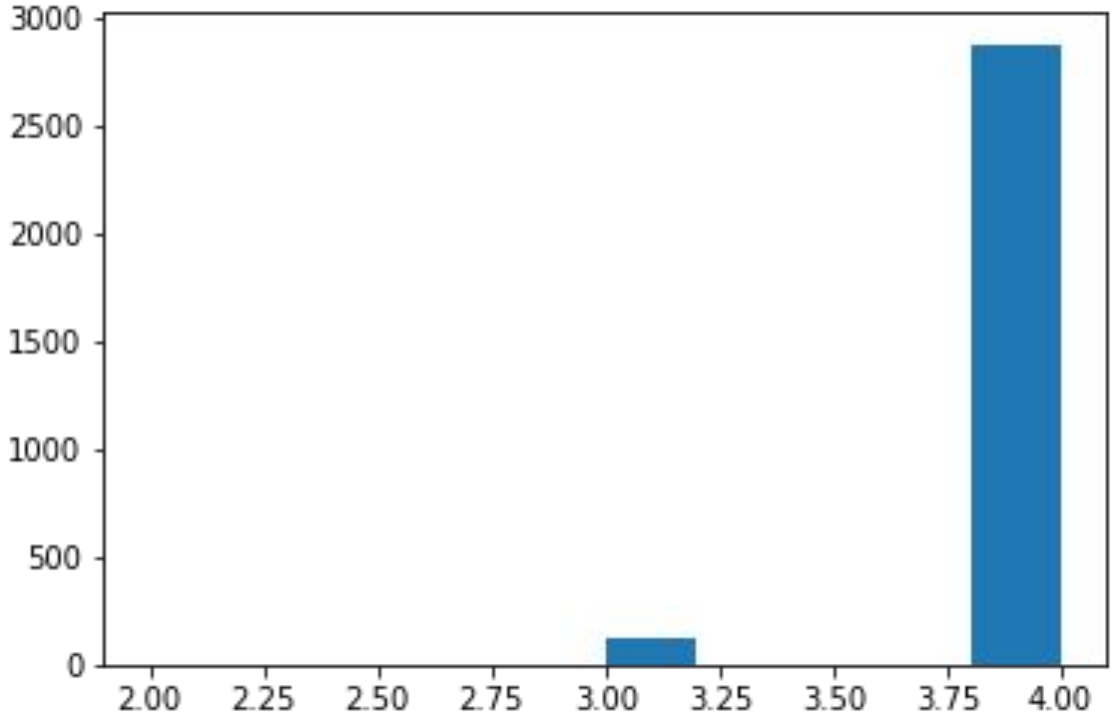
```
# diameter = max(ecc)
ecc[max(ecc,key=ecc.get)] #4

# radius = min(ecc)
ecc[min(ecc,key=ecc.get)] #2

# center => set of nodes with ecc equal to radius
[i for i in ecc if ecc[i] == ecc[min(ecc,key=ecc.get)]]
['Complex Network']

# periphery => set of nodes with ecc equal to diameter
len([i for i in ecc if ecc[i] == ecc[max(ecc,key=ecc.get)]]) #2875
```

```
Counter({2: 1, 3: 119, 4: 2875})
```



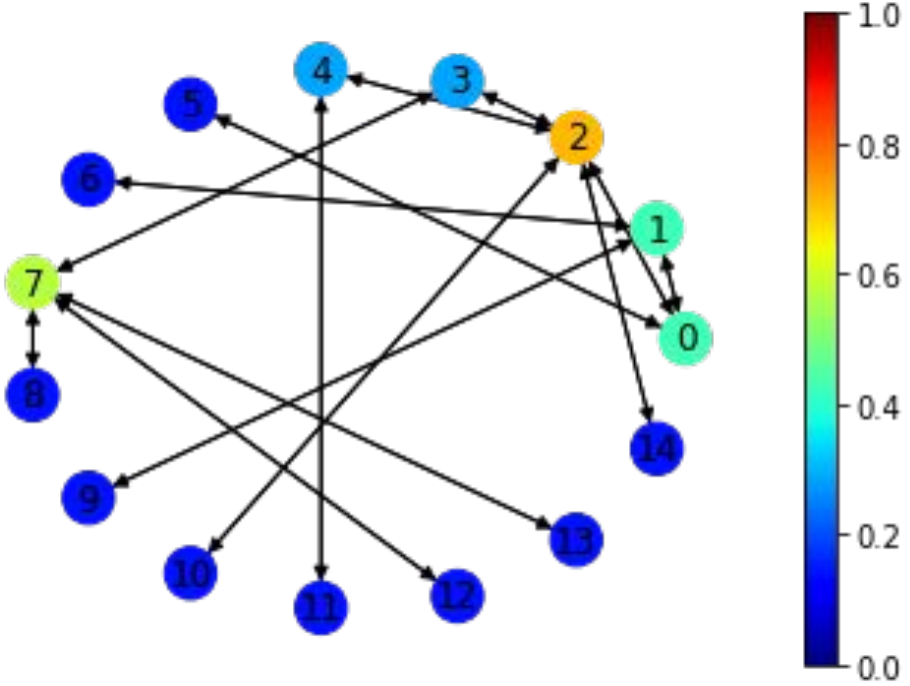
Eccentricity

Choose the Right Centralities

One of the goals of social network analysis is to identify actors with outstanding properties (**most important**)

- Degree centrality
- Closeness Centrality
- Betweenness Centrality
- Eigenvector Centrality

Degree Centrality



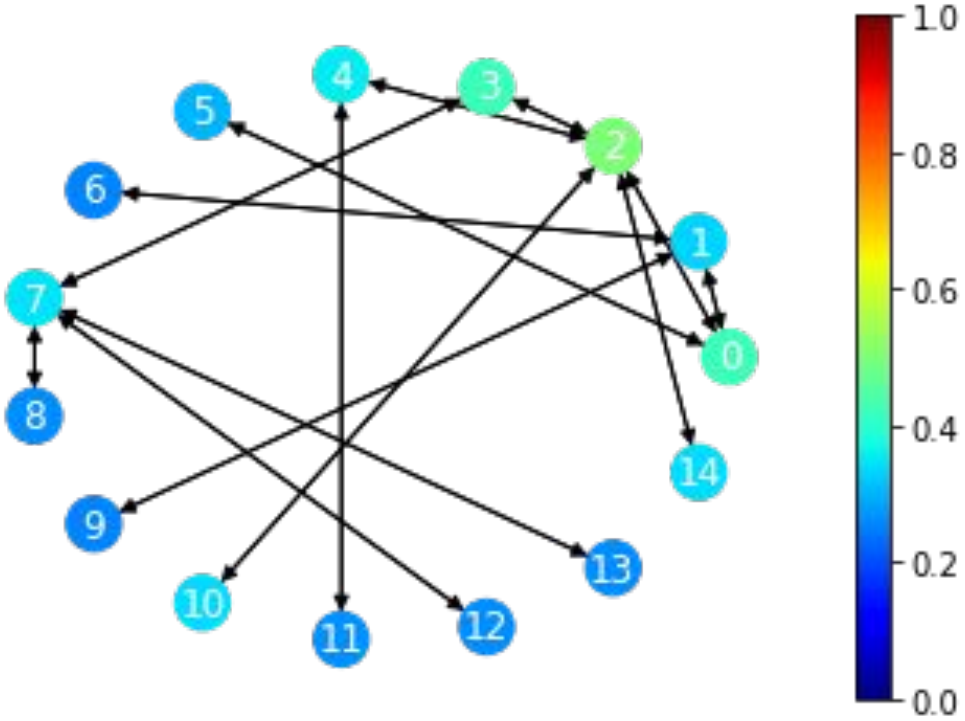
Degree is a simple centrality measure that counts how many neighbors a node has.

If the network is directed:

- in-degree
- out-degree

$$C(node) = \frac{N-1}{\sum_{1}^{N-1} D(node,v)}$$

Closeness Centrality



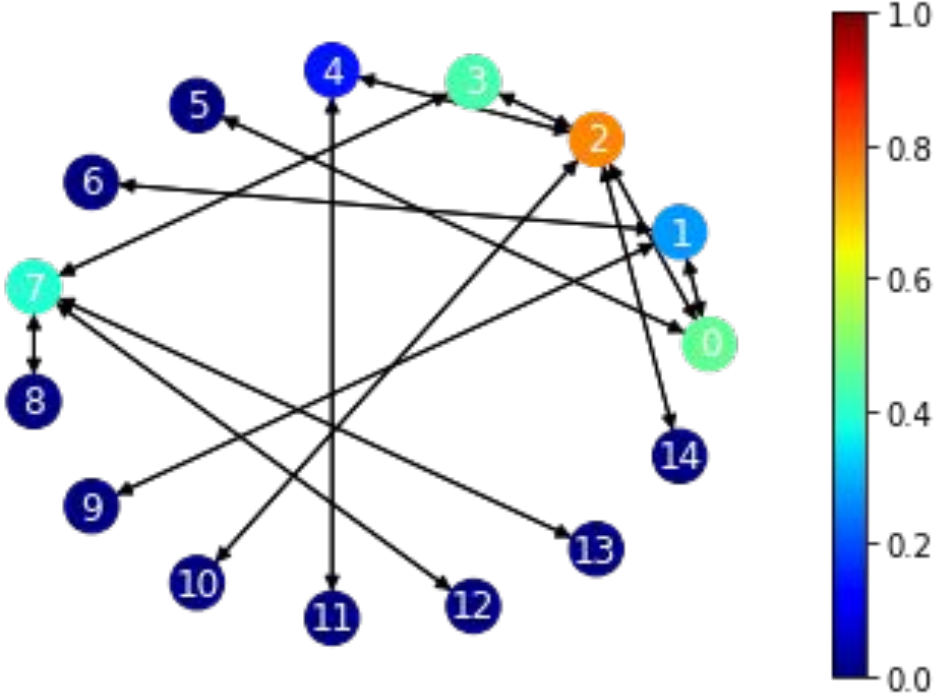
It shows how close the node is to the rest of the graph.

- 0 (the node has no neighbors)
- 1 (the node is the hub of the global star and is one hop away from any other node)

A node with a high degree centrality may be capable of affecting the entire network.

Betweenness Centrality

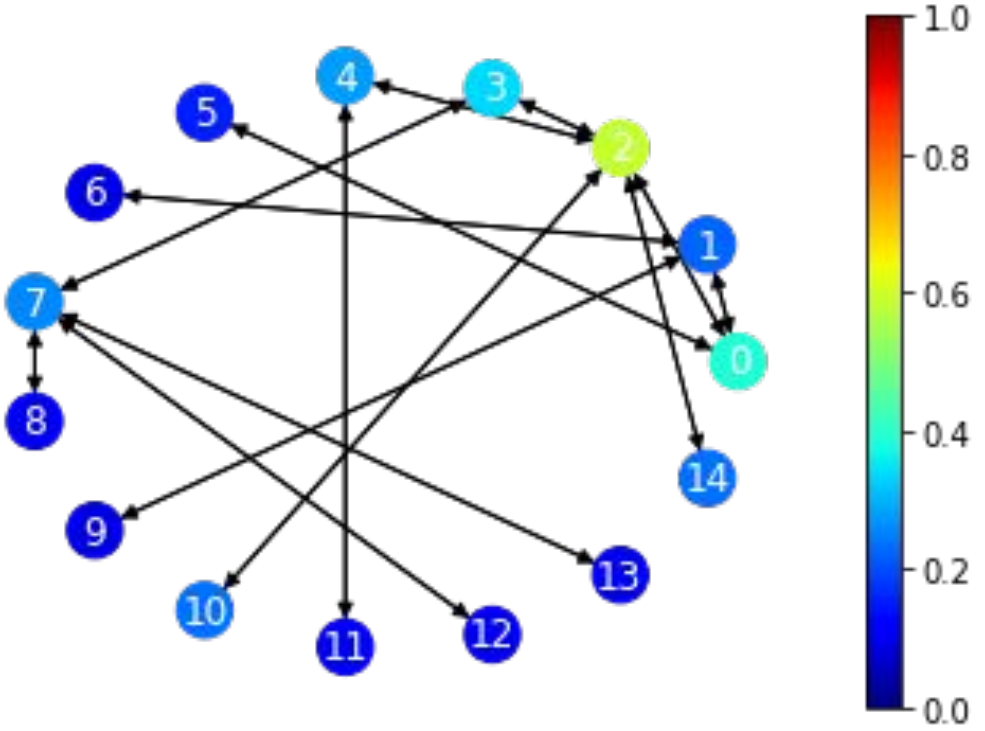
$$betweenness(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$



It measures the fraction of all possible geodesics that pass through a node.

If the betweenness is high, the node is potentially a crucial go-between (thus the name) and has a **brokerage capability**.

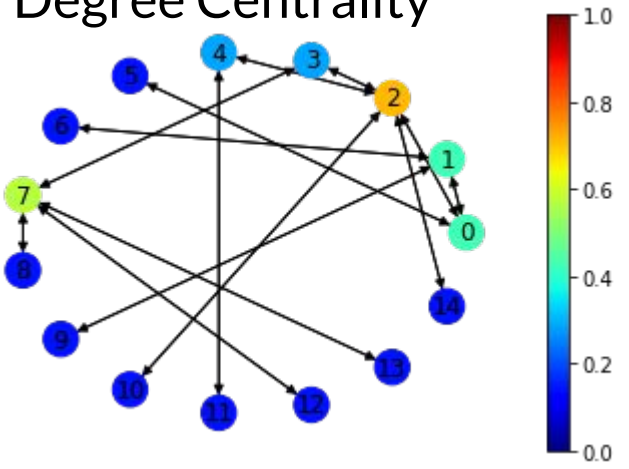
Eigenvector Centrality



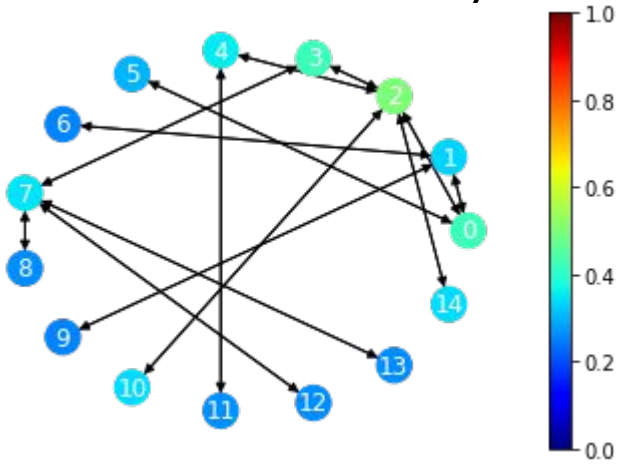
“Tell me who your friends are, and I will tell you who you are.”

You can use this measure to locate groups of interconnected nodes with high prestige.

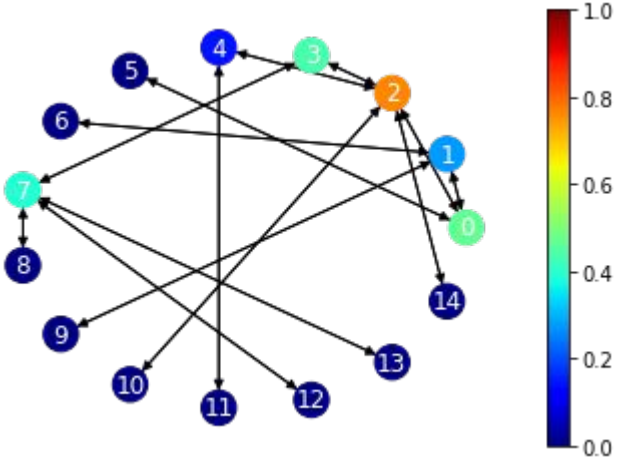
Degree Centrality



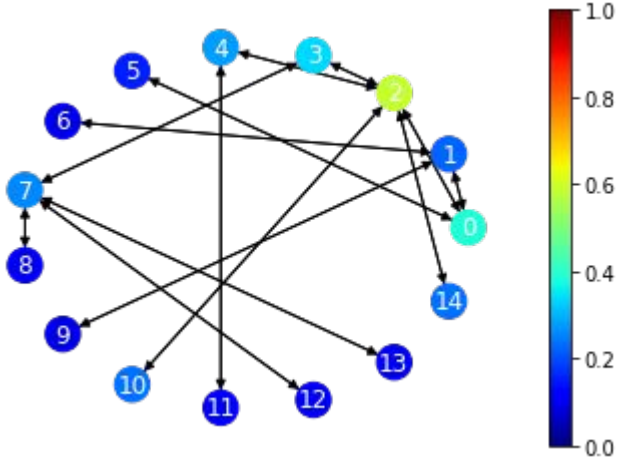
Closeness Centrality



Betweenness Centrality



Eigenvector Centrality



end?