Carlos J.Costa

SNA (SOCIAL NETWORK ANALYSIS)

SNA/Social Network Analysis

- **Conexão/Connections**
  - Homofilia/Homophily
  - Multiplexidade/Multiplexity
  - Reciprocidade/Mutuality-Reciprocity
  - Encerramento da Rede/Network Closure
  - Propinquidade/Propinquity
- **Segmentação/Segmentation**
  - Grupos/Groups
  - Coeficiente de Agrupamento ("Clustering")/Clustering coefficient
  - Coesão/Cohesion
- **Distribuição/Distributions**
  - Ponte/Bridge
    - Centralidade/Centrality
    - Centralidade de Proximidade/closeness centrality
    - Centralidade de Vetor Próprio/ eigenvector centrality
    - Centralidade Alfa/alpha centrality
  - Densidade/Density
  - Distância/Distance
  - Vazio Estrutural/Structural holes
  - Força de Ligação /Tie Strength

SNA: Social Network Analysis

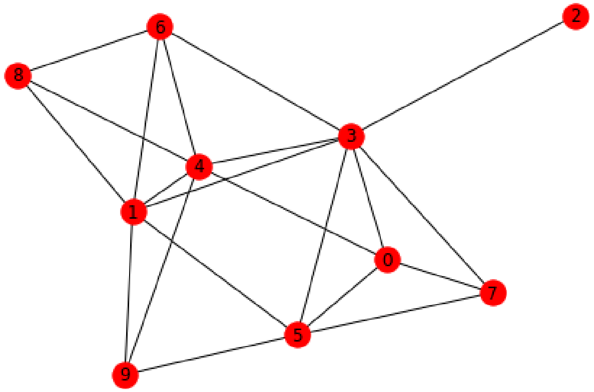
In-Degree	How many people come to this person for information?
Out-Degree	How many people does this person go to for information?
Betweenness	How likely (or how many times) is this person between two other people in network?
Closeness	How fast can this person get information to others in network?
Eigenvector	How well is this person connected to other well connected persons in network?

Importar

- import numpy as np
- import pandas as pd
- import networkx as nx
- import matplotlib.pyplot as plt

Network

```
G = nx.barabasi_albert_graph(10, 3)
nx.draw(G, with_labels=True)
```



Degree

```
# degree of each node  link number that each node has
degrees = [deg for node, deg in nx.degree(G)]
print(degrees)
```

Result:

[4, 6, 1, 7, 6, 5, 4, 3, 3, 3]

Degree

```
# kmin - minimum degree
kmin = np.min(degrees)          → 1

# kmax - maximum degree         → 7
kmax = np.max(degrees)

# kavg - average degree         → 4.2
kavg = np.mean(degrees)
```

Shortest path

```
nx.shortest_path(G,1,2)
```

→ [1, 3, 2]

```
nx.shortest_path(G,1,2, weight=True)
```

Clustering coefficient

```
# triangles
nx.triangles(G)
# clustering coefficient of a node
nx.clustering(G)
# clustering coefficient of all nodes (returns a dictionary)
nx.clustering(G)
# clustering coefficient of the network
cc = nx.clustering(G)
avg_clust = sum(cc.values()) / len(cc)
print(avg_clust)
```

Centrality

- # betweenness centrality of network
- nx.betweenness_centrality(G)
- # closeness centrality of network
- nx.closeness_centrality(G)
- # eigenvector centrality of network
- nx.eigenvector_centrality(G)
- # degree centrality
- nx.degree_centrality(G)

Connected Components

```
# find number of connected components
nx.number_connected_components(G)
# get the nodes in the same component as *n*
nx.node_connected_component(G, 3)
# Assortativity
# Pearson correlation coefficient [-1; 1]
# Social networks are highly assortative (homophily): high degree
# nodes connect to other high degree nodes
# technological are disassortative: high degree nodes connect to low
# degree nodes
nx.degree_assortativity_coefficient(G)
```

Bibliografia

- https://pandas.pydata.org/
- https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html
- https://scikit-learn.org/
- https://scikit-learn.org/stable/index.html
- https://www.statsmodels.org/stable/index.html
- https://networkx.github.io/