

## 1 Networkx and centrality

### 1. Simple network analysis with python and networkx.

`networkx` package has a good documentation. To get started, do the beginning of the tutorial: <https://networkx.org/documentation/stable/tutorial.html>. After that, the best way to find what you're searching for is to ask google. For instance, if you wonder how to compute the betweenness centrality with networkx, just search *betweenness networkx* in google and your first result will certainly contain the answer.

- (a) Using networkx, load the airport dataset using `read_graphml`. (in google colab, you can first retrieve it with `!wget http://cazabetremy.fr/Teaching/CN2020/airportsAndCoord.graphml`)
- (b) Compute the number of nodes and edges of your graph. A simple way to go is to use `len` on `g.nodes` and `g.edges`
- (c) Compute the density and the clustering coefficient (`nx.density(g)`, `nx.transitivity(g)`)
- (d) Compute the average shortest path length and the diameter of the graph. You'll encounter a connected component issue. Can you understand why? As a solution, you need to apply those methods on the largest connected component, that you can extract with `cc = g.subgraph(sorted(nx.connected_components(g), key=len, reverse=True)[0])`
- (e) Obtain the list of the 20 nodes of highest and lowest degrees. You can use `degree` and `sorted(X, key=lambda x: x[1])` for instance.
- (f) Compute the list of the 20 nodes of highest and lowest Pagerank. You might need to use `items()` to transform a dictionary in a list of pair. Observe the differences.
- (g) Do the same for the betweenness (You might need to check parameter `k` of the function). Where in hell is **Anchorage**? And **Port Moresby**? Investigate a little to understand what is going on. How many neighbors do these nodes have, and who are they?
- (h) Check with other typical node centralities.
- (i) Check edge betweenness.
- (j) Would you say that the network is a *small world* network? To compare with a random network, you can generate one with `gnm_random_graph`.
- (k) Plot the network using `draw_networkx`. Check the documentation to see how you could improve your plot (node colors, size, layouts, etc.) Note that in many cases, it is simpler to do it with Gephi.
- (l) Use the `pos` argument to plot nodes according to their geographical position. You can access node attributes either by using `get_node_attributes`, or by simply accessing the node, i.e.: `G.node['node_name']` gives existing attributes of this node, `G.node['node_name']["attribute1"]` gives the value of an attribute for a node
- (m) It can be useful to export a graph with computed node or edges properties. Add their PageRank score to nodes as an attribute using `set_node_attributes`.
- (n) Save the graph in graphml format using `write_graphml`. Check that you can open this file with Gephi and that the PageRank score is available as a node property.

## 2 Communities

### 2. Detecting your first Community Structure

To detect communities, you can use the `cdlib` package. It also contains functions for evaluation and comparison of partitions. For details, check the documentation at <https://cdlib.readthedocs.io/en/latest/>

- Using `networkx`, load the airport dataset, provided as a graphml file. (reminder: you can download it in colab with `!wget URL` with `URL` the url of the file.
- Using `cdlib`, detect communities on this network using the louvain method. You have to use the `algorithms.louvain` method (and do `from cdlib import algorithms` before.
- Visualize the communities found. In order to interpret them, you should draw each node at its geographical location, with a color per community.

There are several ways to draw a spatial network with colors corresponding to communities, from using Gephi to plotting points on an interactive map using `folium`. Here, I provide a simple code to plot the data as a simple scatter plot

Listing 1: plot on a map

```
import seaborn as sns
import matplotlib.pyplot as plt
x= list(nx.get_node_attributes(g,"lon").values())
y= list(nx.get_node_attributes(g,"lat").values())

coms_dict=coms.to_node_community_map()
hues=list(coms_dict[n][0] for n in g.nodes())

plt.figure(figsize=(12,8))
sns.scatterplot(x=x,y=y,hue=hues,palette=sns.color_palette(
"tab20",len(coms.communities)),s=5)
```

- Vary the resolution parameter and observe changes in the community structure.

### 3. Comparing Partitions

- The provided airport data also contains information about the country of each airport, which can be interpreted as a *ground truth* partition of the network. You can obtain it using `nx.get_node_attributes(g,"country")`. Transform this information into a `NodeClustering` object of `cdlib` (`nc = NodeClustering(partition,graph,"GroundTruth")`, with `partition` a list of list of nodes.
- Compute the AMI, NMI, or another similarity score (<https://cdlib.readthedocs.io/en/latest/reference/evaluation.html>) between the community structure and the partition in countries
- By exploring with a for loop the values of the resolution parameter for modularity, find the partition with the highest similarity to the partition in countries.
- Compare visually the results, and try to interpret it. Is the partition in country a meaningful topological partition, i.e., is studying this network by considering that nodes in a same country form a coherent/homogeneous group a good approach? (yes and no, probably...)

### 3 Going Further: Assortativity

#### 4. Characterizing a simple network

- (a) Using `networkx`, load the airport dataset (`graphml`)
- (b) Compute the assortativity of the `country` attribute, using `attribute_assortativity_coefficient` function. What does it mean?
- (c) Compute the degree assortativity using the `degree_assortativity_coefficient` function. What does it mean?
- (d) Compare the degree assortativity with a randomized version of the graph, to check if it is similar or significantly different
- (e) Compute the average degrees of neighbors using `average_degree_connectivity`. You can plot it, for instance as a scatter plot.
- (f) Do the same analysis of degree assortativity and correlation on other networks, for instance those included in `networkx` such as `nx.karate_club_graph()` or `nx.les_miserables_graph()`
- (g) In a previous experiment, you computed communities on the airport graph. You can save those communities as a node attribute in the graph. Get them as a dictionary with `partition=louvain_com.to_node_community_map()`, and insert them using `set_node_attributes`. Be careful, CDlib clusterings allow overlap, so in the dictionary, communities are in lists. So you need to do something like `{k:v[0] for k,v in partition.items()}`. Compute the assortativity coefficient of the community structure(s), and compare it with the one obtained for countries. What do you think of the results?