Experimenting with networkx

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 coefficent (`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 `g.degree` and `sorted(X, key=lambda x: x[1])` for instance.

   (f) 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`.

   (g) Plot the network using `draw_networkx`.

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/`

   (a) 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.

   (b) 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.

   (c) 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)
```

(d) Vary the resolution parameter and observe changes in the community structure.