Experimenting with Community Structure

1. 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/`
>
> **!!!** With google colab, you can install it with `!pip install cdlib`, but, due to a colab bug, you also need to do first:
> `!pip uninstall python-louvain` validate with `y`, then `!pip install python-louvain`.

   (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.

2. Comparing Partitions

   (a) 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.

   (b) 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

(c) 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.

(d) 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 : Intuitions on the SBM

> I propose this exercise using only networkx and cdlib. You could do much more with SBM using `graph-tool` package (real SBM inference, degree-corrected SBM, Hierarchical SBM, etc.), but it requires a little bit more time to get used to at first, so I recommend it only if you're particularly interested in the topic.

(a) Compute the block matrix for a reasonable partition of the graph. For a given partition, you need to count the number of edges between and inside each community.

(b) Using networkx `stochastic_block_model` method, generates a graph based on the computed block matrix

(c) Using the network and node descriptors that you know, compare the properties of this generated graph with the properties of your original graph (and with a simple ER or configuration model). How is it different? How is it similar? Think about clustering coefficient, average distance, distribution of node centralities, degree distribution, etc.

(d) How do these properties change when you increase/decrease the number of blocks?