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/` .

   (a) Using `networkx` , load the second version of the airport dataset, provided as a graphml file

   (b) Using `cdlib` , detect communities on this network using the louvain method. You have to use the `algorithms.louvain` method.

   (c) Add the communities found on your graph as a node property, using `NodeCluster.to_node_community_map()` and `nx.set_node_attributes` .

   (d) 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, the first one, ignoring edges, is to use a simple `scatterplot` . The second is to use the `plot_network_clusters` function of `cdlib` , but you will need to set a custom color palette to see a large number of clusters. Finally, another way is to export the graph in graphml format and to use Gephi to visualize it. You'll need the Gephi plug-in called *Geo Layout*, and choose an **Equirectangular** Projection with the proper **scale**. You can generate your own palette with as many colors as required, from the interface. Although indirect, it is probably the most flexible solution, since it allows to see node labels and to easily change node visualization parameters. You could also use `folium` library to obtain an interactive visualization on a map.

   (e) 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. Transform this information into a `NodeClustering` object of `cdlib` (`NodeClustering(partition,graph,"GroundTruth")` .

   (b) Compute the AMI or NMI between the community structure and the partition in countries

   (c) By exploring systematically 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 a 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?