

1 Null Models

1. Comparing networks and their randomized versions.

- (a) Using `networkx`, load the airport dataset.

It can be useful to work on a smaller network for experimenting (you can go back to the original graph when your functions are ready). To work on a subnetwork composed of the nodes of largest degree, you can do for instance:

Listing 1: Filtering out low degree nodes

```
to_keep = [n for n,v in g.degree() if v > 10]
g_small = g.subgraph(to_keep)
```

- (b) Generate an Erdos-Renyi (ER) random version of it. You can use `gnp_random_graph` and `gnm_random_graph` methods.
- (c) Generate also a configuration model (degree preserving) version of it, using `expected_degree_graph`, and the degrees observed for the real network (e.g., with `g.degree`).
- (d) Compare the network properties of the 3 different versions of the graph, in particular the average degree, average clustering coefficient (`transitivity`), and average path length. Interpret in terms of small-worldness.
- (e) To have a statistically robust approach, generate 100 versions of the configuration model and `gnp` model, and compute their average clustering coefficient. Plot the distribution of values obtained. Compare with the value on the real network to confirm that the observed transitivity is significantly higher (Bootstrapping confidence).

2. Resilience

When analyzing networks of infrastructure, transport or telecommunication, the *resilience* or *robustness* of the network is a popular investigation topic. Resilience can be defined as the capacity of the network to resist attacks and/or failures. The resistance can be measured in several ways, and attacks and failures can be modeled differently too. We will briefly investigate this question on the airport dataset.

- (a) Let's define a simple measure of how *efficient* the network is as the fraction of nodes belonging to the largest connected component. Write a function computing this score, given a network.
- (b) Write a function that, given an ordered list of nodes, removes them one by one from the network, computes the resilience measure at each step, and returns the result.
- (c) First, let's compute the resilience of the airport network to *failures*, i.e., random node removals. Generate a vector containing all nodes in a random order, and call the function defined above on it. Plot the results, i.e., resilience as a function of nodes removed. Repeat a few times to see how much it depends on the order of nodes.
- (d) Let's now test the resilience to *attacks*, i.e., we will remove nodes in a specific order, and search for the most damaging strategy. Compare attacks based on the PageRank and the Betweenness.
- (e) Compare the same process (Random attacks, attacks targeted on PageRank/Betweenness on ER and configuration versions of the airport dataset).

- (f) Try to interpret the resilience in terms of the classic network descriptions, i.e., degree distribution, clustering...

2 Community Detection

3. Understanding Modularity.

- (a) To be sure that you understand correctly the modularity, write a function that computes it for a given partition and a given graph, using the Configuration model as a null model.
- (b) Check that you obtain the same results as using the `modularity` function from `networkx`. (Be careful about weights)(you can start with a small graph, for instance the `karate_club_graph` of `networkx`.)
- (c) If not already done, make your code efficient by remembering that the score depends only on pairs of nodes inside communities. You can also use the formulation by community.
- (d) Add the possibility to choose the null model, so that one can use ER random graph instead of the configuration model.

4. 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")` .
- (b) Compute the AMI (Adjusted Mutual Information), from the library `scikit_learn` 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 think about the reason why we should not expect to find exactly communities corresponding to countries.