Exercise set #8 (17 pts)

- The deadline for handing in your solutions is November 18th 2019 23:55.
- Return your solutions (one `.pdf` file and one `.zip` file containing Python code) in MyCourses (Assignments tab). Additionally, submit your pdf file also to the Turnitin plagiarism checker in MyCourses.
- Check also the course practicalities page in MyCourses for more details on writing your report.

1. Network sampling (9 pts)

Many network data sets are samples of some underlying graphs that we are actually interested in. That is, nodes and edges of these empirical networks have been sampled in a way that we only observe parts of them. This can severely bias even the most simple network measures we compute in a way that the values the measures get for the sampled graph can be very different than for the underlying graph. In this exercise we will see the effect of three different sampling schemes on network transitivity C and derive estimators that can be used to correct for these biases.

Let us first recall the definition of transitivity C, defined during the lectures as:

$$C = \frac{\tau_\triangle}{\tau_\angle} = \frac{\sum_i E_i}{\sum_i \binom{k_i}{2}}\,, \tag{1}$$

where $\tau_\angle$ is the number of two-stars (a node and two of its neighbors), $\tau_\triangle$ is three times the number of triangles [1] (two-stars in which the neighbors are connected), i is a node, k_i is the degree of i, and E_i is the number of triangles centered on i (in other words, how many triangles pass by i).

In this exercise we will use the Horvitz-Thompson (HT) estimator to get a better estimate of $\tau_\triangle$ and $\tau_\angle$ under different sampling schemes. We will then use these estimators to get an estimator for C by simply plugging them in the formula for C. In short, HT is a weighted estimator, where the weights are determined by the inverse of the probability of observations. This way, observations that are more unlikely to be sampled have a larger weight, and observations that are more likely to be sampled have a smaller weight.

The HT estimator is used to estimate totals of the form:

$$\tau = \sum_{i \in U} y_i\,,$$

where U is the underlying population of units (in networks it could be edges, triangles, nodes of certain degree ...), and the y_i is the number associated to unit i. Often we simply want to calculate the number of units, and in these cases we set $y_i = 1$ for every i. The HT estimator can be written as

$$\hat{\tau} = \sum_{i \in S} \frac{y_i}{\pi_i}\,,$$

[1] This is three times the number of triangles as we are going through all the nodes - and thus counting each triangle three times. We could define an alternative estimator that corrects for this overestimation, but for transitivity $\tau_\triangle = \sum_i E_i$ suffices.

where $S \subseteq U$ is the random sample and π_i is the probability of observing the unit i.

a) (1 pt) First let's focus on the case of Bernoulli sampling of nodes, where each node is sampled with probability p, and we observe an edge if and only if we have sampled the two nodes that form it. **Calculate** the probability of observing (i) a two-star and (ii) a triangle, commenting on how you obtained these probabilities. Notice that the probability of observing a stucture refers only to the probability of sampling it given that it exists, not to the probability of that structure first occuring in a network and then sampling it. Then, **write** the HT estimator for the total number of two-stars $\hat{\tau}^n_{\angle}$ and the HT estimator for the total number of triangles $\hat{\tau}^n_{\triangle}$, where the n refers to the fact that we have perfomed sampling of nodes.

b) (1 pt) Now we perform Bernoulli sampling of edges, where each edge is sampled with probability p, and a node is observed if one of its edges is sampled. For this case also **calculate** the probability of oberving (i) a two-star and (ii) a triangle, commenting on how to derive them, and **write** the HT estimator for the total number of triples connected by two edges $\hat{\tau}^e_{\angle}$, and the total number of triangles $\hat{\tau}^e_{\triangle}$, where e refers to the sampling of edges.

c) (1 pt) Our last sampling scheme is star sampling. In this case, we sample each node with probability p, and then observe all of its edges (an example would be a data set obtained by crawling through friendship lists of randomly selected users in a social networking website). As before, **calculate** the probability of observing (i) a two-star and (ii) a triangle, commenting on how to derive them and including the corresponding HT estimators $\hat{\tau}^s_{\angle}$ and $\hat{\tau}^s_{\triangle}$ for star sampling. To make calculations easier, for your two-star estimators use only the sampled nodes, i.e., use the observed degrees of sampled nodes only, and do not use the degrees of nodes that were not sampled but whose neighbors were sampled. Otherwise, the sampling probability will depend on a node's degree, which is something we want to avoid for now.

Hints:

- For obtaining the probability of observing a triangle, consider that this happens when either only two nodes or exactly the three nodes are observed.

d) (1 pt) Finally, let's construct our HT estimators for network transitivity C under our sampling schemes. In all cases we use a plug-in estimator, where you can simply substitute (or "plug-in") the HT estimators found in a), b) and c) into the equation 1. **Report** your HT estimators for transitivity under the sampling schemes τ^n_C, τ^e_C and τ^s_C, and **comment** on their differences, or the effect of the different sampling schemes on the estimators.

Next, let's put our estimators to work. Start by generating a network from a random model. Use the command `nx.relaxed_caveman_graph(200, 10, 0.1)` to generate a graph with 200 communities of ten people each, where each link is then rewired with probability 0.1. We will assume that this is the "true" graph from which we obtain samples.

e) (2 pt) Write functions for sampling under the schemes described on a), b) and c). **Obtain samples** of the network using all schemes with probabilities $p = 0.1, 0.3, 0.5, 1$ (of course, for $p = 1$ there is no need to sample, you can simply use the full network). Then, use your own code to obtain empirical estimates of the number of triangles, the number of two-stars, and transitivity. **Report** your results on three tables, one for each sampling scheme, where the rows represent the different sampling probabilities, and the columns represent the empirical number of triangles, two-stars, and transitivity. **Comment** on the effect of sampling schemes on the network statistics and why you think that is. Are we more likely to observe certain structures under the different schemes? Are all schemes similar in other cases? Why?

Hints:

- In this case, the empirical estimator $\hat{\tau}$ may be understood as simply counting the triangles/two-stars in our the sampled networks, without the HT estimators.
- Since we are dealing with samples of networks, you may get very different numbers every time you run the sampling. It is normal that the specific numbers change (sometimes a lot), but in general you should be able to see differences for the sampling schemes and probabilities.
- For the case of star-sampling, we estimate the number of two-stars using only the sampled nodes and not the complete graph. When writing your function for star sampling, remember to return an object with sampled nodes, as you don't want to include all nodes in your estimates.

f) (2 pt) Implement the HT estimator for the three sampling schemes. Given the selection probability p for all cases, we will sample at least $n = 100$ times to obtain distributions of some of our HT estimators, and to compare HT estimators with the empirical estimator (on the graph without any corrections). In other words, for each $p = 0.3, 0.5$, and for each sampling scheme, obtain $n = 100$ samples from the original network, calculate the HT estimator for the number triangles and for transitivity. For $p = 0.5$ include also the empirical estimator, and for all plots include the true value. **Report your results in six plots**:

- The distribution of estimator $\hat{\tau}^{n}_{\triangle}$ for $p = 0.3, 0.5$ and $\hat{\tau}_{\triangle}$ for $p = 0.5$.
- The distribution of estimator $\hat{\tau}^{n}_{C}$ for $p = 0.3, 0.5$ and $\hat{\tau}_{C}$ for $p = 0.5$.
- The distribution of estimator $\hat{\tau}^{e}_{\triangle}$ for $p = 0.3, 0.5$ and $\hat{\tau}_{\triangle}$ for $p = 0.5$.
- The distribution of estimator $\hat{\tau}^{e}_{C}$ for $p = 0.3, 0.5$ and $\hat{\tau}_{C}$ for $p = 0.5$.
- The distribution of estimator $\hat{\tau}^{s}_{\triangle}$ for $p = 0.3, 0.5$ and $\hat{\tau}_{\triangle}$ for $p = 0.5$.
- The distribution of estimator $\hat{\tau}^{s}_{C}$ for $p = 0.3, 0.5$ and $\hat{\tau}_{C}$ for $p = 0.5$.

Elaborate on the effect on sampling schemes and probabilities. Do the HT estimators lie near the true value? How do they vary? What are the main differences between sampling schemes? Do you notice anything particular about the sixth plot? If so, why is this?

Hints: Since we want to observe how different samples may arise from the same network, we need to take a large number of samples ($n = 100$, for instance). However, while you are coding and testing it may be wise to use a smaller n. Keep in mind that if your code takes too much time to run on your computer, you can report results for a smaller n or a smaller network.

g) (1 pt) Let us now put the estimators we derived in use for real data. Below is a description of two network data sets and their collection methods. **Match** the data sets with one of the three sampling methods used and HT estimators we derived in this exercise. Note that you might need to do some assumptions about the uniformity of the sampling. **Calculate** the empirical esitimator and the plug-in HT estimator for the transitivities of these data sets.

1. A social communication network was built based on call data records of a mobile phone operator. These records contain all the incoming and outcoming calls the customers make and receive, but no data for the calls between pairs of customers who are both with a competitor. The network is built in a way that each phone number appearing in the data becomes a node and there is a link between them if there is at least one call between the numbers. The underlying network we would be interested in is the communication network between all the customers of all companies in the country it operates. The company has $6,421,148$ customers which amounts to a market share of 20% of customers in the country. The network contains $75,048,105$ nodes, $529,816,040$ links, and there are $153,328,324$ triangles and $15,856,481,566$ two-stars around our customers. Note that since our network contains all incoming and outgoing calls - including numbers in other countries, the number of customers is not equal to 20% of the network.
2. Using the same mobile phone data as above, we build a social network between the customers of the company. That is, we leave out the phone numbers which are not operated by our mobile phone operator. The underlying network we would be interested is the actual contact network between the company customers. Based on a small questionnaire we have made, on average only 10% of the contacts take place via mobile phones in that country at that time. The network contains $6,421,148$ nodes, $26,415,938$ links, $11,794,486$ triangles and $634,259,263$ two-stars.

2. Modularity (8 pts, pen and paper)

When nodes of a network are partitioned into communities (or clusters), *modularity* can be used to measure how well the given partition catches network's community structure. The modularity Q of a partition into communities can be written as

$$Q = \sum_{c \in \mathcal{P}} \left[\frac{l_c}{L} - \left(\frac{d_c}{2L} \right)^2 \right] , \tag{2}$$

where the sum runs over all clusters/modules/communities, L is the number of links in the whole network, l_c is the number of links internal to cluster c (internal = both endpoint nodes are in c), and d_c is the total degree of nodes in cluster c (sum of the degrees of nodes in the cluster, such that the degrees count all links attached to the nodes irrespective of if they go inside or outside of the community). In other words, modularity is defined as the difference between the relative number of links inside the communities and the expected relative number of links inside the communities in a randomized network (usually configuration model) without clear community structure.

a) (2 pts) **Calculate** the value of modularity for the two partitions shown in Fig. 1 (in the first, there are two clusters, and in the second, the whole network is taken as a single cluster).

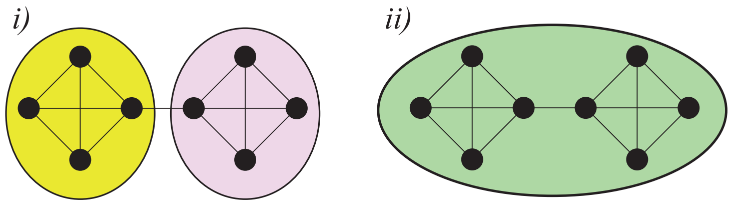

Figure 1: Two module configurations for a)

In the rest of this exeercise, we will show that the modularity Q defined in the above Eq. (2) contains an intrinsic scale. This means that the size of communities that are favored by modularity optimization depends on the number of links L in the network. Therefore, sometimes subgraphs which should by common sense be labeled as separate modules, such as full cliques attached to the rest of the network with a single link, get merged when Q is maximized.

Consider now a very general case – a network where there are altogether L links. Suppose that we have somehow *a priori* identified two groups of nodes a and b which we consider as modules (say, because they are cliques). Let us also assume that there is a single link connecting these two modules[2]. Instead, we make no assumptions about the number of links that connect a and b to the rest of the network. For a schematic illustration, see Figure 2.

Let's now consider two alternative ways of parcellating this network into communities: either 1) consider the two real modules as two modules, or 2) merge them into a single module ab. In community detection based on modularity optimization, we select the partition that yields higher modularity. So, the idea now is to calculate the difference in modularity $\Delta Q = Q_2 - Q_1$ between these two partitions using the formulation of Eq. 2 for modularity. When this difference is positive modularity optimization will merge the two "physical" modules.

b) (2 pts) **Write** ΔQ as a function of L, d_a, d_b, d_{ab}, l_a, l_b, and l_{ab}.
Hint: Note that the part of the modularity that comes from any other community than a, b or their merger ab is cancelled out when Q_1 is substracted from Q_2. This can be seen easily by writing in both formulas Q_1 and Q_2 the part of the sum that deals with communities in the rest of the network as $Q_R = \sum_{c \in \mathcal{P}'} \left[\frac{l_c}{L} - \left(\frac{d_c}{2L} \right)^2 \right]$, where $\mathcal{P}'$ is the set of communties in the rest of the network.

c) (2 pts) **Write** d_{ab} as the function of d_a and d_b, and l_{ab} as the function of l_a and l_b. Using these substitutions **show** that the option of merging two clusters becomes favored by modularity optimization when $L > \frac{d_a d_b}{2}$. **Explain** why this result indicates that modularity optimization must have a *resolution limit*, or a minimum size of community that can be found that depends on the size of the network (where size is the number of links). You may assume that the network is sparse so d_a, d_b, and d_{ab} are constants when L increases.

d) (2 pts) Next, **provide an argument** that the resolution limit of modularity has a form such that in a network with L links it is not possible to find communities of size smaller than $n \propto \sqrt{L}$. **Use** the following steps in your argument:

   i) Assume that community a is a clique of n nodes and that there is one single link

[2]Similar, although slightly more general, calculations could be made by assuming that there are $l_{a \leftrightarrow b}$ links connecting the two communities without much difference in the results. Here we, however, concentrate on the limiting case of only a single connecting link.

connecting a to the rest of the network. Write the formula for d_a as function of n. To make things slightly simpler, you can approximate $d_a \approx n^2$ for the rest of this exercise.

ii) Using the results of c) show that the community a is always merged to some other community when $n < C\sqrt{L}$, where C is some constant that doesn't depend on L.

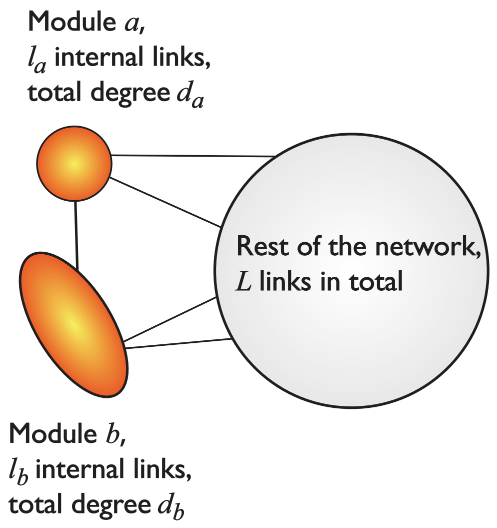

Figure 2: Schematic figure of two modules a and b, connected with a single link. Note that in b) and c) we do not make any assumptions of how many links are connecting the two modules to the rest of the network.

Feedback (1 pt)

To earn one bonus point, give feedback on this exercise set and the corresponding lecture latest two day after the report's submission deadline.
Link to the feedback form: `https://forms.gle/e11d6RuBqrkxqkh16`.

References