Experimenting with Dynamic networks

Several libraries exist to manipulate temporal networks, but most of them are not fully mature. I propose to use `tnetwork`, that can be installed using `pip`.
Documentation and examples: https://tnetwork.readthedocs.io.

1. Characterizing a dynamic network as a sequence of snapshots
   (a) Using `tnetwork`, load the dynamic network of interactions between people in a hospital, collected by the sociopatterns project as a sequence of snapshots. (1 snapshot=20s, analysis over a few days)
   You can do this with: `g = tn.graph_socioPatterns_Hospital(format=tn.DynGraphSN)`

   > You can read about this dataset and other information about the sociopatterns project on the following page: http://www.sociopatterns.org/datasets/primary-school-temporal-network-data

   (b) Count the number of snapshots. You can obtain the list of times at which snapshots occur with `g.snapshots_timesteps()`.
   (c) Obtain the graph corresponding to the first snapshot. You can use the `g.snapshots()` method which return a sorted dictionary, in which keys are times of snapshots and values are snapshots represented by a `networkx` graph object.
   (d) Compute the number of nodes and edges in this snapshot, and plot it.
   (e) What is the total number of interactions among all nodes at all time (edge-time)? What is the total number of different nodes? (you can use a `for` loop, and `set` or `collections.Counter` to handle repeated elements.
   (f) Plot the evolution of the number of nodes, number of edges and density along time. Plot the evolution of the degree of one particular node. Would you say that this network is rather stable or unstable?
   (g) You can plot several graphs simultaneously using the following function:
   `graph = tn.plot_as_graph(g,ts=[1254386420,1254386440,1254386460])`, with ts being a list of timestamps.
   (h) You can plot the presence of nodes in snapshots with the `tn.plot_longitudinal(g,to_datetime=True)`.
   The `to_datetime` parameter allows to transform datetime to their corresponding date. Be careful, it might take about 30s on the full graph.
   (i) Compute the aggregated graph using `g.cumulated_graph()`. Plot this graph, compute its number of nodes, edges, density.
   (j) Compute snapshots aggregating activity every hour using `g2 =g.aggregate_time_period("hour")`. Analyze the resulting network in a way similar to the first graph. Do you think that this network is stable or unstable?
   (k) Compute (you have to write the code yourself) the dynamic version of "number/quantity" of nodes N, "number/quantity" of edges L and density d for the whole period, then for one particular day. You will have to use a `for` loop over all snapshots.

2. Going further: **Dynamic Communities**
   (a) Read the documentation of `tnetwork` about the detection of dynamic communities: https://tnetwork.readthedocs.io/en/latest/notebooks/demo_DCD.html.

(b) Apply several methods on the primary school dynamic network aggregated every hour and compare the results. (`tnetwork.graph_socioPatterns_Primary_School(format=tn.DynGraphSN)`

(c) Compute communities on the graph in its original form (no aggregation). Compute static communities on the cumulated graph (aggregated over the whole period). What do you think of the communities found using those three different approaches?