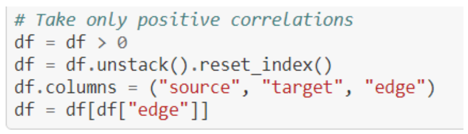
Jerico Miguel T. Rangel
2018-11666

Report 3: Network Tools
AP157 A

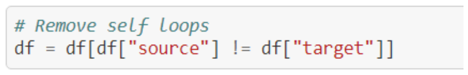
Key concepts

- Networks are represented using graphs which are composed of nodes and edges. These components are characterized by weights, degrees, direction, and other parameters which help define the relationships in the network. These graphs can contain multiple layers with each having certain densities and levels of clustering which help accurately model real-life networks.
- Certain nodes in the network vary in importance, which can be measured using degree, closeness centrality, betweenness centrality, and reach. These parameters are important in the efficiency of analyzing networks which include identifying shortest paths and optimizing applications of the network.
- Real-life networks are often found to be highly clustered with small diameters, and this can be explained most basically by Watts and Strogatz' small world model. These networks are somewhat unorderly, and the constituents are close to one another.

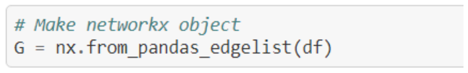
Essential code snippets



```
# Take only positive correlations
df = df > 0
df = df.unstack().reset_index()
df.columns = ("source", "target", "edge")
df = df[df["edge"]]
```



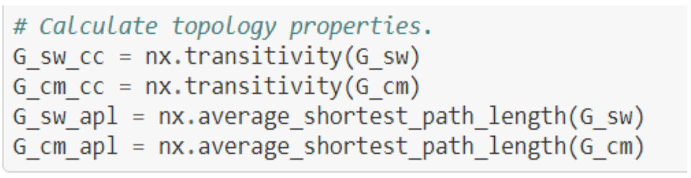
```
# Remove self loops
df = df[df["source"] != df["target"]]
```



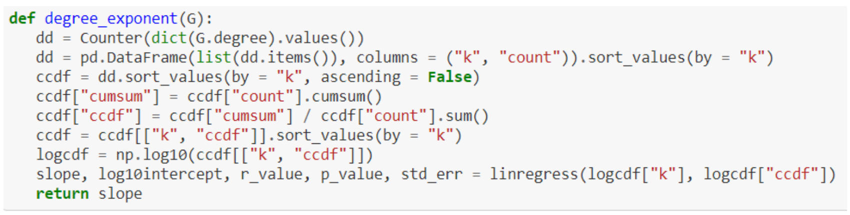
```
# Make networkx object
G = nx.from_pandas_edgelist(df)
```

Shown on the left is code which shows how to display or visualize a simple correlation network at a basic level, without worrying about the more complex parameters.

Being able to use the networkx python package and understanding the corresponding importance of each topological property allows us to analyze these network models properly.



```
# Calculate topology properties.
G_sw_cc = nx.transitivity(G_sw)
G_cm_cc = nx.transitivity(G_cm)
G_sw_apl = nx.average_shortest_path_length(G_sw)
G_cm_apl = nx.average_shortest_path_length(G_cm)
```



```
def degree_exponent(G):
    dd = Counter(dict(G.degree).values())
    dd = pd.DataFrame(list(dd.items()), columns = ("k", "count")).sort_values(by = "k")
    ccdf = dd.sort_values(by = "k", ascending = False)
    ccdf["cumsum"] = ccdf["count"].cumsum()
    ccdf["ccdf"] = ccdf["cumsum"] / ccdf["count"].sum()
    ccdf = ccdf[["k", "ccdf"]].sort_values(by = "k")
    logcdf = np.log10(ccdf[["k", "ccdf"]])
    slope, log10intercept, r_value, p_value, std_err = linregress(logcdf["k"], logcdf["ccdf"])
    return slope
```

For deeper analyses on network models, the degree distribution exponent is a parameter that gives us insight on the varying trends in node importance. These however follow a somewhat preferential model since the method gives natural importance to older nodes. More realistic networks require better models such as the configuration model or the stochastic block model.

Pitfalls

- Understanding semantics and the many functions from the networkx python package.
- Properly inferring applications from the different parameter values obtained.
- Adjacency matrices and being able to intuitively relate it to the network.