Final Project: Part 1

Karros Huang, Girish Narayanswamy, and Clint Olsen

ECEN 5322 Higher-Dimensional Datasets

University of Colorado
Boulder

Introduction

The following is an exploration of disease control and disease-spread analysis. The premise is to understand if co-presence data, a measure of individuals in a similar vicinity, is a reliable indicator of face-to-face contact, a measure of individuals interacting directly in close proximity.

This communication network analysis has applications beyond disease control. Similar analysis can be used for any type of communication network including but not limited to propagation of malware, and the spread of information over social media.

Data

The dataset used for this analysis comprises of face-to-face and co-presence data for 6 different groups of people/locations.

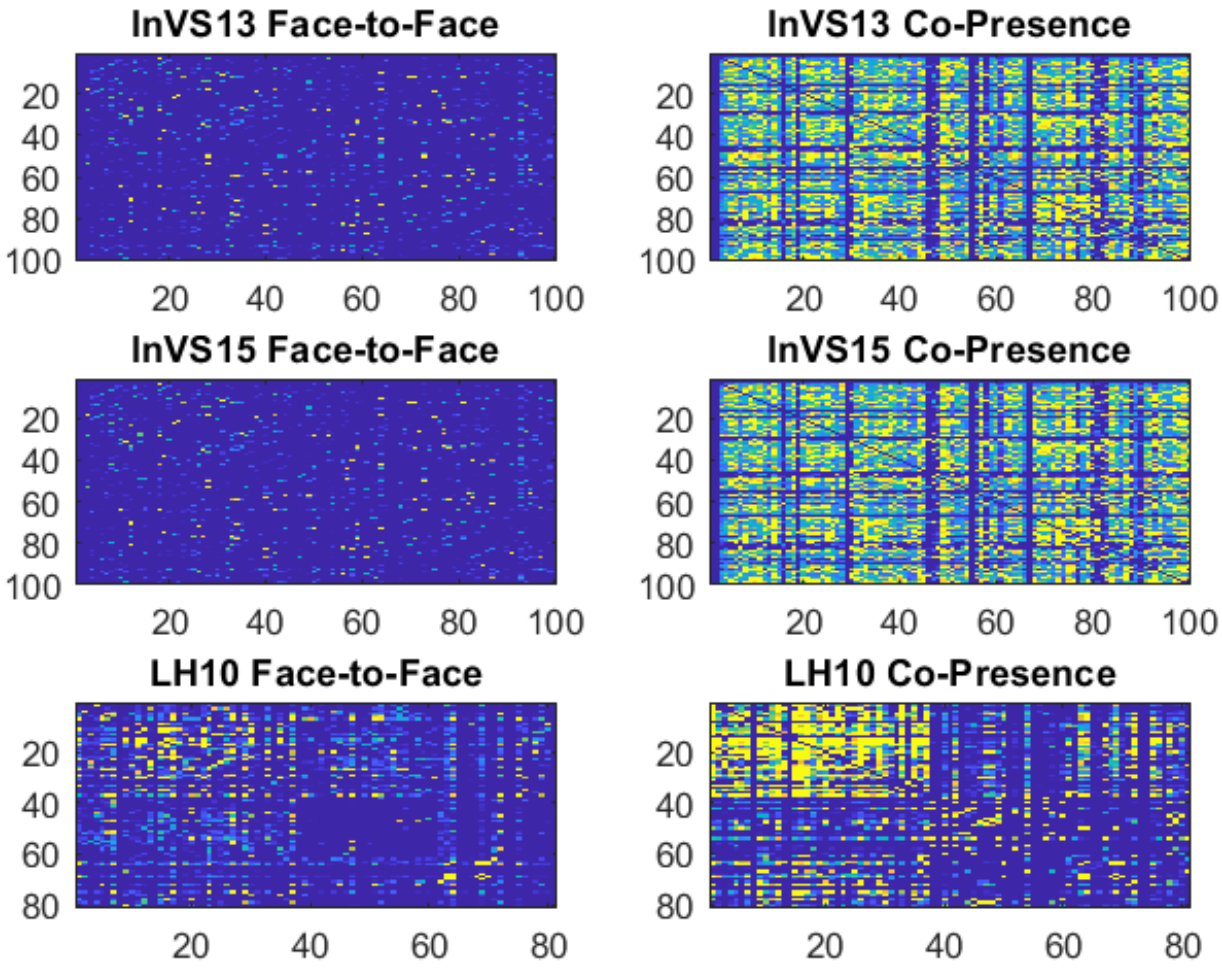
The datasets come from a variety of environments including schools, hospitals, and conferences. Below is a quick summary of the data:

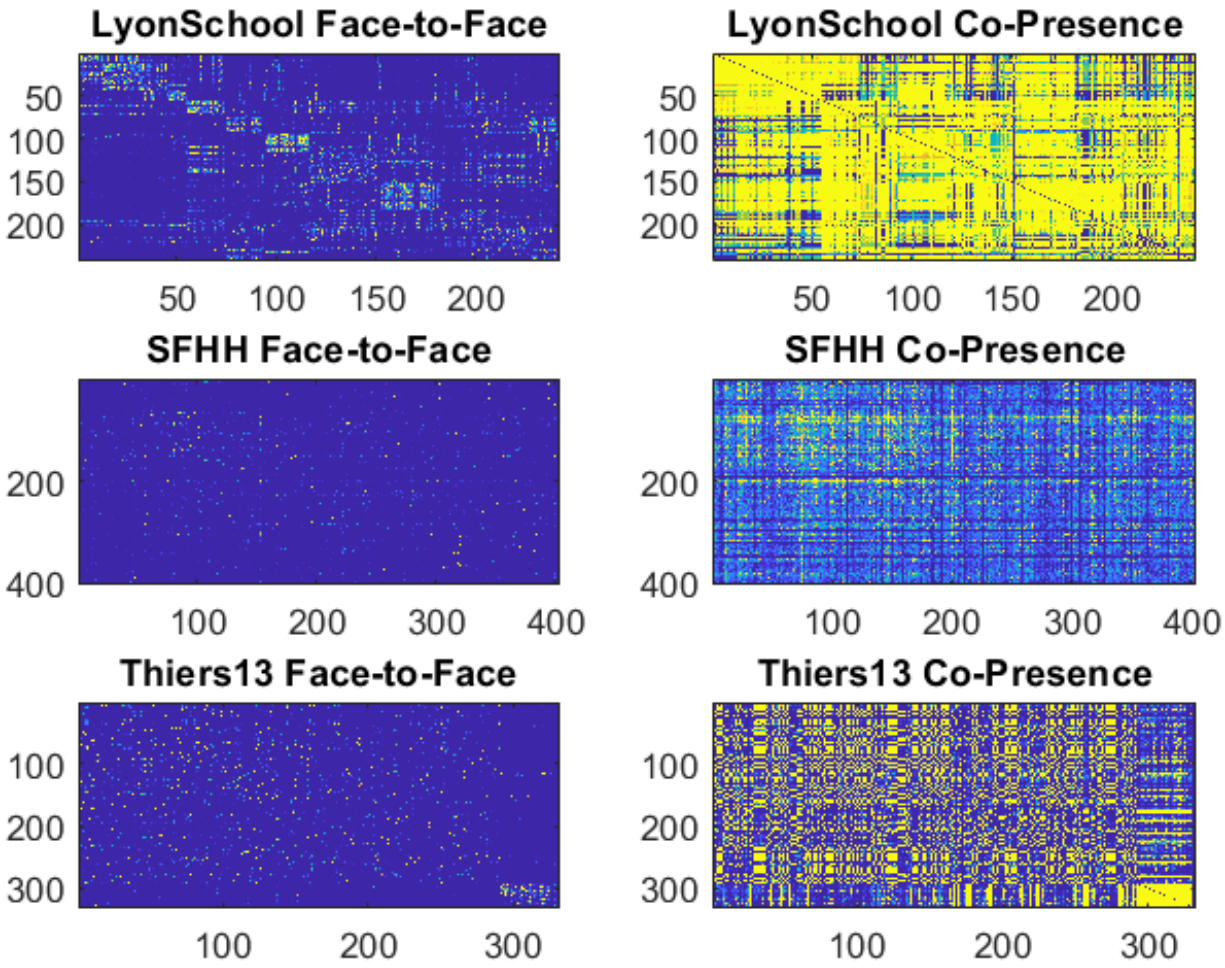
1. **Name:** InVS13 **Location:** French Institute for Public Health Surveillance **Nodes:** 100
2. **Name:** InVS15 **Location:** French Institute for Public Health Surveillance **Nodes:** 232
3. **Name:** LH10 **Location:** Hospital ward (Lyon, France) **Nodes:** 81
4. **Name:** LyonSchool **Location:** Primary school (Lyon, France) **Nodes:** 242
5. **Name:** SFHH **Location:** French Institute for Public Health Surveillance **Nodes:** 92
6. **Name:** Thiers13 **Location:** High School (Marseilles, France) **Nodes:** 326

Part 1

In order to begin deeper analysis on the datasets described above, reformatting of the data is necessary in order to make it easier to manipulate in the coming algorithms and analysis. As is, the data comes as temporal samples over a given period of time, where each row represents a point of contact between node i and node j at time t. The method that is used to condense this data is to form a temporally aggregated adjacency matrix, where each entry $A(u, v)$ in the matrix represents the number of times the two nodes came in contact over the sampling period. In order to accomplish this, a metadata file associated with each pair of datasets is used to re-index all of the nodes from $[1, n]$ due to the fact the given indices in the dataset do not follow a logical order to manipulate. The aggregation can then be completed in a simpler manner. This will result in a condensed nxn weighted matrix representing the strength of connections over time.

The datasets come in 2 forms. The first is the face-to-face dataset, which represents a close proximity interaction between two nodes in the network, such as through direct speech. The other dataset represents co-presence, which is a coarse grained interaction that is nodes in contact over a larger area, such as being in the same room together. The figure below show an image of each pair of adjacency matrices in order to show how the edge weights and edge densites differ between the face-to-face and co-presence datasets:


InVS13 Face-to-Face
InVS13 Co-Presence
InVS15 Face-to-Face
InVS15 Co-Presence
LH10 Face-to-Face
LH10 Co-Presence
20
40
60
80
100



The immediate observations from these plots are that the co-presence matrices are much more dense than their face-to-face counterparts. This result could potentially give much more initial information regarding the dataset, such as how large the environment was the data was taken in, how much collaboration is happening within the environment, and if there are clusters of connections between nodes.

Part 2

As mentioned above the co-presence (CP) graphs have more edge with significantly larger weights than the face-to-face (F2F) graphs. This is easily seen in the images of the adjacency matrices, where the CP image has much brighter and more defined structures than that of the F2F image.

This is due to how the two data sets are gathered. Face-to-face information is gathered only when two people (nodes) are in very close proximity to each other. F2F generally corresponds to direct interaction between two people (nodes). Co-presence data adds edge weights when two people (nodes) are within the same general space. This simply means that the nodes are within a similar space and not that the two people (nodes) are interacting directly. Thus is it is possible to consider the edges/edge weights of F2F data as a subset of the CP data.

For this reason it CP data has significantly larger edge weights between nodes as well as an increased total number of nodes.

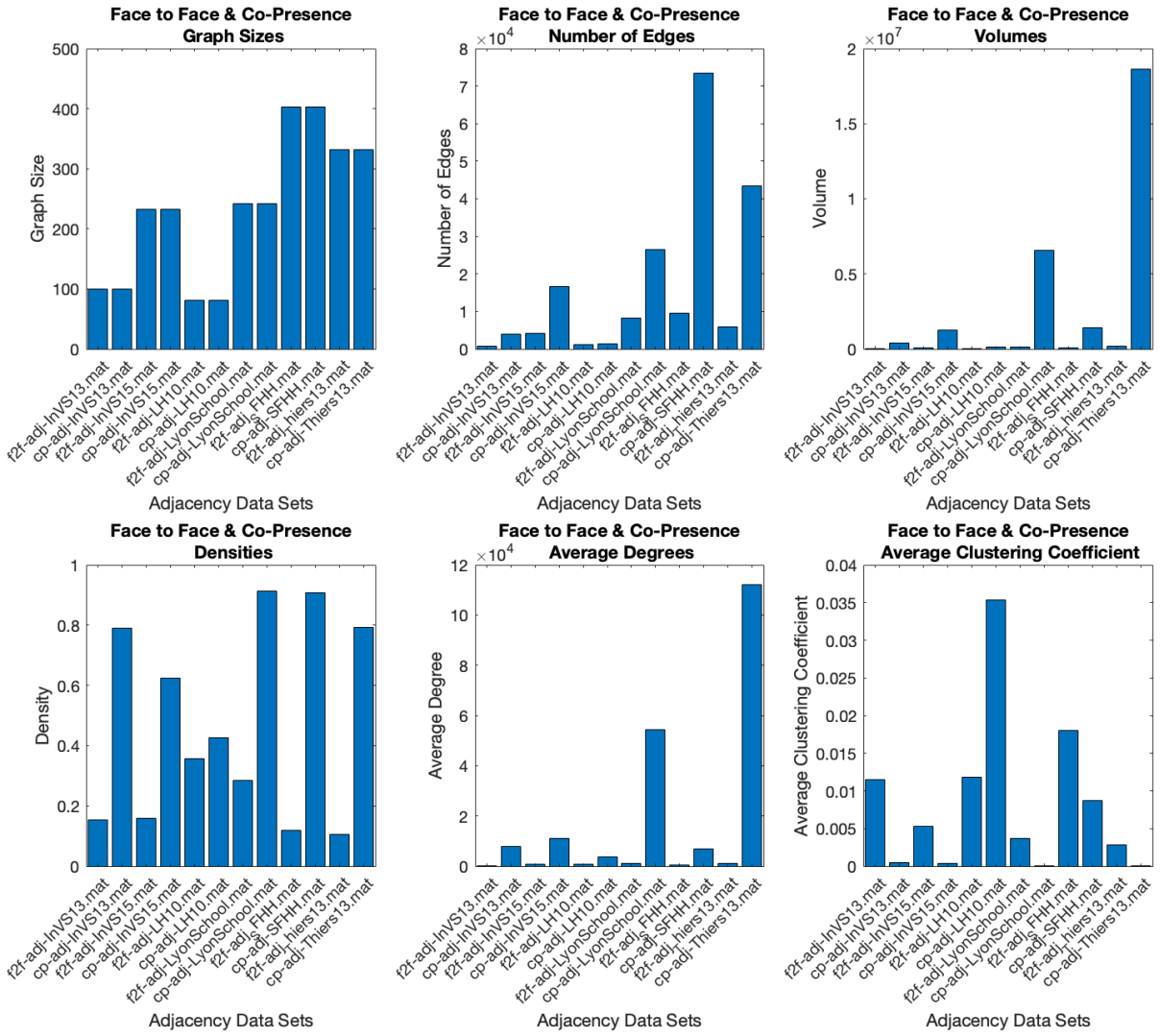
Part 3

To extend the analysis from direct inspection to a numerical representation of the data, 8 statistics were chosen to quantify qualities of the graph:

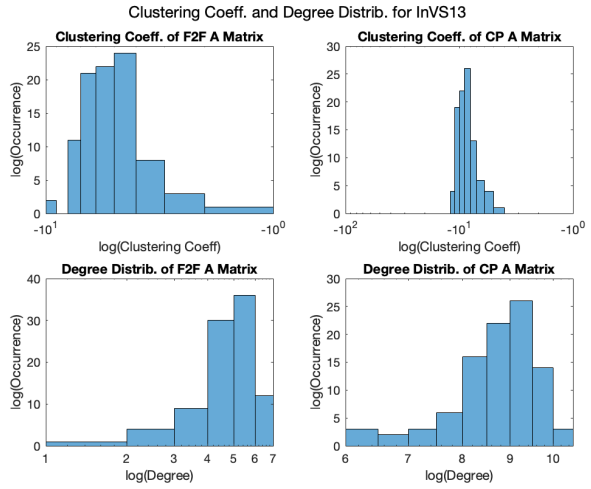
- Graph size: Number of nodes in the graph

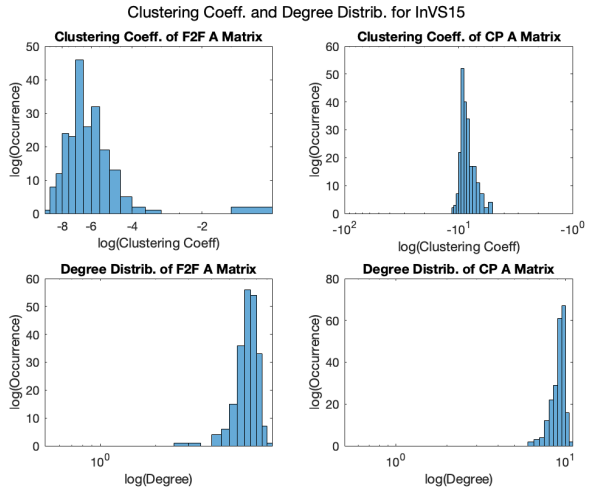
- Number of Edges
- Volume
- Density
- Degree Distribution (vector)
- Average Degree
- Clustering Coefficient (vector)
- Average Clustering Coefficient

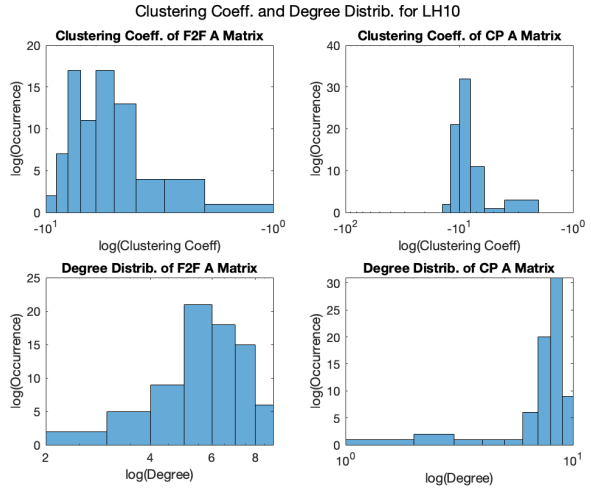
Each of these statistics were gathered for all of the adjacency matrices (face-to-face and co-presence) representing each dataset. The figure below shows the results for all of the scalar quantities across all data sets (12 results per bar graph for 6 scalar quantities):

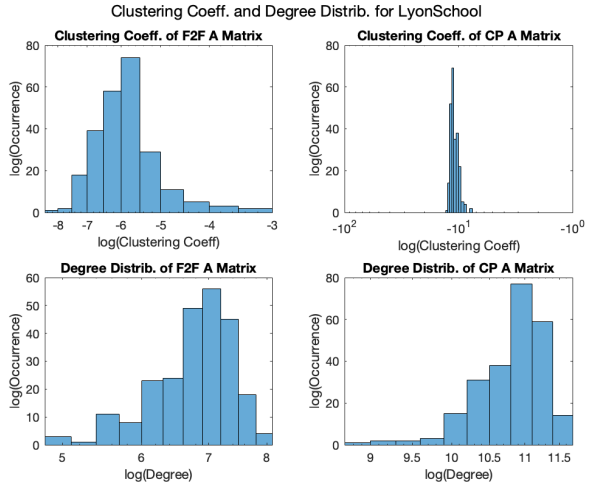


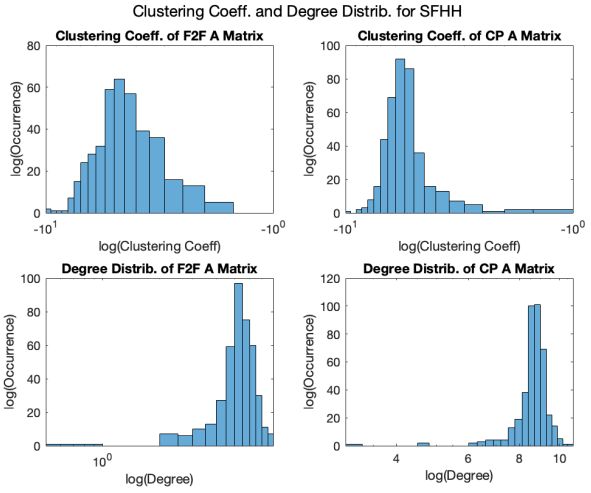
Next, a log-log scale histogram was used to represent the vector quantities of the degree distribution and the clustering coefficient for both the face-to-face and co-presence matrices for each data set. The degree distribution shows how the weights entering a node are concentrated throughout the graph which can insight to social hubs or groups. The clustering coefficient shows how small cliques of nodes are distributed throughout the graph, which also gives a sense of node grouping. Below shows the results of this computation:

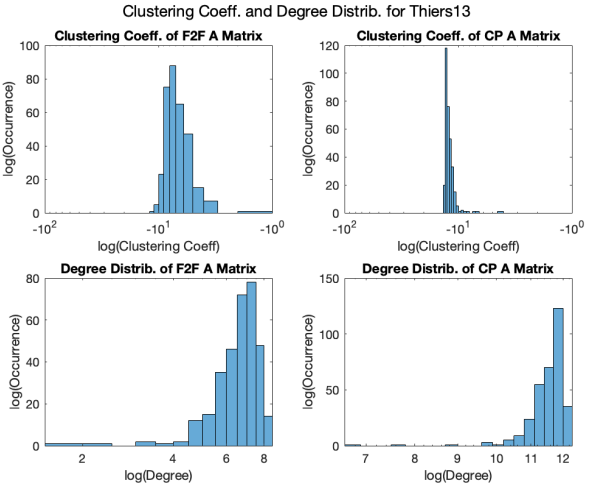












Part 4

Starting with the scalar quantities, each adjacent pair of bars represent the face-to-face and co-presence results side by side for each statistic for easy comparison. The graph size in the case of the unsampled graphs is the same between these matrices due to the face the nodes are not affected by the matrix generation. The number of edges, however, shows a consistent pattern between all of the datasets where the co-presence matrices have a larger number of edges in all cases. This makes sense because as seen in part one, the graphs are much more dense. This metric can also give some insight into the environment the data was taken. For example, the SFHH and Thiers data sets have a substantially larger number of edges in the co-presence compared to the face-to-face, which could mean in these scenarios many of the people were sharing the same

space, but perhaps working independently. The volumes also scale the same way between the datasets where co-presence universally shows higher values, which is a direct result of having more edges. The densities give a representation of how many edges are present compared to the total number of potential edges. This is a metric that can show how complete a graph is. Again, the co-presence is higher in all cases, but a more interesting note is a couple of particular cases. The Lyon Hospital is an interesting case that shows almost equivalent densities between face-to-face and co-presence data, which means that most people in spaces together were interacting together. This could represent large shared patient areas like and emergency room, where several doctors and nurses are making rounds to all of the patients in the room, for example. Another note is how close the Lyon School and SFHH densities are to 1, which mean they are quite close to being complete graphs. This could mean these environments are very collaborative. The average degree value follows the consistent trend of higher values for co-presence and shows that for smaller values, the social groups (or social epicenters) are smaller in datasets like the hospital and lnVS13/15, where people have interactions with some people, but not everyone. In the case of Lyon School and Thiers however, the groups are much more substantial. The final scalar metric is the average clustering coefficient. This value represents transitivity of edges, measuring the relative amount to 3 node cliques(triangles) within the graph. This can give a sense of communities and how nodes are clustered within the graph. Interestingly, all but one dataset exhibit larger values for the face-to-face contact matrices. This means that in the case of recognizing groups of people that are interacting directly, the face-to-face data is much more valuable and the co-presence is too coarse grained to recognize this. In the case of the Lyon Hospital, the clustering coefficient is larger for co-presence, which supports the theory that perhaps patients are organized side-by-side, for example, and a doctor or nurse comes through to make rounds, creating the "triangle" connection between neighboring nodes, but the patients themselves are not interacting with many other people. Overall, these simple statistics give quite a bit of information, not only about general characteristics of the graph, but about the sampling environments they were taken in.

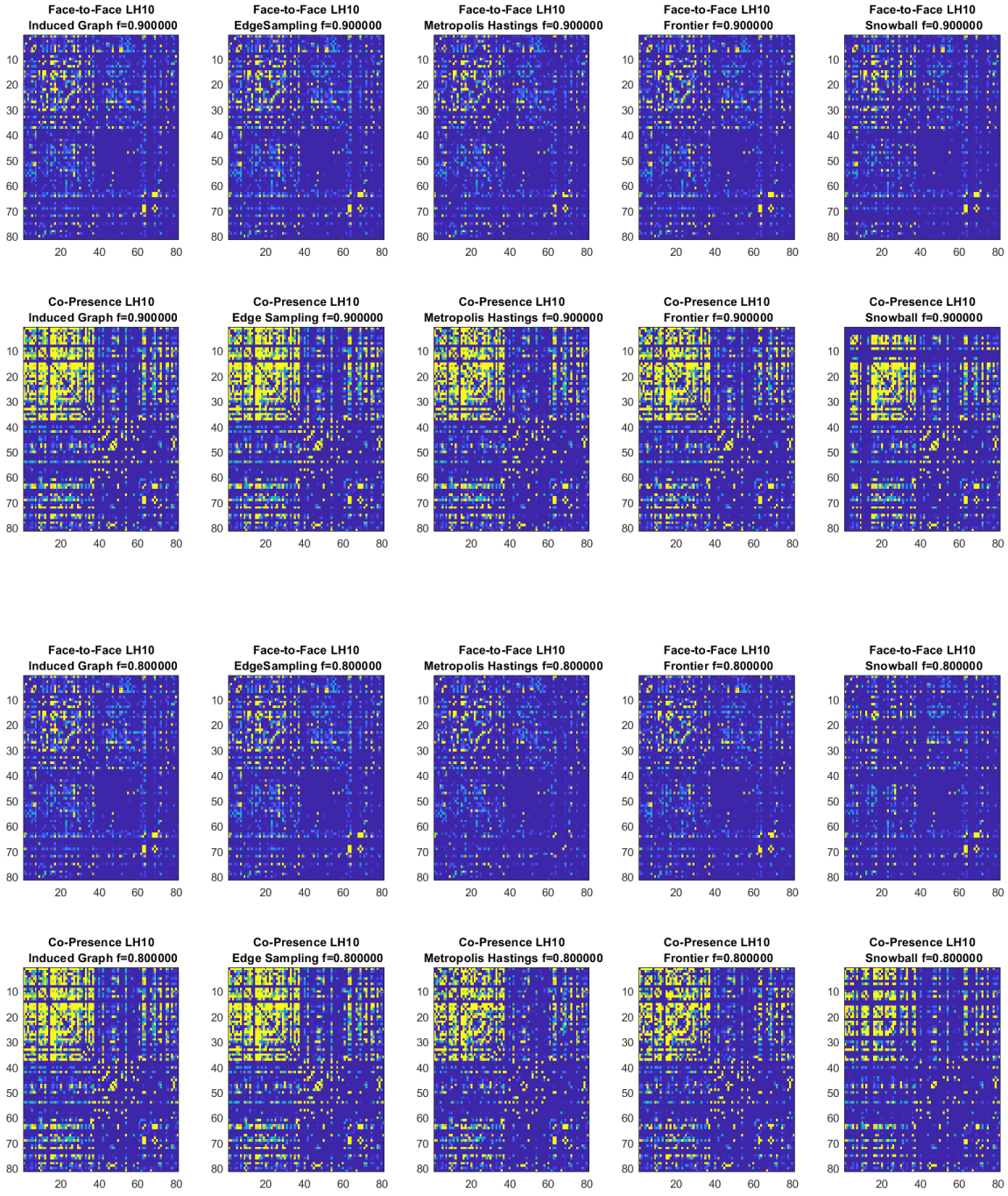
In comparing the vector quantities looking at how the distribution is spread as well as the value it is centered around relative to other datasets provides information regarding how a sampled space is organize, and how large the groups are within those spaces, if any exist. In terms of the degree distribution, a wide distribution would indicate that many nodes have similar numbers of neighbors, and the magnitude shows how large these neighbor hoods are. It does not show if these nodes are a part of the same network though. A thin distribution would indicate that only a small portion of nodes have many neighbors, and perhaps the others are only parts of small groups. In general when comparing the degree distributions between the face-to-face and co-presence matrices, the value that the distribution is centered over is larger for the co-presence, which makes sense due to nodes now being connected to every other node in their general area, so the average degree increases. The distribution on average too get a bit wider, due to the fact that now even the nodes that were in small groups in close proximity are now part of the larger community. An interesting case to look at is the Lyon Hospital example for the co-presence where the distribution is quite sharp, which indicates that at this resolution of sampling, only a few nodes have high degree, so perhaps these nodes travel between isolated areas on a frequent basis, such as a doctor or nurse. The other distribution to analyze is the clustering coefficient. Between the face-to-face and the co-presence, the center of the distribution is generally smaller for the co-presence which shows that at that coarse of a resolution the localized groups decreases due to the increasing overall degree within the graph (denominator of the clustering coefficient equation). Without looking at the average value, however, many of these distributions look similar, so they scale with the increasing degree. One interesting result from these histograms is looking at the Thiers dataset which show a relatively typical distribution for the face-to-face, but the co-presence distribution has a high number of coefficients with the same coefficient. This exposed the feature of isolation in the graph, where perhaps the data was taken in many smaller isolated spaces with just a few people in each. Referring back to the image of the adjacency matrix for this dataset, it looks like many small and divided areas of connections which would support this hypothesis.

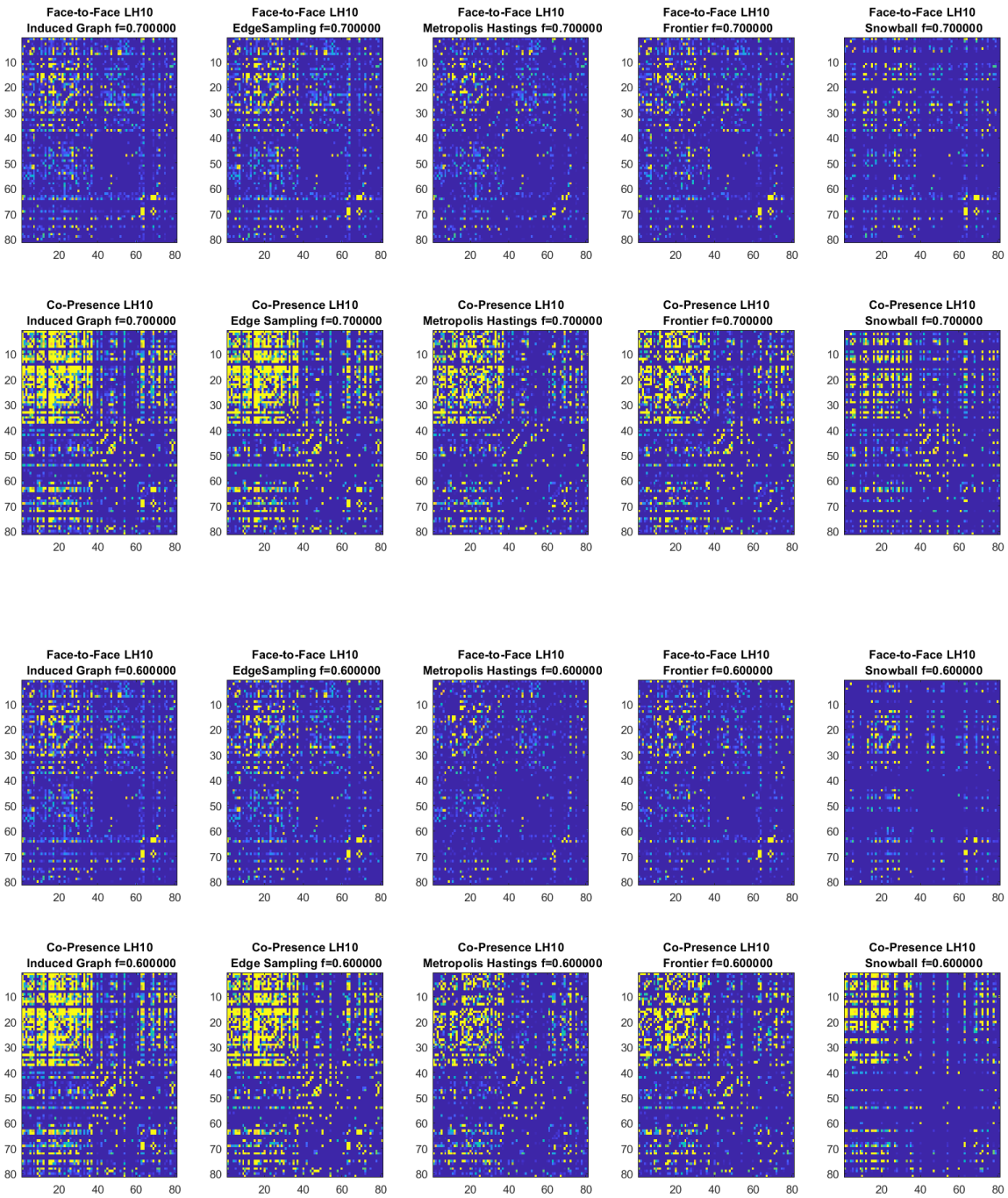
Part 5

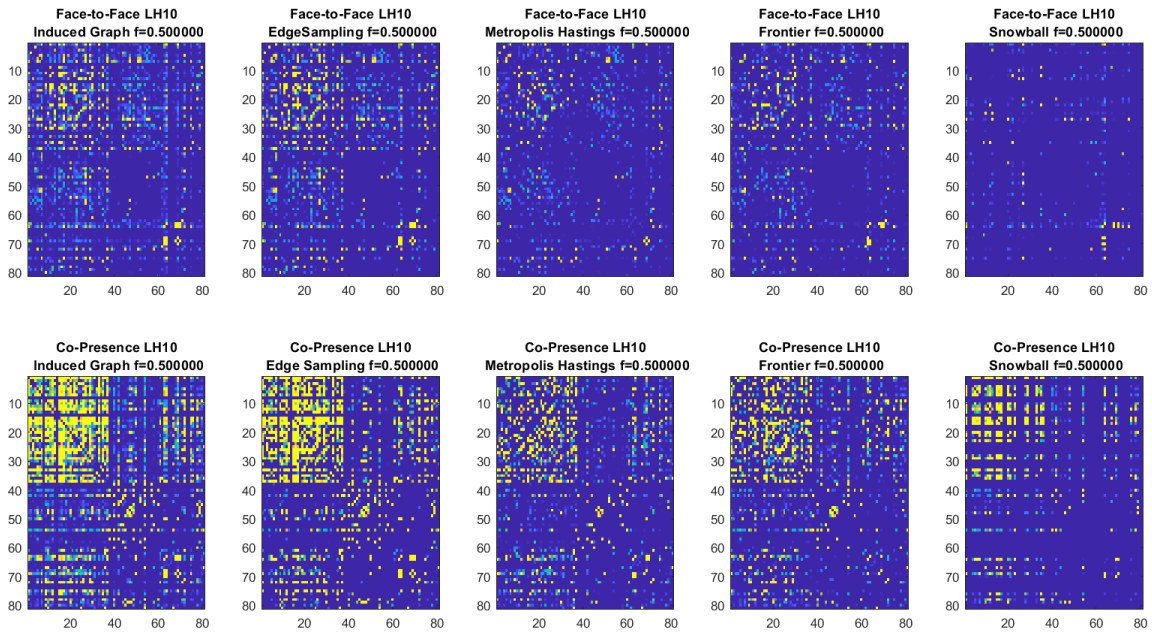
The next step in evaluating the data is to explore methods to reduce the data so that it is more manageable for large datasets, effectively reducing the size of the adjacency matrix. This is done by reconstructing the adjacency matrix by randomly subsampling the original matrix to build a sparser matrix that will hopefully hold the same information. 5 algorithms will be evaluated:

- Induced Graph Sampling
- Edge Sampling
- Metropolis Hastings Random Walk
- Frontier Sampling
- Snowball Expansion Sampling

In order to evaluate the relative performance for each of the 5 algorithms, each algorithm was run with each of the datasets at 5 different sub-sampling ratios $f = .9, .8, .7, .6, .5$. The output below shows the affect on the adjacency matrices from running both of the LH10 datasets through each algorithm at the 5 sampling ratios:




Face-to-Face LH10
Induced Graph f=0.700000
Face-to-Face LH10
EdgeSampling f=0.700000
Face-to-Face LH10
Metropolis Hastings f=0.700000
Face-to-Face LH10
Frontier f=0.700000
Face-to-Face LH10
Snowball f=0.700000
Co-Presence LH10
Induced Graph f=0.700000
Co-Presence LH10
Edge Sampling f=0.700000
Co-Presence LH10
Metropolis Hastings f=0.700000
Co-Presence LH10
Frontier f=0.700000
Co-Presence LH10
Snowball f=0.700000
Face-to-Face LH10
Induced Graph f=0.600000
Face-to-Face LH10
EdgeSampling f=0.600000
Face-to-Face LH10
Metropolis Hastings f=0.600000
Face-to-Face LH10
Frontier f=0.600000
Face-to-Face LH10
Snowball f=0.600000
Co-Presence LH10
Induced Graph f=0.600000
Co-Presence LH10
Edge Sampling f=0.600000
Co-Presence LH10
Metropolis Hastings f=0.600000
Co-Presence LH10
Frontier f=0.600000
Co-Presence LH10
Snowball f=0.600000
10
20
30
40
50
60
70
80
20
40
60
80

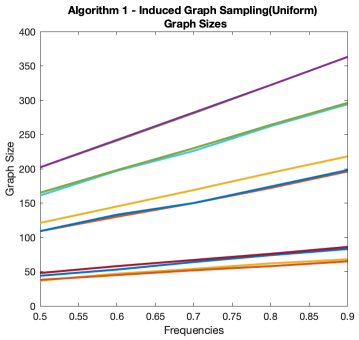


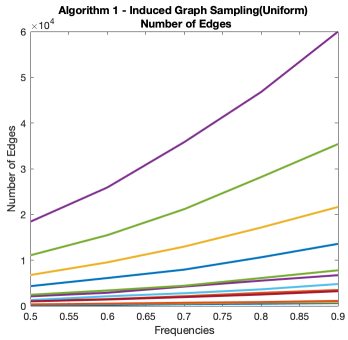
This result can first be used to verify the correct execution of each of the algorithms by observing that the relative shape of the matrix image after running each of the algorithms is roughly the same, but sparser. This confirms that the algorithms are correctly using the probability distributions from the original graph to randomly sample new edges or vertices.

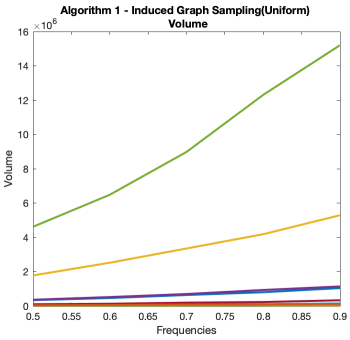
Algorithm's Statistical Evaluation

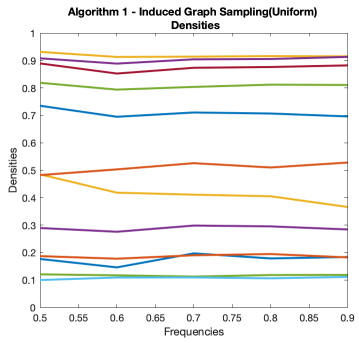
With the sub-sampled generated graphs confirmed, we can evaluate the 8 statistics that have been covered in the previous section, (number of vertex, number of edges, volume, density, degree distribution, clustering coefficient, average degree, and average clustering coefficient,) on all the generated graphs from the 5 algorithms. Instead of using a table, we decided to look at the trend of each stat for every data set as a function of the frequency because this provided a much more visually meaningful representation of the the statistics. Then we compare each trend to each other and look at the growth as the frequency changes. We should expect that all the stats will be lower from the original graph as we reduce the number of edges and vertices. As the frequencies increase, we should expect higher densities of our graph as more nodes/edges are being included. A frequency of 1 would essentially be the original graph because we're including every vertex/edge.

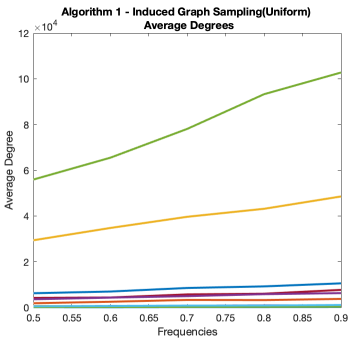
Algorithm 1: Induced Sampling Scalar Statistics

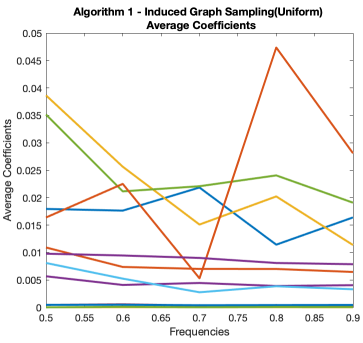


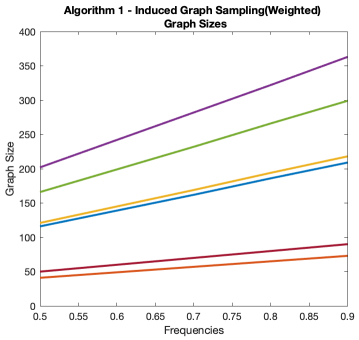


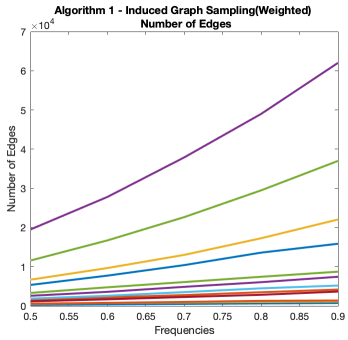


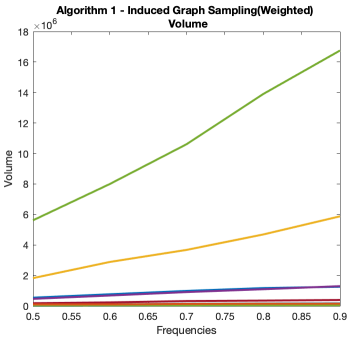


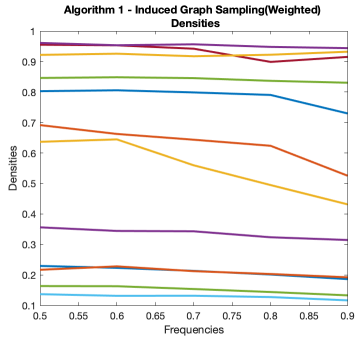


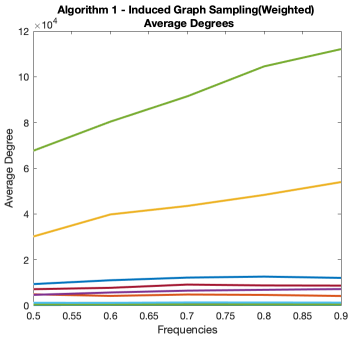


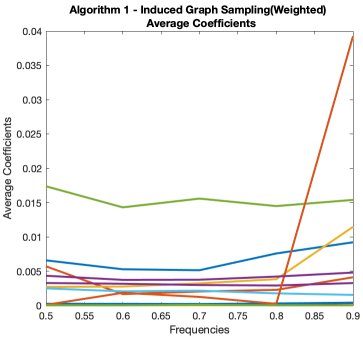


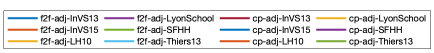




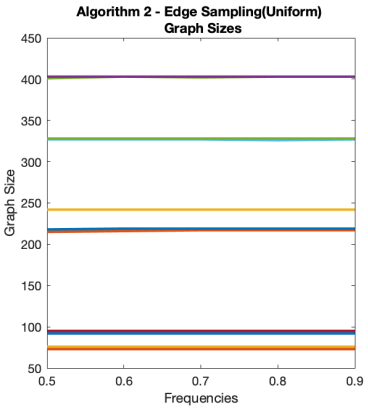


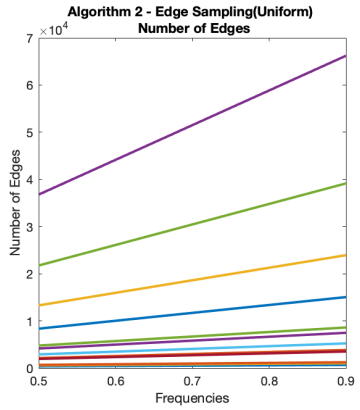


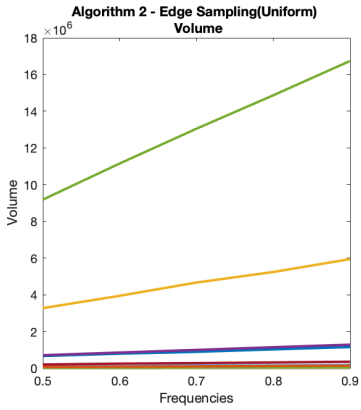


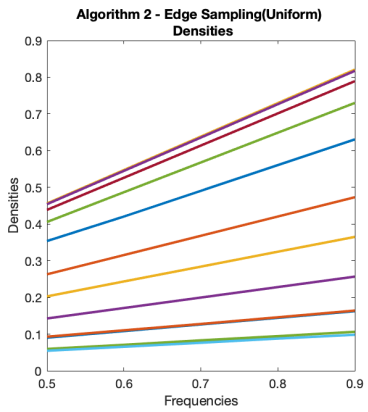


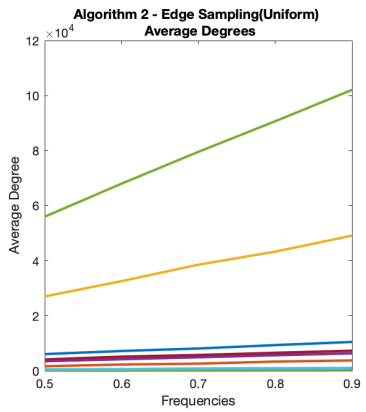
Algorithm 2: Edge Sampling Scalar Statistics

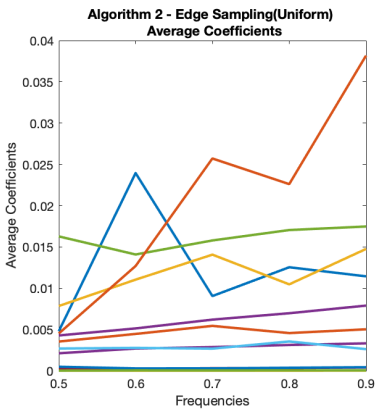


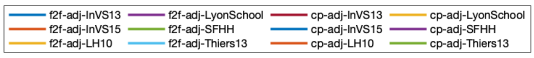


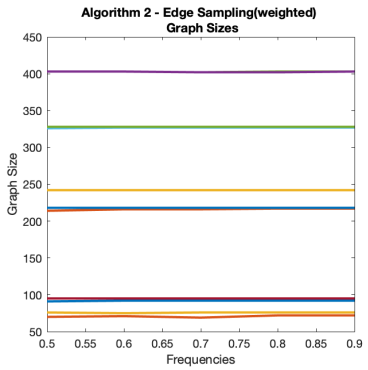


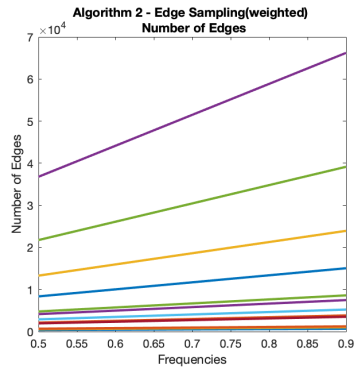


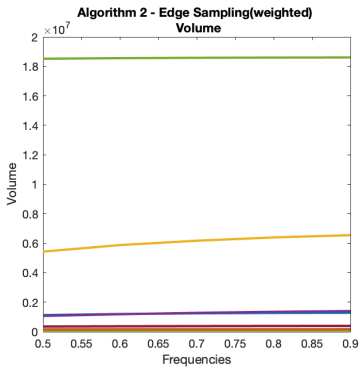


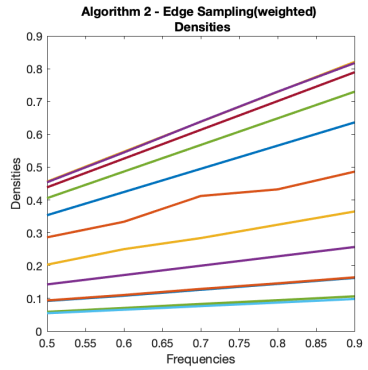


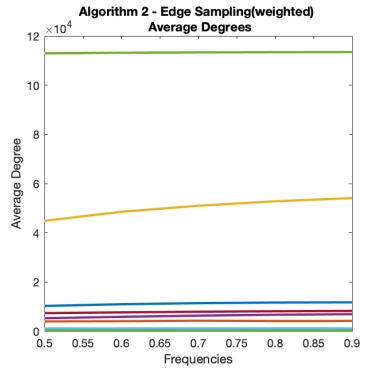


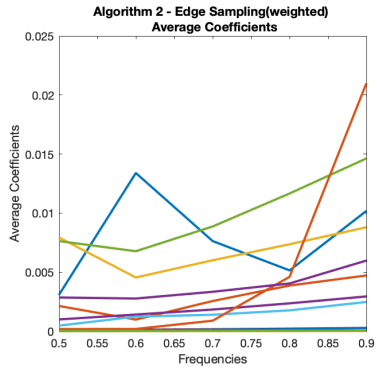




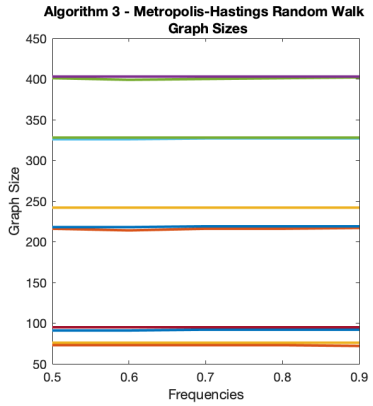


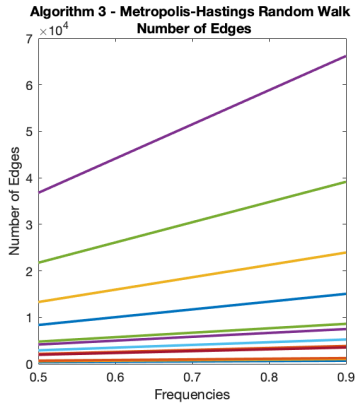


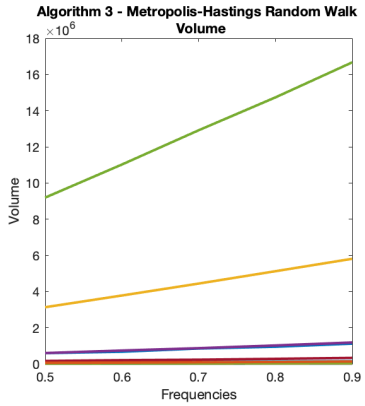


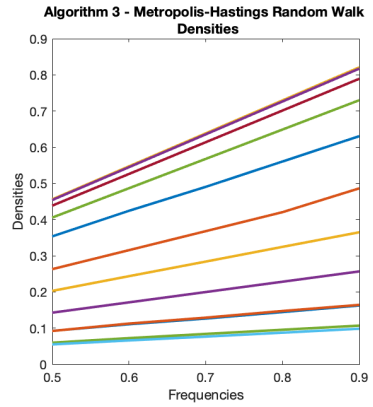


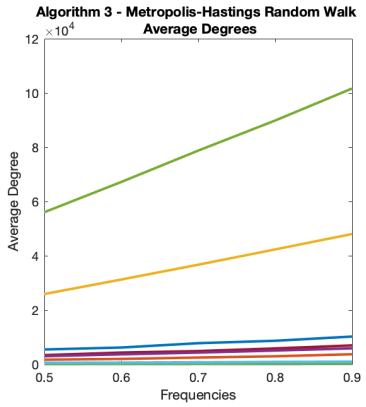
Algorithm 3: Metropolis Hastings Scalar Statistics

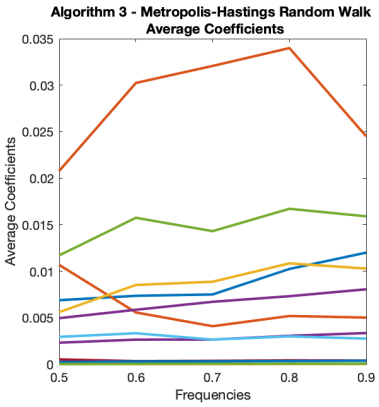


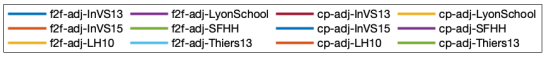




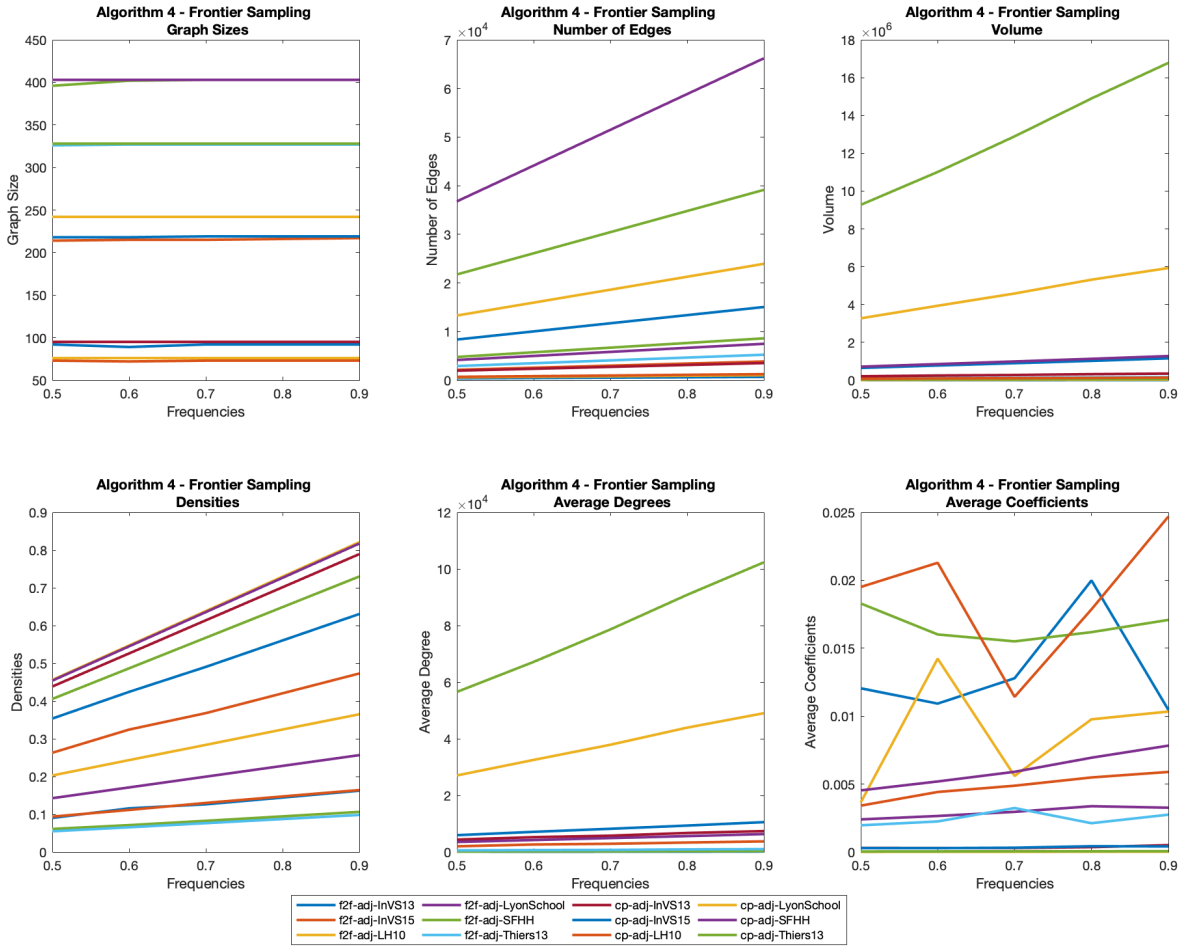




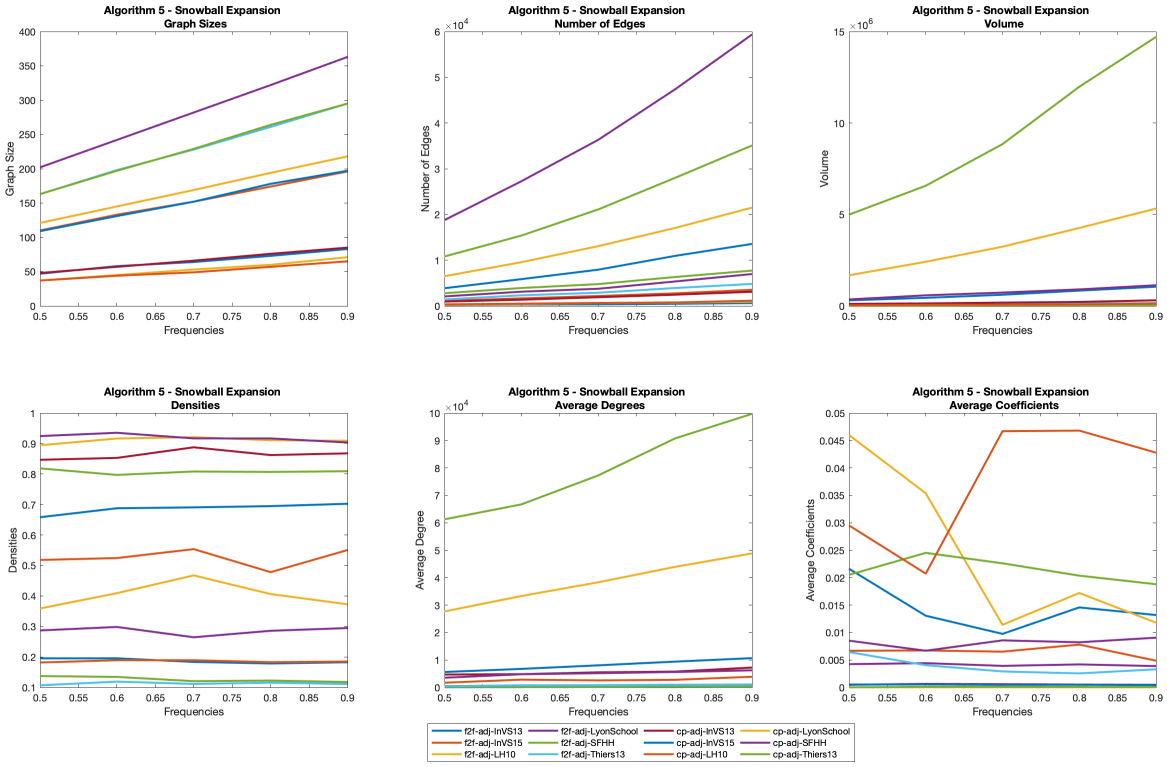




Algorithm 4: Frontier Scalar Statistics

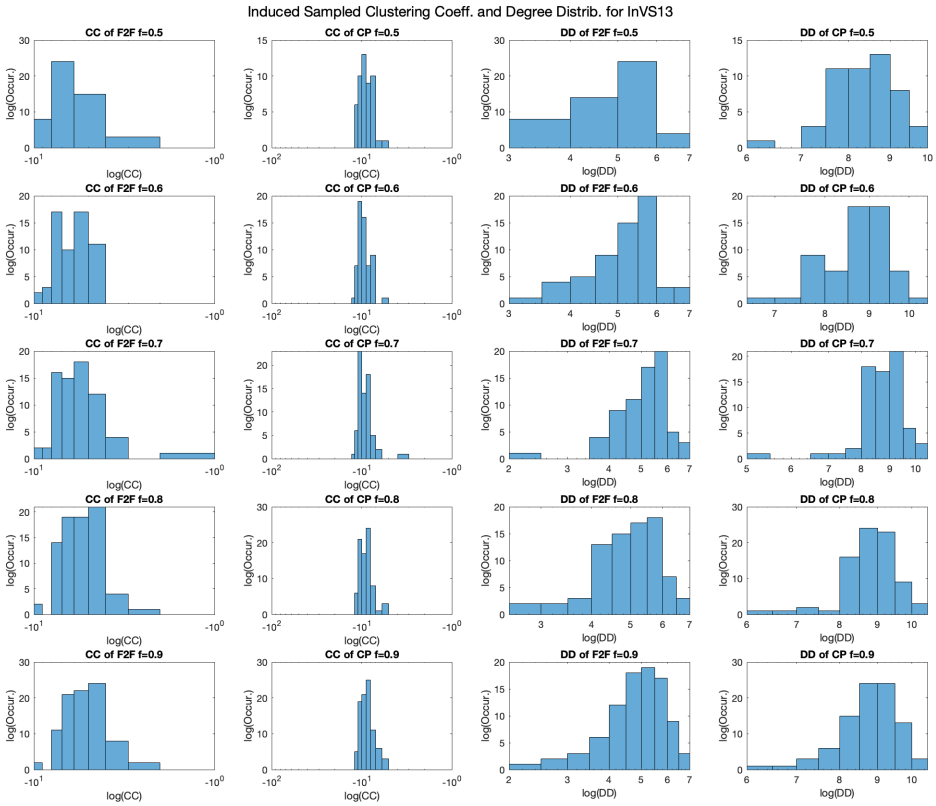


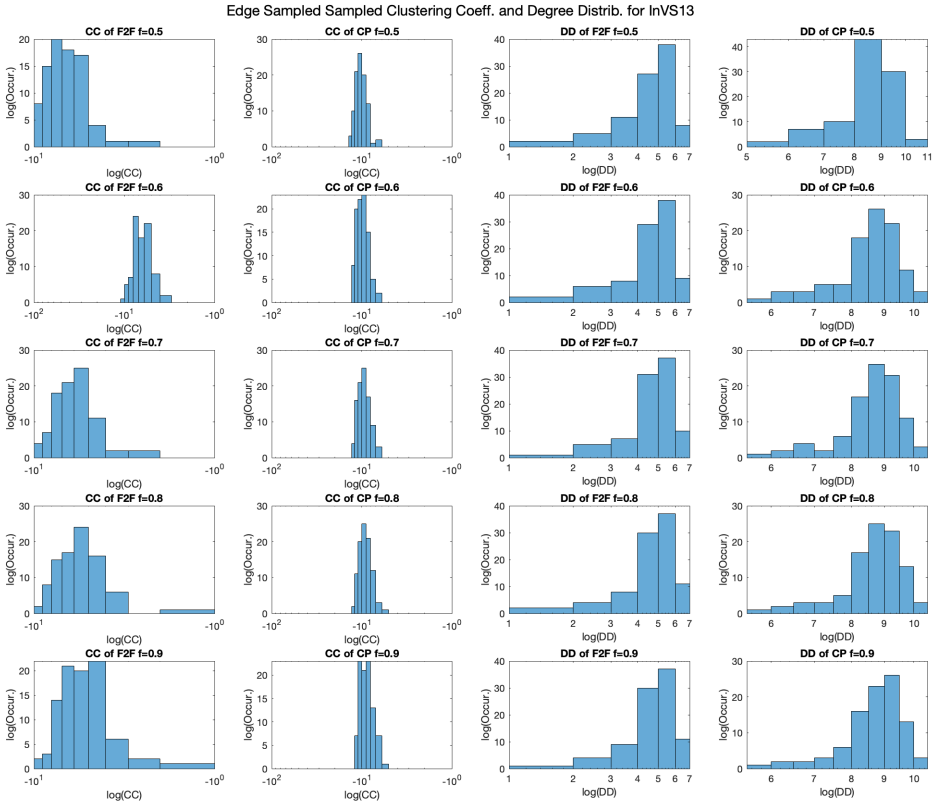
Algorithm 5: Snowball Scalar Statistics

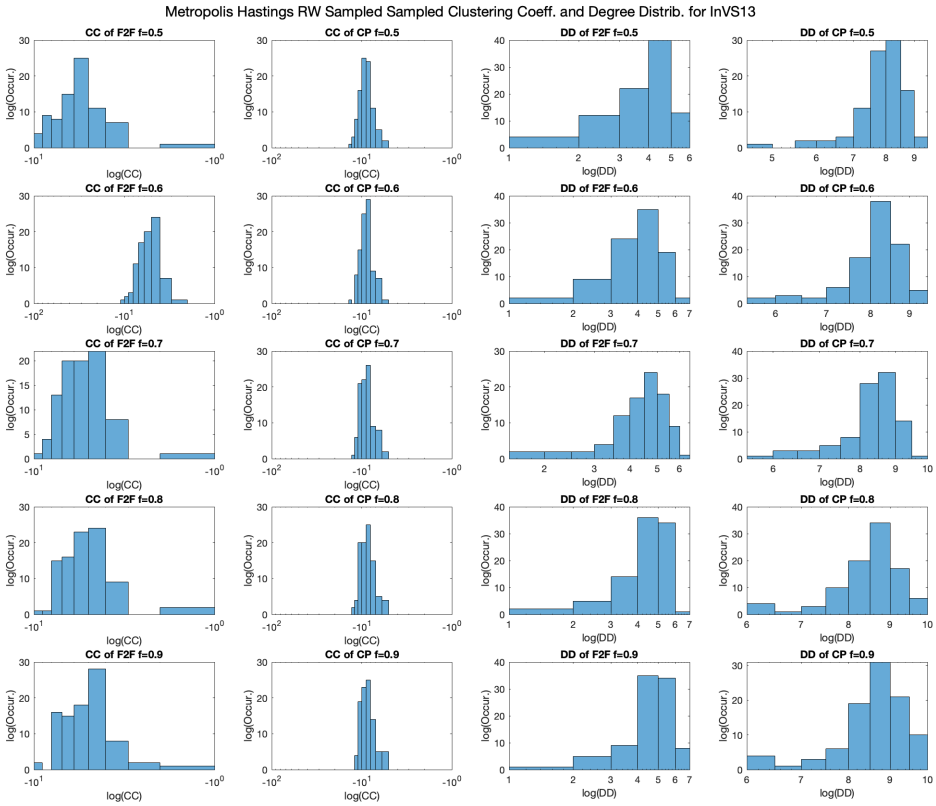


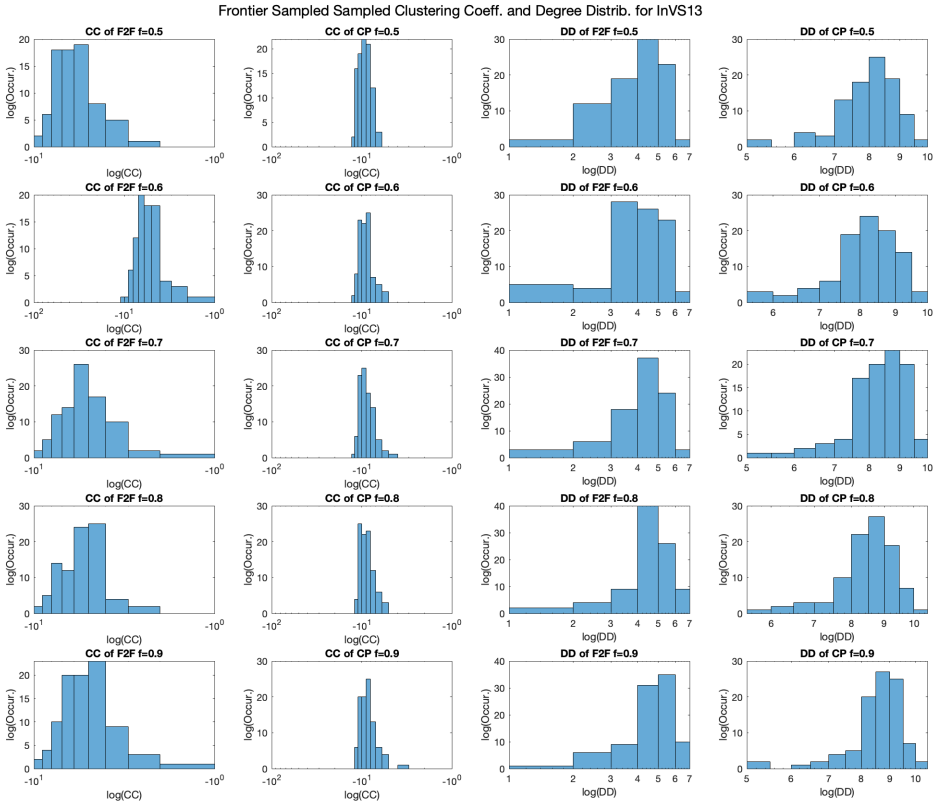
The following are the histograms of the clustering coefficient and the degree distribution for the sampled graphs. The plots are broken into 6 groups by dataset. For each dataset there exists 5 plots each with 20 subplots. Each of the 5 plots correspond to one of the 5 sampling algorithms (induced sampling, edge sampling, metropolis Hastings random walk, frontier sampling, and snow ball expansion). Each row corresponds to a sampling ratio f (0.5, 0.6, 0.7, 0.8, 0.9). In each row the first two figures shows the log-log plot of the clustering coefficient and the degree distribution for the sampled face-to-face and co-presence graphs respectively. The second two figures shows the log-log plot of the degree distribution and the degree distribution for the sampled face-to-face and co-presence graphs respectively.

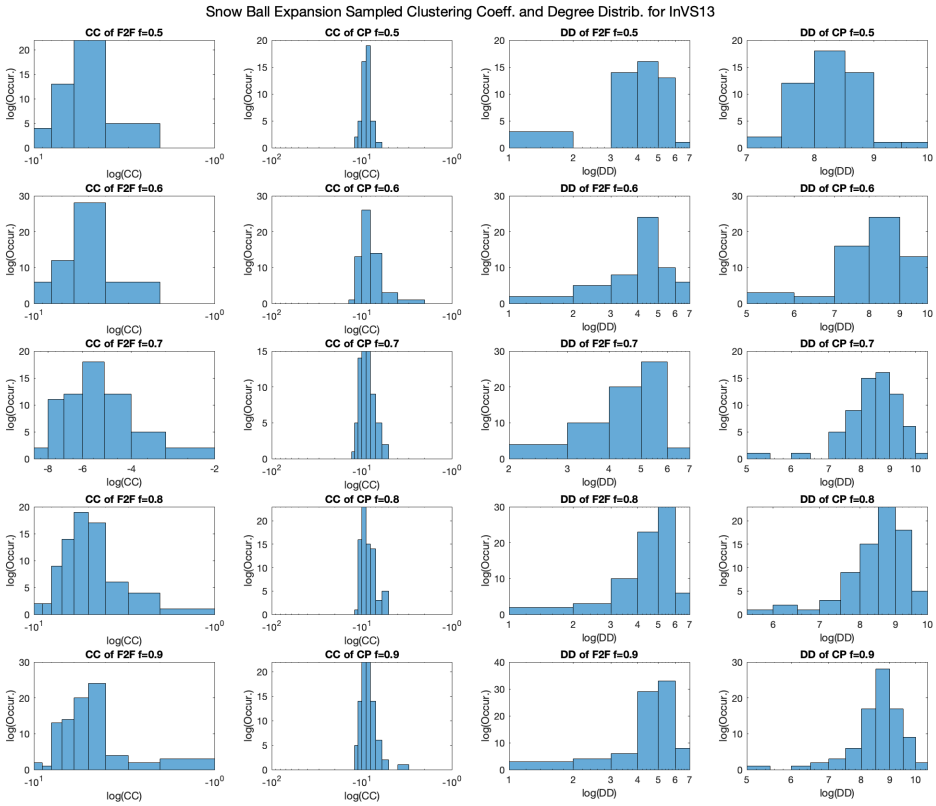
Dataset 1 Sampled Graph Histograms



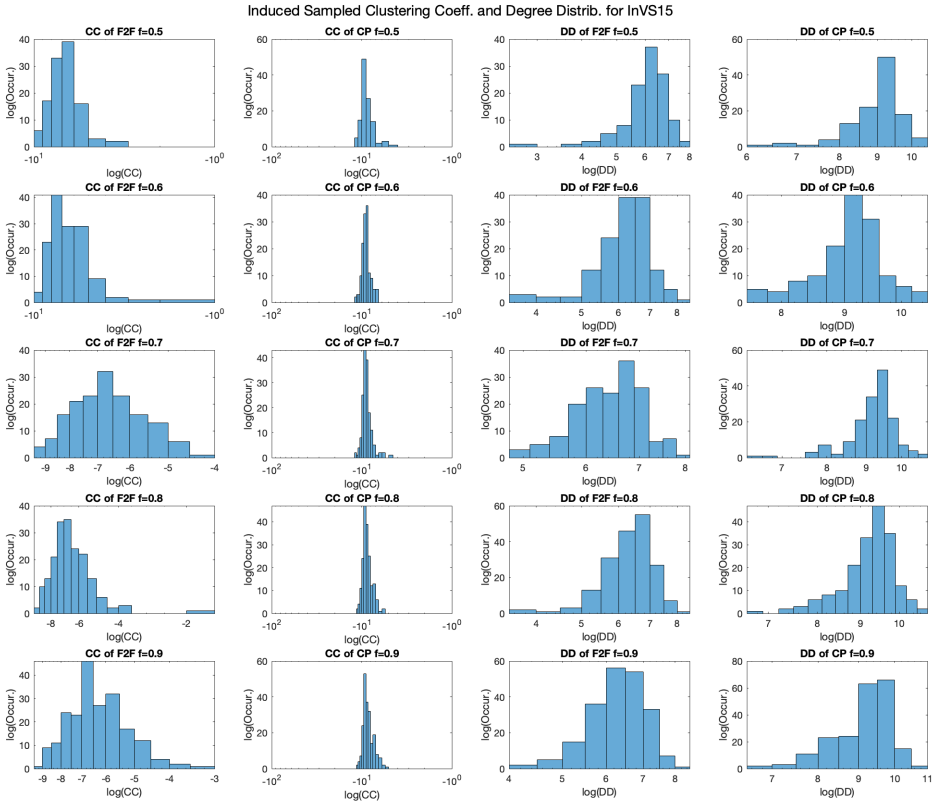


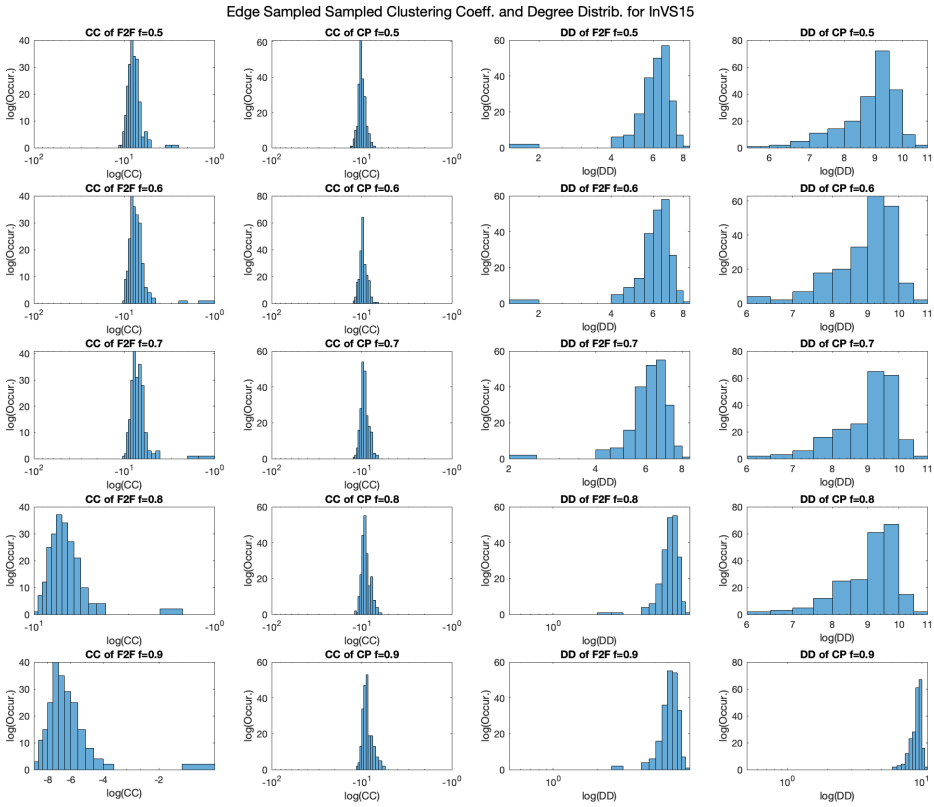

Metropolis Hastings RW Sampled Sampled Clustering Coeff. and Degree Distrib. for InVS13
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

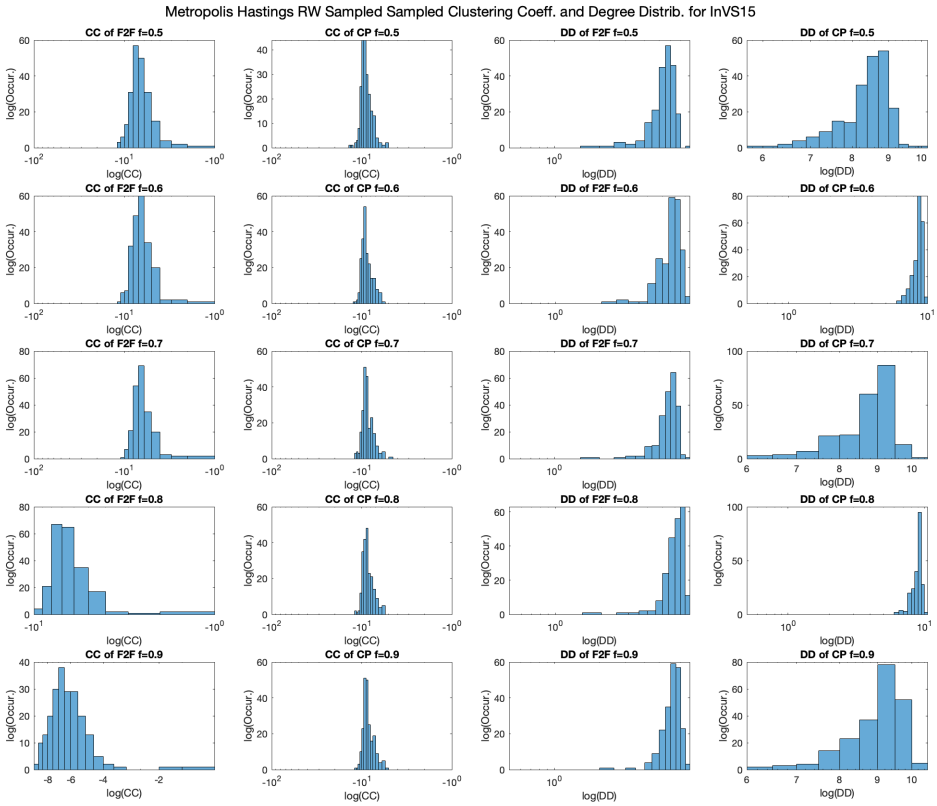

Frontier Sampled Sampled Clustering Coeff. and Degree Distrib. for InVS13
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

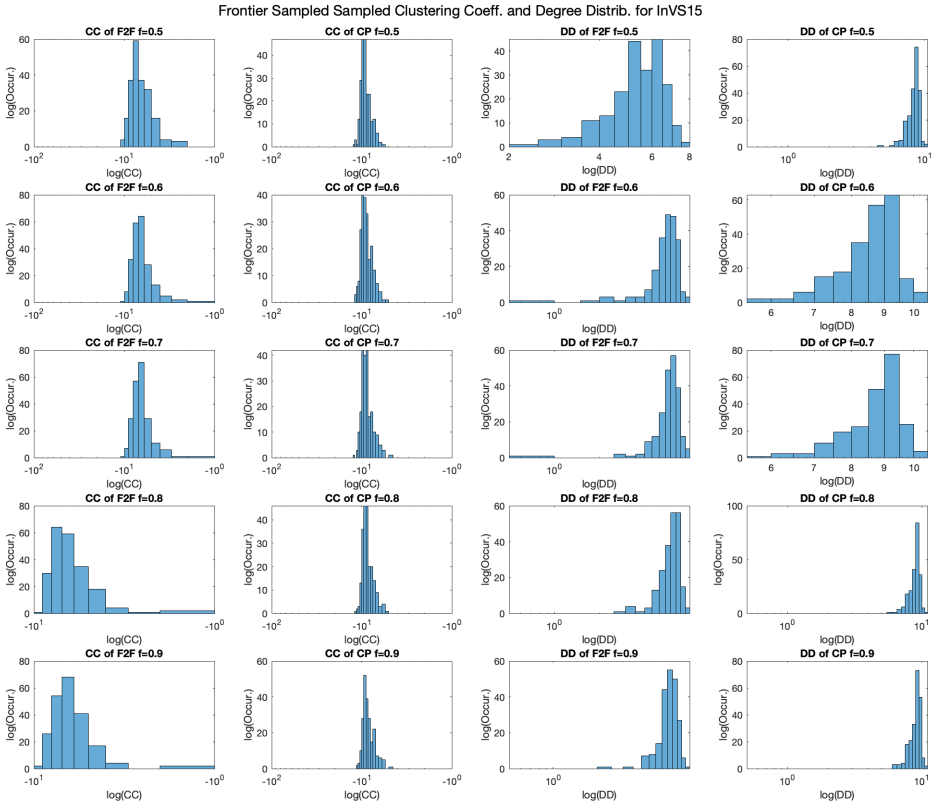

Snow Ball Expansion Sampled Clustering Coeff. and Degree Distrib. for InVS13
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

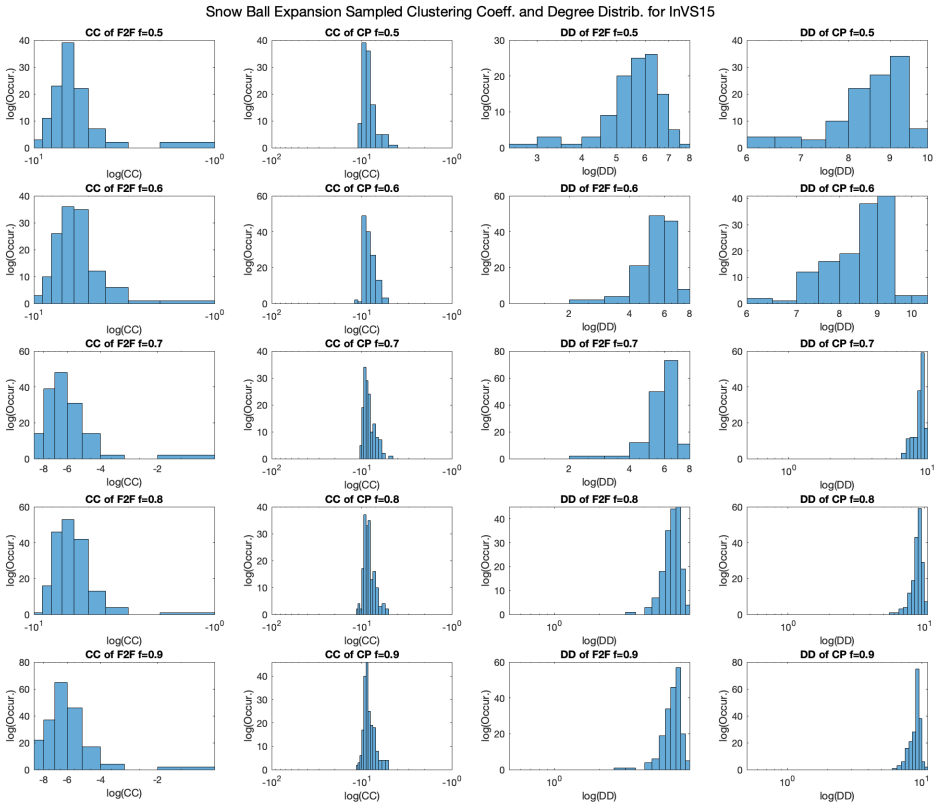
Dataset 2 Sampled Graph Histograms



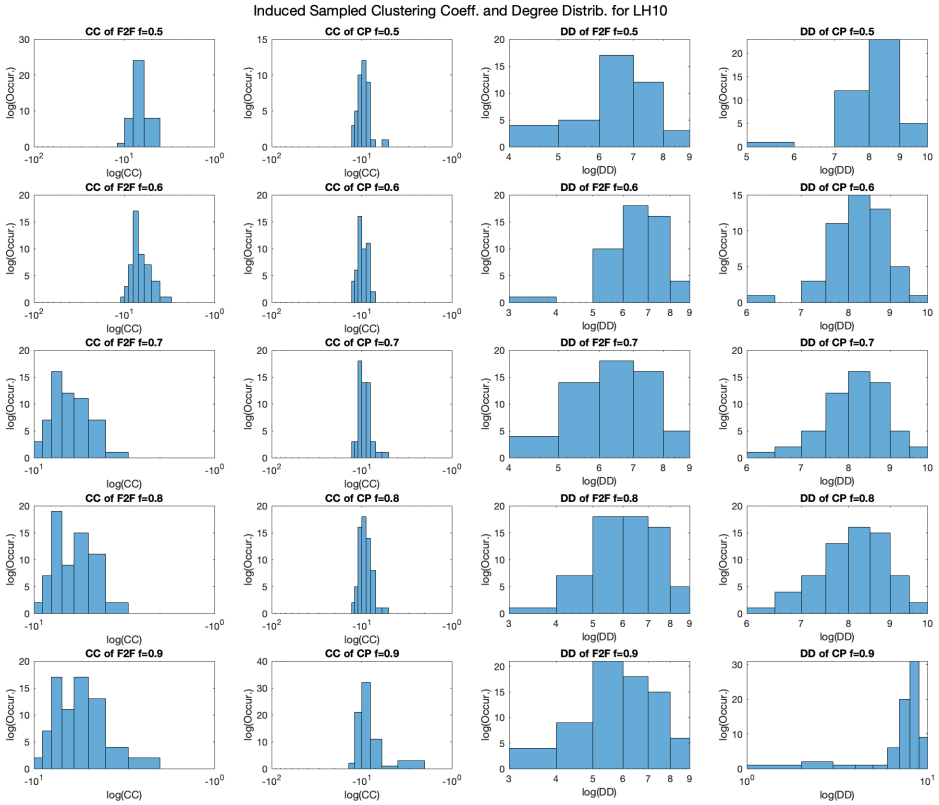


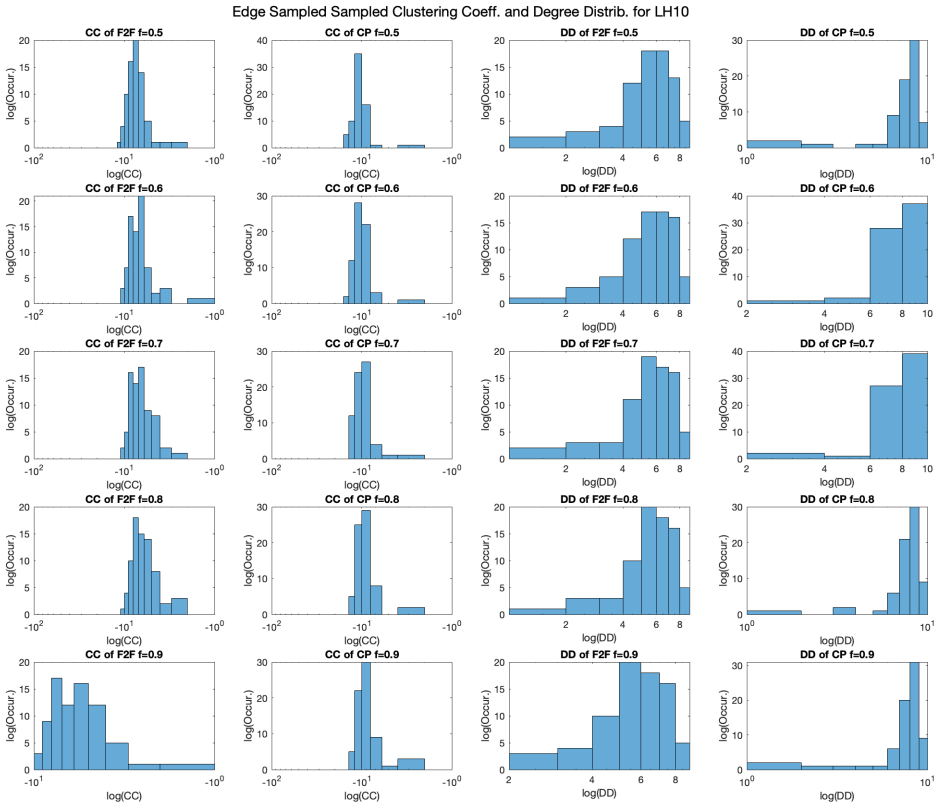

Metropolis Hastings RW Sampled Sampled Clustering Coeff. and Degree Distrib. for InVS15
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

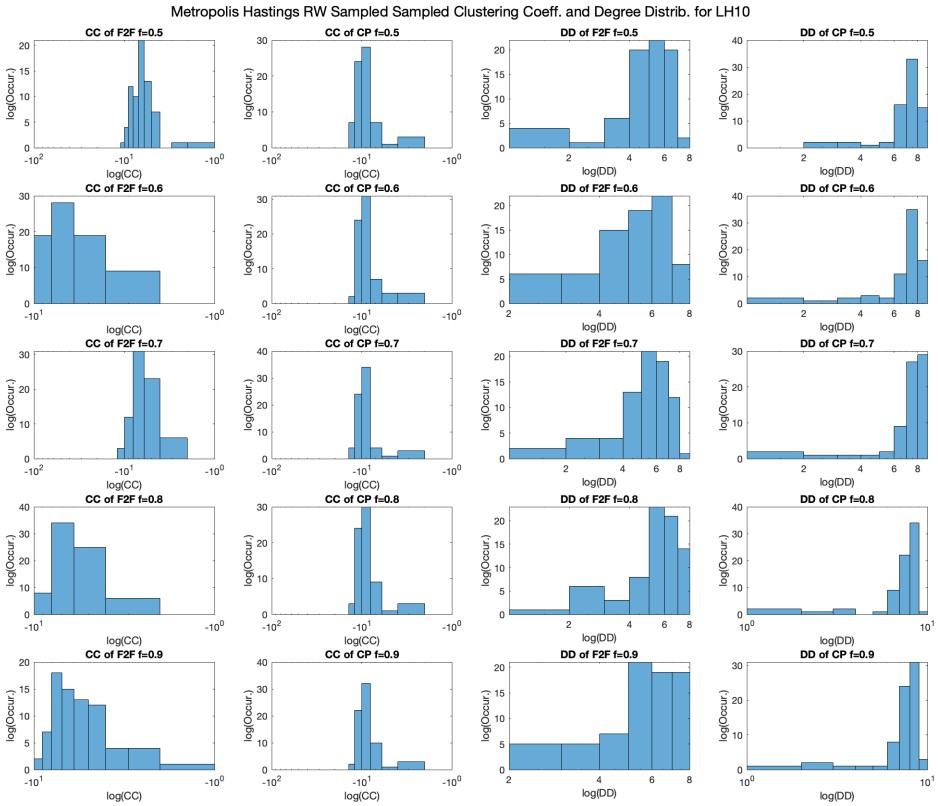

Frontier Sampled Sampled Clustering Coeff. and Degree Distrib. for InVS15
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

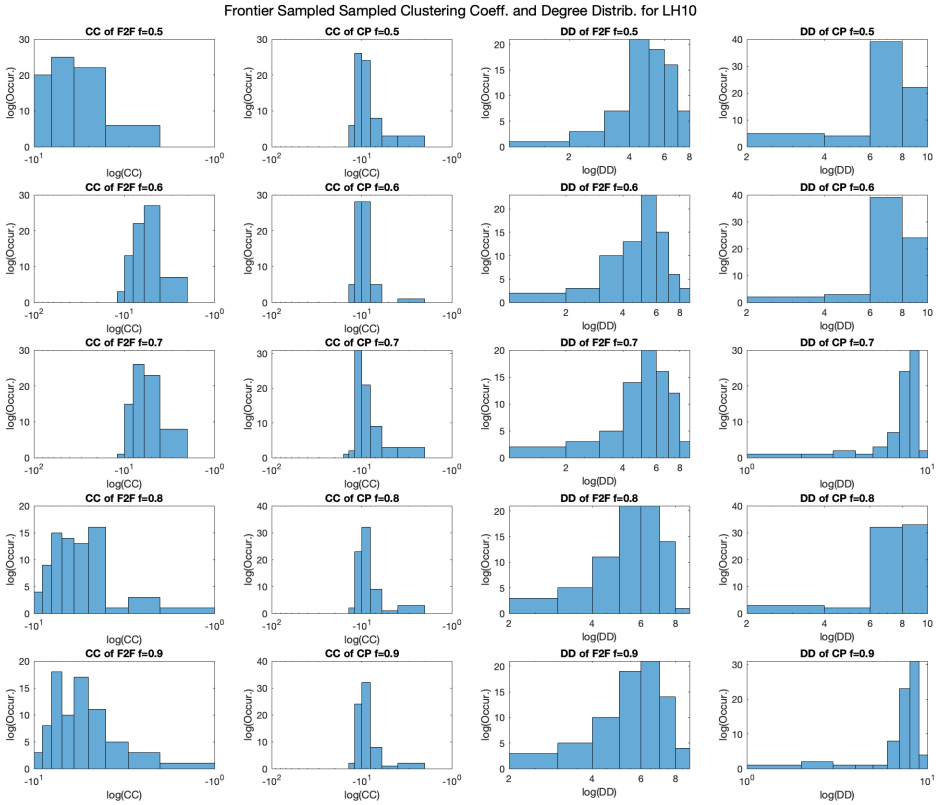

Snow Ball Expansion Sampled Clustering Coeff. and Degree Distrib. for InVS15
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

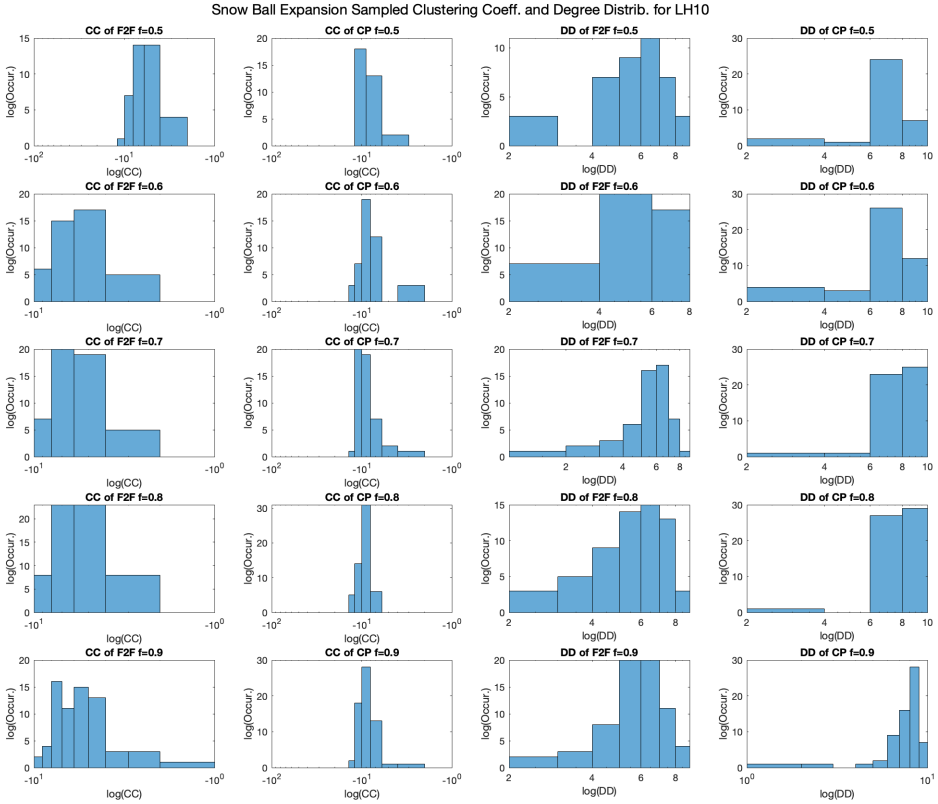
Dataset 3 Sampled Graph Histograms



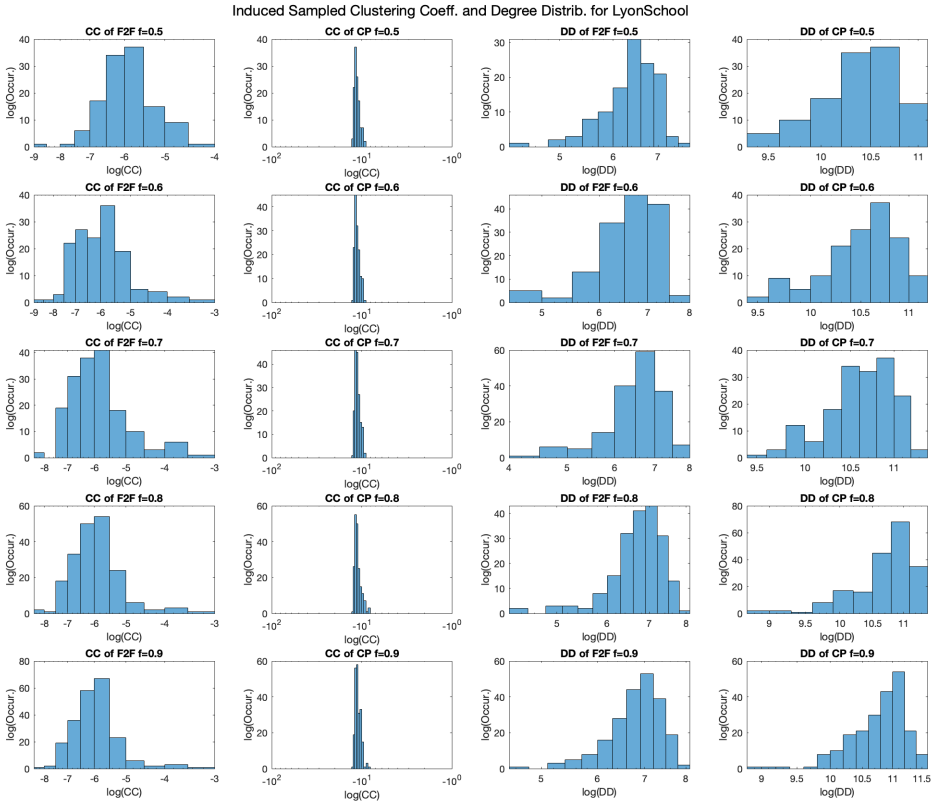


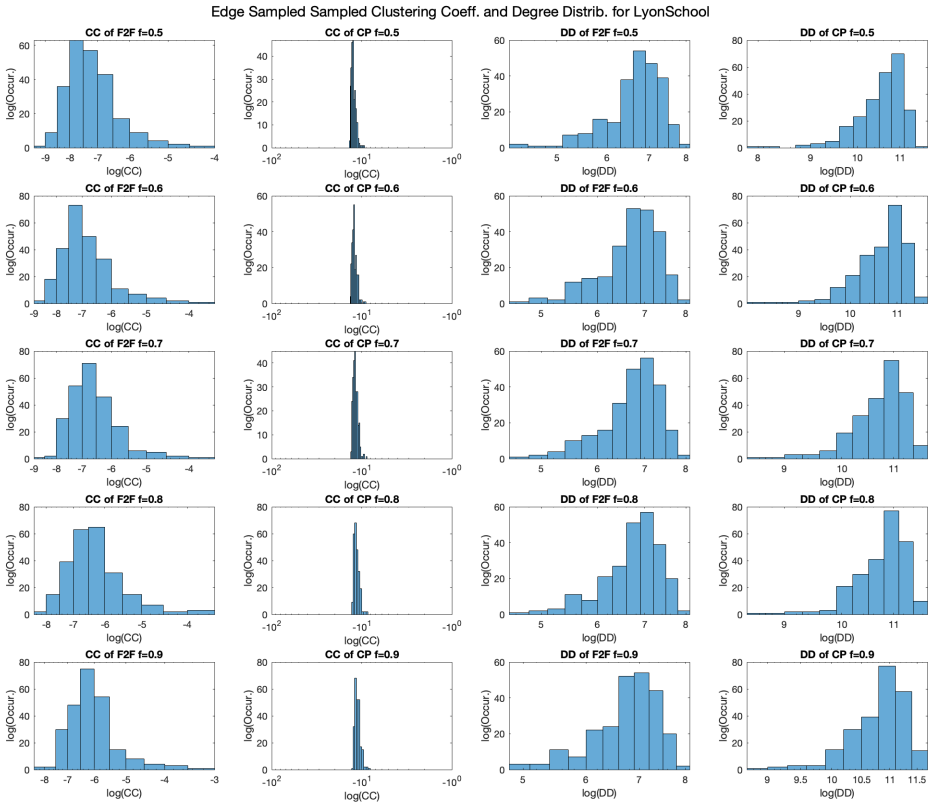

Metropolis Hastings RW Sampled Sampled Clustering Coeff. and Degree Distrib. for LH10
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

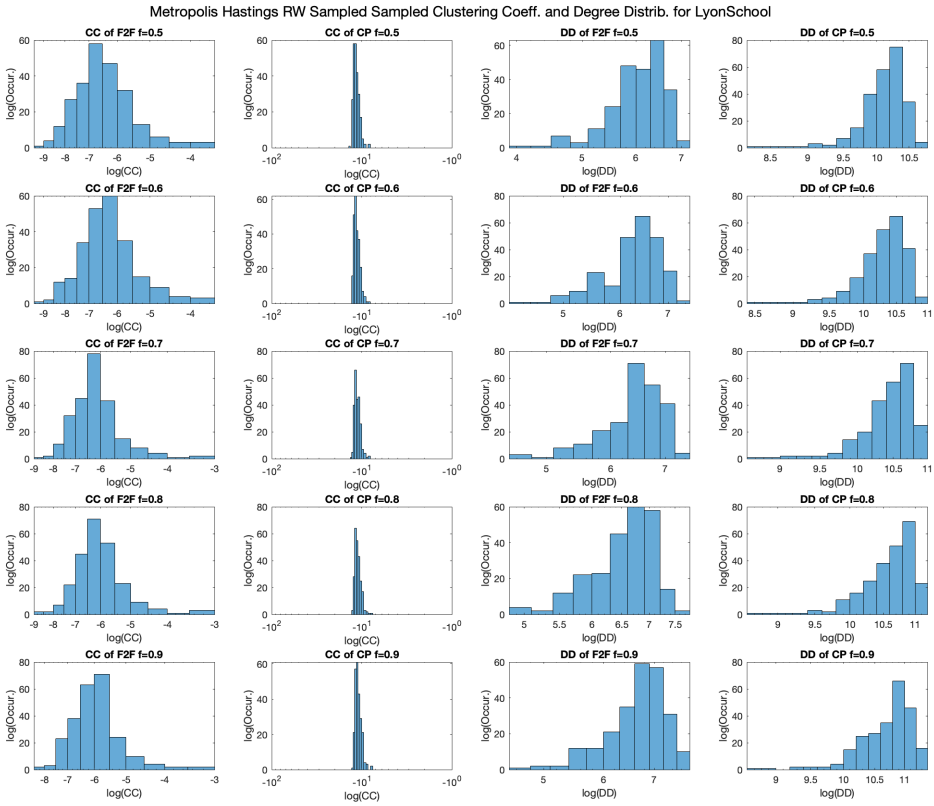

Frontier Sampled Sampled Clustering Coeff. and Degree Distrib. for LH10
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

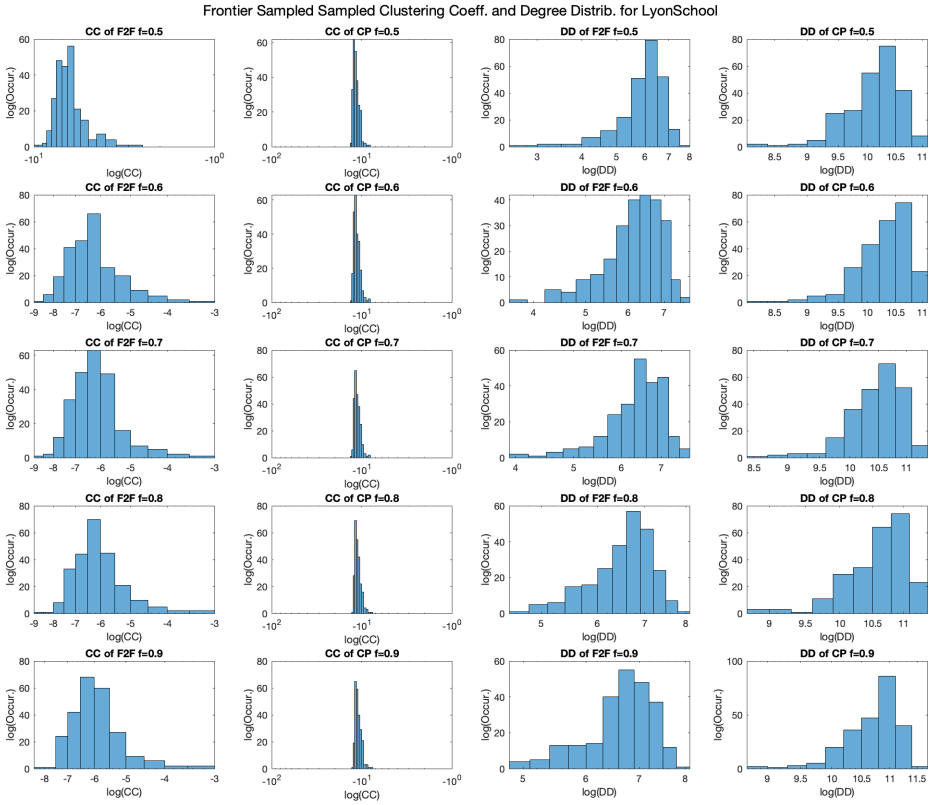

Snow Ball Expansion Sampled Clustering Coeff. and Degree Distrib. for LH10
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

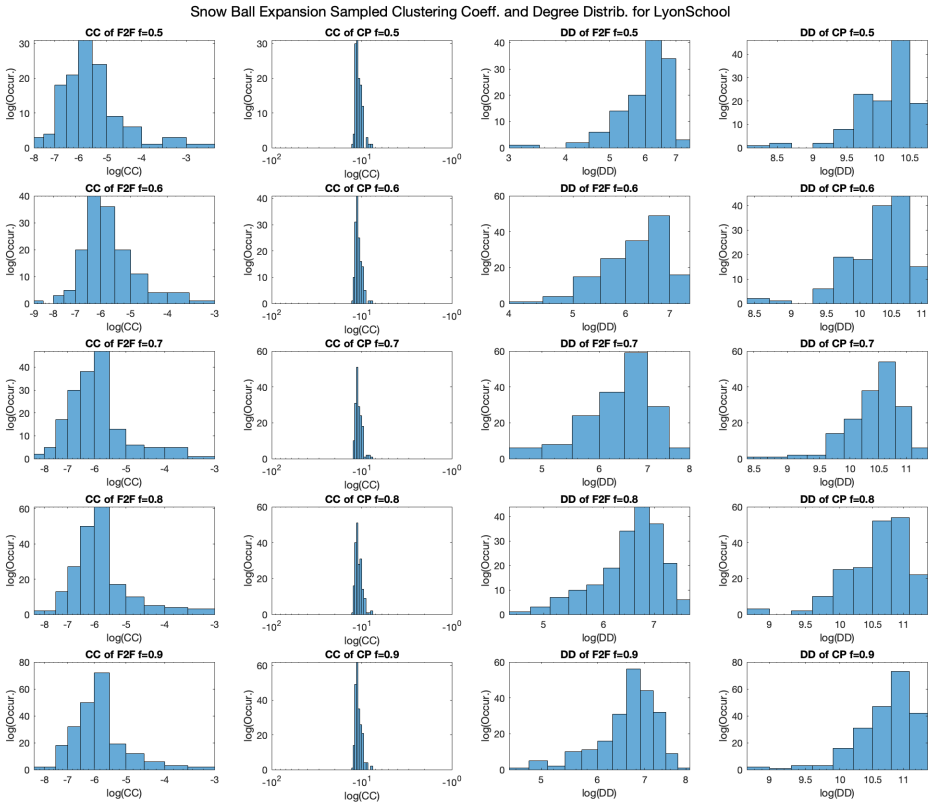
Dataset 4 Sampled Graph Histograms



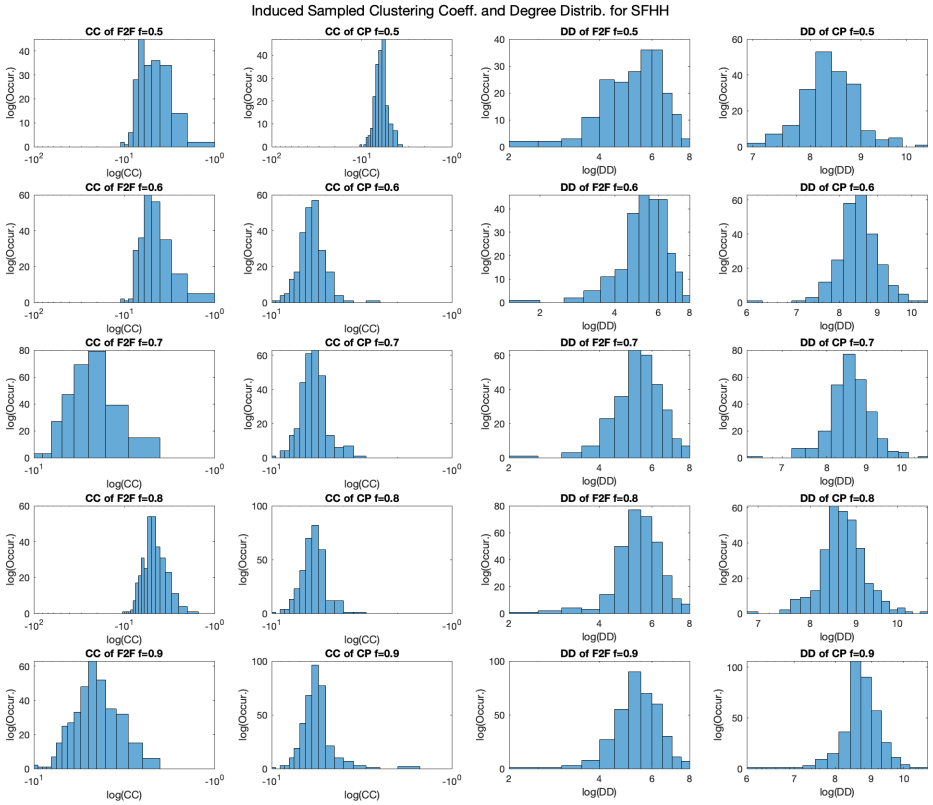


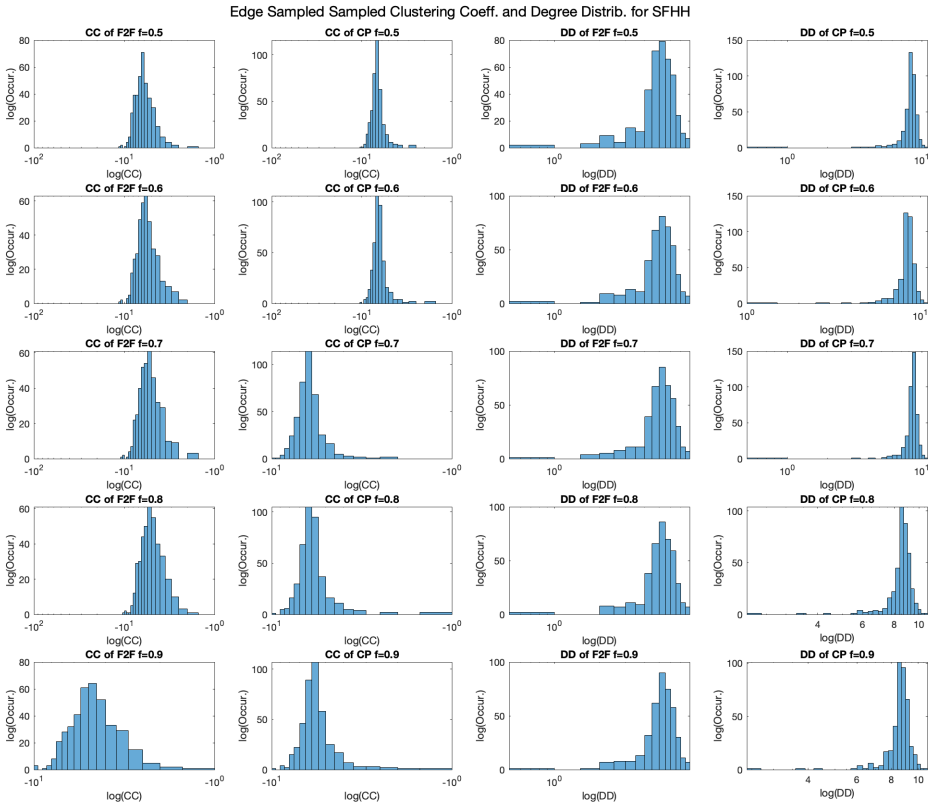

Metropolis Hastings RW Sampled Sampled Clustering Coeff. and Degree Distrib. for LyonSchool
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

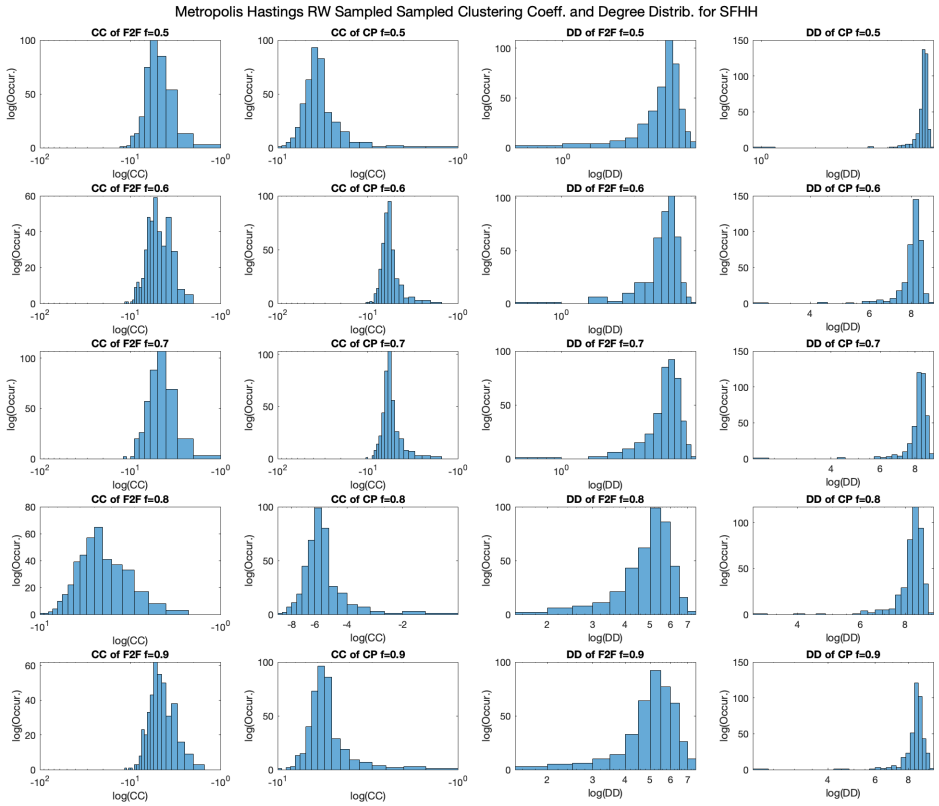

Frontier Sampled Sampled Clustering Coeff. and Degree Distrib. for LyonSchool
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

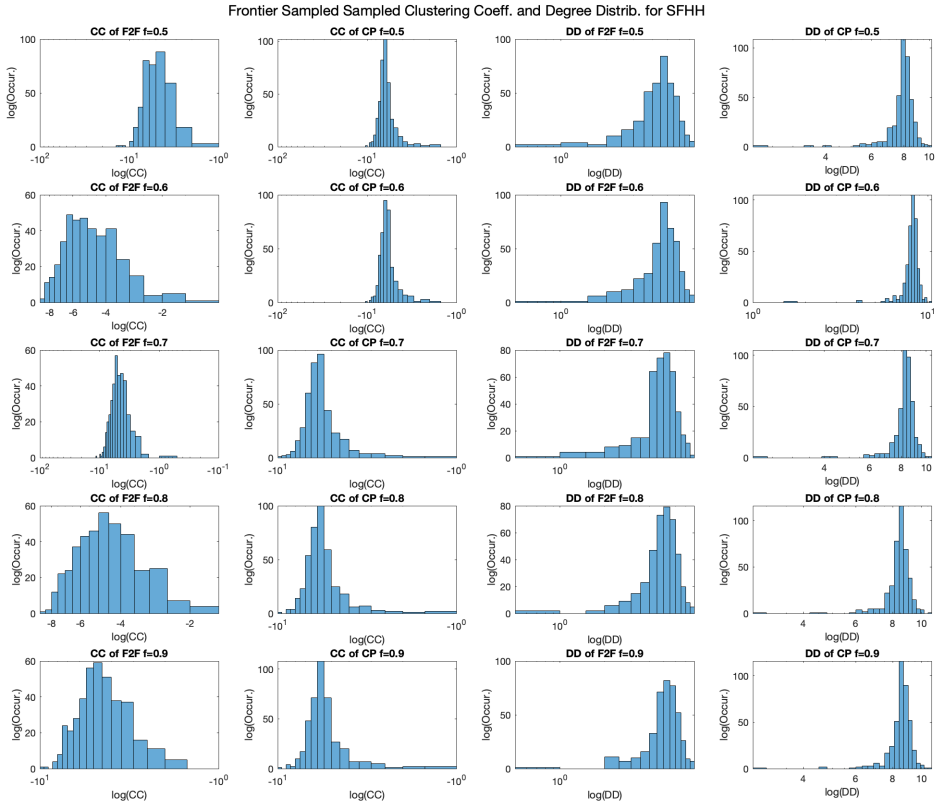

Snow Ball Expansion Sampled Clustering Coeff. and Degree Distrib. for LyonSchool
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

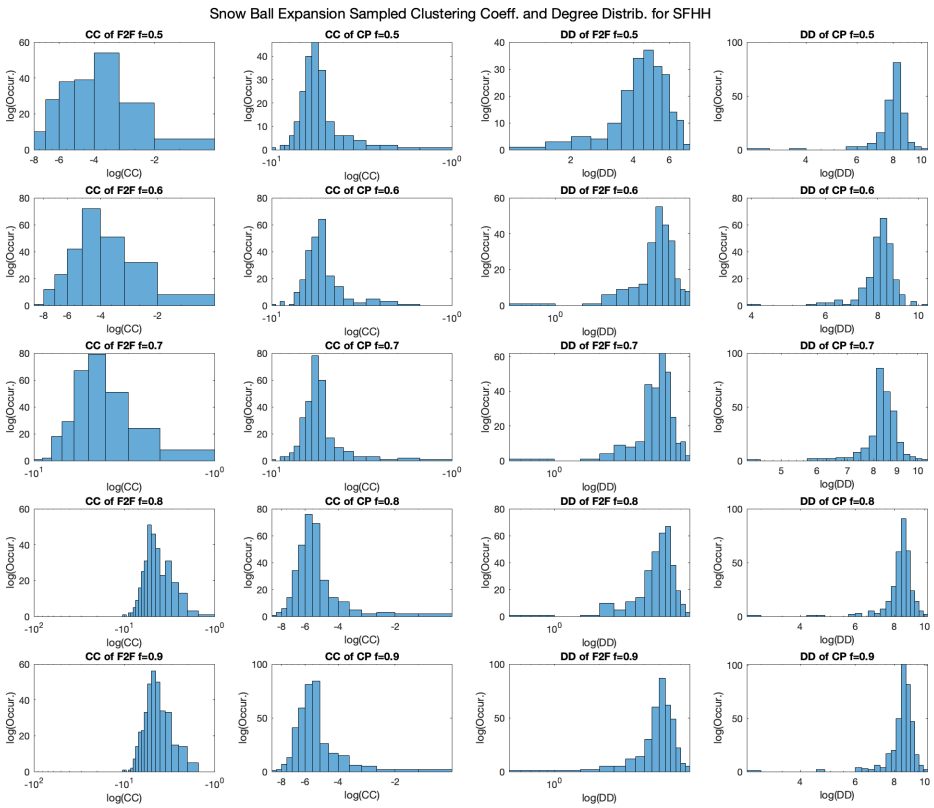
Dataset 5 Sampled Graph Histograms



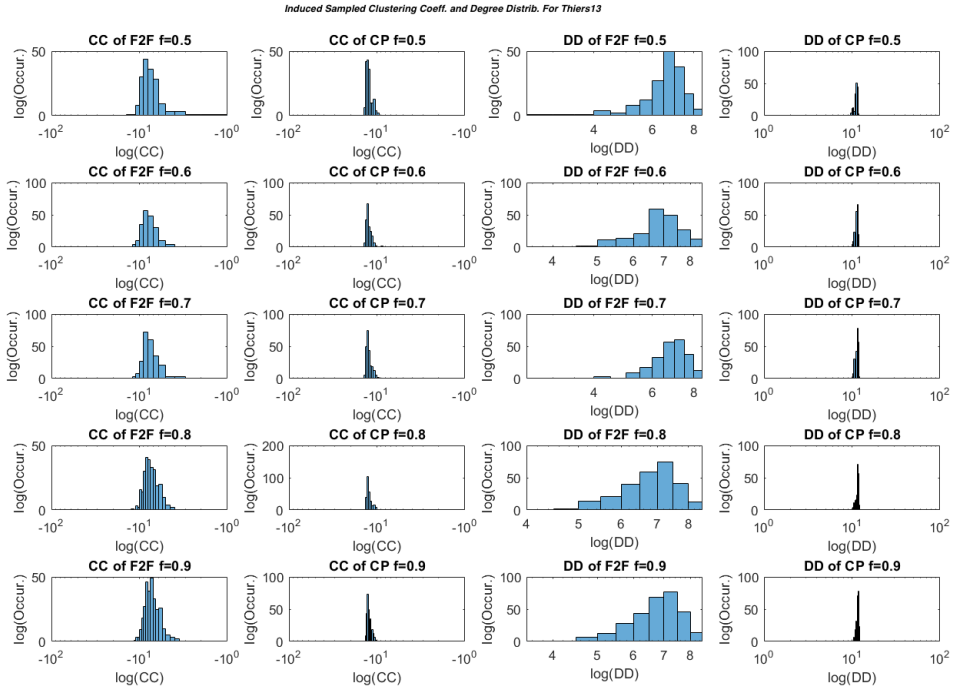


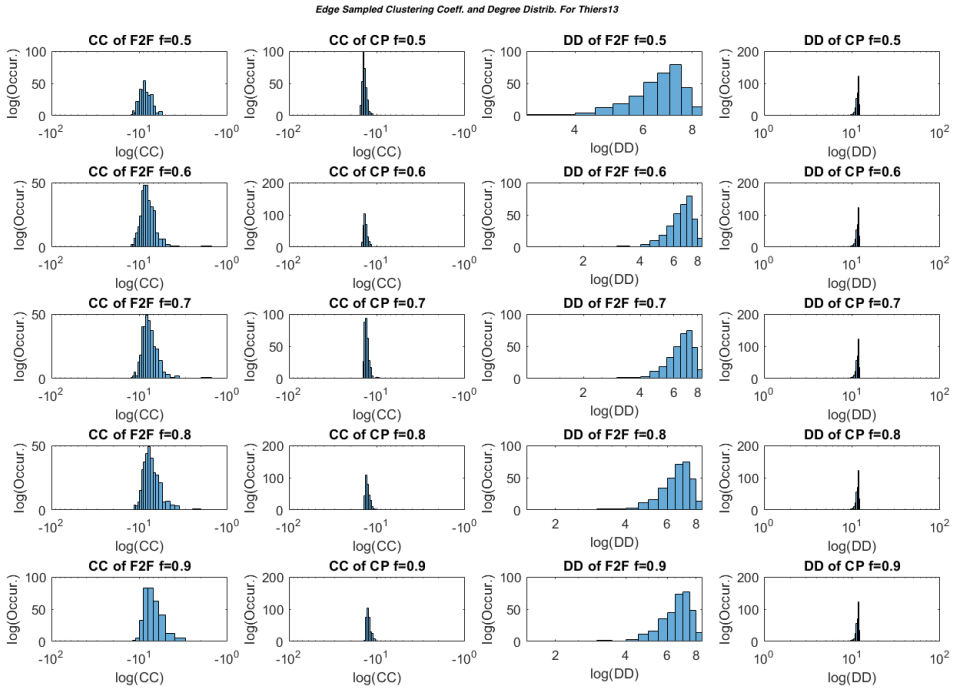

Metropolis Hastings RW Sampled Sampled Clustering Coeff. and Degree Distrib. for SFHH
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

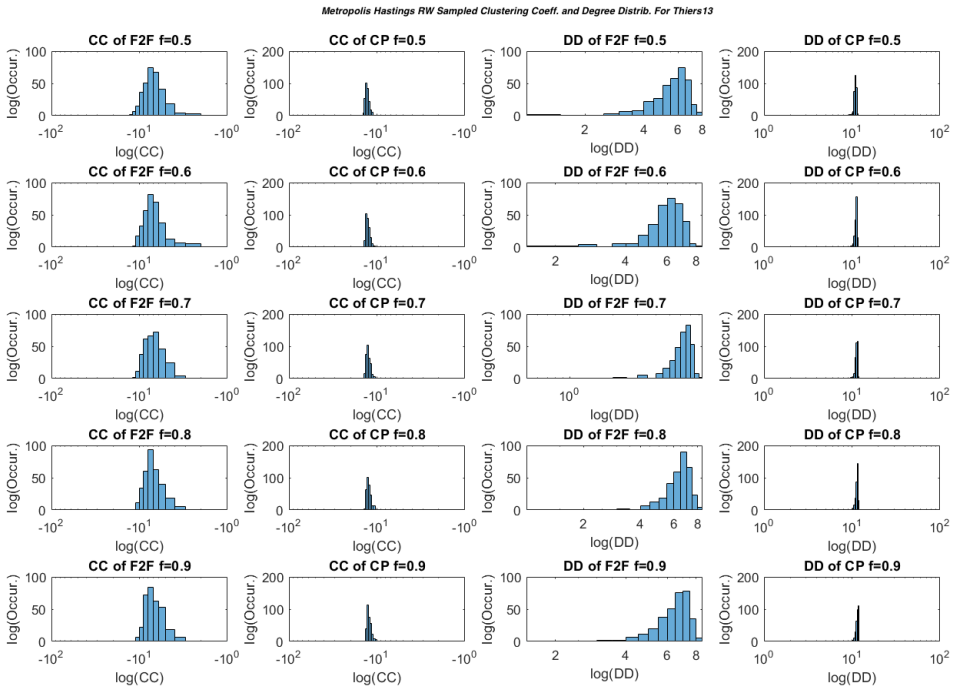

Frontier Sampled Sampled Clustering Coeff. and Degree Distrib. for SFHH
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

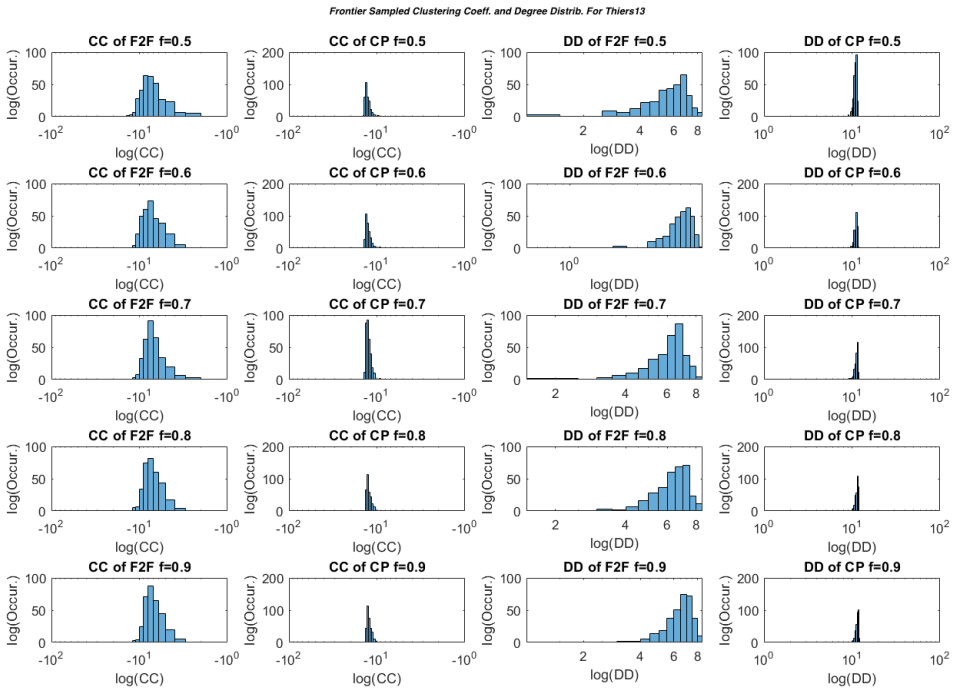

Snow Ball Expansion Sampled Clustering Coeff. and Degree Distrib. for SFHH
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

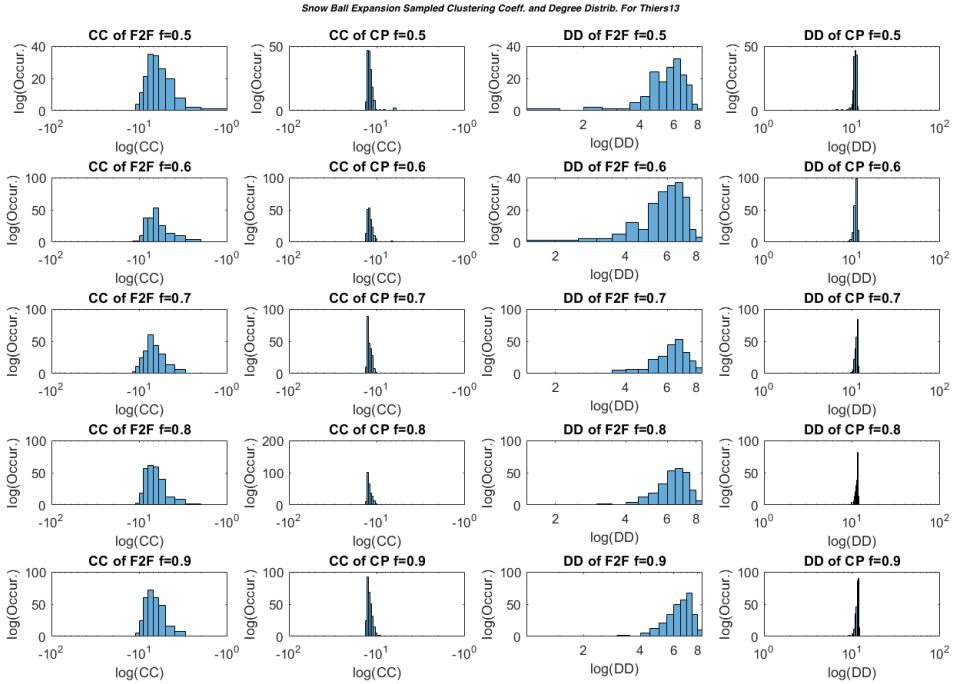
Dataset 6 Sampled Graph Histograms Dataset 5 Sampled Graph Histograms






Metropolis Hastings RW Sampled Clustering Coeff. and Degree Distrib. For Thiers13
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)


Frontier Sampled Clustering Coeff. and Degree Distrib. For Thiers13
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)


Snow Ball Expansion Sampled Clustering Coeff. and Degree Distrib. For Thiers13
CC of F2F f=0.5
CC of CP f=0.5
DD of F2F f=0.5
DD of CP f=0.5
CC of F2F f=0.6
CC of CP f=0.6
DD of F2F f=0.6
DD of CP f=0.6
CC of F2F f=0.7
CC of CP f=0.7
DD of F2F f=0.7
DD of CP f=0.7
CC of F2F f=0.8
CC of CP f=0.8
DD of F2F f=0.8
DD of CP f=0.8
CC of F2F f=0.9
CC of CP f=0.9
DD of F2F f=0.9
DD of CP f=0.9
log(Occur.)
log(CC)
log(DD)

After looking at the scalar statistics, we can confirm our original hypothesis where we'd expect all the stats to decrease as our frequencies decrease because we are leaving out vertices/edges. This would lead to a conclusion that the depreciation of our scalar statistic would mean that we are losing information of our data sets. However the histograms show that the shape of our degree distribution and clustering coefficients are still preserved, and therefore the shape our graphs are still preserved. This is significant because this means that we can reduce our graphs for faster computations and still draw to the same conclusions as the original.

It may be noticed that the Graph Size for the edge dependent algorithms remain the same. This is not due to an error in the algorithm, but rather that there are so many edges, that even with a frequency of 50%, every node gets covered.

Part 6

Comparing the original face-to-face adjacency matrix with the sub-sampled co-presence, after analyzing the data, will be simpler when looking at the vector statistics as opposed to the scalar statistics in this case. Looking at all of the combinations of the vector statistics is a bit overwhelming due to the sheer number of permutations, but after general analysis over the plots there are several interesting things that can be determined. The first is that for both the degree and clustering distributions, the relative shapes of the distributions are maintained as the sampling ratio is decreased. This means that even with thinning out the data, sometimes up to 50%, some of the same information can roughly be conveyed about the data. This is good news for making larger computations more feasible. The next thing to observe is that when looking at the original face-to-face data against the sub-sampled co-presence data, the shapes of the distributions are roughly consistent as well. This is even better in the long run because the coarseness of the sampling and the size of the data set can be reduced while still maintaining many interesting properties of the graph. The final thing that can be evaluated using this data is looking at the algorithm-to-algorithm performance between the statistics when sub-sampling. Based on consistency between each of the data sets, it seems that the Frontier and Metropolis Hastings algorithm maintain the distribution shape the best as the number of sampled edges decreases. These also seem to be the most consistent between different dataset, which is also promising for general purpose use.

There are a number of ways to account for bias in the algorithms presented above. For instance, the algorithms, as they are currently written, jump to a completely random node if there exist not connected neighbors to the current cluster being explored. It may be more beneficial for the algorithms to weigh the degrees of the nodes to find an un-visited node which has the highest degree. By starting at a new node with a high degree the algorithm will be able to maximize the number of additional nodes / edges it is able to add before being force to once again choose a random node.

An additional way to reduce bias in the algorithms may be to start with a subset of starting vertices rather than a single starting vertex. This will give the algorithm a large neighborhood to explore.

Finally, it may be useful to understand how dense and how many edges a graph has before running an algorithm. For extremely sparse graphs the random algorithms may continuous bounce back and forth from isolated nodes. It may be better in this case to choose a starting node with the highest degree. For dense graphs with a lot of edges, a random algorithm may lend an advantage as any starting point is likely to have a significantly sized neighborhood.

Conclusion

The overall take away from running these numerous experiments and algorithms is that it is a feasible idea to represent the data-rich and fine-grained representation of connections between individuals, with a sparser and more coarsely sampled structures. This is promising due to the fact that in general we will not have access to well controlled data sets, so it gives the opportunity to extract meaningful data even in noisy environments. The other take-away is the overall evaluation of the random sub-sampling algorithms based on the datasets they were applied to. We found that the Frontier and Metropolis Hastings Random Walk were the most promising for future use for several reasons. The first is their consistency between datasets. Many of the statistical distributions between data sets and sub-sampled data sets were roughly the same structure when using these algorithms, which means they are more universally effective. Although the snowball expansion algorithm yielded similar results, we found that its computational overhead when computing these statistics was too high to present it as a competitor for the quality difference in the results. Moving forward, there are many key lessons regarding the data that we will be able to use when we begin to look at disease propagation through these graphs in the next portion of the project.

Code Appendix

Adjacency Matrix Generation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 1 and 2
% Temporally Aggregated Adjacency Matrix Generation
% Creates and saves adjacency matrix from list of
% datasets in face2faceData and coPresenceData vectors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - saveAdjacencyMatrix.m: Saves a face-to-face and co-presence
% adjacency matrix to the current directory
%
% - Inputs: face to face file to load (.dat), co-presence file (.dat)
% sorted index file from meta data (.mat), f2f adj. matrix file name
% to save to (.mat), and co presence file name to save to (.mat).
%
% - Outputs: Saved files in current directory
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function saveAdjacencyMatrix(f2fFile, cpFile, sortedIndicesFile, f2fSaveFileName, cpSaveFileName)
    f2fDataSet = load(f2fFile);
    cpDataSet = load(cpFile);
    sortedIndices = importdata(sortedIndicesFile);

    for i = 2:3 %Re-index column 2 and 3 (i and j) between 1 and # of nodes
        for j=1:size(f2fDataSet,1)
            % Face to face data
            reIndex = find(sortedIndices == f2fDataSet(j,i));
            reIndex = reIndex(1);
            f2fDataSet(j,i) = reIndex;
        end
        for j=1:size(cpDataSet,1)
            % Co-Presence data
            reIndex = find(sortedIndices == cpDataSet(j,i));
            reIndex = reIndex(1);
            cpDataSet(j,i) = reIndex;
        end
    end

    % Loop through re-indexed columns to populate the agg. adjacency matrix
    % By adding a edge to the adjacency matrix for every instance in time
    % that i and j are in contact
    fAdjacencyMatrix = zeros(size(sortedIndices,1), size(sortedIndices,1));
    cAdjacencyMatrix = zeros(size(sortedIndices,1), size(sortedIndices,1));
    for i=1:size(f2fDataSet,1)
        fAdjacencyMatrix(f2fDataSet(i,2), f2fDataSet(i,3)) = fAdjacencyMatrix(f2fDataSet(i,2), f2fDataSet(i,3)) + 1;
        fAdjacencyMatrix(f2fDataSet(i,3), f2fDataSet(i,2)) = fAdjacencyMatrix(f2fDataSet(i,3), f2fDataSet(i,2)) + 1;
    end
    for i=1:size(cpDataSet,1)
        cAdjacencyMatrix(cpDataSet(i,2), cpDataSet(i,3)) = cAdjacencyMatrix(cpDataSet(i,2), cpDataSet(i,3)) + 1;
        cAdjacencyMatrix(cpDataSet(i,3), cpDataSet(i,2)) = cAdjacencyMatrix(cpDataSet(i,3), cpDataSet(i,2)) + 1;
    end

    % Save adjacency matrices to current directory
    data = fAdjacencyMatrix;
    save(f2fSaveFileName, 'data')
    data = cAdjacencyMatrix;
    save(cpSaveFileName, 'data')

end
```

Triangle Counter/Calculator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 3 and 4
% Clustering Coefficient
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - numTriang.m: Determines the number of triangle clusters
% a given vertex is in
% - Inputs: Adjacency Matrix A and vertex v
% - Outputs: Number of triangle v is a part of
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function numTriang = triangV(A, v)

    numTriang = 0;
    for j=1:size(A,1) % Origin node
        if A(v,j) > 0 % Loop until neighbor is found
            for k=1:size(A,1)
                if A(j,k) > 0 && k ~= v % Neighbor of neighbor found
                    if A(k,v) > 0 % Neighbor of neighbor is connected to origin
                        numTriang = numTriang + 1; % Indicate v is part of a triangle
                    end
                end
            end
        end
    end

    numTriang = floor(numTriang / 2);
end
```

Statistics Computation

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 3 and 4
% Adjacency Matrix Statistics
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - computeStatistics.m: Computes stats for passed Adj. Matrix
```

```
% - Inputs: Adjacency Matrix A
% - Outputs: Displays Table of stat values, vector of degrees,
% and vector of clusteringCoeffs
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [graphSize, numEdges, volume, density, degreeDistribution, clusteringCoefficient, averageDegree, avgCC] = computeStatistics(adjacencyMatrix, sampledMtx)

    % Statistics to gather:
    graphSize = 0;
    numEdges = 0;
    volume = 0;
    density = 0;
    degreeDistribution = zeros(1,size(adjacencyMatrix,1));
    averageDegree = 0;
    clusteringCoefficient = zeros(1,size(adjacencyMatrix,1));
    avgCC = 0;

    if sampledMtx
        deg = sum(adjacencyMatrix);
        graphSize = sum(deg > 0); % grab all connected vertices
    else
        graphSize = size(adjacencyMatrix,1);
    end

    %Total number of edges
    numEdges = size(find(triu(adjacencyMatrix) > 0), 1);

    %Volume (sum of all edge weights)
    volume = sum(sum(triu(adjacencyMatrix)));

    %Density (2m/(n(n-1))) where m is number of edges and n is num of nodes
    density = (2*numEdges)/(graphSize*(graphSize-1));

    %Degree (sum of all edge weights connected to vertex v)
    for i=1:size(adjacencyMatrix,1)
        degreeDistribution(i) = sum(adjacencyMatrix(i,:));
    end

    %Average Degree
    averageDegree = sum(degreeDistribution) / graphSize;

    %Clustering Coefficient
    for i=1:size(adjacencyMatrix,1)
        deltaV = triangV(adjacencyMatrix, i);
        if degreeDistribution(i) < 2
            clusteringCoefficient(i) = 0;
        else
            clusteringCoefficient(i) = (2*deltaV)/(degreeDistribution(i)*(degreeDistribution(i)-1));
        end
    end

    % Average Clustering Coefficient
    avgCC = sum(clusteringCoefficient) / graphSize;

    % Table output of data
    %table(graphSize, numEdges, volume, density, averageDegree, avgCC)
end
```

Induced Graph Sampling Algorithm

```
function [As] = InducedGraphSampling(G, p, k)
%INDUCED Summary of this function goes here
%   Detailed explanation goes here

    k = round(k);
    Vs = zeros(1, k);

    %Select K vertices based on P from G(E,V)
    [x, y] = size(G);

    if p == "uniform"
        %uniform sampling without replacement
        Vs = randperm(x, k);

    elseif p == "weighted"
        %calculate weights based on vertex degree's
        Vweights = sum(G);
        Volume = sum(Vweights);

        %weighted sampling without replacement
        w = 2.*Vweights./Volume;
        n = 1:x;

        isolatedNodes = sum(w == 0);
        if k > x - isolatedNodes
           k = x - isolatedNodes;
        end

        for i = 1:k
            Vs(i) = randsample(n,1,true,w);
            w = w(n ~= Vs(i));
            n = setdiff(n,Vs(i));
        end
    end

    %Select Edges
    As = zeros(x);
    for m = 1:k
        for n = 1:k
            v = Vs(m);
            u = Vs(n);
            As(v,u) = G(v, u);
        end
    end

end
```

Edge Sampling Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Clint Olsen
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 5 and 6
% Edge Sampling Algorithm 2
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - edgeSampling.m: Graph subsampling algorithm 2
% - Inputs: Adjacency Matrix A, prob. dist p, num of sub
% sampled edges k
% - Outputs: Subsampled graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function As = edgeSampling(A, p, k)

    k = round(k);

    % Subsampled Adjacency Matrix
    As = zeros(size(A,1), size(A,2));
    Es = zeros(1, k);

    edgeIndex = 0;
    % Generate Collection of Edges (only through upper triangular)
    for i=1:size(A,1)
        for j=i:size(A,1)
            if A(i,j) > 0
                edges(edgeIndex+1).i = i;
                edges(edgeIndex+1).j = j;
                edges(edgeIndex+1).weight = A(i,j);
                edgeWeights(edgeIndex+1) = A(i,j);
                edgeIndex = edgeIndex + 1;
            end
        end
    end

    if p == "uniform"
        Es = randperm(edgeIndex, k);

        for i=1:k
            % Place selected Edges and Vertices into Adj. Matrix
            As(edges(Es(i)).i, edges(Es(i)).j) = edges(Es(i)).weight;
            As(edges(Es(i)).j, edges(Es(i)).i) = edges(Es(i)).weight;
        end
    end

    if p == "weighted"
        edgeWeights = edgeWeights ./ sum(edgeWeights);
        for i=1:k
            Es(i) = randsample(1:size(edgeWeights, 2), 1, true, edgeWeights);
            edgeWeights(Es(i)) = 0; % Alters to without replacement

            % Place selected Edges and Vertices into Adj. Matrix
            As(edges(Es(i)).i, edges(Es(i)).j) = edges(Es(i)).weight;
            As(edges(Es(i)).j, edges(Es(i)).i) = edges(Es(i)).weight;
        end
    end
end
```

Metropolis Hastings Random Walk Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Girish Narayanswamy
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 5 and 6
% Edge Sampling Algorithm 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - metropolisHastingsRW.m: Graph subsampling algorithm 3
% - Inputs: Adjacency Matrix A,  num of sub sampled edges ms
% - Outputs: Subsampled graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [As] = metropolisHastingsRW(A,ms)

ms = round(ms);

% Construct sampled A matrix
[m,n] = size(A);
As = zeros(m,n); % zero init As adjacency mtx of the subset
sizeS = 0; % number of edges sampled

v = randi([1 m]); % Random starting vertex
while sizeS < ms

    Nv = find(A(v,:) > 0); % neighbor hood of v
    Nv = intersect(Nv,find(As(v,:) == 0)); % Nv limited to unvisited nodes
    if isempty(Nv)
        v = randi([1 m]); % Random starting vertex
        continue;
    end

    w = Nv(randi([1 length(Nv)])); % choose a random neighbor of v

    p = unifrnd(0,1); % choose uniform-random probability [0 1]
    degv = sum(A(v,:));
    degw = sum(A(w,:));
    if p <= degv/degw
        As(v,w) = A(v,w); % add connection to new adjacency matric
        As(w,v) = A(w,v);

        v = w; % move to new node
        sizeS = sizeS + 1; % add an edge
    end
end
end
```

Frontier Sampling Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Girish Narayanswamy
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 5 and 6
% Frontier Sampling Algorithm 4
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% - frontierSampling.m: Graph subsampling algorithm 4
% - Inputs: Adjacency Matrix A, num of sub sampled edges ms
% - Outputs: Subsampled graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [As] = frontierSampling(A, ms)

ms = round(ms);

% Construct sampled A matrix
[m,n] = size(A);
As = zeros(m,n); % zero init As adjacency mtx of the subset

s = randi(m);
L = randperm(m,s); % random vertex seed of size s
k = 0; % iterator

while k < ms

    % check to see if L must be reshuffled again
    for i = 1:length(L)
        u_test = L(i);
        Nu_test = find(A(u_test,:) > 0);
        Nu_test = intersect(Nu_test,find(As(u_test,:) == 0));

        if ~isempty(Nu_test)
            break % possible accessible nodes still exist
        end

        if i == length(L) % no more possible nodes to visit, reseed L
            L = randperm(m,s);
        end
    end

    degLTotal = sum(sum(A(:,L))); % total degree weights of u in L
    pu = sum(A(:,L))/degLTotal; % probability of each vertex u

    if isempty(find(pu) > 0)
        continue; % if no new edges to add repeat iteration and generate new u
    end

    idx = randsample(s,1,true,pu); % choose a random index in L with probabilities pu
    u = L(idx); % find node number using idx generated above

    Nu = find(A(u,:) > 0); % neighbor hood of u
    Nu = intersect(Nu,find(As(u,:) == 0)); % Nu limited to unvisited nodes
    if isempty(Nu)
        continue; % if no new edges to add repeat iteration and generate new u
    end

    v = Nu(randi(length(Nu)));

    As(u,v) = A(u,v); % add edge to As matrix
    As(v,u) = A(v,u);

    k = k + 1; % iterate number of edges
    L(idx) = v; % replace u with outgoing vertex v
end
end
```

Snowball Expansion Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Girish Narayanswamy
% ECEN 5322: Higher-Dimensional Datasets
% Final Project: Assignment 5 and 6
% Snow Ball Expansion Algorithm 5
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% - snowBallExpansion.m: Graph subsampling algorithm 5
% - Inputs: Adjacency Matrix A, num of sub sampled vertices ms
% - Outputs: Subsampled graph
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [As] = snowBallExpansion(A,ms)

ms = round(ms);

% Construct sampled A matrix
[m,n] = size(A);
As = zeros(m,n); % zero init As adjacency mtx of the subset

v = randi(m); % choose radnom starting vertex
S = v; % contained vertices
sizeS = 1; % number of contained vertices


while sizeS < ms

    diffMax = 0; % value to maximize to add new node
    vMax = 0;

    NS = mod(find(A(S,:)' > 0),m); % neighborhood of N
    NS(NS == 0) = m; % make the indexes work still
    NSUS = union(NS,S); % get union of N(S) and S

    for i = 1:length(NS) % iterate through neighborhood of S

        v = NS(i); % vertex in S
        Nv = find(A(v,:) > 0); % neighborhood of v

        diffNew = length(setdiff(Nv,NSUS)); % difference between v neighborhood and current set Nhood
        if diffNew > diffMax % if higher difference save new node v
            diffMax = diffNew;
            vMax = v;
        end
    end

    % if there exists a node with new vertices
    if vMax > 0
        % populate edges for new v with existing nodes in S
        for i = 1:length(S)
            u = S(i);
            As(vMax,u) = A(vMax,u);
            As(u,vMax) = A(u,vMax);
        end
```

```
56
57          S = [S, vMax]; % add vertex to vertex list
58          sizeS = sizeS + 1;
59
60      else % no new vertices in neighborhood cluster
61
62          % find a new random unvisited node
63          all_v = 1:m;
64          remaining_v = setdiff(all_v, S);
65          idx = randi(length(remaining_v));
66          vMax = remaining_v(idx);
67
68          % populate edges for new v with existing nodes in S
69          for i = 1:length(S)
70              u = S(i);
71              As(vMax,u) = A(vMax,u);
72              As(u,vMax) = A(u,vMax);
73          end
74
75          S = [S, vMax]; % add vertex to vertex list
76          sizeS = sizeS + 1;
77      end
78  end
79  end
```