Selecting Clustering Algorithms for Identity-By-Descent Mapping - Supplemental Information

Ruhollah Shemirani [†] **and Gillian M Belbin**
Institute for Genomic Health, Icahn School of Medicine at Mount Sinai, New York, NY , USA
[†]*E-mail: ruhollah.shemirani@mssm.edu*

Keith Burghardt and Kristina Lerman
Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA

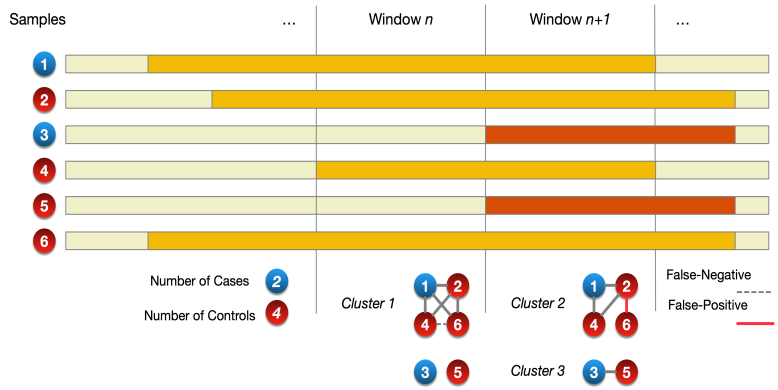
Christy L Avery
Department of Epidemiology, University of North Carolina, Chapel Hill, NC, USA

Eimear E Kenny
Institute for Genomic Health, Icahn School of Medicine at Mount Sinai, New York, NY , USA

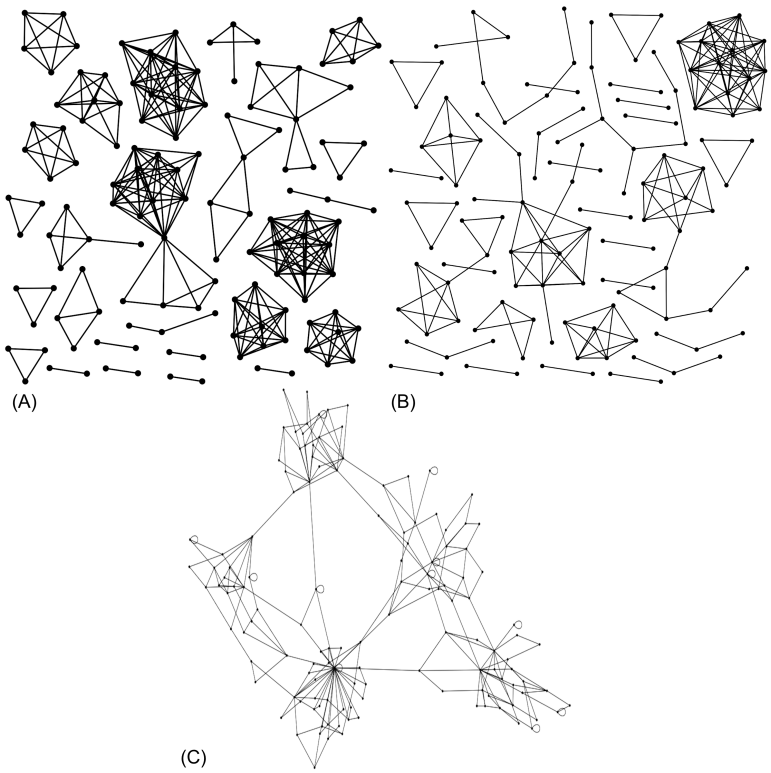
Christopher R Gignoux
Colorado Center for Personalized Medicine, University of Colorado Anschutz Medical Campus, Aurora, CO , USA

JoséLuis Ambite
Information Sciences Institute, University of Southern California, Marina Del Rey, CA, USA

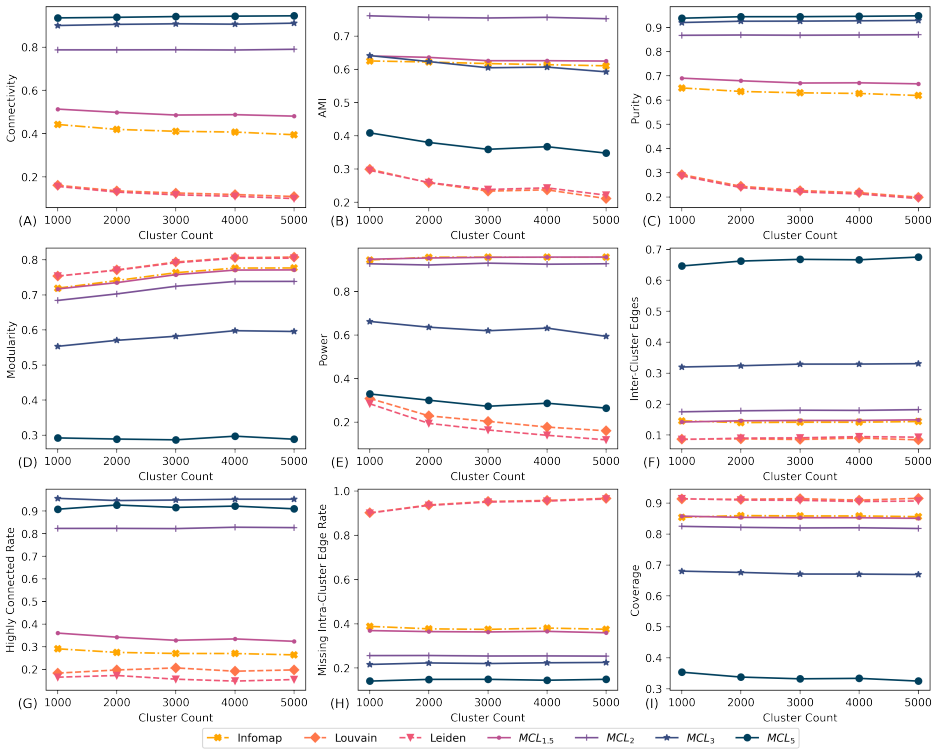
1. Supplementary Figures



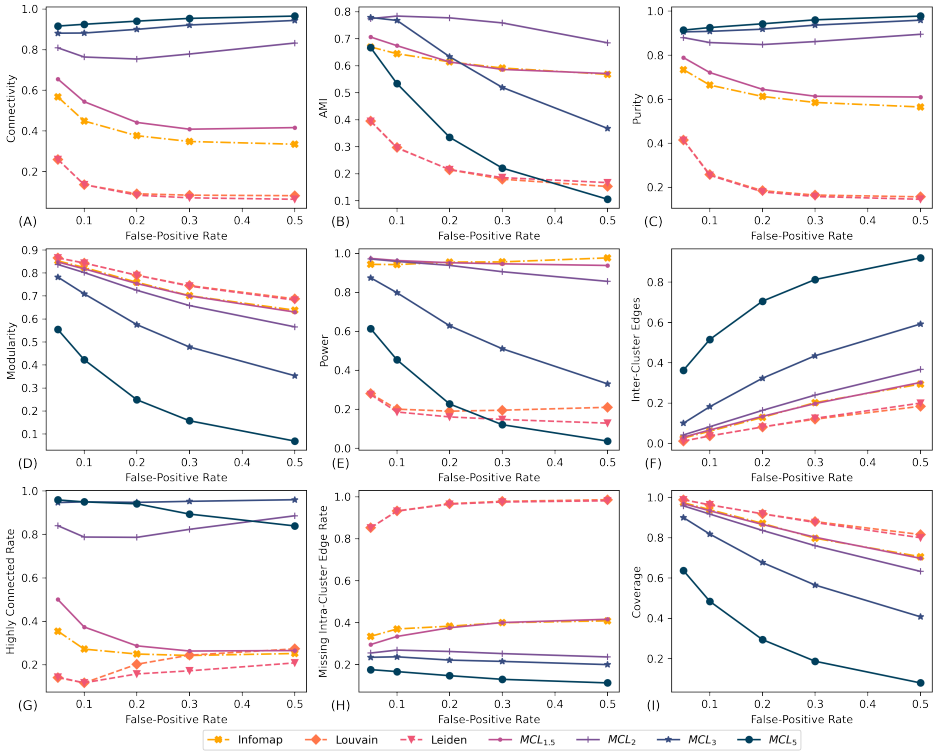
Supplementary Fig 1. **IBD Mapping Process** A general schema of the IBD mapping process. First, IBD segments are estimated and divided into windows. Second, a clustering algorithm finds the underlying communities in each window, eliminating false-positive and recovering false-negative edges. Third, a statistical test is conducted to find associations between cluster memberships and phenotypes. Significant dissimilarities between a family cluster and general population might be caused by an unrecorded rare variant carried on a shared IBD segment by the family.



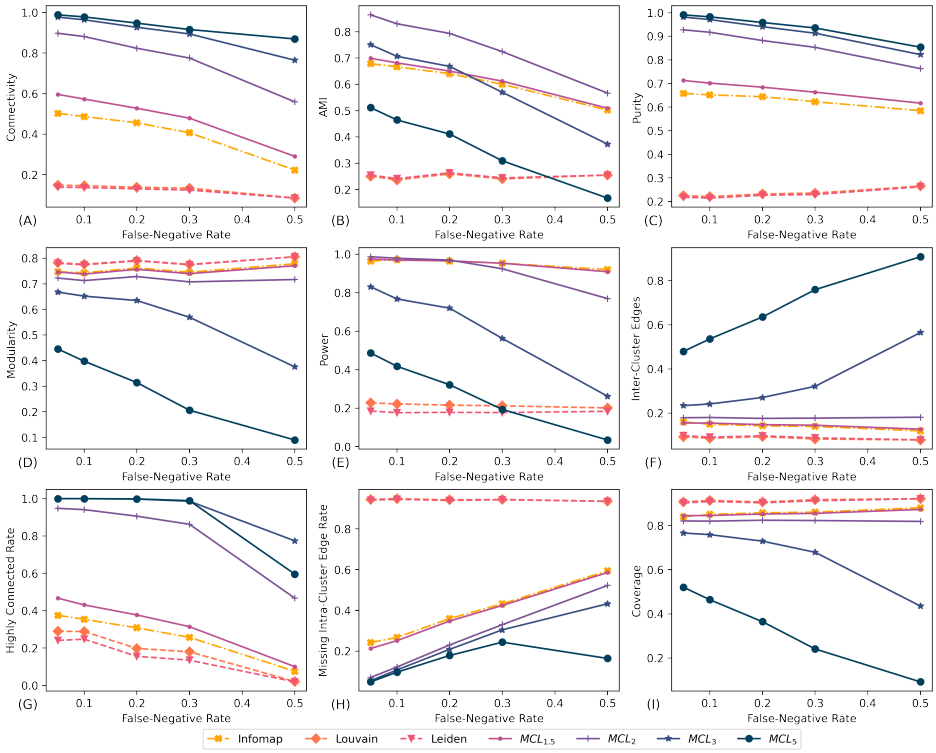
Supplementary Fig 2. **Comparison of graph layouts** Graph layouts with $\sim$ 130 nodes generated (A) from the real IBD data in the PAGE study dataset on chromosome one via random sampling, (B) by our benchmark algorithm, (C) using the LFR method. LFR benchmarks are generally comprised of a single connected component which does not happen in real local IBD graphs. Further, the cluster sizes generated using the power law distribution do not resemble that of the real local IBD graphs. Our benchmark addresses both of these issues.



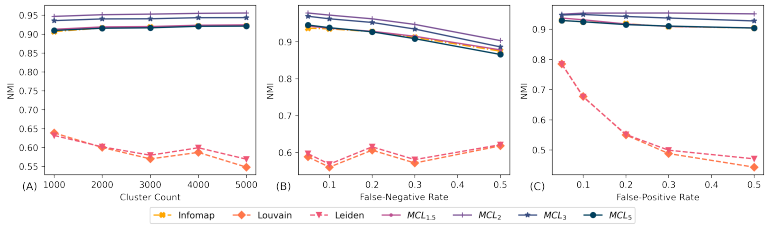
Supplementary Fig 3. **The effects of cluster count on the scores of algorithms** The effects of number of simulated clusters on the performance of algorithms through different metrics using 750 simulated graphs sampled from the PAGE dataset. With false-positive and false-negative edges ranging from 5%-50%. Total number of clusters affected the performance of some algorithms. Such dependency on the number of clusters negatively affects IBD mapping application.



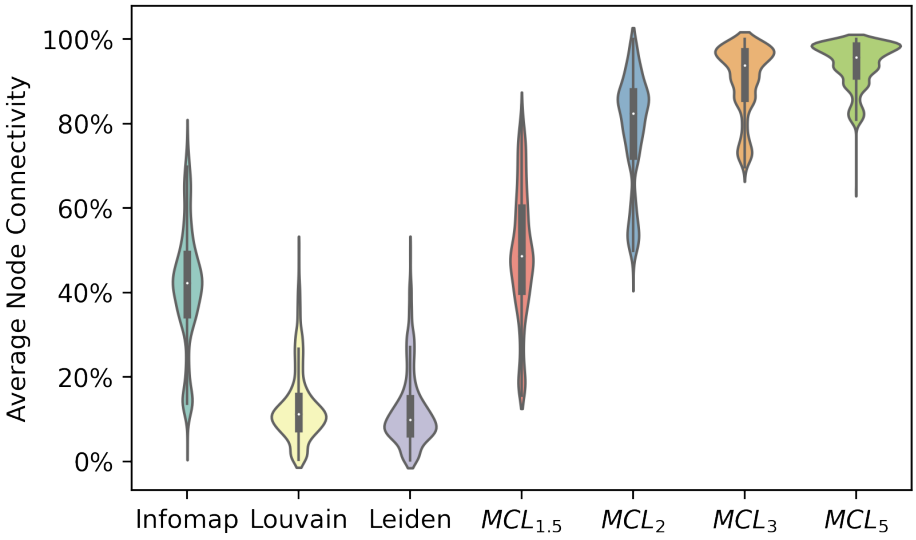
Supplementary Fig 4. **The effects of false-positives on the scores of algorithms** The effects of false-positive rate on the performance of algorithms through different metrics using 750 simulated graphs sampled from the PAGE dataset. False-negative edges ranging from 5%-50% and the number of simulated clusters per graph ranging from 1,000 to 5,000. Infomap and $MCL_{1.5}$ had the most stable performances across various rates of false-positives, followed closely by MCL_2. In real dataset, we expect the false-positive rate to never go above 15%.



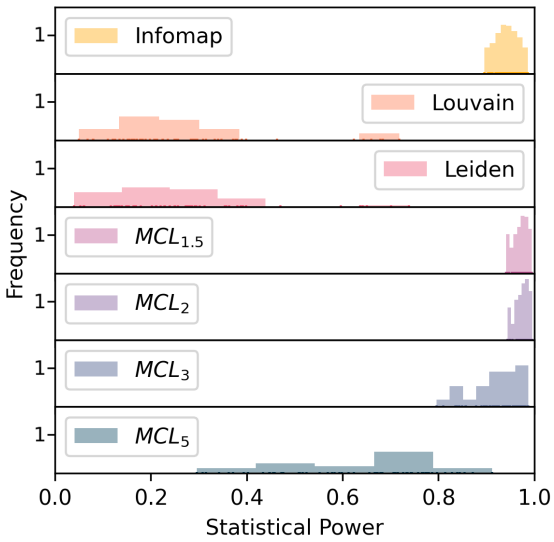
Supplementary Fig 5. **The effects of false-negatives on the scores of algorithms** The effects of false-negative rate on the performance of algorithms through different metrics using 6,250 simulated graphs sampled from the PAGE dataset. False-positive edges ranging from 5%-50% and the number of simulated clusters per graph ranging from 1,000 to 5,000. Compared to observations with the number of clusters and rate of false-positives, Louvain and Leiden have a stable performance when false-negative rate is increased. Although they are still under-performing against Infomap and MCL with an 80% statistical power score gap.



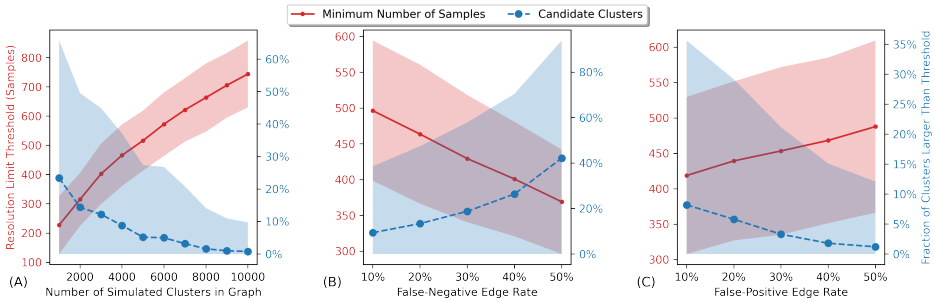
Supplementary Fig 6. **NMI score trends** The NMI score of algorithms as (A) the number of clusters, (B) false-negative rate, and (C) false-positive rate grows over all experiments. For MCL and Infomap, an increased number of clusters can increase NMI score due to a lack of adjustment for the probability of a random clustering conforming to the ground truth. Louvain and Leiden are the exceptions here as they undergo the effects of resolution-limit.



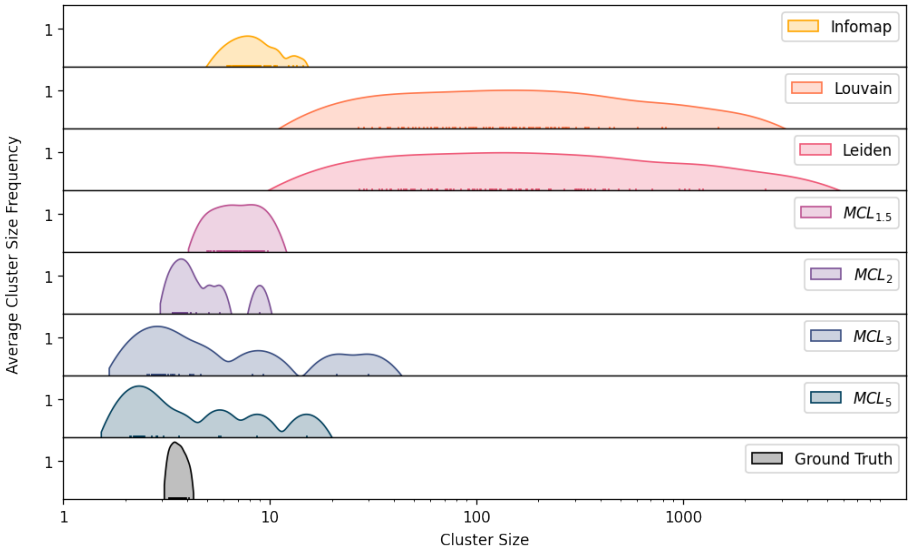
Supplementary Fig 7. **The effect of resolution-limit on the density of clusters** A violin plot for average connectivity score of clustering algorithms across our 750 simulations. Among the top performing algorithms in terms of power, MCL_2 had the highest connectivity score at 79%, suggesting that it finds densely connected clusters that also conform to the ground truth. Infomap and $MCL1.5$ gained mean connectivity scores of 41% and 49% respectively. The lowest average scores belonged to Louvain and Leiden at 13% and 12%, respectively. Connectivity scores of Louvain and Leiden further demonstrate the inability of Louvain and Leiden to accurately extract clusters due to the resolution limit.



Supplementary Fig 8. **Distribution of Statistical Power Scores** The distribution of power scores in 60 experiments with a false-positive rate of 5% and false-negative rate of 20% and 30%. These rates were closest to our predicted rates in the PAGE dataset.



Supplementary Fig 9. **Resolution-limit trends** Trends for the average resolution limit threshold (red), and the expected frequency of clusters that pass the resolution limit threshold (blue) as (A) the number clusters grows, (B) number of false-positive edges grows, and (C) number of false-negative edges grows in 100 simulations sampled from the PAGE study dataset. Clusters that are not large enough to pass this threshold may be merged with other clusters in modularity optimization clustering process. The shaded area shows 95% confidence interval.



Supplementary Fig 10. **Extracted cluster size distribution** Distribution of the average number of nodes per cluster found by each algorithm across 100 simulations. Each simulated graph was comprised of 2,000 clusters, sampled from chromosome 1 in the PAGE study dataset with 25% false-positive and false-negative edges added. MCL_2 had the closest distribution to the ground truth, while modularity optimizing methods had the furthest distribution. They failed to capture many of the small clusters, merging them into larger ones, compared to other methods. Infomap and $MCL_{1.5}$ had slightly coarser-grained clusters that the ground truth. MCL_3 and MCL_5 often over-clustered and returned finer-grained clusters.

2. Supplementary Methods

2.1. Benchmark Algorithm

Our benchmark simulation has four steps. First, we generate a set of cliques, fully connected inside and disconnected from other cliques. These represent distinct IBD families in a given window. Members of a clique share the same DNA inherited from a common ancestor. To generate realistic clique size distributions, we sampled community size distributions on the chromosome one in the PAGE study dataset. In the second step, to simulate false-negative IBD segments, we randomly remove edges from each clique. Thirdly, in the false-positive edge generation step, we add inter-cluster edges to further increase the noise based on a given false-positive rate. Fourth, since the ultimate goal of IBD mapping is to test the family clusters for associations with traits, we simulate a set of binary traits, one for every cluster with more than 10 members. Traits have a significantly different prevalence among their designated cluster compared to the whole population. For example, trait t_i, associated with cluster ω_i, has a prevalence rate of P_i when every node (whole population) is tested. However, when only members of the cluster ω_i are tested, a different prevalence of p_i^i is achieved. The two prevalence rates P_i and p_i^i are chosen based on the cluster size and total node count such that the disparity between them is statistically significant (p-value$_{binomial} \sim 10^{-10}$). The significance threshold was chosen empirically.

2.1.1. Simulation Setup

We generated 750 graphs (one per experiment). The number of clusters in each experiment ranged from 1,000 to 5,000. For each cluster count, we considered 25 sets of false-positive and false-negative combinations ranging from 5% to 50%. Finally, for each combination of cluster count, false-positive, and false-negative rates we simulated six experiments. The mean and standard error of the prevalence of simulated traits among the whole population in each of these six experiments were selected from a set of 3 (0.05, 0.1, 0.15) and 2 (0.01 ,0.1) predetermined values, respectively; for a total of 750 experiments. This added up to a total of 2,274,500 clusters with more than 6 million nodes across all simulated experiments.

We demonstrate the accuracy of our benchmark in Figure 2. The figure illustrates the layout of a random sample of local IBD graphs on chromosome 1 (Figure 2 A) against randomly generated benchmarks, both using our algorithm (Figure 2 B) and the LFR algorithm (Figure 2 C). As shown in the figure, our benchmark simulates the disjointedness of local IBD graphs, unlike the LFR algorithm.

2.2. Metrics

Clustering algorithms categorize nodes into groups called clusters (The words cluster, community, class and category are often used interchangeably in clustering literature). Metrics help analyze various properties of the resulting clusters that are either related to the inherent features of the clusters, such as the density of connections in the clusters, or their concordance with the true structure of the graph, such as the number of nodes that are in the same clusters as they are in the ground truth. We call the first group feature-based metrics in this manuscript to distinguish them from metrics that are based on ground truth. For local IBD clustering, it is important to calculate how much the results reflect the true structure of the cliques underneath the noise and errors. We studied four metrics that aim to quantify such correlation with the ground truth through measuring information recovery. Since ground truth is often not available for real datasets, we also analyzed six clustering feature-based metrics to evaluate their efficiency in the absence of a ground truth.

2.2.1. Purity

Purity measures the the degree to which the clustering results replicate the ground truth clusters using a simple greedy mapping of the two [1]. To calculate purity, for every cluster ϕ_j recovered by the algorithm, we find a ground truth cluster ω_i with the highest number of common nodes and assign all the nodes in ϕ_j to ω_i. Purity is then defined as the fraction of nodes that receive the correct cluster label using this approach. It is calculated using the following formula:

$$purity(\Phi|\Omega) = \frac{1}{|V|}\sum_{j=1}^{M} \underset{i}{max}|\phi_j \cap \omega_i| \tag{1}$$

Where V is the set of nodes in the graph, $\Omega = \{\omega_1, \omega_2, ..., \omega_N\}$ is the set of ground truth clusters, and $\Phi = \{\phi_1, \phi_2, ..., \phi_M\}$ is the set of clusters extracted by the community detection algorithm. Purity is a simple metric to define and to calculate. It helps measure, transparently, how added noise can affect the structure of the clusters extracted compared to the original cliques. However, purity score can be maximized by assigning every node to a one node cluster of its own. For example, breaking down a cluster into two (or more) does not have an effect on the purity score, while it negatively affects power. Thus, for local IBD communities, where the majority of the clusters are doubletons, or tripletons, it can be misleading.

2.2.2. Normalized Mutual Information

Normalized Mutual Information (NMI) is built upon Shanon's information theory [2, 3]. It measures the amount of information shared between the ground truth and the clustering results. It is calculated using the following formula:

$$NMI(\Omega,\Phi) = \frac{I(\Omega,\Phi)}{[H(\Omega)+H(\Phi)]/2} \tag{2}$$

Where

$$I(\Omega,\Phi) = \sum_i \sum_j P(\omega_i \cap \phi_j) \log \frac{P(\omega_i \cap \phi_j)}{P(\omega_i)P(\phi_j)} = \sum_i \sum_j \frac{n_{ij}}{|V|} \log \left(\frac{|V| n_{ij}}{|\omega_i||\phi_j|} \right) \tag{3}$$

$$H(\Omega) = -\sum_i P(\omega_i) \log P(\omega_i) = -\sum_i \frac{|\omega_i|}{|V|} \log \left(\frac{|\omega_i|}{|V|} \right) \tag{4}$$

With $|V|$ as the total number of nodes, n_{ij} as the number of nodes in $\omega_i \cap \phi_j$, and $H(\Phi)$ calculated using the same approach as $H(\Omega)$ (the entropy of the clustering and the entropy of the ground truth). Unlike purity, the value of NMI can only be maximized by replicating the same node assignments as the ground truth. The value of this score is 1 when the mutual information between Ω (ground truth) and Φ (clustering results) is maximized [3]. Compared to purity, increasing the number of clusters will not result in a perfect NMI score of 1, which makes it more dependable as a metric.

2.2.3. Adjusted Mutual Information

The baseline score in NMI can be improved through increasing the number of clusters. Vinh et al [4] first reported that the average NMI score of a random clustering increases as the number of clusters, or the size of the graph increases. To address this issue, they proposed the Adjusted Mutual Information (AMI). AMI is calculated by subtracting the expected mutual information (NMI score) of a random clustering of nodes from the NMI score of the real clustering. The random clustering should have the same number of clusters and number of nodes in each cluster.

$$NMI(\Omega,\Phi) = \frac{I(\Omega,\Phi) - E[I(\Omega,\Phi)]}{[H(\Omega)+H(\Phi)]/2 - E[I(\Omega,\Phi)]} \tag{5}$$

where

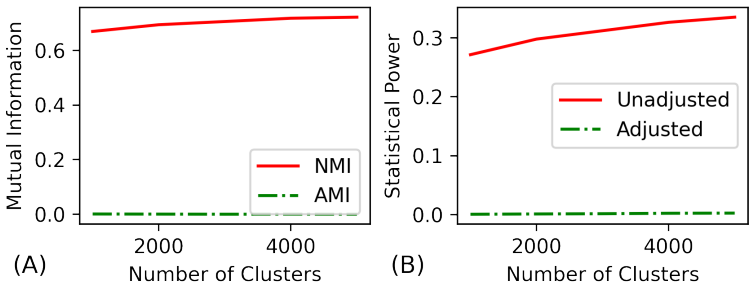
$$E[I(\Omega,\Phi)] = \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{n_{ij}=max(1,|\omega_i|+|\phi_j|-|V|)}^{min(|\omega_i|,|\phi_j|)} \frac{n_{ij}}{|V|} log \left(\frac{|V| \times n_{ij}}{|\omega_i||\phi_j|} \right) \times \frac{|\omega_i|!|\phi_j|!(|V|-|\omega_i|)!(|V|-|\phi_j|)!}{|V|!|n_{ij}!(|\omega_i|-n_{ij})!(|\phi_j|-n_{ij})!(|V|-|\omega_i|-|\phi_j|+n_{ij})!} \tag{6}$$

with all parameters calculated the same as NMI. Unlike NMI, a random assignment of nodes into clusters will yield an AMI score of zero. We report both NMI and AMI because the validity of the assumptions of the random assignment model of AMI is not universally accepted [5]. However, AMI is better calibrated. We confirmed that the AMI score has a zero baseline by randomizing the clustering results from Infomap across experiments so that the number of clusters and their sizes stays the same while the samples are shuffled among them. Figure 11A illustrates both NMI and AMI scores of the randomized clustering results. Randomized clusters continuously yielded a mean AMI score of zero and a mean NMI score higher than 0.6. The mean NMI score increases with the number of clusters. Adjusting the score to calculate AMI solves this problem.

2.2.4. Statistical Power

We simulate a binary trait t_i for every cluster ω_i that has more than 10 members. The prevalence p_i^i of the trait t_i in cluster ω_i is significantly different than the prevalence P_i of the trait among all nodes (p-value$\sim 10^{-10}$). Statistical power measures the degree to which clustering algorithms can preserve this difference. It is calculated in 3 steps. First, for any extracted cluster ϕ_j, the prevalence of every simulated trait t_i in that cluster , p_i^j, is calculated. Second, for each trait t_i, the null hypothesis that the members of ϕ_j are as susceptible to test positive for t_i compared to other nodes in the graph (i.e., the difference between p_i^j and P_i is statistically insignificant) is tested using a binomial test. The negated logarithm of the null hypothesis probability (p-value) is recorded as $S(\phi_j,t_i)$ only if it passes the bonferroni significance threshold in 5,000 experiments (p-value$< 10^{-5}$). Third, for every cluster ϕ_j, we only consider the largest score ($S(\phi_j,t_i)$) value. Statistical power is then defined as the fraction of the maximum score (ground truth clustering score) retrieved by the clustering method. It is calculated using the following formula:

$$Power(\Phi,\Omega) = \frac{\sum_{j=1}^{M} \underset{i}{max}[-\log S(\phi_j,t_i)]}{Power(\Omega)} \tag{7}$$



Supplementary Fig 11. **The effects of score adjustments on the baseline score** (A) Mutual information score (NMI), and (B) statistical power score of a random assignment of nodes to clusters has a non-zero baseline, which increases with the number of clusters. Adjusting these scores results in a score of zero for the same clustering. Adjustment is done through (A) subtracting the expected mutual information from NMI (AMI score), and (B) only including the results that pass the significance threshold when calculating power score. The scores were calculated by generating a random assignment of nodes to the same number of clusters and cluster sizes found by Infomap for each of the 750 simulations done in the paper.

$$Power(\Omega) = \sum_{i=1}^{N} -\log S(\omega_i, t_i) \tag{8}$$

Statistical power ranges from zero to one. Random assignment of nodes to clusters results in a score close to zero; since the difference between the prevalence of any disease in clusters compared to the whole dataset becomes negligible. A clustering that exactly follows the ground truth results in a perfect score of one. Moreover, the value of this score cannot be optimized by extracting more fine-grained clusters as test scores that fail to pass the significance threshold are discarded, addressing the granularity problems with the purity and NMI scores. Figure 11B illustrates the effect of the significance threshold on the baseline power score using the same experiment we did for NMI/AMI comparison. Similar to NMI, a lack of correction for random results will increase the baseline statistical power score.

2.2.5. Modularity

Modularity measures the strength of clustering in terms of the density of links inside the clusters compared to external links connecting them [6]. It is desirable to have clusters densely connected within and sparsely connected to others [7]. Across all clusters, modularity measures the expected probability that any random link is located inside a cluster. It is calculated using the following formula:

$$Q = \sum_{i=1}^{M} (e_{ii} - a_i^2) \tag{9}$$

Where M is the number of clusters, e_{ii} is the fraction of edges that have both of their ends in the cluster i, and a_i is the percentage of edges that have at least one the their ends in the cluster i. Since modularity only measures the strength of a clustering, its calculation does not require ground truth information, in contrast with the previous metrics. We refer to the metrics that do not require ground truth information as feature-based metrics. Examination of modularity is essential since two of the algorithms analyzed in this paper use modularity optimization.

2.2.6. Other Feature-Based Metrics

In addition to modularity we analyze five other feature-based metrics. While modularity aims to quantify the overall strength of clusters, in terms of both their inner edge density, and outgoing connection sparsity, each of the following metrics only focus on one of the two.

- **Connectivity:** The percentage of nodes that are connected to more than half of other nodes in their cluster. This score offers an indirect way to measure the strength of the clustering. Being connected to more than half of a cluster's members is impossible for the nodes in a "cluster" that is made up of smaller clusters with minimal false-positive connections.
- **Coverage:** The percentage of intra-cluster edges out of all edges [8]. Coverage captures an estimation of the fraction of edges that are deemed true-positive by the clustering algorithm.

- **Inter-Cluster Edge Rate:** The percentage of all edges that connect different clusters. This metric can be used to infer about the percentage of available edges that should be false-positives for the clustering to be correct. Subtracting coverage score from its maximum of 1 (where all edges are covered by clustering) yields this score.
- **Missing Intra-Cluster Edge Rate:** The percentage of the edges that are missing from clusters, assuming they are cliques, compared to the total number of edges expected in a set of cliques with the same number of members. This metric can help illustrate the rate of false-negatives necessary for the clustering to be correct.
- **Highly Connected Rate:** Out of all clusters that have more than ten members, the percentage of those that are highly connected. We define highly connected clusters as the ones that posses more than half of the edges of a clique with the same number of nodes. This score is similar to connectivity, however it only concerns graph sizes that are important to statistical power calculations. Also it has a binary nature in that either all the nodes in the cluster pass the test or none of them will.

2.3. Clustering Algorithms

In this section, we describe the clustering algorithms evaluated in this study in detail. We have chosen five algorithms in three categories based on their methodology.

2.3.1. Min-Cut Based Method

Min-cut based approaches use consecutive min-cuts to iteratively divide a graph into subgraphs that are highly connected inside and have few connections to other subgraphs [9]. The Highly Connected Subgraphs (HCS) algorithm is a example of a min-cut based approach. The criteria to decide whether a subgraph is highly connected depends on the algorithm [10]. A simple example is defining highly connected subgraphs as the ones where each node is connected to at least half of the other nodes in the subgraph. Whenever a subgraph reaches this threshold, the algorithm considers it a cluster. DASH [11], the best known local IBD clustering tool, uses a modified version of HCS. To decide whether a subgraph is highly connected or not, DASH uses an optimization function which requires parameter tuning using *a priori* estimation of the rate false-negative and false-positive edges in the graph. To reduce the complexity of our experiments, and to have a fair comparison, we use the fast HCS algorithm described in [12], where a subgraph is called highly connected if a minimum of $|V|/2$ edges need to be removed to break it down into two disjointed subgraphs. Running HCS is equivalent to running DASH without passing the false-positive/false-negative information to the algorithm.

2.3.2. Modularity Optimizing Approaches

Modularity optimization approaches aim to maximize the modularity score function described in the metrics section [6]. Modularity optimization can uncover structures unknown *a priori* [13]. Moreover, high modularity is a preferred structural property for clusters as it indicates the strength of internal connections compared to external ones [14]. This in turn has generated interest in utilizing clustering approaches that are based on modularity optimization [3]. While optimizing modularity is NP-complete [14], greedy algorithms have proven to be fast and approximately accurate [3, 7, 13, 15]. Utilizing greedy heuristics, methods such as the Louvain algorithm achieve a polynomial runtime [13]. Louvain algorithm, the fastest modularity optimization method available today, can analyze the WebBase2001 database with 118 million nodes and 1 billion edges in 152 minutes, a task that takes other clustering algorithms more than 24 hours. We also analyze the Leiden algorithm [15], another clustering algorithm based on greedy optimization of modularity. In Leiden algorithm, Traag et al. have improved Louvain clustering heuristically in scenarios that cause the Louvain algorithm to find clusters that are poorly-connected, or entirely disconnected [15]. Both Louvain and Leiden allow for fine-tuning the granularity of the clusters. In this manuscript, we are reporting the finest-grained level of results for brevity. As described in the Results section, even the finest-grained level of clustering for these two algorithms suffers from under-clustering the nodes.

2.3.3. Information Theory Based Methods

Instead of focusing on the hierarchical structure of clusters and their connectivity, methods based on information theory focus on the flow of information among the nodes. These methods often analyze information flow using random walk probabilities between the nodes. We analyze two methods in this category. Both of these algorithms have been previously applied to bioinformatic problems successfully.

Infomap The Infomap algorithm is the most commonly used information theory based clustering method. It has been successfully utilized for clustering global IBD networks [16]. It aims to minimize the memory required to describe any random walk on the graph, also called path description length, through collapsing groups of vertices

that are more likely to appear consecutively in a random walk into a single vertex. These groups are then labeled as clusters. Infomap uses an objective function called the Map Equation to approximately minimize the path description length [17]. This approximation approach helps infomap to scale to large graphs [3]. For every pair of nodes connected by an edge, Infomap checks whether collapsing them into a single node would help minimize the result of the map equation (and by definition, the path description length) or not; if so, it puts them in the same cluster.

Markov Clustering The Markov Clustering (MCL) algorithm uses flow dynamics to simulate random walks on the graph until they converge to a steady state [18]. To do so, it employs matrix multiplication to simulate the destination probabilities of a single step of a random walk and then inflates those probabilities to weaken the unstable paths in an effort to guarantee that the series of random walk simulations will always converge to a specific, non-random group of clusters. The elimination rate of the lower-probability paths is controlled through a parameter called the inflation rate with a range of 1.2 to 5.0. Theoretically, setting a higher inflation rate will result in finer-grained clusters with dense connections. We explore four inflation rates: $1.5, 2, 3, 5$ in the results section. We use subscripts to denote which parameter was used to run MCL: $MCL_{1.5}, MCL_2, MCL_3, MCL_5$. MCL algorithm is both faster than Infomap and more customizable through its parameters. It could also benefit from parallelization both on CPU and GPUs. Further, MCL is a common method for clustering protein networks [19].

References

1. H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39 (Cambridge University Press Cambridge, 2008).
2. D. J. MacKay and D. J. Mac Kay, *Information theory, inference and learning algorithms* (Cambridge university press, 2003).
3. A. Lancichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," Phys. review E **80**, 056117 (2009).
4. *Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?* (Association for Computing Machinery, New York, NY, USA, 2009).
5. M. Meilă, "Comparing clusterings—an information based distance," J. multivariate analysis **98**, 873–895 (2007).
6. M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," Phys. review E **69**, 026113 (2004).
7. S. Emmons, S. Kobourov, M. Gallant, and K. Börner, "Analysis of network clustering algorithms and cluster quality metrics at scale," PloS one **11** (2016).
8. *Visualizing Graphs as Maps with Contiguous Regions.*
9. M. Stoer and F. Wagner, "A simple min-cut algorithm," J. ACM (JACM) **44**, 585–591 (1997).
10. Springer, *Finding highly connected subgraphs*.
11. A. Gusev, E. E. Kenny, J. K. Lowe, J. Salit, R. Saxena, S. Kathiresan, D. M. Altshuler, J. M. Friedman, J. L. Breslow, and I. Pe'er, "Dash: a method for identical-by-descent haplotype mapping uncovers association with recent variation," The Am. J. Hum. Genet. **88**, 706–717 (2011).
12. E. Hartuv and R. Shamir, "A clustering algorithm based on graph connectivity," Inf. processing letters **76**, 175–181 (2000).
13. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," J. statistical mechanics: theory experiment **2008**, P10008 (2008).
14. U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," IEEE transactions on knowledge data engineering **20**, 172–188 (2007).
15. V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: guaranteeing well-connected communities," Sci. reports **9**, 1–12 (2019).
16. G. M. Belbin, S. Wenric, S. Cullina, B. S. Glicksberg, A. Moscati, G. L. Wojcik, R. Shemirani, N. D. Beckmann, A. Cohain, E. P. Sorokin *et al.*, "Towards a fine-scale population health monitoring system," bioRxiv p. 780668 (2019).
17. M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," Proc. National Acad. Sci. **105**, 1118–1123 (2008).
18. S. V. Dongen, "Graph clustering by flow simulation," Ph.D. thesis, University of Utrecht Amsterdam, Netherlands (2000).
19. A. J. Enright, S. Van Dongen, and C. A. Ouzounis, "An efficient algorithm for large-scale detection of protein families," Nucleic acids research **30**, 1575–1584 (2002).