Proteinarium is a data analysis and visualization tool that helps identify clusters of patients with similar genomic data derived from protein-protein interactions from an input gene set. Additionally, in clusters where two distinct groups can be identified, for example in a case-control study, the network analysis tool provides a measure of both positive and negative correlation through the differences of the respective graphs. All individual networks are inferred by taking every pair of genes in each patient and finding the highest confidence protein pathway with those genes as endpoints and summarizing these results across patients for summary networks which are displayed to provide a more intuitive method of visualizing results.

Initialization of the Program

Either one or two groups of patient gene sets must be provided in the form of a Group 1 and Group 2 gene set file (see Appendix B for file formats). Each file consists of a group of patients and their associated gene sets, which we load for further analysis through protein interaction analyses and similarity clustering.

We initialize the database by going through STRING's protein interactome for humans. Each interaction is undirected and has a confidence score between 0 and 1000; the higher the number, the higher the confidence in the specific protein-protein interaction. For each protein, we find the associated gene name, using the associated HGNC Symbol.

Pairwise Path Finding

For each patient, we identify all proteins associated with the patient's gene set and perform Dijkstra's shortest path algorithm between every pair of proteins. We define the cost of an edge between two proteins in the interactome with STRING confidence score S to be $1000 - S$, which indicates the highest confidence interactions should be the lowest cost edges. The algorithm has been modified to only consider paths whose lengths (number of vertices in the path) and total costs (sum of edge weights) are below the values specified in the configuration options. The purpose is not to find any path between each protein pair, but rather to find short, high-confidence paths between all pairs of proteins where such a path exists.

After the pairwise paths have been computed for a given set of parameters (minimum interaction confidence, maximum path cost, maximum path length), we save the data in a text file for future use to save computation time on later runs of the program. If any of those three parameters change between runs, the pairwise paths will need to be recalculated. This data is stored in the specified folder as <Project Name>_Data.txt. To force recalculation of paths, simply delete this file; however, do not modify the file manually.

For each patient's gene set that we perform this pairwise path finding, we construct a graph consisting of the set of edges contained in all paths for that gene set. We refer to this simple graph of gene set i as $G_i = (V_i, E_i)$ where each vertex $v \in V_i$ corresponds to a protein and each edge $e \in E_i$ corresponds to a protein interaction. Only those proteins that are contained within the paths found are included in the set of vertices V_i; if a protein of a gene in the original input gene set is not found in any path, it will be omitted from the resulting graph.

Path-Aware Graph Reduction

Whenever a graph is to be visualized, we reduce the number of displayed vertices according to the configuration options so as to keep the visualization tractable. In particular, we rank the vertices of a given

graph corresponding to input genes (either Group 1 or Group 2) according to the number of pairwise paths that particular vertex appears in. We then retain the top *k* vertices from this list such that the total number of unique vertices found within all pairwise paths of *k* selected vertices is less than or equal to the maximum number of vertices to be displayed. In doing so, all graphs will show only complete paths whose endpoints originate from the input gene sets. Note that this process is only ever performed at the very end before visualizing any given graph and performing statistical analyses on the displayed graph.

Clustering Algorithm

After obtaining the graphs for all patient gene sets in both the Group 1 and Group 2, we cluster the set of graphs hierarchically using Unweighted Pair Group Method with Arithmetic Mean (UPGMA). We use the Jaccard distance as our distance metric between any two graphs G_i and G_j:

$$d_{i,j} = 1 - \frac{|G_i \cap G_j|}{|G_i \cup G_j|}$$

Initially, all of our clusters consist of a single graph, corresponding to a leaf of the dendrogram. While there exists more than one cluster, we combine the two clusters that are closest to each other according the distance metric d and then update all cluster distances. When combining two clusters i and j, the new distance to a third cluster k is given as follows:

$$d_{ij,k} = \frac{w_i d_{i,k} + w_j d_{j,k}}{w_i + w_j}$$

In the standard UPGMA algorithm, the weight w_i would correspond to the number of graphs in that cluster, so all the weights of all leaves in the tree would equal 1. Our modified algorithm scales either the Group 1 graph factor or the Group 2 graph factor to a constant $\rho \geq 1$ such that the product of the size of the graph set and ρ is equal to the size of the other graph set. This is done for weighting purposes: if we have 100 Group 1 gene sets and 50 Group 2 gene sets, each Group 2 gene set leaf has weight 1 whereas each Group 2 gene set leaf has weight 2.

We output the final dendrogram created by the above clustering algorithm according to the Newick tree format. Any cluster that is further visualized has the corresponding portion of the dendrogram similarly outputted. See below for exact file format.

Clustering Visualization

In the dendrogram visualization, the horizontal length of the line segments is proportional to the height of the cluster in the tree. The lower the height, the more similar the graphs within the cluster are. All heights are normalized between 0 and 1, with the leaves occupying height 0 and the ancestor occupying height 1. The lines are also colored according to the (weighted) percentage of Group 1 versus Group 2 gene sets comprising the cluster: if a group comprises more than 60% of the weight, then the edge is colored by that group; otherwise, the edge is colored black.

Additionally, to the left of the dendrogram is the meta-clustering bar in which the leaves are grouped into easier-to-visualize spans along the dendrogram according to [metaClusterThreshold], labeled with the cluster identifier that contains all the leaves.

In all cases, if a cluster is deemed significant according to the Fisher Exact Test (p-value less than or equal to [significanceThreshold]; see below for details), then the cluster identifier will be colored red as opposed to black.

Annotated Graph Construction

For a given cluster C comprised of n graphs $G_1 = (V_1, E_1), \ldots, G_n = (V_n, E_n)$ we construct the summary layered graph $LG = (V_{LG}, E_{LG})$ consisting of each of the n graphs with the following vertices and edges:

$$V_{LG} = V_1 \cup V_2 \cup \ldots \cup V_n$$
$$E_{LG} = E_1 \cup E_2 \cup \ldots \cup E_n$$

Additionally, the count annotation for each vertex v in LG:

$$LG.\mathrm{count}(v) = \sum_{i=1}^{n} \mathbf{1}_{V_i}(v)$$

Within our application, the graphs $G_1 \ldots G_n$ of a cluster C composed of n graphs refer to the n protein network graphs constructed during the pairwise path finding step described above, one per patient in the cluster. The count annotation similarly becomes the number of patient networks in which a protein of the layered graph is found.

Annotated Graph Subtraction

If a branch of the dendrogram carries samples from *both* groups, then there are 5 possible networks being created on the fly. These networks are as follows: (i) [Group 1] shows the network for only the group1 samples in that branch; (ii) [Group 2] shows the network for only group 2 samples; (iii) [Group 1 plus Group 2] shows the network for combination of group1 and group 2; (iv) [Group 1 minus Group 2] shows the network where group 2 is subtracted from group 1; (v) [Group 2 minus Group1] shows the network where group 1 is subtracted from group 2. It is also possible to choose a specific sample from the dendrogram and visualize the associated network.

Given two layered graphs LG_1 and LG_2, we define the subtraction operation $LG_1 - LG_2$ to be: for each vertex $v \in LG_1$, if either (1) $v \notin LG_2$ or (2) the scaled count of v in LG_1 minus the scaled count in LG_2 is greater than 0.

```
LayeredGraph subtract(LG1, LG2):
1.  G = {vertices, edges}
2.  for v in LG1.vertices:
3.      If LG1.count(v) > LG2.count(v):
4.          G.vertices.add(v, LG1.count(v) – LG2.count(v))
5.  for e in LG1.edges:
6.      If e.source in vertices and e.target in vertices:
7.          G.edges.add(e)
8.  remove extraneous vertices from G.vertices according to G.edges
9.  return G
```

Additionally, we scale the counts of the left-hand-side `LG1.count(v)`or the right-hand-side `LG2.count(v)` with the same parameter ρ as defined above in the description of the clustering algorithm.

Annotated Graph Visualization

Given a layered graph LG corresponding to a cluster C, we lay the vertices according to a simple implementation of a force-directed layout algorithm. For each vertex $v_i \in LG$, we assign its radius r_i linearly according to degree between some specified minimum and maximum radii. Let d_{ij} be the Euclidean distance between the center points of two vertices, v_i, v_j. For any neighboring pair of vertices v_i, v_j, we define the following spring-based attractive force: $F_A = \lambda_A d_{ij}$. For every pair of vertices v_i, v_j, we define the following electrostatic repulsive force: $F_R = -\frac{\lambda_R r_i r_j}{d_{ij}^2}$. We then apply the force $F_A + F_R$ to v_i at the angle between v_i and v_j. Doing this for all vertices constitutes 1 iteration, and we continue iterations until either $[\text{maxIterations}]$ or $[\text{maxTime}]$ is exceeded, or when the total distance updates over all vertices is less than $[\text{deltaThreshold}]$.

Additionally, we color each vertex v according to the presence of v's gene in the various input data sets:
If v's gene is in…

1. only the Group 1 gene set: orange $[\text{group1VertexColor}]$
2. only the Group 2 gene set: blue $[\text{group2VertexColor}]$
3. both Group 1 and Group 2 gene sets: green $[\text{bothGroupsVertexColor}]$ or a 50/50 mixture of (1) and (2)
4. imputed (appears in neither Group 1 nor Group 2 gene sets but inferred from the pairwise path finding algorithm): red $[\text{defaultVertexColor}]$

Exception to the above rules: if a gene is both in one of the input gene sets (WLOG, Group 1) and imputed during the pairwise path-finding of the other group (Group 2), then it is possible that the gene still appears in the (Group 2 – Group 1) graph. An intuitive explanation for such a situation is that the imputed presence of the gene in Group 2's pairwise paths is more important than its presence in Group 1's input gene set. As such, genes that fall under this category will be colored red to indicate its greater importance as an imputed gene. At the same time, to acknowledge that the gene was in fact present in Group 1's input set, the border of the vertex would be orange.

Additionally, the opacity of each vertex v is calculated linearly between a minimum opacity $m = [\text{minVertexAlpha}]$ (default of 50) and 255 according to the following formula:

$$\text{opacity}(v) = m + (255 - m) * \frac{LG.\text{count}(v)}{\max_{v'} LG.\text{count}(v')}$$

Note that an opacity of 0 is transparent and an opacity of 255 is opaque. The same is done for the opacity of the edges, except the maximum count of either vertex of the edge is used in the numerator of the linear scaling.

Bootstrapping Confidence Values

The bootstrapping value is a value of confidence in a given cluster; the higher the bootstrapping value, the less likely it is that the patients clustered together in that exact way by random chance. We implement the bootstrapping as originally described by Felsenstein[1]. For a given phylogenetic tree with n patients, let the set of unique proteins across all patients be denoted as $P = \{p_1, p_2, \ldots, p_m\}$. We can then construct a binary matrix B where $B_{i,j}$ is a 1 if patient i has protein p_j in her protein set. To generate new bootstrapped phylogenetic trees, we sample with replacement m protein columns of the binary matrix to create an alternative binary matrix, then perform the same UPGMA algorithm as described above. We then record whether each cluster of the original phylogenetic tree appears again in the new, bootstrapped tree. The more often it appears in the bootstrapped tree after resampling, the greater the bootstrapping confidence value. We repeat this process $[\text{bootstrappingRounds}]$ times, which defaults to 1000.

Fisher's Exact Test p-Value

The p-value for a cluster is given by a Fisher-Exact and indicates the probability of observing, among all *n* patient samples, a cluster of size *m* with proportions of Group 1 and Group 2 patients relative to the total number of Group 1 and Group 2 patient samples. In other words, it provides a sense of how disproportionate a particular group (either Group 1 or Group 2) is over-represented in a cluster; the lower the p-value, the more confidence we have that the over-representation is *not* due to random chance.

Clustering Coefficients

Additionally, for a given cluster, we calculate the global clustering coefficients for each of the four graphs: the Group 1 graph, the Group 2 graph, and the two graphs resulting from the weighted differences of each graph as described above. The clustering coefficient is a measure describing the tendency of vertices in a particular graph to cluster together. For a graph $G = (V, E)$ with $|V| = n$ vertices, the global clustering coefficient is given by

$$\frac{1}{n}\sum_{i=1}^{n} \frac{\left|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}\right|}{|N_i| * (|N_i| - 1)}$$

where N_i is the set of neighbors of vertex v_i, i.e. $N_i = \{v_j : e_{ij} \in E\}$.

[1] Efron, B., et al. (1996). Bootstrap Confidence Levels for Phylogenetic Trees. *Proceedings of the National Academy of Sciences, 93*(23), 13429. doi:10.1073/pnas.93.23.13429

Configurability

Virtually all parameters used at every point in the execution of the program is configurable. The five categories of configuration are below; the number in parenthesis is the number of configurable options within that category.

1. General Configuration (9)
2. Analysis Configuration (5)
3. Force-Directed Layout Configuration (7)
4. Renderer Configuration (11)

Of the 36 options, only one *must* be provided: within the general configuration, the file path to the Group 1 gene set data, $[\mathrm{group1GeneSetFile}]$.

Appendix A – Basic Configurable Options.

General Configuration

- *activeDirectory* – directory from which all non-absolute paths will be resolved
  Default: <empty_string> (i.e. the directory the program is called from)
- *outputDirectory* – directory path for output to be saved
  Default: folder called "output" within *activeDirectory*
- *group1GeneSetFile* – file path that contains gene set information for Group 1
  Required
- *group2GeneSetFile* – file path that contains gene set information for Group 2
  Default: <empty_string>
- *projectName* – name of project; will be used as prefix for outputted file names
  Default: calculated based on *activeDirectory* and *group1GeneSetFile*

Analysis Configuration

- *minInteractomeConfidence* – the minimum STRING score (an integer between 0 and 1000) for a protein interaction to be considered in pairwise paths
  Default: 0
- *maxPathCost* - the maximum path cost for a pairwise path to be considered valid. A path consisting of interactions with confidence scores $c_1, c_2, \ldots, c_n$ has a path cost:

  $$cost = \sum_i 1000 - c_i$$

  Note that this path cost scheme the maximum confidence score to be 1000, as it is with default STRING downloads; otherwise Dijkstra's algorithm, as implemented, is not guaranteed to find the lowest cost path.
  Default: 200
- *maxPathLength* – the maximum number of vertices in a path. The maximum number of interactions, edges between source and target vertices, is $maxPathLength - 1$
  Default: 5
- *fractionOfVerticesToRender* – the fraction, between 0 and 1, of vertices to render in any graph
  Default: 1

- *maxVerticesToRender* – the maximum integer number of vertices to render in any graph
  Default: 2147483647 (maximum 32-bit signed integer)
- *bootstrappingRounds* – the number of rounds of bootstrapping to perform. If specified, must be greater than 100. A value of 0 indicates no bootstrapping
  Default: 1000

Note: it is wise to set at least one of *fractionOfVerticesToRender* or *maxVerticesToRender*; otherwise, the program will attempt to render a large number of vertices, taking a long time to compute and most likely yielding uninterpretable results.

Renderer Configuration

- *displayRendering* – if false, images of the dendrogram and graphs will only be exported but not displayed to the user; this can provide a small speed improvement
  Default: true
- *significanceThreshold* – the p-value, between 0 and 1, at or below which clusters should be considered significant
  Default: 0.05
- *metaClusterThreshold* – the height, between 0 and 1, at which clusters should be grouped into meta-clusters; the higher the height, the larger the metaclusters
  Default: 0.3333

Appendix B – File Formats.

The input gene set group file format is a series of individual patient gene sets, one per line:
<patient_id1>=<gene_symbol>,<gene_symbol>,<gene_symbol>,…,<gene_symbol>
<patient_id2>=<gene_symbol>,<gene_symbol>,<gene_symbol>,…,<gene_symbol>
Exported files are either standard image files, .csv files, simple .txt tab-delimited tables, or .txt Newick tree files.

The Newick tree file format is as follows:

$$\big((\text{leafA}: 0, \text{leafB}: 0): \text{clusterABheight}, (\text{leafC}: 0, \text{leafD}: 0): \text{clusterCDheight}\big): \text{clusterABCDheight}$$

and can be arbitrarily nested. Programs that expect dendrogram files will recognize this format.

Appendix C – Advanced Configurable Options.

Advanced Configuration Options (most users will **not** need to modify these)

General Configuration

- *reusePreviousData* – if set to false, will force the program to discard any previous pairwise path data
  Default: true
- *calculateGraphDifferences*– whether or not graph differences (Group1 – Group2 and Group2 – Group1) should be calculated, rendered, displayed, and exported
  Default: true

- *proteinInteractomeFile* – an **absolute** file path to a downloaded STRING protein interactome; if the file is compressed with GZIP, it must have the .gz extension
  Default: installation directory + STRING download's file name
- *proteinAliasesFile* – an **absolute** file path to a downloaded STRING protein aliases list; if the file is compressed with GZIP, it must have the .gz extension
  Default: installation directory + STRING download's file name

Force-Directed Layout Configuration

- *repulsionConstant* – the repulsion constant λ_R used in force-directed layout equations
  Default: 0.2
- *attractionConstant* – the attraction constant λ_A used in force-directed layout equations
  Default: 0.0003
- *minVertexRadius* – the minimum radius of vertices in the graphs being rendered
  Default: 15
- *maxVertexRadius* – the maximum radius of vertices in the graphs being rendered or -1 for dynamically determined maximum
  Default: -1

Stopping Conditions for Force-Directed Layout algorithm

- *deltaThreshold* – stop when the total absolute change summed over all vertices is less than or equal to *deltaThreshold*
  Default: 0.001
- *maxIterations* – the maximum number of iterations to update vertex positions
  Default: 10000
- *maxTime* – the maximum number of milliseconds to spend laying out the vertices
  Default: 9223372036854775807 (maximum 64-bit signed integer)

Renderer Configuration

- *minVertexAlpha* – the minimum alpha value (between 0 and 255) of vertices in the graph
  Default: 50
- *minEdgeAlpha* – the minimum alpha value (between 0 and 255) of edges in the graph
  Default: 50
- *drawGeneSymbols* – boolean flag indicating if gene symbols should be written on vertices
  Default: true
- *colorSignificantBranchLabels* – boolean flag indicating if cluster identifiers should be colored red if deemed significant (true), or should always be black (false)
  Default: true
- *defaultVertexColor* – the default color for vertices in a graph
  Default: (255,0,0) (red)
- *group1VertexColor* – the color for vertices whose gene is in from a Group 1 gene set
  Default: (255,200,0) (orange)

- *group2VertexColor* – the color for vertices whose gene is in from a Group 2 gene set
  Default: (0,0,255) (blue)
- *bothGroupsVertexColor* – the color for vertices whose gene is in both Group 1 and 2 gene sets
  Default: (0,255,0) (green)

Color Format: color configuration options are given by "(R,G,B)" where each of R, G, and B are replaced with an integer between 0 and 255 for red, green, and blue components