Proteinarium: ***Multi-Sample Protein-Protein Interaction Visualization and Analysis Tool***

David Armanious[1], Jessica Schuster[2,3], Sorin Istrail[1,4], James Padbury[2,3,4], Alper Uzun[2,3,4]*

1) Computer Science, Brown University, Providence, RI; 2) Pediatrics, Women and Infants Hospital, Providence, RI; 3) Pediatrics, Warren Alpert Medical School, Brown University, Providence, RI; 4) Center for Computational Molecular Biology, Brown University, Providence, RI.

* E-mail: alper_uzun@brown.edu

Abstract

Data visualization has become very crucial in the post genomic era where the accumulation of genomic information is mounting exponentially. Visualizing protein interactions in the context of the interactome is an essential application, especially in association with disease phenotypes. Here, we describe Proteinarium, a multi-sample protein-protein interaction network visualization and analysis tool to identify clusters of samples with genomic data derived from protein-protein interactions. Proteinarium is a command-line tool written in Java with no external dependencies and it is freely available at https://github.com/Armanious/Proteinarium.

1 Introduction

Data visualization is an important application in the post genomic era where the pace of accumulation of genomic data is exponential. Biological data visualization tools assist researchers in understanding the hidden layers of relationships between diseases and their associated phenotypes. This approach helps researchers to identify disease specific biological networks and subnetworks dynamically through the analysis of experimental data. Recently, with the generation of high throughput screening methods, extensive protein–protein interaction (PPI) networks have been built[1,2]. Large scale protein interaction maps show that genes involved in related phenotypes frequently interact physically at the level of proteins in model organisms, and as well as in humans[3-7]. This is also true for the disease phenotypes. Proteins that are associated with similar disease phenotypes have a strong affinity to interact with each other[5,8]**.** In addition, they have a tendency to cluster in the same network zone[9]**.** We developed Proteinarium, a multi-sample PPI visualization tool to analyze the networks for complex diseases and their associated phenotypes.

2 Materials and Methods

Proteinarium has three different input options. Option 1 requires the input in groups, e.g. Group 1 and Group 2. Each group is a text file that has the list of samples and each sample has their own gene list. This option is especially useful for case-control studies. Option 2 requires only one group. Option 3 supports a single gene list that can be used for analyzing the PPI network directly. For all

three options, Proteinarium needs gene list(s). These gene lists may be derived from transcriptome analysis, whole exome sequence data or from any analysis that generates a gene list of interest. Since Proteinarium was specifically designed to analyze multi-sample gene lists, its strength lies in use of gene lists for each sample as an input which are further analyzed through protein interaction analyses. Similarity clustering, where the network-based relations of samples are identified, is then presented as a dendrogram. A flow chart describing the steps is shown in **Figure 1**.

Proteinarium can also be used for single gene sets. The latter is simply based on the gene list that is provided and the resultant PPI network is built. There are several configuration options in Proteinarium (*f*or configuration options see details on GitHub, *https://github.com/Armanious/Proteinarium/blob/master/Configuration.pdf*). Users can make modifications to the maximum number of vertices in a path or define the maximum integer number of vertices to render in any graph.

2.1 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 in Supplemental Material). 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[7]. 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[10].

2.2 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[11]. 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.

2.3 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.

2.4 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)[12]. 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.

2.5 Cluster 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.

2.6 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.

2.7 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.

2.8 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 $[\mathrm{maxIterations}]$ or $[\mathrm{maxTime}]$ is exceeded, or when the total distance updates over all vertices is less than $[\mathrm{deltaThreshold}]$.

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 $[\mathrm{group1VertexColor}]$
2. Only the Group 2 gene set: blue $[\mathrm{group2VertexColor}]$
3. Both Group 1 and Group 2 gene sets: green $[\mathrm{bothGroupsVertexColor}]$ 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 $[\mathrm{defaultVertexColor}]$

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.

2.9 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[13]. For a given phylogenetic tree with n patients, let the set of unique proteins across all patients be denoted as $P = \{p_1, p_2, \dots, p_m\}$. 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 [bootstrappingRounds] times, which defaults to 1000.

2.10 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.

2.11 Clustering Coefficients

Additionally, for a given cluster, we calculate the global clustering coefficients for each of the four graphs: Group 1 graph, 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

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

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

2.12 Outputs

Proteinarium has several outputs:

(1) A dendrogram that shows how samples are clustered based on network similarities;

(2) The ability to display any network from any branch of the dendrogram. This functionality enables users to visualize networks for Group 1, Group 2 or any sample. Since each branch of the dendrogram was built based on network similarities of the samples, any branch can be analyzed and visualized. If a branch of the dendrogram carries samples from *both* groups, then there are 5 possible networks being created on the fly. Please see details in Methods.

(3) Proteinarium provides a gene list file of visualized networks. The file includes information about the genes, if they are imputed or not and the number of samples in which each gene was detected in the network.

(4) Proteinarium provides the PPI interaction file as an output where users can analyze the network(s) of interest on other platforms such as Cytoscape or Gephi[14,15]**.** In addition, since the dendrogram is written in Newick tree format, users can visualize it on other platforms like Dendroscope[16]**.**

2.13 Program Availability

Proteinarium is a command-line tool written entirely in Java with no external dependencies. Java version 1.8 or above (Java 9, or Java 10) must be installed

292 in order to run Proteinarium. It is freely available at
293 https://github.com/Armanious/Proteinarium.
294
295 **Results**
296 In order to test how this tool performs with noisy data, simulations were performed.
297 We simulated two groups each with 50 samples (**Figure 2**). For each sample in
298 group 1, 10 genes were randomly selected from a PPI network 1 and the same
299 was done for samples in group 2 with genes selected from a PPI network 2. We
300 particularly choose these PPI networks because their members are not
301 overlapped. We ran Proteinarium on these 100 Samples. For Group 1, 20 % of the
302 samples were randomly deleted and an increasing percentage of their genes were
303 replaced with genes from PPI network 2 (20%, 30%, 40% and 50%). The same
304 was done for Group 2, replacing genes with randomly selected genes from PPI
305 network 1. In these simulations we observed that samples from Group 1 cluster
306 closer together and samples from Group 2 clustered together. As expected the
307 more noise added to the data sets, the greater number clusters containing samples
308 from both groups.
309
310 **Conclusion**
311 In conclusion, we have introduced a multi-sample protein-protein interaction
312 network visualization tool to support analysis of single or paired samples. The
313 tool allows investigators address important questions on high throughput data for
314 a variety of disease phenotypes based on their associated PPI networks. In

addition, Proteinarium provides several different outputs with more than 30 configurable options.

Acknowledgements

We thank Stephen Kidd from the Department of Classics at Brown University for naming the tool. We thank the Center for Computation and Visualization (CCV) and Computational Biology Core at Brown University. This work was supported by the National Institutes of Health (grants 5P20GM109035-04, 5P30GM114750) and the Kilguss Research Core at Women & Infants Hospital.

References

1 Lage, K. Protein-protein interactions and genetic diseases: The interactome. *Biochim Biophys Acta* **1842**, 1971-1980, doi:10.1016/j.bbadis.2014.05.028 (2014).

2 Xing, S., Wallmeroth, N., Berendzen, K. W. & Grefen, C. Techniques for the Analysis of Protein-Protein Interactions in Vivo. *Plant Physiol* **171**, 727-758, doi:10.1104/pp.16.00470 (2016).

3 Stark, C. *et al.* BioGRID: a general repository for interaction datasets. *Nucleic Acids Res* **34**, D535-539, doi:10.1093/nar/gkj109 (2006).

4 Li, S. *et al.* A map of the interactome network of the metazoan C. elegans. *Science* **303**, 540-543, doi:10.1126/science.1091403 (2004).

5 Oti, M., Snel, B., Huynen, M. A. & Brunner, H. G. Predicting disease genes using protein-protein interactions. *J Med Genet* **43**, 691-698, doi:10.1136/jmg.2006.041376 (2006).

6 Gavin, A. C. *et al.* Proteome survey reveals modularity of the yeast cell machinery. *Nature* **440**, 631-636, doi:10.1038/nature04532 (2006).

7 Szklarczyk, D. *et al.* STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic Acids Res* **47**, D607-D613, doi:10.1093/nar/gky1131 (2019).

8 Chen, Y. *et al.* Variations in DNA elucidate molecular networks that cause disease. *Nature* **452**, 429-435, doi:10.1038/nature06757 (2008).

9 Barabasi, A. L., Gulbahce, N. & Loscalzo, J. Network medicine: a network-based approach to human disease. *Nat Rev Genet* **12**, 56-68, doi:10.1038/nrg2918 (2011).

10 Yates, B. *et al.* Genenames.org: the HGNC and VGNC resources in 2017. *Nucleic Acids Res* **45**, D619-D625, doi:10.1093/nar/gkw1033 (2017).

11 Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik* **1**, 269–271, doi:10.1007/BF01386390 (1959).

12 Sokal R, M. C. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin* **38**, 1409–1438 (1958).

13 Efron, B., Halloran, E. & Holmes, S. Bootstrap confidence levels for phylogenetic trees. *Proc Natl Acad Sci U S A* **93**, 13429-13434 (1996).

14 Shannon, P. *et al.* Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res* **13**, 2498-2504, doi:10.1101/gr.1239303 (2003).

15 Bastian M., H. S., Jacomy M. in *International AAAI Conference on Weblogs and Social Media* (2009).

16 Huson, D. H. & Linz, S. Autumn Algorithm-Computation of Hybridization Networks for Realistic Phylogenetic Trees. *IEEE/ACM Trans Comput Biol Bioinform* **15**, 398-410, doi:10.1109/TCBB.2016.2537326 (2018).

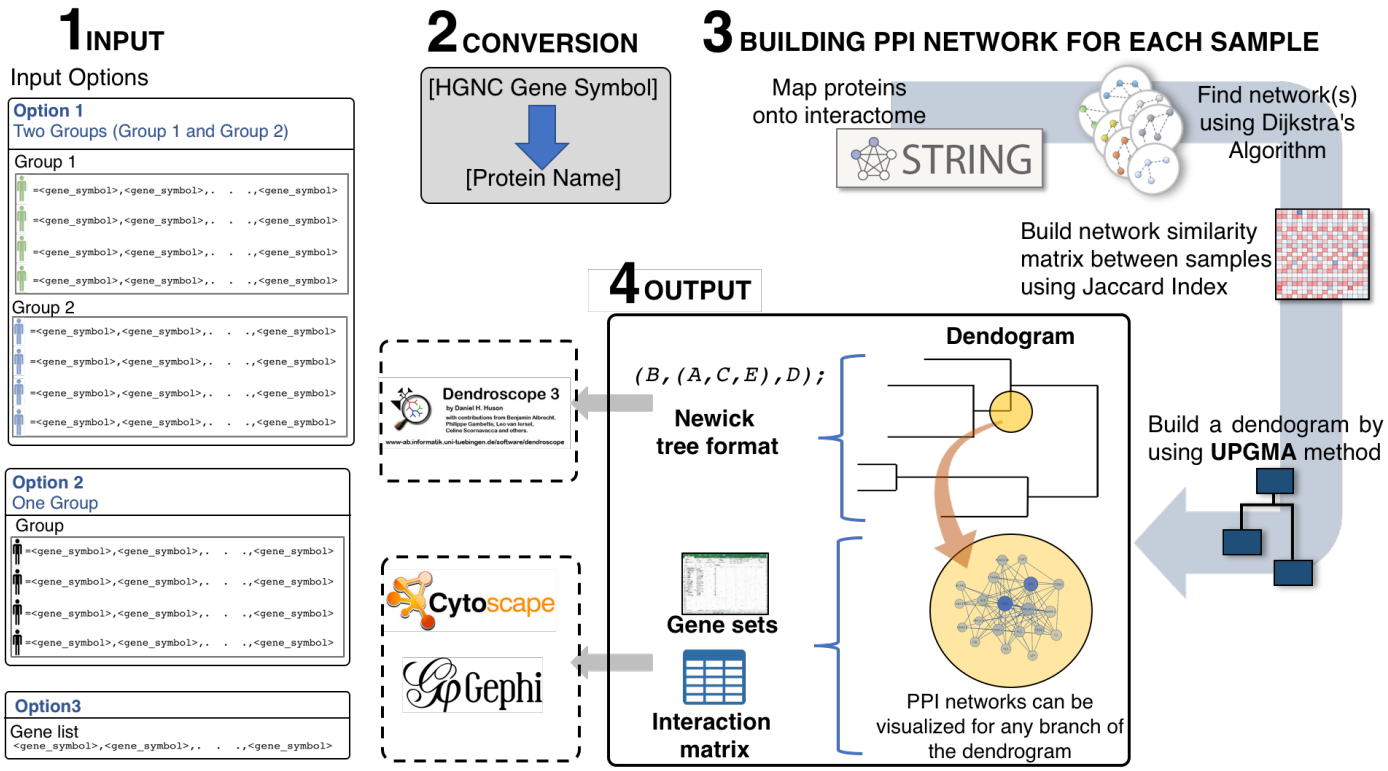

Figure 1. Flow chart shows the steps for Proteinarium. **(1)** There are three input options to run Proteinarium. Option 1 inputs groups of samples and each sample has its own gene list. Genes are represented as gene symbols. Option 2 and Option 3 inputs single groups as gene lists. Genes are represented as symbols. **(2)** All gene symbols are converted to their protein product names. **(3)** The PPI network is built by mapping proteins onto the interactome provided by STRING database, followed by finding the networks for each sample based on Dijkstra's algorithm. Next the network similarity matrix between samples is built using Jaccard index and building the dendrogram based on UPGMA method. **(4)** Proteinarium displays a dendrogram. Any chosen branch of the dendrogram displays the PPI networks which can be visualized on the fly and saved as an interaction file. Proteinarium provides several output files for users to export the results for use with other programs like Cytoscape, Gephi and Dendroskope 3.

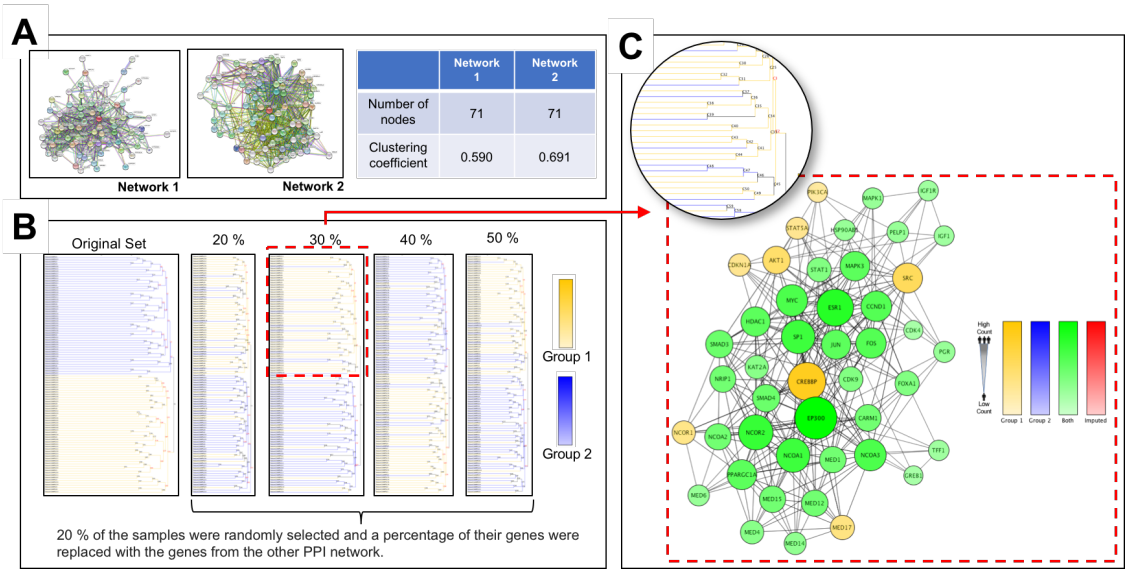

Figure 2. (A) STRING database was used to obtain networks 1 and 2. Both networks have total of 71 proteins and their clustering coefficients were respectively, 0.590 and 0.691. Members of the networks are not overlapped. **(B)** Two groups (Group1 and Group2) each with 50 samples were simulated. For samples in group 1, 10 genes were randomly selected from PPI network 1 and the same was done for samples in group 2 with genes selected from PPI network 2. This was called "Original Set". In order to test how our tool performs with noisy data, additional simulations were performed. For Group 1, 20 % of the samples were randomly deleted and an increasing percentage of their genes were replaced with genes from network 2 (20%, 30%, 40% and 50%). The same was done for Group 2, replacing genes with randomly genes from network 1. In the dendrograms, Group 1 samples are represented in yellow, and Group 2 samples were represented in blue. The five dendrograms show the clustering of the samples based on their network similarities after admixture. Please see supplemental file for larger view of the five dendrograms. **(C)** For representative purposes, we chose the dendrogram with 30 % replaced genes. The red dotted area on the dendrogram shows the cluster the PPI network was based on. In the PPI network, while yellow colored nodes represent the proteins existing only in group1, green colored nodes represent that these proteins existing in both groups. The size of the nodes is based on their number of connections. The intensity of the yellow and green colors signifies the number of samples that share the same node.