Application of Markov Stability for graph-based clustering on protein-protein interaction networks

By Peter von Holy

Supervisors: Dr Dov Stekel, Dr Adam Blanchard

Thesis submitted to the University of Nottingham for the Degree of Master of Research
September 2022

Contents

Abstract

Protein-Protein interaction networks are one of the most well-explored and documented parts of the interactome, as such, they have had a variety of databases and analyses developed for them, in order to harness this highly useful abstraction of very complex systems. Community detection is a popular analysis for many datasets which can be abstracted onto graphs and otherwise is a concept still performed on non-graph-based datasets through clustering methods. Community detection can also be performed at varying scales through the introduction of artificial time parameters, which in this case is a result of the use of a measure called Markov Stability. Markov Stability is also used as a measure to define a good graph partition but optimizing by having it be the objective function of the Louvain algorithm. In this study, we implement a framework for multiscale community detection governed by Markov stability, which has been previously used in other studies and apply this framework to an example protein-protein network of the proteins related to the 20 most frequently mutated human cancer genes from the STRING database. The results of this application are then explored and we show that due to the underlying properties of the example, robust partitions are obtained across varying Markov times.

Acknowledements

Many people have been a part of my Bioinformatics Scientist apprenticeship at the University of Nottingham and I would like to acknowledge all of that support here. I would like to thank the University of Nottingham School of Veterinary Medicine and Science for providing the facilities and virtual environments provide to partake in my Apprenticeship. I would like to thank the University of Nottingham libraries for all of the support provided when requiring research material as well as guidance on scientific writing. I would like to thank both my Supervisors Dr Dov Stekel and Dr Adam Blanchard for allowing me to make use of their technical and scientific support during my Project. I would like to again thank Adam Blanchard for his support throughout the apprenticeship, by continue to ensure my well being and compliance with the course. I would like to thank all the University of Nottingham staff who contributed to the taught course content which formed the basis of my capabilities to undertake this project. I would like to thank the apprenticeship team for their support in facilitating my Bioinformatics Scientist apprenticeship and for my chance to participate in proving feedback on the course. And finally I would like to thank my colleagues and my Family for supporting me through the entire apprenticeship, but especially their support during Covid-19 lockdowns.

1.0 Introduction

Proteins are known to be the main proponents of biological function through their interactions with proteins and other molecules, such as RNA and DNA. These interactions result in the meditation of metabolic and signalling pathways, cellular processes and organismal systems. Since proteins play such a pivotal role in biological function, there has been increased research dedicated to better understanding their relation to the treatment of diseases [1]. All possible molecular interactions that can occur within an organism are defined by its interactome and as such many subsections of specific interactions between two molecule groups exist as their own fields of research [2]. Of these, protein-protein interactions are the most well-documented and explored. These interactions are recorded through a variety of methods, which make use of both experimental identification and computation prediction. Experimental identification methods include biophysical methods, whereby interactions are captured through methods such as spectroscopy, x-ray crystallography and fluorescence, as well as high-throughput methods, whereby direct methods such as the Yeast two-hybrid model can be used to fuse two proteins to a transcription binding domain so that their interactions may be examined or indirect methods such as examining co-expression of genes which are assumed to have interacting proteins due to their co-expression. Computational predictions include those based on empirical evidence, such as network properties, co-expression and examination of the relative frequency of interacting domains, and those based on theory, such that as interactions happen in the same organism and the evolution of interacting proteins are linked to preserving function [3].

In order to achieve abstract and computational interactions with the protein-protein interactome, protein-protein interactions are often represented as graphs from the mathematical discipline of graph theory, a graph here being a set of vertices that define each protein with an associated set of edges which encompass the interaction between each protein. A set in this case refers to an unordered collection of elements of some kind, where no element is repeated. These graphs are often referred to as protein-protein interaction networks. However, one key aspect of a graph of this variety is that they are not random and so adhere to principles which govern their structure and evolution. As a result of this protein-protein interaction networks display characteristics which are not expected of a usual graph. The degree distribution of these graphs is unusual as they contain a usually rare feature, hubs, being vertices that have many more connected vertices than the average vertex within the graph. The graphs also display the small world property, as there are relatively short paths, an edge sequence between two vertices, and between protein vertices. Motifs and modules are also features of these graphs, motifs being groups of vertices taken from the original graph that appear more or less frequently than expected and modules being highly interlinked sets of vertices which cause most graphs to display high degrees of clustering. Finally, protein-protein interaction networks display a tendency to include vertices with a high betweenness centrality, being vertices that have a high number of

shortest paths between vertices passing through them. These characteristics add a series of challenges to the application of graph theory-based methods to protein-protein interaction networks [4]. This paper, [4], also explored the context of human disease and how network-based approaches could fit into a better understanding of human diseases, based on the underlying hypotheses and structure of disease networks. These Disease networks are large, highly connected and also have interaction data available from more than just one interactome, making them great resources for testing methods that would try and bridge community detection across multiple networks. Of the hypotheses summarised, I will outline those which are particularly important in the protein interactome. A common hypothesis is that Protein hubs are associated with essential genes, which don't as often interact with disease-related genes, since they generally appear at the fringes of protein functional networks. Another accepted belief is that any interacting Proteins which share a common disease network are likely to interact with each other. Similarly, Proteins related to certain diseases or cell types are very likely to have high interconnectivity as they are present in the same/similar networks. And finally, the shortest molecular path between disease-associated interactome information is expected to align with causal pathways.

Due to how pivotal protein-protein interaction networks are to research based upon further understanding molecular biology, several open-source data repositories have been created to make these networks publicly available. One popular source is IntACt, a molecular interaction database, which allows for a wide variety of visualisation and analysis of molecular interaction networks across multiple species [5]. Another critical database, which focuses on protein-protein associations specifically is STRING, which integrates various publicly available protein-protein interaction sources and also scores these interactions based on the evidence related to the interactions [6]. This functionality of being able to assign a score to each interaction can be very useful for graph-based analysis, which can take into account an associated weight for each edge.

Given that there are large quantities of interaction data publicly available it has been of interest to analyse these networks with various techniques [7-11]. A common practice when working with graph-based objects is to understand how the vertices within a graph can be grouped based on the underlying structure of the graph. An algorithm which performs this task is often referred to as a community detection algorithm.

Although graph-based methods will be used within this it is worth noting that another popular form of sorting data into groups is hierarchical clustering.

The data to which hierarchical clustering is usually applied will be a dataset which captures the relationship between at least two dimensions. When a data set is defined by higher dimensions the first aim of hierarchical clustering analysis is to reduce the dimensionality of the data set through methods such as a Principal component analysis [12] These datasets are also represented in such a manner that any two data points be measured, leading to many interconnected data points. Once the dataset is represented in such a manner that its

underlying structure can be assessed hierarchical clustering can be performed. A choice of algorithm and objective function must be made as well as the number of clusters to obtain. When considering algorithms and their objective functions hierarchical clustering falls into two approaches. One initialises the algorithm by assigning each representative data point its own cluster and then taking successive hierarchy steps by which two existing clusters are merged based on the best available measurement of the chosen objective function. The algorithm will stop merging clusters once the desired number of clusters is reached and so this results in a very greedy method, which only values the measured objective function at each hierarchy step. This approach is known as Agglomerative or "bottom-up" hierarchical clustering. The other is a Divisive or "top-down" hierarchical clustering whereby instead each representative data point is initially assigned to the same cluster and at each hierarchy step an existing cluster is split into two, based on the algorithm and objective function chosen. Again, this method also stops when it obtains a given number of expected clusters. These methods then result in an expected number of clusters which does not necessarily inform upon the underlying structure of the dataset; however, the constructed hierarchy does allow for investigation of how the final clusters were obtained and so important structural features may be found. Often the algorithm chosen will have to be carefully considered as most require the fine-tuning of some parameter as well as the number of desired clusters [[13], [14], [15]].

Graph-based community detection algorithms follow a similar process by which to construct an analysis. If the data is not already in a form by which it is represented as a set of vertices connected by edges, then certain methods can be applied to obtain this. These methods aim to take the underlying data set and connect data points which would now represent vertices within a graph usually based on a measure of proximity between data points. Two kinds of methods should be considered here, as one method is unlikely to capture both local and global features. Neighbourhood-based methods construct a graph by connecting vertices to their neighbouring data points, if they are considered to be local neighbours by their pairwise distance, such as K nearest neighbours and ϵ-ball graphs. These methods lead to a good representation of the local features of a graph but are unlikely to capture global properties. Minimum spanning tree based methods seek to find the global connectivity for a dataset, such that the overall distance of all created edges is minimum. As they seek to capture global connectivity, minimum spanning trees are great at representing the global features of a graph but are unlikely to retain much local information about highly connected data points. The best method to apply depends heavily on the underlying data, however, it has been found that Neighbourhood based approaches often yield better representations of underlying data [16]. Once the dataset of interest is abstracted into a graph object, community detection algorithms and their objective functions can be applied. Unlike hierarchical clustering which usually measures distances between representative data points, community detection methods make use only the graph topology to decide which community a given vertex belongs to, such as which vertices it

is connected to, the weight associated with those edges and the connectivity of the entire graph for a suggested community assignment. This allows community detection methods to generate communities based more on the structural significance of a given vertex, however, this does result in only the final partition being a reliable representation of the underlying graph structure. This is due to the steps within community detection algorithms consisting of considering each vertex once, with a random initialisation as to the order of consideration. This results in requiring multiple unused calculations creating large computation times [17]. As protein-protein interaction networks are the focus of this study no transformation is required, instead we perform the abstraction mentioned earlier.

Given a dataset represented as a graph, a community detection analysis now requires an algorithm to use and that algorithm requires an objective function to optimise for. Within graph theory, a set of communities which together contain all vertices of a graph G but do not share any vertices is defined as a partition of G.

In this study we have made use of a modified version of the Louvain algorithm, the modification being its objective function. The unaltered Louvain algorithm makes use of a measure called modularity to assess which vertices to assign to which nodes. Modularity as a measure evaluates the density of edges within the proposed communities of a partition compared to the density of edges between the proposed communities [18]. The steps to performing the Louvain algorithm will be discussed in the methods section.

For our objective function, we will consider a measure named Markov Stability, which will be further explored in the methods section. A key aspect of Markov stability is that it introduces a dynamical Markov process onto the graph and so this methodology introduces an artificial time parameter. Markov Stability measures whether a given partition has a higher probability than just by mere chance to have random walkers start and end in the same community at a given Markov time. Due to the introduction of the Markov time parameter, the modified Louvain algorithm can be applied at different Markov times to obtain multiple partitions of the same graph, revealing varying resolutions of the local and global features of the graph. This leads to a framework that can be used to perform multiscale community detection, which can have its generated partitions evaluated over periods of the artificial time component. This allows us to obtain a method which makes use of the strengths of both graph based community detection methods and hierarchy based methods, as we obtained an evolution of reliable underlying structure which results in a number of communities based upon the graph itself and not what we believe the number may be. This method has been successfully applied to various problems, including protein structure organisation [19], and uncovering pathways in enzymes [20].

The Protein structure organisation study, [19], made use of a multiscale community detection as well in order to obtain their self-defined robust partitions at varying values of the time parameter. The implementation is very similar to the one we will propose, as this study is one of the ones which forms our

implementation. The context to which they chose to apply it was a network of physico-chemical atomistic interactions which describes the structure of the interaction protein of the myosin tail. The 3D structure data is larger and more structurally significant than Protein-Protein Interaction Networks as they are a much similar abstraction. This study highlights the structural significance these methods can help explore as they obtained multiple structures across Markov time and were even able to build up a structural picture of how the functional domains are constructed down to the helical turns, amino acids, peptide bonds and bonds and chemical groups. This study then very aptly highlighted how many of the structures which were captured by different communities each have had different research papers allude to such structures, as well as a potentially influential residue, A809, for the binding of MyoA, which had not had a prior experimental investigation. This paper shows that community detection methods can capture global and local features of datasets as well as lead to novel results when known influential molecules are assigned to communities with similar unexplored molecules.

In this study, we will seek to apply this framework to protein-protein interaction networks and aim to evaluate whether relevant partitions are constructed for protein-protein interaction networks. As part of this study, we will also explore how hubs behave as partitions merge through varying scales and seek to understand whether partitions at later time points are consistent with earlier time points.

The remainder of this thesis shall thus cover the following. The methods used, state the underlying mathematics which governs Markov stability, the steps needed to perform the Louvain algorithm, the framework which enables multiscale community detection and the validation of any results obtained. Then how these methods were implemented within the R programming language shall be covered, followed by the results obtained and their discussion.

2.0 Methods

2.1 Markov stability

The following methods seek to outline the definitions required to define Markov stability and are based on a review of the following studies [16,19-23].

We define a graph G as a combination of three sets. The first set is the vertex set V, which captures the set of vertices present in the graph. If two vertices are to be connected to one another, this is represented by an edge. All edges within the graph are captured by the second set, the edge set E. Edges may have a varying value associated with a graph, which is captured by the weight of the edge. E, therefore, comes with a corresponding weight set W, which defines the weight of each edge present within E. G is therefore a weighted, undirected graph representing V and E.

Let a specific entry within an object be denoted as O_e, where O is the object of interest and e is the specific entry within O.

Consider a graph G defined as above, with vertex set V, edge set E and weights set W. Let V be comprised of n vertices and let E be comprised of m Edges with their associated weights in W. From this definition, we can now derive further properties and representations of G.

G can be represented by the n by n adjacency matrix A, where A_{ij} is the weight of the edge between vertex V_i and vertex V_j. If no edge exists between V_i and V_j, A_{ij} is zero.

A can be used to define the n-sized weighted degree vector d, as $d_i = \sum_j^n A_{ij}$

d can be used to define the n by n diagonal weighted degree matrix D, as $D_{ii} = d_i$ and 0 otherwise.

A can be used again to define the total number of edges m within G, as $m = 1/2 \sum_i^n \sum_j^n A_{ij}$

Given G we can define a random walk on G as a process governed by a time step parameter t. This random walk will start on a vertex within V and at each time step t will move to a vertex connected to the vertex the process is currently on.

A key property of this Random walk is its n by n probability matrix M, where M_{ij} is the probability of moving from V_i to V_j in one time step. Since G is a weighted graph, one should note that the probability of moving to a connected vertex will be proportional to the weight of the edge between connected vertices.

This random walk can also be considered as a discrete-time Markov process, as the probability of moving to another vertex is only governed by which vertex the random walk is currently on and so no other previous step information is required. A key property associated with a Markov process is its transition matrix which defines the probabilities associated with moving each time step within the Markov process. From that definition, it is clear to see that M would fulfil the role of the transition matrix.

Considering what M represents and the note that each probability will be proportional to the weight of the edge connecting two vertices, it is clear that the components of M will include the adjacency matrix, as it describes G and its edge weights, and a term which captures the proportional weights related to each vertex. The latter term can be obtained by taking the inverse of the earlier defined diagonal degree matrix D. Therefore M can be defined as the product of these two terms and results in the probability matrix

$$M = D^{-1}A$$

Note M is a right probability matrix, as each row within the matrix sums to 1.

Now that we have defined the probability matrix associated with this process, we can define the probability distribution for each time step t. Let the n-sized vector $p(t)$ encodes these probability distributions, where $p(t)_i$ would correspond to the probability of being on vertex V_i at time t for this process. Clearly, $p(t)$ would have M as one of its components, as it captures the proportional probabilities present for a random walk on G and also contains a component which considers the previous steps within the random walk. However as was already discussed, only the previous step is of interest when considering the next. We can therefore define the next step $p(t+1)$ as

$$p(t+1) = p(t)M$$

An important property of $p(t+1)$ is its stationary distribution, which is the probability distribution π such that $\pi = \pi M$. The stationary distribution for this random walk would therefore be

$$\pi = d^T/2m$$

Let the n by c matrix H denote a partition of the graph G, such that H_{ij} is equal to 1 if vertex i is within community j and 0 otherwise.

When considering the Markov process defined above, this can instead be considered as a random variable X_t with an associated autocovariance matrix $cov(X_t, X_{t+\tau}) = E[X_t X_{t+\tau}] - E[X_t]^2$, where E denotes the expected value and τ denotes a lag in time [21]. The autocovariance allows for a quantification of phenomena whereby it is expected that over a given time span the state of the Markov process considered as a random variable is more likely to remain within the starting community, as compared to random chance.

Consider that the vertices of G can be put into c non-overlapping communities. Given a partition as defined by H the clustered autocovariance matrix of the random walk at time step t is given by

$$r(H)_t = H^T[\Pi M^t - \pi^T\pi]H$$

a matrix which describes the t-dependent probabilities of transferring between different communities, as each element $(r(H)_t)_{ij}$ denotes the probability of

starting in the community i and then after t time steps being in community j, minus that two independent random walkers are in communities i and j when evaluated by the stationary distribution. It is important to note the expected behaviour for a good partition H. A good partition would persist over a timespan t and would also yield high values along the trace of $r(H)_t$, as the probability of leaving a community and re-entering it later is low. As a result of this the discrete-time Markov stability can be defined as

$$b(t, H) = trace(H^T[\Pi M^t - \pi^T \pi]H)$$

and so be used as an objective function to find good partitions of a graph over all t when maximised

$$b(t) = {}^{max}_{H} b(t, H)$$

We will now consider a continuous-time Markov process associated with our random walk. One way to arrive at a continuous-time process is to assign a continuous Poisson process at each vertex of G. We will assume that for all nodes the Poisson process is evenly distributed. We thus obtain the following diffusion dynamics related to our continuous-time Markov process:

$$\frac{dp}{dt} = -p[I - D^{-1}A] = -p[I - M]$$

resulting in the following probability matrix for this process $p(t) = e^{-t[I-M]}$, as this is the solution to the differential, excluding a potential constant. Note that e is the matrix exponential as $ -t[I-M] $ is a matrix.

Note that the stationary distribution of this process is the same as that of the Random walk, thus the clustered autocovariance matrix for the time-continuous process is:

$$r_{continuous}(H)_t = H^T[\Pi e^{-t[I-M]} - \pi^T \pi]$$

As with the discrete-time Markov process, this then results in the definition of continuous-time Markov stability as

$$b_{continuous}(t, H) = trace(H^T[\Pi e^{-t[I-M]} - \pi^T \pi]H)$$

With associated objective function

$$b_{continuous}(t) = {}^{max}_{H} b_{continuous}(t, H)$$

2.2 Louvain algorithm

Now that we have defined an objective function, we will need to make use of an algorithm to apply the objective function to a graph. For simplicity, a popular and rather simple-to-implement algorithm was chosen and modified, the Louvain algorithm [18].

The Louvain algorithm consists of two phases to obtain optimised partitions.

The first phase consists of the flowing steps:

1. Given a Graph G, as defined previously, assign each vertex within the graph to its own unique community
2. Complete a pass of the vertices, where a pass consists of:

- Select a vertex at random, calculate the modularity of G when that vertex is part of its current community, as well as each of the communities of the vertices connected to the selected one
- Assign the selected vertex to the community which yielded the highest modularity for the graph
- Repeat the steps of the previous two tasks for each vertex not already selected

3. If the resulting partition is equivalent to the partition obtained before the pass of the vertices, proceed to Phase 2, otherwise apply steps 2.-3. to the resulting partition

The Second phase consists of the following steps:

1. Given the partition obtained from phase 1, collapse each of the vertices with the same community into a single vertex, which has an edge set consisting of one edge starting and ending at the single vertex for each vertex collapsed and all edges belonging to the collapsed vertices that didn't start and end within the same community
2. Apply Phase 1 to this collapsed graph
3. If the resulting partition has the same number of communities as there are vertices, construct a partition by assigning the communities obtained to the vertices of G which made up the collapsed vertices and end, otherwise repeat steps 1. and 2.

The modification step is that of the objective function, as the Louvain algorithm optimises for communities which maximise the modularity of a graph, for which we instead maximise Markov stability.

2.3 The Markov stability framework

The time parameter defined within the Markov process will from now be referred to as Markov time. Markov time is the critical component which enables community detection at multiple scales by acting as a resolution parameter. This resolution is in practice characterised by the number of communities detected as t varies. When t is small, more communities are detected leading to the identification of a graph's local features, since as t increases the autocovariance for weaker communities degrades, meaning at high t, fewer communities are present, revealing the global landscape of the graph.

Hence important parameters of this methodology are the number of iterations for which to vary t for, n_t and how much to vary t by for each step, t_{step}. In order to ensure a maximised partition is obtained at each t, we will also be applying the modified Louvain algorithm multiple times and choosing the resulting partition with the highest Markov stability, introducing another parameter n_L, the number of times to optimise for Markov stability at each t.

In order to assess the outputs of this method, we will apply a dissimilarity metric to the n_L resulting partitions obtained within each t and also apply a dissimilarity metric to the resulting partitions between each t. The dissimilarity metric of choice in both cases will be the variation of information between two partitions $VI(H_1, H_2)$ [24], which was not implemented in this study, an implementation from an open-source software package was used. As a measure, the variation of information will be zero when two given partitions are the same and otherwise small if they are similar, where small is relative to the range that the variation of information can take. As a metric the variation of information measures how assigning the two different partitions results in a change of information and its value is bounded as such $0 \leq VI(H_1, H_2) \leq log(n)$.

Within this methodology, the resulting metric for with t variation will therefore be

$$VI_{within}(t) = \frac{1}{n_L(1-n_L)} \sum_{s=1}^{n_L} \sum_{s'=1}^{n_L} VI(H_s^*(t), H_{s'}^*(t))$$

where $H_s^*(t)$ is a maximal partition obtained at optimisation step s. As a result of this, an associated average variation of information across n_L for each t is calculated. Note that a result of 0 for a given t indicates that all n_L partitions were the same.

The resulting metric for between t variation will therefore be

$$VI_{between}(t, t') = VI(H^*(t), H^*(t'))$$

where H^* is a maximal partition and the resulting $VI(t, t')$ is t by t matrix where each entry i, j contains the variation of information between the maximal partition obtained at Markov time i and j.

The methods described above can now be summarised to describe a framework by which multiscale community detection can take place. Given a graph G a modified version of the Louvain algorithm as described above can be applied to G in order to obtain a partition optimised for time-continuous Markov stability. This process of application will also have a sequence of Markov time t recorded, the first instance of which being $t_{initial}$ and thereafter will be varied by t_{step} until $t = t_{max}$. This application will be applied n_L times in order to ensure the randomly initiated Louvain algorithm has a better chance of finding an actual maximum solution. Of these n_L partitions, the $VI_{within}(t)$ will be recorded and the partition with the highest Markov stability will be kept as the maximal partition for that Markov time $t_{initial}$. Next $t_{initial}$ will then be varied by t_{step} and the process will repeat itself, resulting in an average variation of information measure for the n_L partitions and a maximal partition associated with the current Markov time step. Once n_t applications have been made, the $VI_{between}(t, t')$ will be calculated for all obtained maximal partitions, resulting in a matrix structure with entries i and j such that $VI_{between}(t_{initial} + i(t_{step}), t_{initial} + j(t_{step}))$. This framework will therefore result in a matrix of between t variation and a dataset consisting of a maximal partition with an average variation of information for each associated time step t.

When considering the results of the framework described above, a variety of methods will be used to evaluate the quality of the partitions established at certain timescales. A high quality partition will be defined as robust if a partition occurred with either few or no differences in community structure when comparing the n_L partitions obtained and when comparing the maximal partitions with the same number of communities obtained. The quantification of how much difference in community structure exists will be done through the use of $VI_{within}(t)$ and $VI_{between}(t, t')$. The flow of communities assigned as t varies will be assessed by visualising how vertices are assigned to communities across the varying t, with the intent of finding that local communities are strong enough so as not to split into different more global community structures. In order to assess the presence of Hubs within a given G, we will make use of a simple topological measure proposed in the study [25]. The measure described is a connectivity of the induced subgraphs of G, being subgraphs where only a certain set of vertices and their associated edges start and end within those vertices are captured and so is a measure of relative subgraph connectivity. These subgraphs of G are constructed by sorting all of the vertices present in G in descending order by their degree, the number of edges that are incident to a vertex. The first x vertices of this order then make up the subgraph G_x. For a given induced subgraph G_x, this connectivity measure is defined as

$$f = C_x^{max}/|V(G_x)|$$

where C_x^{max} is the number of nodes for the largest component in G_x, $|V(G_x)|$ is the total number of vertices in G_x. This results in a measure that is expectant to identify hubs of a graph G when for a given increase in x, the resultant connectivity is much closer to 1 than previously.

3.0 Implementation

To implement the methods described above we require the use of a programming language, as to ensure reuse of the methods and to allow access to existing open-source tools. The methods described above were applied to a specific altered example from the STRING database, which can be obtained in the following way.

Within STRING an interface for examples can be navigated to, "Example 2" being the one which forms the basis of the dataset used within this study. "Example 2" consists of the 20 most frequently mutated human cancer genes, which have related proteins: APC, BRAF, CTNNB1, DNMT3A, ERBB3, FLT3, NCOR1, NF1, NARS, PIK3CA, PTEN, SMAD4, TP53, SF3B1, FBXW7 and LPHN2. Once this example has been selected within STRING an interface will appear visualising the proteins related to those 20 genes and a button which says "more" is visible with an associated plus sign. This button was used to expand the network until 96 vertices were present in the network. This network results in only one protein not having an interaction, LPHN2, and so since no interactions take place it is excluded from the protein annotations file. On the Analysis page of the interface, the "All enrichment terms" file was downloaded and on the "Exports" page the protein annotations file was downloaded. The exact String network used is available at [26] if it cannot be constructed as described.

The protein annotation file was used to construct a graph object of this network, by making use of its "x.node1" and "node2" columns to construct an edge list, a matrix in this case where the first column indicates a starting node for an edge and the second column indicates an ending node for an edge. This edge list was converted into a graph object using a graph constructions function within one of the packages described below and then the "assigned_score" associated with each interaction was used to assign weights to the edges of the graph object. The enrichment file is made use of within the discussion to explore highly enriched categories for given proteins. The calculation for the assigned score is derived from the combination of all evidence scores associated with a given interaction. These evidence scores related to the interacting proteins consist of scores that quantify their connected proteins, gene fusions, co-expression, database imports, large-scale experiments and literature co-occurrence. A given interaction will not always receive a score for each of these. These scores are combined by assigning Bayesian probability priors to each of the scores, removing the prior distributions from each channel, combining the scores and then finally the prior is added back only once.

This graph object was formatted as seen in figure 1 and will form the basis for all other network plots seen later. The layout of this graph is constructed using an implementation of the Fruchterman Reingold algorithm [27] with an $area$ value of $5n^2$ and a $repulse.rad$ value of $n^{2.5}$.

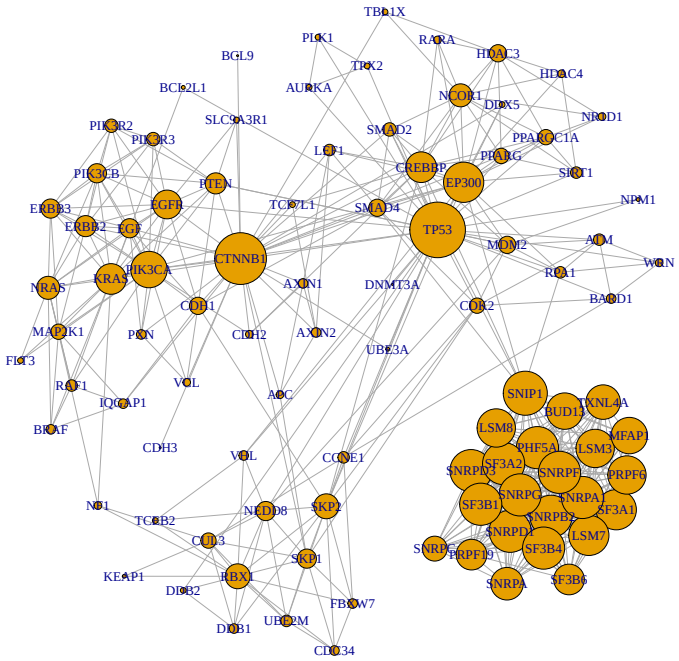

Figure 1: Our example protein-protein interaction network, with vertices sized to degree of the vertex

The framework for multiscale community detection was then applied to this graph network and the results are given in the Results section.

In this study, we have made use of the R coding language [28], as it is a popular choice for performing analysis in the field of biology and it is also equipped with a variety of useful tools to help explore the results obtained. Other than base R and its associated base packages, we also made use of the following packages. The most instrumental package used was igraph [29], which forms the basis for all definitions of objects used within this implementation and provided many other useful functionalities, such as the plotting and formatting of network graphs, graph object creation and editing and implementation for calculating the variation of information between two partitions. The packages tidyverse [30] and dplyr [31] were used for data transformation operations applied to R data objects. The package expm [32] was used in order to calculate the exponential of a matrix and the package knitr [33-35] was used for formatting this pdf output derived from a markdown script.

Graphing packages used include ggplot2 [36] which was used for plotting a heatmap and dual axes plots, netwrokD3 [37] which was used for the creation of a Sankey graph and lastly qgraph [38] which was used to create the layout of vertices used for the network plots.

4.0 Results and Discussion

4.1 Inital Framework results

For the application of the framework described above, we will be using the following parameters: $t_{initial} = 1$ (as this is hard-coded into implementation), $t_{step} = 0.5$, $t_{max} = 49$, $n_L = 20$. This will result in a Markov time span starting at 1, increasing by 0.5 each step for 49 steps. This will result in a Markov time span of 1 to 25. The Louvain optimisation for Markov Stability will be run for 20 iterations at each time step.

Having applied the proposed framework our example from STRING, with the above-specified parameters, we obtain table 1 which shows the expected trend of having Markov stability decrease as Markov time increases due to degradation of the autocovariance matrix. We also plot Markov time against Markov stability in figure 2 to visually confirm the expected decreasing trend of Markov Stability and communities as Markov time increases.

Table 1: Framework application results

markov_time	communities_count	maximal_stab	within_t_variation
1.0	7	0.6116352	0.0000000
1.5	6	0.5789904	0.1190966
2.0	5	0.5579874	0.1530839
2.5	4	0.5426200	0.2187274
3.0	4	0.5206031	0.2376802
3.5	4	0.5026493	0.2064303
4.0	4	0.4878835	0.3027109
4.5	4	0.4751222	0.3516538
5.0	4	0.4621922	0.2816973
5.5	4	0.4501387	0.3563733
6.0	3	0.4552679	0.3922370
6.5	3	0.4463691	0.4415684
7.0	3	0.4378755	0.3879952
7.5	3	0.4300084	0.4894167
8.0	3	0.4223729	0.4370096
8.5	2	0.4371279	0.3877788
9.0	2	0.4339345	0.5224536
9.5	2	0.4307683	0.5292873
10.0	2	0.4276287	0.4079856
10.5	2	0.4245153	0.3714550
11.0	2	0.4214275	0.4509322
11.5	2	0.4183649	0.3006463
12.0	2	0.4153271	0.2262549
12.5	2	0.4123137	0.3315315
13.0	2	0.4093244	0.2771651
13.5	2	0.4063587	0.2333254
14.0	2	0.4034164	0.2356822
14.5	2	0.4004971	0.1979731
15.0	2	0.3976005	0.2356822
15.5	2	0.3947264	0.2356822
16.0	2	0.3918743	0.0000000
16.5	2	0.3890442	0.0000000
17.0	2	0.3862357	0.0000000
17.5	2	0.3834485	0.0000000
18.0	2	0.3806825	0.0000000
18.5	2	0.3779373	0.0000000
19.0	2	0.3752129	0.0000000
19.5	2	0.3725088	0.0000000
20.0	2	0.3698251	0.0000000
20.5	2	0.3671613	0.0000000
21.0	2	0.3645174	0.0000000
21.5	2	0.3618931	0.0000000
22.0	2	0.3592883	0.0000000
22.5	2	0.3567028	0.0000000
23.0	2	0.3541364	0.0000000
23.5	2	0.3515888	0.0000000
24.0	2	0.3490601	0.0000000
24.5	2	0.3465499	0.0000000
25.0	2	0.3440581	0.0000000

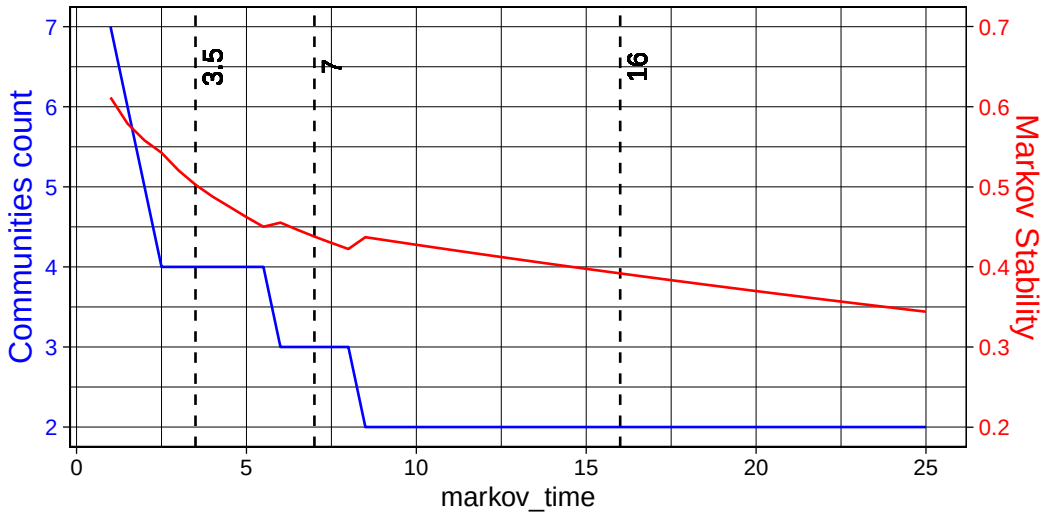

Figure 2: Markov stability and communities obtained, over Markov time

4.2 Evaluating partitions based on the variation of information

In order to better understand which Markov times are associated with interesting and robust partitions of our example, figure 3 plots Markov time against the variation of information within t and figure 4 plots the variation of information between t as a heatmap. Note that for our example $0 \leq VI(H_1, H_2) \leq 4.5538769$. As such Variation of information will be considered low enough, when values fall bellow the 20th percentile of the values $VI(H_1, H_2)$ can take, being 0.9107754. The choice of the 20th percentile and not a lower value is due to the nature of our example dataset containing many vertices which edges which connect to very high degree vertices, meaning they are likely to change community assignment based on which of these high degree neighbours is considered first in the Louvain optimisation.

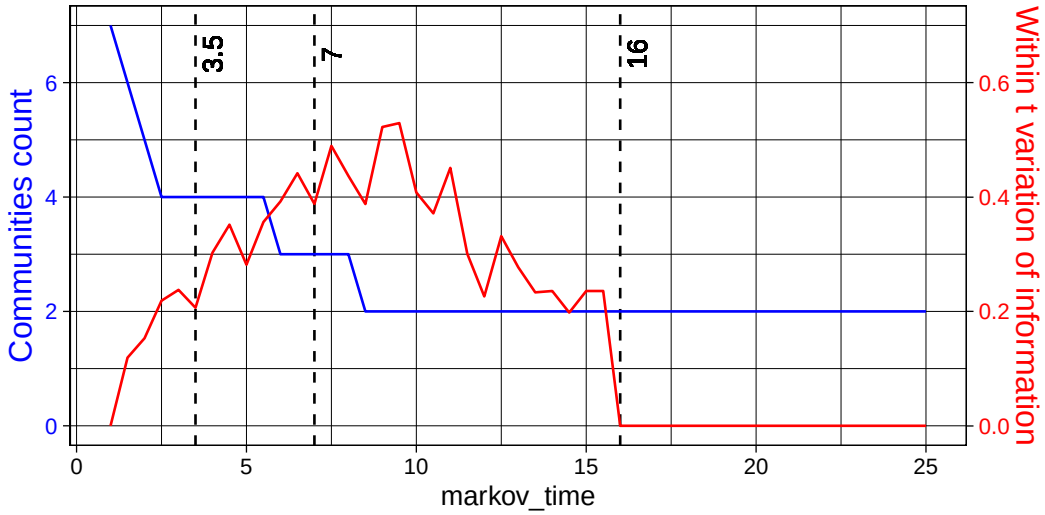

Figure 3: Average Louvain optimisation Variation of information and communities obtained, over Markov time

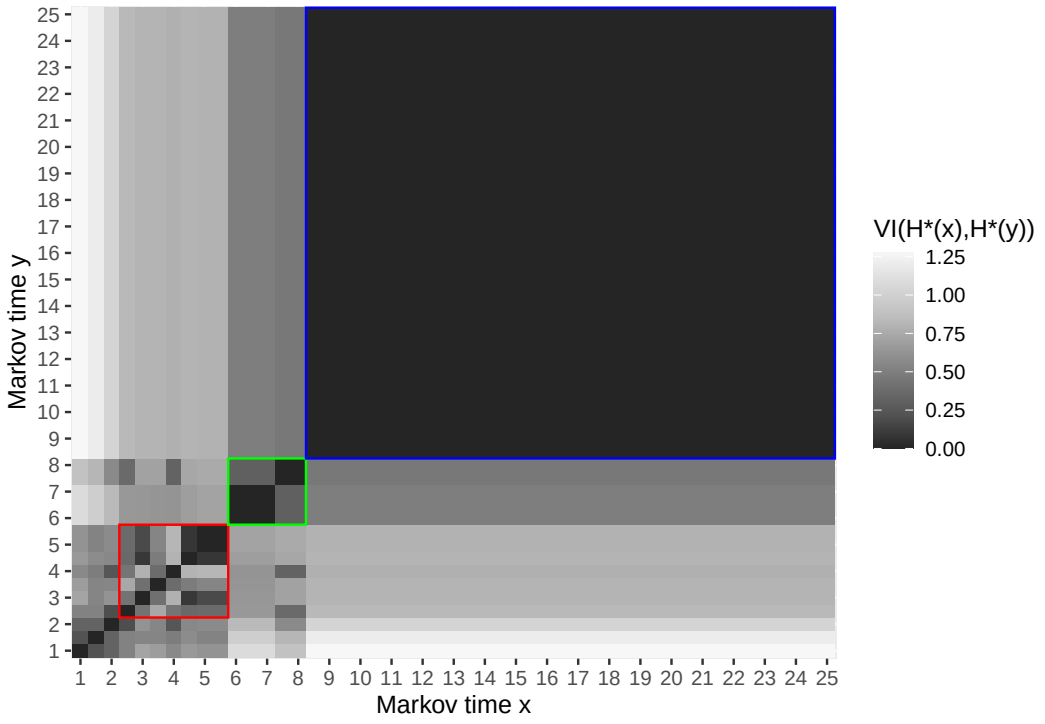

Figure 4: Variation of information between Markov time points

Interesting partitions to further investigate will be those which exist with a community count that persist for multiple Markov time points and yields the lowest $VI_{within}(t)$ for their community count. A selection of three interesting partitions has been made to further explore these results, being those with associated Markov times 3.5, 7 and 16. Markov time 3.5 and 7 were both chosen as they yield the lowest $VI_{within}(t)$ from the partitions they share a community count with and a community count of 4 and 3 both persist for multiple Markov time points.

The results displayed in figures 3 and 4 can be used to determine whether robust partitions of G were obtained. From figure 4 we are interested in large sections of 0 or low variation, as this indicates across that section of Markov time the variations between partitions were low. This would support that a robust partition was obtained as it continuously resulted in a partition with either few or no differences in community structure as Markov time varied for that community count. Interestingly our example does not seem to behave too well, as the only large section of near zero variation of information occurs towards the later Markov time span, as the autocovariance matrix likely degrades to point a of convergence. There are however sections of low variation of information between t present when looking at the Markov time span for which 3 and 4 communities are consistently obtained.

Considering now the within t variation displayed in figure 3, it is clear to see that the Louvain method optimised for Markov stability results in quite a few variations depending on the random initialisation. Based on this measure alone it would seem as though there is no support for robust partitions present, other than those obtained towards the end of our Markov time span, resulting in identical partitions at Markov time 16 with 2 communities. When considering figures 3 and 4 together, it is evident that interesting partitions do exist for a reasonable Markov time span, being when 4, and 3 communities are identified through Markov time spans 2.5-5.5 (the red area in figure 3) and 6-8 (the green area in figure 3) respectively.

We now plot the area of Figure 4, where with Markov time spans when only 4 communities are present and introduce labels to the value of each sector.

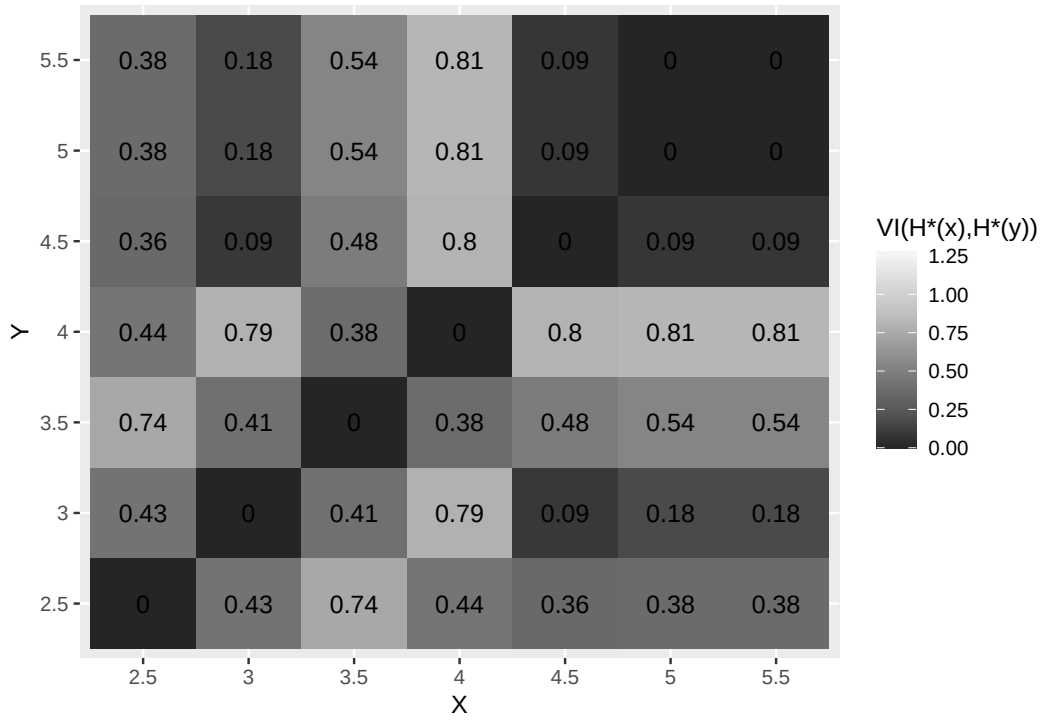

Figure 5: Variation of information between Markov time points for 4 communities

In the case of when 4 communities are identified, although the Markov time span 2.5-5.5 is not long it does persist and the between t variation for consists of low variation of information block, especially for Markov time 3.5. Interestingly the within t variation lends more support for the robustness of this partition as its values are lower than the between t values for the partitions with the same community count. It is worth noting however that ideally our variation of information values for both within and between t would ideally be nearer to zero than the 20th percentile. This suggests that although the maximal partitions obtained for our example are likely capturing the underlying structure, the Louvain algorithm optimised for Markov stability may require more iterations to achieve a near zero $VI_{within}(t)$. The maximal partition obtained at Markov time 3.5 does however still meet our definition of robustness as all variation measures related to the time point and its community count are below 0.9107754.

We now plot the area of Figure 4, where with Markov time spans when only 3 communities are present and introduce labels to the value of each sector.

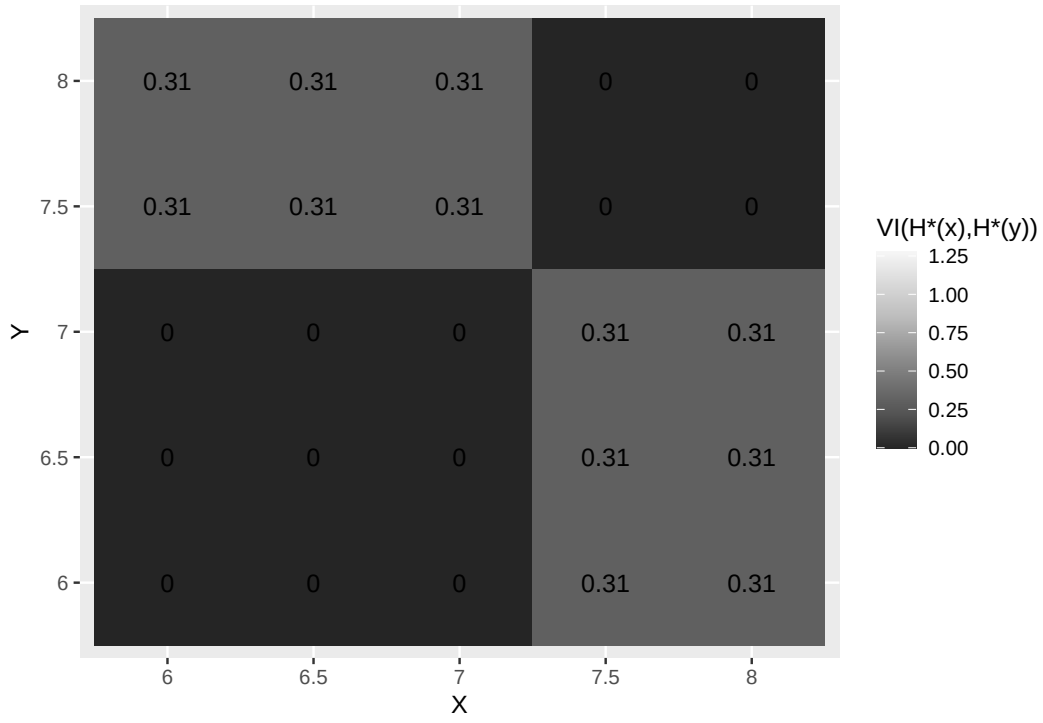

Figure 6: Variation of information between Markov time points for 3 communities

When 3 communities are obtained, they can persist for a greater amount of Markov time than the partitions of 4 communities and have the inverse when considering which variation measures lend their support to the robustness of a partition, as the between t variation block holds values lower than the within t variation. The maximal partition obtained at Markov time 7 does however still meet our definition of robustness as all variation measures related to the time point and its community count are below 0.9107754.

Finally, we consider when only 2 communities are obtained and again make use of figures 3 and 4 to determine whether the chosen partition obtained at Markov time 16 is robust. We obtain partitions with 2 communities in the Markov time range 8.5-25 (the blue area in figure 3), giving us the most persistent community count. From figure 3 we can see that although for about half of the Markov time span there is variation between the within t partitions obtained, however, this quickly decreases when the autocovariance matrix degrades to point of convergence, resulting in no variation within t. However, from figure 4, we can see that the maximal partition obtained across the entire time span always results in the same partition due to the block of zero between t variation. This time span has much clearer support for defining an important feature of the underlying structure of our example and definitely meets our definition of robustness.

As such the partitions obtained at time points 3.5, 7 and 16 do all have evidence to support that they describe important underlying structural features of our example.

The partitions obtained for these Markov time points are shown in figures 7, 8 and 9, again with vertices sized to the degree of the vertex. Communities of these resulting partitions are marked by colouring vertices the same if they fall within the same community. Edges are then highlighted red if they start and end in different communities and are otherwise black.

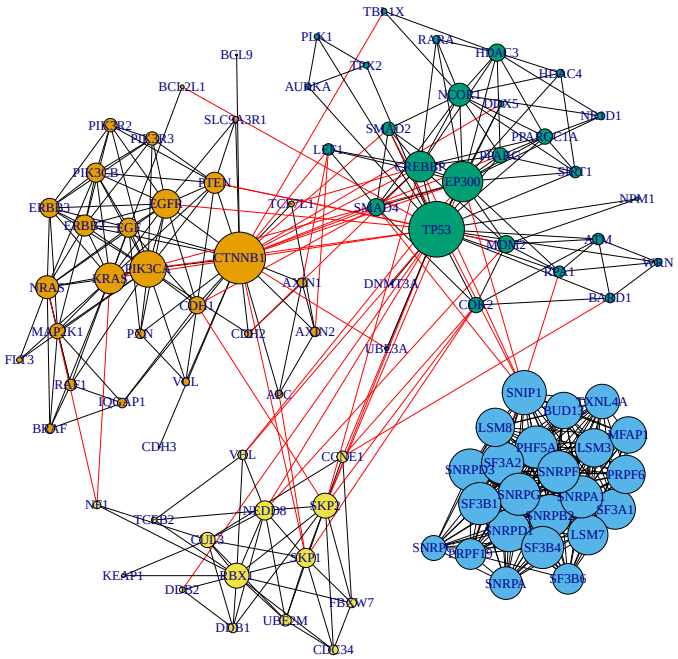

Figure 7: Our example network with the partition obtained at Markov time 3.5

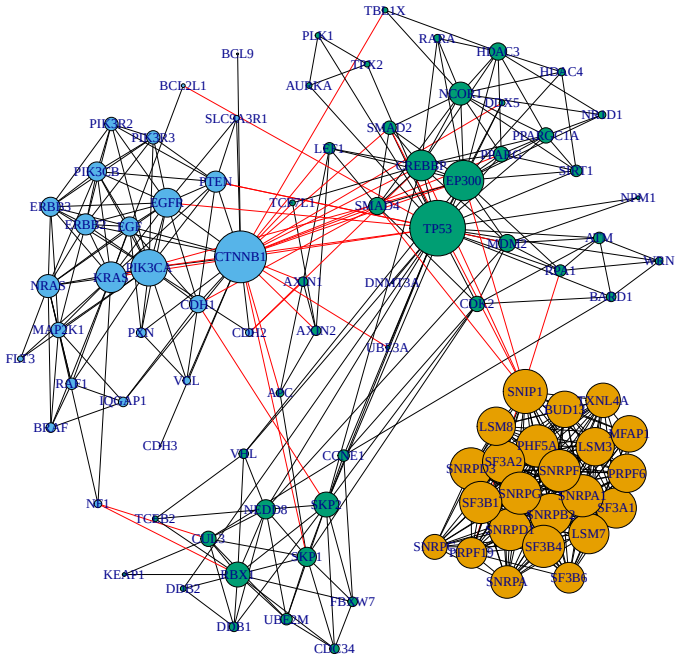

Figure 8: Our example network with the partition obtained at Markov time 7

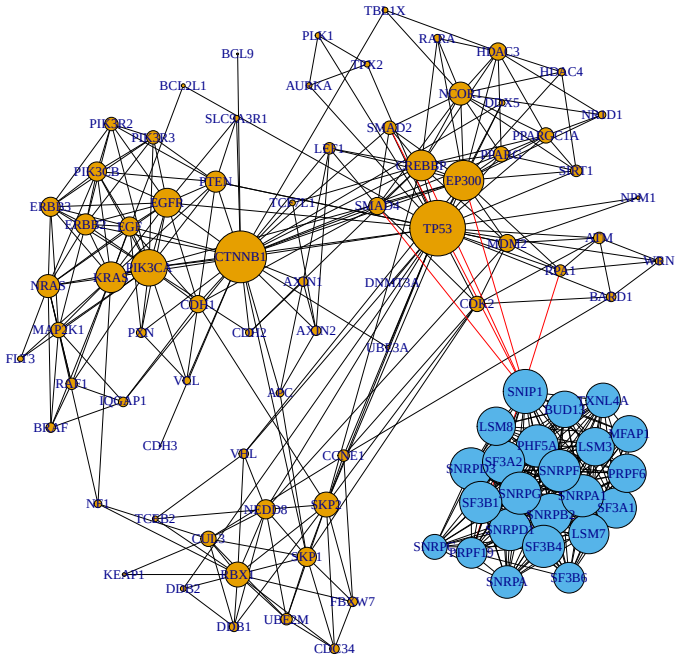

Figure 9: Our example network with the partition obtained at Markov time 16

4.3 Evaluating partitions based on the flow of communities across Markov time

To assess the flow of our community detection cross Markov time figure 10 shows a Sankey diagram depicting the flow of proteins being assigned their communities across t. Table 2 shows the underlying data used to create figure 10.

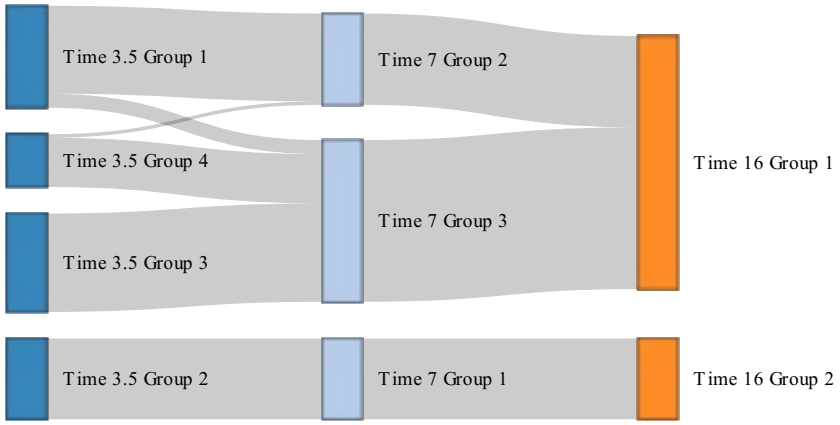

Figure 10: Flow of community construction over Markov time

Table 2: Flow of community construction over Markov time

source	target	value	IDsource	IDtarget
Time 3.5 Group 2	Time 7 Group 1	23	0	6
Time 3.5 Group 1	Time 7 Group 2	25	1	4
Time 3.5 Group 4	Time 7 Group 2	1	2	4
Time 3.5 Group 1	Time 7 Group 3	4	1	5
Time 3.5 Group 3	Time 7 Group 3	28	3	5
Time 3.5 Group 4	Time 7 Group 3	14	2	5
Time 7 Group 2	Time 16 Group 1	26	4	7
Time 7 Group 3	Time 16 Group 1	46	5	7
Time 7 Group 1	Time 16 Group 2	23	6	8

Given these specific Markov times explored above, we can now consider the Sankey diagram constructed in figure 10 and its corresponding table 2. Considering the flow of protein community assignment from left to right, it is clear to see that there is quite a consistent community assignment between the partitions obtained at Markov times 3.5 and 7, as only 5 proteins do not follow their original community members in merging into larger communities.

A large majority of proteins stay within their previous community members, indicating that the flow is good between Markov time points 3.5 and 7. For the Markov times 7 and 16, no proteins disconnect from their previous communities' members signifying that both partitions obtained at Markov times 7 and 16 are consistent in their community assignment. Having quite consistent community assignment between our varying Markov time points signifies that the partitions obtained are reflective of the underlying structure of our example. Considering the whole flow, it is clear that the community which defines group 2 at Markov time 16 is intrinsic to the structure of our example, as no proteins leave that community nor does the community merge to create any new ones over the whole Markov time flow. As such it is likely that at least one protein hub exists within that group and it is keeping all of its connected proteins together. Since there was also little shift between communities at any given time it is likely that at least one hub also exists within groups 2 and 3 at Markov time 7.

Now considering the proteins which broke away from their previous community members at least once, it is of interest to consider the degree and weighted degree of these proteins. The non-weighted and weighted degree distribution of these proteins is in table 3.

Table 3: Weighted and non-weighted degree distribution of the escaped proteins

	proteins	weighted.degree	degree
APC	escaped proteins	3.907	4
AXIN2	escaped proteins	4.893	5
AXIN1	escaped proteins	4.871	5
TCF7L1	escaped proteins	2.857	3
NF1	escaped proteins	3.839	4

Table 4 provides the weighted and non-weighted degree distribution of our example network.

Table 4: Weighted and non-weighted degree distribution of our network

	Weighted.degree	degree
Min.	0.97900	1.00000
1st Qu.	3.91400	4.00000
Median	7.81600	8.00000
Mean	10.24389	10.42105
3rd Qu.	16.10150	16.50000
Max.	28.30300	29.00000

Comparing table 3 and table 4, we see all of the escaping proteins have a low degree and weighted degree when compared to the other proteins present in the

network. It, therefore, makes sense that they are more likely to shift between communities as Markov time increases since they contribute very little to the local structure they are a part of and so aren't strongly associated with any particular community around them.

4.4 Identified protein hubs and their behaviour

To evaluate the presence of hubs figure 11 plots the relative subgraph connectivity for our protein-protein interaction network and table 5 gives the associated data table for the first 25 vertices added.

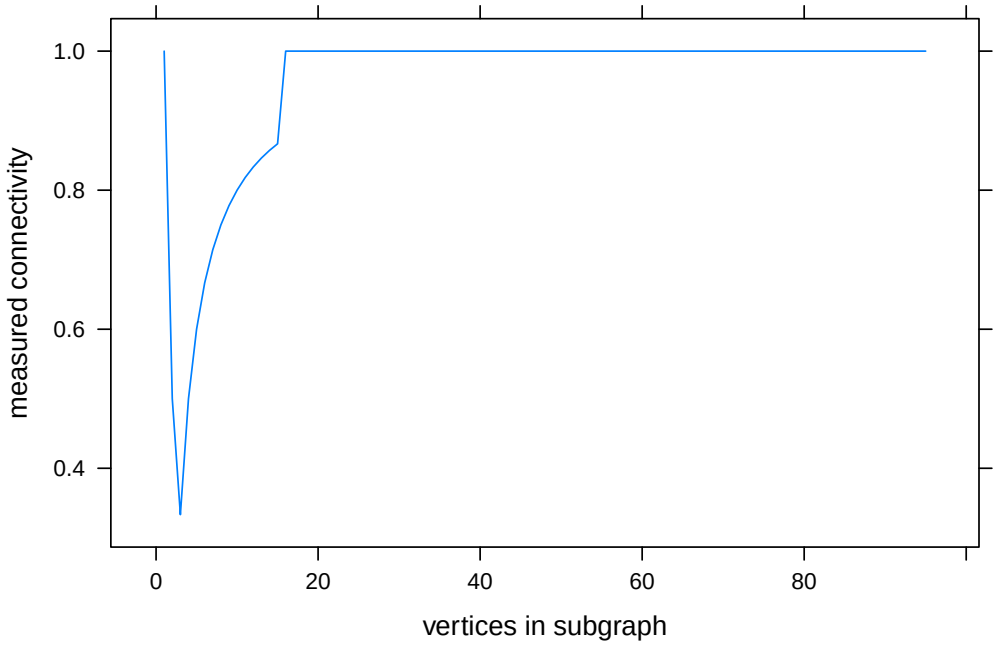

Figure 11: Relative subgraph connectivity of our network

Table 5: Relative subgraph connectivity of our network of the first 25 vertices

vertices added count	connectivity	components count	vertices added
1	1.0000000	1	TP53
2	0.5000000	2	CTNNB1
3	0.3333333	3	SNIP1
4	0.5000000	3	SNRPD3
5	0.6000000	3	PHF5A
6	0.6666667	3	SF3A2
7	0.7142857	3	SNRPB2
8	0.7500000	3	SNRPA1
9	0.7777778	3	SNRPD1
10	0.8000000	3	SNRPG
11	0.8181818	3	SNRPF
12	0.8333333	3	SF3B4
13	0.8461538	3	SF3B1
14	0.8571429	3	SF3A1
15	0.8666667	3	LSM7
16	1.0000000	1	EP300
17	1.0000000	1	LSM8
18	1.0000000	1	PRPF6
19	1.0000000	1	LSM3
20	1.0000000	1	BUD13
21	1.0000000	1	MFAP1
22	1.0000000	1	PIK3CA
23	1.0000000	1	TXNL4A
24	1.0000000	1	SNRPA
25	1.0000000	1	KRAS

Based on the findings of the flow of proteins across Markov time, we will now consider the measure of relative subgraph connectivity to investigate protein hubs within our example which are shown in figure 11 and table 5. As EP300 is added to the subgraph, the subgraph returns to becoming one connected component as the measured connectivity of the subgraph has returned to 1. This indicates that any hubs that may be identified by this method were introduced to the graph before the 16th vertex was added. Looking toward the start of the process we can see that clearly TP53, CTNNB1 and SNIP1 are important to the structure of our example since every added protein after those three does not result in a fourth component being created. However considering figure 1, SNIP1 doesn't seem to have a much higher degree than its interacting proteins as they all interact with each other. Due to this only TP53 and CTNNB1 can be considered as protein hubs, since SNIP1 and its interactions are just very heavily interconnected and so form an integral part of the structure of our example. Considering the community in which SNIP1 is in, it is of interest to examine the weighted and non-weighted degree distribution of this community.

The non-weighted and weighted degree distributions of this community is as follows,

Table 6: Weighted and non-weighted degree distribution of the constant community

	Weighted.degree	degree
Min.	12.82500	13.00000
1st Qu.	18.76150	19.00000
Median	20.84300	21.00000
Mean	19.93013	20.13043
3rd Qu.	21.92450	22.00000
Max.	22.49800	23.00000

Comparing table 6 to table 4, we see that range of degrees within this community is much higher than that of the average of the network. This highly connected community is likely a result of the construction of our example within STRING, as many interactions of SF3B1 were introduced and the majority were also interactions with one another.

When considering enrichment information for this community, we obtain the following results when we only look for enrichment where our false discovery rate (FDR) is less than 0.05, the strength, associated with the enrichment term annotation is greater than or equal to 1.5 and at least 12 of the 23 proteins are associated with the enrichment term.

Table 7: Enrichment terms of community Time 16 Group 2 of strength >= 1.5, FDR <= 0.05 and with at least 12 matched proteins

enrichment category	enrichment term ID	enrichment term	matched proteins
COMPARTMENTS	GOCC:0005681	Spliceosomal complex	23
COMPARTMENTS	GOCC:0005684	U2-type spliceosomal complex	21
COMPARTMENTS	GOCC:0071005	U2-type precatalytic spliceosome	19
COMPARTMENTS	GOCC:0071013	Catalytic step 2 spliceosome	12
COMPARTMENTS	GOCC:0097525	Spliceosomal snrnp complex	18
GO Component	GO:0005684	U2-type spliceosomal complex	21
GO Component	GO:0071005	U2-type precatalytic spliceosome	19
GO Component	GO:0071013	Catalytic step 2 spliceosome	13
GO Component	GO:0097525	Spliceosomal snrnp complex	17
KEGG	hsa03040	Spliceosome	20
STRING clusters	CL:1688	U2-type spliceosomal complex, and mRNA cis splicing, via spliceosome	22
STRING clusters	CL:1690	U2-type precatalytic spliceosome, and U1 snRNP	21
STRING clusters	CL:1692	U2-type precatalytic spliceosome	19
UniProt Keywords	KW-0747	Spliceosome	22

The strength is a measure of the enrichment effect of the annotation, as it is the $log10(a_{expected}/a_{observed})$ where $a_{expected}$ is the number of proteins within the network that are annotated with the given term and $a_{observed}$ is the expected of proteins to have that annotation in a random network for the same size. From the enrichment table 7 it is clear to see why this is such a strong community since all of the proteins within the community are associated with the spliceosomal complex and otherwise more than half of the community has a more accurately annotated role with the complex.

We will now examine the flow which our hubs undertook as communities were merged between Markov time scales compared to the flow shown in figure 10. The flow of those two protein hubs is captured in table 8.

Table 8: Community flow of CTNNB1 and TP53

proteins	group	markov_time
CTNNB1	Time 3.5 Group 1	3.5
CTNNB1	Time 7 Group 2	7.0
CTNNB1	Time 16 Group 1	16.0
TP53	Time 3.5 Group 3	3.5
TP53	Time 7 Group 3	7.0
TP53	Time 16 Group 1	16.0

When comparing table 8 to figure 10, it is clear to see that although CTNNB1 and TP53 did merge into a single community, they did manage to create their separate communities at Markov time points 3.5 and 7. CTNNB1 is part of community Time 3.5 Group 1 and Time 7 Group 2 and TP53 is part of community Time 3.5 Group 3 and Time 7 Group 3. Interestingly this then shows that hubs underpin the expected structure, for our example, when transitioning across Markov times, except for community Time 3.5 Group 4 and persisting community related to SNIP1 which has already been discussed. The highest degree vertex within community Time 3.5 Group 4 is potentially also a hub and so it is of interest to understand the degree distribution of this community.

Table 9: Weighted and non-weighted degree distribution of the community Time 3.5 Group 4

	Weighted.degree	degree
Min.	1.9980	2.000000
1st Qu.	4.3430	4.500000
Median	4.9660	5.000000
Mean	6.3876	6.533333
3rd Qu.	8.7380	9.000000
Max.	12.8990	13.000000

Comparing table 9 the table 5, we see that NEDD8, SKP2 and RBX1 have the highest degrees within that community and so due to their interaction with each other it is likely they exert enough influence on each other and their other interaction proteins to create their community, however, none of them is likely a hub.

The fact that protein hubs have shaped our example's results based on topological structure highlights how difficult it is for analysis methods to lead to novel results and in this case we only have 3 out of the total 95 nodes as hubs.

From the earlier discussed research [4], it is clear that the effect Hubs have on the topology of a graph makes it difficult to have these methods result in novelty as it relates to disease markers since they generally avoid hubs. Considering the generally accepted hypotheses discussed, it is not a surprise that this network converges to 2 communities fairly quickly and as all of these Proteins already share a common association with Cancer and the constant communities have a high overlap of tissue. The previously conducted research also further supports that a likely disease-varied network would be of use to investigate with these methods. However, since pathways are expected to mimic molecular pathways for those in disease-related interactomes, studying one larger disease network which focuses on just one or a few connected diseases could lead to actionable molecular pathways.The inability of the multiscale community detection framework to result in consistent communities for the Proteins on the fringes of our network, suggest that this method may not be useful when trying to explore novel disease pathways, as hubs influence the topology of any Protein-Protein interaction network greatly.

From the earlier discussed research, [19], the value of trying to understand how a method applies to a certain interactome dataset lies in large datasets which have previously been studied so that the empirical evidence of others can be used to assess whether the communities obtained are of significance. However similar to this paper in this context we have been able to see both local features and global features of the dataset, although the range from which structural features exist between feature states, a larger set will likely yield grander build-ups from building blocks.

5.0 Conclusion

We have applied an implementation of multiscale community detection to an example protein-protein interaction network to assess whether we can obtain robust partitions across scales. This research serves as comparison to those preforming similar research around the application of methods which make use of the graph based representations of Protein-Protein interaction networks in order to investigate if the underlying graph structure can be used to gleam non-obvious protein relations through the interactions of the proteins they share communities with. We showed that multiscale community detection is a valid avenue for this kind of research as we were able to see how the underlying structure of the network evolved at different scales and how the intrinsic properties of Protein-Protein interaction networks can undermine efforts to gleam novel structural insights. Considering here epecially the impact that the three protein hubs identified have on the overall structure of the graph, the partitions obtained clearly have their community assignment influenced, as distinct hubs avoided merging into common communities during the increasing Markov time and we still managed to have separate hubs in the final 2 communities. This research also supports Markov stability as a sensible objective function which allows methods to introduce a resolution parameter to a graph partitioning problem, especially given its capability to converge within a reasonable amount of steps. This study would likely have benefited from investigating more examples of similar interactomes to allow for a comparison of these findings. As for future avenues of research, a example Protein-Protein interaction network where a known distinction between the proteins exist, so that it may be assessed how these methods preform at correctly assigning communities to the know labels. Other research may also want to investigate the use of other partitioning algorithms which do not rely upon as heavily on a random initialisation step, in order to remove the need of multiple unused partitions being computed as well as introduce a hopefully more powerful measure related to the algorithm to assess robustness at each t, rather than the average within t variation. This research has shown Multiscale community detection through the use of Markov Stability can generate interesting results when applied to Protein-Protein interaction networks and so this forms a basis which could be used for further avenues of research in either the same or different interactomes, which would hopefully reveal an actionable pattern.

References

1. Gonzalez, M.W. and Kann, M.G. (2012). Chapter 4: Protein Interactions and Disease. PLoS Computational Biology, 8(12), p.e1002819. doi:10.1371/journal.pcbi.1002819.

2. Vidal, M., Cusick, Michael E. and Barabási, A.-L. (2011). Interactome Networks and Human Disease. Cell, 144(6), pp.986–998. doi:10.1016/j.cell.2011.02.016.

3. Alighiarloo N, Taghizadeh M, Rezaei-Tavirani M, Goliaei B, Peyvandi AA. Protein-protein interaction networks (PPI) and complex diseases. Gastroenterol Hepatol Bed Bench 2014;7(1):17-31)

4. Barabási, A.-L., Gulbahce, N. and Loscalzo, J. (2011). Network medicine: a network-based approach to human disease. Nature Reviews. Genetics, [online] 12(1), pp.56–68. doi:10.1038/nrg2918.

5. Hermjakob, H. (2004). IntAct: an open source molecular interaction database. Nucleic Acids Research, 32(90001), pp.452D455. doi:10.1093/nar/gkh052.

6. Szklarczyk, D., Gable, A.L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J., Simonovic, M., Doncheva, N.T., Morris, J.H., Bork, P., Jensen, L.J. and Mering, C. von (2019). STRING v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. Nucleic Acids Research, 47(Database issue), pp.D607–D613. doi:10.1093/nar/gky1131.

7. Vazquez, A., Flammini, A., Maritan, A. and Vespignani, A. (2003). Global protein function prediction from protein-protein interaction networks. Nature Biotechnology, 21(6), pp.697–700. doi:10.1038/nbt825.

8. Brohée, S. and van Helden, J. (2006). Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics, 7(1). doi:10.1186/1471-2105-7-488.

9. Nepusz, T., Yu, H. and Paccanaro, A. (2012). Detecting overlapping protein complexes in protein-protein interaction networks. Nature Methods, 9(5), pp.471–472. doi:10.1038/nmeth.1938.

10. Dittrich, M.T., Klau, G.W., Rosenwald, A., Dandekar, T. and Muller, T. (2008). Identifying functional modules in protein-protein interaction networks: an integrated exact approach. Bioinformatics, 24(13), pp.i223–i231. doi:10.1093/bioinformatics/btn161.

11. Rahiminejad, S., Maurya, M.R. and Subramaniam, S. (2019). Topological and functional comparison of community detection algorithms in biological networks. BMC Bioinformatics, 20(1). doi:10.1186/s12859-019-2746-0.

12. Van Der Maaten, L.; Postma, E. & Van den Herik, J. (2009), 'Dimensionality reduction: a comparative review', J Mach Learn Res 10 , 66-71.

13. Xu, R. and Wunsch, D.C. (2010). Clustering Algorithms in Biomedical Research: A Review. IEEE Reviews in Biomedical Engineering, 3, pp.120–154. doi:10.1109/rbme.2010.2083647.

14. Jain, A.K., Murty, M.N. and Flynn, P.J. (1999). Data clustering: a review. ACM Computing Surveys, [online] 31(3), pp.264–323. doi:10.1145/331499.331504.

15. Murtagh, F. and Contreras, P. (2011). Algorithms for hierarchical clustering: an overview. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2(1), pp.86–97. doi:10.1002/widm.53.

16. Liu, Z. and Barahona, M. (2020). Graph-based data clustering via multiscale community detection. Applied Network Science, 5(1). doi:10.1007/s41109-019-0248-7.

17. Fortunato, S. (2010). Community detection in graphs. Physics Reports, [online] 486(3-5), pp.75–174. doi:10.1016/j.physrep.2009.11.002.

18. Blondel, V.D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008). Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10), p.P10008. doi:10.1088/1742-5468/2008/10/p10008.

19. Delmotte, A., Tate, E.W., Yaliraki, S.N. and Barahona, M. (2011). Protein multi-scale organization through graph partitioning and robustness analysis: application to the myosin–myosin light chain interaction. Physical Biology, 8(5), p.055010. doi:10.1088/1478-3975/8/5/055010.

20. Amor, B., Yaliraki, S.N., Woscholski, R. and Barahona, M. (2014). Uncovering allosteric pathways in caspase-1 using Markov transient analysis and multiscale community detection. Mol. BioSyst., 10(8), pp.2247–2258. doi:10.1039/c4mb00088a.

21. Delvenne, J. - C., Yaliraki, S.N. and Barahona, M. (2010). Stability of graph communities across time scales. Proceedings of the National Academy of Sciences, 107(29), pp.12755–12760. doi:10.1073/pnas.0903215107.

22. Lambiotte, R., Delvenne, J.-C. . and Barahona, M. (2014). Laplacian Dynamics and Multiscale Modular Structure in Networks. IEEE Transactions on Network Science and Engineering, [online] 1(2), pp.76–90. doi:10.1109/TNSE.2015.2391998.

23. Liu, Z. and Barahona, M. (2017). Geometric multiscale community detection: Markov stability and vector partitioning. Journal of Complex Networks, 6(2), pp.157–172. doi:10.1093/comnet/cnx028.

24. Meilă, M. (2003). Comparing Clusterings by the Variation of Information. Learning Theory and Kernel Machines, pp.173–187. doi:10.1007/978-3-540-45167-9_14.

25. Vallabhajosyula, R.R., Chakravarti, D., Lutfeali, S., Ray, A. and Raval, A. (2009). Identifying Hubs in Protein Interaction Networks. PLoS ONE, 4(4), p.e5344. doi:10.1371/journal.pone.0005344.

26. version-11-5.string-db.org. (n.d.). 16 items (human) - STRING interaction network. [online] Available at: https://version-11-5.string-db.org/cgi/network?networkId=bg40F4UoRFPX [Accessed 3 Feb. 2023].

27. Fruchterman, T.M.J. and Reingold, E.M. (1991). Graph drawing by force-directed placement. Software: Practice and Experience, 21(11), pp.1129–1164. doi:10.1002/spe.4380211102.

28. R Core Team (2022). R: The R Project for Statistical Computing. [online] R-project.org. Available at: https://www.r-project.org/.

29. igraph.org. (n.d.). igraph – Network analysis software. [online] Available at: https://igraph.org.

30. www.tidyverse.org. (n.d.). Tidyverse. [online] Available at: https://www.tidyverse.org/.

31. A Grammar of Data Manipulation [R package dplyr version 1.0.9]. (20202). R-project.org. [online] doi:https://CRAN.R-project.org/package=dplyr.

32. Maechler, M., Dutang, C., Goulet, V., up, D.B. (cosmetic clean, r42), and TaylorO")), D.F. (expm(method= "PadeO, and TaylorO")), M.S. (expm(method= "PadeO, methods, M.S. ("Higham08* and see ?expm.Higham08...) (2023). expm: Matrix Exponential, Log, 'etc'. [online] R-Packages. Available at: https://CRAN.R-project.org/package=expm [Accessed 3 Feb. 2023].

33. Yihui Xie (2022). knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.40.

34. Yihui Xie (2015) Dynamic Documents with R and knitr. 2nd edition. Chapman and Hall/CRC. ISBN 978-1498716963

35. Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible Research in R. In Victoria Stodden, Friedrich Leisch and Roger D. Peng, editors, Implementing Reproducible Computational Research. Chapman and Hall/CRC. ISBN 978-1466561595

36. Tidyverse.org. (2019). Create Elegant Data Visualisations Using the Grammar of Graphics. [online] Available at: https://ggplot2.tidyverse.org.

37. Allaire, J.J., Ellis, P., Gandrud, C., Kuo, K., Lewis, B.W., Owen, J., Russell, K., Rogers, J., Sese, C. and Yetman, C.J. (2017). networkD3: D3 JavaScript Network Graphs from R. [online] R-Packages. Available at: https://CRAN.R-project.org/package=networkD3 [Accessed 3 Feb. 2023].

38. Epskamp, S., Cramer, A.O.J., Waldorp, L.J., Schmittmann, V.D. and Borsboom, D. (2012). qgraph: Network Visualizations of Relationships in Psychometric Data. Journal of Statistical Software, 48(4). doi:10.18637/jss.v048.i04.