Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Corso di Dottorato di Ricerca in Informatica, ciclo 29/2014

Biological network analysis: from topological indexes to biological applications towards personalised medicine.

Candidato:
Gabriele Tosadori

Relatore:
Prof. Fausto Spoto

Correlatori:
Prof. Carlo Laudanna
dott. Giovanni Scardoni

Anno Accademico 2015-2016

Contents

Abstract

Systems Biology encompasses different research areas, sharing graph theory as a common conceptual framework. Its main focus is the modelling and investigation of molecular interactions as complex networks. Notably, although experimental datasets allow the construction of context-specific molecular networks, the effect of quantitative variations of molecular states, i.e. the biochemical status, is not incorporated into the current network topologies. This fact poses great limitations in terms of predictive power. To overcome these limitations we have developed a novel methodology that allows incorporating experimental quantitative data into the graph topology, thus leading to a potentiated network representation. It is now possible to model, at graph level, the outcome of a specific experimental analysis. The mathematical approach, based on a demonstrated theorem, was validated in four different pathological contexts, including B-Cell Lymphocytic Leukaemia, Amyloidosis, Pancreatic Endocrine Tumours and Myocardial Infarction. Reconstructing disease-specific, potentiated networks coupled to topological analysis and machine learning techniques allowed the automatic discrimination of healthy versus unhealthy subjects in every context. Our methodology takes advantage of the topological information extracted from protein-protein interactions networks integrating experimental data into their topology. Incorporating quantitative data of molecular state into graphs permits to obtain enriched representations that are tailored to a specific experimental condition, or to a subject, leading to an effective personalised approach. Moreover, in order to validate the biological results, we have developed an app, for the Cytoscape platform, that allows the creation of randomised networks and the randomisation of existing, real networks. Since there is a lack of tools for generating and randomising networks, our app helps researchers to exploit different, well known random network models that could be used as a benchmark for validating the outcomes from real datasets. We also proposed three possibile approaches for creating randomly weighted networks starting from the experimental, quantitative data. Finally, some of the functionalities of our app, plus some other functions, were developed, in R, to allow exploiting the potential of this language and to perform network analysis using our multiplication model. In summary, we developed a workflow that starts from the creation of a set of personalised networks that are able to integrate numerical information. We gave some directions that guide the researchers in performing the network analysis. Finally, we developed a Java App and some R functions that permit to validate all the findings using a random network based approach.

Ringraziamenti dovuti

Dal lontano Gennaio 2013 al più vicino Dicembre 2016 tre anni, molto intensi, sono passati. Per arrivare a scrivere questa tesi è stato necessario tanto lavoro. Mi sono preso anche molto tempo per pensare e ripensare, valutare, definire e riscrivere. Tanti dubbi mi hanno attraversato la mente ma anche, e meno male, qualche risposta; ho avuto soddisfazioni e incertezze (soprattutto incertezze che, tutt'ora, sono irrisolte). Riuscire ad arrivare qua, oggi, da solo, sarebbe stato molto più difficile visti gli scogli e le secche che ho scansati per un soffio, le vette raggiunte e i fondi dai quali sono, fin'ora, tornato, le pianure ed i deserti che ho, metaforicamente, attaversato. Praticamente è stato un lavoro di squadra: gente è entrata, gente è uscita e, alla fine, quelli che contano sono ancora qua. Va da sé, quindi, che io abbia diverse persone a cui spettano sinceri ringraziamenti per averci messo il loro, in questo grande viaggio! I rimborsi, purtroppo, non sono previsti, per ora.

Andando in disordine, ma non troppo, abbiamo **i miei genitori** che mi hanno sempre lasciato fare quel che ho voluto, sebbene a volte abbiano espresso la loro opinione con toni poco democratici, che hanno spinto molto per questo dottorato e che non smetteranno mai di dire che si può, e si dovrebbe, comunque far meglio! Non preoccupatevi che io ci provo sempre, anche se con fortune alterne.

A **Sara** che, quasi dieci anni dopo e nonostante tutto, è ancora qua a farmi, un'ottima, compagnia. Ha pure deciso di prendersi un nuovo impegno a tempo pieno, ed indeterminato, e quindi ci siamo sposati. Squadra che vince non si cambia e io sono sicuro che come sempre sarà una bellissima avventura. Spero che la nostra strada sia ancora molto lunga e io, da parte mia, sono prontissimo a percorrerla insieme che si debba farlo a piedi, in bici o in treno non importa. L'importante è farlo insieme!

A **mia sorella** che ha frequentato per qualche mesetto l'università, ha mollato, e ricomincerà (a breve). Ovviamente non sa cosa la aspetta. Consiglio: tieni duro e non smettere mai di fare quel che vuoi. Se ancora non sai che cosa sia, quel che vuoi, tranquilla; sono in pochi a saperlo e forse, ma quasi certamente, si sbagliano. In ogni caso non chiedere consiglio a me: io non lo so che cosa farò da grande!

Agli **amici di sempre**[1], agli **amici del Gasv**[2] e ai **nuovi amici**[3]: grazie di esserci

[1]in ordine alfabetico: Alberto, Alberto, Andrea, Enrico, Leonardo, Nadir, Riccardo, Sebastian et Al. (Montorio, 1990).

[2]in ordine alfabetico: Alessandro geom., Alessandro "mastro birraio", Alfonsina, Andrea, Barabba, David, Dolby, Francesca, Franchino, Giancarlo, Glauco, Mauro, Mihaela, Robi, Uccio et Al. (Gasv 2007).

[3]Alessio ;-)

stati e grazie a chi ci sarà. Grazie di aver rotto le balle quando c'era da romperle e grazie di aver sopportato quando c'era da sopportare. Grazie di avermi (r)accolto quando ero da (r)accogliere e grazie di aver alzato il bicchiere quando c'era da festeggiare. State sereni e aiutate anche me a capire come si fa, per piacere!

Menzioni d'onore (alfabeticamente ordinate) per **Alessio**, **Enrica**, **Genny**, **Jasmina**, **Lara**, **Laura**, & **Michela** i quali mi hanno fatto molta, e buona, compagnia da quando mi sono trasferito al CBMC. Tra cappuccini, torte, brioches, gelati e cioccolati, in totale faranno centomila calorie.

Grazie a **Fausto**[4], che ha creduto in me quando gli abbiamo proposto un progetto che con il suo lavoro scientifico centrava più o meno niente. Credo che l'apertura mentale sia una grande dote e vada grandemente apprezzata. Infiniti ringraziamentui per l'opportunità!

Grazie a **Carlo** e a **Giovanni**[5] che da qualche anno assistono e insistono sul fatto che il mio lavoro sia servito e servirà a qualcun altro. Ogni passo è incerto e io, di incertezze, son di certo espertissimo. Non ho ancora imparato nulla, tranquilli!

Per finire, volevo ringraziare tutte le persone che hanno partecipato in qualsiasi modo al mio lavoro. Una su tutte è **Giorgio Roffo** a cui vanno ringraziamenti speciali. Abbiamo poi **Manuele Bicego**, **Vincenzo Manca**, **Szuzanna Liptak**, **Marco Cristani** & **Lucia Calciano** che hanno dato il loro fondamentale contributo!

L'ultimo grazie va a **Laura Marcazzan** senza la quale superare la burocrazia sarebbe stato un purgatorio.

[4]il mio fiduciosissimo relatore
[5]i miei mentori scientifici

Chapter 1

Introduction

1.1 Systems Biology

Systems Biology is a relatively new and inter-disciplinary field that involves different subjects. It takes great advantage of network-based approaches to model and study complex biological processes [1]. Classic biology is, and historically was, focusing on single biological entities like, for instance, a protein. The innovative perspective that the Systems Biology's approach proposes aim at studying what happens when different actors play their role together, giving rise to complex systems. This goal is achieved by considering the set of all the interactions that take place between these objects in order to build, and model, a system. The system can finally be exploited by investigating *i)* the role of each single actor and also, more interestingly, *ii)* the emergent properties that arise because of their interactions. This is a new, and very promising, approach that gives us the possibility to obtain a comprehensive view of complex biological systems like, for instance, a cell [2].

Denis Noble, one of the pioneers in this field, said that *"Systems Biology... is about putting together rather than taking apart, integration rather than reduction. It requires that we develop ways of thinking about integration that are as rigorous as our reductionist programs, but different.... It means changing our philosophy, in the full sense of the term"* [3]. This quote is a very good summary of what is waiting in the future of biology. A very big effort is currently carried out and aims at obtaining a complete description of living organisms by understanding how cells, tissues and all the biological bricks interact with each other in order to carry out what we call *life*. Systems Biology is not only devoted to investigate what happens in human or other organisms but it is also a way of thinking that permits to describe any complex systems like, for instance, the interactions that take place in the Great Barrier Reef or between each single component of a complex organic molecule.

In this scenario it is easy to understand how many different subjects are involved. Systems Biology is a mixture of very different techniques, skills and technologies and requires knowledge from very different fields. Genetics, genomics, proteomics, metabolomics and transcriptomics are current high-throughput technologies that generate a very large amount of data. Mathematical skills are required to define and build the systems of interacting actors [4, 5]. Statistical techniques are used to investigate the data and are used to test the hypothesis and results. Finally information technologies play a very important role since they allow managing the big amount of data, to extract useful information through algorithms and to visualise models and results in a meaningful way and, especially, in a human-readable format.

Bioinformatics and computational biology are the natural conjugations of the skills that are considered necessary to investigate all the complexity that comes from this amount of information. There are many different methodologies to exploit all the data produced by the modern *-omics* techniques and it is possible to find many softwares, webservices and programming languages. We could use software for the visualisation of molecular structures, e.g. PyMol [6] or SwissPDB-Viewer [7], suites that permit to perform all kinds of statistical analysis and tests, e.g. R [8], programming language like BioPython [9] and BioJava [10] that are designed to deal with biological data and manage biological problems. Webservices, like Blast [11] or ClustalW [12], that permit to investigate proteins structures, homologies between a great number of protein sequences at the same time and, also, provide a lot of other features. There are also many biological databases that we can easily access to download protein structures, sequenced genomes, biological and disease networks, ontologies, standardised names and so on: UniProt [13], GeneOntology [14] and Kegg [15, 16] are just very few examples. But, because of this huge amount of information, we also have to face some drawbacks. Different data format used to describe the same process and data type, softwares with an interesting feature but without another one, different platforms and operative systems, huge datasets that are increasing their dimension every day. Another issue comes from the interface between biologists and computer scientists and mathematicians because of the different spoken languages. A biologist may find a lot of difficulties in managing a programming language and a programmer must understand what the physician really wants to investigate and what is the most informative, and useful, output. One of the purposes of our work, and of all the other works that consider biological and computational aspects two faces of the same coin, refers to the nature of these kinds of works. They can be considered as bridges between the two mentioned worlds, i.e. biology and computer science. Developing novel tools and providing working examples, showing the algorithms behind the softwares, contextualising these mathematical formalisms using real, biological examples, showing numerical results and interpreting them are practices that create links between different worlds. These works allow to understand and to obtain a different point of view, wether more mathematical or more biological, that, ideally, works as a bridge. We aim at creating new bridges.

This is the general context for the modern transition from a reductionist to a holistic view of biology. Reductionism is, and was, the main method used to address biological problems but things seem to be changing. Some scientists in the past, i.e. in the sixties, seventies and nineties, and an increasing number of researchers today, are considering biology as a set of interacting objects that should be considered toghether. In this sense, the holistic approach is (re)gaining more and more interest and an interdisciplinary knowledge that crosses different fields like, for instance, mathematics and biology, is becoming more and more important.

A very interesting, current, model that allows investigating biological systems properties takes advantage of graph theory. Networks, or graphs, are mathematical abstractions that could be used to model complex biological systems. Biological networks are composed by different kinds of actors that may be metabolites, genes, RNAs, and proteins [17]. From a mathematical perspective, networks, abstracted as graphs, are sets of objects that are used to describe interactions, called *edges*, between actors, called *nodes*. Formally a graph is defined as $G = (E, V)$ where E is a set of edges and V the set of vertexes, i.e. the nodes. This simple notation is very useful to model complex, static biological processes where hundreds of genes, RNAs, proteins and metabolites interact together generating thousands of connections [18]. This static representation gives us a frame of the cell. It is not able to take into account the time, which means that the interactions do not change over time. But graphs can be used to represent dynamic processes. These dynamic models permit to describe a changing system and the kinetics that take place, for instance, inside a cell. Some of these models are *i)* Lotka-Volterra based on differential equations systems, *ii)* Petri-nets [19], *iii)* P-systems [20] and iv) π-calculus models [21] just to cite few interesting methodologies in the field.

1.2 Personalised Medicine

A branch of Systems biology, called Network Medicine [22], takes advantage of all the technologies and methodologies we mentioned to study biological systems. It has specific applications in the field of life science. Some of the topics are, for instance, the comprehension of pathological mechanisms in human cells, tissues and organs, the functioning of cell signalling, or how diseases interact with each other [23]. It is possible to discover new, useful biomarkers, to personalise therapies or to suggest potential targets to develop new drugs.

Personalised medicine is a discipline that is emerging and aims at addressing diseases at a single individual level. New analytical tools and new approaches to the study of human pathologies is guiding medicine towards this new and participatory approach. It could be summarised by using four terms that are *predictive, personalised, preventive and participatory*. Predictive in the sense that it aims at finding factors that increase the risk of developing specific diseases, like for instance, genetic predisposition to certain pathologies. Personalised means that the medicine should be tailored to the single individual by defining therapies and treatments that depend on the individual background. Predictive in the sense that developing tools that aim at obtaining an early diagnosis are necessary in order to tackle diseases while they are still in their first stages. Finally, participatory because this new approach requires that the individuals take an active participation becoming an integrating part of the whole diagostic and therapeutic path [24]. It is more and more clear that the current approach based on the use of drugs suffers because of the intrinsic complexity of the human organism. Not all the people have the same response to a drug. Pharmacokinetics greatly varies between different individuals and these variations affect the effect a drug has with respect to a specific tissue, organ or disease. For some people a drug may cause light or heavy side effects, for some other people it may lose its therapeutical function, and for some other it works as it is expected. Personalised medicine aims at overcoming these limitations, taming the great uncertainty that is one of the main limitations of the current medicine.

Personalised medicine could be considered as the future of the medicine and, in this sense, it is possible to foresee that technological advancement will give a very strong support to the personalised approach [25, 26]. At a high level we think about wearables like, for instance, watches that are able to measure body temperature, hearth rate, body weight, and several other parameters, are becoming more and more affordable. Personal technologies that check health-related parameters are becoming part of our daily lives. These instruments create a wealth of data that is very useful in generating information and predictive models to tackle unhealthy behaviours. On the other hand, they allow measuring how healthy a person is. At a low level researchers are becoming more and more able to model specific pathologies by considering the interactions between molecules like proteins and genes. By doing so it is now possible to investigate the role of a specific player in the development of a disease and it is possible to simulate biochemical variations at different time points.

In this context, Systems biology plays a very interesting role. This emerging approach, that conjugates very different subjects, allows creating very specific models that permit to investigate human diseases as a complex systems. By doing so it is possible to highlight emergent properties and complex behaviour that are the result of the interaction of several parties that cannot be unveiled by using the classic, reductionist approach. In this sense new methodologies that are able to create better representations of the biological process are needed. New models, faster algorithms, user friendly softwares are required in order to allow researchers from different backgrounds to exploit the advantages of the systems biology approach. In this panorama, our work aims at creating a new network model that is able to represent the individual variation, to create a simple workflow that allows creating a network from a set of proteins, to allow the investigation of the network properties through a comparison between different classes of network and, finally, to validate the experimental results by using a random network based approach.

1.3 Graphs

Systems Biology takes great advantage from graph theory. Graphs was firstly introduced by Leonard Euler [27]. In his paper, Euler, used for the first time, two terms that are *vertex* and *edge*. Here, the idea of graph was used to model the city of Königsberg in order to address the famous problem related to the city's bridges. The question, addressed by using a graph-based approach, aimed at finding a path through the city that would cross each one of the seven bridges once and only once. Euler demonstrated that the problem does not have a solution and opened a new field related to the study of graphs and their mathematical properties. Currently, graphs are used to model complex systems like, for instance, communities and social interactions, biological processes, and the web. Also they are studied from a theoretical point of view and are used to solve algorithmic problems.

In mathematics, a graph $G(V, E)$ is a set that contains a set of vertices and a set of edges edges. We denote the set of vertices by $V(G)$ and the set of edges by $E(G)$. The cardinality of V is denoted by n and the cardinality of E by m. A *vertex*, also called *node*, represent an abstract and featureless object that describes a particular entity in a specific context. An *edge* is a connection, e.g. a link, between two vertices and connects them. Two vertices that are linked by an edge are said to be *adjacent* and we call them neighbours. Two vertices joined by an edge are called the *endvertices* for that specific edge.

Graphs can be *undirected* or *directed*. In an undirected graph the edges uv and vu are the same edge and they are denoted by (u, v) or (v, u) without difference. In a directed graph each edge has an origin, i.e. a source, and a destination, i.e. a target, and these two specific nodes form an ordered pair. Hence, the edge uv, (u, v), is different from the edge vu, (v, u) since their origin and target are different.

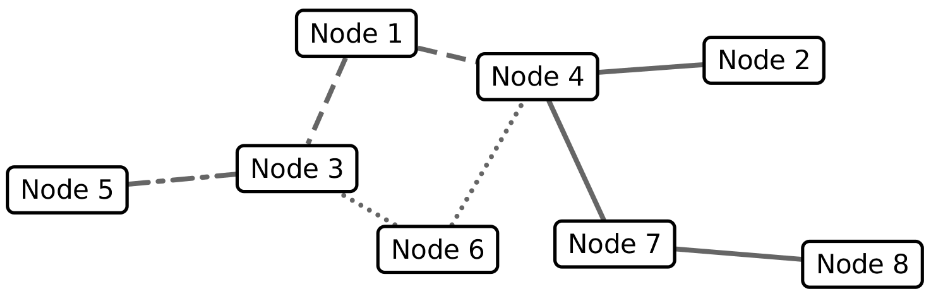


Figure 1.1: **An example of graph**, where Nodes and edges allow the emergence of complex properties that are not peculiar of the single actors alone but are a consequence of their interaction. The different kinds of lines indicate two different shortest paths. They differ because of the node involved in the path but not in the length. Hence they are considered, in terms of distance, the same path. They are different in terms of node composition since the first path, the dashed one, starts from the Node 5 and passes through the Node 3, Node 1 and reach the Node 4. The other path, i.e. the dotted line, starts from the Node 5 and passes through the Node 3, Node 6 and reach the node 4. In this sense they are considered different paths.

An edge can be associated to a numerical value, that can be defined *weight*. Weights can be represented by a function $\omega : E \rightarrow \Re$ that assign a value $\omega(e)$ to each edge $e \in E$. Edge weight describes a specific feature, depending on the context. For instance, weights can abstract the cost that is necessary to travel from u to v, they may represent a distance or, finally, the strength of a chemical bond.

Nodes have some specific properties that emerge from the topology of the network and that can be investigated by means of specific mathematical tools. In an undirected graph the *Degree*, denoted by $d(v)$, is the number of edges that have v as an endvertex. In a directed graph the *out-degree*, denoted by $d^+(v)$, is the number of edges that have an origin in v and the *in-degree*, denoted by $d^-(v)$, is the number of edges that have v as a destination. Nodes that have very high values of Degree, if compared to the other nodes in the network are said *hubs*.

A graph could be defined through a matrix-based notation. The *adjacency* matrix $A_{i,j}$ is a 0-1 matrix that describes the connections between the nodes. It has only zero on the diagonal and is symmetric if the graph is undirected. The position (i, j) is equal to 0 if there is no edge connecting the node i and j, 1 otherwise. The adjacency matrix that represents the graph in Fig. 1.1 is:

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

By using the edges it is possible to *walk* around the graph and connect different nodes by means of *paths*. Let $e_1, e_2, \ldots, e_n - 1$ be a sequence of elements of $E(G)$ for which there is a sequence $v_1, v_2, \ldots, v_n$ of distinct elements of $V(G)$ such that $\phi(e_i) = a_i, a_{i+1}$ for $i = 1, 2, \ldots, (n-1)$. This sequence of edges is called a *path*. There are different kinds of paths depending on their characteristics. A *shortest path* is a path, chosen by a set of paths, that minimises the number of steps that are necessary to connect two nodes. A mathematical properties that depends on the paths is called *diameter* and represents the longest path between two nodes in the graph.

There are also different kinds of graphs that have very different properties. In the time, some theoretical models were proposed and aims at describing some characteristics of some *real world* graphs. For instance the world wide web was abstracted as a graph and it was found to have a very peculiar property that depends on the Degree of the nodes that compose the network. This property, that show a specific Degree distribution that has a power-law trend, is now called scale-freeness [28]. The theoretical models that describe scale-free networks is called Barabàsi-Albert model. There are some other model like Erdős–Rényi or Watts-Strogatz that have their specific properties.

1.4 Biological Networks

Graphs, as mathematical objects, are used in order to model complex biological processes that involves different kinds of objects. It is possible to imagine a living cell as a set of interacting subcellular components like, for instance, membranes, cytoskeleton, genetic material, i.e. the nucleus, and several kinds of organelles that are necessary to maintain the cell a living component for tissues and for organisms. Also it is possible to model the interactions between different cells like neurons in order to model the complex structure that originates all the complex tasks our brain daily carry out. Physiological interactions and pathological interactions are all faces of the same system and as the model become more and more detailed, we gain more details that permit to investigate its properties.

From the point of view of the single components, i.e. the nodes, that in biology is a molecular point of view, it is possible to observe how a lot of different, interacting components that are proteins, metabolites and other kinds of molecules give rise to complex networks that have significant emergent properties. Indeed, different networks work together at different levels in order to keep the homoeostasis, that is the necessary equilibrium that allows life. These different layers are strictly connected and it is impossible to define where a network begins and where it ends. Cell signalling, metabolic pathways, genetic background work together and cannot be separated. Of course, for carrying out research it is necessary to obtain a model of the process which is under investigation like, for instance, a specific pathology but it is important to consider that this is just a tile in a very big and complex puzzle.

On the other hand, it is possible do define a sort of hierarchy or to describe some general, and common, set of biological networks depending on the kind of molecules involved. So, for instance, starting from the genetic level it is possible to define gene interactions networks as those networks that are used to describe the interactions between DNA and RNA sequences. Genes constitute the basis for encoding the aminoacids, that are the smaller blocks that build proteins. Then we have protein-protein interaction networks that are those networks depicting how proteins interact each other in different metabolic and regulatory networks. Finally, it is possible to define a set of networks that consider different kinds of molecules. These networks model the complex interactions that give rise to their life-cycle and consider chemical reactions and compounds. In this sense, metabolic networks are used to describe the reactions that occur between a set of very different molecules.

The idea of a cell like a set of different networks that interact each other could be used to define a sort of dependency. Starting from the lowest level, that is represented by the genetic information stored in the genes, to the high level, represented by the duties carried out by the proteins. In this context, the aim of our work is to investigate the topological characteristics of these protein-protein interaction (PPI) networks in order to extract information starting from experimental data[29] to comprehend how the interactions give rise to complex properties. Also, we are looking for shared characteristics that may shed a light on whether similar biological mechanisms are shared between similar classes of networks.

When we refer to biological networks it is important to note that each kind of them, i.e. genes, metabolic or protein, has a specific aim and describe a different kind of interaction. Indeed, there is a great difference between signalling networks and metabolic networks. The aim of the cell signalling is the transport of a signal from a source S to target T that regulates a specific process. On the other hand a metabolic network aims at representing the flow of mass and energy that takes place in a specific chemical reaction. Generally speaking, there are differences also from the point of view of the mathematical model used to represent these processes. If we consider a signalling pathway it is easy to understand that the direction of the edges in the network is a fundamental information since chemical reactions have a very specific input and output. On the other hand, an interaction between proteins does not have, necessarily, a direction so the edges we use to model it can be both directed and undirected.

Finally, it is important to note that, even though from a theoretical point of view it is easy to separate a PPI network from a metabolic network, then in the real world this separation does not exist. In biology there is a single network that comprises different subnetworks, subunits, subparts that aim at solving a specific task that do not work alone. All these parts work togheter and it is important to consider them as a whole, even though the current knowledge, technologies, methods and algorithms do not allow to model a comprehensive view of this complexity.

Protein-Protein interaction networks

This kind of networks, also known as PPIs networks, aim at modelling interactions between proteins. Proteins are aminoacidic sequence that are generated by translating the DNA strand into specific kind of RNA. Each protein has different levels of structure, from a linear to a three-dimensional structure, that define the role it plays and its duties and, as a consequence, its interactors. Proteins tend to interact since their functions are usually the result of some sort of regulation. These interactions happen by means of electrostatic forces and, sometimes, by means of covalents bonds. An interaction can be stable or transient, depending on its duration over time. Stable inteactions generate complex structure where each subunit, i.e. the protein, has a structural or functional duty. These complexes may be homo or hetero-holigomeric depending on the number of different subunits. An homo-holigomeric complex has only a kind of protein that aggregate togheter. Hetero-holigomeric complexes have different proteins constituing different subunits. Transient interactions are, by definition, short and reversible and generate rapid modifications. Proteins interact by means of specific domains that determine whether two different proteins are able to share, or not, a link. To infer protein-protein interactions there are many experimental methodologies, i.e. yeast-two-hybrid screening [30] (Y2H) and affinity purification-mass spectrometry [31] (AP-MS).

Gene regulatory networks

Genes regulatory networks are defined through the set of molecules that, interacting with each other, cause the expression of a gene. This kind of network comprise receptor proteins, transcription factors, inhibitory factors, mRNA and several other molecular actors that are required in order to let a gene being expressed. Other kind of networks that comprise genes are gene co-expression networks that are, usually, inferred from an expression array. The interactions between the set of genes that are considered of interest may represent, for instance, the level of co-expression between the genes in the array. In this sense it is possible to generate a network of interacting genes and investigate its properties.

A gene that is expressed generate, through a chain of biological events, a final product that is a protein. The production of proteins changes in different kind of cells so a neuronal cell express different genes with respect to a liver cell. Genes can also be expressed as a consequence of a stress, like for instance heat shock, or because of some environmental stimuli that, somehow, affect the behaviour of a specific kind of cells. All the expressed genes interact and give rise to genes regulatory networks that can be investigated.

Generally speaking, the nodes in a gene regulatory network are genes, but some other actors may be involved in this kind of network like, for instance, different kind of RNAs or enzymes involved in RNA transcription and translation. The edges between these actors represent the activation or inhibition of reactions and can also be represented by boolean functions. It is also possible to model gene co-expression networks where the links represent the interaction between two genes whose expression is related.

Metabolic networks

Metabolism is represented by the set of chemical reactions that take place in an organism. Thanks to these enormous amount of biochemical transformations like, for instance, the citric acid cycle (see Fig. 1.2[1]) a series of biological functions ranging from growth, reproduction, and maintenance of cellular structures allow to respond to environmental stimuli.

Metabolism is a general term that includes two kinds of chemical reactions depending on the input and the output. Catabolism aims at producing energy by breaking down organic matter. On the other hand, anabolism uses energy to build cellular components like, for instance, proteins. So, a metabolic network is useful to describe pathways that are sets of chemical reactions that transform a component into another. Here, nodes are chemical components and edges, i.e. directed edges, are chemical reactions. The molecular components that are involved in this kind of network are aminoacids, proteins, lipids, carbohydrates, nucleotides, co-enzymes, minerals and cofactors.

1.5 Related Works

In literature it is possible to find a lot of different works, whose goal is to address different biological questions, that take advantage from network-based approaches. PPI network and, more generally, network-based methodologies are attracting a lot of interests and a lot of works are taking advantage from these techniques. A brief discussion about the current methodologies and the current

[1]`https://commons.wikimedia.org/wiki/File:Citric_acid_cycle_with_aconitate_2.svg`

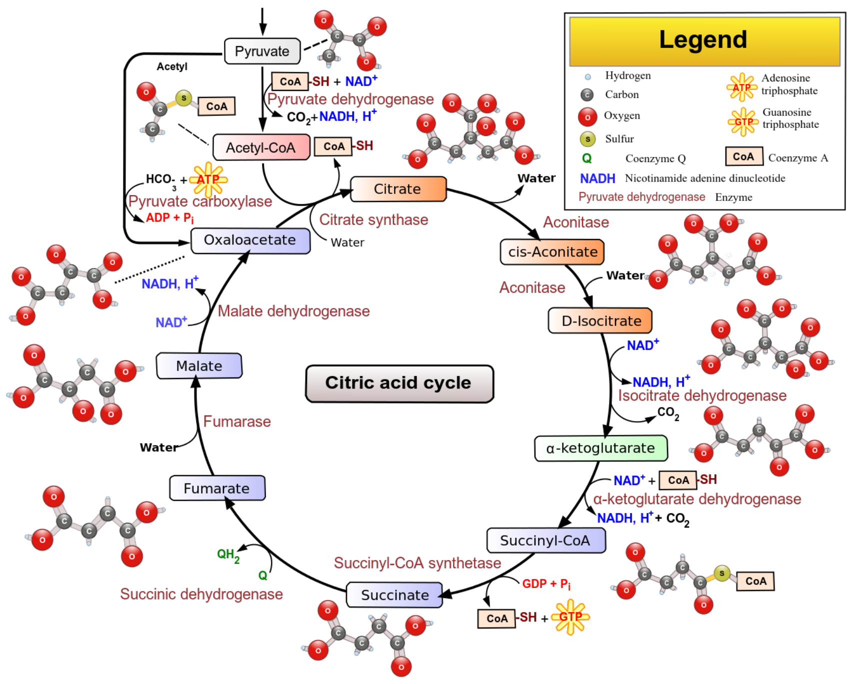


Figure 1.2: **Krebs Cycle** depicted as a network. The nodes represents different kinds of molecules. The edges are directed, since a chemical reaction has a very specific direction.

research that is carried out is necessary in order to place our work in this important and challenging field.

Generally speaking it is possible to affirm that current approaches and applications of network biology are strongly related to the field of biomedicine. A lot of efforts are carried out in order to investigate the molecular mechanisms that underlie the insurgence of the pathological state. All kinds of cancer, neurological disorders, metabolic disfunctions are currently modelled as networks of interacting metabolites, genes, proteins, in order to investigate their properties and dynamics. These are works that take advantage of several methodologies like -omics technologies, laboratory experiments and in-silico models in order to obtain a comprehnsive view of a disease. There are also more theoretical works, since graph theory is an hot topic not only in the biomedical field. Computer scientists are also working on new softwares and analysis to make them more accessible to the broader audience. In this sense, it is possible to create some subsets of works that result in different classes of papers. Major topics could be: introduction to systems biology, biological network analysis, biological network modelling, applications in biomedicine, softwares and databases.

A very interesting set of papers, that can be considered as manuals for the network analyst, refer to those manuscripts that aim at giving a guide for the readers throguh the main methodologies that could be used to infer the networks properties. There are a lot of this kind of works. For instance, the work of Pavlopoulos et Al. [32] aims at summarising all the mathematical properties that are peculiar of graphs and how they can be used in order to investigate, in a context of biological networks, the specific properties of set of interacting molecules. There are very different kinds of properties that interest a network. These properties can be global and are computed over the whole network, or local and consider the characteristics of a single node. Also it is possible to infer properties that arise from the interaction of set of nodes like clustering and network motifs.

Other examples of these kind of works are [33, 34, 35] that have a main focus on mathematical properties of the networks and describe several graph-based techniques that allow extracting networks properties. Generally speaking, it is possible to affirm that graph theory allows to model different kind of biological processes and permits to investigate their topological characteristics to rank the nodes in terms of topological importance, to extract specific subset of interacting nodes which may form complex biological entities, and so on. This subset of papers include some other works like [36, 22, 37]. They aim at describing the possibilities that systems biology offers by giving some, very interesting perspectives about this emerging field. It is important to note that these papers, like many other published works we are not including here, were published some years ago, when network-based methologies were still emerging as a new approach to study molecular interactions. Indeed, systems biology moved its first steps in the late 1900 and became an established field of research in between the last decade of 1900 and the early 2000. The works we listed belong to this period and should be considered as introductory to the field of network medicine.

Another very interesting class of works, related to the field of biological network analysis, also based on the investigation of network mathematical properties refer to the studies of the emerging properties and network modules. The, probably, most famous work about network motifs was written by Milo et Al [38]. Here the researchers investigated a wealth of different networks, ranging in a lot of different fields like biochemistry and engineering. What they found refers to the so called *network motifs*. Network motifs are recurring patterns of interactions that appeared in all the networks they analysed. These motifs represents different ways, that are shared between different kinds of networks, to process informations. Feed-forward loops, chains, and triads appeared as patterns of interactions allowing the proper functioning of the networks. Another, very important paper from Bhalla et Al. [39] refers to the emerging properties of biological networks. Here, researchers highlighted the role of emerging behaviours that are strictly dependent from the interactions between the nodes. They described how the signal is processed in different kind of signalling networks and they affirmed that the biochemical response to stimuli are stored into these networks of interacting molecules and give rise to complex responses that enable biochemical activities. Some other works, i.e. [40, 41] focused on the emergence of network modules and other recurring patterns in networks that allow understanding the role of the nodes and the interactions between different parts of the network.

Modelling of biological networks that comprise the highest and more detailed set of interactions is a hot topic. A very interesting project that aims at mapping the whole interactome in several organisms is currently ongoing. The first paper describing this effort, i.e. [42], shows how researchers are mapping the human interactome by using high quality interactions that are generated through an experimental approach. All these data are avaiable on the project website. Interactome is a term that indicates the whole set of molecular intractions that take place inside a specific cell. In this sense it is possible to dissect the term in different meanings. There exist protein interactomes, e.g. proteome, disease interactomes, e.g. diseasome, etc.... A lot of works appeared, aiming at giving directions about the investigation of interactomes.

An interactome that aims at investigating the interactions between different diseases becomes a diseaseome, like the one described in the work from Janjić et Al. [43]. Here the researchers carried out a survey in which they describe different kinds of associations between diseases. What emerges from this paper is that diseases could be linked through shared metabolic processes, miRNAs or genes. The idea is to link two disease if they have something in common and then, to investigate the properties of the generated network in order to uncover the relationships between pathologies and the main actors involved in these connections.

Huge protein-protein interactions networks become proteomes like the networks described in the work from Yook et Al. [44]. In this work the researchers aimed at comparing four different interactomic databases in order to evaluate the large-scale properties of the interactions occurring in yeast, i.e. *Saccaromyces cerevisiae*. What they suggest is that the networks constructed from the different databases share a common, large-scale topological structure. Also they found that manually curated datasets show a stronger relationship between topological role of a node and its function/localisation. What emerge from these two works is that they are not focused on a particular biological process. The goal here, is to uncover the general properties of these huge networks and to investigate similarities and differences between the interactome in different organisms.

Other works that take advantage from the interactome-based approach were written by Liang et Al. [45] and by Sharan et Al. [46]. Here the point was to compare different interactomic networks, based on protein interactions, of different organisms in order to investigate the similarities. Liang and colleagues defined a pipeline that allows comparing these kinds of networks while Sharan and colleagues applied different methods for network alignments. What came out is that different networks share common characteristics in terms of evolutionary conservation at network level. These two examples show how these approach should be used in order to infer general insights and are not useful when it comes to infer particular molecular target that could be useful in specific diseases.

But, as we said, networks could be applied to a lot of different fields and in biology they can be used to model any kind of process and to perform very different kinds of analysis. For example in Scardoni et Al. [47] a semi-dynamic approach was used to simulate the inhibition of proteins. Scardoni introduced the notion of Interference to investigate the topological effects that are caused by the removal of one, or more, node in a network. In the paper they give some interesting examples of positive and negative Interference and describes its reverse side that is the Robustness. Biologically the interference process can be associated to the inhibition of a specific protein and, in this work, such kind of knock-out experiment were applied to a a protein-protein interaction network. They studied the properties of the network that is involved in the regulation of the integrin activation in human primary leukocytes with a specific focus on the Betweenness Interference of three kinases: SRC, HCK, FGR and JAK2. This choice was related to the fact that these kinases are negative regulators of the integrin affinity maturation and also because several specific inhibitors are available: they enhanced the value of their predictions by comparing the results with the known effects described in literature. The results show how the four proteins are highly involved in the process they studied and corroborate the utility of such approach. The described workflow is very interesting and could be used as a promising methodology when a knock-out experiment is needed, allowing the scientists to perform all the tests in-silico.

In Li et Al. [48] the researchers performed a system-level study that was aiming at uncovering the pharmacological effects of a compound, i.e. Compound Danshen Formula (CDF) currently used in the traditional Chinese medicine as a treatment for cardiovascular diseases. They proposed a complex model that combines bio-availability prediction with multiple drug-target predictions and network pharmacology techniques. Different type of data, i.e. genetical, physiological and biochemical, were used in order to build the model under study. They identified some potential molecular target, i.e. protein. Then, by using molecular dynamics they were able to validate the compound-target associations. Also, the researchers used all the chemical compounds and their targets to build an interaction network. Finally, they used such network to perform a topological analysis in order to uncover all the non obvious relationships between the pharmacological actors and targets involved in the CDF treatment.

Another kind of very interesting application is described in Nair et Al. [49]. here the researchers show a network-based approach that was used to investigate all the complex processes involved in the development of inflammation-driven atherosclerosis. The first step required the identificaion of two sets of those genes that are related to inflammation. Over these two sets, 124 candidate genes were extracted and then used as seed proteins in order to build a network. The construction step was performed by using the STRING databse as a reference. Then the network was analysed and the topological information was extracted. Then they built a regulatory network by using the transcription factors (TF) that are related to the hub genes previously identified. In conclusion the researchers were able to identify five genes that play a central role in atherosclerosis by using a network-based analysis and three transcription factors was shown to be highly involved in the network. In this example Nair and colleagues were able to obtain comparable results by using two completely different approaches that are in-vivo and in-vitro.

Generally speaking it is possible to affirm that the topological analysis of biological networks is widely applied in biology for many different reasons like, for instance, finding drug targets, highlighting the more important nodes, comparing network topological properties, and to many different kinds of biological data. Micoarray data, proteomic data, database data could be used to model a network and obtain interesting and useful insights. We decided to start from the existing models and technologies to define a new, network and topology-based approach that allows creating personalised biological models. Indeed, current models are not able to include numerical information in terms of copy of nodes. The model we propose reflect the need of a model that move the current research to a more personalised medicine approach.

1.6 Our approach

In the field of Network Medicine, static networks are used to model and to investigate the complexity of the human maladies. Inferring the topology of these networks allows computing several properties like, for instance, clustering coefficient and nodes centralities. The aim is the study of emergent properties [39] and the role of the nodes in the system. In this scenario, that is the topological analysis of the network, there exist some metrics that allow the categorisation of the nodes by the role they play in the network [50]. These metrics are called *topological indexes* or *centralities*, and are computed by using the information related to the paths between the nodes in a network [51]. These values allow *i)* to depict the role a node plays in the network, and *ii)* to describe the properties of the network like, for instance, its clustering coefficient, i.e. the tendency of a network in creating highly connected subsets of nodes, its sparseness, i.e. the number of edges compared to the number of the total edges, or the scale-freeness, i.e. a property that make the network robust and fault tolerant thanks to a hub-based topology. These are only a few examples of topological characteristics that allow obtaining a mathematical description of the process. There are a number of other indexes that are node-specific that permit to study the properties of the network at a single node level. The most known indexes are Degree, Betweenness and Closeness but there are many other metrics that are commonly used in social and economics network analysis, that permit to define nodes characteristics.

In this work we focused on protein-protein interactions (PPI) networks since it appears that proteomics and protein-protein interactions are a very promising field of research [52, 53]. The protein composition of a sample, e.g. a tissue, provides a snapshot of the state of an individual in that particular organ. The biochemical activity of a tissue results from the interactions between proteins, environment, metabolites, genes.... Currently, it is not possible to construct a comprehensive model that is able to take into account all the mentioned factors but some steps in this direction were done. Studying networks of interacting molecules allows for a more systemic view, i.e. the holistic approach, of a disease and using PPI networks allow us creating a very precise map of a disease and the proteins that are, at different levels, involved. Each protein has its own structure and functions and carry out its duty as a single unit or as part of a bigger mechanism. If the mechanism is disrupted a pathological state emerge and that is why, we believe, studying the interactions between proteins is fundamental to understand what happens when a disease emerges. PPI networks can be abstracted as graphs that describe how these biological bricks interact in order to generate complex biological functions, e.g. DNA replication, to work as transporters and carriers or as receptors in order to respond to a specific stimulus. There are different kinds of protein-protein interactions and there are different, cellular and extracellular, locations where proteins interact with different properties. Once we model a specific biological process it is important to consider the biological role a protein is playing in that specific context. One of the analysis that can be performed over a network consists in enriching the network itself, using biological information. This step could be very useful in order to define the boundaries of the analysis. When studying the mechanisms behind a complex biological process like signalling, it is fundamental to understand that there is different locations where signalling takes place. Intracellular signalling and intercellular signalling are necessary to one another but in PPI networks they could be studied separately. Signals that are sent from a cell to another give rise to cascades of reactions into the receiving cell but the network of molecules involved in these cascades could be considered separately from the complex mechanisms that allow cells communications. These reactions occurs at different time points since the intracellular cascade takes place only when the extracellular receptor received and processed the message. A lot of other biological process takes place at different time but can be modelled in the same network. The amount of data available makes it possible to build bigger network without considering these aspect but it is crucial to consider the biology behind the model.

Current experiments produce huge amounts of datasets that are described in literature and stored in different databases and file formats. Some of the available databases are STRING [54], BioGrid [55], MINT [56], HPRD [57], MIPS [58], DIP [59] and PINA2 [60]. By using this information it is possible to build comprehensive and reliable networks but, currently, this is not enough. Indeed, a network may, for instance, tell us which component is more susceptible to failure and which one is replaceable. But, in order to obtain a better understanding of the system, what we need is to develop new tools and analysis that gives to the results a strong statistical and numerical support [61]. Also experimental validation is fundamental, but it comes after the data analysis phase which is, currently, one of the main focus in biology. There is a rather consistent literature that describes biological networks investigation by using topological indexes [62, 43] but, there is not a standard approach defining how to extract biological information. The fact that topological analysis are somehow preliminary and do not provide solid findings suggests that there is substantial room for improvement [63]. Improvements may come from novel approaches to the network analysis, or new centralities. Advanced methodologies exploit the biological knowledge to add an important layer of information. In this sense, new models may be useful to link the biological aspect, for instance the experimental results, to the topology. A number of published articles ascribe to molecular network topology a central role in the global functioning of living organisms [64, 65]. In this sense, a current network analysis aims at investigating the functional outcome of biological phenomena by identifying the nodes that are more likely to be critical, or central, to cell pathways regulation. It appears that the topological role a node, i.e. gene, protein or metabolite, plays in a network de facto reflects its biological function [66, 67]. Notably, current methodologies of network topological analysis focus on static models representing sets of well-defined edges and nodes, but are not designed to incorporate the quantitative variability of molecular functional states, occurring during the experiments, or characterising individual subjects, such as different patients [68]. This fact implies that the information stored in the quantitative variation of a biological event is simply not considered and it is lost by the analysis, thus affecting the predictive power of the model. In this sense, current networks can be used to infer general insights about their global structure and functioning but are not actually useful to model, for instance, the disease onset and evolution in a specific subject [69]. This limitation currently hampers the application of network analysis to personalised medicine.

The approach we propose for inferring network properties and useful biological insights, is based on the topological information that is extracted, by using specific metrics, from a new kind of static network. These new networks, that we called *potentiated* or *personalised* networks, allow incorporating numerical information into the graph structure. This means that, for instance, in a biomedical context we are able to incorporate the biochemical variations that occur between different subjects, into a shared network, obtaining a number of new, personalised networks. The idea is to investigate the properties of the network at the individual level incorporating the single subject biochemical information. By doing so we are now able to exploit such potentiated network by using some, well established, methodologies [70] like topological network analysis and machine learning. The approach for creating personalised networks is very general and may apply to any kind of network. It allows modelling very different kinds of interaction between any kind of actor like, for instance, proteins or genes but also non biological networks too.

We focused on the properties of human pahologies. Main goal is to understand whether the topological centrality indexes can actually improve our predictive power in terms of highlighting useful biomarkers. To reach this goal, we combined the biological information, form different high-throughput datasets, into complex, topological structures obtaining 183 potentiated networks. Then, we used such potentiated networks to investigate the centrality-based information. To extract information from the topological data we took advantage of machine learning. Through a feature selection algorithm we extracted the most informative topological centralities, and then by using a classification algorithm we verified the existence of similarities, and differences, between different classes of subjects, i.e. healthy and unhealthy. The workflow we designed allows reconstructing and analysing potentiated biological networks in a context of personalised medicine.

Finally we proposed a methodology which allows creating a validation benchmark based on random networks, and a new application, specifically developed for the Cytoscape [71] [72] [73] [74] users, that allow generating and randomising networks. The application, which is NetworkRandomizer, allows generating random network and randomising existing networks. Our framework was validated in four experimental datasets, each one consisting of two classes, respectively healthy subjects and unhealthy subjects, and including proteomic datasets of amyloidosis (AM), myocardial infarction (MI), a phosphoproteomic dataset of Chronic Lymphocytic B-Cell Leukaemia (B-CLL), and a gene expression profiles dataset of pancreatic endocrine tumours (PET). We carried out several experiments based on randomly created datasets in order to understand which random model better fits our need. Finally some utilities were developed by using the R statistical language, in order to improve the existing IGraph functionalities and to let more advanced users being able to perform some of the functions that are present in the NetworkRandomizer and in the PeSca [75] Cytoscape's apps.

In this thesis we will use the terms *healthy subjects* and *controls* with the same meaning. The same apply for *unhealthy subjects* and *patients*. The network centralities and the main algorithms we used are explained in the proper sections. A brief biological introduction for each dataset we analysed is used as introduction for each dataset's description. In the Appendices, we show the main class of the NetworkRandomizer app and the R code we developed.

1.7 Contributions and other works

Contributions Summarising, our contributions are several and provided theoretical and application-driven results. We:

1. defined and exploited a new network-based model that allows the integration of experimental, quantitative data into complex biological networks;
2. obtained a mathematical proof that supports our model;
3. designed a pipeline that exploits the possibilities offered by the model;
4. validated the pipeline using four pathological, experimental datasets;
5. designed and developed a Cytoscape app that allows creating random networks to validate in-silico results;
6. created three multiplied random networks models for validating the results;
7. analysed and compared the multplied randon models with the results we obtained in the pathological contexts;
8. developed a set of functions, using R, that allow to exploit the multiplied model and another Cytoscape app, i.e. PeSca;

Our pipeline, see the points n. 1-4, is presented in a manuscript we are writing.

The Cytoscape app we developed, see the point n. 5, was described in a manuscript published, in *F1000 Research* [76]. Currently the app has reached more than 800 downloads. It was released 7 months ago.

The R code, see points n. 8, was presented, as a poster contribution, at the *NetSciX* conference in Wroclaw in January 2016.

The point n. 6 and 7 are still under evaluation since the results we obtained are still not suitable for a publication.

Other works Some other network-based works were done. One of them concerning networks of miRNAs was accepted for publication, in *Oncotarget*, in December 2016. Another work related to miRNA is ongoing. These two works were/are carried out with a research group based in Verona.

A third project is ongoing, and is part of a joint project with an Irish research group. The project aims at studying the Burning Mouth Syndrome, and other mental disorders, through an in-vitro and an in-silico approach. We are working on the network-based analysis.

Finally, the work that inspired the random networks simulation, required the development of an algorithm, precisely a Monte Carlo Simulation, for the validation of some biological findings related to the Alzheimers' disease that was published in *Nature Medicine* [77]. It was done in conjunction with a research group based in Verona.

Part I

Analysing multiplied networks using topological centralities

Chapter 2

Multiplication model

2.1 Introduction

The core of this work rely on a new network model which consists of a potentiated topology. This is achieved by using quantitative experimental values that are integrated in the network as nodes attributes. The starting criterion of the study is based on the consideration that the topological structure of a PPI network is essentially determined by the interactions between protein domains [78]. Since the specific structure of a protein domain is genetically determined [79], then the network topology is genetically determined as well. Consequently, given an invariant protein composition in different datasets obtained by analysing the functional state of defined set of proteins, in different experimental conditions the inferred PPI network will be invariant too. Thus, the topological analysis of the reconstructed network will generate identical results even though the datasets provide subject-specific molecular fingerprints. This fact prevents considering the quantitative variation of molecular states, often related to individual genetic background, as part of the topological analysis leading to an informational loss. To enhance the predictive power of network topological analysis and improve its application in the context of personalised medicine, we treated experimental quantitative data as a metric allowing the multiplication of static network topology. The multiplication procedure allows incorporating the molecular activation level, derived from the experimental evidences, into PPI graph models thus accounting for the specificity of the individual biochemical background. In the field of personalised medicine the multiplied networks aim at modelling what actually happen between proteins, metabolites and all the molecular actors. Indeed these actors are responsible for the homoeostasis and for the pathological mechanisms that induce altered states. Each protein, in each tissue, has a very specific expression level that determines the number of molecules. The amount of molecules defines and modulates the effect a protein has in that specific context. The number of proteins present and the level of expression of a gene are two example of biological parameters that are fundamental and that determine the correct functioning of all the biological processes that take place in a healthy cell. Even a small modification that alters this equilibrium may result in a pathological state that potentially affects the whole system, i.e. the biological and biochemical activity of the organism. So, modelling these variations become fundamental in personalised medicine since it aims at creating individual models that allow investigating the properties of a single person. By using the experimental values, obtained by single subject experiments or analysis, we are now able to abstract the biological state in form of a multiplied, i.e. potentiated, network and then investigate its properties by using the pipeline we defined.

The current, static interactome is a huge PPI network that contains thousands of nodes and hundred thousands interactions. It aims at describing the whole set of interactions that take place inside a human, healthy cell [80, 81]. Several projects, that can be compared to the Human Genome Project, aim at obtaining the most comprehensive view of the interactions between the proteins that are present in an organism. This task is not easy at all since the interactions change in time and since the technologies that allow retrieving a real, physical interaction require experiments, expensive tools, time and money and have different level of accuracy. Importantly, not all the interactions take place in a cell at the moment we are experimenting. Finally, it is important to consider that the estimated number of proteins is around 20000, and many of them are not considered as priority proteins since they are not involved in the most studied and well known pathologies. Indeed, they suffer of a lack of consideration and may not be included in expensive experiments or may not be considered important in a pathology since there is a lack of evidences. It is important to note that there are different kinds of interactions and that interactions can be retrieved in different ways and, as we said, they have different levels of reliability. For instance, by using an algorithmic approach it is possible to infer interactions from the literature, i.e. text mining. In this sense it is possible to define, under certain user-defined conditions and filtering options, an interaction if two or more, proteins appear in the same published manuscript. Also co-expression could be a parameter that is used to infer protein-protein interactions. If two genes are co-expressed in a sufficient number of experiments then the two protein are said to be interacting. It remains of primary importance to know the sources of information that are used to build a network and how a specific interaction is obtained. This is particularly important when considering non physical or experimentally validated interactions.

Generally speaking, it is possible to say that an hypothetical proteome, representing the whole set of protein-protein interactions, is not useful in the field of personalised medicine. Indeed, this kind of interactome represents a very general model describing the interactions that take place in a specific cell in a human organism. In other words, if we consider different subjects, the proteome they share is represented by a network involving the same amount of proteins where a protein corresponds to a node. A model like the one we just mentioned could, potentially, represent the whole set of molecular interactions but, clearly, lacks of the personal layer that enable the personalised medicine approach. In this sense, a personalised network analysis is not feasible since such interactome lacks of the *personal*, biochemical information. This is a substantial limitation since each person has his own peculiarities, his own biology and metabolism that are represented by the activation and expression level of the proteins. Since the study of the whole biochemical activity that takes place in an organism is not yet feasible, we decided to focus on the protein-protein interactions. Moreover, we decided to add the missing informational layer, to the static network analysis, by considering the experimental data available that are, in our opinion, a fundamental source of information. Indeed, the current high throughput techniques allow obtaining more and more personal data at decreasing prices and are becoming a very reliable tool to predict disease at their very early stages. Hence a new network model that is able to weigh the networks, by multiplying the nodes, thus obtaining the potentiated networks that model a specific experimental condition, is currently required. The aim is to enhance the research at the single subject level and to make personalised medicine effective.

2.2 How it works

Our approach is based on the idea that, in the context of PPI network analysis, two concurrent informational components exist: *i)* static, invariant, component, represented by the network topological architecture; *ii)* variable component represented by the biochemical status, i.e. activation or inhibition, of the proteins participating to the network. The level of biochemical activity, for instance the level of phosphorylation, is normally considered a further dimension, i.e. a layer, of information, but is excluded from the generation of the network topological structure. This is a problem since, in principle, the more a protein is activated the more it is involved, in terms of copy number, in the regulation of a specific network, thus implying a dominant topological role linked to its increased biochemical activity. Furthermore, the absolute copy number of molecules functionally involved in a specific biological process changes depending from individual genetic background and biochemical status and, thus, represents individual molecular fingerprint that must be considered in a context of personalised analysis. To achieve this result, it becomes necessary to add new nodes and edges. In particular:

1. a new edge is added for each copy of the node that will be multiplied. Each edge connects the source with its new copy;

2. each copy must have the same neighbours as the original, which means that each copy should be connected to each one of the neighbours the original node has;

3. each copy must be connected to all the other copies a node has.

The last point of the list, i.e. connecting each copy with all the other copies, is fundamental in order to not generate new shortest paths. If two copies of a node are not directly linked using a new edge, then a new short path, not belonging to the original network, is generated from a copy to another copy. This path will pass through the original node causing an incorrect increase of some of its centralities. For example adding ten copies of a node and not connecting them through an edge will lead to an augmented Stress and Betweenness of the original node. Indeed, 90 new shortes paths arise between the ten new nodes, and all of them will pass through the original node. Linking the copies makes sure that this incorrect behaviour does not emerge and keeps the topology of the network safe from the emergence of wrong paths.

The correctness of the approach was proven by means of a mathematical theorem. This method could be applied to several kinds of data. We took advantage from three different high-throughput technologies for modelling personalised networks. Two dataset are obtained using a proteomic profile, one dataset is derived from gene expression microarray and one dataset is derived from phosphorylation microarray. We must say that the model could potentially be used for describing a lot of different data sources. Indeed, it is not important that the model deals with pathological datasets describing set of healthy and unhealthy networks. The nodes of the modelled network may represent any kind of object and, as long as the data available allow the nodes multiplication, the methodology could be applied to a lot of different fields. For instance, it is possible to imagine that the model could work with temporal data. Each multiplied network could represent a specific time point and analysing different time points, i.e. different networks, and comparing them allows obtaining a sort of semi-dynamic representation of the same system through time. These simulation can be seen as a sort of evolution of a systems in which numerical variations, in terms of numbers of copies of a node, appear. It is important to note that the relationship between the different transitions, i.e. the time points or the networks, and their differences, or similiarities, must be liked through some meaningful measures or parameters. Indeed, it is possible to foresee that, for such applications, further assumptions are required even though the model could surely aid these kinds of investigations.

In Fig. 2.3 a static, non multiplied network is shown. It consists of four objects, e.g. biological entities, that are represented by four nodes, one for each object. Two of these objects, in particular the Node 2 and the Node 3, are weighted. The Node 2 has 2 copies, the Node 3 has 3 copies of it. So, we multiply the topology of the network in order to better model its peculiarities. In Fig. 2.3 there are different colour for nodes and edges. The green nodes represent the original network. The orange nodes represent the copies we added. The edges represented by a red, dashed line, are used to connect the original node with its copies and the copies are also linked together. The blue, dotted edges represent the edges between each copy and all the neighbours of the original node. These edges are required in order to let the shortest paths unchanged, in terms of length as the theorem proof (see Section 2.2).

By adding new nodes we are not weighting a network. A weighted network requires that the edges, or the nodes, have an associated numerical value that represent a characteristic of the relation. The edge weight could be a distance, the strenght of a chemical bond, a co-expression value or an activation energy. The weight over a node represents, like in the case of binary search trees, the label of that node or the content of a node, since this kind of trees are used to abstract a data structure storing items. Weighted edges will result in an adjiacency matrix with values that are different from 0 and 1. Weighted nodes will result in an adjiacency matrix with a diagonal that contains values that are different from 0 and 1 (1 in case of loops). By adding these numerical values we could not achieve the same goal we are pursuing, or if we can we haven't found how to do it, yet. Indeed, what we are doing is totally different. We are not weighting nodes neither we are weighting edges. We are adding new nodes and new edges to better represent the biochemical activity of the modelled interacting molecules. By doing so, we are introducing new paths that modify the whole topology of the network withouth disrupting the original structure and making the network a finer model to achieve the required level of personalisation.

The multiplication pipeline

The multiplication model offers the possibility to investigate a new kind of networks. To let other researchers exploiting the methodology we propose, we defined a workflow that allows the creation of the potentiated networks and a straightforward analysis. The workflow can be summarised in some steps. Starting from a quantitative dataset that defines the amount of detected proteins, or another molecule, the first step concerns the extraction of a list of those proteins, i.e. the *probe*, that feature a non-zero value, for each sample. In this work the samples are healthy and unhealthy subject. A protein with a value equal to zero, i.e. that was not detected in the sample, clearly cannot be represented by a node. Hence, these proteins are excluded from the network.

Once the probes are created, one for each sample, it is necessary to merge them in order to obtain the list of proteins shared into a specific class, i.e. a list of shared proteins for healthy subjects and a list for unhealthy subjects. It is interesting to note that, if a protein is missing in one subject, it is excluded. Importantly, while dealing with subjects, it must be considered that it is possible to have different kinds of unhealhty subjects, depending on the clinical characteristics of the disease. For instance the patients, affected by amyloidosis, in the AM dataset we analysed, could be divided in κ or λ amyloids. Here, we decided to extract an average list of proteins for each subclass by intersecting the probes for each subset of unhealthy subjects. By doing so we were able to obtain very specific models, even though they are, in the end, all labelled as "unhealthy". Finally, we obtained a probe of shared proteins for each class, i.e. healthy and unhealthy, for each dataset.

After the merging phase, a mapping of the probes into a bigger PPI network permitted the construction of several networks, one for each subject, representing the interactions between the proteins in each probe. To achieve this goal, each average network, i.e. the result of the merging phase, was mapped into the Human Interactome. The matching proteins are extracted with their edges, in order to create a subnetwork of the interactome that consists of the proteins in a probe. We obtained an average network for each class of subjects. It is important to note that all these networks consist of only one connected component, which is mandatory in order to carry out a proper topological analysis. The interactome used for the mapping phase includes only proteins with a valid Gene Ontology (GO) name, thus permitting an easy identification of each node. The complete GO annotations, i.e. the .goa file, were validated on April 3, 2015. This network is a human interactome comprising 14960 nodes and 291952 interactions.

These interactions are physical and the network was compiled by using different databases, i.e. PathwayCommons_hs, MiMI, BCI, DIP, BioGRID, HPRD, HiNT, UniHI, ConsensusPathDB, and information from literature [82].

Finally, by multiplying the average networks using the experimental information, we were able to construct the potentiated networks. Each protein, of each network, has its own quantitative value that depends on the experimental conditions. It is important to note that each value was augmented by a unit in order to let the values in the range $0 < value < 1$, to become, when rounded, at least equal to one. Indeed, the minimum number of copies a node could have is one, so values that are lower than 0.5 would be rounded at zero thus resulting in a missing node.

Once the network construction and analysis was done, the machine learning phase took place. To perform a meaningful comparison, we extracted a list of those proteins that are shared between the healthy and the unhealthy subjects, for each dataset. This allowed the extraction of a set of features that refer to those proteins that are shared between the two average networks. Indeed, the use of these proteins as a common reference enabled us investigating the similarities and the differences that occur between the two classes of networks.

Why it works, a mathematical proof

In order to let the multiplication methodology being sound and mathematically solid we proved, by means of a mathematical theorem, that the addition of copies of existing nodes, to the original topology, does not affect the global properties of the network, if the addition follows some defined conditions.

The idea

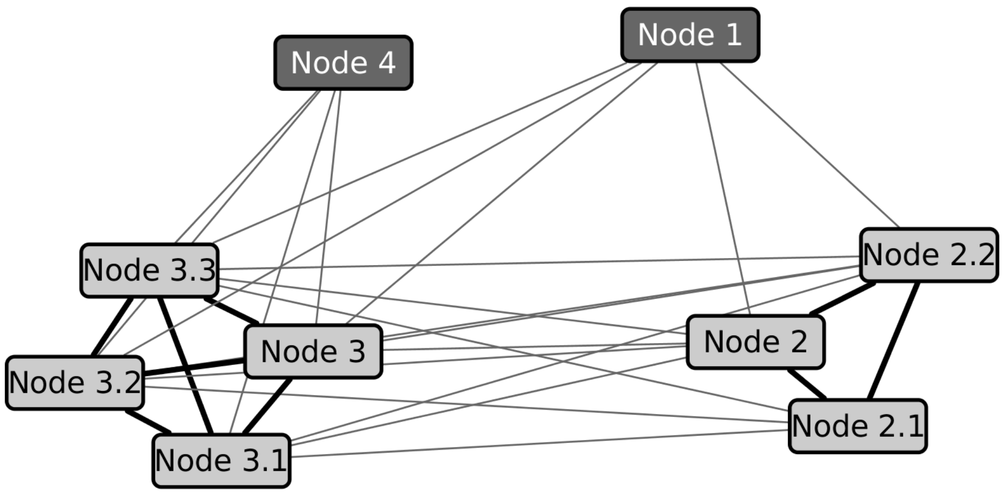


Figure 2.1: **Some cliques**, are presented in this network containing two different set of fully connected subgraphs. Here, they are shown through the grey nodes and the black, solid lines represent the interactions.

Multiplying nodes allows creating potentiated networks whose goal is modelling a subject, i.e. a person, or a specific process. More interestingly, the methodology allows investigating the quantitative variation in terms of total amount of objects that interact. It is very important to note that when adding copies of nodes to an existing network, under certain conditions, we are not modifying the basic topological structure in terms of shortest paths length. Even though the length of the path is not affected, properties like the average distance:

$$l_G = \frac{1}{n \cdot (n-1)} \cdot \sum_{i \neq j} d(v_i, v_j)$$

are affected since the sum of the distances, i.e. $\sum_{i \neq j} d(v_i, v_j)$, increase as the number of nodes increase, i.e. n, hence the average value varies accordingly. By adding new nodes while keeping the shortest paths length fixed does not guarantee that all the properties of the network are fixed. The shortest paths, intended as sequences of nodes, between the nodes remain unchanged but, as shown, some other characteristics, highlighted by many established topological centralities, become peculiar of a specific, multiplied topology, i.e. average path length, Stress, Betweeness and all the shortest path based centralities in general. These two aspects allow to study the actual topology of a specific process but, at the same time, allow the integration of the fingerprint representing, as we intended, the single subject biological state.

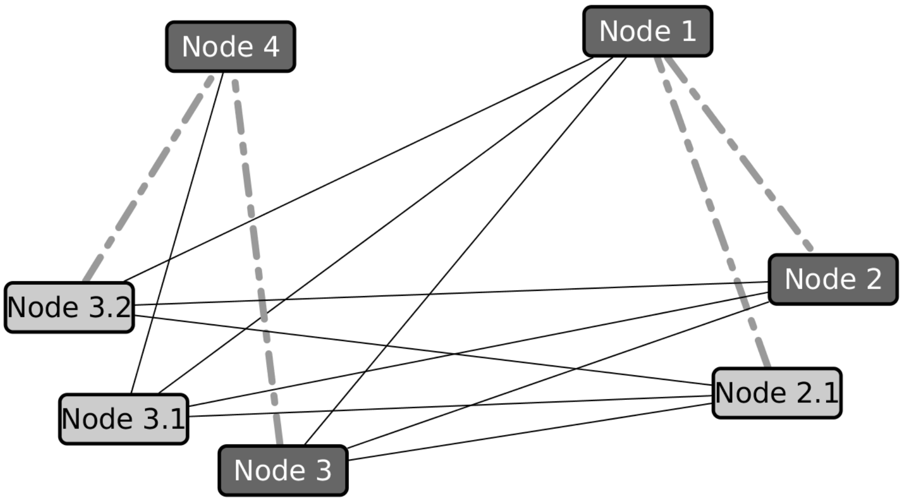


Figure 2.2: **An incorrect multiplication** is shown in this network. The grey dotted-dashed lines represent two new shortest paths that emerged because the new copies were not connected to each other and to the original node. These new short paths do not belong to the original network and lead to a disrupted topology.

In graph theory what we are representing as a multiplied node, i.e. a set of nodes that are fully connected, is called *clique* (see Fig. 2.1). Cliques are subgraphs in which every two vertices are adjacent and are structures that are investigated in graph theory. More formally a clique C in an undirected graph $G = (V, E)$ is a subset of V such that $C \subseteq V$, where every two distinct vertices are adjacent.

Finally, there are some theorems that concern the properties of these cliques but we were not able to find a link between these kind of subgraph and the length of the shortest paths. Because of this missing information, we decided to prove our own hypothesis. This result should be considered as an additional evidence that supports our methodology.

Our theorem basically prove how, by adding copies of nodes to an existing topology we are not modifying the length of the shortest paths, hence we are somehow keeping the underlying structure comparable between different subjects. This is fundamental in order to define and analyse a network which is shared between a set of subjects or biological process belonging to the same class. The number of copies, for each node, vary depending on their biology and it can be considered the fingerprint for that specific subject, or sample. To prove our theorem we used a proof by contradiction.

Definition

Given $v \in V$, define a graph $G' = (V', E')$ with k copies of v by:

$V' = V \cup \{v_1, \ldots, v_{k-1}\}$,

$E' = E \cup \{(v, v_i) \mid i = 1, \ldots, k-1\} \cup \{(v_i, v_j) \mid 1 \leq i, j < k, i \neq j\} \cup \{(v_i, w) \mid (v, w) \in E)\}$.

Proposition

By adding k copies of v to G, we are not creating new shortest paths in G' that are shorter, or longer, than any of the shortest paths already existing in G.

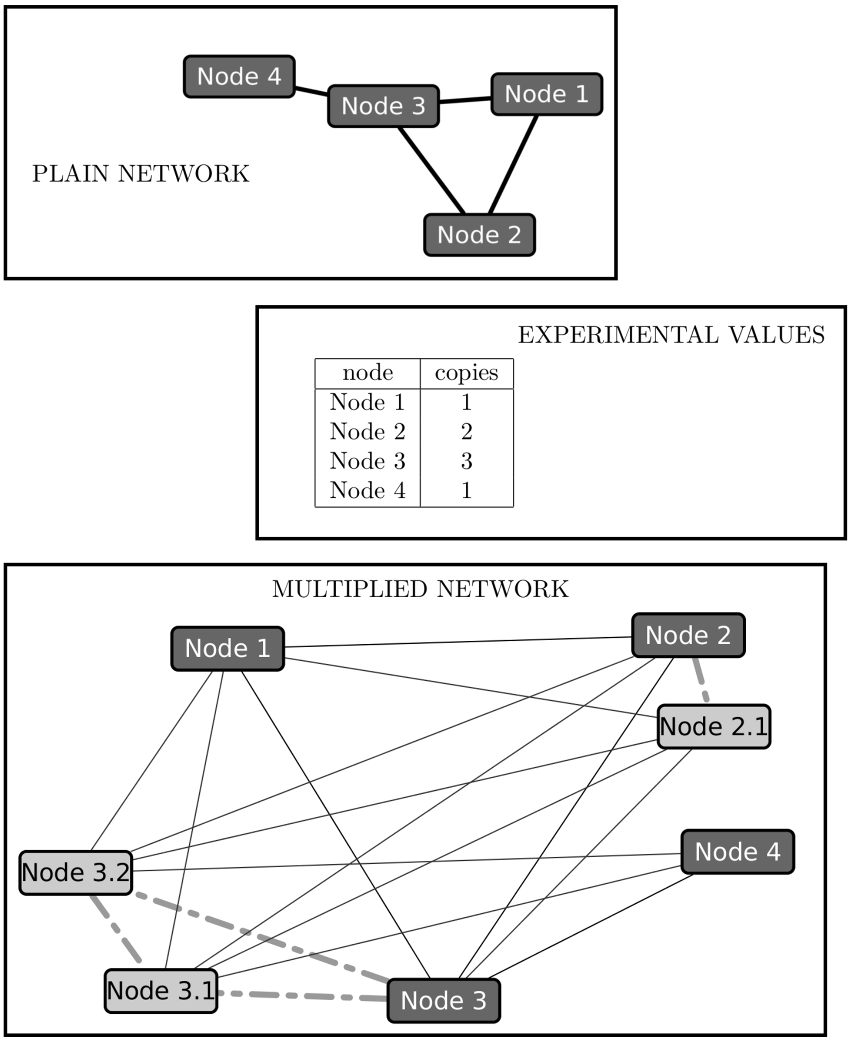
Proof

Assume that exists a shortest path $p' \in G'$ between some $u, u' \in V$ such that $length(p')$ is less than the length of any path between u and u' in G. Necessarily p' passes through one of the v_i, and two of its incident edges, such that $p' = (u, \ldots, x, v_i, y, \ldots u')$, for some $x, y \in V$.

Now define $p = (u, \ldots, x, v, y, \ldots, u') \in G$, which is a path in G. Note that the edges (x, v) and (v, y) exist in G since $(x, v_i) \in E'$ and $(v_i, y) \in E'$.

Clearly $length(p) = length(p')$ hence the assumption must be wrong.

It is important to note that the addition of a set of edges that connect all the copy nodes is trivial. If we are not adding these edges then new shortest paths will pop-up and allow the new copies of each original node to communicate together and with the original source (see Fig. 2.2). This fact have to be avoided since a multiplied network that does not respect the assumptions we exposed becomes totally different from the starting network and must be considered as a completely unrelated network. In these cases the multiplication and the analysis become pointless. By adding the edges between each copy and the original node we are creating an interaction that prevents the emergence of new, different in terms of involved nodes and length, short paths and this allow to reshape the original network while considering the numerical variation of its interacting objects.



node	copies
Node 1	1
Node 2	2
Node 3	3
Node 4	1

Figure 2.3: **The multiplication pipeline**, consists of a network which is not yet multiplied, an experimental output that is used for the multiplication step and, finally, the same plain network after the multiplication step. To stick to the biological application we are describing in this thesis, we assume that this network represents the Human Interactome. It is important to note that a network like the plain one is created to describe all the protein-protein interactions in a cell and not to fit an individual interactome. Hence, to exploit the multiplicative approach, in a context of personalised medicine, it becomes necessary to multiply the nodes according to the experimental values. We presented only four nodes and two of them resulted in more copies after the multiplication. More specifically, the Node 2 is present in two copies while the Node 3 has three copies. The Node 1 and Node 4 has only a copy which means that the original node is the only copy present in the network. Each copy share the same neighbours of the original node and have an edge in common with both the original node, that is highlighed as a dashed-dotted line, and all the other copies. This methodology allows creating weighted network starting from a common backbone, i.e. the Human Interacome, and to effectively tackle the questions that the personalised medicine poses.

Chapter 3

Real data experiments

3.1 Introduction

Four different datasets obtained through high-throughput technologies were analysed, in this work. In particular, AM and MI data were obtained by using proteomic profiles and a mass spectrometry analysis. BCLL data were obtained through antibodies microarrays for the investigation of proteins phosphorylation levels. Finally, PET data were obtained from gene expression microarrays. More specifically, the AM set is composed of 11 healthy and 24 unhealthy subjects affected by systemic amyloidosis [83]. The MI dataset is composed of different samples of cardiac tissue derived from the left ventricle of 18 farm pigs, i.e. *Sus scrofa*. The BCLL dataset contains 10 healthy subjects and 31 affected patients. The antibody microarrays data, inclusive of Z-scores and ratios, were previously filtered and analysed by Kinexus. The PET dataset included of 71 primitive pancreatic endocrine tumours of which 46 samples with an histological grading of G1 and 25 with a G2 grade [84, 85]. We modelled healthy and unhealthy average networks since it permitted us to build, and hence analyse, bigger networks in terms of number of participating proteins. Finally, only the nodes that are shared along all the networks, for each specific disease, were used for the classification step.

The interactome used for the mapping phase includes only proteins with a valid Gene Ontology (GO) name, thus permitting an easy identification of each node. The complete GO annotations, i.e. the .goa file, were validated on April 3, 2015.

Myocardial Infarction dataset

Myocardial infarction, commonly known as hearth attack, is one of the major cause of death in the world. The term *infarction* refers to the irreversible necrosis of the heart muscle caused by an ischaemic event. The term *ischaemia* describes an insufficient blood supply to tissues that causes a lack of oxygen and glucose leading to the necrosis of the affected tissues. In the case of the myocardial infarction, the ischaemia causes the death of the myocites, that are the cells that compose the hearth tissue. The insufficient blood supply, in the case of the myocardial infarction, may be caused by the rupture of an atherosclerotic plaque that blocks the blood flow through a coronary artery. Also coronary artery spasms are considered a cause of myocardial infarction. It is important to note that an ischaemic event does not affect myocardial tissues only but it may happens to several other tissues. This pathology is related to several factors like the high blood pressure, smoke, diabetes, obesity, and high levels of cholesterol. This pathology is highly dependent on the life style of the individual but some other diseases, like for instance diabetes, are risk factors. Gene variants were also related to myocardial infarction.

The dataset we are investigating was obtained from myocardial tissues in the context of a study whose goal was testing the effects of the implantation of human stem cells in an animal model, i.e. *Sus scrofa*. The healthy (N) samples were collected from normal tissues while the 18 unhealthy samples were collected from tissues that were affected by myocardial infarction. The infarctuated samples were divided in three groups depending on different treatments. Group U was treated with Phosphate Buffered Saline (PBS); group F with pre-treating placenta-derived human mesenchymal stem cells (FMhMSCs); group H with FMhMSCs preconditioned with a mixed ester of hyaluronan, butyric and retinoic acid [86]. Since not enough interactomic information were retrieved for the organism *Sus scrofa*, we decided to map the proteomic expression profiles over the Human interactome, since the pig is used as animal model for studying several human diseases [87, 88] and due to the fact that these two organisms are genetically very similar [89]. We checked the protein names using the STRING database in order to validate their names across the human and the pig.

The data come from a proteomic profile obtained through a gel-free, multidimensional protein identification (MudPIT) technology [90]. This methodology permits the identification of the proteins that are present into a complex mixture of peptides. These mixtures are proteolised causing the generation of a huge number of peptides. These peptides are separated by using a bi-dimensional liquid chromatography and, finally, analysed through the mass-spectrometer. More specifically, once the sample is collected, it is loaded into a specific column that is placed in-line with a mass spectrometer in order to obtain the data that are used, by computational algorithms, to determine the original content of the sample of interest.

This proteomics datasets was normalised by using the molecular weight of each protein, in order to let the amount of a protein being comparable within the subjects. Nonetheless, this is not enough to be able to compare the amounts of distinct proteins for a specific subject. This happens since the proteins have different molecular weights and it is not correct to compare two values that are normalised with different weights. This fact could potentially affect the analysis, since we are comparing a protein over the subjects we just needed a common reference along all the subjects, and finally we decided that using the molecular weight was the best solution.

The MI dataset required to construct four different probes, one for the healthy subjects, i.e. H, and three for the different kinds of patients, i.e. H, F and U, depending on the different treatment, by following the general work-flow listed above. Since the data are, again, originated by using proteomic profiles, then the normalisation was performed by using the molecular weight. No further normalisation or node addition was performed. Four average networks were built, one for the healthy subjects and three for each subset of treatments in the unhealthy subset.

The nodes involved in the six subjects that we modelled, for the F subset:

- PKM, SPTBN1, CFL2, ATP5A1, ALDOC, ATP1A1, ACO2, HSP90AB1, TUBB, EEF1A1, SOD2, ACTN1, TUBA1C, HBB, ACTB, VIM, COL1A1, PGK1, PGAM1, TGM2, VCL, HBD, MYL3, ACTG1, ACTC1, CYB5R3, CDC42, MDH2, MYH2, ACTN2, APOA1, DPYSL2, LAMC1, IDH3A, PARK7, HSP90AA1, COL6A3, HIST1H4A, YWHAE, PGAM2, COL6A2, DES, PDLIM1, HSPB1, HADHB, MYH7, MYH1, C3, ENO1, ANXA2, CAV1, GAPDH, EEF2, MYL1, FH, TPM1, IDH2, ACTA1, YWHAZ, MDH1, TPI1, LUM, ACTG2, HSPA1A, AHNAK, HSPG2, CRYAB, HADH, HK1, ENO3, SLC25A5, PDHB, HSPD1, HSPA1L, LDHB, LMNA, PDIA3, HNRNPK, UBA52, HADHA, ETFA, TPM3, HSPA8, PGM1, SPTAN1, HIST1H2BC, GPI, ATP5B, TPM4, TPM2, HIST1H2AB, HSP90B1, ALDOA.

The nodes involved in the six subjects that we modelled, for the H subset:

- ANXA5, ATP5A1, PRDX6, TNNI3, TPM2, ATP5B, ETFA, VIM, HSP90B1, EEF1A2, MSN, ATP5F1, TTN, DES, HIST1H1D, MYH9, ALDOA, MYH3, OGDH, MYH2, ACTN2, ACTG1, NDUFS2, TPI1, HADH, CRMP1, MYL1, CS, EEF1A1, CP, SOD2, SLC25A3, COL6A2, COL6A1, HSPA2, GDI1, ACTB, SPTB, MYOZ2, LGALS1, C4A, GDI2, FGA, SERPINA3, COL1A1, TGM2, TUBB, PRDX3, CAMK2G, PGM1, MYL2, ACTG2, AK1, MDH2, HSPA5, MYL3, HP, UQCRC1, MYH11, PGK1, PGAM1, YWHAZ, VDAC2, COL6A3, ENO3, LAMC1, ATP2A2, HIST1H2BC, PDIA3, C3, LDHB, FN1, IDH2, ATP1A1, CLTC, CSRP3, ITIH2, SEC23A, SPTBN1, TLN1, MYBPC3, IDH3A, TNNT2, CRYAB, ACTN1, ENO1, ATP5O, FLNC, HNRNPU, TUBA1C, ACTN4, HNRNPK, GPI, HSPD1, PHB2, LMNA, SLC25A5, TPM3, HIST1H4A, ACADL, CALU, COX5A, HIST1H2BN, A2M, LUM, PGD, AHCY, ACTC1, PKM, HADHA, APOA1, HSPA8, EZR, SPTAN1, DPYSL2, ANXA1, HSPA1A, TNNC1, COMT, AHSG, CYCS, VCL, ACO2, ORM2, ACAA2, CFL2, FH, HSP90AA1, MYL6, DLD, HSPA1L, UBA52, YWHAE, VDAC3, TPM1, ALB, HBB, NNT, SRSF2, ACTA1, PDLIM1, TXN, MYH1, TPM4, ANXA2, TF, HSPB1, P4HB, MYH7, ITIH4, VDAC1, WDR1, DPYSL3, GAPDH, MDH1, ACADM, DLAT, CTNNA1, HADHB, SDHA, LDB3, GSN, UQCRC2, HBD, SLC25A4.

The nodes involved in the eighteen subjects that we modelled, for the N subset:

- LMNA, UBA52, ACTN2, MYH2, ACTG1, HIST1H4A, TPM2, DES, ANXA5, VCL, YWHAE, IDH2, HIST1H1D, VIM, PGAM2, HNRNPA2B1, LDB3, HIST2H2AB, GSN, APOA1, CP, FABP4, DPYSL2, GDI1, ENO3, A2M, GDI2, CLTC, CFL2, C3, DPYSL3, TF, LUM, ALDOC, MYL3, TPI1, HIST1H2BN, HBB, SPTBN1, TUBA1C, ACO2, TUBB, EEF1A2, ETFA, ACTB, ACTN1, HADHB, HIST1H2BC, ACTN4, HADH, ACTG2, LDHB, EZR, ANXA1, HSPA1L, HADHA, CAMK2D, HSPA8, ALB, TPM4, PGK1, PRDX6, PGAM1, HSPA5, MDH2, HP, SPTAN1, HSPA9, HSPD1, HSPA1A, YWHAZ, ATP5A1, TPM3, HSP90AA1, ACTA1, ATP5B, VDAC1, SOD2, EEF2, GAPDH, ATP5J2, TPM1, GPI, COL1A1, TGM2, P4HB, MYH7, VDAC2, MYH1, FGA, COX5A, ANXA2, HBD, LGALS1, CRYAB, CS, HSPB1, ACTC1, PKM.

The nodes involved in the six subjects that we modelled, for the U subset:

- MDH1, ETFA, PGK1, APOA1, FGA, ANXA5, HSPA5, PGAM1, MDH2, COL6A2, HSPA8, HIST1H1A, PRDX6, HBD, ACTG1, MYH2, DPYSL2, ACTN2, IDH2, TGM2, HBB, MYBPC3, LGALS1, ATP1A1, LAMC1, FHL1, PKM, COX5A, MYL3, TPM2, CALU, HSPA6, HSP90B1, OXCT1, ACTC1, YWHAZ, HIST1H1D, ATP5B, DES, PDLIM1, A2M, PGAM2, ATP2A2, ALDOA, ORM2, VIM, HSPA1A, MYL6, TTN, HADHA, CFL2, CRYAB, PDHB, HADH, VDAC1, FH, S100A1, HP, LDB3, GPI, LMNA, HNRNPK, HMGB1, HSPB1, ACTB, ACTA1, TPM1, FLNC, HIST1H2AB, PDIA3, ANXA2, WDR1, EEF1A2, YWHAE, VCL, LDHB, TPM4, ACO2, HSP90AA1, EEF1A1, GAPDH, NME2, SOD2, IDH3A, DLAT, H2AFZ, ACTG2, TPI1, MSN, TUBA1C, HIST1H2BN, CLTC, ATP5A1, HIST1H2BC, HSPD1, FGG, UBA52, HIST1H4A, RAP1A, TUBB, TPM3, COL6A3, ACTN1, SPTBN1, MYL2, COL6A1, AHSG, TF, SPTAN1, C3, ALB, MYH7, CAPNS1, CP, HSPA1L, ENO3, MYH1.

Amyloidosis dataset

Amyloidosis is considered a rare disease since it affects less than 200,000 people in the U.S. population[1]. The term amyloidosis refers to a various set of pathologies that are characterised by an abnormal production of a specific protein that is called *amyloid*. An amyloid appears when some proteins aggregate and stick together. Usually these proteins lose their physiological function and form these unhealthy aggregates that look like fibrils. These protein fibers accumulate into organs and tissues and cause an augmented risk of functional failure. The more they accumulate, the higher the risk for the health and the organ dysfunction. More than 20 different known human pathologies are associated to the accumulation of these amyloids. Some examples are Atherosclerosis, Diabetes mellitus type 2 and Alzheimer's disease.

Amyloidosis can be systemic or localised, depending on the site where the amyloids are accumulating. If it occurs in several places in the body then we are talking about systemic amyloidosis. If they concentrate in a single organ or tissue, the amyloidosis is said to be localised. Also there are different kinds

[1]http://www.amyloidosis.org/facts/

of amyloid proteins. Depending on these kinds of proteins the most common systemic amyloidosis can be divided in *i)* AL, where L stands for Light Chain, *ii)* AA where A stands for the Serum A Protein and, finally, *iii)* $ATTR$ where TTR represents the Transthyretin protein. The dataset we analysed refers to the AL amyloidosis. It is caused by a bone marrow disorder and the production of plasma cells. Plasma cells belong to the immune system that has the duty of fighting infections through the production of antibodies. Immunoglobulin comprehend different proteins that act as antibodies and are composed of four protein chains. An immunoglobulin has two light chains, either kappa or lambda light chains, and two heavy chains, of which there are several types. A subject affected by AL amyloidosis produces abnormal antibodies. More specifically we are focusing on ALλ and ALκ amyloidosis, with Congo red score greater or equal to $3+$. AL amyloidosis is caused by misfolded monoclonal immunoglobulin light chains that can be both λ and κ. The data were obtained by using the same MudPit technology described for the MI dataset.

This proteomics datasets was normalised by using the molecular weight of each protein, in order to let the amount of a protein being comparable within the subjects. Nonetheless, this is not enough to be able to compare the amounts of distinct proteins for a specific subject. This happens since such proteins have different molecular weights and it is not correct to compare two values normalised by using different weights. This fact could potentially affect the analysis, since we are comparing a protein over the subjects we just needed a common reference along all the subjects, and finally we decided that using the molecular weight was the best solution.

The following step concerns the retrieval of a set of proteins shared between the different classes under study, i.e. the healthy subjects and the unhealthy subjects. This goal was achieved by extracting a probe for each subject. In the AM dataset, three different average probes were obtained, one for the healthy subjects and two for the two different subsets of patients (ALλ, ALκ). The mapping step highlighted the fact that the average probes, in the case of healthy subjects, are very small in terms of nodes and edges. Merging the probes leaded to only eight shared proteins that are connected into a single component. In order to overcome this limitation, a supplementary set of proteins for each subject was added, both for healthy and unhealthy average networks. The supplementary set of proteins was obtained by considering the first neighbours of the shared protein. Once the probe is mapped in the interactome, the selection is extended to the first neighbours and the resulting network extracted. From this average network, we computed a network for each subclass, we performed a further sampling. Only those nodes that are present in a probe, i.e. a subject, are considered. The first neighbours that do not belong to a probe were discarded since it was not possible to associate an experimental value.

The nodes involved in the healthy subjects we modelled:

- FABP4, S100A10, ANXA2, ANXA1, TUBA1C, VIM, CAV1, ACTB, LMNA, BANF1, MYH9, SPTBN1;
- STOM, APOC1, SPTBN1, TUBA1C, ANXA2, HBG1, HBA1, HBD, HBB,

ANXA1, VIM, LMNA, ACTB, MYH9, APOA1, FGG, FGB, FGA, SLC4A1, ANK1;

- CRYAB, SPTBN1, APOA1, APOA2, TUBA1C, S100A10, ANXA2, HBA1, HBD, HBB, FABP4, PTRF, PLIN1, CAV1, ANXA1, VIM, LMNA, ACTB, MYH9;
- TUBA1C, BANF1, LMNA, FABP4, MYH9, SPTBN1, S100A10, VIM, ANXA1, ACTB, ANXA2;
- TUBA1C, PTRF, CAV1, LMNA, MYH9, SPTBN1, S100A10, VIM, ANXA1, ACTB, ANXA2;
- CAV1, SPTBN1, PLIN1, MYH9, TUBA1C, ANXA1, ANXA2, VIM, LMNA, ACTB;
- FABP4, S100A10, TUBA1C, LMNA, RPS15A, ANXA2, VIM, ANXA1, RPLP1, ATP5J, MYH9, SPTBN1, ACTB, NDUFA4, ATP5I;
- FABP4, S100A10, ANXA2, ANXA1, TUBA1C, VIM, LMNA, PRKCDBP, ACTB, MYL6, MYH9, SPTBN1;
- FABP4, HBD, S100A10, ANXA2, APOA2, ANXA1, TUBA1C, VIM, LMNA, ACTB, CRYAB, APOC1, DCD, HBA1, APOC3, HBB, APOA1, MYH9, SPTBN1;
- TUBA1C, PTRF, LMNA, CAV1, CALM1, MYH9, SPTBN1, VIM, ANXA1, ACTB, ANXA2;
- TUBA1C, S100B, S100A11, LMNA, FABP4, MYH9, SPTBN1, VIM, ANXA1, ACTB, ANXA2.

The nodes involved in the λ-Amyloidosis subjects:

- LGALS1, HBD, SAA1, HBB, HBA1, FABP4, S100A11, ATP5B, FGB, VTN, FGG, APOA1, APOC3, FGA, APOA4, S100A10, ANXA2, ACTB, VIM, APOE, TTR;
- LGALS1, HBD, HBG2, HBB, HBA1, FABP4, ATP5B, FGB, VTN, FGG, APOC3, APOA1, APOA4, FGA, S100A10, ANXA2, ACTB, VIM, APOC1, APOE, F2, TTR;
- HBD, HBB, HBA1, FABP4, ATP5B, APOA2, FGB, VTN, FGG, APOA1, APOC3, FGA, APOA4, CAV1, S100A11, ANXA2, ACTB, VIM, APOE, TTR;
- UQCRH, HBD, HBB, HBA1, FABP4, ATP5B, APOA2, FGB, VTN, FGG, APOC3, APOA1, FGA, APOA4, S100A10, ANXA2, ACTB, VIM, APOC1, APOE, TTR;

- HBD, SLC4A1, ANK1, HBB, HBA1, FABP4, ATP5B, FGB, VTN, FGG, APOC1, APOA1, APOC3, FGA, DCD, APOA4, VIM, ACTB, ANXA2;
- LGALS1, HBD, PTRF, HBB, HBA1, FABP4, ATP5B, FGB, VTN, FGG, APOE, APOA1, APOC3, FGA, APOA4, S100A11, CAV1, CALM1, VIM, ACTB, S100A10, ANXA2LGALS1, HBD, PTRF, HBB, HBA1, FABP4, ATP5B, FGB, VTN, FGG, APOE, APOA1, APOC3, FGA, APOA4, S100A11, CAV1, CALM1, VIM, ACTB, S100A10, ANXA2;
- COX5A, LGALS1, HBD, HBB, HBA1, FABP4, CAV1, ATP5B, FGB, VTN, FGG, APOE, APOC1, APOA1, APOC3, FGA, APOA4, VIM, ACTB, ANXA2;
- LGALS1, HBD, SLC4A1, HBB, HBA1, FABP4, S100A11, ATP5B, APOA2, FGB, VTN, FGG, APOA1, APOC3, FGA, APOA4, VIM, ACTB, ANXA2;
- LGALS1, HBD, HBB, HBA1, FABP4, ATP5B, FGB, VTN, FGG, APOE, APOA1, FGA, APOA4, VIM, ACTB, S100A10, ANXA2;
- LGALS1, HBD, PTRF, HBB, HBA1, FABP4, CAV1, CRYAB, ATP5B, FGB, VTN, FGG, APOE, APOA1, APOC3, FGA, APOA4, VIM, ACTB, S100A10, ANXA2;
- LGALS1, HBD, HBB, HBA1, FABP4, ATP5I, APOA2, FGB, VTN, FGG, ATP5B, ATP5J2, IGHG1, APOE, APOC1, APOA1, FGA, APOA4, VIM, ACTB, S100A10, ANXA2;
- LGALS1, HBD, HBB, HBA1, FABP4, ATP5B, APOA2, FGB, VTN, FGG, APOE, APOC1, APOA1, FGA, APOA4, S100A10, ACTB, ANXA2, VIM, RPLP1, ATP5I, ATP5L;
- HBD, SLC4A1, HBB, HBA1, FABP4, ATP5B, FGB, VTN, FGG, APOE, APOA1, APOC3, FGA, APOA4, VIM, ACTB, ANXA2.

The nodes involved in the κ-Amyloidosis subjects:

- MYO1C, ATP5A1, S100A11, TUBA1C, SPTAN1, HBD, ANXA1, S100A10, ANXA2, HBB, APOC3, APOE, SAA1, CLU, HBA1, ATP5B, APOA1, MYL6, ACTB, VIM, FABP4;
- MYO1C, ATP5A1, TUBA1C, SPTAN1, HBD, APOA4, ANXA1, S100A10, ANXA2, HBB, APOE, CLU, HBA1, ATP5B, APOA1, ACTB, VIM, FABP4;
- MYO1C, ATP5A1, TUBA1C, SPTAN1, HBD, APOA4, ANXA1, S100A10, ANXA2, HBB, APOE, SAA1, CLU, HBA1, ATP5B, APOA1, ACTB, VIM, FABP4;
- FGB, MYO1C, ATP5A1, TUBA1C, SPTAN1, SLC4A1, HBD, APOA4, ANXA1, ANXA2, HBB, TTR, APOC3, APOE, CLU, HBA1, ATP5B, APOA1, ACTB, VIM, FABP4;

- MYO1C, ATP5A1, TUBA1C, SPTAN1, HBD, ANXA1, S100A10, ANXA2, HBB, APOE, IGHG1, CLU, HBA1, TTR, ATP5B, APOA1, ACTB, VIM, FABP4;
- SLC4A1, HBD, PLIN1, PTRF, HBB, APOE, CLU, FGG, APOA1, FGB, MYO1C, ATP5A1, CAV1, FGA, HBA1, ATP5B, TUBA1C, SPTAN1, ANXA1, S100A10, ANXA2, ACTB, VIM, FABP4;
- CALM1, MYO1C, ATP5A1, TUBA1C, SPTAN1, HBD, PTRF, CAV1, S100A10, ANXA1, ANXA2, HBB, APOC1, APOC3, APOE, CLU, HBA1, ATP5B, APOA1, ACTB, VIM, FABP4;
- MYO1C, ATP5A1, TUBA1C, SPTAN1, S100B, S100A11, HBD, PTRF, CAV1, S100A10, ANXA1, ANXA2, HBB, APOC3, APOE, CLU, HBA1, ATP5B, APOA1, MYL6, ACTB, VIM, FABP4;
- SLC4A1, HBG2, HBG1, HBD, HBB, APOE, CLU, FGG, APOA1, FGB, MYO1C, ATP5A1, FGA, APOC3, HBA1, ATP5B, TUBA1C, SPTAN1, S100A10, ANXA1, ANXA2, ACTB, VIM, FABP4;
- MYO1C, ATP5A1, TUBA1C, SPTAN1, SLC4A1, HBD, CAV1, ANXA1, ANXA2, FGB, FGG, FGA, HBB, APOC3, APOE, CLU, HBA1, ATP5B, APOA1, ACTB, VIM, FABP4;
- MYO1C, ATP5A1, TUBA1C, SPTAN1, SLC4A1, HBD, CAV1, ANXA1, ANXA2, FGG, FGB, FGA, HBB, APOC3, APOE, APOC2, CLU, HBA1, ATP5B, APOA1, ACTB, VIM, FABP4.

B-Cell Chronic Lymphocytic Leukemia dataset

BCLL, also known as Chronic lymphocytic leukemia, or CLL, is a blood neoplasia, that affects a specific kind of white blood cells, called B-lymphocytes or B-Cells. These cells have the task of producing immunoglobulins to fight infections and disease. In CLL neoplastic conditions these cells show an altered behaviour, characterised by alteration of apoptosis, thus leading to uncontrolled progressive accumulation in bone marrow, secondary lymphoid organs and blood, causing altered immune system and other tissue functionalities. Leukemic cells, i.e. the affected lymphocytes, are generated by the bone marrow and have specific characteristics. These cells lose their immunocompetent function and do not mature like normal cells. They have an higher rate of growth in terms of number of cells and an altered life cycle which help them surviving for longer time period if compared to the normal lymphocytes. All these facts allow an increased amount of these altered cells accumulating in the bone marrow until they are released in the blood causing an abnormal number of white cells in the flow. Leukemia can be chronic, as in our case, or acute.

The technology used for obtaining the data is based on the Kinexus antibody microarray. These arrays includes more or less 500 panspecific and about 350 phosphosite-specific antibodies for each chip. They are present in duplicate in order to enhance the reliability. This kind of analysis covers some proteins that regulate some of the cell common behaviour like, for instance, the proliferation, apoptosis, stress, adhesion, secretion, and motility. The data were normalised with respect to the control conditions, i.e. healthy B-Cells, and filtered by using a z-score and a ratio between signal and rumour which is greater than 1.1. All the subjects were treated with Stromal cell-derived factor 1 (SDF-1).

These data required a different normalisation, since the data used to weigh the personalised networks represent phosphorylation levels. These values are Z-ratios obtained by comparing the Z-scores for each subject before and after the SDF-1 treatment. Since these values derive from microarrays, they refer to the intensity of the spots, for each protein and each antibody; only the spots that met the quality criteria in accordance to the Kinexus guidelines [91], were used.

Another issue was related to the fact that Z-ratios can be either positive or negative. Hence, since a node cannot be represented by a negative number of copies, we computed the absolute value for each ratio. Moreover, since a protein has more than one phosphosite that could be recognised by more than one antibody, we decided to consider the highest values among the absolute Z-ratios, for each protein. Finally, the set of probes was used to construct the average networks for both healthy and unhealthy subjects.

The nodes involved in the healthy subjects, used as controls:

- CREB1, KIT, GSK3A, GFAP, AKT1, GRIN1, ATF2, BRCA1, MAP2K4, SRC, PRKCD, PRKCB, JAK1, PRKCE, EIF4E, PDGFRA, SOX9, LCK, MAP2K6, MAPK14, PEA15, PRKCA, MET, DAB1, PTPN11, FOS, FOXO3, HTT, MAPKAPK2, AKT1S1, PTK2B, ZAP70, CTTN, BMX, CFL2, CRYAB, MAPK8, ADD1, RPS6KA5, MAP2K3, JUN, EIF2AK2, PAK1, GAP43, TP53, SYN1, RPS6KB1, PTEN, MYL12A, MAP3K5, EIF4EBP1, MAP3K11, PRKACB, CFL1, SMAD1, EIF2S1, RB1, CHEK2, PXN, PRKAR2A, MTOR, MARCKS, PGR, PRKD1, IRS1, IGF1R, MAPT, CAMK2A, ITGB1, LIMK1, KDR, PKN1, MAP2K2, MAPK3, NPM1, RPS6, STAT1, BAD, RAD17, HIST1H1A, PDGFRB, EGFR, PRKCG, MAP2K1, DOK2, PRKCI, HSPB1, MAPK7, INSR, ERBB2, PRKCH, ACACA, PTK2, STAT2, CDK1, PRKCZ, TH, BLNK, CAV2, RAC1, H3F3A, STAT3, RPS6KA1, STAT5A, CHUK, ADRBK1, RAF1.

The nodes involved in the unhealthy subjects:

- CTTN, CRYAB, MAPK8, MAP2K3, JUN, EIF2AK2, PAK1, TP53, SYN1, RPS6KB1, PLK1, EIF4EBP1, CFL1, PRKACB, EIF2S1, RB1, PXN, PRKD1, PGR, GSK3A, KIT, AKT1, GRIN1, PRKCD, PRKCB, SRC, PDGFRA, JAK1, MAPK14, LCK, PRKCA, FOS, ZAP70, AKT1S1, MAPKAPK2, HIST1H1A, MAPK3, NPM1, RPS6, STAT1, HSPB1, EGFR, PRKCG, MAP2K1, CDK1, PRKCH, ERBB2, PTK2, ACACA, STAT3, STAT5A, RPS6KA1, BLNK, H3F3A, KDR, MAPT, IRS1, PRKAA1, PKN1, MAP2K2.

Pancreatic endocrine tumours dataset

Pancreas is an endocrine gland lying behind the stomach and in front of the spine. It produces several hormones that have very important functions in regulating different biological processes. Some examples of hormones produced by the pancreas are insulin, glucagon, somatostatin, and a polypeptide that has various functions. It is formed by two different kinds of cells: endocrine pancreas cells and exocrine pancreas cells. Endocrine cells produce hormones and are grouped in clusters that are called *islets*, i.e. the so called *islet of Langherans*. The exocrine cells produce enzymes that are released into the intestine. The Pancreatic Endocrine Tumours we are interested into affect these specific kind of cells. There are two kinds of PET depending on the fact that there is an extra production of hormones, i.e. functional tumours, or that there is not an extra production of hormones where the symptoms are caused by the growth of the tumour itself, i.e. non-functional tumours. Functional tumours can be benign or malignant, non-functional are only malignant. Functional tumours, depending on the hormones that are produced in an higher amount, can be divided in: Gastrinomas, Insulinomas, Glucagonomas, VIPomas and Somatostatinomas. VIPomas and Somatostatinomas are quite rare, the treatments are very similar hence they are considered as a single group. The World Health Organization divides these kinds of tumours in three different groups, depending on the histological classification. These groups are *i)* Well Differentiated (Low Grade, G1), *ii)* Moderately Differentiated (Intermediate Grade, G2), and *iii)* Poorly Differentiated (High Grade, G3). In the dataset we are analysing we have only two classes that represent G1 and G2 samples.

The data comes from a gene expression microarray. This technology allows to measure how much a gene is expressed in a specific cell at a certain time point. A microarray is formed by a set of DNA spots, a very high number of spots, that are attached to a solid surface. These spots contain specific nucleic sequences, called oligos or probes, that are specifically designed to hybridise with their complementary sequence like, for instance, a specific series of nucleotides in a gene. Oligos that are hybridised, under very stringent conditions, to their target sequence can be detected through specific, labelled targets that emit luminescence, i.e. a dye, if stimulated. These kind of arrays allow the detection of thousands of DNA sequences. Indeed the data we filtered and normalised allows the analysis of fifteen thousands genes. A network composed by so many nodes is very hard to handle and, also, it is important to note that that the multiplication step will generate a bigger network since each node will be multiplied by its expression. To avoid creating networks that could generate a very high amount of data, we decided to reduced the number of genes under investigation by considering only those genes that shown a differential expression between the G1 and the G2 samples. Namely, genes showing a p-value lower than 0.0025 were selected. The p-value was computed by using a T-test. For genes appearing more than once in the microarray, the expression was averaged before computing the T-test. In this dataset, only one average network was used to describe both classes, because no gene had a value of zero in any of the subjects. The only differences, in this case reflect the different biological states.

Both networks, i.e. G1 and G2, are formed by these nodes:

- ABHD14A, ABLIM1, APEX1, APOE, ARHGAP6, ARHGEF7, ASAP1, BAD, BBS7, BRMS1, BUB1, BZW2, CBR1, CCNB1, CCNB2, CCND3, CCNO, CD99, CDC45, CDK1, CDKN3, CELSR3, CENPE, CEP72, CEP76, CHCHD3, CKS2, CLDN1, CRYBA2, DAZAP2, DDX23, DLGAP3, DRG1, DSN1, DST, DZIP3, ECT2, EIF2AK1, EIF2B2, EIF4E3, EIF5B, ELL, EML2, EPB41L3, ERG, FADD, FAM46C, FBXO7, FGD1, FGF13, FKBP5, GALK1, GBF1, GEMIN6, GLTSCR2, GRPEL1, GTF2A2, GTF2H1, H2AFZ, HMGB2, HSF4, HSPA1L, ICT1, IDO1, IFIT3, IGF2BP1, IL6R, ING1, KCNMA1, KLF5, KLHL20, KRT18, KRT8, LDLRAP1, LEF1, LIG4, LPIN2, MAP2K1, MED19, MEIS2, MELK, MITF, MPP6, MS4A2, NCAPG, NCAPG2, NDC80, NDUFAF3, NDUFB10, NDUFB8, NEK2, NEK6, NPDC1, NUP160, NUP85, NUSAP1, NXT2, OTUB1, PAX6, PBK, PEA15, PIAS1, PIK3R3, PIM2, PKMYT1, PLAG1, POLD3, POLD4, POLR2H, PPP1CA, PRC1, PRDX1, PRKCD, PSEN1, PTK2, PTPN6, PTPRH, RHOBTB3, RHPN2, RNF126, RPL10, RPL15, RPL26L1, RPL3, RPS20, SDC2, SESN1, SHMT2, SLBP, SLC1A5, SRBD1, SRPK1, SSR1, SSX2IP, STAT4, STMN1, STMN2, STRBP, SUPV3L1, TERF2IP, TFF1, THOP1, TMOD1, TOP2A, TRIP13, UBE2C, UBE4B, USP46, WDHD1, ZNF143, ZNF462.

3.2 Constructing a network

The network construction phase is a very important step in a network-based analysis. The construction depends from several parameters that are the biological data that are available, the process we want to model, the goal of the study and the results we want to obtain. The limitations, in these kinds of analysis usually depend on the data. The lack of information, the high dimension of the datasets, the fact that data combines numerical and biological information are different aspects that should be considered while building a network. As said, for instance, in the PET dataset we were dealing with a gene expression dataset that contains thousands of nodes. Constructing a network consisting of thousands nodes result in a complex model that is difficult to analyse by using our multiplicative approach. On the other hand, a network which contains fifteen thousands nodes could be easily analysed by using a classical topological analysis. Indeed, extracting several centrality indexes, clustering coefficients and other parameters is very easy. But, the simple addition of some biological information that imply to cross-analyse numerical values together with the biological function of a specific node become a complex task, since it should be done thousands times. In this sense filtering and normalising the raw data becomes a necessary step in order to deal with the biological complexity even though these operations may lead to a, measured, loss of information of the modelled biological process. Again, when we were dealing with the PET dataset we eventually ended up analysing 151 of the thousands of genes the dataset was composed of. This fact leads to an enormous loss of information since we are not considering the vast majority of the data but this was the most meaningful way to analyse the dataset.

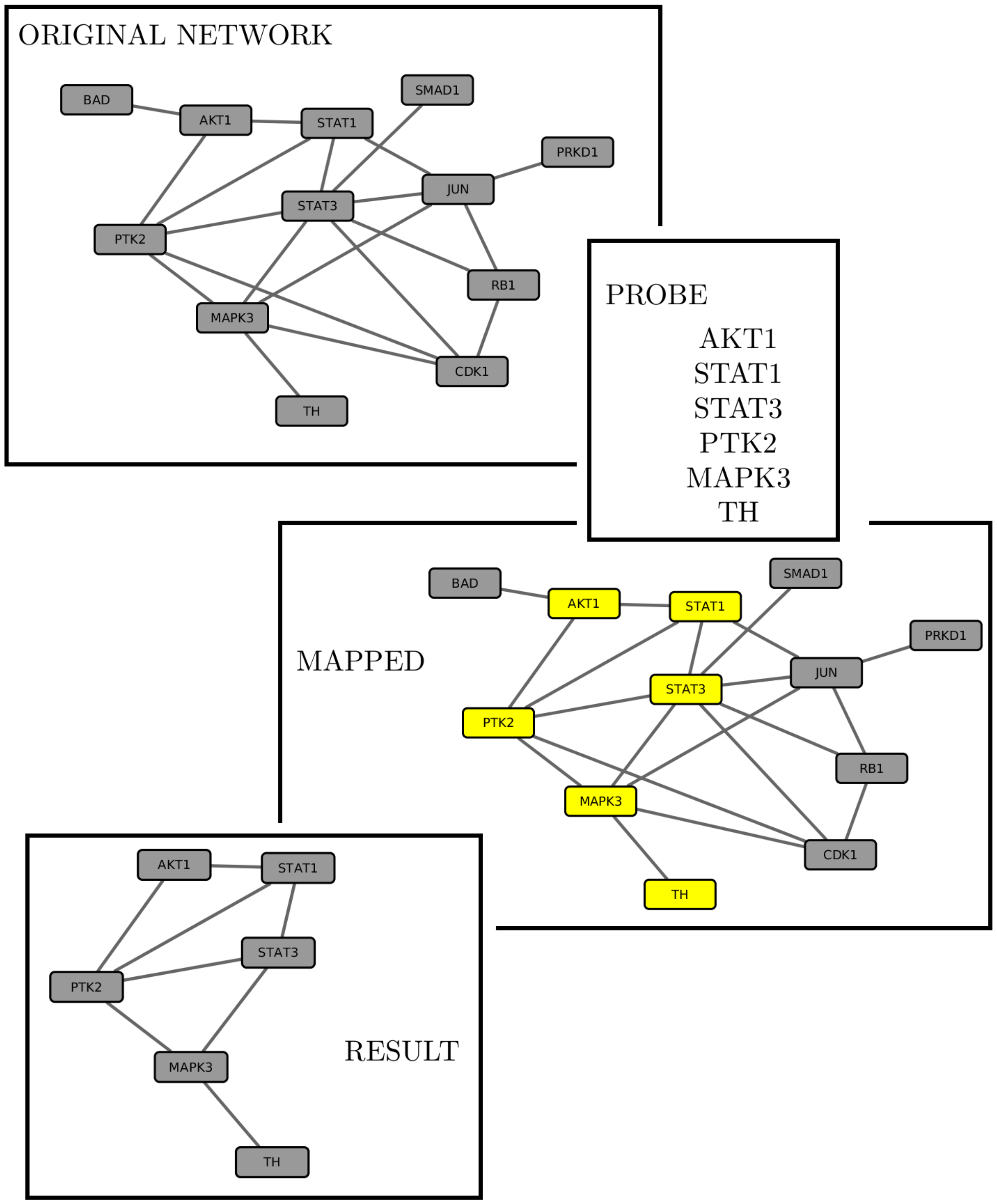


Figure 3.1: **The construction of a network**, consists of several steps. Starting from an existing network by using a probe it is possible to map the proteins in the starting network (nodes in yellow). Once the mapping is done, by extracting the proteins and their interactions a network is obtained.

Two possible approaches are available and allow constructing a biological network. The first methodology is based on the modelling of a specific process which may, or may not, depend on the raw data. Modelling a well known process like, for instance, the MAPK pathway, requires a fixed and very specific set of nodes, representing molecular actors, that were found to be interesting in that specific biological context. On the other hand it is possible to model a more personalised experimental dataset by using a *probe*. A probe is a set of nodes that are present in the dataset and that are used to construct a network. This is done by extracting a list of nodes, i.e. the probe, and then, by using a bigger network, e.g. usually an interactome, it is possible to map these nodes of interest and extract the subnetworks they form by exploiting the existing interactions in the bigger network. In both cases it is possible to obtain a set of nodes and edges that describe a biological process.

The next important step refers to the newly created subnetwork. In this sense it is important to verify if the network is *connected*. This means verifying that each node is able to interact, directly of through other nodes, with all the other nodes in the network. Only connected networks give meaningful results when analysed. Non connected nodes are useless and should not be included when performing a topological analysis. When studying pathological processes, it may happen that some nodes are disconnected. This lack of connections may be due to the fact that a specific pathway is affected by some pathological mechanisms and, through a loss-of-function event, some nodes result to be disconnected. It is very important to consider this aspect but, from a topological point of view, it is not possible to include such information in the analysis since centralities are computed only for those nodes that belong, i.e. are connected, to a network. A disconnected component, i.e. a subnetwork, that is the result of a missing node that worked as a bridge between two parts of network is different. If we have more than two nodes that communicate, then it is possible to extract their topological properties.

Also, it is important to note that dealing with disconnected nodes is possible. In this sense, disconnected nodes should be connected to the other nodes and this operation could be done in different ways. Again, in order to perform this operation a bigger network is required, in order to look for interactions that allow the actual connection. Connecting nodes can be done in different ways. It is possible to look for paths, shortest in our case, between the node which is disconnected and at least a node of the connected network. Once the connection is found new nodes will be added to the network and become the bridge that permits to obtain a single, connected network. Adding new nodes is trivial when we are modelling an experimental context. It is possible that the newly added nodes do not belong to the set of nodes that were analysed, e.g. by using a microrray, hence they should not be included in the final network. On the other hand, if we are modelling a process which is not representing a very specific context, e.g. a well known pathway, the addition of new nodes is usually the starting point for new, interesting results. The second possible approach consists in the analysis of the neighbourhood of the network. Starting from a subnetwork, mapped in a bigger network, and extracting the neighbours of the nodes of interest could be enough in order to obtain a fully connected network.

Data normalisation and multiplied networks construction

The multiplied network construction followed a standardised procedure consisting of some, clear steps that we defined as part of the framework. Each multiplied network starts with a plain network, as we said in the paragraph above. While plain networks do not require further information, a multiplied network, as said, requires numerical values that are associated to nodes. These numerical values, belonging to different datasets, required distinct normalisation approaches, reflecting the different technologies that originated the data. A normalisation step is necessary since each technology returns, as output, different quantifications which we use to create the potentiated network. These numerical values have different scales and they should be normalised in order to fit a network-based modelling. The issues we addressed were two. The first problem arises when the quantified data have very high values like, for instance, in a range like $[0 - 50000]$. Clearly, it is not possible to assign to a network of dozens nodes an attribute whose values are, for instance, 35392 for the Node1, 12394 for the Node 2 and so on. If we let this happen then the resulting network will have a number of nodes and edges that makes the network impossible to analyse in reasonable time. In this sense it is important to normalise the values in a range which remains significant, but, and here it comes the second issue we addressed, considering the fact that the lowest value should be one. It must be one since a protein, for being represented in a network, must have at least a copy. Generally speaking we assumed that the minimum value the multiplying attribute may have is 1 and the maximum value should be in the order of one hundred, in the worst case. This decision is strictly related to the number of nodes our networks have. On the other hand, a dataset may contain very low values that are, for instance, very small fractions of one like $1 * 10^{-4}$ or similar values, depending on the technology and its final, numerical output. In these cases the same criteria for the normalisation should be considered in order to obtain a meaningful multiplication attribute since the lowest value, again, should be one.

Cytoscape, CentiScaPe and PeSca

The networks we have analysed were created and managed by using a specific software which is Cytoscape [92]. Cytoscape *is an open source software platform for visualizing molecular interaction networks and biological pathways and integrating these networks with annotations, gene expression profiles and other state data.*[2]. Its importance is rapidly growing and it is considered as a general platform for complex network analysis in different fields. It exploit the possibilities offered by a high number of standard file formats that are SIF, GML, XGMML, BioPAX, PSI-MI, SBML, OBO, and Gene Association files. Importing and exporting data from Excel, in CSV format, is necessary when loading attributes files and saving the results from the network analysis. Cytoscape also works as a web service client allowing to connect it to public databases like Pathway Commons, NCBI Entrez Gene and a few others in order to download already compiled networks

[2] `http://www.cytoscape.org/what_is_cytoscape.html`

and annotation datasets. It is very useful when it comes to draw nice graphs in the sense that it allows creating personalised views and exporting them in different, common file formats. It is written in Java by using specific API, and is composed by an internal set of applications which allow common network analysis tasks but it also take a very great advantage of an app-based nature. This fact allows an extensive development of very useful and different tools that can be downloaded and shared. By using this software it was possible to construct the networks and to take advantage of some, well known tools that allowed us in order to extract fully connected network and then to compute the centralities for the multiplied networks.

CentiScaPe [93] is a Cytoscape App that allows computing centralities in directed and undirected networks. It implements a version of the all pair shortest paths algorithm based on the Djikstra algorithm and permits to compute and visualise meaningful plots about the centrality values the nodes score. The new version, not yet released, allows computing centralities in multiplied networks.

PeSca [75] is a Cytoscape App that allows computing shortest paths between set of specific nodes. There are several options that allow for searching different combinations of set of nodes. For instance, it is possible to connect a single node to a set of nodes, or to connect a set versus of nodes to another set of nodes. Also it allows extracting the tree of paths that are originated from a specific source. Its main application is related to connecting isolated nodes in order to obtain connected networks and we used it extensively.

Both these apps were exploited for constructing and analysing the networks we presented in this thesis. So we decided to implement the main and most useful functions we used, for the R library IGraph. Our implementation allows R users to perform the same operations we described, in a personalised way that do not require to use Cytoscape. This, in our opinion is very important since performing several operations by using Cytoscape is very time consuming and creating a pipeline in R is easier and results in faster and more reproducible experiments. However not all the functions are yet implemented.

Topological centralities

Topological centralities are mathematical tools that were developed and are used to compute nodes properties describing the role each node plays in a network. These tools take advantage from the definition of paths and distances in order to rank the nodes and highlight the more central and the more periferic. Distance-based centralities comprise Eccentricity, Radiality, Closeness, Centroid and Eigenvector. Path-based centralities comprise Stress, Betweenness, Bridging. Also there is another centrality, i.e. the Degree, that considers the number of neighbours a node has.

Degree

According to the mathematical definition of Degree, this index measures the number of incident edges a node has. Loops counts as two edges. The sum of all this edges represents the Degree of the node. In the multiplied network model the Degree considers, as neighbours, also the new copy a protein has. The other approach for computing the Degree in the multiplied network model consider only the actual neighbours of a protein without taking into account the new copies. It is not yet clear which approach is better and, in this sense, the use of this centrality may be misleading. We included the version that considers the copy of a node as part of its neighbourhood, in order to add this information, i.e. the numbers of copies a node has, available from a topological perspective.

Eccentricity

The Eccentricity centrality [94] is a distance-based metric that describes the neighbourhood of a node considering all the nodes in the network. This means that, by definition, the Eccentricity is represented by the reciprocal of the maximum distance between the node of interest and its most distant neighbour.

The mathematical definition:

$$C_{ecc}(v) := \frac{1}{max\{dist(v,w) \, : \, w \in V\}}$$

This topological index gives an idea of the maximum path length a node develops and, consequentially, how close it is to its farthest neighbour. An Eccentricity which is equal to one means that the node and its neighbours are separated by only one edge and indicates a very central node. The smaller the Eccenticity the more periferic the node is.

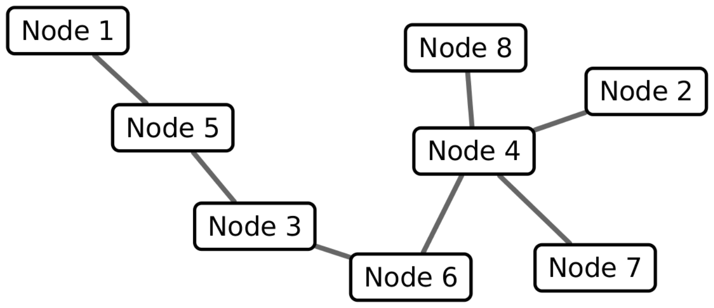


Figure 3.2: In this network we have the Node 1, 2, 7 and 8 that scored the lowest value of Eccentricity since they are very periferal node and require five steps to reach each other, in the worst case. Node 4 and 6 have the best Closeness, Radiality and Centroid values. These results highlight the central role these nodes play in the network. The same applies for the Eigenvector which is higher in Node 4 and 6 and give to these node a leading role in controlling the network.

Closeness

The Closeness centrality [95] is another distance-based index which aims at determining how long are the paths between a node and all the other nodes in the network. A high Closeness highlight a node that minimises all the distances between itself and all the other nodes in the network.

The mathematical definition:

$$C_{clo}(v) := \frac{1}{\sum_{w \in V} dist(v, w)}$$

This index sums up the distances between a node and all the other nodes in a network and gives a more precise idea, if compared to the Eccentricity, about the node and how it interacts with the other nodes. Ideally, the most central node has a distance of one with respect to all the other nodes, making its Closeness equal to the reciprocal of the number of nodes in the network. The lower the Closeness the less central the node is.

Radiality

The Radiality [96] centrality, like Closeness, is distance-based and allows highlighting central and periferal nodes. This means that the closer the node is to other nodes the better it is integrated into the graph. Like Closeness, high values of Radiality suggest that the node can easily reach other nodes.

The mathematical definition:

$$C_{rad}(v) := \frac{\sum (\Delta_G + 1 - dist(v, w))}{n - 1}$$

Where n is the number of nodes in the network and Δ_G that is the diameter of the graph that is the longest shortest path found in the network.

Centroid

Centroid centrality is distance-based and describes tha neighbourhood of a node in terms of vicinity. This means that, the more a node has nearest neighbours, if compared to the other nodes in the network, the higher its Centroid will be. This centrality is computed for each couple of nodes in the network since it requires to compare the neighbourhood of each node with the neighbourhood of each one of the other nodes. The obtained index is somehow weighted with respect to all the other nodes in the network.

The mathematical definition:

$$C_{cen}(v) := min\{f(v, w) : w \in V\{v\}\}$$

Where $f(v, w) = \gamma_v(w) - \gamma_w(v)$ with $\gamma_v(w)$ the number of neighbours that are nearer to v rather than to w. The index is computed by calculating $\gamma_v(w)$ and $\gamma_w(v)$ for each node v and then by extracting the minimum.

Eigenvector

The Eigenvector centrality measures the influence a node has in a network. The centrality assigns relative scores to each node in the network by assuming that being connected to a high-scoring node has a major contribution to the node centrality, if compared to a low-scoring node.

The mathematical definition takes advantage of the adjacency matrix $A_{v,t}$. The relative centrality value for the a node v is denoted by:

$$e_v := \frac{1}{\lambda} \sum_{t \in M_{(v)}}$$

$$e_t := \frac{1}{\lambda} \sum_{t \in G} a_{v,t} e_t$$

where $M_{(v)}$ is the set of the neighbours of v and λ is a constant.

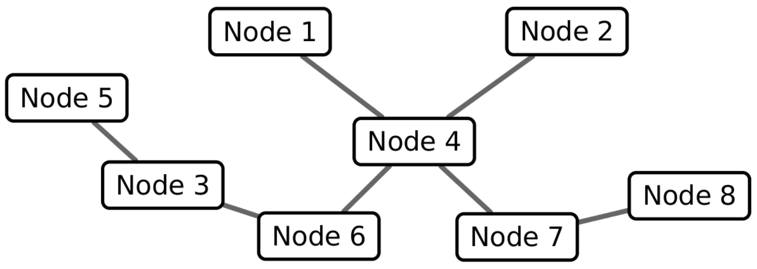


Figure 3.3: In this network the Node 4 has the highest value of Betweenness and Stress, if compared to the other nodes. In terms of Bridging the node that scored higher is the Node 6 while the Node 4 has a very low Bridging score.

Stress

Stress centrality[97] is based on shortest paths. Eccentricity, Radiality and Closeness are based on distances between node hence they could be considered as partially overlapping. Stress is very similar to the Betweenness centrality, that is shortest path based too. The Stress centrality gives us information about the number of shortest paths passing through a node and tells us how much work a vertex has to do in a graph.

The mathematical definition:

$$C_{str}(v) := \sum_{s \neq v \in V} \sum_{t \neq v \in V} \delta_{st}(v)$$

The value $\delta_{st}(v)$ is the number of paths that pass through a node starting from node s and ending in node t. Shortest paths starting and ending in v are excluded because Stress measures the paths that pass through a node. This assumption is also used for Betweenness centrality.

Betweenness

Betweenness[98] like Stress is based on the number of shortest paths. This centrality is computed by counting the number of shortest paths starting from the node v_1 and ending in another node v_2 that pass through a third node which is the one we are interested into, i.e. n. The computation of the Betweenness is done by couples of nodes since, for each node in the network, two of the remaining nodes are chosen at each iteration and then the number of paths between them is counted. Then the value is compared to the number of paths that pass through a certain node, and then to another node for all the nodes in the network obtaining partial Betweenness. Then another couple is chosen and the computation restarts. In the end we have to sum all the partial Betweenness for each node.

The mathematical definition:

$$C_{bet}(v) := \sum_{s \neq v \in V} \sum_{t \neq v \in V} \delta_{st}(v)$$

where:

$$\delta_{st}(v) := \frac{\sigma_{st}(v)}{\sigma_{st}}$$

if s reaches t. Else $\delta_{st}(v) = 0$.

Bridging

The Bridging centrality [99] is another path based centrality that aims at finding those nodes that could be considered as bridges. A bridge is commonly intended as a structure that connects two parts that alternatively are not in direct connection. In this sense we can say that a bridge node is a node that connects two region of a network, i.e. modules, that are considered *dense* in terms of number of edges and nodes. Hence a bridge node helps in enhancing the communication between different parts of the network. This index is a sort of weighted Betweenness.

Indeed, the mathematical definition takes advantage from the Betweeness definition. The Bridging is computed by:

$$C_{bri}(v) := BC(v)xC_b(v)$$

where $BC(v)$ is the Bridging coefficient that is computed by:

$$BC(v) := \frac{d(v)^{-1}}{\sum_{i \in N(V)} \frac{1}{d(i)}}$$

Biological meanings of centrality indexes

Centrality indexes were initially defined in the context of the social networks analysis. With the new applications that take advantage of graph theory in different field like, for instance, biology, it has emerged the necessity to contextualise these mathematical metrics to different fields. In Systems Biology we are dealing with biological properties of the nodes so, for instance, a node with a high Degree could represent a protein that has different biological properties that interacts with a very wide set of other molecules to carry out very different duties. Each centrality should be interpreted in the proper context, taking into account all the available information. Also, different metrics should be used together in order to obtain a comprehensive view about the real role a node plays. In this sense our work aims at obtaining the wider and comprehensive mixture of information in order to exploit all the topological centralities and the biological information together.

Centrality measures like Closeness, Radiality, Centroid and Eccentriticy are based on the distance between nodes. These kind of centralities give us a snapshot that allows to comprehend how a node is interacting with the other nodes in the network. These measures are able to depict how much a node is central, i.e. which distances separate a node from the other nodes in the network. The more central a node, i.e. the lower the distances, the more central it is. Higher distances may indicate that the node has, potentially, a marginal role since it reaches the other nodes through longer short paths.

On the other hand, centralities like Stress, Betweenness and Bridging are based on the number of short paths that passes through a node. These kind of measures give us a different kind of information, with respect to the distance-based centralities. Nodes that have a high value of the cited centralities are nodes which functions as passageways. This means that a lot of communications pass through these nodes and their absence may result in the network disruption and a possible loss of communication. In particular the Bridging, as the name suggests, indicate nodes that function as bottlenecks while Stress and Betweenness indicate node that are subject to a high number of short paths passing through them. In this case, nodes with high values of these centralities are considered very important nodes, in terms of topological functionalities, while nodes with low values of centrality are considered marginal nodes.

Giving a biological meaning to a centrality is not straightforward, since they are mathematical properites that describe specific topological characteristics of a node. For instance, a node with a high Betweenness is considered as a central node since the information flow that is processed by the network passes through it. But, from a biological point of view, what does it means to be a high Betweeness node is strongly related to the biology of the network. The same applies for the Stress and the Bridging centralities. Both these centralities, when the score is high, represent nodes that have a high ratio of shortest paths passing through them. The fact that a node is traversed by a lot of paths may be important since its absence may affect the whole network and may be the cause of a disease. These consideration should be done through a proper analysis of the biological process that the network aims at modelling. The same applies for Closeness, Radiality and Eccentricity. High Closeness nodes are nodes that minimises the distances between all the other nodes in the network. From a biological point of view this means that they could be regulators for the network since they are near to a lot of other nodes. On the other hand, a node with a very low Closeness is considered as a periferic node. What happens, from a topological point of view, when a node represents a receptor that, once activated, causes a biochemical cascade of events that are necessary to the correct functioning of a cell? It happens that the Closeness of this node will be lower, or very low, if compared to a more central but less interesting node. This example clarify how, giving a biological meaning to a centrality is possible and should be done, but this meaning should be considered very carefully. Knowing the process under investigation is fundamental to give the meaning to a numerical value.

3.3 Finding patterns

The four dataset we analysed comprise dozens of subjects and each network consists of dozens of nodes and hundreds of edges. The substantial complexity of such datasets does not allow to analyse the data by using a manually curated approach. Moreover, our idea considered the analysis of several centralities which define the role a node play in the network. To achieve this goal, computing nine different centralities for each node and for each subject, globally we have 183 subjects, means obtaining some high dimensional tables of double values. Such wealth of information become tractable only by using a computational-based approach. Moreover, we wanted to exploit the information contents of all the centralities as an ensemble, aiming at uncover emergent patterns that could be shared along the different classes of subjects. In practical terms this operation aims at finding those similarities, if there are some, that define the class of the BCLL patients and the class of the BCLL controls. Eventually, these intra-class similarities also allow to discriminate the two classes since they become inter-class differences. This operation, which concerns the retrieval of occurring patterns in complex dataset, in computer science, belongs to a field which is defined as *machine learning*. Currently there are some powerful techniques that allow training different algorithms in order to look for hidden patterns.

Machine learning aims at addressing the problem of teaching to an algorithm which characteristics make an object belonging to that specific class and then trying to classify some new samples by assigning them a label and verifying how many times the algorithm assigned the correct class. This task is called classification. Teaching to an algorithm to decide which is the class of a new sample is very useful since a computer can exploit a very high amount of information which may come from very different sources. In our case what we wanted to teach to the algorithm is what does it means to be a healthy and an unhealthy subject in different, pathological contexts. To address this problem we needed to tell to the computer which characteristics, called *features* make a patient a patient. The features we decided to use for describing the subjects are the centralities computed over the multiplied networks. Our idea was related to the fact that, if centralities describe the role a node play in a network, then a set of centralities should describe specific characteristics of that network. Then, having a set of network belonging to the same class would result in a set of features that describes the characteristics of the whole class. Hence, by using a classification approach we should be able to find similarities between healthy subjects and differences between healthy and unhealthy subjects. For instance, the Betweenness computed over a network from the PET datasets belonging to the G1 class is clearly a characteristics which may help in identifying such class.

Feature Selection

Learning algorithms require tables of data in order to learn which object belong to which class. These data are called features and are used to describe peculiar characteristics of the objects we are analysing. For instance glasses could be considered objects that have several features like diameter, height, material, presence or absence of an handle, colour, thickness, and weight. By measuring these characteristics we are describing the features of a common glass, a mug, a shot glass, a cocktail glass etc.... By describing a glass in terms of its features we are assigning a specific set of measures to a specific kind of glass. The idea here is that we are able to say that a set of measures refers to a mug instead of a cocktail glass. The same principle applies to a lot of different objects and, in this work, we are using topological centralities as measures to describe a class of networks and to discriminate this class from another, different class. Each object can have a lot of different features and, in our experiments, each protein in each network, has nine different centrality measures that describe its topological role. Moreover, a network has a set of features, that are all the features of all its proteins that defines its nature, i.e. the class. Nine features for each protein means, for instance, that in a network with one hundred proteins, the total number of features describing that network are nine hundred.

Features are modelled in a multidimensional space and, to reduce its dimensionality and to avoid redundant features which may negatively affect the learning process, several feature selection algorithms [100] were developed. These algorithms, by reducing the number of features that are used by the model allow reducing computational times and address the problem of the over fitting. Over fitting happens when a model is not able to generalise since it has too many parameters that describe the features and it loses its ability in classifying. To perform the feature selection we used the Infinite Feature Selection (Inf-FS) [101]. The Inf-FS is a graph-based method that exploits the convergence properties of the power series of matrices to evaluate the importance of a feature with respect to all the other ones taken together. Indeed, in the Inf-FS formulation, each feature is mapped on an affinity graph, where nodes represent features and weighted edges relationships between them. In particular, the graph is weighted according to a function which takes into account both correlations and standard deviations between feature distributions. Each path of a certain length l over the graph is seen as a possible selection of features. Therefore, varying these paths and letting them tend to an infinite number permits the investigation of the importance of each possible subset of features. The Inf-FS assigns a final score to each feature of the initial set; where the score is related to how much the given feature is a good candidate regarding the classification task. Therefore, ranking in descendant order the outcome of the Inf-FS allows us to perform the subset feature selection throughout a model selection stage to determine the number of features to be selected. In the experiments of this work, the Inf-FS has also the advantage of providing insights about the features that drive the outcome of the classification and, indirectly, about the features that may become relevant biomarker.

The features that were used for the classification tasks are topological indexes commonly used in biological network analysis to highlight the role a node plays in a network. All the indexes in the AM, MI and BCLL datasets were computed by using Cytoscape 3.2.0, and a modified version of CentiScaPe 2.1 that allows computing centralities over multiplied networks and that allows multiple network analysis at a time. The multiplied networks in the PET dataset comprehend a high number of nodes and this fact required that a first multiplication step was performed with the same softwares to obtain the potentiated networks. Eventually, the topological analysis was performed by using the online version of CentiScaPe which allows computing centralities in big networks.

Each centrality describes a different topological characteristic; by using its values it is possible to rank the nodes according to their topological relevance. In this work, nonetheless, since the network is personalised for each subject, the multiplication algorithm enables a very sharp focus on the real role of a node in each specific network. The main drawback concerns the fact that, instead of a single network, there is now a network for each subject and a wealth of data are generated, which is not easily tractable. In this scenario, the extremely high dimensions of the datasets limit data analysis and it is not possible to find patterns, similarities, or general insights by using manually curated techniques. The only methodology that allows the investigation of this amount of information is related to computer science; in particular, we have chosen a pattern recognition based approach in order to exploit the information obtained.

Inf-FS method

Given a set $F = \{f^{(1)}, \dots, f^{(n)}\}$ of features that represents distributions and a sample of the generic distribution f, it is possible to construct an undirected and fully-connected graph where the nodes represents each distribution and the edge represent a pairwise relation between two distributions. The weight that are associated to the edges are called energies and are represented as a linear combination of two measures depending on the correlation and the standard deviation weighted by a parameter, i.e. α, $a_{ij} = \alpha\sigma_{ij} + (1-\alpha)c_{ij}$. The correlation and the standard deviation are thus used to compute the energy of each edge in order to consider both the feature dispersion of each distribution and their correlation. In order to expand this idea to set of more than two distribution it is necessary to introduce the definition of *path* between two features. In this graph a path represents a subset of the available features, i.e. the nodes in the graph and is defined as $\gamma = \{v_0 = i, v_1, \dots, v_{l-1}, v_l = j\}$. So now the energy can be considered as the product of all the energies that forms the path, i.e. all its edges. It is now possible to compute the single feature energy score for a feature f(i), at a specific path length.

Once all the energies for all the features are computed it is possible to rank these values to obtain a subset of relevant features. But to compute all these energies the complexity is equal to $\mathcal{O}(n^4)$ hence the computation results not feasible for large set of features. To overcome this limitation, the Inf-FS approach assumes that the length of the paths tends to infinite in order to compute a new feature score for the features:

$$s(i) = \sum_{l=1}^{\infty} s_l(i) = [(\sum_{l=1}^{\infty} A^l)\mathbf{e}]_i = [S\mathbf{e}]_i$$

Then, by using the geometric series of the adjacency matrix is it possible to compute these scores while ensuring that this sum converges. By using the convergence property of the geometric power series of a matrix it is possible to obtain the matrix that contains the information about the energy of the set of features. By marginalising the energy scores it is now possible to obtain the score for each feature and order these values in order to obtain a ranked set of features.

Support Vector Machines

There are several, different algorithms that allow to implement automated pattern recognition methodologies. These algorithms belong to the class of machine learning algorithms and are designed to find recurring patterns in data. To achieve this goal it is necessary to train the algorithm in order to teach what does it means, for instance, being a dog instead of being a cat. To do so a set of observations, i.e. data describing a specific object, is required. This set is said *training set*. Pattern recognition algorithms can be divided in two major classes depending on the training set. Supervised learning in the case that the label of each object is known, i.e. we have a set of object that are dogs and cats and each one has its label ”dog” or ”cat” associated, and Non-Supervised when the label remains unknown. For both classes there are a number of algorithms which allow finding recurring patterns. Finding a pattern means to search the data and see if the algorithm is able to classify a sample that does not belong to the training set. Testing sets are used to verify the performances of the algorithm.

The data that such algorithms consider are numerical vectors, grouped in tables, that describe specific characteristics of the object we want to investigate. Lets assume we want to separate male from female by using a supervised pattern recognition approach. We need some characteristics that are shared along the two classes. For instance we can investigate the height and the weight and see if males share some recurring values that are different from the females group. Height and weight are called *features* and are used to create the training test which is composed, lets say, by 50 males and 50 females. We have a table which contains three columns that are, respectively, the label, the height and the weight. The label could be a zero for male and a one for female. Now the algorithm tries to retrieve some recurring patterns related to the label zero and to the label one, that hopefully will be useful to separate the two classes. Once the training is done we could submit to the algorithm a new pair of values $(height, weight)$, whose label is known to us, and see if it is able to give the correct class. This is a common classification task, that consider a two classes problem.

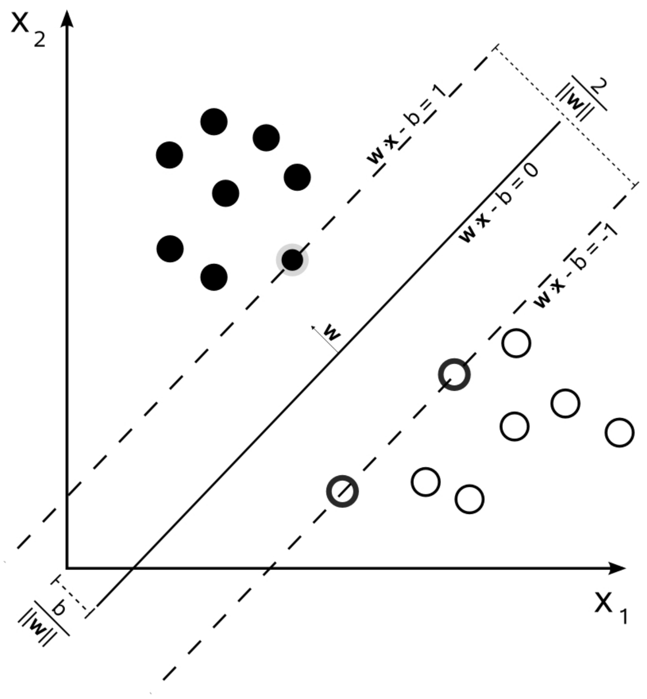


Figure 3.4: **The SVM approach**. The solid black line represent the separating plane which creates two subspaces describing the two different classes. The dashed lines represent the distance between the plane and the nearest observation. This is the distance that should be maximised by the algorithm while trying to separate the classes.

We applied a classification algorithm to the classes Healthy and Unhealthy, since we are investigating the properties of disease networks and our datasets could be divided in unhealthy patients and healthy controls. In particular, we used a supervised approach, since we know who is a patient and who is a healthy control, based on the SVM algorithm. SVM, like other linear classifier, work into a multidimensional space (see Fig. 3.4[3]). The goal of the linear SVM is to find an hyperplane which separates the two classes of data. Ideally the distance between the hyperplane and the nearest point of each class should be maximised.

More formally we can define a dataset as a set of points $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$ where x is a vector of features and y is the label that is associated to that observation. In our case each x_n contains nine centralities and y could be healthy or unhealthy, respectively 1 and -1. The SVM aims at finding the best hyperplane which is represented by this function: $\vec{w} \cdot \vec{x} + b = 0$. In the linear classification we could search for soft and hard margins. An hard margin is used whenever the classes are linearly separable, the soft margin is used in the case that the data are

[3]`https://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`

not linearly separable. A soft margin could behave like the hard margin in case that the data are linearly separable.

Support Vector Machines (SVMs) is a very well known and widely used classification model. Is is a theoretically well-motivated algorithm that allows empirically good performance in many fields of application ranging from computer vision to biology. For instance, the SVM algorithm has been applied in the biological field for several medical applications [102]. More formally, an SVM constructs a hyperplane in a high-dimensional features space that is used to separate observations belonging to a category from the others [103]. In this work, the hyperplane, created by using the topological features, is used to discriminate between healthy and unhealthy subjects.

We decided to use the linear SVM since all our datasets, which describes four different pathological conditions, consist of two classes that represents healthy and unhealthy subjects. Actually, the AM dataset comprehend two different kinds of unhealthy subjects but we decided to consider them as belonging to the "unhealthy" class. The same happens in the MI dataset where there are three different subclasses of unhealthy subjects. The PET dataset is composed by two classes and, finally, the BCLL dataset is also composed by two classes.

3.4 Results

Node multiplication algorithm

One of the novelties of this work concerns the ability to weigh the networks by multiplying the nodes, thus obtaining a set of potentiated networks that better represent experimental conditions. By assigning a numerical value to each node in a network, using the multiplication algorithm, it becomes possible to reshape the whole network structure by augmenting the copies of each node. By doing so, it is now possible to personalise a network fitting it to very specific and individual experimental conditions enabling the investigation of the properties of such potentiated topologies. This algorithm allows the creation of experimental networks and permits to define a new set of analysis that aim at investigating the properties of this new model.

Our approach is based on the idea that, in the context of the PPI networks analysis, different, concurrent components exist. The firts one is static and invariant and is represented by the network topological architecture which strictly depends on the genetic background. The second component is variable and comprehends the biochemical status, i.e. the activated or inhibited state, of the proteins that participate in the network topology. The level of biochemical activity, which may represent the phosphorylation level of a protein, is currently considered a further layer of information and is not considered as part of a topological analysis. But this is a great limitation since in biology, the more a protein is activated the more it is involved, in terms of number of copies actually present, in the regulation of a biological process. It is important to note that the more a protein is present in a specific process the higher it is involved hence, representing these numbers in terms of topological characteristics aims at giving a more precise mathematical description of the actual role that molecule plays. Finally, the number of a specific molecule is related to the genetic background, the environmental conditions and the biochemical status so, a topology that is able to represent the individual molecular fingerprint becomes necessary in the context of the personalised medicine. To reach the goal we developed a new model that is the multiplied model that aims at creating the potentiated topologies representing a single individual.

To obtain potentiated topologies it is necessary to add new nodes and edges to an existing network. More specifically a new node is added depending on the experimental values it has and is connected to its original source by an edge. Moreover each copy is connected to all the neighbours of the original source and to all its copies. The result will be a clique that comprehend a number of nodes that are copies of an already existing node, that has the same neigbours of the original node. This also means that each copy has the same Degree of the original source.

After the multiplication step a potentiated network is obtained and it could be investigated by using specific algorithms. In this work we focused on topological centralities and it required to develop a specific method that allows computing a centrality for a node when this node is represented by more copies. The idea here is that each original node and its copies share the same value of centrality, for each centrality, and the sum of all the contributions is used to define the topological role of a specific node. By doing so it is now possible to compute all the centralities for all the multiplied nodes by summing up the contributions of each copy computing the role each node has in the potentiated network.

Computational analysis

The multiplication algorithm was applied to all the biological datasets and permitted to construct around 180 personalised networks, in order to create individual personalised models. After the construction, by computing the topological centralities we were able to determine the topological role each node plays in each specific network. Finally, by using these values, we were able to infer hidden patterns along the different classes of networks by applying some machine learning techniques. The centralities we computed are nine: Degree, Eccentricity, Closeness, Radiality, Centroid, Stress, Betweenness, Bridging and Eigenvector [104]. The last step in our workflow, concerns the classification of the network by means of an algorithm that is commonly used to search for shared patterns. The first part addressed the problem of finding the best set of features for the classification. The second step refers to the actual classification of each network, i.e. finding the common patterns.

After computing the graph centralities, a feature selection algorithm let us rank the centralities in order of discriminative power. This step was performed with the Inf-FS algorithm. Finally, a learning algorithm based on a linear Support Vector Machines (SVM) approach was applied to each set of selected features in order to investigate the presence of similarities and recurring patterns. Here we used a leave-one-out approach in order to generate training and testing set.

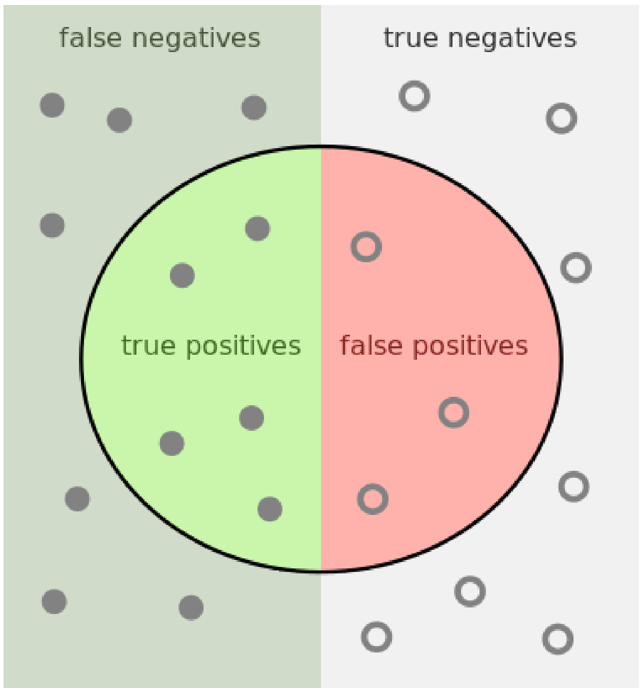


Figure 3.5: **Statistical parameters made easy**. This image allows an easy understanding of how Precision and Recall are computed. Precision is computed by $\frac{truepositive}{truepositive+falsepositive}$. Recall is computed by $\frac{truepositive}{truepositive+falsenegative}$.

The results refer to the ability of the algorithm in giving the correct label to a specific subject. They are computed with respect to the best Precision the algorithm was able to achieve during the classification step. Precision (see Fig. 3.5[4]) describes the number of actual healthy subjects that were classified as healthy versus the total number of subjects that were classified as healthy even though they belong to the unhealthy class, i.e. the ratio between true positive and all the positive. The higher the value, the more precise is the algorithm in finding which set of features belongs to the healthy class. As pointed out, for each dataset the best Precision was computed and used as a reference. For each best Precision some other parameters, i.e. Accuracy, Recall, and the Area Under the Curve (AUC), were computed as useful indexes for better describe the real performances of the classifier. Accuracy refers to the fraction of subjects that are classified as healthy over the actual number of healthy subjects. Recall, also known as Sensitivity, refers to the number of healthy subjects that are classified as healthy, over the number of healthy and unhealthy subjects that were correctly classified. Finally we computed the Receiver Operating Characteristic or ROC curve comparing the true positive rate to the false positive rate. Here we reported a widely used measurement that summarises ROC curves which is the Area Under the ROC Curve (AUC).

For each dataset, several feature selections were performed since the algorithm requires the exact tuning of a specific parameter, i.e. the α parameter, which directly influences the underlying model for the selection and ranking of the fea-

[4]https://commons.wikimedia.org/wiki/File:PrecisionRecall.svg

alpha	performance	AM	MI	BCLL	PET
0.0	Accuracy	1	1	0.9512195	0.7464789
	AUC	1	1	0.9290323	0.8147826
	PrecisionF1	1	1	1	0.65625
	RecallF1	1	1	0.9	0.84
0.2	Accuracy	0.7142857	1	0.9512195	0.8169014
	AUC	0.7992424	1	0.9032258	0.8765217
	PrecisionF1	0.5789474	1	1	0.76
	RecallF1	1	1	0.9	0.76
0.4	Accuracy	0.7428571	1	0.9268293	0.7887324
	AUC	0.7689394	1	0.8354839	0.8252174
	PrecisionF1	0.6153846	1	1	0.7727273
	RecallF1	0.7272727	1	0.7	0.68
0.6	Accuracy	0.7714286	0.9722222	0.9268293	0.7746479
	AUC	0.8219697	1	0.7870968	0.7930435
	PrecisionF1	0.6428571	1	1	0.75
	RecallF1	0.8181818	1	0.7	0.72
0.8	Accuracy	0.8	1	0.9512195	0.6901408
	AUC	0.8333333	1	0.9	0.7591304
	PrecisionF1	0.6666667	1	1	0.6296296
	RecallF1	0.9090909	1	0.8	0.68
1.0	Accuracy	0.8285714	1	0.9268293	0.7605634
	AUC	0.9015152	1	0.8935484	0.8095652
	PrecisionF1	0.75	0	1	0.8
	RecallF1	0.8181818	1	0.8	0.64

Table 3.1: **The classification results for all the alphas**

tures. This parameter was tested in between 0 and 1 with a step of 0.2 in order to obtain six different experiments for each dataset. For each α a classification was performed in order to uncover potential patterns (see Appendix A for all the results) as mentioned above, i.e. SVM with leave-one-out. In the results it is possible to appreciate how the Precision remains above 0.57, which means that the algorithm, even in the worst case, is able to find, at least, few recurring patterns 3.2. In particular, for the MI dataset, the classifier was able to label each subject with its actual class, i.e. it scored 100%, while for the other dataset the Accuracy remains generally high, i.e. some patterns appeared. Notably, it must be pointed out that the dimension of the datasets and the kind of data that are used are two very important parameters that could affect this kind of tasks. Balanced datasets, i.e. with a similar number of subjects in both classes, are more reliable but, due to the intrinsic complexity of experimental and clinical analysis they are difficult to obtain. Hence, in order to assess whether the features are good at generalising the classification task to new subjects, a leave-one-out cross validation was performed before computing the parameters that were used for classifying.

Also, we performed several other comparisons using topological features versus raw data features. Five different algorithms were exploited to perform several

Dataset	Accuracy	AUC	Precision	Recall
AM potentiated	0.714	0.799	0.579	1
AM raw data	0.629	0.284	0.324	1
MI potentiated	1	1	1	1
MI raw data	0.50	0.759	0.70	0.778
BCLL potentiated	0.951	0.903	1	0.90
BCLL raw data	0.805	0.932	0.667	1
PET potentiated	0.817	0.877	0.76	0.76
PET raw data	0.873	0.93	0.793	0.92

Table 3.2: **Numerical results that allow the interpretation of the learning step** Here, the numerical results represent the values for each parameter computed during the pattern recognition phase. The values we computed are Accuracy, Area Under Curve (AUC), Precision and Recall. They are common statistical measures that are associated to the classification task and allow interpreting the output of the learning algorithm and to verify if it is capable of classifying the samples. All values in the table were computed for $\alpha = 0.2$ and refer to the best Precision the algorithm scored in all the iterations.

Accuracy comparisons. These algorithms are Minimum least square linear classifier (fisher), Support Vector Machines (SVM), Logistic linear classifier (loglc), Breiman's Random Forest (randomforest) and K-nearest-neighbours (KNN). The results are listed in Appendix A. Furthermore, we computed the statistical significance of the differences between the performances of these classifiers (see Table 3.3 and Appendix A).

Feature Selection

As said, the feature selection is a crucial step since a poor set of features could result in a failure while classifying. Hence, some automated algorithms that are able to rank the most informative features and to discard redundant, not useful, information are needed. As said, the feature selection ranked the features for each dataset and the best one were used for the classification. In this work we refer to $\alpha = 0.2$. The results shows how some centralities occur more frequently than others while some others are under represented in all the different experiments. The histograms in Fig. 3.6 and Fig. 3.7 show how many times a centrality occurs.

Eigenvector, for instance, is a feature that is very often selected in AM, MI and BCLL datasets. Also Eccentricity has, usually, an high rank for these datasets. On the contrary, the PET data shows a different behaviour and the best centralities used for classifying are Stress, Bridging and Betweenness. In numerical terms we have that in the BCLL dataset Eigenvector and Eccentricity dominate the scene with two very prominent peaks, and the other centralities are used very rarely if compared to this two. The situation is smoother in both AM and MI datasets. In the AM dataset we have a small number of features, 18 in total, and Stress, Radiality and Eigenvector play the leading role but some other centralities follow. Here there are not very interesting peaks. In the MI dataset we have,

dataset	result	loglc	randomforest	SVM	KNN	fisher
AM	H_0	1	1	1	0	1
	p-value	1.69e-12	1.62e-19	1.01e-09	8.98e-02	6.62e-15
BCLL	H_0	1	1	1	0	1
	p-value	2.96e-26	6.17e-21	3.57e-26	1.58e-01	1.05e-16
MI	H_0	1	1	1	1	1
	p-value	6.33e-29	8.35e-12	3.49e-16	6.01e-27	2.86e-18
PET	H_0	1	1	1	1	1
	p-value	3.02e-02	1.94e-14	4.49e-22	7.17e-19	7.13e-20

Table 3.3: **Statistical significance of the classification results** are presented. In this table it is possible to appreciate the statistical significance of the difference between the errors the classified scored, while classifying the different features. We performed 50 repetitions over a 2-fold cross-validation for each set of features and for each classifier. Then we compared the 50 errors of each dataset for both the kinds of features, i.e. topological versus raw data, using a T-test. This was done in order to validate that the differences we note are statistically significant. Here we reported, for each dataset the result of the t-test in terms of $[1, 0]$ is the H_0 is rejected or not, and its p-value. 1 means that the two samples are different, 0 means that they may come from the same distribution.

again, a smooth trend but Closeness and Centroid are under-represented.

When performing a classification task by using a set of features it becomes important to consider the actual nature of the features. This means that complex features, such as centralities, could be useless if compared to other features that are easier to compute like, for instance, the raw values coming from a proteomic analysis. Constructing a network and extracting the centralities values for each protein is computationally expensive. It becomes meaningful if the indexes contain more information than the raw data. Therefore, in order to verify the benefit of topological indexes as features, a comparison with the raw data features was performed. By using these data, i.e. -omics data, some recurring patterns also appeared, since the algorithm showed a good ability in separating the classes, but it also emerged that better results are obtained by analysing the features extracted using the networks as starting models. Only in one case, i.e. the PET dataset, the raw data scored better.

The fact that centrality-based scores remain, in general, higher is probably due to the fact that the models, from which such features are extracted, represent complex sets of interacting objects. Indeed, such models are able to consider not only the quantitative information but also the properties that arise from the interactions, thus enhancing the ability of the model at describing a specific process. On the contrary, raw data can only represent the numerical amount of a set of proteins, measured with a specific technique. The results demonstrated that the network analysis allows investigating complex behaviours and uncover hidden properties that emerge because of the complex interactions modelled as graphs.

The last step of the feature selection analysis aimed at testing some different feature selection algorithms on the same datasets, in order to support the results obtained through the Inf-fs approach we used. A specific Matlab library allowed us in order to test the Infinite Feature Selection (InfFs), Relief-F [105], Mutual Information (MutInf) [106], Concave minimisation (FSV) [107], Laplacian Score [108], Recursive Feature Elimination (RFE) [109], L0-based method [109], and Fisher method [110]. The features were ranked according to these algorithms and then a classification step, based on the SVM algorithm, allowed to test each ranking and to compute the classification Accuracy and error rate, i.e. $(1 - Accuracy)$. The results (see Fig. 3.8 and Fig. 3.9) show how, in general, the features we are using, i.e. the centralities, are good descriptors and allow separating the two classes, i.e. healthy versus unhealthy subjects, by using different approaches. It is important to note that we imposed to all the feature selection methods to extract a fixed number of features. This is the same number that was used to classify the results we presented (see Table 3.2). The same number of features let these results being comparable. It is also important to highlight the fact that we took advantage of the inf-FS algorithm present in the library. Again, we did it, in order to let the result being comparable with the results we already obtained, and with the other results coming from the same library, i.e. the other feature selection algorithms.

Some interesting biological insights

In order to obtain biological information about the classification result, it is necessary to retrieve information about the proteins that are linked to a specific topological feature. To do so, we decided to merge all the sets of selected features, e.g. we merged the 18 best features for all the feature selection algorithms in the AM dataset. We obtained a list of potential molecular targets that are always chosen as most informative, i.e. as best features, for all the datasets.

Amyloidosis The results that come from the feature selection algorithms we run over the Amyloidosis dataset (see Figs. A.13, A.14, A.15, A.16) shows few discrepancies and some similarities. It is possible to notice how the histograms obtained by using the algorithms Laplacian, InfFs, Rfe, Relieff and Fsv, show similar trends, since there is not a specific feature that emerges as most chosen centrality. In the other histograms, i.e. Fisher, L0 and Mutinffs, there is a tendency in preferring certain centralites over some others but, it is possible to say that, generally speaking the algorithms take advantage from a very complex set of features that somehow involves most of the centrality indexes we computed without a specifc preference. This fact suggests how all the topological indexes have a comparable informational content even though, in some cases, some features are preferred. It may be caused by the fact that this dataset has a limited number of features due to the very low amount of shared proteins, i.e. 3 proteins and 27 features, and, since 18 of them were chosen as best features, the range of choice is very limited. Finally it is possible to affirm that these results are similar to the results we presented above, even though some differences are present. It happens because the InfFS algorithm is not deterministic, thus the same input generates different features ranks.

B-Cell Lympocytic Leukemia The results that come from the feature selection algorithms we run over the B-Cell Leukemia dataset (see Figs. A.17, A.18, A.19, A.20), are very different in terms of selected features. Some of them took advantage from all the available features, few others only exploited the information of a smaller set of features. Globally it is possible to state that the four features, i.e. Stress, Betweenness, Centroid, and Degree, that are found to be useful with the Laplacian Score algorithm, appear to be selected as best features by all the feature selection algorithms.

Moreover, merging all the results permitted to obtain only a shared protein between all the best features selected. This fact suggests how different algorithms allowed selecting a very wide range of different features suggesting that the information that allow separating healthy and unhealthy subjects come from different sources. It is worth noting that it is possible to perform several merging operations, using a subset of the algorithms, in order to see if more shared proteins emerge. In this sense, the fact that the Laplacian algorithm only considered four features over nine available may be a limitation factor.

Myocardial Infarction The results that come from the feature selection algorithms we run over the Myocardial Infarction dataset (see Figs. A.21, A.22, A.23, A.24) show how all the features are selected as best features. The only difference that emerge from the comparison of the histograms refers to the RFE algorithm that show how the Closeness was not considered as a best feature.

By observing the proteins we extracted from the merging step it is possible to say that, apparently, the features are really robust and that these proteins, above the others, seem to be strongly related to the ability the classifier has, in separating the two classes of subjects. It is worth noting that, even thoug 155 couples $(centrality, protein)$ were found shared between all the algorithms, only 44 proteins are unique, in this set. These unique proteins are: ACO2, ACTA1, ACTB, ACTC1, ACTG1, ACTG2, ACTN1, ACTN2, ANXA2, APOA1, ATP5A1, ATP5B, C3, CFL2, CRYAB, DES, DPYSL2, ETFA, GAPDH, HADH, HADHA, HBD, HIST1H2BC, HIST1H4A, HSP90AA1, HSPB1, LDHB, LMNA, MDH2, MYH2, MYH7, SOD2, SPTAN1, SPTBN1, TGM2, TPI1, TPM1, TPM2, TPM4, TUBA1C, UBA52, VCL, VIM, YWHAZ.

Pancreatic Endocrine Tumours The results that come from the feature selection algorithms we run over the Pancreatinc Endocrine Tumours dataset (see Figs. A.25, A.26, A.27, A.28) show a high rate of variation. It is interesting to note that, like in the BCLL set of histograms, the Laplacian algorithm selected only three features as best features for classifying. It is also worth noting that there is not a specific feature that is always selected as best features. The variations occur also in the number of times a centrality appear. Here, we were able to retrieve a single protein, and this is probably due, as we pointed out for the BCLL, the high variations that occur between the histograms.

Finally it is important to note that a centrality describes a topological characteristic of a protein that plays a determined role in the network. Hence the feature comprises some biological implications that directly depend on the molecular composition of the network and that are used to retrieve recurring patterns.

Here we reported the tables for all the datasets.

Centrality	Protein
Stress	PRKACB

Table 3.4: **Centralities and proteins retrieved in the BCLL dataset**. In this table it is possible to appreciate the pairs $(centrality, protein)$ that resulted as most selected features by all the feature selection algorithms, for the B-Cell Lymphocytic Leukemia dataset.

Centrality	Protein
Stress	RPL3

Table 3.5: **Centralities and proteins retrieved in the PET dataset**. In this table it is possible to appreciate the pairs $(centrality, protein)$ that resulted as most selected features by all the feature selection algorithms, for the Pancreatic Endocrine Tumours dataset.

Centrality	Protein
Stress	ANXA2
Centroid	ACTB
Betweenness	ANXA2
Degree	VIM
Degree	ACTB

Table 3.6: **Centralities and proteins retrieved in the AM dataset**. In this table it is possible to appreciate the pairs $(centrality, protein)$ that resulted as most selected features by all the feature selection algorithms, for the Amyloidosis dataset.

Softwares

The results of the feature selection comparisons were computed using the Infinite FS, Relieff, Mutinf FS, FSV, Laplacian, RFE, L0, and Fisher. The accuracies are obtained using an SVM classifier. To compute all these values we took advantage of an existing Matlab library, i.e. Feature Selection Library, version 2.1.1 [111]. The version of Matlab used for all the computations is the R2016A, the R version we used to depict the histograms and plots is 3.3.2 (2016-10-31), "Sincere Pumpkin Patch". To load the .mat files in R, we used the R.matlab v3.6.0 library.

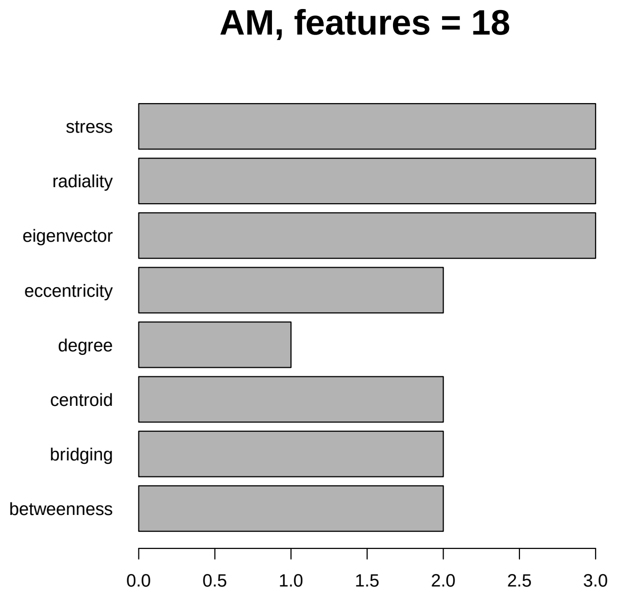
The classification algorithms comparsions were obtained using PRTOOLS [112]. We reported all the plots obtained by comparing five different classifiers. We compared topological versus raw data, i.e. proteomics, phosphorilation and gene expression raw data, using five different algorithms that are Minimum least square

Centrality	Protein	Centrality	Protein	Centrality	Protein
Stress	HIST1H2BC	Degree	HSPB1	Betweenness	TUBA1C
Stress	VIM	Centroid	HSPB1	Eccentricity	ATP5A1
Stress	ACTB	Centroid	TPI1	Bridging	TGM2
Radiality	HBD	Betweenness	MYH2	Eccentricity	UBA52
Bridging	TPM4	Stress	TPI1	Radiality	SOD2
Stress	ANXA2	Centroid	HADH	EigenVector	DES
Stress	ACTN1	Bridging	MYH2	EigenVector	CRYAB
Betweenness	TPM4	Betweenness	DES	Eccentricity	MYH7
Stress	APOA1	Eccentricity	HADH	Radiality	DES
Stress	ACTA1	Degree	C3	Betweenness	TGM2
Stress	LDHB	Degree	TPI1	Degree	ACTG1
Stress	CRYAB	Radiality	HSPB1	Betweenness	YWHAZ
Betweenness	ETFA	Degree	LDHB	Degree	CFL2
Betweenness	ANXA2	Centroid	UBA52	Radiality	ACTG1
Stress	SPTBN1	Degree	MDH2	Degree	TGM2
Betweenness	SPTBN1	Centroid	MDH2	EigenVector	ACTG1
Stress	TPM4	EigenVector	ACTA1	Eccentricity	DES
Stress	ETFA	Bridging	MDH2	Radiality	CRYAB
Bridging	SPTBN1	Eccentricity	LDHB	Degree	TPM1
Stress	ATP5A1	EigenVector	MDH2	Centroid	TPM1
Stress	DPYSL2	Bridging	DES	Eccentricity	ACTG1
Bridging	ACO2	Eccentricity	ACTA1	Degree	ATP5A1
Betweenness	CRYAB	Centroid	LDHB	Degree	TPM2
Bridging	DPYSL2	Bridging	UBA52	Eccentricity	APOA1

Table 3.7: **Centralities and proteins retrieved in the MI dataset, part 1 of 2**. In this table it is possible to appreciate the pairs $(centrality, protein)$ that resulted as most selected features by all the feature selection algorithms, for the Myocardial Infarction dataset.

linear classifier (fisher), Support Vector Machines (SVM), Logistic linear classifier (loglc), Breiman's Random Forest (randomforest) and K-nearest-neighbours (KNN).

The results of the feature selection methods comparison were computed using the Infinite FS, Relieff, Mutinf FS, FSV, Laplacian, RFE, L0, and Fisher. The accuracies are obtained using an SVM classifier. To compute all these values we took advantage of an existing Matlab library, i.e. Feature Selection Library, version 2.1.1.



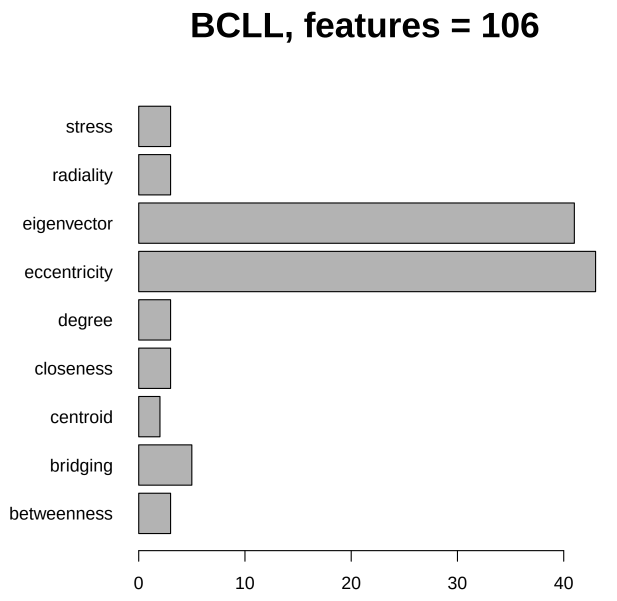


Figure 3.6: **The number of features chosen for each dataset in order to classify the subjects**. The histograms show how often a centrality was chosen by the feature selection as the best feature for the classification task. Each dataset has its own set of best features that depends from the parameter alpha, which was set at level 0.2. The bins represent, from the top to the bottom, Betweenness, Bridging, Centroid, Degree, Eccentricity, Eigenvector, Radiality and Stress centralities. The values represented in the X axis define the total number of times a feature is chosen in the set of the best features. This number varies accordingly with the dataset. It is a median value extracted from the set of trials the algorithm automatically performs.

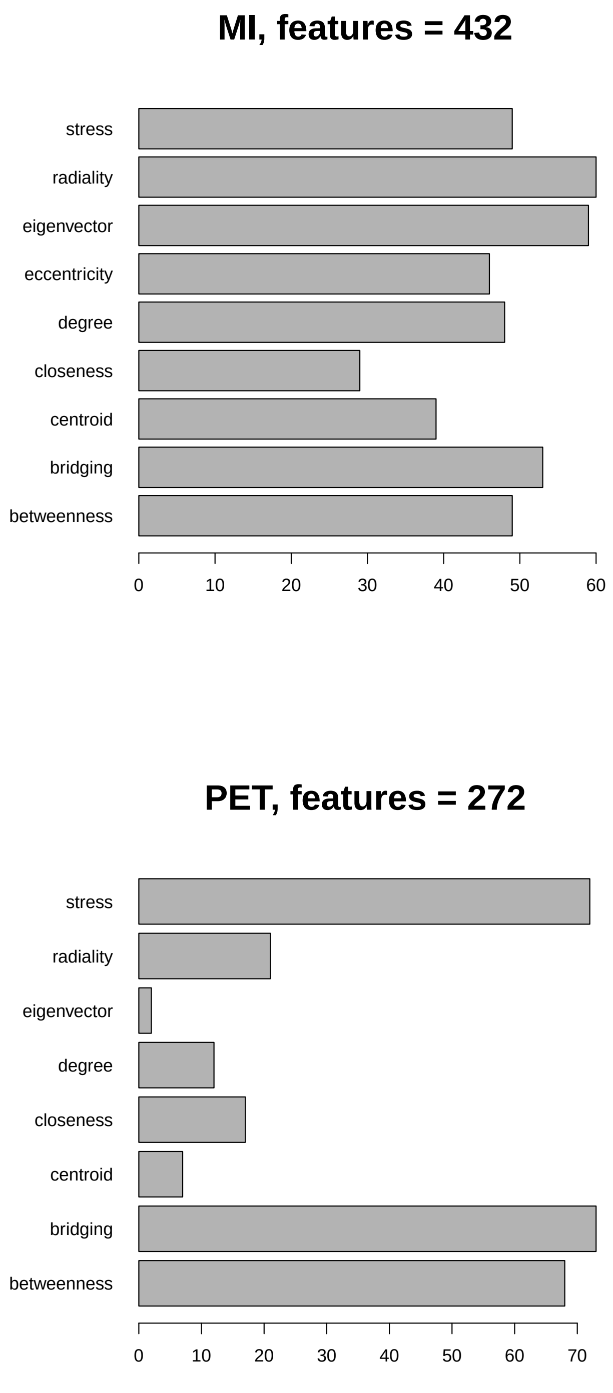
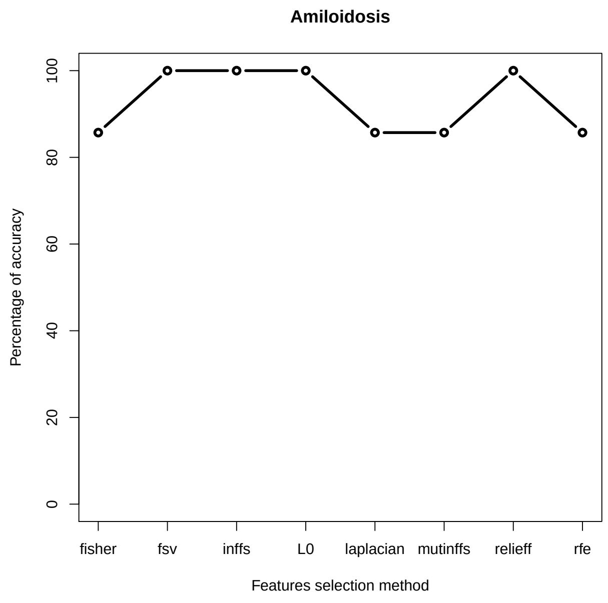


Figure 3.7: **The number of features chosen for each dataset in order to classify the subjects**. the histograms show how often a centrality was chosen by the feature selection as the best feature for the classification task. Each dataset has its own set of best features that depends from the parameter alpha, which was set at level 0.2. The bins represent, from the top to the bottom, Betweenness, Bridging, Centroid, Degree, Eccentricity, Eigenvector, Radiality and Stress centralities. The values represented in the X axis define the total number of times a feature is chosen in the set of the best features. This number varies with the dataset. It is a median value extracted from the set of trials the algorithm automatically performs.



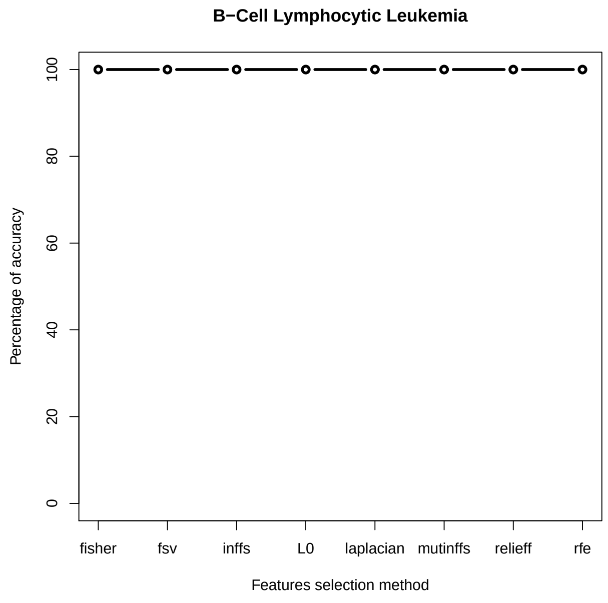
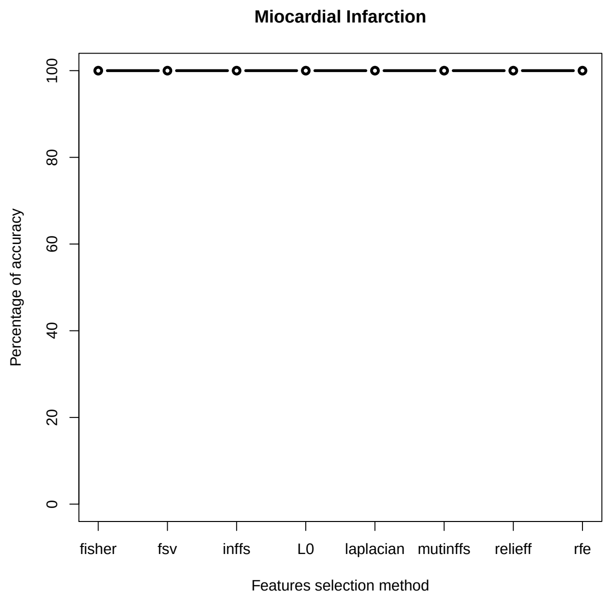


Figure 3.8: **Showing the percentages of Accuracy of the SVM algorithm.** On the X axis the algorithms that were used are listed, on the y-axis it is possible to appreciate the percentage of Accuracy each algorithm scored. The error rate results from this subtraction: $100 - Accuracy$, since the values that are show are considered in a range of $[0 - 100]$.



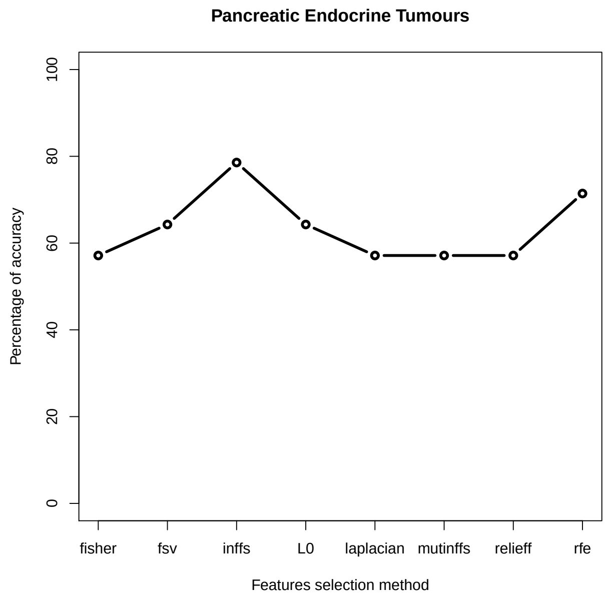


Figure 3.9: **Showing the percentages of Accuracy of the SVM algorithm.** On the X axis the algorithms that were used are listed, on the y-axis it is possible to appreciate the percentage of Accuracy each algorithm scored. The error rate results from this subtraction: $100 - Accuracy$, since the values that are show are considered in a range of $[0 - 100]$.

Centrality	Protein	Centrality	Protein	Centrality	Protein
Betweenness	DPYSL2	Betweenness	ACTC1	Bridging	ACTB
Stress	HADHA	Eccentricity	MDH2	EigenVector	TPM1
Stress	TUBA1C	Radiality	ACTA1	Centroid	ATP5A1
Stress	MDH2	Eccentricity	ANXA2	Bridging	TUBA1C
Bridging	CRYAB	Eccentricity	CRYAB	EigenVector	ATP5A1
Betweenness	UBA52	Degree	DPYSL2	Eccentricity	HSP9AA1
Degree	APOA1	Bridging	ACTN2	Bridging	TPI1
Bridging	HIST1H4A	Betweenness	ACTB	Degree	SPTAN1
Betweenness	HIST1H4A	Betweenness	VCL	Eccentricity	HADHA
Stress	HIST1H4A	Bridging	ATP5A1	Eccentricity	ACTG2
Betweenness	ATP5A1	EigenVector	LDHB	Radiality	ACTN2
Stress	UBA52	Bridging	HSP9AA1	Eccentricity	CFL2
Betweenness	HSPB1	Radiality	MDH2	Eccentricity	TPM2
Stress	DES	Betweenness	MDH2	Eccentricity	DPYSL2
Bridging	LMNA	Radiality	APOA1	Radiality	ACTB
Stress	VCL	Degree	ACTG2	Centroid	APOA1
Betweenness	C3	Radiality	LDHB	Centroid	LMNA
Stress	TGM2	Centroid	ACTN2	Radiality	SPTAN1
Betweenness	TPM2	Degree	HADHA	Eccentricity	TPM1
Stress	TPM2	Centroid	GAPDH	Bridging	TPM1
Degree	HADH	EigenVector	MYH7	Radiality	ETFA
Stress	MYH2	Degree	MYH7	Radiality	DPYSL2
Betweenness	HSP9AA1	Degree	DES	Radiality	C3
Betweenness	APOA1	Centroid	MYH7	Radiality	ATP5A1
Eccentricity	HSPB1	Betweenness	TPI1	Radiality	TPM2
Degree	UBA52	Eccentricity	ACTB	EigenVector	MYH2
Bridging	TPM2	Centroid	DES	Degree	HSP9AA1
Centroid	ANXA2	Centroid	ATP5B		

Table 3.8: **Centralities and proteins retrieved in the MI dataset, part 2 of 2.** In the table it is possible to appreciate the pairs $(centrality, protein)$ that resulted as most selected features by all the feature selection algorithms, for the Myocardial Infarction dataset.

Part II

Validating experimental findings using random networks

Chapter 4

Network Randomizer

4.1 Introduction

In network analysis a lot of tools have been developed to address different problems related to the extraction of useful information from systems modelled as graphs. Cytoscape [113] is a very well known platform that supports hundreds of apps that simplify the information mining of complex networks, with a specific focus on different kinds of biological applications. By using Cytoscape it is possible to perform topological analysis, cluster and motifs retrieval, biological enrichment, draw nice graphs, search for ontologies, etc. As the number of apps increases, the possibilities of performing more and more complex analysis grow together with the amount of information that could be retrieved. The problem is that these analyses remain preliminary to further experimental validation and, in this sense, a sort of benchmark for an in-silico validation is required [114]. A possible solution to this issue may come from the literature or experimental data about the process under investigation [115]. But this is not always possible since the network may represent some complex processes whose biology is not well understood, or that take advantage of some novel insights that require a different validation. In these cases, a possible solution refers to the generation of random experiments and the comparison of the results that come from the real networks with the results that come from the random networks analysis. This is basically a statistical validation which allows researchers to state that the results are not randomly distributed.

Indeed, Cytoscape has some apps that allow comparing networks. These apps have different features and goals. The most similar app, with respect to the one we developed, is Randomnetworks[1] that works for Cytoscape 2.x. Currently, it is not available for the new version of the Cytoscape platform, hence older versions are required in order to exploit the app. Another app that takes advantage from random network models is NetMatch*[2]. It allows retrieving all the occurences of a subgraph in a bigger network. Other apps that allow comparing

[1]`http://apps.cytoscape.org/apps/randomnetworks`

[2]`http://apps.cytoscape.org/apps/netmatchstar`

networks agains each other are GASOLINE[3] and DyNet[4]. GASOLINE allows retrieving subnetworks that are shared along a set of networks for obtaining a similarity value. DyNet has several functionalities that allow comparing nodes and edges between networks. Considering that the RandomNetwork app is no longer maintained, Cytoscape lacks of an app that allows randomising, creating and comparing networks in order to validate the results. This is another, important, reason that guided the develoment of NetworkRandomizer. Allowing Cytoscape users to perform the same pipeline we are describing here is essential in order to spread this methodology, hence generating random network becomes a fundamental step.

In this sense, our app was created to address the problem of creating a validation layer which allows simulating random experiments. By using randomly created networks it becomes possible to compare, through specific statistical tests, the numerical results that come from a common network analysis. To do so, our NetworkRandomizer [76] allows randomising existing networks by using a simple shuffling algorithm and a Degree preserving algorithm, since these are the most used shuffling algorithms. A Degree preserving algorithm is useful when the Degree represents an important feature of a network like, for instance, in scale-free networks where some nodes are hubs and some others are not. By keeping the Degree fixed we are sure that the scale-freeness of the shuffled network will be guaranteed. A scale-free network could be achieved by using the Barabási-Albert model but the Degree of the nodes, as it is in the original network, will be lost.

Moreover, it is possible to create Erdős-Rényi, Barabási-Albert, Watts-Strogatz, Lattice, and Community Affiliation models. We also implemented a new model, called Multiplication model, which is designed to generate weighted networks where nodes are multiplied, i.e. represented in different copies which have the same topological characteristics, to fit experimental quantitative data. The models we implemented are not all actually *random* network models. This means that, generating a Watts-Strogatz network does not mean that we are generating a totally random network. What we are doing, while genererating a Watts-Strogatz network, is creating a network with specific characteristics that fits some parameters. If we generate a network 100 times, using the Watts-Strogatz algorithm, it will not be the same network repeated 100 times. Instead, we will obtain 100 different networks, depending on how we set the parameters. The app aims at generating networks that follow the most known models of complex networks in oder to let the users comparing their biological, real networks with a set of synthetic, randomly generated networks.

Finally a statistical module, based on the two-sample Kolmogorov Smirnov test, compares network attributes in order to see if their values come from the same distribution or if they should be considered different. The original idea behind the app was to compare topological attributes computed by using CentiScaPe or the Cytoscape's built-in NetworkAnalyzer. But, our app, also supports the comparison of all kinds of numerical attributes in order to be useful to any user performing any analysis.

[3]http://apps.cytoscape.org/apps/gasoline

[4]http://apps.cytoscape.org/apps/dynet

The application is released with an Apache License 2.0. Its code is freely available as Git repository and downloadable from the Cytoscape app store[5].

4.2 Implementation

NetworkRandomizer is a Cytoscape app, developed in Java, which is designed for the Cytoscape 3.x environment. Its source code is available from the GitHub website where it is kept updated. Each new version of the app is released through the Cystoscape app store and the latest version is always available, as Java source code, from the Git repository. Our application has a very simple and modular structure that makes it easy to add additional network models and features in future releases.

Each random model we implemented is represented by its specific class which extends an abstract class that defines the common behaviour all the models share (see Section 4.3). Each model is initialised by using some specified parameters that are passed as input to the algorithm through the main GUI. Once the generation, or the randomisation, of the network(s) is done, the app deploys the newly created network(s) into the Cytoscape's Network Control Panel making them instantly available to the user for the further analysis. Since the users may want to generate a number of random networks, the network views are not created automatically by the app. This feature was developed to avoid a high number of pop-up networks in the main Cytoscape window while the algorithm is still working. Moreover, visualising huge network results in a memory consumption that slow down the entire computation. Eventually the user could create the view for a single, or more, network.

As we already said, the app is divided in several classes which allow a very flexible and modular structure. The CyActivator class runs the application and communicates with Cytoscape. The RandomizerCore class is used as a model of the current Cytoscape state like, for instance, the network handling. The MenuAction class allows the app initiation. The OptionsMenu class refers to the main GUI that is used to interact with the app. The ThreadEngine class allows creating and handling single and multithreads tasks even though the multithread is not yet implemented. Finally, the AbstractModel is the abstract class that defines the basic random model and offers several methods that could be useful when defining a new model. This is the starting point for each random models we developed. The other classes in the app refer to the models we actually implemented. In order to add a new model a new class should be instantiated by following the AbstractModel implementation and then it is necessary to modify the GUI in order to let the new model become part of the app.

The statistical module we implemented exploits the two-sample Kolmogorov Smirnov test [116]. This statistical test considers each pair of real and random network(s) and, for each attribute, computes the difference between the two distribution. The K$-$S definition of the distribution difference is obtained by computing the maximum gap between the cumulative probability functions of those

[5]`http://apps.cytoscape.org/apps/networkrandomizer`

probabilities. We decided to use this method since it relies on the cumulative probability functions withouth having the need to explicitly calculate them. Indeed, the algorithm sorts the two series of values and then, after normalising by the highest value, it sums up these values while performing the comparison. Finally, only the largest difference is considered.

4.3 AbstractModel class

The AbstractModel class is the basic class that describes the behaviour of all the random network models we implemented. Indeed, it is extended by all the other random network models of the NetworkRandomizer app. The code we present here is not the full code. It lacks of the algorithms that defines the behaviour of the methods of the class, i.e. we just reported the names of the methods. Some comments are also available in order to let the code be more readable.

A more comprehensive view of the code of the entire app, and the source code, is freely available online. We store it in a GitHub repository at `https://github.com/gabrielet/Network-Randomizer`.

In this class we implemented several methods that has many different functions. Some of them allow initialising and executing a random model. Some other methods allow creating a network with a specific number of nodes and edges. Finally, there are few other methods that allow handling and managing the networks.

```
public abstract class AbstractModel {

    protected class Edge {
        public Edge(int a, int b){}
    }

    public AbstractModel(RandomizerCore core){\dots}

    public void InitializeAndExecute() throws Exception {
        Initialize();
        Execute();
    }

    /**
     * Initialisation independent of the model.
     */
    public void Initialize() {
        initializeSpecifics();
    }

    /**
     * Initialisation specific of each model, called from
         Initialize. If unused,
     * leave empty.
     */
    abstract protected void initializeSpecifics();

    /**
```

```
     * Main execution point. This method is called by the
         ThreadEngine after
     * Initialisation.
     */

    public abstract void Execute() throws Exception;

    /**
     * Generate a network with a given number of nodes and no
         edges. For network
     * naming convention, see <code>getStandardNetworkName</code>
         method. Nodes
     * would be named by integers from 0 to <code>(numbderOfNodes
         - 1)</code>.
     *
     * @param numberOfNodes
     * @return CyNetwork generated
     */
    protected CyNetwork generateEmptyNetwork(int numberOfNodes) {
        //returns a network;
    }

    protected CyNetwork generateNetworkFromNodeList(ArrayList<
        String> nodesNames) {
  //returns a network;
    }

    protected CyNetwork copyOfCurrentNetwork(boolean directed) {
        //returns a copy of the Existing Network;
    }

    protected CyNetwork copyOfExistingNetwork(List<CyNode>
        nodelist, List<CyEdge> edgelist, boolean directed) {
        //returns a network;
    }

    /**
     * Send network to Cytoscape.
     *
     * @param net network to be sent to Cytoscape.
     */
    protected void pushNetwork(CyNetwork net) {\dots}

    /**
     * Model name to be used for network naming
     *
     * @return
     */
    protected abstract String getModelName();

    /**
     * Network naming convention is "model_time_rand" where: model
          is the name
     * returned by <code>getModelName</code> method, time is the
         current time in
     * milliseconds returned by the system and rand is a random
```

```
        three-digit
     * number.
     *
     * @return network name following the naming convention.
     */

    protected String getStandardNetworkName() {
        //returns a name;
    }

    protected boolean randomBoolean(float probabilityOfTrue) {
        //returns a boolean;
    }

    protected void endalgorithm() {
        stop = true;
    }

    protected CyNetwork getCurrentNetwork() {
        //returns the current network;
    }

    protected String getEdgeName(CyNode source, CyNode target,
       CyNetwork net) {
        //returns a String "Edge_" + sourceName + "_" + targetName
            ;
    }
}
```

4.4 Workflow

The NetworkRandomizer app was initially designed to interact with the CentiScaPe centralities. Then, we decided to let the user decide which kind of attributes to compare so, currently, our application works for all kinds of numerical attributes. These information may come from a specific app or could be defined by the user itself. The workflow we defined consists of a few distinct steps, in order to make it very easy to follow. The first step requires to load one, or more, networks that model any kind of process. After the loading phase, the user select one, or more, random model in order to generate some random networks. Also it is necessary to decide how many random networks will be generated. These random networks could be created in two different ways. Either by randomising the real network(s), using an algorithm between the Degree Preserving shuffle and the Completely Random shuffle (see Fig. 4.1), or by generating one, or more, new random network(s) by using one of implemented models (see Fig. 4.2 and Fig. 4.3) by filling the text field representing the input parameters. As said, the user could specify how many networks the app will generate, for each selected model.

After the real networks are loaded and the random networks are generated, the last step concerns the analysis of the shared attributes. In this sense the app provides a statistical module that allows the comparison of some, interesting attributes. Since the app was initially designed to be used in conjunction with the topological centralities parameters, we recommend the use of the CentiScaPe app which is completely compatible with our app. It is very important to note that the app allows comparing the attributes that have the same name in all the networks that were previously selected through the menu in the statistical module. It is also case sensitive, so particular care should be used when the attributes are manually curated.

Finally, when all the data are available, the statistical test (see Fig. 4.5) compares the selected attributes, highlighting their differences and similarities. The output is returned as a textual file which summarises the results. It is important to note that, depending on whether one or multiple networks were loaded, the final output file will be different. More specifically, if the user selected only one real network then the output file will have an higher level of details about the network. Otherwise, in the case that multiple real networks were selected, then the output file will contain a brief summary of all the statistical tests that were carried out.

Importantly, in order to make all the workflow completely accessible, each model, plus the statistical module, has a question mark button which provides some information to help to the users exploiting its full potential.

Data preparation

The idea behind Cytoscape is to allow the investigation of biological network properties through the use of the implemented tools. In order to exploit the functionalities offered by this software it is required to construct a network. The data preparation starts by obtaining a network and once it is loaded, it is possible to go through its analysis. In order to exploit the statistical module of our app some numerical information about the network's characteristics are required. As we said, we focused on topological information but any kind of numerical value works. There are multiple ways to obtain numerical information that mainly depends on the aspects that the user is interested into. Also it is possible to import a *.csv* file by using the Cytoscape loader, or to create a new attribute and manually fill it or, finally, to use one of the built-in network analysing function that generates numerical information.

The approach we recommend, to easily generate correct attributes, refers to the use of an app like, for instance, CentiScaPe. This app provides a lot of numerical information about the network(s) topology, grouped in distinct and meaningful attributes. Also it provides an interesting perspective about the characteristics of a network. The advantage that comes from the use of an app is that the user automatically obtains some standard names for the attributes and also some, very meaningful information, that could be compared among different networks. This makes the whole analysis easier and useful. Importantly, when using user defined attributes, the same attribute names must be used for both the real and the random networks. Indeed, no attribute will appear while performing the statistical analysis, if the names don't match.

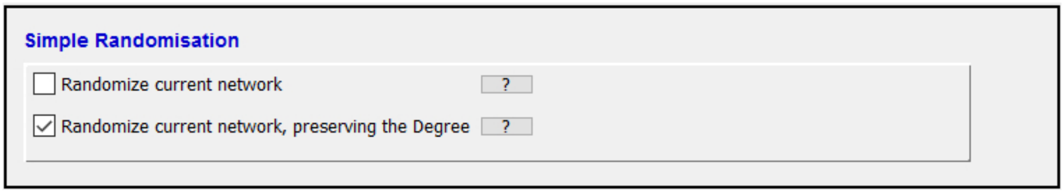
Generating random networks

The most trivial point while comparing a real dataset to a random generated set of samples concerns the methodology that should be used to generate the random samples. Generally speaking we suggest that, the more the random models mimic the characteristics of the real experiments, the better it is. Creating a random dataset that is comparable, in biological terms, gives very strong foundations to all the differences that is possible to highlight and gives a solid support while stating that the real data are consequence of the biology, and not due to chance. It is not possible to define, a priori, which model better fits a biological experiment and this is why we decided to implement more, different models.

Each user should select the best one to fits his/her own requirements and this is not easy. Moreover there is not a specific way to follow, hence the selection of the model is strongly related to what the researcher wants to show. To help in performing random experiments, our app provides two main methodologies for obtaining a set of random networks. The first method allows the randomisation of the current, real networks. This is probably the most common method that allows copying an experimental set-up. On the other hand, the second, available, option refers to the creation, from the scratch, of some parameters-dependent networks.

The randomisation of an existing network requires to select a network into the control panel. It is not necessary that a view of the network is available to Cytoscape, to perform the computation. So, if the view is not necessary, we suggest to not create it, since the visualisation of huge network requires a lot of memory and the computation will be slower. Once the network is selected, i.e. it's name is highlighted in the panel, there are two randomising methods that can be used *i)* the simple edge shuffle, and *ii)* the Degree preserving edge shuffle. The users should select one, or both, model(s) by checking the boxes (see Fig. 4.1). Once the module is selected, the Start button runs the randomisation. Here it is also possible to define the number of network that will be generated, as for the other models. It must be considered that the simple edge shuffle and the Degree preserving algorithm allow randomising existing networks but the two methodology may result in networks that are not very well randomised. This fact is more common when using the Degree preserving algorithm which has fewer degrees of freedom available, when it randomises the network. It may happen that the resulting randomised network is the same as the original since it was not possible to shuffle the nodes in order to keep the Degree fixed. This is more likely to happen with very small networks that have a few nodes and edges or by selecting the generation of a high number of networks. With few nodes and edges there are a few numbero of networks that can be created, while keeping the Degree, and if the number of the requested networks is higher than the number of possible permutation of nodes and edges, then some networks will be present in more than one copy.

The simple edge shuffle, from the version 1.1.2, allows randomising the existing network while keeping the names of the nodes as they were defined in the original network. This is particularly important for those networks which node names are important, like for instance biological networks where a node represent not only an abstract object but also some, very important biological information, e.g. a protein. Importantly, the earlier versions of the apps have a simple shuffle that creates a network where the nodes names are represented by sequential numbers and do not keep memory of their real names.



Simple Randomisation

- [ ] Randomize current network ?
- [✓] Randomize current network, preserving the Degree ?

Figure 4.1: **The randomisation interface**, from where the users can choose between a simple edge shuffle and the Degree preserving algorithm. Obviously it is possible to choose both. These two algorithms are used only for the randomisation of an existing network.

Random network models

The main part of the NetworkRandomizer consists of the most known, and widely used, random network models. The currently implemented models are:

- Erdős-Rényi [117];
- Watts-Strogatz [118];
- Barabási-Albert [119];
- Lattice graphs;
- Community Affiliation Graph [120];
- Multiplication model.

Erdős-Rényi

The Erdős-Rényi model (see Fig. 4.2) generates random networks by either uniformly choosing M pairs of nodes to connect or by connecting each pair with a specific probability p. It is very similar to the Gilbert model for creating random networks. These two models were introduced independently and generates a random graph. The difference rely on the fact that for the ER model, all the graphs generated with a fixed set of edges and nodes are equally likely to be chosen, while for the Gilbert model an edge has a fixed probability of being present of absent, independently from the other edges. The model we implemented is available in two versions. The $G(n, m)$ model allows selecting a random graph in the set of all the graphs that has n nodes and M edges. The $G(n, P)$ model assigns edges at random with probability p, independently from the other edges. These kind of graphs have a very low clustering coefficient and the Degree distribution converges to a Poisson distribution.

Watts-Strogatz

Watts-Strogatz model (see Fig. 4.2) generates networks which have some specific properties like, for instance, the generation of the small-world phenomenon. This model depends on three parameters that are n, κ, and β. The parameter n specifies the number of nodes the new network will have. The parameter κ is the mean Degree the network will have, and is expressed as an even integer. Finally, β is a parameter that ranges between $[0 - 1]$. If it is equal to zero then the algorithm generates a regular lattice, if it is equal to 1 the algorithm generates a random graph that is comparable to an Erdős-Rényi $G(n, M)$ model where n is equal to N, and $M = NK/2$. The model was proposed to overcome the limitation of the ER model and WS models are able to generate small world properties. On the other hand, the Degree distribution varies as the β parameter varies. With $\beta = 0$ the Degree distribution follows a Dirac Delta Function. With $\beta = 1$, as for the ER model, the distribution is a Poisson.

Barabási-Albert

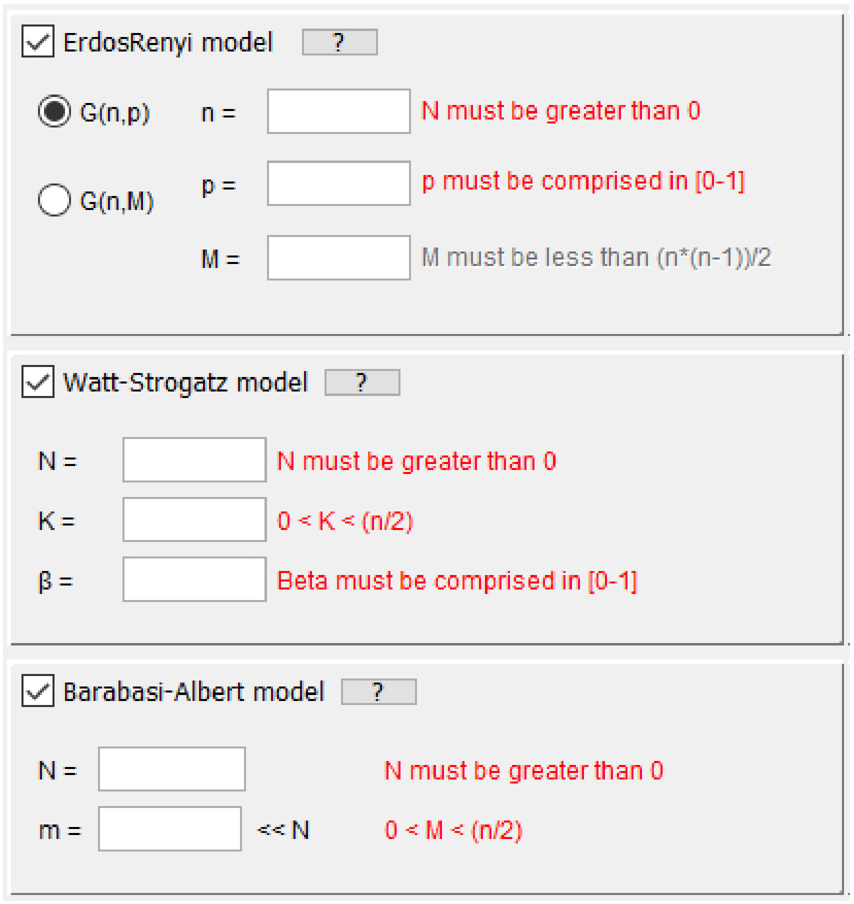
The Barabási-Albert model (see Fig. 4.2) generates scale-free networks that are extensively proven to be real world networks. In this sense it allows modelling networks that are found to be close to many natural and human-based systems. Scale-free networks means that the Degree distribution follows a power law distribution of the form $P(k) \sim \kappa^{-\gamma}$ and are built by using the preferential attachment growth. Preferential attachment means that *i)* the network grow over time and *ii)* each new node is connected to m other nodes, choosing them with a probability which is proportional to their Degree. The parameters that the algorithm requires, in order to generate a network, are m the number of the nodes at the time zero and N the final number of nodes the network has. At the first step, our algorithm construct a connected graph with m nodes and $\frac{m*m0}{2}$ edges. If $m0 > 3$ then $m0 = m * 2$ else $m0 = 6$. Then the algorithm iteratively adds one node at a time until the threshold N is reached. Each new node has an initial Degree equal to m.

Lattices

Lattice graphs model (see Fig. 4.3) generates regular, multidimensional lattices. Multidimensional lattices are grids of nodes, each of which is connected only to its first neighbours. For example, the one-dimensional lattice is a path graph, while the two-dimensional lattice is a square grid. Additionally, the users have the option of generating torus-shaped lattices, where there are no end-nodes. For instance, the generation of a one-dimensional torus lattice will result in a cycle graph.

Communities

The Community Affiliation model (see Fig. 4.3) generates random networks by using the community information that are passed as input by the user. Given



ErdosRenyi model ?

G(n,p) n = N must be greater than 0

G(n,M) p = p must be comprised in [0-1]

M = M must be less than (n*(n-1))/2

Watt-Strogatz model ?

N = N must be greater than 0

K = 0 < K < (n/2)

β = Beta must be comprised in [0-1]

Barabasi-Albert model ?

N = N must be greater than 0

m = << N 0 < M < (n/2)

Figure 4.2: **Some random network models**, like the Erdős Rényi, Watts Strogatz, and Barabási Albert models. These models require some parameters as input in order to generate a network. These parameters are chosen by the user. In the figure it is possible to note that some labels, in red, appears once the check box of a model is selected. These labels help the user in order to correctly fill all the required fields.

a list of communities, their members in terms of nodes, and the probability, in terms of p-value, of having an edge between two members of each community, the algorithm randomly generates realistic social networks.

Multiplicative

Finally, the Multiplication model (see Fig. 4.3) generates randomly weighted network starting from the existing network. This method does not generate a new network from the scratch but, by using an existing network, it creates a new, or more, multiplied network(s). The idea here, relies on the fact that integrating quantitative, experimental information is becoming a fundamental step in biological data analysis. The model generates a random array which defines a weight for each node belonging to an existing, user-defined network, starting from an attribute file which contains quantitative information about the nodes. Then the algorithm creates a new network which contains a number of copies that

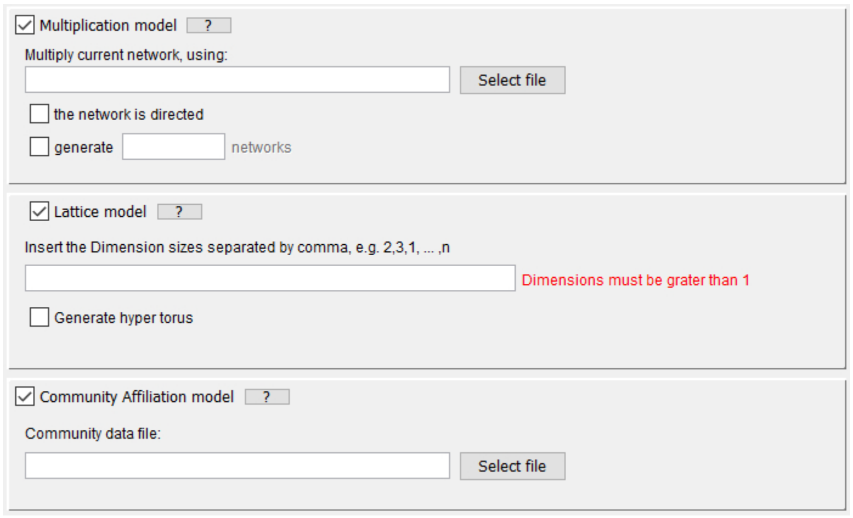
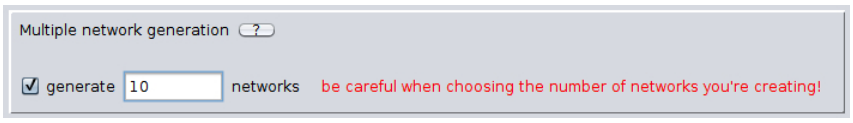

☑ Multiplication model ?
Multiply current network, using:
Select file
☐ the network is directed
☐ generate networks
☑ Lattice model ?
Insert the Dimension sizes separated by comma, e.g. 2,3,1, ... ,n
Dimensions must be grater than 1
☐ Generate hyper torus
☑ Community Affiliation model ?
Community data file:
Select file

Figure 4.3: **Some random network models**, like the Multiplication model network generator, lattice generator and the Community Affiliation Graph model. Both the Multiplication and the Community Affiliation requires a file as input, in order to generate random networks. Some red labels, as for the other models, appear in order to guide the user. Also the question mark boxes were implemented to give some further information.

depend on the random weights that were assigned to each node. This random weighted network represents a possible experimental set-up which derives from the attribute file. In this sense it is possible to generate a number of randomly weighted networks, in order to simulate a set of experiments.

App input

Through the main GUI whenever the user click on the check mark of a model, some info will appear to help and guide the users while filling the text boxes regarding the input parameters constraints (see Fig. 4.2). These labels change and update their values, in order to be more helpful, once they are correctly filled-in and after pressing the Enter button. Also, few implemented models require specific input files which define their behaviour. Currently only two of the implemented models require an input file. They are the Multiplication and the Community Affiliation model. The Multiplication file must contain numerical values, one for each row. They can be integers or double. It is expected, but not mandatory, that the number of rows is equal to the number of nodes and the NetworkRandomizer will pop-up a dialog which asks the user if the number of values found in the file is correct. The Community Affiliation model requires a file which contains a set of rows where each line represents a community. The rows starts with a p-value, i.e. the probability of an edge between two members of the community, and is followed by the names of the nodes inside that community, separated by spaces. There is a known bug, that affects the Multiplication input file. This issue is probably due to the Java version. It, sometimes require double that have a comma and sometimes double that have a dot as separator between the integer and decimal part of a number. It is quite easy to detect the error since the multiplication range, suggested by the dialog tha appears when the network generation starts, is wrong. In this case, the best solution is to substitute the commas/dots with dots/commas in the text file.


Multiple network generation ?

☑ generate 10 networks be careful when choosing the number of networks you're creating!

Figure 4.4: **Selecting the number of networks to be created**, is done using this text box. Once the check mark is selected, it allows to choose how many networks will be created for each selected model. It is important to note that a number of networks will be created and their dimension may affect the memory consumption of the whole Cytoscape software.

It is worth noting that once a model is selected in the GUI, then each one of the selected model will result in the desired number of random networks, every time the Start Randomisation button is pressed. So, removing the check mark from all the models that are selected, will prevent the generation of unwanted networks that may slow down the analysis. This recommentadion becomes very important if more networks are supposed to be created (see Fig: 4.4). Since it is possible to select the number of networks to be created for each model then, for instance, if the user inserted a value equal to 10 as a result, ten networks will be created, for each selected model, every time NetworkRandomizer runs.

Comparing networks

The final step of the workflow we designed concerns the comparison of the networks. In order to perform this task a random network must be generated and at least a numerical attribute should be present in all the networks that are supposed to be compared. To run the statistical comparison module it is necessary to specify which networks the algorithm will use. To do so, the user needs to select the real networks in the Networks Control Panel and press the Selected button in the NetworkRandomizer GUI (see Fig. 4.5), to confirm the selection. Then, some random networks must be selected in the same way. These random networks will be used as benchmark to compute the difference. If all the selected networks do not share at least one attribute, i.e. the names of the attributes do not match, then it is impossible to perform the comparison. Note that the attributes names are case sensitive. Also even if a network does not have the proper attribute name, this attribute will not appear in the list, so the naming of the attributes is very important. So, once the networks are selected, the shared attributes appear in the module and it is now possible to select one, or more, attribute(s) for the final analysis. If there are no attributes that are available, the user must check their names in the Node Table panel of Cytoscape, making them suitable and then he/she must re-select the real and the random networks to compare. If it is not possible to find a shared attribute, then a dialog will appear suggesting to the user to check again the attributes names or to select different networks. The attributes

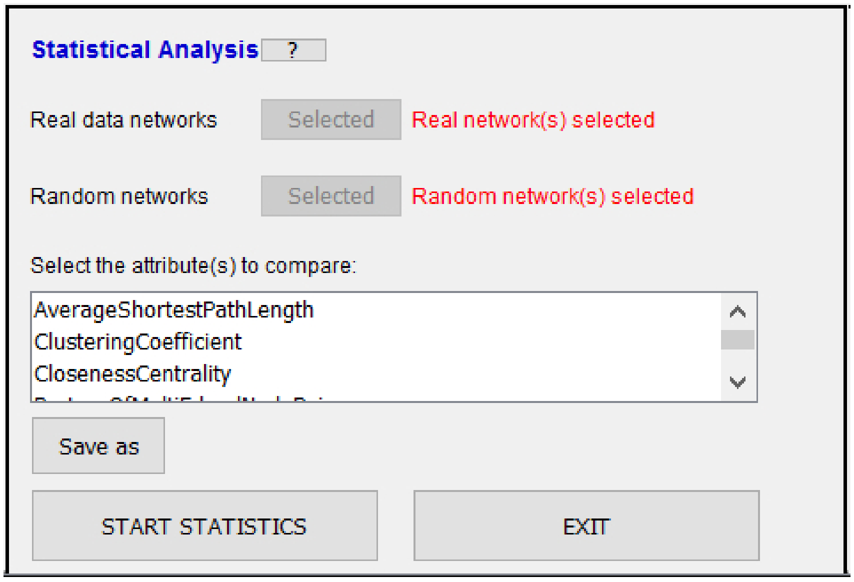


Figure 4.5: **The statistical module implemented in the app.** After the data are generated, the statistical module allows comparing all the shared network attributes in order to find important patterns. The attribute to be compared must have the same name in all the networks that are selected. Importantly, the name match is case-sensitive so some care should be taken while using upper or lower case. In this example it is possible to note that both real and random networks are already selected. The red label appears once the button Select is clicked. Also it is possible to note that some attributes are present in the box and are waiting to be selected.

could be selected by using the left-click and some, additional keys that are *Ctrl* for one-by-one multiple selection, *Shift* for the range selection and *Ctrl+A* to select all the attributes. Once the attributes are selected, it is necessary to specify an output file name and the directory where the file will be created and saved. Finally the comparison can be executed by clicking the Start Statistics button.

Two-Sample Kolmogorov Smirnov

The two-sample Kolmogorov-Smirnov test is a non parametric hypothesis test that permits to evaluate how different are the cumulative distribution functions of the one-dimensional distributions that come from the two samples. The H_0, or null hypothesis, for this test assumes that the two samples comes from the same distribution.

The K-S does not give an idea about the distributions the two samples come from. The statistics is computed by:

$$D_{n,n'} = \underset{x}{sup}|F_{1,n}(x) - F_{2,n'}(x)|$$

where $F_{1,n}(x)$ and $F_{2,n'}(x)$ are the empirical distribution functions of the two samples. sup is the supremum function, i.e. the least upper bound. n and n' are the dimension of the two samples.

H_0 is rejected at level α if:

$$D_{n,n'} > c(\alpha)\sqrt{\frac{n+n'}{nn'}}$$

$c(\alpha)$ is defined in a table, depending on the value of α.

Interpreting the results

The statistical module returns a textual file that consists of several, comma-separated-values fields. Each field has a specific function, marked by the $>$ symbol. The first and the second fields give some information about the networks, real and random, that were compared. The third field summarises the names of the attributes that were used for the statistical analysis, and works as a remainder. From the fourth field on, we listed the actual statistical results that come from the K-S test. These fields are different depending on whether only one or multiple real networks were selected for the comparison. In the case the user select a single real network the output will comprehend these fields:

- Average difference across all centralities between real and random networks;
- Difference between real network and its most similar random network for each centrality;
- Difference matrix between real and random networks for each centrality.

Here, the first field indicates how different is each random network from the real one by using the average difference across all centralities. The second field represents the difference between the real network and its most similar random network, according to each centrality measure individually. The last field provides more in-depth information, by specifying the difference between the real network and each random one, for each centrality measure.

On the other hand, if multiple real networks were selected the output file will show these fields:

- Average difference across all centralities between real networks and their most similar random network;
- Difference between real networks and its most similar random network for each centrality;
- Average difference matrix between real and random networks across all centralities.

We decided to summarise the results since showing all of the generated data to the user, in the case of multiple real networks, would result in a very unreadable output. To avoid this possible outcome, only the most interesting points were chosen and are shown. Either the pairs of real and random networks, or the pairs of real networks and centrality measures which show the least statistical differences are put in the output file. In this way the user can check the networks for important non-random processes. The first field specifies the average difference across all centralities between real networks and their most similar random network. The second field shows the most similar random network with respect to the real one, for each real network and each centrality, and specifies the level of the difference. The last field refers to the real-random distance matrix. The distances are defined as the average difference across all the centralities. A value of 0 indicates that the distributions are completely the same, up to a normalisation factor that was used to let the value being comparable. The normalisation factor here, refers to the number of elements in the series. So, for instance, the series $[1, 2, 3]$ and $[1, 1, 2, 2, 3, 3]$ would be completely the same. On the other hand, a value that is close to 1 indicates the existence of an important difference between the distributions. This result is rarely achieved in real datasets. It may happen, for instance, when the elements of one series are all greater than the elements of the other.

4.5 The R implementation

The multiplication model allows the multiplication of the number of nodes describing a protein, by using a specific attribute. Then the resulting number of copies is added to the original network, causing the generation of some new interactions that better fits the actual biology of a process. We have implemented this algorithm through two functions by using the R language. The idea here was to allow more advanced users to exploit the existing *IGraph* library[6]. IGraphs is a very powerful tool that can be used, in the R, Python and C environments, to perform different kind of network analysis.

In particular, we implemented two main functions that allow the basic operations to multiply and analyse a multiplied networks. The function *addEges()*, allows to create and analyse weighted networks. Currently, three topological metrics, over the whole set of centralities that are implemented by the R IGraph library could be used to extract topological information. These centralities are Closeness, Betweenness, and Alpha Centrality.

The second function we have implemented, i.e. *rgSim()*, allows to perform a random simulation that generates a number of weighting arrays, depending on how many simulations the user wants to perform, that can be seen as a set of random experimental data. These arrays multiply the original network, substituting the experimental data, to obtain a dataset of randomly personalised networks and their average, selected centrality. The intent of the multiplication function is to analyse personalised datasets, starting from a traditional static interactome, in order to model a real experimental context. The random simulation is designed as a validation benchmark for the experimental results, by using a set of randomly generated networks as controls.

We decided to not implement a statistical module, like the one available in our NetworkRandomizer app, since R is a statistical programming language and

[6]`http://igraph.org/r/`

it is assumed that an R user is able to perform its own statistical tests. Also, R has a wealth of already implemented statistical tests.

The library we developed, contains the two mentioned functions and some other functionalities that are designed to help researches in order to interface R with Cytoscape. Since Cytoscape uses some very specific file format that are not standardised, we decided to develop a function that allows the users importing their networks while keeping the nodes names, into the IGraph environment. To this purpose we developed a function, called *loadSif*, which allows creating a network from a sif file.

Also, we developed some functions that supply to the lack of very specific functionalities, that are developed in our Cytoscape apps like the PeSca app. These functions allow finding shortest paths and using them to build meaningful, connected networks.

The two, main functions are:

- *addEdges*: allows creating a multiplied network by adding copies of node according to a weighting array randomly generated. This function is called by **rgSim** and takes, as inputs, the network, the array of weights and a boolean value which is TRUE if the network is directed, FALSE otherwise. By default it is set as FALSE;

- *rgSim*: allows simulating several multiplied network by creating some random arrays that are used to weigh the networks. As input it takes several parameters that are the network to multiply, the number of simulations the algorithm will perform, the min and max values that determines a range in which the weight will be randomly chosen, a string that describes which centralities will be used for the comparison and a boolean value which is TRUE if the network is directed, FALSE otherwise. By default it is set as FALSE. The centrality is set to "Closeness" by default but, currently it may also include "Betweenness" and "Alpha.Centrality"

The functions that are implemented as part of the PeSca App project allow performing specific short path search in the networks. The function we developed are:

- *loadSif*: allows loading sif files in order to create a very specific network while keeping the nodes names. This is of basic importance in a biological set-up where the nodes has some, very specific names;

- *spCluster*: allows finding the shortest path among a set of selected nodes, as in the PeSca app, that are not connected. It requires, as input, a probe containing the nodes of interest and returns, as output, the connected network if the probe result in a disconnected network, the network formed by the nodes in the probe otherwise;

- *connectIsolated*: allows finding the shortest paths along a set of selected nodes that are not directly connected. It requires, as input, a probe containing the nodes of interest and returns, as output, the connected network if the probe result in a disconnected network, the network formed by the nodes in the probe otherwise;

- *findNodesInPaths*: allows showing the nodes that forms the shortest paths that were used to connect isolated nodes;
- *firstNeighbours*: allows creating a subnetwork which comprehend the nodes of interest and their first neighbours. It can be used as a preliminary analysis while trying to connect isolated nodes;
- *findNotConnectedNodes*: allows finding the nodes that are not connected to a network.

A future improvement will allow the function *rgSim* to exploit all the centrality indexes implemented in the IGraph library.

4.6 R code

The R code we developed requires that the IGraph library is installed in order to work. The code we listed starts by loading such library, since it is necessary to perform some of the operations that were used in our functions.

The term *component* indicates a subgraph that is formed by a set of nodes that are present in a bigger network. Here, as giant component, we mean the component with the higher number of connected nodes. In other words, all the nodes in the subgraph share interactions between each others but no interactions reach nodes outside this set. It is also possible to obtain more components. A disconnected component is usually created when, by using a probe, we extract a subnetwork from a bigger graph and we obtain more than one component. It is possible that some of the nodes in the probe does not share interactions hence, the subnetwork we created may contain different components that do not interact, like in Fig. 4.6. To overcome this limitation, i.e. to connect isolated components into subnetworks, it is possible to remap the subnetwork in the bigger graph and locate the probe. Once the probe is located, by finding paths between the disconnected component, new nodes will be integrated in the the subnetwork allowing the creation of new interactions which keep the isolated components communicating.

```
library("igraph"); #load the IGraph library

#this function allows finding those nodes that forms the shortest
    paths we are interested into and that connect two components.

findNodesInPaths <- function(edgelist){

  #initilise an array
  nodes<-array();

  #how many paths have i found?
  n_paths<-length(edgelist);

```

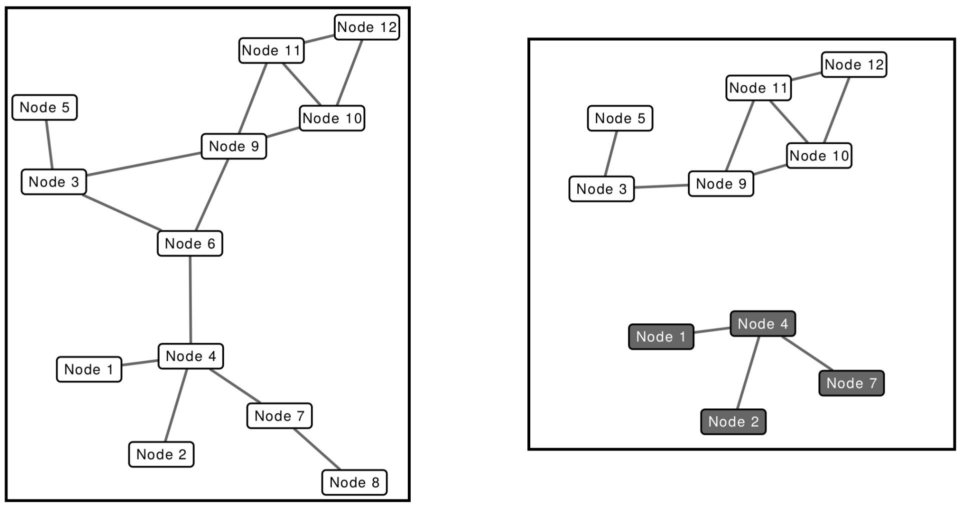


Figure 4.6: **An example of main component**. On the left it is possible to appreciate a network where all the nodes can communicate through, at least, a shortest path. On the right, two networks that are a subnetwork of the original graph. Suppose the probe contains the nodes in the two separated subnetworks, i.e. Nodes 1, 2, 3, 4, 5, 7, 9, 10, 11 and 12 . Once the probe is mapped in the left network and the resulting subnetworks are extracted the result will be two disconnected networks. As main component we mean the bigger subnetwork, i.e. the one on the right with the white nodes. The disconnected component, here, is the smaller subnetwork, i.e. the one with the dark gray nodes. If we look for connecting paths, using the bigger network, between the two components, i.e. the giant and the disconnected, then the resulting network will include the Node 6 in addition to the nodes in the probe. The Node 6 is essential to keep the two subnetwork togheter. This operation is done by using some of the functions we developed or, in Cytoscape, by using the PeSca app.

```
  #for each path
  for(i in 1:n_paths){
    #i need to know which nodes are involved
    nodes<-c(nodes,V(network)$name[unlist(edgelist[[i]])]);
  }

  #as output i return the values which occurs only once. i found
      them in the list minus the first element that is always null!
  return(unique(nodes[-1]));
}

firstNeighbours <- function(network,probe){
  first<-induced.subgraph(network,unlist(neighbourhood(network,
      order=1,probe)));
  return(first);
}

#this function allows finding the nodes that are disconnected.
#it is usually used after running the DECOMPOSE function

```

```
findNotConnectedNodes <- function(decomposednet){

  #how many disconnected components are there?
  cmps<-length(decomposednet);

  #initialise an array
  not_connected<-array();

  #for each component, starting from the second one since i assume
       the first one as the main component to which the other one
      will be connected
  for(i in 2:cmps){
    not_connected<-c(not_connected,V(decomposednet[[i]])$name);
  }

  return(unique(not_connected[-1]));
}

#this function allow importing .sif files

loadSif <- function(filename){
  sif_file<-read.table(filename);

  #creating the network by using the column 1 and 3 from the .sif
      file
  tmp_net<-graph.data.frame(sif_file[,c(1,3)],directed=FALSE);

  #settling the connected components by removing duplicated nodes
      and loops
  network<-simplify(tmp_net, remove.multiple=TRUE, remove.loops=
      TRUE);

  if(length(decompose(network))>1){
    print("verify that this network does not contain any
        disconnected component. if it does try spCluster or
        connectIsolated");
  }

  return(network);
}
```

The functions listed below are supposed to be used in order to find the paths between the disconnected components. They are developed in order to simulate the Cytoscape app, PeSca. It can be downloaded for Cytoscape 3.x and now it is possible to exploit some of its functionalities in R. This fact allows saving a lot of computational time, when dealing with very big networks, since R better deals with big networks than the Cytoscape software. Also, loading huge networks in Cytoscape requires machines that have an important amount of virtual memory making it difficult to proceed with further analysis.

Here, the term component has the same meaning as above. But we add a specification. Again, a component may be *giant* and *disconnected*. The *giant* component is the bigger component we are able to find in a subnetwork, in terms of number of nodes. All the other components are said *disconnected*. The idea here is that we want to connect the disconnected components to the giant component in order to obtain a connected network.

```
#the variable probe must contain the list of those node i am
    interested into. are they connected?

spCluster <- function(probe){
  #creating the subnetwork and computing how many components it
      has
  subnet<-induced.subgraph(network,probe);
  decomposednet<-decompose(subnet);

  if(length(decomposednet)>1){#if there is more than one component
       it means that they are disconnected
    giant<-V(decomposednet[[1]])$name;
    not_cnctd<-findNotConnectedNodes(decomposednet);

    #sp-cluster
    sp<-get.all.shortest.paths(network,giant,not_cnctd);

    #creating the list of those nodes that belong to the shortest
        paths just computed
    nodeslist<-findNodesInPaths(sp$vpath);

    #adding the two components (giant and disconnected)
    nodeslist<-unique(c(nodeslist,giant,not_cnctd));

    #and then i construct the network
    connected_net<-induced.subgraph(network,nodeslist);
    return(connected_net);
  }
  else{
    return("no disconnected component");
  }
}

connectIsolated <- function(probe){
  #creating the subnetwork and extracting its components
  subnet<-induced.subgraph(network,probe);
  decomposednet<-decompose(subnet);

  if(length(decomposednet)>1){#if there is more than one component
       it means that they are disconnected
    giant<-V(decomposednet[[1]])$name;
    not_cnctd<-findNotConnectedNodes(decomposednet);

    #connect disconnected component
    connect<-get.shortest.paths(network,giant,not_cnctd);

    #creating the list of those nodes that belong to the shortest
        paths just computed
    nodeslist<-findNodesInPaths(connect$vpath);

    #adding the two components (giant and disconnected)
    nodeslist<-unique(c(nodeslist,giant,not_cnctd));

    #and then i construct the network
    connected_net<-induced.subgraph(network,nodeslist);
    return(connected_net);
  }
  else{
    return("no disconnected component");
  }
}
```

The following functions allow simulating multiplied networks and creating experimental, biological data through a multiplied network approach. The **addEdges** function allows creating multiplied networks, the **rgSim** function allow generating randomly weighted networks. In order to let the user computing the topological centralities over multiplied networks we implemented some functions. These functions allow summing up the centralities of the original node and its copies and return a single value which reflects the weighted centrality.

The IGraph library implements several centralities, from the IGraph docs page [7]:

- alpha.centrality: find Bonacich alpha centrality scores of network positions;
- authority.score: Kleinberg's authority centrality scores.;
- Betweenness.estimate: Vertex and edge Betweenness centrality;
- centralization.evcent: Centralise a graph according to the Eigenvector centrality of vertices;
- centr_eigen: Centralise a graph according to the Eigenvector centrality of vertices;
- Closeness.estimate: Closeness centrality of vertices;
- edge.Betweenness: Vertex and edge Betweenness centrality;
- edge.Betweenness.estimate: Vertex and edge Betweenness centrality;
- edge_Betweenness: Vertex and edge Betweenness centrality;
- eigen_centrality: Eigenvector Centrality Scores of Network Positions;
- hub.score: Kleinberg's hub centrality scores;
- power_centrality: Find Bonacich Power Centrality Scores of Network Positions;
- subgraph.centrality: Find subgraph centrality scores of network positions.

[7] `http://igraph.org/r/doc/`

The **rgSim** implements the computation for three specific centralities that are: Closeness, AlphaCentrality and Betweenness. It is possible to extend the set of available centralities by adding an *if* loop for each new centrality. The algorithm, for each simulation, creates a numerical weights array in between a range, defined by the user. The number of simulation and the input network are defined by the user. The network can be directed and undirected.

The function **addEdges** algorithm works as follows. Starting from the network and a random array, the edgelist is visited; for each node the algorithm performs a loop for each copy the node has. The node 1 has 0 copies means 0 loops, the node 4 has 3 copies means 3 loops. For each loop a new node is added and a new set of edges is created. First the algorithm finds the position of the original node in the edge list. This step permits to get the neighbours of the original node that will be neighbours of each copy (rows: 33-42). Then an edge is added between the node and its copy (row: 43). Another edge is added if the network is considered as directed: the edges between the original node and its copies are not directed so a second edge A,B and B,A is added (rows: 44-46). Finally the new edges are added to the edgelist and the loops begin again with a new copy, if it exists.

```
1  #network is the graph we want to weigh in order to perform the simulation
2  #simulations is the number of networks the function will generate
3  #[val_min-val_max] is the range in between the weight vectors will be generated
4  #centrality could be a single centrality or a vector of centralities (i.e. alpha_centrality,authority_score,Betweenness,Closeness,edge.Betweenness,eigen_centrality)
5
6  addEdges <- function(network, weights, direction=FALSE){ #input: network, the weights and if the network is directed. default: undirected network
7
8    all_the_edges <- get.edgelist(network); #the edgelist of the network
9    new_edges <- c();
10   len <- nrow(all_the_edges);
11   new_node <- length(V(network)); #the new node i should add is nodes + 1
12   copies <- weights - 1; #total number of copies each node will add to the network
13
14   for(n in V(network)){ #for the node n
15     if(copies[n]>0){ #if there is at least a copy
16       for(cp in 1:copies[n]){ #for each copy
17         new_node <- new_node + 1; #a new node is going to be added
18         pos <- which(all_the_edges==n); #find the node in the edge_list
19         new_edges <- c();
20         for(p in pos){ #for each position create a new couple of values: (n,old_neighbour_of_n_father)
21           if(p<=len){
22             new_edges <- rbind(new_edges,c(new_node,all_the_edges[p+len]));
```

```
          }
          else{
            new_edges <- rbind(new_edges,c(all_the_edges[p-len], new_node));
          }
        }
        new_edges <- rbind(new_edges,c(n,new_node)); #add an edge between the new node and its father
        if(direction==TRUE){
          new_edges <- rbind(new_edges,c(new_node,n));
        }
        all_the_edges <- rbind(all_the_edges,new_edges);
        len <- nrow(all_the_edges); #how many edges
      }
    }
  }
  #COMMENT THIS TWO LINES
  new_network <- graph(c(t(all_the_edges)),directed=direction); # the new network
  return(new_network);
  #IF THE USERS NEEDS AN EDGE LIST INSTEAD OF A NETWORK THEN UNCOMMENT THE FOLLOWING AND COMMENT THE TWO LINES ABOVE
  #return(all_the_edges);
}

##############################################################
#use this method when the network nodes has specific names!#
##############################################################

addEdgesWithNames <- function(network, weights, direction=FALSE){ #input: network, the weights and if the network is directed. default: undirected network

  all_the_edges <- get.edgelist(network); #the edgelist of the network
  new_edges <- c();
  len <- nrow(all_the_edges);
  nnod<-length(V(network));
  new_node <- length(V(network)); #the new node i should add is nodes + 1
  copies <- weights - 1; #total number of copies each node will add to the network

  for(n in 1:nnod){ #for each original node
    if(copies[n]>0){ #if there is at least a copy
      for(cp in 1:copies[n]){ #for each copy
        new_node <- paste(V(network)$name[n],cp,sep="_"); #a new node is going to be added
        pos <- which(all_the_edges==V(network)$name[n]); #find the node in the edge_list
        new_edges <- c();
        for(p in pos){ #for each position create a new couple of values: (n,old_neighbour_of_n_father)
          if(p<=len){
            new_edges <- rbind(new_edges,c(new_node,all_the_edges[p+len]));
          }
          else{
            new_edges <- rbind(new_edges,c(all_the_edges[p-len],
                new_node));
          }
        }
        new_edges <- rbind(new_edges,c(V(network)$name[n],new_node
            )); #add an edge between the new node and its father

        if(direction==TRUE){
          new_edges <- rbind(new_edges,c(new_node,n));
        }
        all_the_edges <- rbind(all_the_edges,new_edges);
        len <- nrow(all_the_edges); #how many edges
      }
    }
  }
  #COMMENT THIS TWO LINES
  new_network <- graph(c(t(all_the_edges)),directed=direction); #
      the new network
  return(new_network);
  #IF THE USERS NEEDS AN EDGE LIST INSTEAD OF A NETWORK THEN
      UNCOMMENT THE FOLLOWING AND COMMENT THE TWO LINES ABOVE
  #return(all_the_edges);
}

rgSim <- function(network, simulations, val_min, val_max,
    centrality="Closeness", direction=FALSE){

  weight <- array();
  centralities <- array();
  clo <- array();
  betw <- array();
  alpha <- array();
  vertexes <- vcount(network);
  means_clo <- array();
  means_alpha <- array();
  means_betw <- array();
  label <- c("Closeness","alpha.centrality","Betweenness");
  results <- data.frame();

  if(all(centrality %in% label, na.rm=FALSE)){#if all the
      centralities are correctly typed then go on
    for(i in 1:simulations){#for each simulation an array of
        random weights is generated
      weight <- round(runif(vertexes,val_min,val_max)); #generates
          #vertexes weights within val_max and val_min
      #add the new edges to the network and then compute the
          centralities

      #IF THE ADDEDGES FUNCTION RETURNS AN EDGE LIST THEN
          UNCOMMENT THIS TWO LINES \dots
      #new_edges_list <- addEdges(network,weight);
      #improved_network <- graph(c(t(new_edges_list)));
      #AND COMMENT THE LINE BELOW
      improved_network <- addEdges(network,weight,direction);
      #plot(improved_network);dev.new(); #to print the multiplied
          networks, uncomment the line.

      if("Closeness" %in% centrality){
        centr_tmp <- Closeness(improved_network);
        centralities_clo <- centralitySum(centr_tmp,weight);
        clo <- c(clo,centralities_clo);
        #means_clo[i] <- mean(centralities_clo); #compute the mean
            Closeness for the simulation #i
      }
      if("alpha.centrality" %in% centrality){
        centr_tmp <- alpha.centrality(improved_network)
        centralities_alpha <- centralitySum(centr_tmp,weight);
        alpha <- c(alpha,centralities_alpha);
        #means_alpha[i] <- mean(centralities_alpha);
      }
      if("Betweenness" %in% centrality){
        centr_tmp <- Betweenness(improved_network);
        centralities_betw <- centralitySum(centr_tmp,weight);
        betw <- c(betw,centralities_betw);
        #means_betw[i] <- mean(centralities_betw);
      }
    }
  }
  else{
    stop("One of the centralities you selected does not exist");
  }
  #results <- c(mean(means_clo),mean(means_betw),mean(means_alpha)
      );
  row <- (length(V(network))*simulations);
  results <- matrix(nrow=row,ncol=3);
  if("Closeness" %in% centrality){results[,1] <- t(clo[-1]);}else{
      results[,1] <- t(array(0,row));}
  if("alpha.centrality" %in% centrality){results[,2] <- t(alpha
      [-1]);}else{results[,2] <- t(array(0,row));}
  if("Betweenness" %in% centrality){results[,3] <- t(betw[-1])}
      else{results[,3] <- t(array(0,row));}
  colnames(results) <- label;
  return(results);
}

createWeights <- function(howmany,min,max){

  val_min<-round(min);
  val_max<-round(max);
  weight<-round(runif(howmany,val_min,val_max)); #generates
      howmany weights within val_max and val_min
  return(weight);
}

centralityPos <- function(n_nodes,weight){ #finding the position
    that correspond to a node and its copies in the centrality
    array

  copies <- weight-1;
  maxval <- max(weight); #find the max multiplication factor (the
      maximal number of copies)
  what_sum <- rep(0,maxval);
  total_pos <- data.frame();
  pos <- 1;
  for(i in copies){ #how many copies a node has?
    what_sum <- rep(0,maxval);
    if(i == 0){ #if the node has 0 copies the centrality is just
        itself
      what_sum[1] <- pos; #in pos[1] will put the position of the
          original node in the centrality array
      pos <- pos + 1;
    }
    else{ #otherwise it should sum the centralities of all the
        copies and the original
      what_sum[1] <- pos; #the pos[1] is always for the real
          position of the original node
      for(j in 1:copies[pos]){ #then for the other copies
        n_nodes <- n_nodes + 1; #copy j
        what_sum[j+1] <- n_nodes; #add the position of the copy j
            to the array of this original node
      }
      pos <- pos + 1; #go for the next node
    }
    total_pos <- rbind(total_pos,what_sum); #add a row, one for
        each node
  }
  return(total_pos); #the final matrix has nrow as the node in the
      original network and ncol has the maximum number of copies a
      node has in the network
}

centralitySum <- function(centrality_tmp,weight){ #sum up the
    centralities (compute the centralities of the multiplied
    network in "centrality_tmp") of each original node and its
    copies

  centrality <- rep(0,length(weight)); #the centrality array has
      the same length of the number of original nodes
  tmp <- centralityPos(length(weight),weight); #computing the
      position of the centralities to sum for each original node

  for(i in 1:nrow(tmp)){ #for each row (a node)
    for(j in 1:ncol(tmp[1,])){ #for each column (the original node
        and its copies)
      pos <- unlist(tmp[i,][j]); #get the position of the
          centrality to add to total sum
      if(length(centrality_tmp[pos])!=0){ #if there actually is a
          centrality to add (there is at least a copy)
        centrality[i] <- centrality[i] + centrality_tmp[pos]; #sum
            the corresponded value
      }
    }
  }
  return(centrality);
}
```

We also developed some utilities for those users that exploit our functions in order to perform the network analysis. The function **eccentricityMining** allows finding those nodes that have, in different networks, different values of eccentricity. As we suggested in the Section 6.2 these nodes could be very interesting and we made it simple to find them.

```r
#####################################################
#a_csv: must be a csv table which contains information about the nodes in the graph and their eccentricity
#another_csv: must be a csv table which contains information about the nodes in the graph and their eccentricity
#centrality: is a string which represents the name of the attribute relative to eccentricity
#flag: does the csv have an header?

#return a list of nodes, as defined by the IGraph library, which shows a difference in the Eccentricity centrality between the two graphs
#####################################################

#this function allows the user to use as input two loaded csv files

eccentricityMiningFromCsv <- function(a_graph, another_graph, centrality, flag=TRUE){
  if(centrality %in% colnames(a_graph) && centrality %in% colnames(another_graph)){
    diff_nodes<-vector();
    for(i in a_graph$name){
      a_pos<-which(a_graph$name==i);#get the position of the node i in a_graph
      another_pos<-which(another_graph$name==i);#the same for another_graph
      a_tmp<-a_graph[,centrality][a_pos];#get its eccentricity
      another_tmp<-another_graph[,centrality][another_pos];#again, for another_graph
      if(length(a_tmp)){#if a_tmp contains something
        if(length(another_tmp)){#and another_tmp too
          if(round(a_tmp,4)!=round(another_tmp,4)){#then store the name of the node
            diff_nodes<-append(i,diff_nodes);
          }
        }
      }
    }
    return(diff_nodes);
  }
  else{
    print("chosen centrality is not an existing column. Check these lists:");
    print(colnames(a_graph));
    print(colnames(another_graph));
  }
}

#this function allows the user to use as input two strings which
    represents the filenames

eccentricityMiningFromFilename <- function(a_csv, another_csv,
    centrality, flag=TRUE){

  a_graph<-read.csv(a_csv,header=flag);
  another_graph<-read.csv(another_csv,header=flag);
  if(centrality %in% colnames(a_graph) && centrality %in% colnames
      (another_graph)){
    diff_nodes<-vector();
    if(file.exists(a_csv) && file.exists(another_csv)){
      for(i in a_graph$name){
        a_pos<-which(a_graph$name==i);#get the position of the
            node i in a_graph
        another_pos<-which(another_graph$name==i);#the same for
            another_graph
        a_tmp<-a_graph[,centrality][a_pos];#get its eccentricity
        another_tmp<-another_graph[,centrality][another_pos];#
            again, for another_graph
        if(length(a_tmp)){#if a_tmp contains something
          if(length(another_tmp)){#and another_tmp too
            if(round(a_tmp,4)!=round(another_tmp,4)){#then store
                the name of the node
              diff_nodes<-append(i,diff_nodes);
            }
          }
        }
      }
      return(diff_nodes);
    }
    else print("one of the files does not exist");
  }
  else{
    print("chosen centrality is not an existing column. Check
        these lists:");
    print(colnames(a_graph));
    print(colnames(another_graph));
  }
}


#######################################################
#a_graph is a graph as defined by the IGraph library
#a_node is a node as defined by the IGraph library

#return a list of paths
#######################################################

findLongestPath <- function(a_graph,a_node){

  paths<-get.all.shortest.paths(a_graph,from=a_node,to=V(a_graph),
      mode="all");
  tmp<-1;
  int_paths<-list();
  #find the length of the longest shortest path generated from a_
      node
  for(i in 1:length(paths$res)){
    l<-length(paths$res[[i]]);
    if(l>tmp){
      tmp<-l;
    }
  }
  #now check if there are more paths which have that length
  j<-1;
  for(i in 1:length(paths$res)){
    if(length(paths$res[[i]])==tmp){
      int_paths[[j]]<-paths$res[[i]];
      j<-j+1;
    }
  }
  return(int_paths);
}

#####################################################
#targets is a file which contains the proteins to search
#paths is a list of paths

#return a list of paths associated to the targets
#####################################################

findTargets <- function(targets, paths){

  l<-1;
  targeted_paths<-list();
  if(file.exists(targets)){
    file<-read.table(targets);
    for(i in 1:length(a_paths)){#how many lists are in paths?
      for(j in 1:length(a_paths[[i]])){#for each list, how many
          objects are stored?
        for(k in 1:dim(file)[1]){#search a protein in all the
            objects
          if(file[k,] %in% a_paths[[i]][[j]]$name){
            targeted_paths[[l]]<-append(targeted_paths,toString(a_
                paths[[i]][[j]]$name));
            l<-l+1;
          }
        }
      }
    }
  }
  else{
    print("the chosen file does not exist");
  }
  return(targeted_paths);
}

#the same but with the specification does_not_contain a set of
    proteins
```

```

findTargetsWithoutSet <- function(contains, not_contains, paths){

  l<-1;
  targeted_paths<-list();
  if(file.exists(targets)){
    contains<-read.table(targets);

    for(i in 1:length(a_paths)){#how many lists are in paths?
      for(j in 1:length(a_paths[[i]])){#for each list, how many
          objects are stored?
        for(k in 1:dim(contains)[1]){#search a protein in all the
            objects
          if(file[k,] %in% a_paths[[i]][[j]]$name){
            targeted_paths[[l]]<-append(targeted_paths,toString(a_
                paths[[i]][[j]]$name));
            l<-l+1;
          }
        }
      }
    }
  }
  else{
    print("the chosen file does not exist");
  }
  return(targeted_paths);
}
```

Chapter 5

Validating experimental findings

5.1 Introduction

The final step of our workflow refers to the validation of the findings that resulted from the analysis of the real, biological networks. This last step allow us giving a statistical foundation to the results. We decided to develop an approach that is based on the comparison of real versus random data. This is a very well known methodology which permits to affirm that the obtained results are not due to chance. If they are statistically relevant they could really have a strong, biological background and potential clinical implications. Using random tests to validate experimental results is a common practice which is applied in several field. The idea of creating such benchmark for the validation was born when I had the opportunity to perform a work, as bioinformatician, that required to validate some biological evidences in the context of the Alzheimer's disease [77]. Here, I designed and developed a simple algorithm, by using R, which allowed simulating a biological experiment using a number of randomly generate in-silico experiments. After extracting the required information it was possible to validate the biological results, to support the hypothesis presented in the paper, by comparing the real versus the random generated data. The final comparison was done by using a statistical test for evaluating the different data distributions. What I learnt was the importance of the experimental validation and how it can be applied in the biological field. So we decided that a tool that allows creating random experiments was needed also in the field of biological network analysis and we decided to fill the gap between random networks and Cytoscape.

5.2 Random network analysis

Validating biological networks is a process which is not straightforward in the sense that there are several, different possible paths to follow and there are not guidelines which define what is the best way to get the results. In this sense there are several different random network models that were developed over time and has different mathematical and topological properties. The first model refers to the researches of Erdős & Rényi [117]. They defined a graph whose main characteristic is a complete random topology where some nodes are linked by some randomly distributed edges. Then we have the Watts & Strogatz [118] model. This is the first random graph model that shows the small-world property. A more modern model refers to Barabási & Albert [119]. It allows the creation of scale-free networks. These are networks with a hub-based structure, where an hub is a node with a high number of neighbours. Finally there are also Community Affiliation Graphs [120] that are designed to describe social networks between people and lattices that are grid of nodes. Currently, there is also the new model we designed, called the Multiplication model that allows integrating all the existing models by randomising the weight that are assigned to each node.

This amount of possibilities make it difficult to choose which model better fits a specific biological experiment. Each model has its own characteristics and, probably, the best way to create a random experiment is by trying to fit the characteristics of our network and choose the model which recreates the most similar behaviour. In this sense a very detailed analysis should be carried out over a network aiming at describing its characteristics. Is our network scale-free? Has it an high clustering coefficient? Is it showing small-world properties? These questions should drive the decision for the best random model to use. Indeed, it is not obvious which way one should choose and the multiplication model make it more complex since there are no guidelines that may suggest how to proceed. Also, the validation step is somehow not straightforward. It is possible to validate different characteristics of the network. For instance, it is possible to compare the Degree distribution of the real and the random networks or, generally speaking, a specific centrality. Also, it is possible to compare the clustering coefficient, the diameter and other networks parameters. But, since we are not interested in comparing the topological characteristics one by one, we decided to compare the classification results obtained from the real data with the same results from the randomised data. The idea is to verify if the classifier is able to separate the healthy and the unhealthy class even in a randomised set-up. What we expected is that the algorithm should not be able to discriminate between two random classes since the data come from the same, random distribution. In other words, the classifier Accuracy should be around 50% since the algorithm should choose at random one of the two classes, leading to a situation in which it is throwing a coin. To achieve this goal, we decided to follow an experimental-based approach in order to model the random networks. We performed several experiments in order to test which is the best way to use random models. This means that we tried to mimic as much as possible the real experiments using different, meaningful, approaches. Also we exploited the same pipeline, i.e. the same algorithms, we used to perform the computational analysis related to the multiplication model. We created different sets of random experiments and we tested the pipeline for each one of them. Following the experimental set-up allowed us creating a set of random datasets that can be considered as plausible biological datasets.

Creating random networks

We designed three different experiments based on the random networks. We decided to investigate the properties of only three, out of four, datasets that are respectively MI, BCLL and PET. AM dataset was left out since each network, both for the healthy and the unhealthy subjects, has a variable number of nodes making it very difficult to randomise the whole dataset. Hence, we created a set of random networks for each of the outlined datasets obtaining a total number of nine random datasets. The three experiments was designed in order to have an increasing level of randomisation. In this sense, the first experiment has a very low level of randomisation, the second one introduces another level of randomisation and, finally, the third one shows the highest level of variations between the original and the random data.

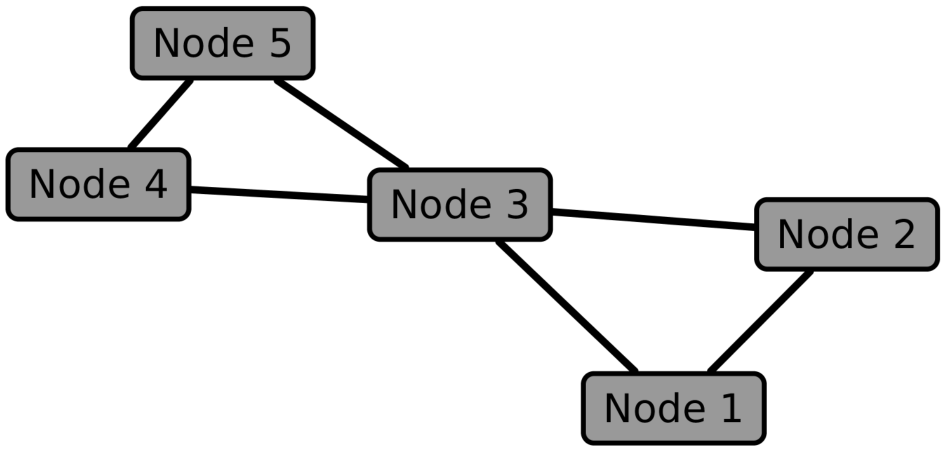


Figure 5.1: **An example of network**, that represents the interactions between a set of proteins. Starting from this network we proceeded in creating some random models, by applying different layer of randomisation. The first experiment used this network and applied some random weights in order to multiply its topology.

The first experiment we designed required us to find the range of variation in which the nodes experimental values vary. We found the minimum and maximum number of copies of node each class of network has. For instance, in the MI dataset we found four ranges even though the classes are two, i.e. healthy and unhealthy. We had to compute four ranges since the unhealthy class is divided in different treatments, i.e. F, H, U and N. To better fit the experimental background, we found a range for each treatment. Once the ranges were found we created an equal number of networks with randomly distributed nodes weights, i.e. we generated a random array of weights for each network and we multiplied it. The topology of the network in terms of nodes, edges and their connections remained unchanged but the weights are now randomly distributed.

The second experiment we designed, aimed at adding an additional layer of randomisation. The networks, each network for each class of each dataset, were randomised by using a Degree preserving shuffling algorithm and then multiplied using some randomly computed weights. The Degree preserving algorithm allowed randomising a network while keeping the Degree of the nodes fixed. In this sense, if the Node 1 has five neighbours before the randomisation, it will have five neighbours after the randomisation but these neighbours will, usually, be different. It is important to note that the five neighbours before and after the

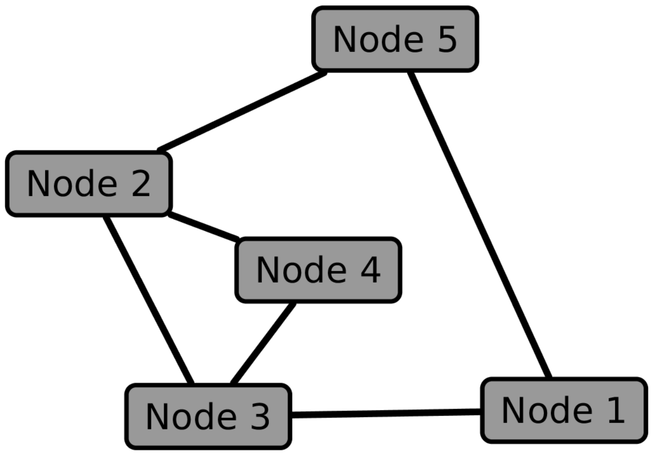


Figure 5.2: **An example of random network**. This is the same toy network that has been randomised using the Degree Preserving shuffling algorithms. It allowed obtaining a network with a fixed Degree but with randomly assigned neighbours. This means that, for instance, the Node 3 have had a Degree equal to 3 in the original network and it have a Degree equal to 3 in the randomised network. The same happens for all the other nodes. Over this network an additional multiplication, by using random weights, was done.

randomisation could be different, partially overlapping or the same depending on the network. Clearly networks with a reduced number of nodes are more difficult to randomise using an edge shuffling algorithm. Finally, the random weights were computed by using the same approach that takes advantage from the ranges. This further randomisation step allowed to test the biological information that arise from the actual biological interactions.

The third, and the last, experiment we designed and carried out, required the creation of a set of random networks by using a randomising algorithm and then by multiplying these networks by a set of random weighting array. What we did was to randomise each network, for each dataset, we analysed by using the Erdős–Rényi model randomisation. This model allows creating random network which have the same number of nodes and edges as the starting network, but these nodes and edges are randomly placed without considering any prior or posterior information or design. Then, each node was multiplied by using the usual array or random weights computed in the ranges for each class.

Results

The three different experiments allowed to generate different datasets of random networks which aimed at modelling the same experimental set-up we were investigating. Then we applied the same pipeline that was used over the real, biological data in order to extract the same topological features. Finally the same classification step was performed, for each of the random sets, in order to compare the real and the random classification performances. By applying the same methodology we made the results from the biological and random networks comparable. The results are showing very different classification outcomes that are very interesting

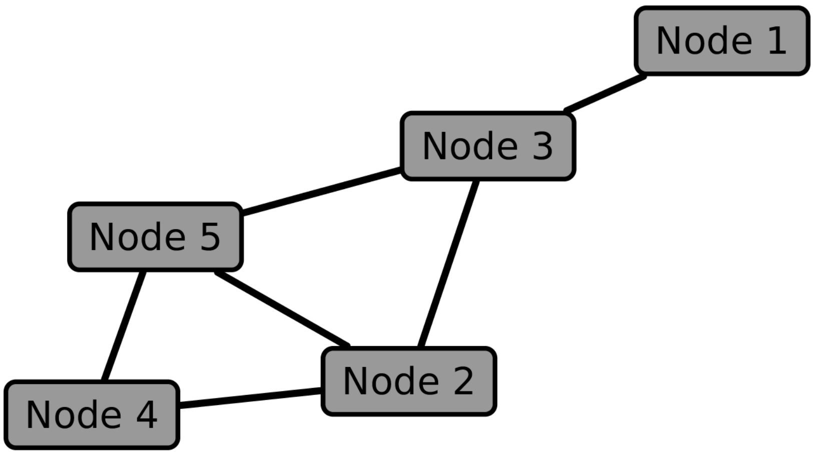


Figure 5.3: **An example of random network**. This is the same toy network that has been randomised using the Erdős–Rényi model. It allowed obtaining a network with a fixed number of nodes and edges, randomly distributed. This means that, starting from an original network with five nodes and six edges, the algorithm created a new network with five nodes and six edges randomly assigned. Over this network an additional multiplication, by using random weights, was done.

and open new challenges. One of the results, i.e. the one from the PET analysis, we obtained was expected. The two others give some interesting insights about the nature of the networks and the features we used for describing them. Generally speaking, it is possible to affirm that for both the MI and BCLL random datasets the classifier is able to find patterns and separate the classes with a very high level of Accuracy. On the other hand the PET random dataset shows very poor classification Accuracy.

Pancreatic Endocrine Tumours The results (see Fig. 5.4) obtained from the PET random datasets show how the classification task resulted in a lower rate of performance, in terms of classification ability, for all the three random datasets, when compared to the biological dataset. This result is expected because of the structure of the healthy and the unhealthy networks. Both these networks were designed by using the same set of nodes and resulted in a network that is the same for both classes, i.e. healthy and unhealthy. What happened while classifying was that when multiplying and shuffling, using random information, we are making the learning step more difficult. This means, in classification terms, that the classifier is not able to trace a clear threshold to decide to which class a sample belong. Since the two networks are the same and since none of them, 71 in total, contain a real, experimental and biological information, then the algorithm seems to be not able to assign the sample to its class. Indeed, the Accuracy remains below 60% and, more generally, around 50% which is the same, as already mentioned, as throwing a coin. This is even more clear when looking at the Accuracies. The lowest score where obtained using the completely random network. This means that, keeping the Degree fixed hence considering some original, topological characteristics, is fundamental to enhance the classifier performances. When the information about the origina topology is completely lost, i.e. in the random set-up, then the Accuracy decrease reaching the expected percentage, i.e. 50% or so. From a network perspective, a topology that is randomly weighted lose the information that is derived from the biochemical activity that allows potentiating it. Indeed, the algorithm is no more able to find recurring patterns since the value we assigned are completely random. This result is a very good example that support the hypothesis we wanted to validate.

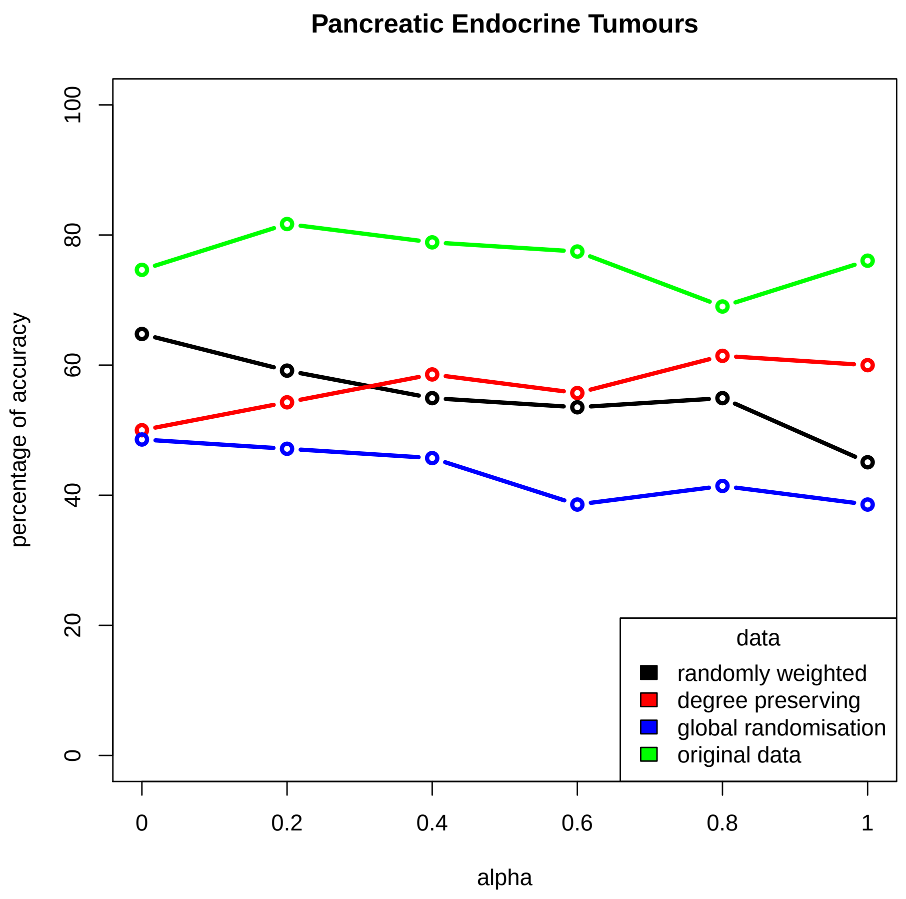


Figure 5.4: **Showing the percentages of Accuracy of the original data versus the random experiments, for the Pancreatic Endocrine Tumours dataset.** On the X axis the different alpha that were used to compute different accuracies, on the y-axis it is possible to appreciate the percentage of Accuracy that each dataset scored at different alpha level.

B-Cell Lymphocytic Leukemia The results obtained from the BCLL dataset (see Fig. 5.5) shows a more linear behaviour. The results shows how the original dataset scored the lowest values of Accuracy. This result is totally unexpected since, in principle, random data should not be useful for separating two random classes. But, a second possible explanation exists. Since the two average networks

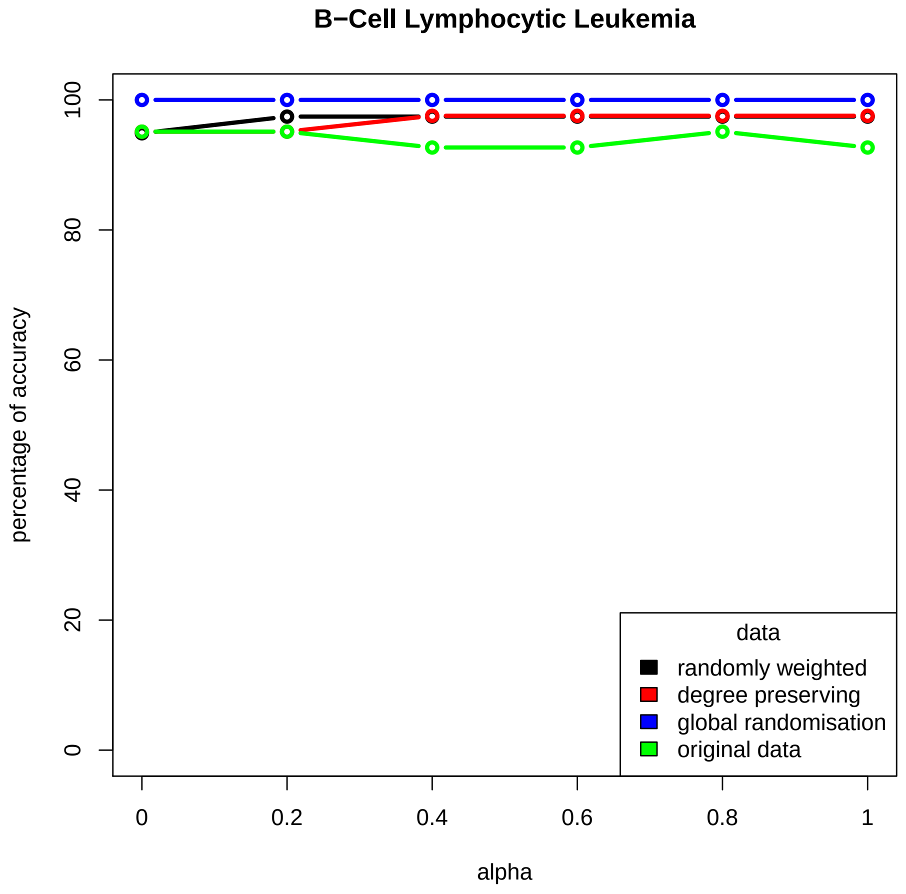


Figure 5.5: **Showing the percentages of Accuracy of the original data versus the random experiments, for the B-Cell Lymphocytic Leukemia dataset**. On the X axis the different alpha that were used to compute different accuracies, on the y-axis it is possible to appreciate the percentage of Accuracy that each dataset scored at different alpha level.

are different, it is possible that, by randomising the weights, the differences between these networks are, somehow, highlighted. Hence, a classifier will gain in terms of classification performances since the task is easier, if compared to its biological version where differences exist but are, somehow, more subtle. Obviously this is only an hypothesis but, surely, when classifying two networks whose structure is different, but that share some common proteins, the algorithm takes great advantage from the differences between the two networks.

Myocardial Infarction The results obtained with the MI dataset (see Fig. 5.6) show another very interesting, altough unexpected, situation. As already said, the original data allowed separating the two classes, i.e. healthy versus unhealthy, with an Accuracy equal to one. What it is possible to appreciate from the results in the plot is that, when we randomise the weights and the whole topology, we are losing some useful information and the learning algorithm is somehow

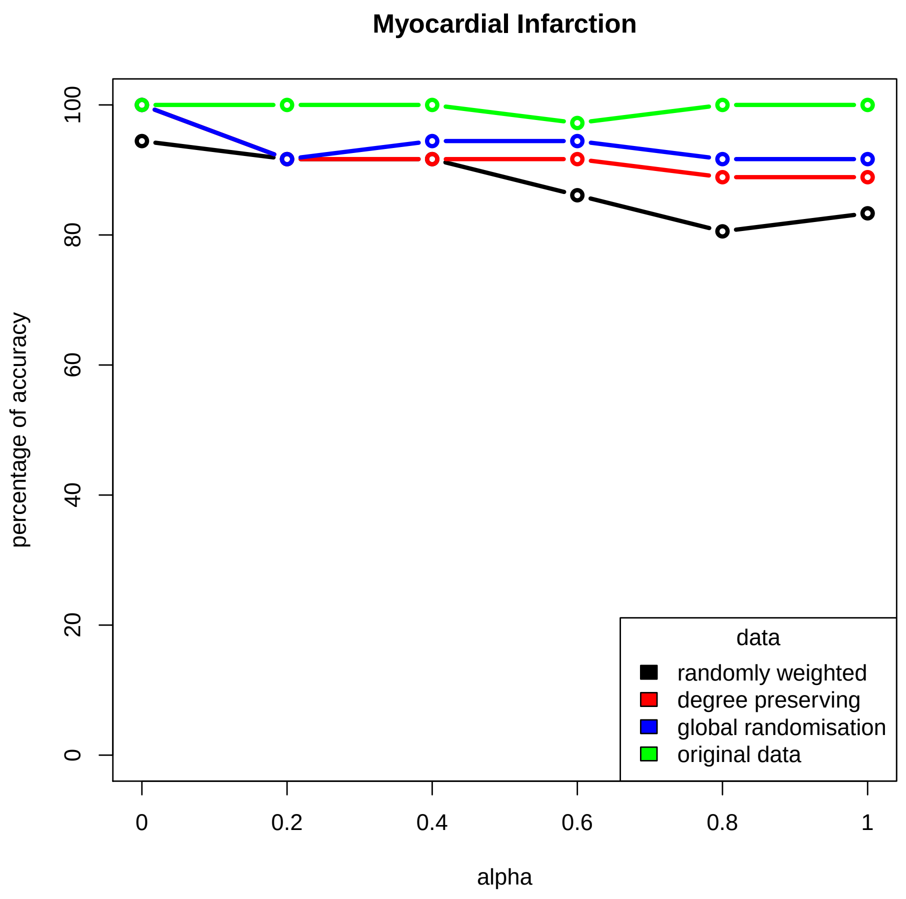


Figure 5.6: **Showing the percentages of Accuracy of the original data versus the random experiments, for the Myocardial Infarction dataset.** On the X axis the different alpha that were used to compute different accuracies, on the y-axis it is possible to appreciate the percentage of Accuracy that each dataset scored at different alpha level.

mislead and its Accuracy has a limited drop. Surprisingly, the worst accuracies were obtained with the dataset which was randomised by computing only random weights. On the other hand, the other two randomised datasets, that should contain a bigger amount of noise, show higher levels of Accuracy.

Both the MI and the BCLL random datasets are suggesting that the experiments we designed may be not the proper way to address our question. In this sense, randomising a network may require some different steps. What we did was, by starting from the average networks for each class of subjects, to *i)* randomise its weights or *ii)* randomise, at different level, its topology and weights. In both cases, for the MI and the BCLL, it seems that these two directions do not represent the best approach for comparing real versus random data. The PET dataset showed the expected results but, as already pointed out, this is probably due to the fact that the two average networks, i.e. G1 and G2, are represented by the same network.

It is possible to define few new lines of research that aim at addressing this question. The first, possible, experiment will generate a dataset that consider the same network multiplied with two different sets of weights, one for the healthy and one for the unhealthy class. The second possibility concerns the generation of two distinct network, i.e. the same used in the original experiments, but a single set of weights that is applied to both the networks. By doing so we are investigating the properties of the weights, in the first case, and the properties of the networks, in the second case. The last option refer to the fact that performing only one experiment, i.e. we randomised only once and we classified only a dataset for each random model, is not enough to evaluate a classifier over a set of random networks. Probably, by generating more random datasets and performing more classification will give us better results.

Chapter 6

Discussion and future works

6.1 Considerations about the topological and computational analyses

We have defined a new approach to analyse complex biological systems, exploiting experimental, quantitative datasets in the context of static topological analysis. The two main goals were:

- the definition of a new modelling technique for integrating graphs and quantitative data;
- the investigation of several topological indexes that could be used as potential biological markers;
- the definition of a computational pipeline that allows performing the same analysis we carried out;
- the development of some tools that are required to perform the analysis;
- to give some biological directions in order to demonstrate how the model could be useful.

Consistently, four biological datasets, obtained using high-throughput technologies, were modelled as potentiated networks. Then, all the proteins in the networks were categorised by using nine different topological features, well known in biological network analysis. The categorisation allowed the investigation of these features and also the validation of their biological relevance in the context of the new, multiplied models. In particular, the classification algorithm revealed that topologies belonging to the same class share recurring patterns, which were inferred by using specific topological descriptors. Healthy and unhealthy topologies were correctly classified, thus showing that the proposed multiplied model is able to mimic the nature of a specific process. Moreover it appeared that some of the features are more useful than others during the learning step.

A very interesting aspect, concerning the fact that the features that were used are based on two different network characteristics, emerged. Indeed, a group of centralities focuses on the length of the shortest paths, while the second set focuses on the number of shortest paths related to a specific node. The node multiplication technique, as the theorem proved, only affects the second aspect, i.e. the number of paths, since it involves the addition of some nodes and edges leading to an increased number of shortest paths. Notably, in the PET dataset the two more informative features are Betweenness, Bridging and Stress. To contextualise this result it is important to note that these centralities are computed using the number of shortest paths passing through a node. Also, the network used for describing the two classes is the same, hence the distance between the nodes remain the same. These two facts are two faces of the same coin and are strongly dependent from each other. Indeed, the two networks differ only in terms of multiplication values. For this reason the classifier take great adavantage, in terms of performances, from the augmented number of short paths. In other words, the classifier takes great advantage from the information contained in the shortest path based centralities to create its discriminative threshold. The other centalities, in contrast, do not contain a comparable amount of information, since the paths do no change, thus are not selected as best, i.e. most informative, features.

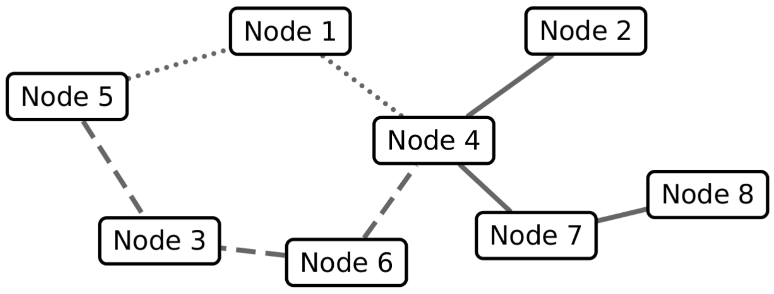


Figure 6.1: **Showing a network which has a variable Eccentricity depending on the presence of a node**. The Node 1 allows the Node 5 to be connected to the Node 4 in two steps, i.e. by the dotted lines. If the Node 1 is missing in a subject, then it will not result in the network modelling that subject. In this case the Node 5 should pass through the Node 3 and the Node 6, i.e. the dashed lines, to reach the Node 4. The presence or absence of the Node 1, in this example could be the difference between the two classes of subjects.

On the other hand, also shortest paths based centralities should be considered very interesting indexes. In particular, we are referring to those proteins that are present in a class of networks and not in the other class (see Fig. 6.1). Indeed, centralities like Closeness, Radiality and Eccentricity are directly affected since specific shortest paths become peculiar of a class. This is a very interesting aspect, in the sense that a protein which is systematically missing in one class, becomes a very interesting target. This fact allows the creation of novel shortest paths, making this protein a marker for a class, in terms of topological characteristics of the network. In this sense, it becomes fundamental to understand, for each protein which is present in a class and not present in the other class, why it is missing and, more interestingly, which paths are affected. Indeed, different physio-pathological conditions activate different molecules that are present in distinct amounts. By using a topology that is able to emphasise such specificity, it becomes possible to investigate particular experimental set-ups. It will be very interesting to develop a computational methodology that allows the automated discovery of missing proteins and, also, the paths that are influenced by such events. Missing proteins are not informative because they are not present in a class. They become informative, from a network perspective, in the sense that the new paths that are generated to supply to the limitation of a missing protein, may suggest novel potential molecular targets and pathways. These facts surely affect the learning step, since the new centrality indexes that the nodes have, are surely affected by the new role the node play when a protein is present or not.

Also, it resulted that specific features have an higher score, i.e. they have a better ranking, in terms of best features, more frequently than others. For instance the Eigenvector appears to be relevant in the AM, MI and BCLL datasets. This centrality refers to nodes that have a relevant neighbourhood thus, the central the neighbours the higher the Eigenvector of a specific node will be. In the BCLL there are two peaks that correspond to Eccentricity and Eigenvector. The presence of the Eccentricity could be explained by looking at the structural, i.e. topological, differences between the two networks. Here, we have two average networks, one for each class, that are different in terms of nodes interactions but share some common proteins. These differences influence the distances between the nodes. Also the shared proteins has a very high rate of variation in terms of neighbourhood. Not only the nodes are different but also the number of copies they have is different. These two facts greatly influence the Eccentricity that becomes a very informative index. In other words, the same node, in the two networks, may have a very different neighbourhood depending on the multiplication values. The range in which the multiplication values vary may be different, i.e. it is possible to obtain a specific range for healthy and for unhealthy subjects. As we said the Eccentricity for the multiplied topology sums up the contribution of each copy of a node. So, the number of copies a node has strongly influence the Eccentricity. Also the number of nodes a node reach, i.e. through short paths, is an important factor that influences the Eccentricity. Hence, both these values, i.e. the number of copies and the number or reached nodes, become a very important information that is reflected, at the network level, by the Eccentricity. Also, the Eigenvector is strongly influenced by the number of nodes and its variation, i.e. high number or low number. The Eigenvector is a complex centrality and it appeared as one of the best features in the BCLL, MI and AM datasets. Here, as already highlighted, the two average networks are different. Also, they differ in terms of relevance of the shared proteins, i.e. the number of copies they, and their neighbours, have. Indeed, this happens because of the multiplication step, since the Eigenvector is a sort of weighted Degree since it considers the Degree of the neighbours too. The multiplication affects the number of nodes, hence the shared proteins inherit very different roles depending on the class to which they belong.

This means that their neighbourhood largely varies between the two classes, suggesting a different level of involvement for the analysed proteins, in the global topology of the networks. It is important to note that the Eigenvector is surely influenced by the number of copies a node has since they are considered their neighbours too. As we mentioned for the Degree (see Section 3.2), the number of copies is considered as a contribution to the Degree a node has. Hence, nodes with a high Eigenvector could be nodes that have:

- a high number of copies;
- neighbours with a high number of copies;
- a high number of neighbours.

Obviously the three possibilities may occur at the same time. Eigenvector is able to take into consideration all these aspectes and nodes with a high Eigenvector centrality should be carefully considered.

In the future, the role of the centralities we are using should be clarified in terms of biological meaning, see Section 3.2 since the current interpretation is too general to be actually helpful. Using other metrics we didn't exploited will be surely interesting. Also contextualising the results is fundamental to understand what a centrality is telling us. Finally, a deep knowledge of the model is basic in order to consider all the information that come from this kind of analysis. Another future improvement for our model and its investigation must focus on the mathematical definition of the centralities we are currently using. We are dealing with a new model and we defined some centralities that take into account the existence of new nodes, i.e. the copies, but this may be good or not. A further investigation should consider the fact that, for instance, the Degree should not consider the copies of the nodes but only the actual neighbouts of a node. Also, using other centralities could be meaningful, in order to obtain more information. It is hard to define which centralities better describe the properties of a node or a network. It is possible to investigate some characteristics rather than others but, to obtain a comprehensive description of a process, different classes of centralities, i.e. shortest path based and distance based, may be a good choice.

6.2 Considerations about the biological usefulness of our approach

The datasets we analysed, that are very different in terms of biological context and raw data, show some similarities when modelled as networks. It is important to note that all the networks we built are, at some level, similar. This similarity is due to the fact that they were constructed starting from the human proteome, hence its underlying structure is found in all the subnetworks. But, even though, the nodes multiplication, which is the crucial step allowing the personalisation of a network, greatly affects the results. Indeed, multiplying the nodes allows generating a great number of networks that become the tool that exploit the personalised medicine approach.

Obtaining a lot of networks describing a class of subjects permits a pattern recognition approach whose predictive ability is directly influenced by the number of networks we are able to construct. Extracting a lot of topological information from PPI networks allow the classifier, as we showed and discussed, to take great advantage of these numbers in order to identify the specific class of a network making the multiplication step fundamental for this task. The study we carried out demonstrated that the proposed model is reliable and precise and, as we mentioned, has very strong applications in the field of personalised medicine. In this sense, it is easy to foresee that a multiplied model can easily be used to model the whole proteome, once and if the data will be available, at the individual level. The approach also helps us investigating -omics data from a system perspective, enhancing the predictive ability of the raw data, e.g. proteomics or genomics, by using specific features, i.e. the topological centralities, extracted from the networks. It would be very interesting to create a healthy interactome that could be used as a benchmark for validating different pathological interactomes. Moreover, in a biomedical context, the investigation of those molecules that potentially affect the length of the shortest paths along the different classes, as already mentioned in Section6.1, could give interesting insights and suggest new pharmacological targets. In these scenarios novel algorithms are required and, from a computational perspective, there is the need for a benchmark which can be used as a validation layer. In this sense the tool we developed, capable of creating random networks, which aims at simulating real data PPI networks, becomes necessary to compare real data PPIs and randomised experiments to support this kind of analysis.

From the biological point of view the investigation is not yet started and it is very difficult to define the boundaries of our work in these terms. Indeed, our contribution does not have a biogical focus and the results we obtained, from a biological perspective, should be considered as suggestions for further, in-depth investigation. On the other hand, the identification of the most interesting molecules through the classification algorithms, is one of the results we were expecting. There are a wealth of bioinformatical analysis that are designed to aid the experimental side of the biomedical research. Our pipeline aims at enhancing these practices in a personalised medicine context, presenting a novel model for construcing precise networks. At this stage it is not possible to discuss the relevance of the biological findings since we don't have an experimental validation.

What we actually did was to show how the pipeline allow modelling -omics data. We suggested some targets that we consider of sure interest since their biological role and relevance emerged from a complex analysis. Finally, we showed how to obtain these molecular targets through our analysis. All these steps do not concern the laboratory practice but, as already highlighted, become fundamental in order to perform appropriate and very specific laboratory analysis. What follows is a targeted, experimental approach. It will be interesting to evaluate the role of the proposed molecules through a preliminary experimental quantification in order to assess whether they are present in the pathological cell lines or not. These results should be compared with control cell lines in order to verify if some differences exist and if they are statistically relevant. Then, investigating the behaviour of their first interactors could be meaningful in order to assess the role these molecules are playing in regulating other proteins. Indeed, a question that need to be addressed refers to the effect of the molecules we are suggesting with respect to other proteins, e.g. their first neighbours, in a pathological setup. What happens if the molecule is, somehow, silenced, i.e. not activated? What happens if it is overexpressed? As we know, proteins work togheter in order to carry out complex biological tasks. Identifying these tasks in a complex network could help reducing the complexity and the number of molecules under investigation, while keeping an eye on the proteins that emerged from the computational analysis. Also, as we suggested, the Eccentricity centrality highlighted that missing proteins, i.e. underexpressed or silenced proteins, could play a very interesting role. In this sense, an experimental approach that consider not only a target but also a pathway, will be surely of interest.

Another important aspect of the biological data we investigated, that should be considered, refer to their temporal and spatial characteristics. Protein composition varies in time and in different families of cells. A protein may be present in a pancreatic cell but not in a liver cell. These facts do not affect the analysis we are proposing since, when dealing with a pathology, it is fundamental to compare the same kind of cells. This means that, if we want to investigate breast cancer, we will not use healthy glial cells as control cell lines. This is a basic consideration but it is important to highlight the fact that it is not always possible, or easy, to deal with specific cell lines because some can be difficult to isolate or to grow in an artificial environment. Comparing meaningful samples is fundamental to obtain useful results. The same considerations apply to the temporal aspect. Moreover, time requires a further consideration. When we build a network what we are doing is basically to construct a steady model that represent a specific time point. The exact time point is represented by the data, that comes from a biological sample. Obtaining a biological sample in the morning may be different from obtaining it at noon or in the evening. This fact becomes particularly relevant when we are dealing with clinical samples that are obtained from real subjects. The temporal aspect is very interesting and challenging since it can be used as an advantage. It is possible to model different time points, generating a sort of semi-dynamical network-based approach but it require a lot of data and it is not the goal of this thesis. Summarising, the source of the data is fundamental, in particular in a computational analysis like the one we are presenting that aims at comparing different classes of the same objects.

In the future, enriching the networks by using a layer of biological information giving insights about the role each protein plays, could be surely very informative. Considering the experimental numerical values we used for the multiplication step in an enriched biological context will give a different perspective. It may potentially helps in retrieving new, interesting evidences about the pathological mechanisms. Also, the approach we are proposing become more and more important if we consider that we have access to both healthy and unhealthy networks. In the future, it will surely be interesting to compare different set of unhealthy subjects. As pointed out, we have two datasets, i.e. AM and MI, where the unhealthy classes are composed by different kinds of unhealhy subjects. By applying our workflow over these data it may be possible to find specific molecules that directly depend on the kind of pathology and its, very specific, mechanisms. It could be very interesting to investigate the properties of a specific pathology since our approach points out the differences that occur in two, or more, different samples.

These considerations are particularly interesting if we contextualise the work in the field of personalised medicine. As we reported (see Section 1.2, the personalised approach requires very fine representations that allow depicting a single individual and its molecular characteristics. This could be done by exploiting the approach we proposed:

- it takes strong advantage from the increasing usage of high throughput analysis that allow obtaining personal data in a very short time and at decreasing costs;
- it does not require expensive technologies and can be easily extended to a lot of different biological data; also it theoretically works for any kind of pathology;
- the data that are necessary to construct the network are constantly increasing and are more and more reliable; so, it is possible to foresee that the methodology will increase its predictive power.

On the other hand, our technique has some drawbacks. The pipeline requires different softwares that should be used. We took advante of Cytoscape, Matlab and R in order to perform the whole analysis and this may be not accessible to a researcher that is not fluent in programming. Also, the workflow is quite time consuming, since the steps that regarded the construction of the network were manually curated. The higher the number of the networks, the longer the workflow. Furthermore, our pipeline requires that the dataset consist of, al least, two classes in order to perform the comparison and, ideally, they should have the same number of samples.

Algorithmic drawbacks could be tackled, in the future, by developing specific algorithms that are able to construct meaningful networks starting from a probe and a well defined set of interactions. Ideally, these algorithms will be able to cross the information from different databases in order to obtain a reliable model in a very short time. Also, the pattern recognition step should be integrated into both the Cytoscape side and our R library in order to make it availabe to the broader audience, even to those researchers who don't know what a classifier is.

Data drawbacks are, and probably will be, the most challenging issues since the model we propose greatly depends from the numerical attributes that could be obtained only through expensive -omics technologies. The advent of new, cheaper technologies will surely enhance the quality and the amount of the available data but this depends on multiple factors that it is not possible to foresee. We believe that the whole pipeline could be automatised. The bottleneck is, as mentioned, directly dependent from the availability of clinical data.

6.3 Considerations about the random experiments

The app we developed, i.e. NetworkRandomizer, allows Cytoscape users in order to create sets of random network which aim at validating real data networks. This app offers some, different models that were proposed over time from different, expert researchers and that aimed at addressing some specific issues that they were facing. For instance the Erdős-Rényi model was defined in 1959, which means that the Systems Biology, as we intend it today, was very far from being conceived [121]. This model aimed at creating networks by adding edges between randomly chosen couples of nodes. Then some other properties of complex network were uncovered, e.g. the small world property, and new models were developed like, for instance the Watts-Strogatz model. This fact is somehow confusing when it is time to decide which model better fits our requirements.

To validate our findings we constructed different datasets of networks, by using three different levels of randomisation. This was done by applying different techniques but still it is possible that the methodologies we used could be improved or proven to be not suitable. What we was aiming at, was to find relevant differences between the biological data and the random data. We wanted to achieve this result using a set of networks that strictly mimic the biological experimental design. In other words, we wanted to introduce the lower level of random noise that was enough to show, by using the same learning algorithm we used for the analysis, an acceptable difference in terms of classification Accuracy. What we actually found out was that, no matter how much we randomise a network, the random data seemed to be not suitable, for this task, in all the experiments we carried out. Indeed, the random data show very different classification results, which could be interpreted by saying that, probably, we are using a wrong approach.

So, the approach we used for the creation of random dataset requires some adjustments. The first problem that arose while designing the randomisation step was related to the fact that the AM dataset, which have a very high level of variations along the networks, does not have two average networks. In this context it is not yet clear how we are supposed to proceed. Should we create a set of random networks with random weights? Should we create a random network that only considers the frame, i.e. the core structure, that is shared between all the networks and lose the information that come from the first neighbours? We still don't have an answer to these questions, and this is a point that should be deeply investigated, since biological datasets are very far from being homogeneous.

Also, the approach we used for randomising the other dataset could be certainly improved. Is that correct to use a random network for the healthy subjects and one for the unhealthy subjects? Could it more meaningful to use a single random network and weight it by using some random values? All these questions arose when the results showed how in two datasets the classification task scored very high accuracies. This is an unexpected result that poses some very interesting questions but also that supports the results we obtained in the real, experimental set-up. On the other hand, the PET random results demonstrated how, by using the same network for both classes, the randomisation works and the classifier is no more able to separate the two classes. Indeed, the classification Accuracy is around the 50% which means quite random. Also, these results highlight the fact that managing random data in the context of biological network is not as easy as it initially seemed. This means that it will be necessary to reconsider the approaches we proposed and evaluate the possibility to define some new, random model which better fits the original experimental design.

Finally, and more interestingly, the multiplied model we propose, aiming at modelling experimental values, is not studied and it may hide some interesting properties that may be peculiar. The use of the cliques to model over-represented proteins may result in some emergent properties we have not yet uncovered. So, maybe, a new algorithm for generating this kind of networks may be defined, giving rise to new possibilities for what it concern the randomisation of real, experimental networks.

6.4 Conclusion and further considerations

The work we carried out has a multi-disciplinary nature and includes very different fields ranging from the information technologies to the applied clinical practice. The model and the computational analysis we presented, aim at providing some ideas and guidelines that may potentially improve the investigation of complex biological process. We designed the whole work in order to aid researchers in the field of personalised medicine. Indeed, we showed four different pathological contexts in which our analysis gave interesting results and little biological insights. Managing mathematics, computer science, biology and pathology is surely challenging. We provide some tools that allow to interface these different worlds. Also, our approach will hopefully open new perspectives and aims at giving a new point of view, i.e. a system-based point of view, that is not yet fully investigated and understood.

One of the main issues with Systems Biology reflect the difficulty of explaining to those people who are not working in the field why and how it is a very powerful tool and how it could potentially affect the way they usually manage a laboratory and the way they carry out their everyday research. In-silico predictions allow reducing time and cost for a very wide range of wet-lab experiments that are currently carried out. Moreover, it can significantly enhance our predictive power through the development of very detailed models for the diseases and, generally speaking, for all the complex processes that take place in every organism and, also, between different organisms. The approach we proposed has very interesting applications in the field of personalised medicine but, since it involves complex subjects like machine learning, the motivations that induced us at using a pattern recognition algorithm to investigate a set of topological centralities could be misunderstood.

A very important future development, in this sense, must aim at overcoming the communication limitations. How to present these kinds of works to medical doctors, biologists, biotechnologist, laboratory technicians is very challenging but also explaining what a molecular mechanism is to programmers and software developers requires big efforts. Ideally, these two worlds should work together in order to face the new challenges that are arising from a complexity that cannot be tamed in a wet-lab, by using a Petri dish. Finding a common language will be more and more necessary since the collaborations between each other, considered as a big, interacting system should be exploited.

Bibliography

[1] T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life: Systems biology. *Annual Review of Genomics and Human Genetics*, 2(1):343–372, 2001. PMID: 11701654.

[2] Barabasi, A.L., Oltvai, Z.N. Network biology: understanding the cell's functional organization. *Nat. Rev. Genetics*, 11:101–113, February 2004.

[3] D. Noble. *The Music of Life: Biology Beyond Genes*. Oxford University Press, USA, April 2008.

[4] H.L. Allen, K. Estrada, G. Lettre, S.I. Berndt, M.N. Weedon, F. Rivadeneira, C.J. Willer, A.U. Jackson, S. Vedantam, S. Raychaudhuri, et al. Hundreds of variants clustered in genomic loci and biological pathways affect human height. *Nature*, 467(7317):832–838, 2010.

[5] J. Li, Z. Zhang, J. Rosenzweig, Y.Y. Wang, and D.W. Chan. Proteomics and bioinformatics approaches for identification of serum biomarkers to detect breast cancer. *Clinical chemistry*, 48(8):1296–1304, 2002.

[6] W.L. Delano. The PyMOL Molecular Graphics System. http://www.pymol.org, 2002.

[7] N. Guex and M. C. Peitsch. SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling. *Electrophoresis*, 18(15):2714–2723, December 1997.

[8] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[9] B. Chapman and J. Chang. Biopython: Python tools for computational biology. *SIGBIO Newsl.*, 20(2):15–19, August 2000.

[10] A. Prlić, A. Yates, S.E. Bliven, P.W. Rose, J. Jacobsen, P.V. Troshin, M. Chapman, J. Gao, C.H. Koh, S. Foisy, R. Holland, G. Rimša, M.L. Heuer, H. Brandstätter–Müller, P.E. Bourne, and S. Willis. Biojava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, 28(20):2693–2695, 2012.

[11] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

[12] J.D. Thompson, D.G. Higgins, and T.J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[13] The UniProt Consortium. Activities at the Universal Protein Resource (UniProt). *Nucleic Acids Research*, 42(D1):D191–D198, January 2014.

[14] M. Ashburner, C.A. Ball, J.A. Blake, D. Botstein, H. Butler, J.M. Cherry, A.P. Davis, K. Dolinski, S.S. Dwight, J.T. Eppig, M.A. Harris, D.P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J.C. Matese, J.E. Richardson, M. Ringwald, G.M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*, 25(1):25–29, May 2000.

[15] M. Kanehisa, S. Goto, Y. Sato, M. Kawashima, M. Furumichi, and M. Tanabe. Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic acids research*, 42(Database issue):D199–D205, January 2014.

[16] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, January 2000.

[17] U Alon. Biological networks: the tinkerer as an engineer. *Science*, 301(5641):1866–1867, 2003.

[18] C. Pastrello, E. Pasini, M. Kotlyar, D. Otasek, S. Wong, W. Sangrar, S. Rahmati, and I. Jurisica. Integration, visualization and analysis of human interactome. *Biochemical and biophysical research communications*, 445(4):757–773, 2014.

[19] P.J.E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic petri nets. *Proceedings of the National Academy of Sciences*, 95(12):6750–6755, 1998.

[20] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108 – 143, 2000.

[21] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 459–470, 2001.

[22] A.L. Barabási, N. Gulbahce, and J. Loscalzo. Network medicine: a network-based approach to human disease. *Nature Reviews Genetics*, 12(1):56–68, 2011.

[23] A.L. Barabási, N. Gulbahce, and J. Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews. Genetics*, 12(1):56–68, January 2011.

[24] L. Hood and S.H. Friend. Predictive, personalized, preventive, participatory (p4) cancer medicine. *Nature Reviews Clinical Oncology*, 8(3):184–187, 2011.

[25] J. Bourdon, D. Eveillard, and A. Siegel. Integrating quantitative knowledge into a qualitative gene regulatory network. *PLoS Comput Biol*, 7(9):e1002157, 2011.

[26] M.L. Kuijjer, M. Tung, G.C. Yuan, J. Quackenbush, and K. Glass. Estimating sample-specific regulatory networks. *arXiv preprint arXiv:1505.06440*, 2015.

[27] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

[28] A.L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1):69–77, 2000.

[29] D. Koschützki and F. Schreiber. Centrality analysis methods for biological networks and their application to gene regulatory networks. *Gene regulation and systems biology*, 2:193, 2008.

[30] K.H. Young. Yeast two-hybrid: so many interactions, (in) so little time... *Biol Reprod*, 58(2):302–311, February 1998.

[31] W.H. Dunham, M. Mullin, and A.C. Gingras. Affinity-purification coupled to mass spectrometry: Basic principles and strategies. *PROTEOMICS*, 12(10):1576–1590, 2012.

[32] G. Pavlopoulos, M. Secrier, C. Moschopoulos, T. Soldatos, S. Kossida, J. Aerts, R. Schneider, and P. Bagos. Using graph theory to analyze biological networks. *BioData Mining*, 4(1):10, 2011.

[33] W. Winterbach, P.V. Mieghem, M. Reinders, H. Wang, and D. Ridder. Topology of molecular interaction networks. *BMC Systems Biology*, 7(1):90, 2013.

[34] K. Raman. Construction and analysis of protein–protein interaction networks. *Automated experimentation*, 2(1):1, 2010.

[35] Eric de Silva and Michael PH Stumpf. Complex networks and simple models in biology. *Journal of the Royal Society Interface*, 2(5):419–430, 2005.

[36] Tero Aittokallio and Benno Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in bioinformatics*, 7(3):243–255, 2006.

[37] T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life: systems biology. *Annual review of genomics and human genetics*, 2(1):343–372, 2001.

[38] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[39] U.S. Bhalla and R. Iyengar. Emergent properties of networks of biological signaling pathways. *Science*, 283(5400):381–387, 1999.

[40] R. Guimera and L.A. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, February 2005.

[41] Koyel Mitra, Anne-Ruxandra Carvunis, Sanath Kumar Ramesh, and Trey Ideker. Integrative approaches for finding modular structure in biological networks. *Nature Reviews Genetics*, 14(10):719–732, 2013.

[42] J.F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G.F. Berriz, F.D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, et al. Towards a proteome-scale map of the human protein–protein interaction network. *Nature*, 437(7062):1173–1178, 2005.

[43] Vuk Janjić and Nataša Pržulj. Biological function through network topology: a survey of the human diseasome. *Briefings in functional genomics*, page els037, 2012.

[44] Soon-Hyung Yook, Zoltán N Oltvai, and Albert-László Barabási. Functional and topological characterization of protein interaction networks. *Proteomics*, 4(4):928–942, 2004.

[45] Zhi Liang, Meng Xu, Maikun Teng, and Liwen Niu. Comparison of protein interaction networks reveals species conservation and divergence. *BMC bioinformatics*, 7(1):457, 2006.

[46] Roded Sharan, Silpa Suthram, Ryan M Kelley, Tanja Kuhn, Scott McCuine, Peter Uetz, Taylor Sittler, Richard M Karp, and Trey Ideker. Conserved patterns of protein interaction in multiple species. *Proceedings of the National Academy of Sciences of the United States of America*, 102(6):1974–1979, 2005.

[47] G. Scardoni, A. Montresor, G. Tosadori, and C. Laudanna. Node interference and robustness: performing virtual knock-out experiments on biological networks: the case of leukocyte integrin activation network. *PloS one*, 9(2):e88938, 2014.

[48] X. Li, X. Xu, J. Wang, H. Yu, X. Wang, H. Yang, H. Xu, S. Tang, Y. Li, L. Yang, et al. A system-level investigation into the mechanisms of chinese traditional medicine: Compound danshen formula for cardiovascular disease treatment. *PloS one*, 7(9):e43918, 2012.

[49] J. Nair, M. Ghatge, V.V. Kakkar, and J. Shanker. Network analysis of inflammatory genes and their transcriptional regulators in coronary artery disease. *PloS one*, 9(4):e94328, 2014.

[50] M. Jalili, A. Salehzadeh-Yazdi, Y. Asgari, S.S. Arab, M. Yaghmaie, A. Ghavamzadeh, and K. Alimoghaddam. Centiserver: A comprehensive resource, web-based application and r package for centrality analysis. *PloS one*, 10(11):e0143111, 2015.

[51] S.P. Borgatti and M.G. Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

[52] A. Pandey and M. Mann. Proteomics to study genes and genomes. *Nature*, 405(6788):837–846, 2000.

[53] A.G.N. Wetie, I. Sokolowska, A.G. Woods, U. Roy, K. Deinhardt, and C.C. Darie. Protein-protein interactions: switch from classical methods to proteomics and bioinformatics-based approaches. *Cellular and molecular life sciences*, 71(2):205–228, 2014.

[54] L.J. Jensen, M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering. STRING 8–a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, 37(Database issue):D412–D416, January 2009.

[55] C. Stark, B.J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers. BioGRID: a general repository for interaction datasets. *Nucleic acids research*, 34(Database issue):D535–D539, January 2006.

[56] A. Chatr-Aryamontri, A. Ceol, L.M. Palazzi, M.V. Nardel, L. Castagnoli, and G. Cesareni. Mint: the molecular interaction database. *Nucleic acids research*, 35(suppl 1):D572–D574, 2007.

[57] T.S. Keshava Prasad, R. Goel, K. Kandasamy, S. Keerthikumar, S. Kumar, S. Mathivanan, D. Telikicherla, R. Raju, B. Shafreen, A. Venugopal, L. Balakrishnan, A. Marimuthu, S. Banerjee, D.S. Somanathan, A. Sebastian, SD. Rani, S. Ray, C.J. Harrys Kishore, S. Kanth, M. Ahmed, M.K. Kashyap, R. Mohmood, Y.L. Ramachandra, V. Krishna, B.A. Rahiman, S. Mohan, P. Ranganathan, S. Ramabadran, R. Chaerkady, and A. Pandey. Human protein reference database—2009 update. *Nucleic Acids Research*, 37(suppl 1):D767–D772, 2009.

[58] H.W. Mewes, K. Albermann, K. Heumann, S. Liebl, and F. Pfeiffer. MIPS: a database for protein sequences, homology data and yeast genome information. *Nucleic Acids Res*, 25(1):28–30, January 1997.

[59] L. Salwinski, C.S. Miller, A.J. Smith, F.K. Pettit, J.U. Bowie, and D. Eisenberg. The Database of Interacting Proteins: 2004 update. *Nucleic acids research*, 32(Database issue):D449–D451, January 2004.

[60] J. Wu, T. Vallenius, K. Ovaska, J. Westermarck, T.P. Mäkelä, and S. Hautaniemi. Integrated network analysis platform for protein-protein interactions. *Nature methods*, 6(1):75–77, 2009.

[61] H. Azevedo and C.A. Moreira-Filho. Topological robustness analysis of protein interaction networks reveals key targets for overcoming chemotherapy resistance in glioma. *Scientific reports*, 5, 2015.

[62] B. Orlando, N. Bragazzi, and C. Nicolini. Bioinformatics and systems biology analysis of genes network involved in olp (oral lichen planus) pathogenesis. *Archives of oral biology*, 58(6):664–673, 2013.

[63] M. Jalili, A. Salehzadeh-Yazdi, S. Gupta, O. Wolkenhauer, M. Yaghmaie, O. Resendis-Antonio, and k. Alimoghaddam. Evolution of centrality measurements for the detection of essential proteins in biological networks. *Frontiers in Physiology*, 7, 2016.

[64] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[65] C. Mitrea, Z. Taghavi, B. Bokanizad, S. Hanoudi, R. Tagett, M. Donato, C. Voichita, and S. Draghici. Methods and approaches in the topology-based analysis of biological pathways. *Frontiers in physiology*, 4:278, 2013.

[66] J. Sun and Z. Zhao. A comparative study of cancer proteins in the human protein-protein interaction network. *BMC genomics*, 11(3):1, 2010.

[67] X. Ma and L. Gao. Biological network analysis: insights into structure and functions. *Briefings in functional genomics*, 11(6):434–442, 2012.

[68] L. Hood. Systems biology and p4 medicine: past, present, and future. *Rambam Maimonides Med J*, 4(2):e0012, 2013.

[69] R. Chen and M. Snyder. Systems biology: personalized medicine for the future? *Current opinion in pharmacology*, 12(5):623–628, 2012.

[70] H. Jeong, S.P. Mason, A.L. Barabási, and Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[71] R. Saito, M.E. Smoot, K. Ono, J. Ruscheinski, P.L. Wang, S. Lotia, A.R. Pico, G.D. Bader, and T. Ideker. A travel guide to Cytoscape plugins. *Nat. Methods*, 9(11):1069–1076, Nov 2012.

[72] M.E. Smoot, K. Ono, J. Ruscheinski, P.L. Wang, and T. Ideker. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, Feb 2011.

[73] M. S. Cline, M. Smoot, E. Cerami, A. Kuchinsky, N. Landys, C. Workman, R. C.tmas, I. Avila-Campilo, M. Creech, B. Gross, K. Hanspers, R. Isserlin, R. Kelley, S. Killcoyne, S. Lotia, S. Maere, J. Morris, K. Ono, V. Pavlovic, A.R. Pico, A. Vailaya, P.L. Wang, A. Adler, B.R. Conklin, L. Hood, M. Kuiper, C. Sander, I. Schmulevich, B. Schwikowski, G.J. Warner, T. Ideker, and G.D. Bader. Integration of biological networks

and gene expression data using Cytoscape. *Nat Protoc*, 2(10):2366–2382, 2007.

[74] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Res.*, 13(11):2498–2504, Nov 2003.

[75] G. Scardoni, G. Tosadori, S. Pratap, F. Spoto, and C. Laudanna. Finding the shortest path with pesca: a tool for network reconstruction. *F1000Research*, 4, 2015.

[76] G. Tosadori, I. Bestvina, F. Spoto, C. Laudanna, and G. Scardoni. Creating, generating and comparing random network models with network randomizer [version 1; referees: awaiting p. review]. *F1000Research*, 5(2524), 2016.

[77] E. Zenaro, E. Pietronigro, V. Della Bianca, G. Piacentino, L. Marongiu, S. Budui, E. Turano, B. Rossi, S. Angiari, S. Dusi, et al. Neutrophils promote alzheimer's disease-like pathology and cognitive decline via lfa-1 integrin. *Nature medicine*, 21(8):880–886, 2015.

[78] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–913, 2002.

[79] Francis Crick et al. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[80] Q.C. Zhang, D. PeT., J.I. Garzón, L. Deng, and B. Honig. Preppi: a structure-informed database of protein–protein interactions. *Nucleic acids research*, page gks1231, 2012.

[81] O. Keskin, N. Tuncbag, and A. Gursoy. Predicting protein–protein interactions from the molecular to the proteome level. *Chemical reviews*, 116(8):4884–4909, 2016.

[82] C. Laudanna. Human interactome in sif format, from public databases., Year of Access: April 2015.

[83] F. Brambilla, F. Lavatelli, D. Di Silvestre, V. Valentini, R. Rossi, G. Palladini, L. Obici, L. Verga, P. Mauri, and G. Merlini. Reliable typing of systemic amyloidoses through proteomic analysis of subcutaneous adipose tissue. *Blood*, pages blood–2011, 2011.

[84] P. Capelli, M. Fassan, and A. Scarpa. Pathology-grading and staging of gep-nets. *Best Practice & Research Clinical Gastroenterology*, 26(6):705–717, 2012.

[85] E. Missiaglia, I Dalai, S. Barbi, S. Beghelli, M. Falconi, M. Della Peruta, L. Piemonti, G. Capurso, A. Di Florio, G. Delle Fave, et al. Pancreatic

endocrine tumors: expression profiling evidences a role for akt-mtor pathway. *Journal of Clinical Oncology*, 28(2):245–255, 2010.

[86] A. Simioniuc, M. Campan, V. Lionetti, M. Marinelli, G.D. Aquaro, C. Cavallini, S. Valente, D. Di Silvestre, S. Cantoni, F. Bernini, et al. Placental stem cells pre-treated with a hyaluronan mixed ester of butyric and retinoic acid to cure infarcted pig hearts: a multimodal study. *Cardiovascular research*, 90(3):546–556, 2011.

[87] F. Meurens, A. Summerfield, H. Nauwynck, L. Saif, and V. Gerdts. The pig: a model for human infectious diseases. *Trends in microbiology*, 20(1):50–57, 2012.

[88] C. Braicu, R Cojocneanu-Petric, A. Jurj, D. Gulei, I. Taranu, A.M. Gras, D.E. Marin, and I. Berindan-Neagoe. Microarray based gene expression analysis of sus scrofa duodenum exposed to zearalenone: significance to human health. *BMC genomics*, 17(1):646, 2016.

[89] H. Yu, Q. Wu, J. Zhang, Y. Zhang, C. Lu, Y. Cheng, Z. Zhao, A. Windemuth, D. Liu, and L. Hao. Genome-wide characterization of pre-1 reveals a hidden evolutionary relationship between suidae and primates. *bioRxiv*, page 025791, 2015.

[90] Eric C. Schirmer, J.R. Yates, and L. Gerace. Mudpit: A powerful proteomics tool for discovery. *Discovery medicine*, 3(18):38–39, 2003.

[91] Kinexus. Kinexus microarrays information, Year of Access: December 2010.

[92] P. Shannon, A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, 2003.

[93] G. Scardoni, G. Tosadori, M. Faizan, F. Spoto, F. Fabbri, and C. Laudanna. Biological network analysis with centiscape: centralities and experimental dataset integration. *F1000Research*, 3, 2014.

[94] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social networks*, 17(1):57–63, 1995.

[95] G Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[96] T.W. Valente and R.K. Foreman. Integration and radiality: measuring the extent of an individual's connectedness and reachability in a network. *Social networks*, 20(1):89–105, 1998.

[97] A. Shimbel. Structural parameters of communication networks. *The bulletin of mathematical biophysics*, 15(4):501–507, 1953.

[98] D.R. White and S.P. Borgatti. Betweenness centrality measures for directed graphs. *Social Networks*, 16(4):335–346, 1994.

[99] W. Hwang, Y. Cho, A. Zhang, and M. Ramanathan. Bridging centrality: identifying bridging nodes in scale-free networks. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 20–23, 2006.

[100] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[101] G. Roffo, S. Melzi, and M. Cristani. Infinite feature selection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4202–4210, 2015.

[102] M.W. Libbrecht and W.S. Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332, 2015.

[103] A. Ben-Hur, C.S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. Support vector machines and kernels for computational biology. *PLoS Comput Biol*, 4(10):e1000173, 2008.

[104] G. Scardoni, M. Petterlini, and C. Laudanna. Analyzing biological network parameters with centiscape. *Bioinformatics*, 25(21):2857–2859, 2009.

[105] H. Liu and H. Motoda. *Computational methods of feature selection*. CRC Press, 2007.

[106] M. Zaffalon and M. Hutter. Robust feature selection using distributions of mutual information. In *Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pages 577–584, 2002.

[107] P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

[108] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in neural information processing systems*, pages 507–514, 2005.

[109] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[110] Q. Gu, Z. Li, and J. Han. Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*, 2012.

[111] G. Roffo. Feature selection library (matlab toolbox). *arXiv preprint arXiv:1607.01327*, 2016.

[112] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, and S. Verzakov. Pr-tools4.1, a matlab toolbox for pattern recognition, 2007. `http://prtools.org`.

[113] M.S. Cline, M. Smoot, E. Cerami, A. Kuchinsky, N. Landys, C. Workman, R. Christmas, I. Avila-Campilo, M. Creech, B. Gross, et al. Integration of biological networks and gene expression data using cytoscape. *Nature protocols*, 2(10):2366–2382, 2007.

[114] P. Sah, L.O. Singh, A. Clauset, and S. Bansal. Exploring community structure in biological networks with random graphs. *BMC bioinformatics*, 15(1):220, 2014.

[115] B. Haibe-Kains and F. Emmert-Streib. *Quantitative assessment and validation of network inference methods in bioinformatics*. Frontiers Media SA, 2015.

[116] R.R. Wilcox. Some practical reasons for reconsidering the kolmogorov-smirnov test. *British Journal of Mathematical and Statistical Psychology*, 50(1):9–20, 1997.

[117] P. Erdös and A. Rényi. On random graphs, i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[118] D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[119] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

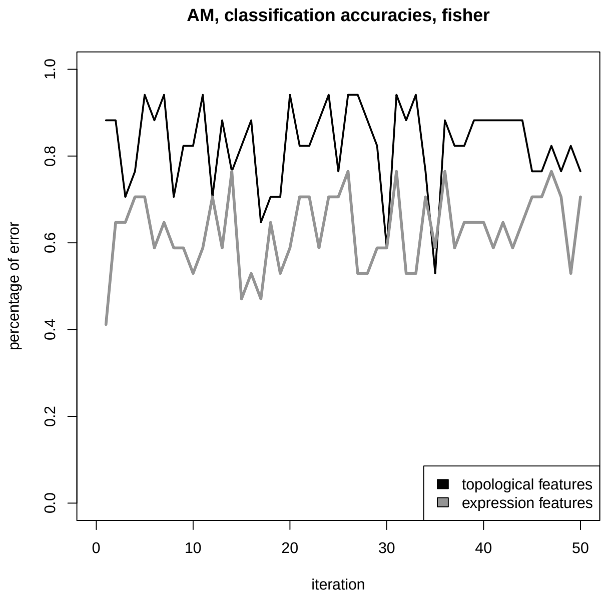
[120] J. Yang and J. Leskovec. Community-affiliation graph model for overlapping network community detection. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1170–1175. IEEE, 2012.

[121] T. Anthony. A brief history of systems biology. *The Plant Cell*, 18:2420–2430, 2006.

Appendices

Appendix A

Graphical Results



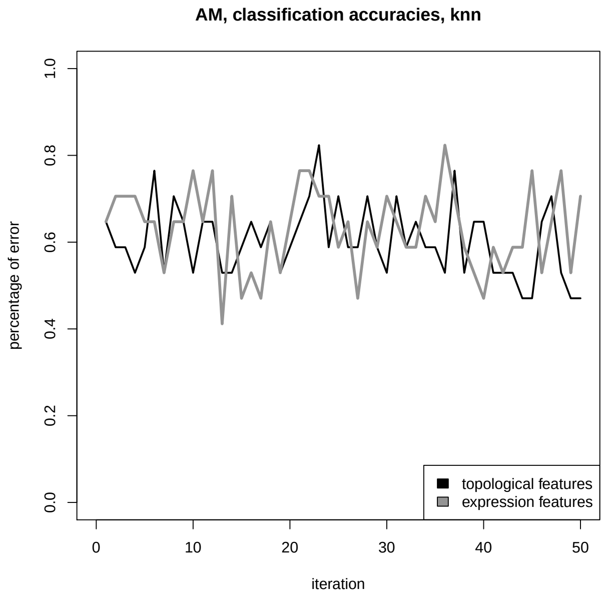
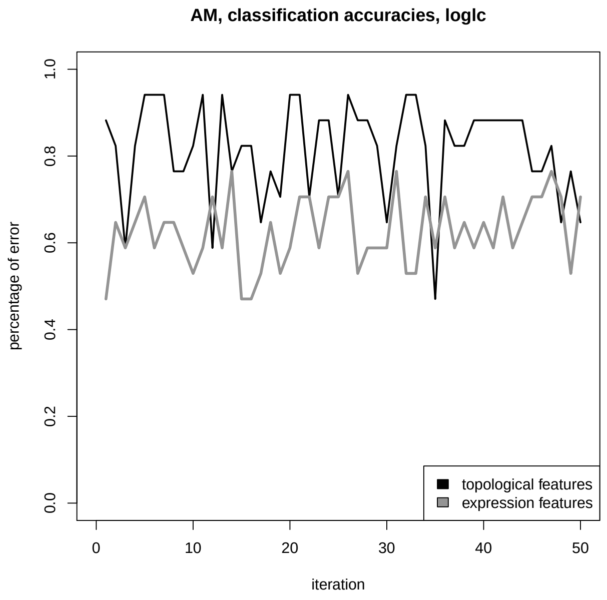


Figure A.1: Comparison between topological and raw data features, AM dataset.



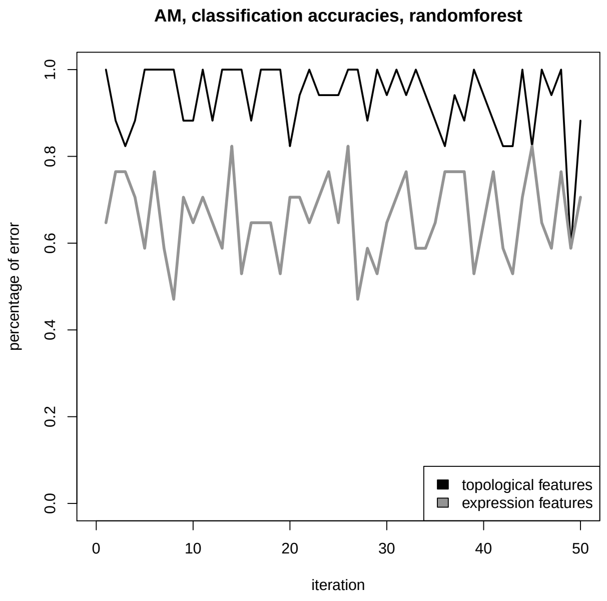


Figure A.2: Comparison between topological and raw data features, AM dataset.

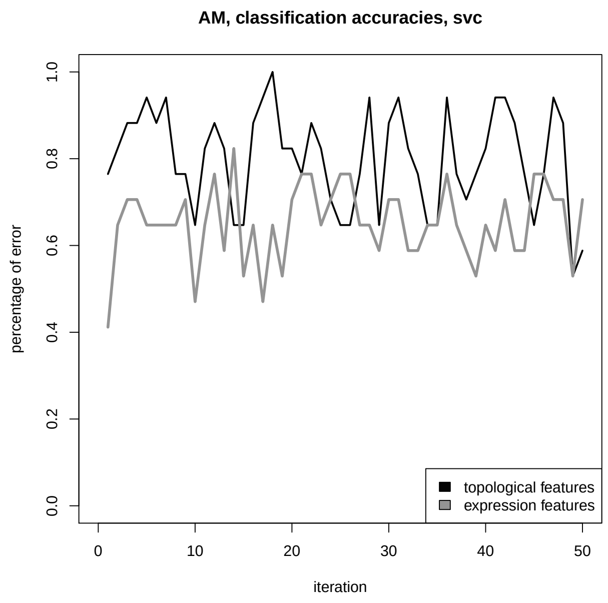
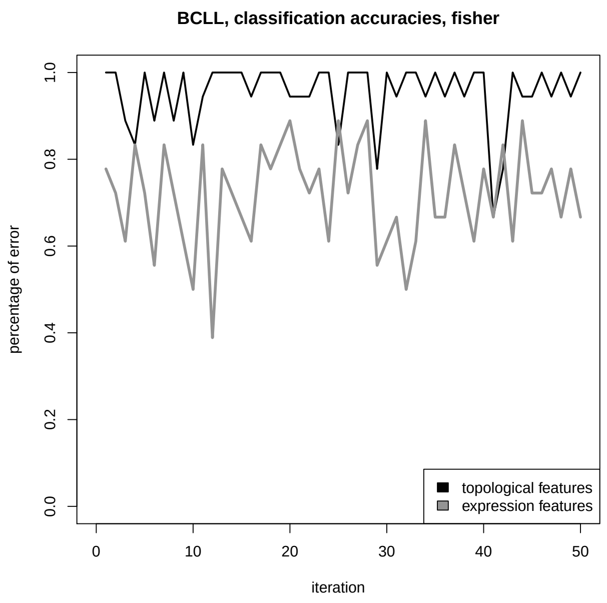


Figure A.3: Comparison between topological and raw data features, AM dataset.



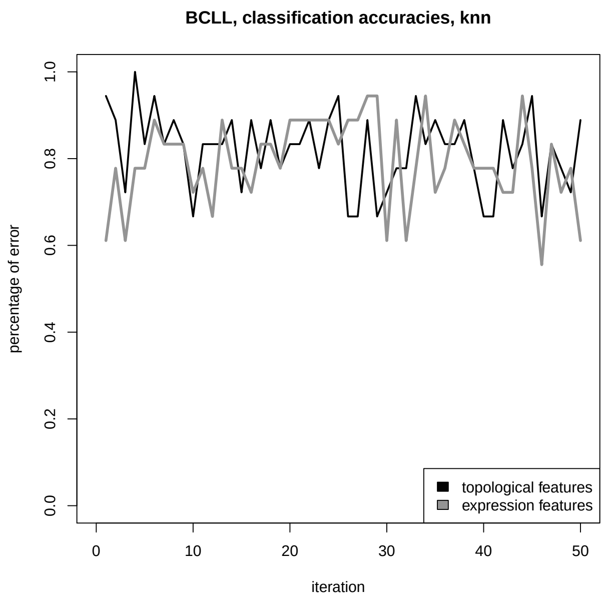
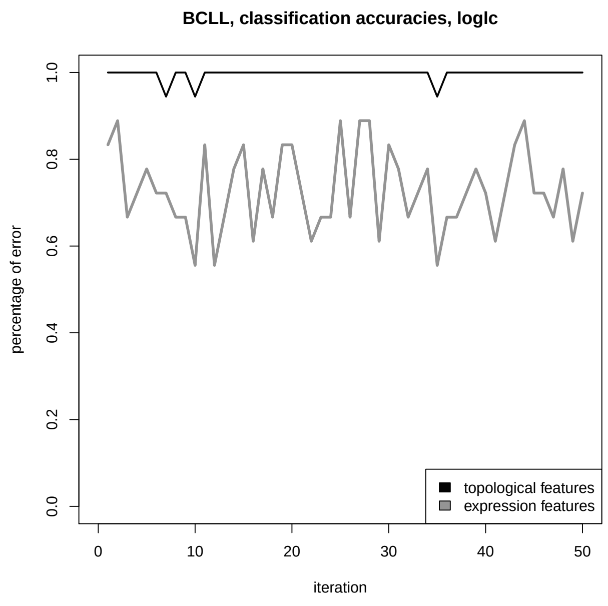


Figure A.4: Comparison between topological and raw data features, BCLL dataset.



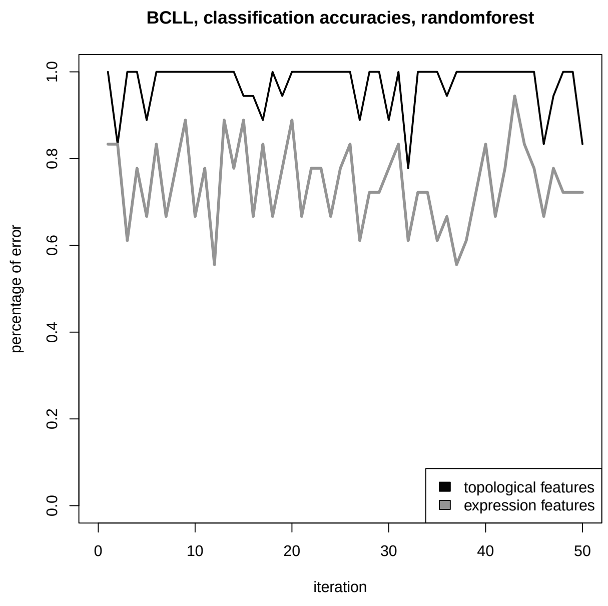


Figure A.5: Comparison between topological and raw data features, BCLL dataset.

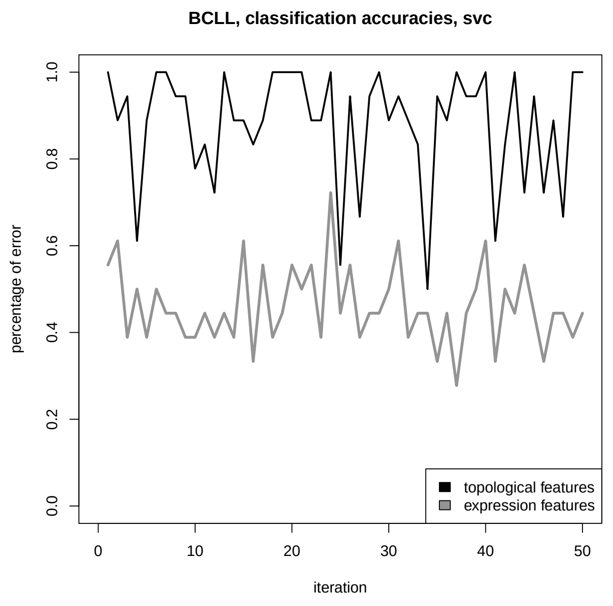
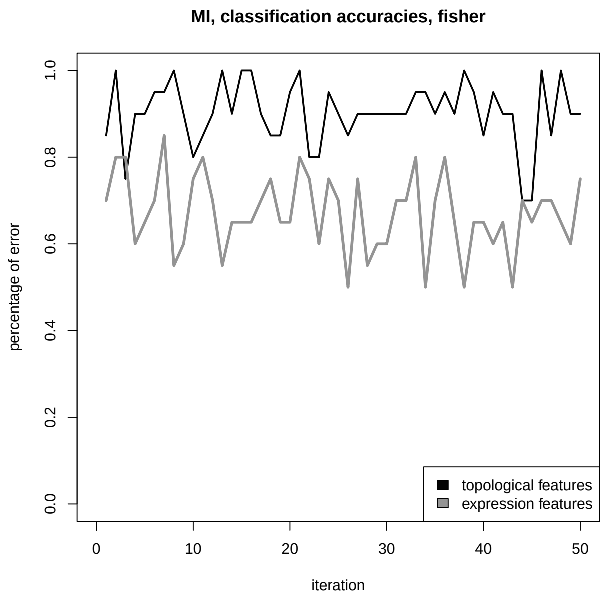


Figure A.6: Comparison between topological and raw data features, BCLL dataset.



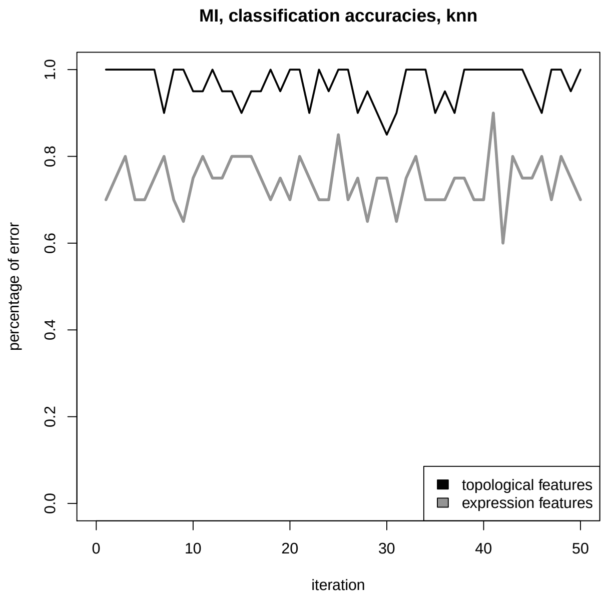
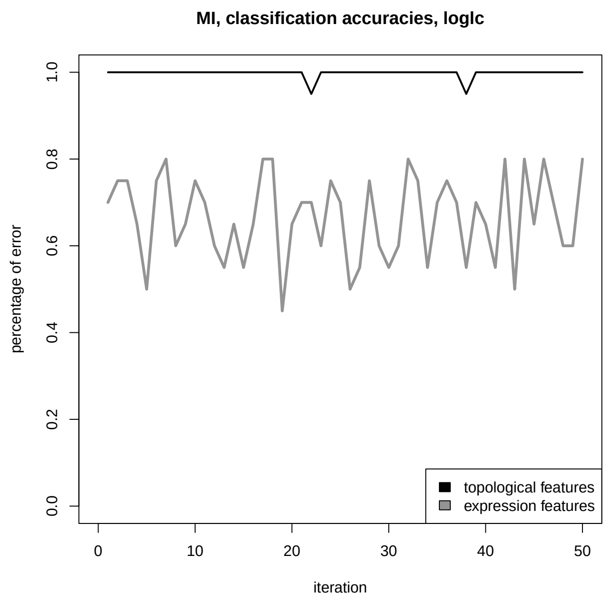


Figure A.7: Comparison between topological and raw data features, MI dataset.



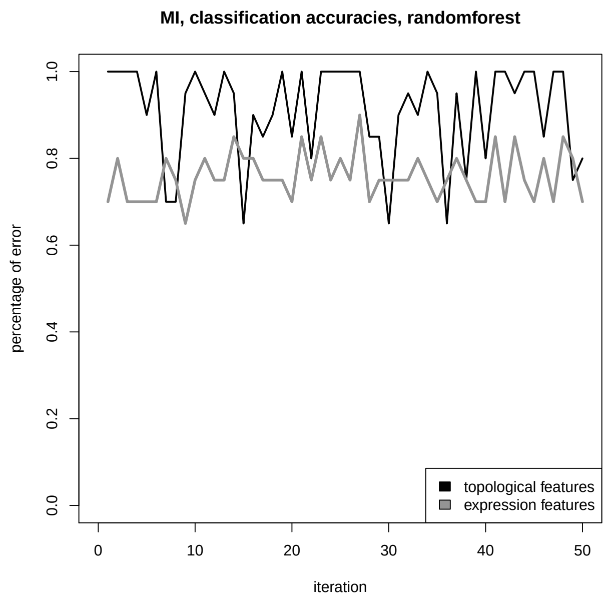


Figure A.8: Comparison between topological and raw data features, MI dataset.

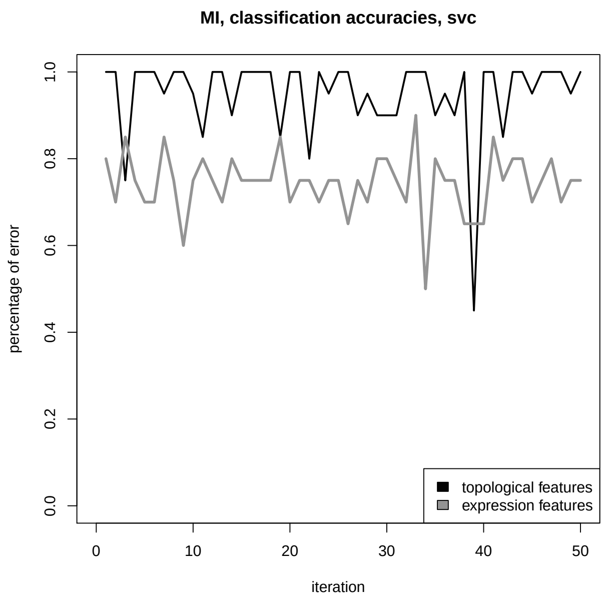
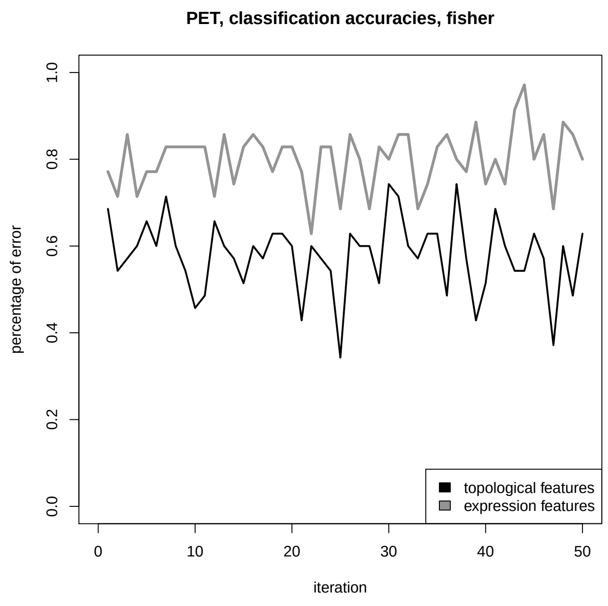


Figure A.9: Comparison between topological and raw data features, MI dataset.



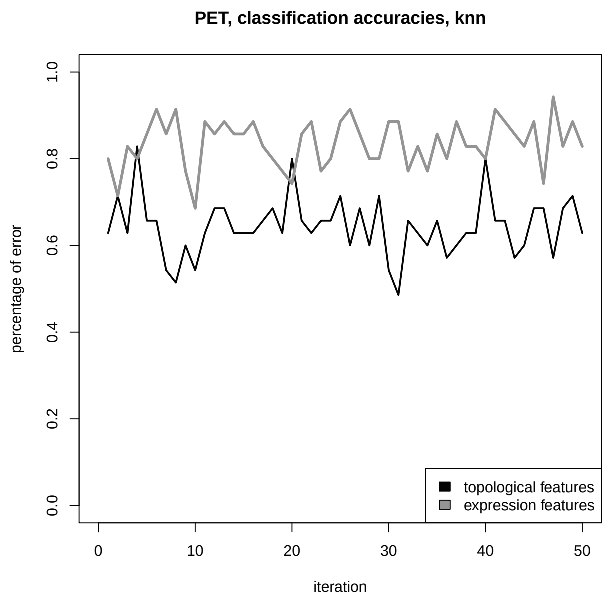
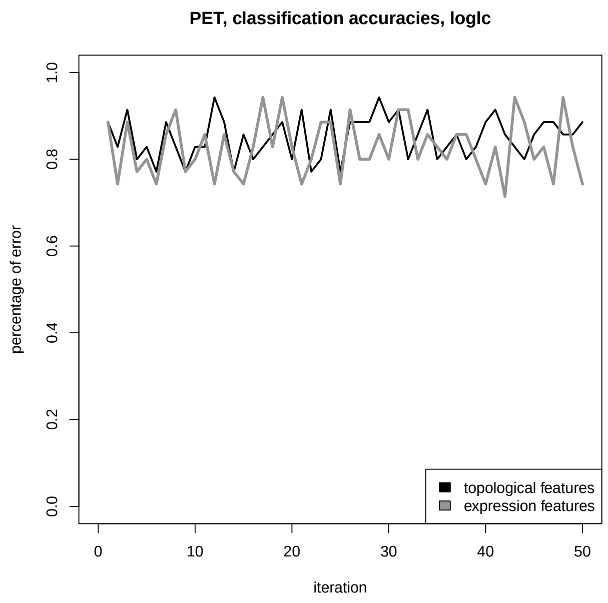


Figure A.10: Comparison between topological and raw data features, PET dataset.



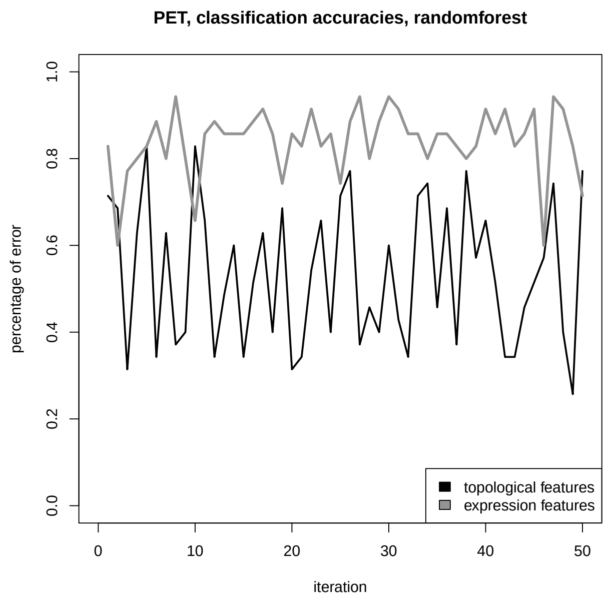


Figure A.11: Comparison between topological and raw data features, PET dataset.

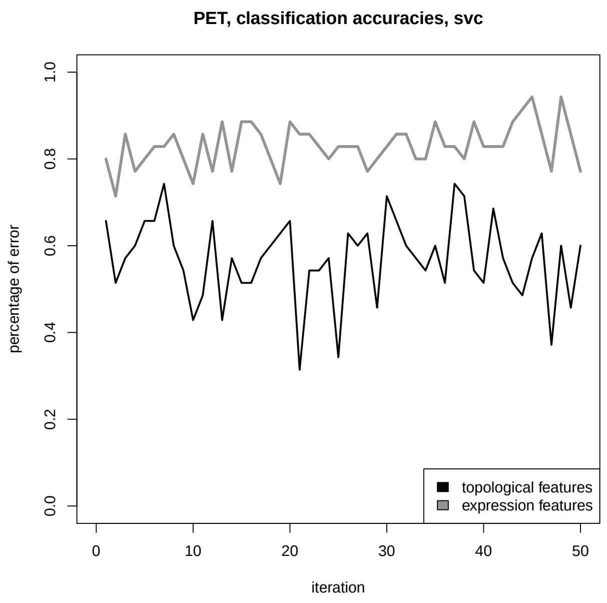
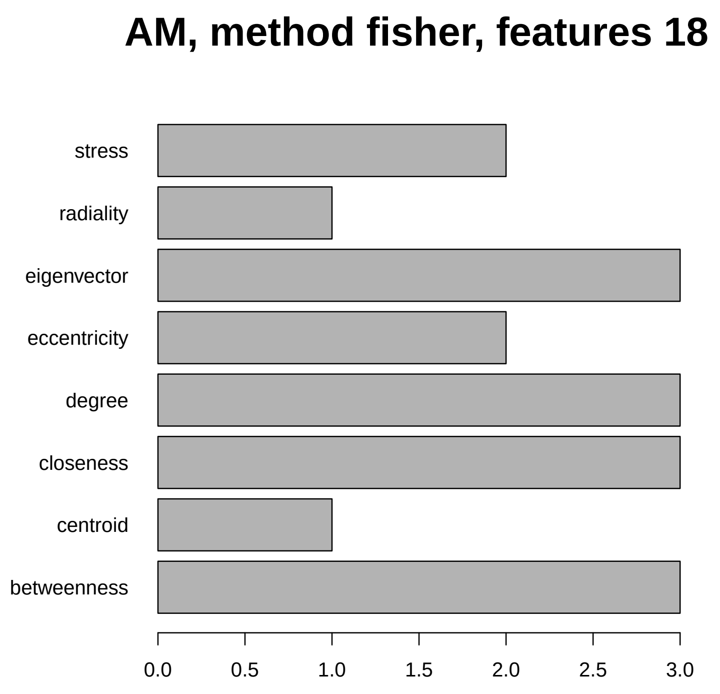


Figure A.12: Comparison between topological and raw data features, PET dataset.



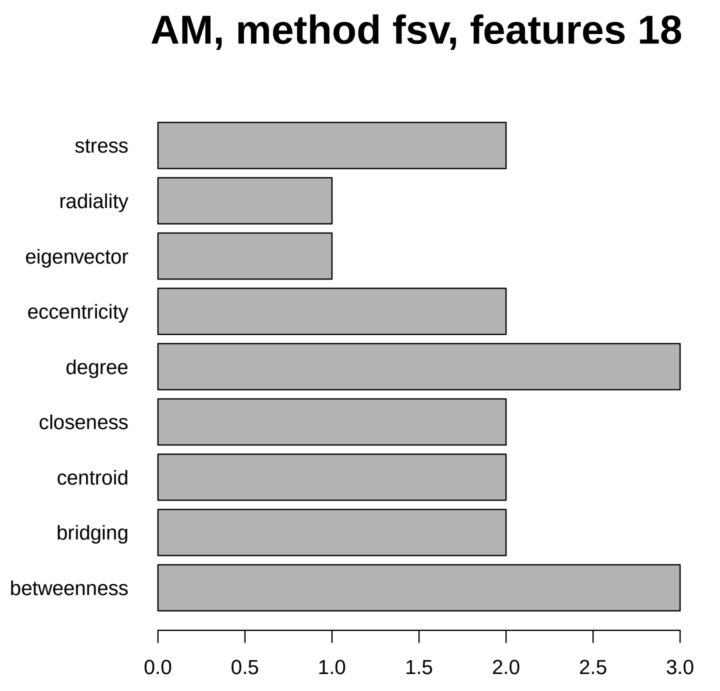
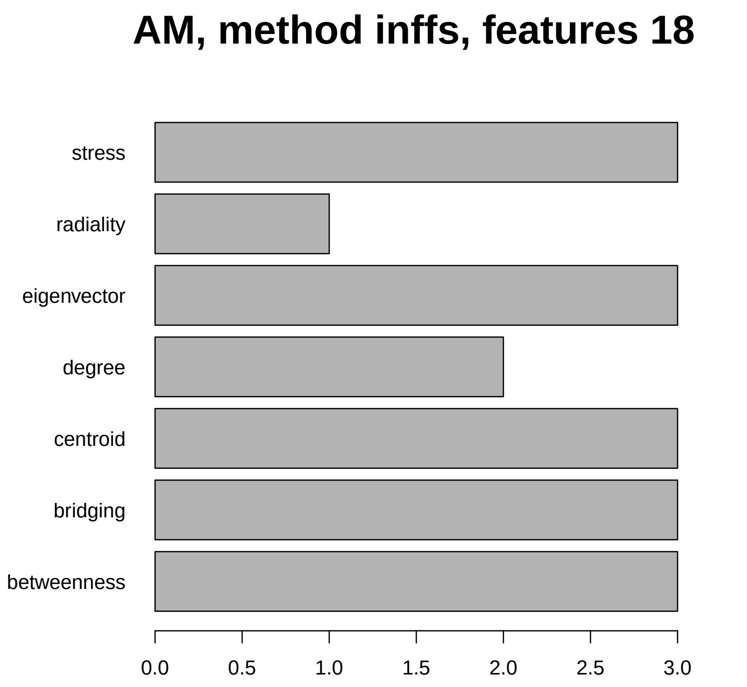


Figure A.13: Histograms that shows all the feature selections algorithms that were run.



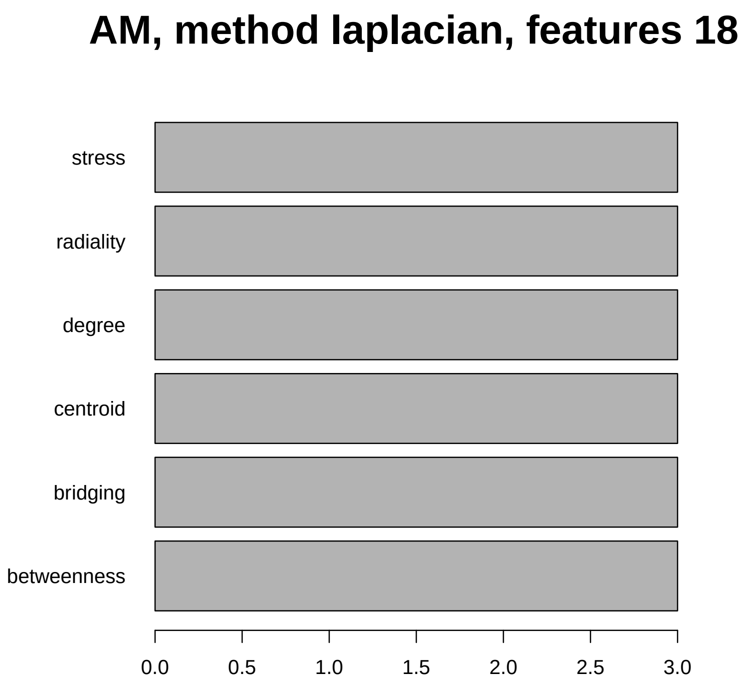
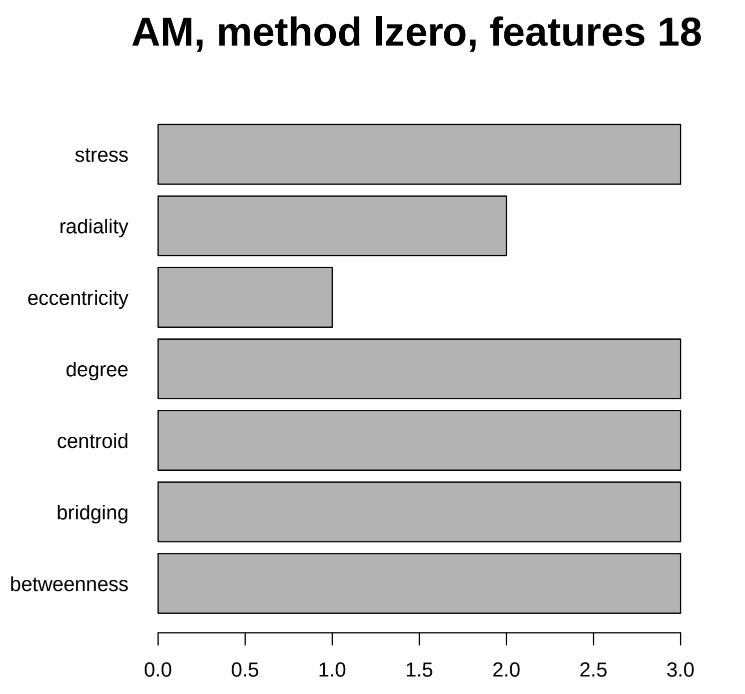


Figure A.14: Histograms that shows all the feature selections algorithms that were run.



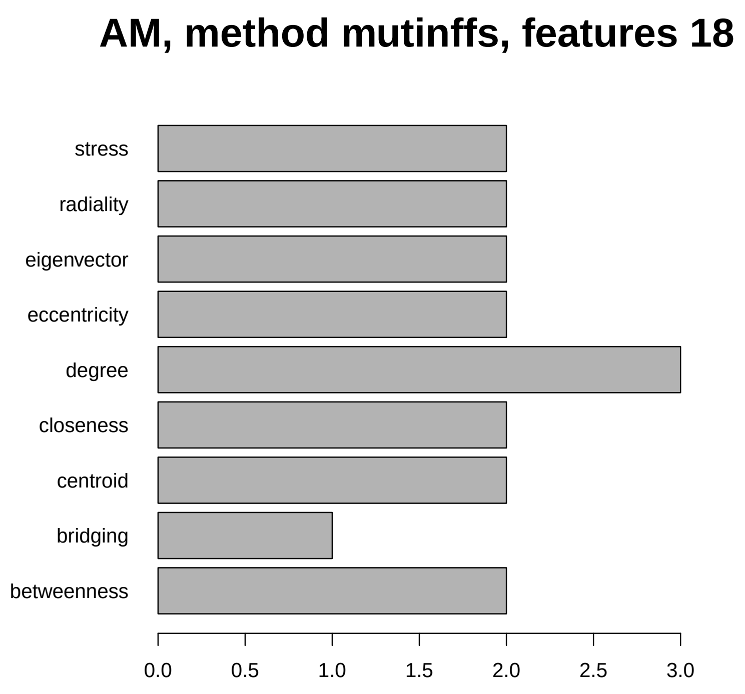
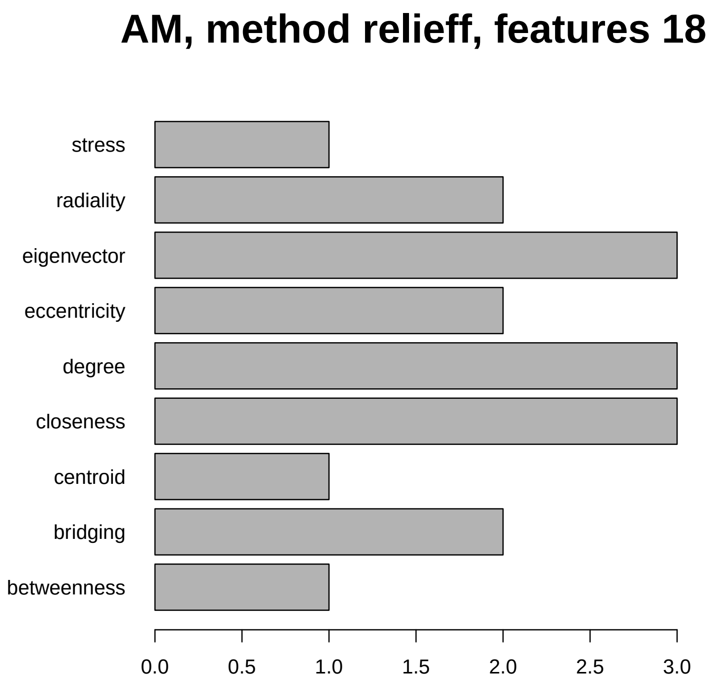


Figure A.15: Histograms that shows all the feature selections algorithms that were run.



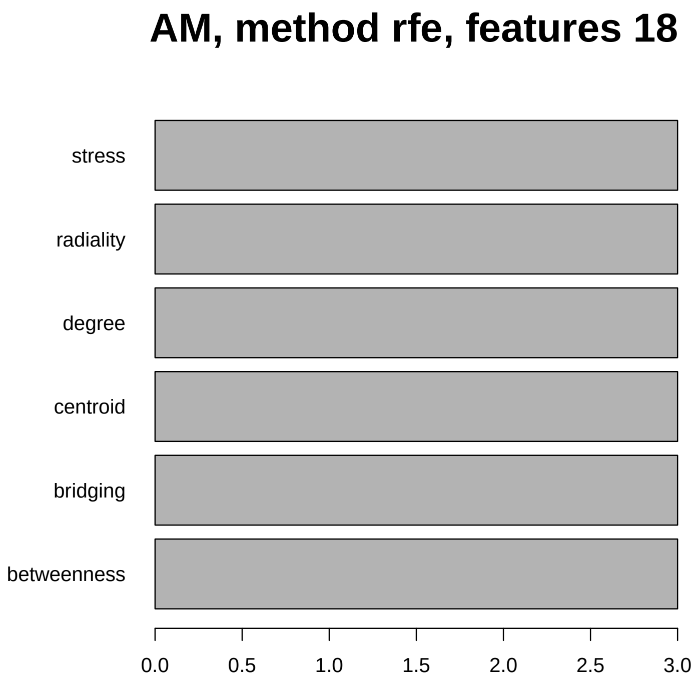
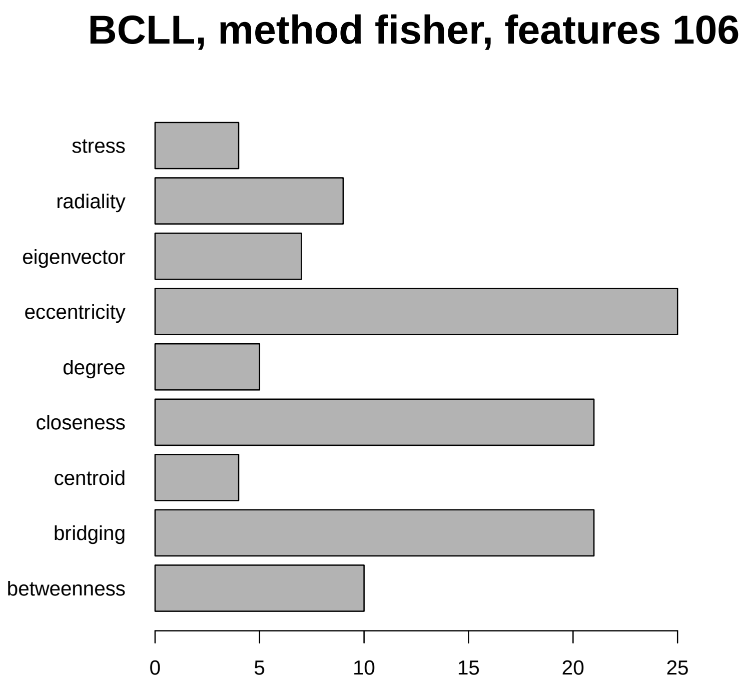


Figure A.16: Histograms that shows all the feature selections algorithms that were run.



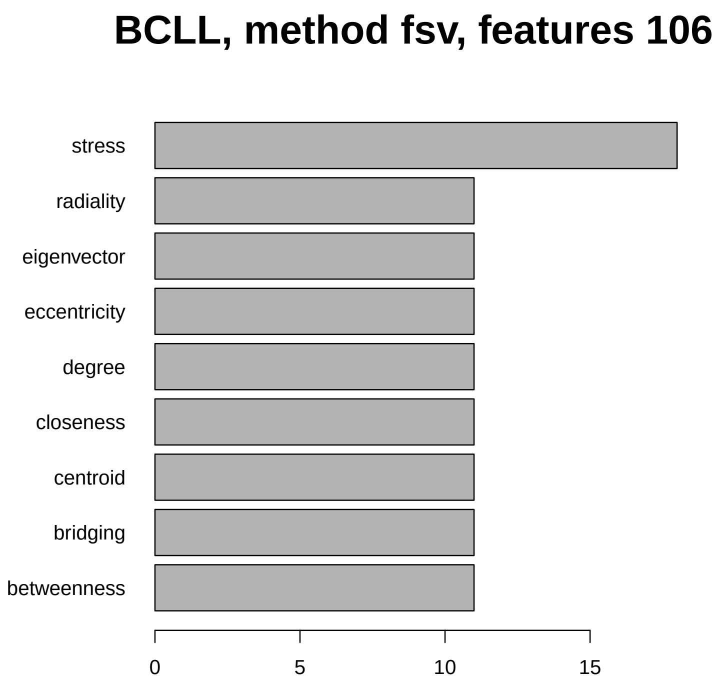
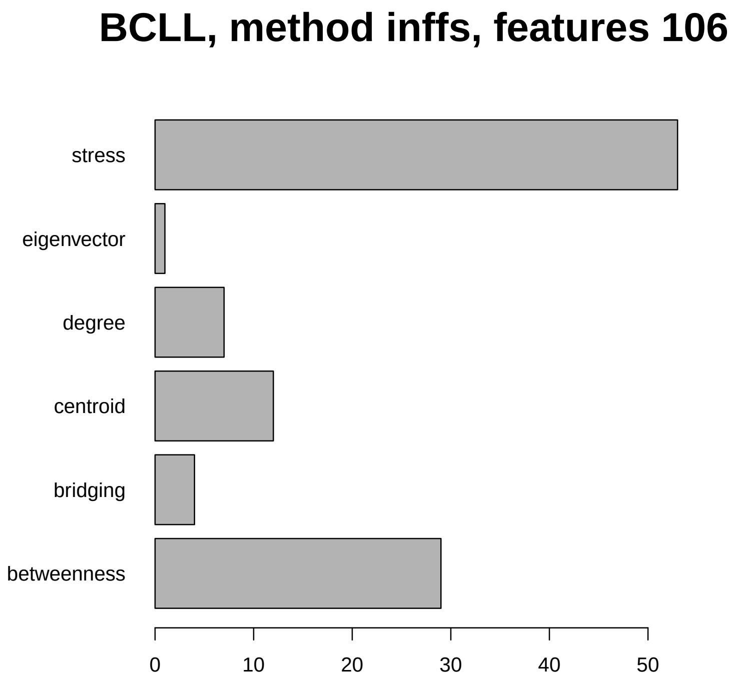


Figure A.17: Histograms that shows all the feature selections algorithms that were run.



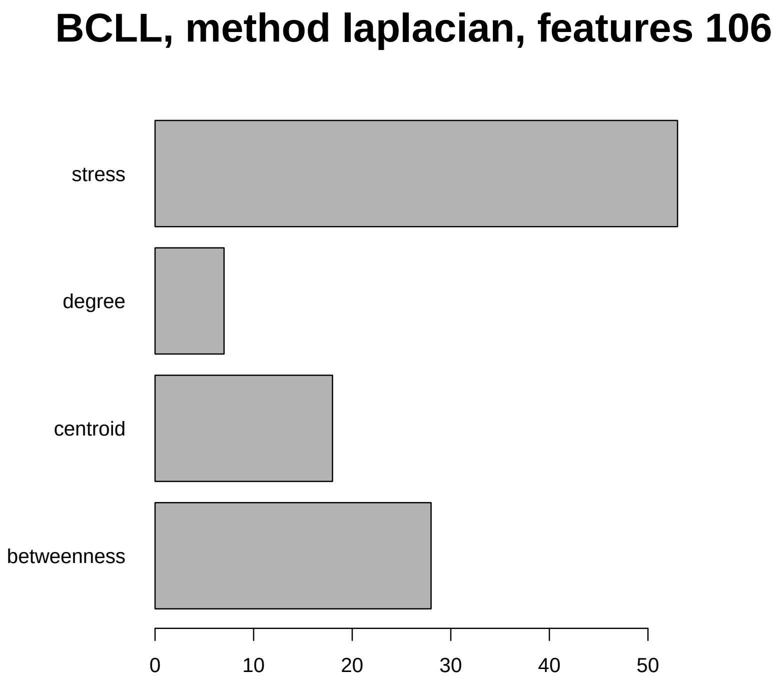
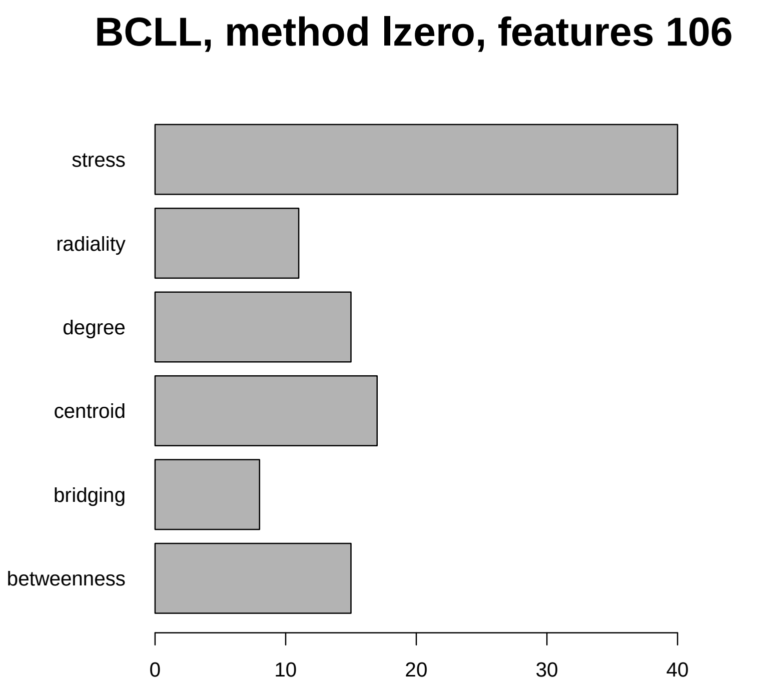


Figure A.18: Histograms that shows all the feature selections algorithms that were run.



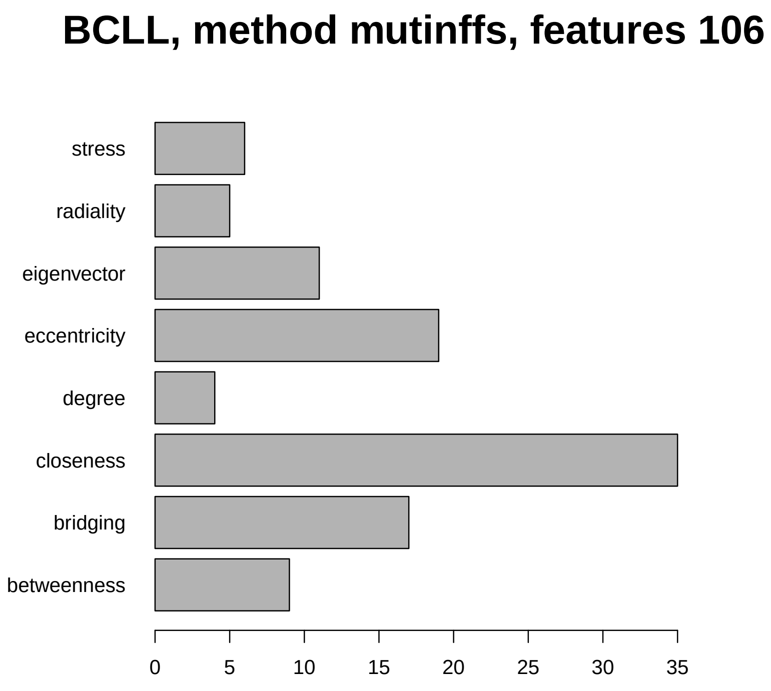
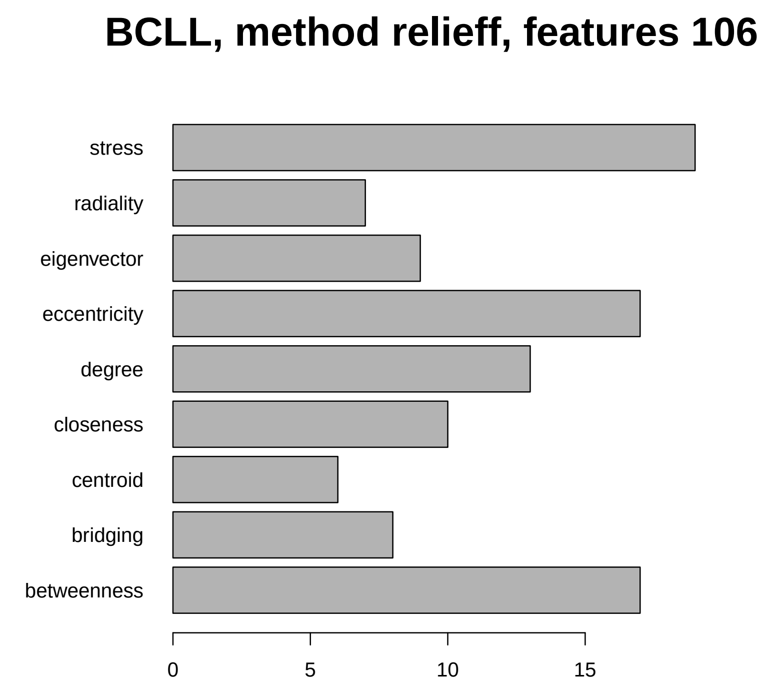


Figure A.19: Histograms that shows all the feature selections algorithms that were run.



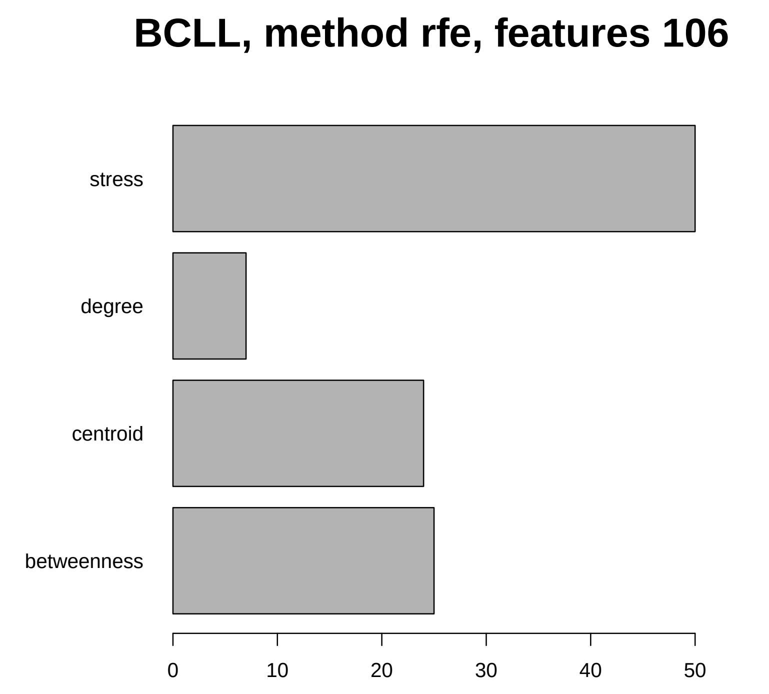
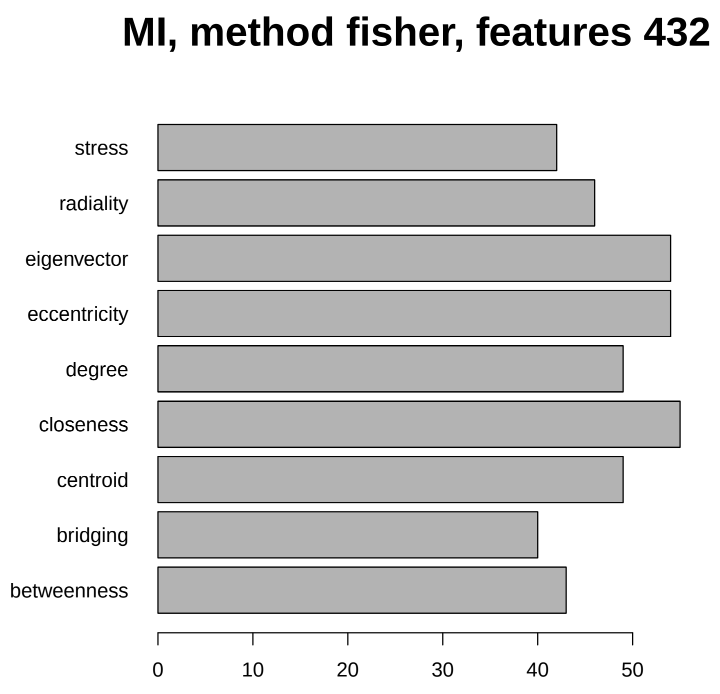


Figure A.20: Histograms that shows all the feature selections algorithms that were run.



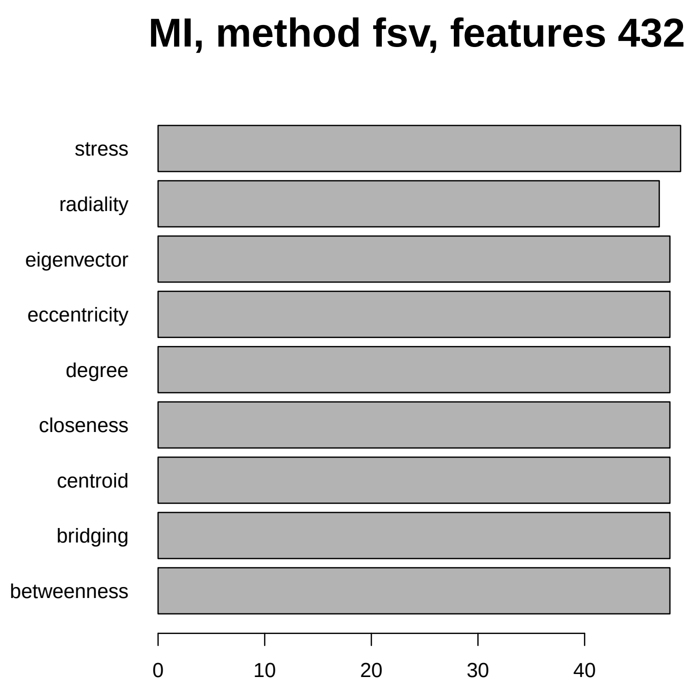
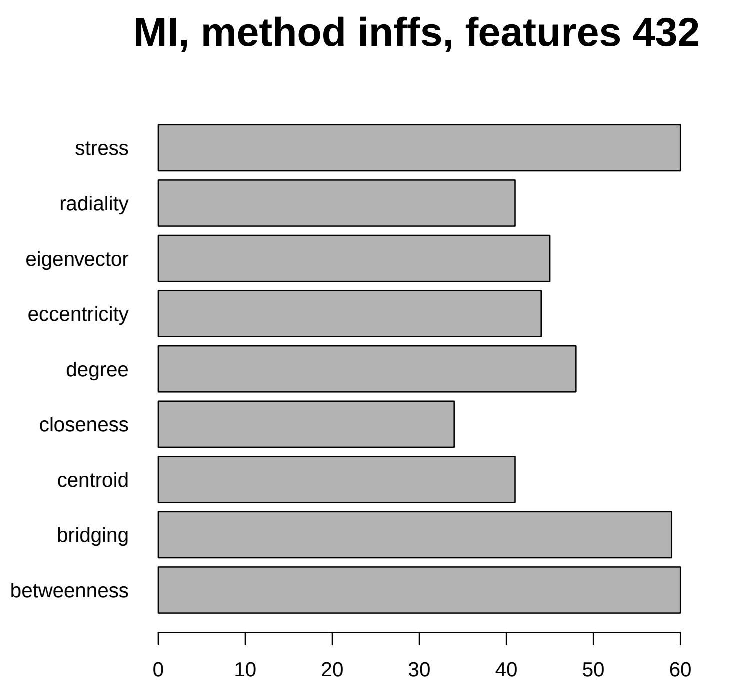


Figure A.21: Histograms that shows all the feature selections algorithms that were run.



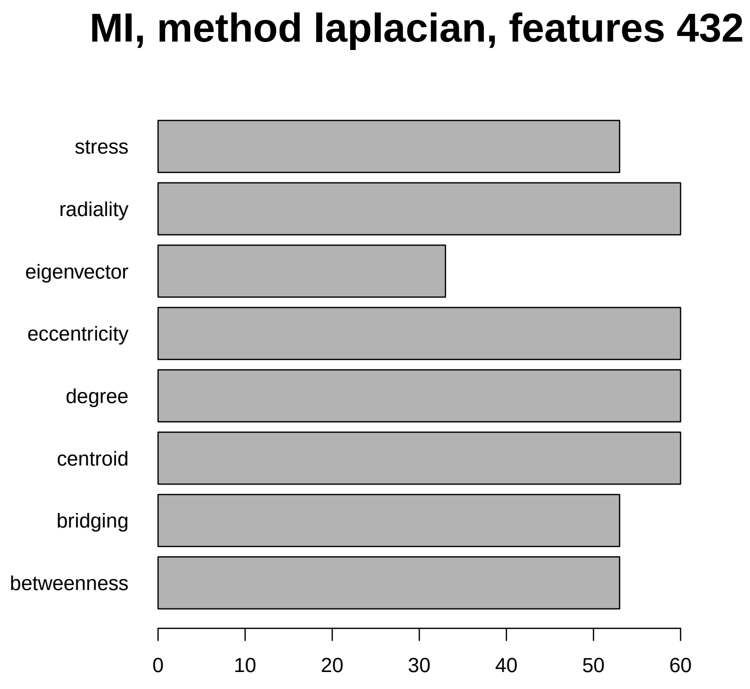
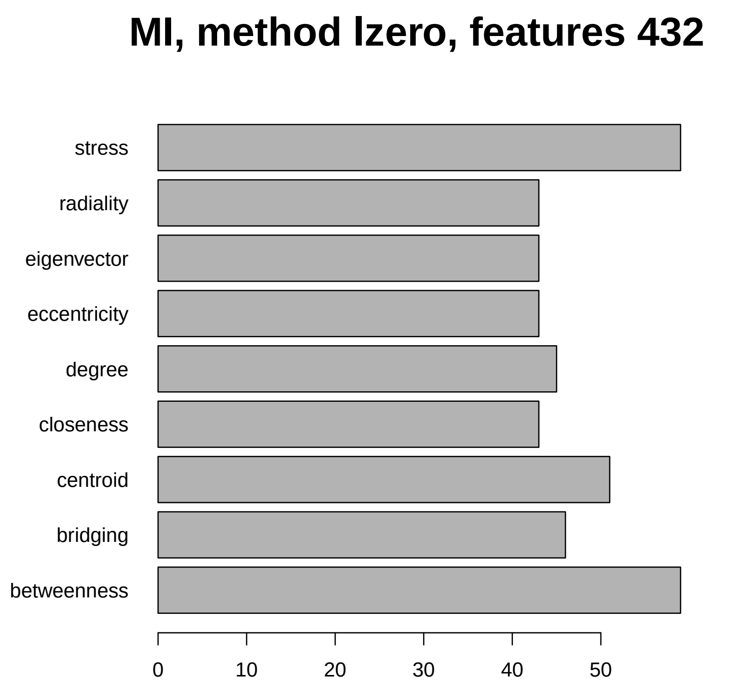


Figure A.22: Histograms that shows all the feature selections algorithms that were run.



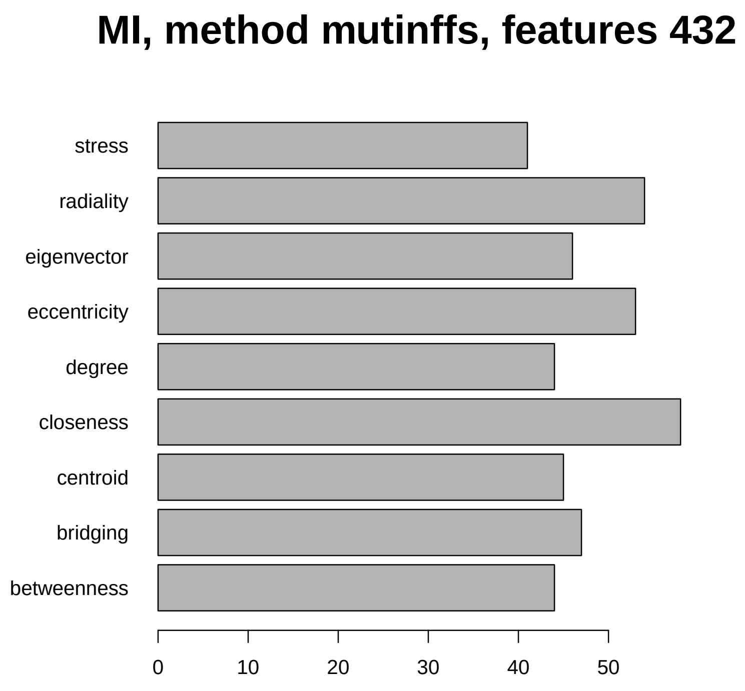
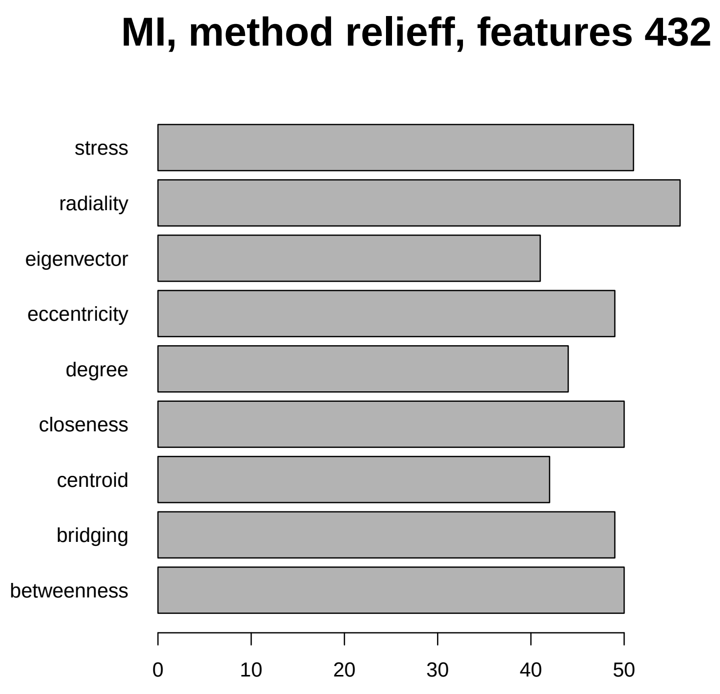


Figure A.23: Histograms that shows all the feature selections algorithms that were run.



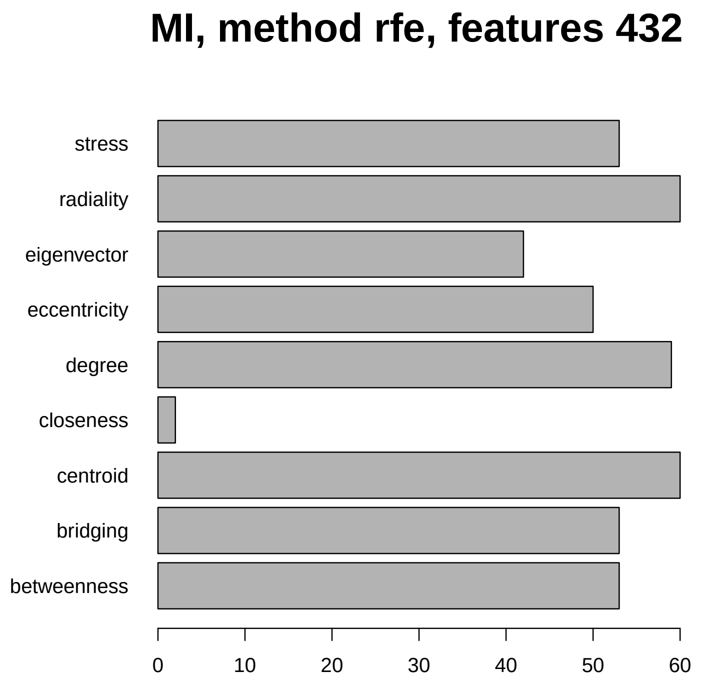
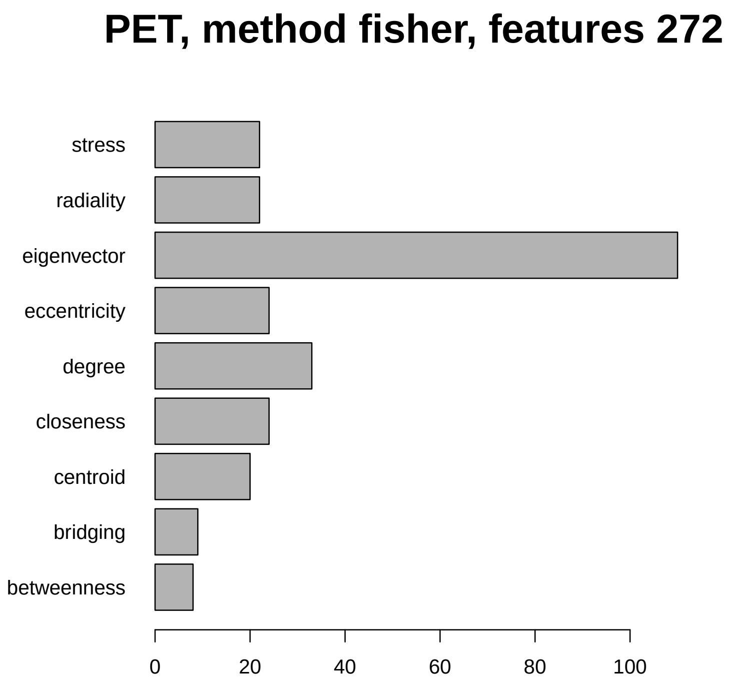


Figure A.24: Histograms that shows all the feature selections algorithms that were run.



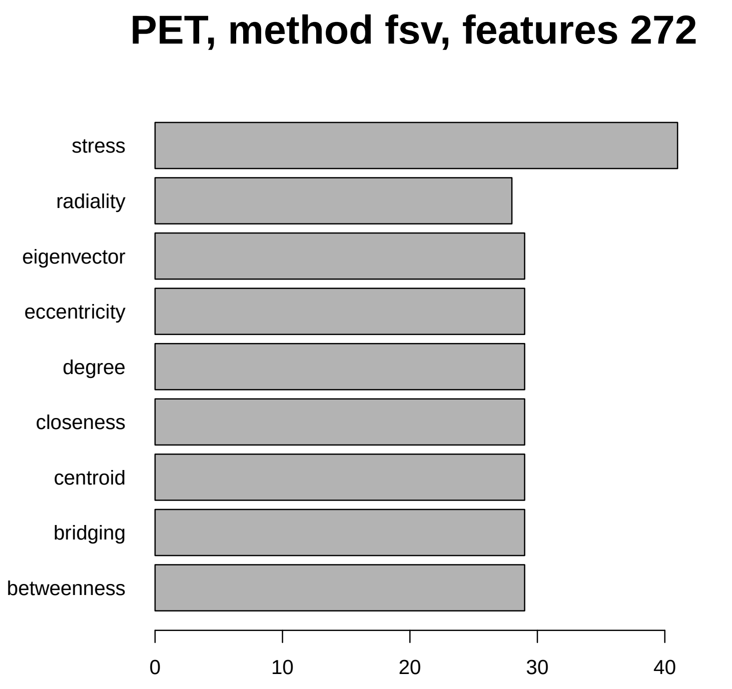
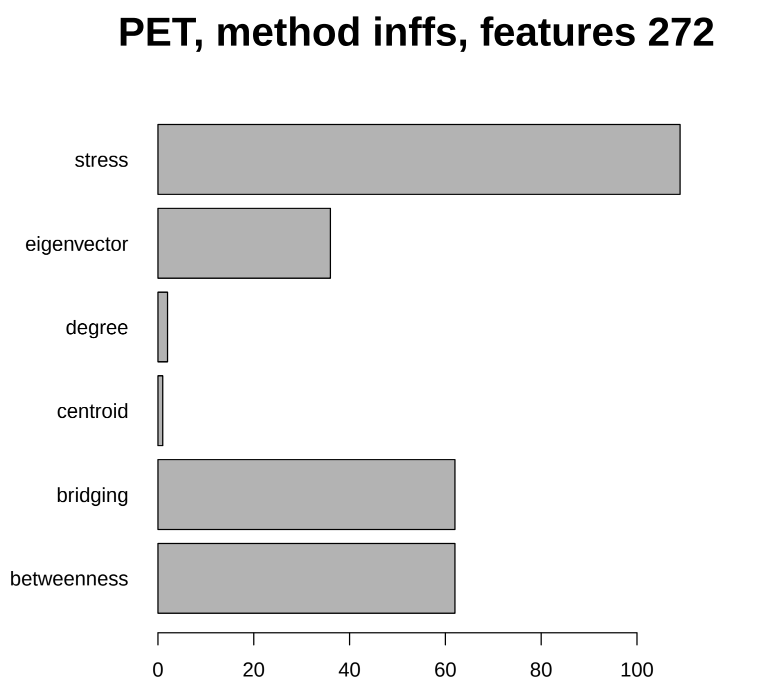


Figure A.25: Histograms that shows all the feature selections algorithms that were run.



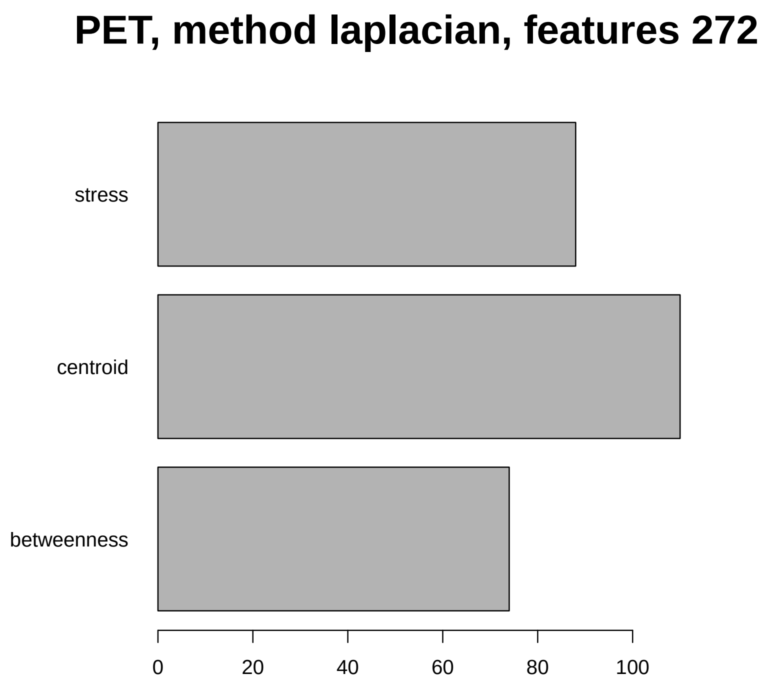
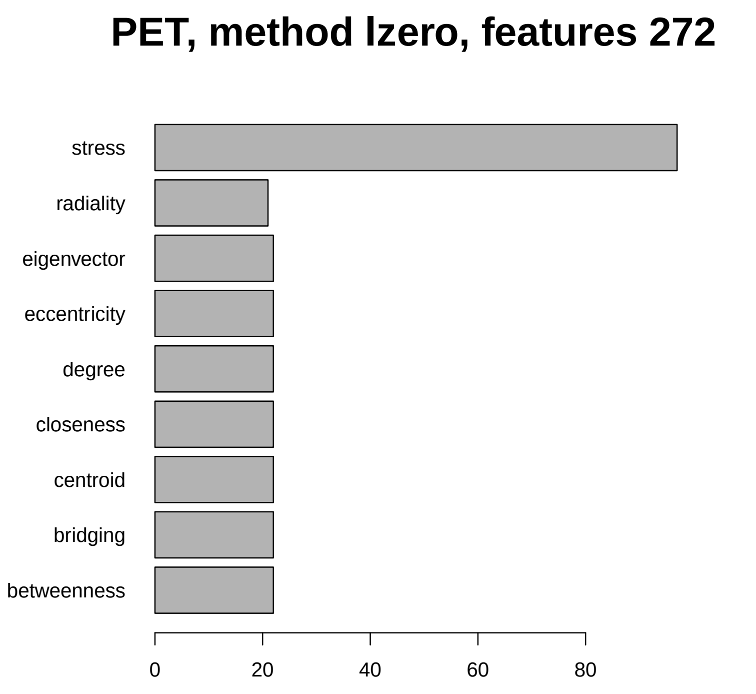


Figure A.26: Histograms that shows all the feature selections algorithms that were run.



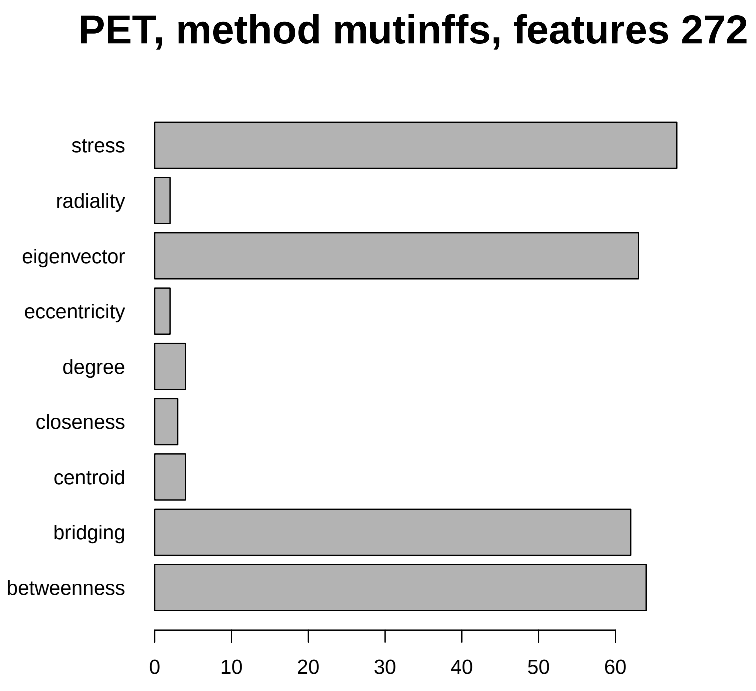
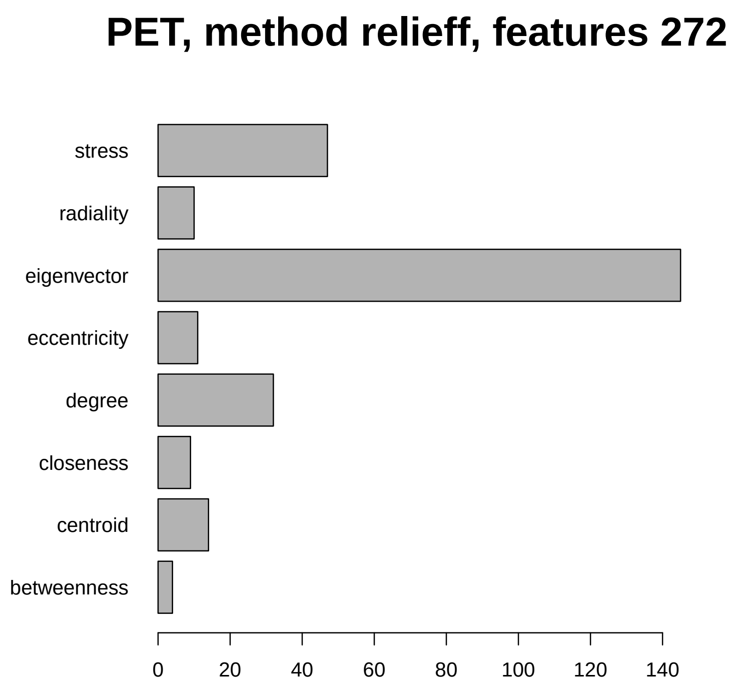


Figure A.27: Histograms that shows all the feature selections algorithms that were run.



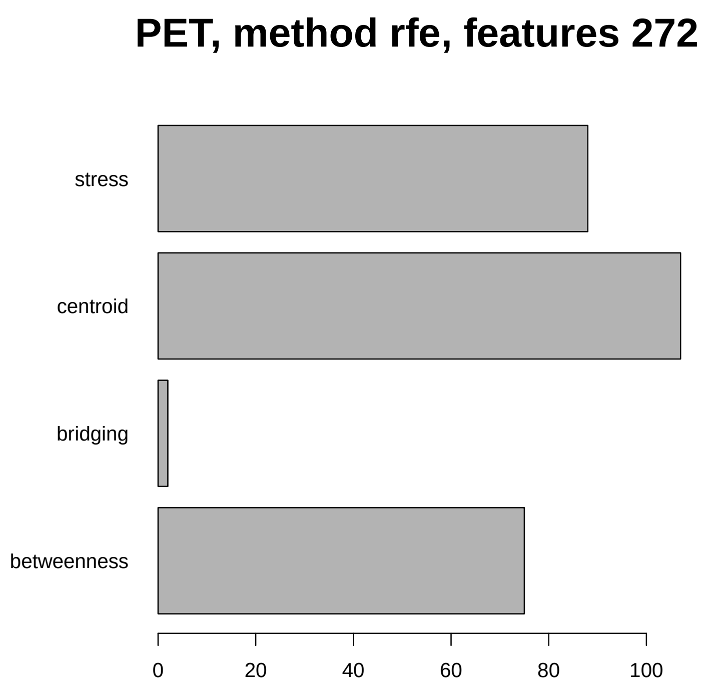


Figure A.28: Histograms that shows all the feature selections algorithms that were run.