UNIVERSITY OF PAVIA

DOCTORAL THESIS

Causal Network Inference of High-Throughput Data with Structural Equation Models

Author:
Barbara TARANTINO

Supervisor:
Prof. Paolo GIUDICI
Prof. Mario GRASSI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

XXXV CYCLE
Electronics, Computer Science and Electrical Engineering

Declaration of Authorship

I, Barbara TARANTINO, declare that this thesis titled, "Causal Network Inference of High-Throughput Data with Structural Equation Models" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Discipline not just determines success but determines how long you can stay successful."

C Muthu Palaniappan

Abstract

Biological and therapeutic uses for network-based approaches to human illness treatment are numerous. A deeper understanding of the role of cellular interconnection in disease progression may lead to the identification of disease genes and disease pathways, which in turn could lead to the development of better therapeutic targets. Scalable statistical solutions for modeling complex biological systems have become critically important with the introduction of high-throughput sequencing (HTS) in molecular biology and medicine. In order to test new hypotheses and deepen our understanding of physiological processes and diseases, we need to incorporate vast amounts of newly collected heterogeneous data and current knowledge. This difficulty was brought on by the growing number of platforms and potential experimental scenarios. Despite the fact that network theory gave us a framework to explore the hidden features of biological systems and to describe them, diverse algorithms still have low reproducibility and robustness, depend on user-defined configuration, and are difficult to understand. This thesis is divided into seven chapters, including an introduction, a conclusion, and five independent sections that report the related studies. The R package **SEMgraph**, which combines network analysis and causal inference within the context of Structural Equation Modeling (SEM), is proposed in *Chapter 1*. It offers a completely automated framework for managing complex biological systems as multivariate networks, ensuring adaptability and accuracy through data-driven model construction and perturbation evaluation, and making it simple to understand in terms of causal relationships between system components. For the analysis of high-dimensional networks, **SEMgraph** provides a number of algorithms. In particular, *Chapter 2* introduces `SEMgsa()`, a topology-based algorithm created within the SEM framework. It uses statistics of route perturbations and topological information to disclose biological information. Compared to some other approaches, `SEMgsa()` outperforms current software tools and is very sensitive to the disease-specific pathways. `SEMtree()`, a tree-based structure learning approach with SEM, is introduced in *Chapter 3*. Starting with the data on the interactome and gene expression, it recovers the tree-based structure. `SEMtree()`, compared to other methods, is able to capture biologically significant sub-networks with straightforward directed route visualization, effective perturbation extraction, and good classifier performance. `SEMbap()`, a two-stage deconfounding method included into the SEM framework and based on the Bow-free Acyclic Paths (BAP) search, is covered in *Chapter 4* of the thesis. It deals with unobserved confounding factors to correctly quantify interesting biological signals. When compared to previous approaches, the BAP search algorithm is able to accurately find hidden confounding while limiting the false positive rate, attaining acceptable fitting and perturbation metrics, and other desirable characteristics. In the end, *Chapter 5* presents the `SEMdag()` algorithm, a two-stage order-based search with prior knowledge-based or data-driven approach, under the assumption of a linear SEM with equal variance error terms. Our methodology has been compared to existing literature, showing low computational burden and high classification performance in out-of-sample disease predictions.

Contents

Declaration of Authorship iii

Abstract vii

Introduction 1

1 SEMgraph 7
1.1 Structural equation models (SEM) framework . . . 7
1.1.1 SEM basics . . . 7
1.1.2 SEM fitting . . . 8
1.1.3 Global model evaluation . . . 9
1.1.4 Evaluating system perturbation . . . 10
1.1.5 Decomposition of effects . . . 13
1.1.6 Graph-weighting methods . . . 14
1.1.7 Network clustering and scoring . . . 17
1.2 Knowledge-based model improvement in Amyotrophic Lateral Sclerosis (ALS) . . . 19
1.2.1 SEM fitting functions . . . 19
1.2.2 Total effect estimation . . . 22
1.2.3 Model estimation strategies . . . 23
1.2.4 Communities and factor scores . . . 27
1.3 Network interrogation in Frontotemporal Dementia (FTD) . . . 29
1.3.1 Gene set analysis . . . 29
1.3.2 Fitting active disease modules . . . 31
1.3.3 Locating differentially connected genes . . . 33

2 SEMgsa( ) 37
2.1 Background . . . 37
2.2 Method and implementation . . . 39
2.2.1 SEM framework . . . 39
2.2.2 User interface . . . 42
2.3 Experimental design . . . 43
2.3.1 Benchmark data . . . 43
Coronavirus disease (COVID-19) . . . 43
Frontotemporal Dementia (FTD) . . . 44
2.3.2 Data simulations . . . 44
Pathway deregulation . . . 45
Pathway enrichment methods . . . 46
Evaluation measures . . . 48
2.4 Results . . . 48
2.4.1 Benchmark results . . . 48
2.4.2 Simulation results . . . 49
2.5 Discussion . . . 52

2.6 Conclusions . . . 54

3 SEMtree( ) 55
3.1 Background . . . 55
3.2 Method and implementation . . . 56
3.2.1 Causal tree recovery . . . 57
3.2.2 User interface . . . 59
3.3 Experimental design . . . 59
3.3.1 Benchmark data . . . 59
3.3.2 Subnetwork detection methods . . . 61
3.3.3 Tree (CAT) extraction . . . 63
3.3.4 Evaluation metrics . . . 64
3.3.5 Data simulations . . . 66
3.4 Results . . . 66
3.4.1 Benchmark results . . . 66
3.4.2 Simulation results . . . 71
3.5 Discussion . . . 74
3.6 Conclusions . . . 75

4 SEMbap( ) 77
4.1 Background . . . 77
4.2 Method and implementation . . . 79
4.2.1 SEM . . . 79
4.2.2 BAP deconfounding . . . 80
4.2.3 PCA deconfounding . . . 82
4.2.4 Spectral transformation . . . 83
4.2.5 Gaussian Graphical Models . . . 84
4.2.6 User interface . . . 85
4.3 Experiments . . . 86
4.4 Metrics . . . 91
4.5 Results . . . 92
4.6 Discussion . . . 100
4.7 Conclusions . . . 102

5 SEMdag( ) 103
5.1 Background . . . 103
5.2 Method and implementation . . . 104
5.2.1 Graphical and structural equation models . . . 104
5.2.2 Structure learning methods . . . 106
5.2.3 SEMdag algorithm . . . 112
5.3 Experimental design . . . 117
5.3.1 Benchmark data . . . 117
5.3.2 DAG structure recovery . . . 118
5.3.3 Disease prediction . . . 121
5.3.4 Evaluation metrics . . . 122
5.4 Results . . . 123
5.5 Discussion and conclusions . . . 126

Conclusions 129

A Supplementary material **131**
A.0.1 SEMgraph 131
A.0.2 SEMgsa 133
A.0.3 SEMtree 133
A.0.4 SEMbap 133
A.0.5 SEMdag 133

Bibliography **135**

List of Abbreviations

ACE **A**verage **C**ausal **E**ffect
AD **A**lzheimer's **D**isease
AIC **A**kaike's **I**nformation **C**riterion
ALS **A**myotrophic **L**ateral **S**clerosis
ANN **A**rtificial **N**eural **N**etwork
BAP **B**ow-free **A**cyclic **P**ath
BRCA **BR**east **CA**ncer
BU **B**ottom **U**p
CAMERA **C**orrelation **A**djusted **M**ean rank gene set test
CAT **Ca**usal **A**dditive **T**rees
cDNA **c**omplementary **D**Deoxyribo**N**ucleic **A**cid
CFA **C**onfirmatory **F**actor **M**odel
CGGM **C**onstrained **G**aussian **G**raphical **M**odel
CLE **C**hu-**L**iu–**E**dmonds' algorithm
COVID-19 **CO**rona**VI**rus **D**isease of 20**19**
CPDAG **C**ompleted **P**artially **D**irected **A**cyclic **G**raph
CpG 5′—**C**—**p**hosphate—**G**—3′
DAG **D**irected **A**cyclic **G**raph
DCI **D**ifference **C**ausal **I**nference
DE **D**irect **E**ffect
DEG **D**ifferentially **E**xpressed **G**ene
df **d**egrees of **f**reedom
DNA **D**Deoxyribo**N**ucleic **A**cid
DRE **D**ifferentially **R**egulated **E**dge
DRN **D**ifferentially **R**egulated **N**ode
FCS **F**unctional **C**lass **S**coring
FDA **F**isher **D**iscriminant **A**nalysis
FDR **F**alse **D**iscovery **R**ate
FN **F**alse **N**egative
FP **F**alse **P**ositive
FTD **F**ronto**T**emporal **D**ementia
gLPCA **g**raph **L**aplacian **P**rincipal **C**omponent **A**nalysis
GSA **G**ene **S**et **A**nalysis
GSEA **G**ene **S**et **E**nrichment **A**nalysis
HRPD **H**uman **P**rotein **R**eference **D**atabase
HTS **H**igh **T**hroughput **S**equencing
IE **I**ndirect **E**ffect
KBM **K**nowledge **B**ased **M**odel
LASSO **L**east Absolute **S**hrinkage and **S**election **O**perator
LRT **L**ikelihood **R**atio **T**est
MANOVA **M**ultivariate **AN**alysis **O**f **VA**riance
MCC **M**atthews **C**orrelation **C**oefficient

MDD **M**ajor **D**epressive **D**isorders
MDS **M**ulti **D**imensional **S**caling
MLE **M**aximum **L**ikelihood **E**stimate
miRNA **m**icro **R**ibo**N**ucleic **A**cid
mRNA **m**essenger **R**ibo**N**ucleic **A**cid
MST **M**inimum **S**panning **T**ree
NGS **N**ext **G**eneration **S**equencing
NP-hard **N**on-deterministic **P**olynomial-time **hard**
ORA **O**ver-**R**epresentation **A**nalysis
O-set **O**ptimal adjustment **set**
PCA **P**rincipal **C**omponent **A**nalysis
pFDA **p**enalized **F**isher's **D**iscriminant **A**nalysis
PPI **P**rotein–**P**rotein **I**nteraction
RF **R**andom **F**orest
RICF **R**esidual **I**terative **C**onditional **F**itting (RICF)
RNA **R**ibo**N**ucleic **A**cid
RNA-seq **R**ibo**N**ucleic **A**cid **seq**uencing
SARS-CoV-2 **S**evere **A**cute **R**espiratory **S**yndrome **CO**rona**V**irus **2**
SDs **S**tandard **D**eviations
SE **S**tandard **E**rror
SEM **S**tructural **E**quation **M**odeling
SHD **S**tructural **H**amming **D**istance
SRMR **S**tandardized **R**oot **M**ean **S**quare **R**esidual
ST **S**teiner **T**ree
STEMI **ST** **E**levation **M**yocardial **I**nfarction
SVD **S**ingular **V**alue **D**ecomposition
SVM **S**upport **V**ector **M**achine
TAHC **T**ree **A**gglomerative **H**ierarchical **C**lustering
TB **T**opology-**B**ased
TD **T**op-**D**own
TE **T**otal **E**ffect
tiRNA **t**RNA-derived stress-**i**nduced **R**ibo**N**ucleic **A**cid
TL **T**opological **L**evel
TN **T**rue **N**egative
TO **T**opological **O**rder (vertex)
TP **T**rue **P**ositive
WTC **W**alkTrap **C**ommunity Detection algorithm
1LV **L**atent **V**ariable model
1CV **C**omposite **V**ariable model
1UV **U**nobserved **V**ariable model

Dedicated to my family . . .

Introduction

The main goal of biomedical research is to identify and comprehend the mechanisms behind complex phenotypic features (Barabási and Loscalzo, 2011). The majority of cellular components in humans, like those in other species, carry out their jobs through interactions with other cellular components; the sum of these interactions is known as the human interactome. The number of distinct proteins and functional RNA molecules that make up the nodes of the interactome easily exceeds 100.000, and there are an estimated 25.000 protein-encoding genes, 1.000 metabolites, and an undetermined number of metabolites and different proteins in this network. This network's components act as the linkages of the interactome, and it is believed that there are many more functionally significant interactions between them. This subcellular interconnectedness suggests that the effects of a particular genetic anomaly do not just affect the activity of the gene product that it is present in, but can also propagate down the connections of the network and affect the activity of genes products that are otherwise healthy. Consequently, the phenotypic impact of a defect is not exclusively defined by the known function of the mutant gene, but also by the functions of the components with which the gene and its products interact as well as of the partners with whom they connect, i.e., by its network context. The phenotype of a disease reflects numerous pathobiological mechanisms that collaborate in a complicated network. This theory's consequence is that the interdependencies between the molecular constituents of a cell result in profound functional, molecular, and structural changes and causal connections between seemingly unrelated traits.

Understanding biological network structure. Networks of molecules, including genes, proteins, and metabolites, are crucial in molecular biology (Oates, 2012). A biological network can be represented as a graph, $G = (V, E)$, where V stands for molecular components, and E represents the regulatory connections between those components. The nodes in a gene regulatory network (Babu MM, 2004; Davidson, 2001) represent genes, and the edges represent transcriptional regulation, while the nodes in a protein signaling network (Yarden and Sliwkowski, 2001) represent proteins, and the edges could represent the parent's enzymatic influence on the child's biochemical state, such as via phosphorylation. The edge structure of the network may be ambiguous in various biological circumstances, including illness states (for instance, as a result of genetic or epigenetic changes). Then, a crucial biological objective is to describe the edge structure (commonly referred to as the "topology" of the network) in a context-specific way, that is, utilizing data collected in the biological context of interest (for example, a form of cancer, or a developmental state). Such data-driven characterisation of biological networks has attracted a lot of interest as a result of developments in high-throughput data capture. Statistical methods are becoming more and more crucial in these "network inference" projects. The objective can be understood statistically as drawing conclusions about the edge structure, E in light of the biochemical data, Y.

Identifying disease genes and pathways. Network-based methods for treating human disease offer a variety of biological and therapeutic applications. Uncovering disease genes and disease pathways, which in turn could provide better targets for drug development, could result from a greater knowledge of the effects of cellular inter-connectivity on disease progression. The development of improved and more precise biomarkers for monitoring the functional integrity of the network disrupted by diseases, as well as improvements in disease classification, could also transform clinical practice and pave the path for individualized medicines and treatments.

Evaluating knowledge-based models. The High Throughput Sequencing (HTS) era brought to a deeper understanding of the true complexity underlying diseased (and generally phenotypical) features, ushering in the big data age in molecular biology and medicine (Shendure and Aiden, 2012). The availability of curated biological models is no longer a restriction given the enormous number of publicly accessible bio-medical datasets. The availability of organized biochemical and biomedical data, which can easily be transformed into networks and statistical models and is what we typically refer to as knowledge-based models (KBMs), is the essential characteristic of these databases. KBMs serve as a foundation and the industry standard for enhancing exploratory techniques, but they have some serious flaws (Ritchie et al., 2015). In order to evaluate particular biological hypotheses and mechanisms, it is essential to define a set of criteria for converting a KBM into a causal model. However, this is not always straightforward due to missing data or varying levels of evidence (such as experimental evidence versus inference from similarity or electronic annotation). Second, a KBM reflects the state of the art, which is continually being tested by fresh experimental evidence that could disclose brand-new interactions and pathways. The original causal model must also be evaluated using explicit statistical criteria that reflect the biological characteristics of the system and enhance the model's descriptive and prediction abilities (Liu et al., 2020; Ritchie et al., 2015).

Providing powerful statistical techniques. Investigations of complex functional networks involving DNA, RNA, proteins, and other biological constituents in a living cell often adhere to a standard methodology. A cDNA microarray is used to assess the mRNA level following the completion of a DNA sequence in order to disclose the gene expression profiles under diverse circumstances. With this knowledge, strong statistical techniques are needed for extensive quantitative comparisons of populations/conditions as a result of the ongoing development of high-throughput sequencing technologies. For instance, one of the most important tasks in the analysis of gene expression data is to select genes from a long gene list that express differently under various contexts, such as various disease statuses. See Robinson, McCarthy, and Smyth, 2009 and Love, Huber, and Anders, 2014, and the references therein for a variety of ways that have been suggested for this purpose and that offer crucial insights into the molecular basis of many complex human features and disorders.

Exploiting the SEM modelling approach. The Structural Equation Modelling (SEM) approach is one option. SEM has been used effectively to clarify causal linkages in diverse domains like sociology, psychology, and econometrics. It has been approximately 80 years since Wright, 1921, and Wright, 1934 first proposed the idea of SEM. He established a method of decomposing the observed correlations into a system of equations that mathematically reflected his theories regarding causal links while focusing on patterns of covariation between different features. His approach became known as "path analysis" and these connections between the variables were shown

in a "path diagram" (Wright, 1921; Wright, 1934). Later, economists and sociologists separately rediscovered and developed this method. The most notable examples are Jöreskog, 1973; Jöreskog and Sörbom, 1982. By combining factor analysis and path analysis, they created a new technique called "structural equation modeling," which can evaluate causal claims rather than just describe them (Shipley, 1999). This method replaced Wright's original "path analysis" technique. SEM analyzes causal linkages in observational data under the assumption of linear relationships, while non-linear correlations can also be modeled. These methodologies aid in the selection of pertinent hypotheses by eliminating those that lack supporting empirical data, even though they do not actually establish causality. Although a correlation between two variables does not always imply a causal connection between them, a causal connection between two variables does suggest the presence of a correlation between them. The SEM strategy's fundamental premise is as follows. In order to create a theoretical covariance structure between a vector of random variables, SEM presupposes the existence of an underlying mechanism. The goal is to put out and evaluate a model that adequately expresses the fundamental nature of this underlying mechanism (Malaeb, Summers, and Pugesek, 2000). A number of constraints on the variance/covariance matrix are implied by the causal links defined in a starting hypothesis. If the variance/covariance matrix derived from observational data complies with the restrictions given by the hypothesis, the model is not discarded. The various kinds of causal interactions that can cause two variables to covariate must be recognized in order to comprehend the SEM process. Basically, four main categories can be identified:

- the direct effect of one variable on another;
- indirect effects: one variable affects another variable through a direct path of intermediate variables;
- common causes: X affects both Y and Z (this is spurious association);
- correlated causes: X is a cause of Z and X is correlated with Y;
- reciprocal causation: each variable is a cause and effect of the other.

A SEM is a multivariate regression model that goes beyond conventional regression by allowing numerous outcomes, often known as "endogenous" variables and "latent" variables that are not observed in the data. There is a corresponding regression equation for each endogenous variable, and these equations might depend on both exogenous and other endogenous factors. The predictor variables (covariates) that are not influenced by any other variable in the model are referred to as "exogenous" variables in this context.

Implementing an efficient and user-friendly R toolkit. The challenge is to update and test network models using a straightforward and transparent workflow, starting from current knowledge. From a computational standpoint, the difficulty is to free the user from initial setup selection by immediately determining the algorithm and model parameters from quantitative data using effective and parallelizable methods.

Chapter 1 exposes the main contribution of this thesis to the existing literature, i.e. the development of the R package **SEMgraph** (Grassi, Palluzzi, and Tarantino, 2022), based on SEM (Bollen, 1989), enabling causal inference on complex biological networks. The use of SEM is now currently common in the fields of causal inference and causal discovery (Pearl, 2009). Model learning, data-driven model refinement,

causal inference, and discovery are all supported by path diagrams, which are frequently represented as acyclic mixed graphs. HTS data is frequently organized into networks or pathways, allowing for the confirmatory or exploratory examination of key biological characteristics.

In **SEMgraph**, this is achieved effectively through algorithm-assisted search for the best trade-off between the best model fitting (i.e., the optimal context) and perturbation (i.e., external influence) given data, in which knowledge is employed as supplemental confirmatory information. As a collection of interdependent variables connected by causal links, the input network and the underlying statistical model in **SEMgraph** are interchangeable representations of the same framework. Through a series of intermediate steps, including causal backbone estimation, adjustment of hidden confounding variables, graph extension, and model refinement to improve fitting, with scalable solutions for large graphs, this dual representation is effectively modified to produce the final causal model.

The other chapters report a more detailed analysis of some of the most relevant **SEMgraph** algorithms: `SEMgsa()` (*Chapter 2*), `SEMtree()` (*Chapter 3*), `SEMbap()` (*Chapter 4*) and `SEMdag()` (*Chapter 5*). These chapters aim to present novel algorithms and provide a meaningful comparison of the state-of-the-art methods on real and synthetic data. Applications involve RNA-seq and gene expression data about cases and control subjects. In detail, the contributions are presented in four self-contained chapters:

- *Chapter 2: identifying disease-specific biological functions*. By allowing extensive monitoring of a biological system, techniques like high-throughput sequencing and gene/protein profiling have revolutionized biological research. The examination of high-throughput data often results in a list of differentially expressed genes or proteins, regardless of the technique employed. This list is quite helpful for locating genes that could be involved in a specific phenomena or phenotype. For many researchers, however, this list frequently falls short of offering mechanistic insights into the underlying biology of the illness under study. One way to tackle this problem is to make analysis simpler by breaking up long lists of individual genes into smaller groups of similar genes or proteins, lowering the complexity of the analysis. For two main reasons, functional level analysis of high-throughput molecular data is particularly interesting. First, simplifying hundreds of genes, proteins, and/or other biological molecules into groups according to the routes they participate in simplifies the complexity to just a few hundred pathways for the experiment. Second, finding active pathways that are different between two conditions may have a stronger explanatory impact than simply compiling a list of various genes or proteins. We suggest `SEMgsa()`, a topology-based algorithm built inside the structural equation models framework. After statistically fitting for the biological relationships between the genes inside pathways, `SEMgsa()` combines the SEM p-values for node-specific group effect estimates in terms of activation or inhibition. In order to find biologically significant results in a frontotemporal dementia (FTD) DNA methylation dataset (GEO accession: GSE53740) and a COVID-19 RNA-seq dataset (GEO accession: GSE172114), we employed `SEMgsa()` and evaluated its performance against various other approaches.

- *Chapter 3: discovering regulatory and signaling mechanism*. The complex interactions between the various biological components of a cell give rise to the biological function at the molecular level (Emmert-Streib, 2007; Emmert-Streib and Glazko, 2011). Specifically, depending on an organism's tissue type and

environment, many molecules, such as proteins, metabolites, miRNA, and tiRNA, can interact with one another in a wide variety of ways. Three different types of networks—metabolic, transcriptional regulatory, and protein interaction networks—can be used to broadly classify the connections between biological entities (Förster et al., 2003; Vidal, Cusick, and Barabasi, 2011). These networks must be inferred from experimental data produced by various high-throughput platforms, like as microarrays, proteomics, and next-generation sequencing (NGS). Nowadays, it is well acknowledged that biological networks do not connect at random but rather follow specific structural patterns that result in (I) a scale-free topology, (II) a hierarchical structure, and (III) a modular structure (Han et al., 2004; Regan, 2009). The functional activity of modules in various illness situations has also been identified using a number of approaches that have been developed to detect and integrate protein networks with gene expression or other datasets like disease-gene association (Dittrich et al., 2008; Taylor et al., 2009). Trees and arborescences are useful models as a preliminary step towards understanding the overall dependence structure of high-dimensional data. These are unrealistically simple models for biological systems, but can nevertheless provide useful insights. We can use trees with different objectives: (i) to search for distinct connected components, which can be analysed separately (dimension reduction); (i) to identify neighbourhoods for more detailed analyses, or (ii) to identify other interesting features, such as hub (i.e., nodes of high degree) that may play a special role in the true network. The primary contribution is the creation of a self-contained tree-based structure learning method that was integrated into the SEM framework with the `SEMtree()` function. As well as simulated datasets with varied differential expression patterns, we used `SEMtree()` to analyze the RNA-seq data from the COVID-19 virus disease (GEO accession: GSE172114).

- *Chapter 4: obtaining unbiased causal estimates*. The SEM's structure can be represented as a directed graph, with the vertices being variables and the edges denoting direct causal connections. The directed acyclic graph (DAG) is created when the structure is thought to be recursive (acyclic). Many structure learning methods use DAGs as models of conditional independence to locate all DAGs that are consistent with the observed conditional independencies (Spirtes, Glymour, and Scheines, 2000). But frequently, not all important variables are noted. Although some conditional independencies may still be satisfied by the resulting marginal distribution over the observed variables, generally speaking, these will not have a DAG representation (Richardson and Spirtes, 2002). We take a look at a class of models that can contain some hidden variables. To be more precise, we assume that the graph over the measured variables is a bow-free acyclic path diagram (BAP). This indicates that it can have both directed and bidirected edges (the bidirected component being acyclic), with the bidirected edges denoting hidden confounders and the directed edges denoting direct causal effects. Due to the bow-freeness constraint, an edge between two variables cannot be both directed and bidirected. The main contribution is the development of a two-stage deconfounding method based on BAP search that was integrated into the SEM framework with the `SEMbap()` function. For the simulations, we replicate different structures for the hidden covariance matrix and, for real data, we make use of the (pre-processed) breast cancer (BRCA) RNA-seq dataset from TCGA project, also analyzed in Jablonski et al., 2021.

- *Chapter 5: discovering the optimal causal structure*. A formal representation of the

interactions between the observable variables, such as a casual graph, is crucial for causal inference, or the process of quantifying the influence of a cause on its consequence. A DAG offers an elegant way to describe directional or causal structures among collected nodes. Learning the DAG structures from observable data has received a lot of attention recently. Structure learning is well known to be computationally challenging, and various methods have been developed to solve it, using one of three alternative approaches: score-based algorithms (Chickering, 2003; Yuan et al., 2018), constraint-based algorithms (Spirtes, Glymour, and Scheines, 2000) and hybrid algorithms. The majority of the methods listed above, however, can only recover a DAG's Markov equivalence class. There has recently been a lot of interest on exact DAG recovery. Various DAGs in the same equivalence class can be distinguished by additional assumptions about the data distribution than just conditional independence relations. The main contribution is the development of a two-step algorithm for learning high-dimensional sub-Gaussian linear SEMs with the same error variances (Peters and Bühlmann, 2014), implemented in the `SEMdag()` function. The extracted DAG is estimated using the two-step order search approach. First a vertex (node) or level (layer) order of p nodes is determined, and from this sort, the DAG can be learned using in step 2) penalized (L1) regressions (Shojaie and Michailidis, 2010). To investigate the utility of our approach, we conducted a series of experiments using RNA-seq data, taking into account a pair of training and testing datasets for four distinct diseases: Amyotrophic Lateral Sclerosis (ALS), BRCA, COVID-19 and ST-elevation myocardial infarction (STEMI). Comparisons in terms of disease predictive performance have been made with: (i) state-of-art structure discovery methods and (ii) a traditional supervised learning algorithms (Random Forest (RF)).

In the end, the ultimate goal of our research is to infer complex biological networks from gene expression data, enhancing the understanding of gene networks and discovering novel biological relationships in order to characterize and prevent specific diseases.

Chapter 1

SEMgraph

1.1 Structural equation models (SEM) framework

1.1.1 SEM basics

SEM is a statistical framework for causal inference based on multivariate linear regression equations, where the response variable in one regression equation may appear as a predictor in another equation (Bollen, 1989; Shipley, 2016). SEM may be formulated to explicity include latent unobserved variables, but here we consider a setup in which the latent variable have been marginalized out and represented in the model only implicitly through possible correlations among unobserved latent confounders (Pearl, 1998).
A SEM, is based on a system of structural (i.e., linear regression) equations definig a *path diagram*, represented as a graph $G = (V, E)$, where V is the set of nodes (i.e., variables) and E is the set of edges (i.e., connections). The set E may include both directed edges $k \to j$ if $k \in \text{pa}(j)$ and bidirected edges $k \leftrightarrow j$ if $k \in \text{sib}(j)$, where the *parent* set $\text{pa}(j)$, and the *siblings* set $\text{sib}(j)$, determine the system of linear equations, as follows:

$$Y_j = \sum_{k \in \text{pa}(j)} \beta_{jk} Y_k + U_j \qquad j \in V \tag{1.1}$$

$$\text{cov}(U_j; U_k) = \begin{cases} \psi_{jk} & \text{if } j = k \text{ or } k \in \text{sib}(j) \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

where Y_j and U_j are an observed variable and an unobserved error term, respectively; β_{jk} is a regression (path) coefficient, and a covariance ψ_{jk} indicates that errors are dependent, which is assumed when there exists an unobserved (i.e.latent) confounder between k and j.
A path diagram is also a formal tool to evaluate the hierarchical structure of a system, where we can identify *exogenous variables* as system elements with empty parents set, and *endogenous variables*, having at least one parent variable in at least one structural equation of the SEM. In graph theory, exogenous variables are *source* nodes, with incoming connectivity equal to 0, whilst endogenous variables are nodes with non-zero incoming connectivity. Endogenous variables can be further divided into *connectors*, with non-zero outgoing connectivity, and *sinks*, having no outgoing connections. Given these notions, we consider three types of fundamental path diagrams to describe high-throughput data structure:

- Directed Acyclic Graphs (DAGs), composed by directed edges ($k \to j$) only, whose magnitude is quantified through path coefficients β_{jk}, and all covariances are null (i.e., $\psi_{jk} = 0$). In addition, loops are not allowed in a DAG.

- Bow-free Acyclic Paths (BAPs), having acyclic directed edges ($k \rightarrow j$), and bidirected connections ($k \leftrightarrow j$) only if the k-th and j-th variable do not share any directed link (i.e., they are bow-free). As a consequence, in a BAP, if $\beta_{jk} \neq 0$ then $\psi_{jk} = 0$.
- Covariance models, as a special case of BAP in which all $\beta_{jk} = 0$. Therefore, only covariances ψ_{jk} may have non-zero values.

These three models are simple graphs; i.e., they have at most one edge between any pair of nodes, and are all identifiable, such that the parameter matrices B and Ψ can be uniquely estimated from the population covariance matrix of the observed variables for nearly every parameter choice (Brito and Pearl, 2002; Drton, Foygel, and Sullivant, 2011).

1.1.2 SEM fitting

From the computational point of view, it is convenient to write Equations 1.1 and 1.2 in matrix form as: $Y = BY + U$ and $\text{cov}(U) = \Psi$. Assuming random variables with zero mean vector ($\mu(\theta) = 0$), the covariance matrix of the joint distribution of p variables Y is given by:

$$\Sigma(\theta) = (I - B)^{-1}\Psi(I - B)^{-T} \tag{1.3}$$

where the set of free parameters $\theta = (\beta; \psi)$ has dimension t. B is the path coefficient matrix, Ψ is the covariance matrix, and I is the identity matrix, all of them having dimension $p \times p$. Generally, in the SEM framework, free (i.e., unknown) parameters θ are computed by Maximum Likelihood Estimation (MLE). Assuming all model variables as jointly gaussian, a covariance-based procedure is performed, so that the estimated covariance matrix $\Sigma(\hat{\theta})$ is close to the observed sample covariance matrix S. This is obtained by maximizing (up to an additive constant) the model log-likelihood function $\log L(\theta)$ given data (Bollen, 1989, p. 135).

$$\underset{\theta \in (B;\Psi)}{\arg\max}\ \log L(\theta) = -\frac{n}{2}(\log\det\Sigma(\theta) + \text{tr}[\Sigma(\theta)^{-1}S]) \tag{1.4}$$

To maximize the objective function in (1.4) an optimization method is needed. Popular choices include Newton–Raphson, Fisher scoring, and various quasi-Newton methods. The **lavaan** package (Rosseel, 2012) uses by default a quasi-Newton method implemented in the `nlminb()` function. Standard Errors (SEs) are extracted from the diagonal of the expected (or observed) Fisher's information matrix of the likelihood function. MLEs, $\hat{\theta}$'s of the parameters, θ's has been shown asymptotically $N(0,1)$ and efficient (minimum variance):

$$\frac{(\hat{\theta}_{jk} - \theta_{jk})}{SE(\hat{\theta}_{jk})} \rightarrow N(0,1) \tag{1.5}$$

P-values for testing the null hypothesis, $H_0 : \theta_{jk}$ are computed through the test statistic, $z = \hat{\theta}/\text{SE}(\hat{\theta})$, and 95% confidence intervals are: $\hat{\theta} \pm 1.96\,\text{SE}(\hat{\theta})$. An advantage of MLE is that its estimates are in general scale invariant and scale free (Bollen, 1989, p. 109). Therefore, the values of the fit function do not depend on whether correlation or covariance matrices are analyzed, and whether original or transformed data are used.
For large graphs with the node size, ($|V| > 100$), SEs computation will be skipped,

and parameter estimates will be computed through residual iterative conditional fitting (RICF) via the **ggm** R package (Marchetti, Drton, and Sadeghi, 2020). The RICF solver is an efficient iterative algorithm, that can be implemented using only least squares computations, has clear convergence properties and yields exact MLE after the first iteration whenever the MLE is available in closed form (Drton, Eichler, and Richardson, 2009). A large number of bootstrap samples, $b = 1, ..., B$ are independently drawn , and the SEs are computed as the Standard Deviations (SDs) of the parameter estimates evaluated for each bootstrap sample (Bollen and Stine, 1992).

If the graph is a DAG, an alternative to NLMINB or RICF *covariance*-based algorithms for large graphs ($V > 100$), consists into determining the MLE parameters via *nodewise*-based regression, i.e. by least-square (LS) procedure repeatedly with a single response variable (a mediator or sink node) each time. In the regime $V >> n$, a LASSO regression (Tibshirani, 1996) procedure and de-biasing asymptotic inference can be implemented. Assuming a sparsity assumption, and the non-zero edges is not too large relative to sample size, i.e., the maximum node degree of the graph satisfies, $s = o(\sqrt{n}/log(p))$, the de-sparsified (or de-biased) P-values can be find asymptotically from the $N(0,1)$-distribution via constrained Gaussian Graphical Model (CGGM), as outline in Jankova and Van De Geer, 2019.

SEM fitting can be applied to any graph with the `SEMrun()` function (see help documentation: `?SEMrun`). By default `algo=lavaan"`), the input graph is converted to a "lavaan" model, the data is used to compute the covariance matrix and then SEM parameters is fitted via covariance-based procedure with Newton–Raphson algorithm and $N(0,1)$ P-values. If `algo="ricf"`, covariance-based procedure with RICF algorithm and bootstrap P-values is performed. If `algo="cggm"`, nodewise-based procedure with LS or LASSO and $N(0,1)$ or de-biased $N(0,1)$ P-values is activated.

1.1.3 Global model evaluation

The assessment of the model involves the Likelihood Ratio test (LRT) converted to a Chi-square test of the fitted model. Specifically, let $\Sigma = S$ the observed covariance matrix, and $\Sigma(\hat{\theta})$. the model-implied covariance matrix, the null hypothesis to be tested is: $H_0 : \Sigma(\hat{\theta}) = S$. The chi-square test, also known as model deviance, is then:

$$\chi^2 = -2 \log \text{LRT} = -2 \left[\log L(\hat{\theta}) - \log L(\theta_{\max}) \right] \tag{1.6}$$

where $\log L(\hat{\theta})$ is the log-likelihood Equation (1.4) evaluated to model-implied covariance matrix, $\Sigma(\hat{\theta})$ and $\log L(\theta_{\max})$ is the log-likelihood for an exact fit; i.e., $\Sigma(\hat{\theta}) = S$. P-values are derived either from the $\chi^2(\text{df})$ distribution with $\text{df} = p(p+1)/2 - t$ degrees of freedom, or from a resampling bootstrap distribution (Bollen and Stine, 1992). Non-significant P-values (P > 0.05) indicate that the model provides a good fit to data (i.e., the elements of $S - \Sigma(\hat{\theta})$, should be close to zero). Alternatively to the chi-square test, the Akaike's information criterion (AIC) (Akaike, 1974) can be used to compare fitted to saturated model, defined in SEM as (Bentler, 2016):

$$\text{AIC} = -2 \log L(\hat{\theta}) + 2t \approx \chi^2 - 2\text{df} \tag{1.7}$$

where the rightmost member in Equation (1.7) is equal to the left member minus the constant term $p(p+1)/2$. The model with the minimum AIC value is regarded as the best fitting model. In the chi-square (or deviance) metric it has been suggested that a ratio between the magnitude of χ^2 and the expected value of the sample distribution $E(\chi^2) = \text{df}$ less than 2 and between 2 and 3 is indicative of a good and acceptable data-model fit, respectively (Schermelleh-Engel and Moosbrugger, 2003).

While, for high-dimensional model with large df(large sparsity), $\chi^2/\text{df}<1$ is indicative of over-fitting. The relationship between AIC and χ^2/df thresholds become more evident by comparing Equation (1.6) and Equation (1.7). For the saturated model AIC $= 0$, and the fitted model should be selected if AIC < 0, which is equivalent to the condition $\chi^2/\text{df} < 2$.
Another approximate SEM fit index comparing two models (fitted vs. saturated) as the chi-square test (or the chi-square ratio), is the Standardized Root Mean-squared Residual (SRMR), an overall descriptive statistic based on all pairwise differences between observed sample covariances (s) and implied model covariances ($\hat{\sigma}$):

$$\text{SRMR} = \sqrt{\frac{\sum_{j=1}^{p-1}\sum_{k=j+1}^{p}(s_{jk}-\hat{\sigma}_{jk})^2/s_{jj}s_{kk}}{p(p+1)/2}} \tag{1.8}$$

SRMR values range from 0 to 1, where 0 is equivalent to a perfect fit. The acceptable range for the SRMR index is between 0 and 0.08 (Hu and Bentler, 1999).
If the model is a DAG, a global fitting statistic, based on the directed separation (d-separation) concept, can be applied (Shipley, 2000). In a DAG, missing edges between nodes imply a series of independence relationships between variables (either direct or indirect). These independences are implied by the topology of the DAG and are determined through d-separation: two nodes, Y_j and Y_k, are d-separated by a set of nodes S if conditioning on all members in S blocks all confounding (or *backdoor*) paths between Y_j and Y_k (Pearl, 1998; Verma and Pearl, 1990a). In a DAG, with Y_j having a higher causal order than Y_k, it is possible to find a minimal set of conditional independencies B_U implying all the other possible independencies, defined by: $B_U = \{Y_j \perp Y_k \,|\, \text{pa}(j) \cup \text{pa}(k),\, j > k\}$. The number of conditional independence constraints in the basis set B_U equals the number of missing edges, corresponding to the number of degrees of freedom (df) of the model. If the graph is not very large or very sparse, it is possible to perform local testing of all missing edges separately, using the Fisher's z-transform of the partial correlation. An edge $(k;\, j)$ is absent in the graph when the null hypothesis H_0: $\text{cor}(Y_j; Y_k \,|\, \text{pa}(j) \cup \text{pa}(k)) = 0$ is not rejected. These individual tests implied by the basis set B_U are mutually independent, thus their P-values p_r can be combined in an overall test of the fitted model (i.e., the DAG) using Fisher's statistic:

$$C = -2\sum_{r=1}^{R}\log(p_r) \tag{1.9}$$

This statistic follows a chi-squared distribution with df $= 2\times$ (number of missing edges). A non-significant P-value (P > 0.05) of C indicates that the model provides a good fit to data.
`SEMrun()` function (see help documentation: `?SEMrun`) print on the console deviance/df, and SRMR statistics with all `algo= c("lavaan, "ricf", "cggm")`. While, the C-statistic is implementated in `Shipley.test()` function (see help documentation: `?Shipley.test`)

1.1.4 Evaluating system perturbation

In several applications, the concept of *perturbation* arises when a system is altered (i.e., changed) by one or more external influences affecting its *behaviour* respect to a reference state (often described as *physiological* or *healthy*). However, in most cases, the mechanisms and extent of the alterations are unknown and data-driven discovery based on the comparison between experimental (i.e., altered) and healthy samples is the best possible option.

A possible approach to the evaluation of system perturbation is multigroup SEM (Bollen, 1989, p. 355). We consider a two-group SEM procedure either using an exogenous group variable acting over a common model, or building a separate model for each group and comparing them. In the former, the experimental condition is compared to a control one through the use of an exogenous binary *group* variable $X = \{0, 1\}$ acting on every node of the network. This model is converted to a system of linear equations that is common to both conditions, with $\mu(\theta) = 0$ and $\Sigma(\theta)$ being the implied mean vector and covariance matrix of the *common model*:

$$Y_j = \beta_j X + U_j \qquad j \in V(x) \tag{1.10}$$

$$Y_j = \sum_{k \in \mathrm{pa}(j)} \beta_{jk} Y_k + \beta_j X + U_j \qquad j \in V(y) \tag{1.11}$$

where $V(x)$ and $V(y)$ are the sets of exogenous (i.e., sources) and endogenous (i.e., connectors and sinks) variables, respectively. Coefficients β_j (adjusted by the parents of the j-th node) determine the effect of the group on the j-th node, while the *common* path coefficients β_{jk} represent regression coefficients, adjusted by group effect. This type of SEM enables the identification of differentially regulated nodes (DRNs); i.e., variables showing a statistically significant variation in their activity (e.g., gene expression) in the experimental group respect to the control one.
Alternatively, the two groups of samples are kept separated, with two different systems of linear equations, that is the *two-group model*:

$$Y_j^{(1)} = \sum_{k \in \mathrm{pa}(j)} \beta_{jk}^{(1)} Y_k^{(1)} + U_j^{(1)} \qquad j \in V(y) \tag{1.12}$$

$$Y_j^{(0)} = \sum_{k \in \mathrm{pa}(j)} \beta_{jk}^{(0)} Y_k^{(0)} + U_j^{(0)} \qquad j \in V(y) \tag{1.13}$$

This enables the identification of differentially regulated edges (DREs). We define $\mu_1(\theta) = 0$ and $\Sigma_1(\theta)$ as the model-implied mean vector and covariance matrix for the experimental group (group 1), and $\mu_0(\theta) = 0$ and $\Sigma_0(\theta)$ the corresponding moments for the control group (group 0), respectively.
Perturbation tests in the common-model and two-models approaches are based on the definition of two different test statistics:

- $z_j = \hat{\beta}_j / \mathrm{SE}(\hat{\beta}_j)$, tests the null value for path coefficient, $H_0 : \beta_j = 0$ of the group variable X and z-sign evaluates node activation or inhibition for $j \in V$;
- $z_{jk} = (\hat{\beta}_{jk}^{(1)} - \hat{\beta}_{jk}^{(0)}) / \mathrm{SE}(\hat{\beta}_{jk}^{(1)} - \hat{\beta}_{jk}^{(0)})$, tests the null value for path coefficients difference, $H_0 : \beta_{jk}^{(1)} - \beta_{jk}^{(0)} = 0$ between groups and z-sign evaluates edge activation or inhibition for $(j; k) \in E$.

In both approaches, parameters are estimated through MLEs and the Standard Errors (SEs) are derived from the usual **lavaan** output. Then, for the *two-group* model the group-specific SEs are combined as (Hudson and Shojaie, 2022, p. 355):

$$SE(\hat{\beta}_{jk}^{(1)} - \hat{\beta}_{jk}^{(0)}) = \sqrt{SE(\hat{\beta}_{jk}^{(1)})^2 + SE(\hat{\beta}_{jk}^{(0)})^2} \tag{1.14}$$

P-values for these z statistics are derived asymptotically from the $N(0, 1)$ - distribution under the H_0 for sample sizes, n or n_1 and n_0 sufficiently large, respectively.

In the *common* model an additional node (group) and |V| edges are added in the input graph. For reduce computational time, in some cases, edge weights in B matrix are set to discrete values indicating gene activity derived from biological database (e.g. KEGG). Usually they are: -1 for repressed or inactive, 0 for neutral, and +1 for enhanced or activated. These values can be scaled so that they can be used for model fitting. As a rule of thumb, when fixed weights are in the set [-1, 0, 1], $B = 0.1$ should perform well on any network. Using fixed weights (when this makes sense) reduces significantly computational demand, avoiding regression parameter estimation.
For large graphs ($|V| > 100$) the standard error (SE) computation can be disabled and model parameters will be estimated through residual iterative conditional fitting (RICF). Group effect P-values can be computed by randomization of group labels comparing the estimated parameters by RICF with their random resampling distribution after a sufficiently high number of case/control labels permutations using the R package **flip** (Finos et al., 2018). Accurate small P-value estimations are possible with no need for a large number of permutations (`SEMrun()` makes default = 1000 permutations), using the moment based approximation proposed by Larson and Owen, 2015. Once the empirical distribution of the permuted statistic T is obtained, the two-sided P-values are computed from the normal distribution with mean and standard deviation estimated by the empirical distribution.

In high dimensionality ($|V| > 100$) *two-group* model, if the graph is a DAG, parameters can be estimated through constrained gaussian graphical model (CGGM) solver, and P-values are obtained from $N(0,1)$-distribution combining the group-specific z-tests, using equation 1.14.

Finally, the descriptive overall group perturbation on either nodes or edges can be computed, for both node and edge differences, based on the Brown's method for combining non independent, one-sided significance tests (Brown, 1975). The method computes the sum of one-sided pvalues: $X^2 = -2\sum_j \log(p_j)$, where the direction is chosen according to the alternative hypothesis (H_1), and the overall P-value is obtained from the chi-square distribution with new degrees of freedom f and a correction factor c to take into consideration the correlation among P-values (Brown, 1975). The conversion of two-sided pvalues in one-sided pvalues is performed according to the sign of the z-test:

$$H_1\text{: with at least one } \beta_j > 0 \implies p_j^{(+)} = \begin{cases} p_j/2 & \text{if } z_j > 0 \\ 1 - p_j/2 & \text{if } z_j < 0 \end{cases} \tag{1.15}$$

$$H_1\text{: with at least one } \beta_j < 0 \implies p_j^{(-)} = \begin{cases} p_j/2 & \text{if } z_j < 0 \\ 1 - p_j/2 & \text{if } z_j > 0 \end{cases} \tag{1.16}$$

If the overall P-value < α (i.e., the significance level), we define node (or edge) perturbation as *activated* when the direction of the alternative hypothesis is positive. Conversely, the status is *inhibited* if the direction is negative.
SEM with exogenous two-group influence can be applied to any graph with the `SEMrun()` function (see help documentation: `?SEMrun`) with the argument `fit=1` or `fit=2` for *common* model or *two-group* model, respectively. Both global fitting statistics, and combined overall P-values are always printed on the console.

1.1.5 Decomposition of effects

In observational studies, as in network biology and medicine, there is the need for assessing causality over paths (i.e., chains of direct effects $X \to \cdots \to Y$) having biological relevance. One important feature of SEM is the decomposition of effects between variables. We may define three types of causal effects: direct effect (DE), indirect effect (IE), and total effect (TE). A DE is the causal effect $X \to Y$ of the j-th variable (X) on the k-th variable (Y) of the model, when all other variables are kept constant (i.e., the effect quantified by path coefficients β_{jk}). Keeping the other variables constant will exclude all causal paths between X and Y, with the exception of the direct connection $X \to Y$ (Pearl, 1998); therefore the DE does not consider mediators effect. In a graph, a *path* between two nodes X and Y can be viewed as a sequence of edges that may have either the same or different direction respect to neighbouring connections. A *directed path* between two nodes is a sequence of edges with the same direction, where node X is an *ancestor* of Y, and Y is a *descendant* of X. The TE includes the contribution of all directed paths connecting X and Y, whereas the IE can be defined as the difference TE $-$ DE.
Let us consider an acyclic mixed graph G (either a DAG or a BAP) and a directed path $\pi \in G$, traveling from node X to node Y, having length (i.e., number of edges) equal to r. Every j-th directed edge in π correspond to a DE quantified by a path coefficient $\beta_{j;j+1}$. The causal effect of X on Y through all the intermediate edges is given by the product of the underlying beta coefficients along a directed path from X to Y. In other words, we may consider π as the path through which information is propagated from the source node X to the target node Y. If there is more than one directed path $\pi_s(s = 1, ..., r(s))$ from X to Y in G, the TE will be the sum of the contribution of each alternative path π through which information propagates from X to Y:

$$\mathrm{TE} = \sum_s \pi_s = \sum_s \prod_{j=0}^{r(s)} \beta_{j;j+1} \tag{1.17}$$

The nodes of an acyclic mixed graph can be ordered topologically, such that we observe a directed edge $j \to k$ only if $j < k$. All possible paths from j to k are given by $[\sum_{r=0}^{\infty} B^r]_{jk}$. Under node topological ordering, the path coefficents matrix B is strictly lower-triangular, it is invertible, and $(I - B)^{-1} = I + B + B^2 + ...$, implying (Drton, Foygel, and Sullivant, 2011):

$$\mathrm{TE}_{jk} = (I - B)^{-1}_{jk} \tag{1.18}$$

$$\mathrm{DE}_{jk} = B_{jk} \tag{1.19}$$

$$\mathrm{IE}_{jk} = (I - B)^{-1}_{jk} - B_{jk} \tag{1.20}$$

Generally, in observational studies and genomics, the interaction between pairs of variables is estimated as the direct effect of the source variable X on the target variable Y, when all other predictors are kept constant. However, this interpretation is incomplete for systems in which mediators effects is not negligible, as in case of perturbation propagation though nodes of a community or a signaling pathway. In these cases, the TE is a more appropriate estimation, considering the simultaneous variation of all mediators. A formal definition of TE, as average causal effect (ACE), is provided by the post-intervention *do*-calculus, defined in Pearl, 2009:

$$\mathrm{ACE} = \mathrm{E}[Y \,|\, \mathrm{do}(X = x + 1)] - \mathrm{E}[Y \,|\, \mathrm{do}(X = x)] \tag{1.21}$$

where $\mathrm{E}[Y \mid \mathrm{do}(X = x)]$ denotes the expected value of Y when X is fixed to a reference value x by external intervention, as in a randomized experiment. In nonlinear models, the ACE will depend on the reference point. However, in a linear Gaussian SEM, x can assume every arbitrary value and the intervention effect (or causal effect) will be a real-valued parameter, given by Pearl, 2009:

$$\mathrm{ACE} = \frac{\partial}{\partial x}\mathrm{E}[Y \mid \mathrm{do}(X = x)] \tag{1.22}$$

In acyclic mixed graphs, this constant parameter is given by the TE computed with the path method as $\mathrm{ACE}_{jk} = (I - B)^{-1}_{jk}$. Alternatively, when the causal model is a DAG, a simple way to compute the ACE is by applying Pearl's backdoor criterion (Pearl, 1998), allowing ACE estimation through regression. The parent set $\mathrm{pa}(X)$ of X blocks all backdoor (i.e., confounding) paths from X to Y, and the ACE is equal to the $\theta_{YX|Z}$ coefficient in a multiple regression of Y on $X + \mathrm{pa}(X)$ (Pearl, 2009). However, adjusting for $\mathrm{pa}(X)$ is typically inefficient with respect to its asymptotic variance, and an optimal adjustament set (O-set) with smallest asymptotic variance is obtained using the parent set of Y, $\mathrm{pa}(Y \mid D^{XY})$, in a suitable latent projection graph D^{XY}, called the forbidden projection (Witte et al., 2020). The ACE is then computed as the $\theta_{YX|Z}$ coefficient in a multiple regression of Y on $X + \mathrm{pa}(Y \mid D^{XY})$.
ACE estimation can be applied to any graph with the `SEMace()` function (see help documentation: `?SEMace`). `SEMpath()` function (see help documentation: `?SEMpath`), search and fit all directed or shortest paths between two source-sink nodes of a graph. In addition, `pathFinder()` function (see help documentation: `?pathFinder`) uses `SEMace` to find significant causal effects between source-sink pairs and `SEMpath` to fit them and test their edge perturbation.

1.1.6 Graph-weighting methods

The input interactome, $G(V,E)$ can be converted into a weighted "perturbed" network, $G(V_W,E_W)$ endowed with nodes and edges weight reflecting their perturbation status. Genes (nodes) can be weighted by bivariate P-values testing the group (1= experimental, 0=control) effect on each gene with t-test o similar tests for two-group differences (via **limma**, **SAM**, etc. in R/Bioconductor packages), or a binary "seed" attribute (1=seed, 0=non-seed) can be associated on each node, if P-value < alpha. As per node weights, gene-gene interactions (edges) can be weighted based on the group difference of pairwise correlation measures.
In **SEMgraph** with the `weightGraph()` function, four trivariate procedures can be performed based on: (i) SEM (regression) model, (ii) Covariance model, (iii) Confirmatory Factor Analysis (CFA) model, and (iv) Fisher's transformation z-to-r correlation method, as follow.

SEM (regression) model. The SEM (regression) model implies testing the group effects on the source-sink link. A common group effect model of C = {0: control; 1: case} is fitted (see Figure A.1) on the source node k, and the sink j:

$$Y_k = \beta_{k0} + \beta_{\mathrm{k}C}C + U_k$$

$$\beta_{j0} + \beta_{jC}C + \beta_{jk}Y_k + \beta_{jkC}Y_jC + U_j$$

or splitting the groups:

$$Y_{k|C=0} = \beta_{k0} + U_j$$

$$Y_{k|C=1} = \beta_{k0} + \beta_{kC}C + U_k$$

$$Y_{j|C=0} = \beta_{j0} + \beta_{jk}Y_k + U_j$$

$$Y_{j|C=1} = (\beta_{j0} + \beta_{jC}) + (\beta_{jk} + \beta_{jkC})Y_k + U_j$$

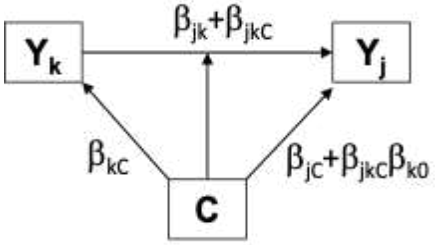

FIGURE 1.1: **SEM model.**

the coefficient, β_{kC} represents the intercept (mean) difference between groups in C for source k, while the coefficients, β_{jC} and β_{jkC} represent intercept and slope differences between group $C = 1$, equal to $(\beta_{j0} + \beta_{jC})$ and $(\beta_{jk} + \beta_{jkC})$, and group $C = 0$, equal to β_{j0} and β_{jk}, respectively.

Through causal inference framework, we define two group effect indices (a_1 and a_2), measuring node perturbance (or disease relevance), i.e., the causal total effect (TE) of C on sink j:

$$a_{1j} = (\beta_{jC} + \beta_{jkC} \cdot \beta_{k0}) + \beta_{kC} \cdot (\beta_{jk} + \beta_{jkC})$$

the causal TE, equal to direct effect (DE), of group, C on source k:

$$a_{2k} = \beta_{kC}$$

and a weighted sum defines the new parameter, w combining the TEs of group, C on source and sink nodes:

$$w_{jk} = \text{abs}(a_{ij})/d_j + \text{abs}(a_{2k})/d_k$$

where d_k and d_j are the outgoing degrees (i.e., the number of all direct outgoing connections in the input graph, for each node) of the k-th sources and j-th sinks, respectively. The P-values are computed through the z-test = w/SE(w) of the combined TE of the group effect on the source node j and the sink node k. Of note, the SE(w) is obtained by lavaan syntax specifying w with the ":=" operator.

Covariance model. The covariance model implies testing the group effects at the same time on the source j, the sink k and their interaction (j, k). Two-group covariance model with an intercept parameter is fitted:

$$Y_j^{(0)} = \alpha_j + U_j^{(0)};\ \ Y_k^{(0)} = \alpha_k + U_k^{(0)};\ \ \text{cov}\left(Y_j^{(0)}; Y_k^{(0)}\right) = \phi_{jk}$$
$$Y_j^{(1)} = \alpha_j + U_j^{(1)};\ \ Y_k^{(1)} = \alpha_k + U_k^{(1)};\ \ \text{cov}\left(Y_j^{(1)}; Y_k^{(1)}\right) = \psi_{jk}$$

A weighted sum defines the new parameter, w combining the group effect on source node (mean difference, $\beta_k - \alpha_k$), sink node (mean difference, $\beta_j - \alpha_j$), and source-sink link (correlation difference, $\phi_{jk} - \psi_{jk}$):

$$w_{jk} = \text{abs}\left(\beta_j - \alpha_j\right)/d_j + \text{abs}\left(\beta_k - \alpha_k\right)/d_j + \text{abs}\left(\phi_{jk} - \psi_{jk}\right)$$

where d_k and d_j are the degree (number of all direct connections in the input interactome, G) of the source k and sink j nodes. P-values are yielded by the t-test=w/SE(w) on the combined difference of the group over the source node j, the sink k, and their connection $j \rightarrow k$. Of note, the SE(w) is obtained by lavaan syntax specifying w with the ":=" operator.

Confirmatory Factor Analysis (CFA) model. The CFA model assumes that the pairwise connected genes j and k are related with a latent variable (LV or factor, F) of unknown common(s) cause(s), and this LV is associated to the disease class (C=0,1), see Figure A.2:

$$Y_k = \lambda_k F + U_k; \quad Y_j = \lambda_j F + U_j$$

$$F = \beta C + D \quad \& \quad \text{var}(D) = \phi$$

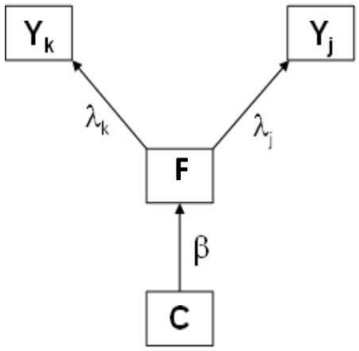

FIGURE 1.2: **CFA model.**

For model identification the "common" gene-factor-coefficients, λ's and the variances of U terms (measurement error variables), are fixed values equal to:

$$\lambda = \sqrt{\text{cor}\left(Y_j^{(0)}; (Y_k^{(0)}\right)} \quad \& \quad \text{var}\left(U_k\right) = \text{var}\left(U_j\right) = 1 - \lambda^2$$

For square root computing, if $cor(Y_k; Y_j) < 0$ a negative scaling of Y_j (or Y_k) in $-Y_j$ (or $-Y_k$) is performed. The fitted model with one factor (F), two indicators (Y_j and Y_k) and an exogenous variable (C) has two free parameters, the error variance of the latent variable, ψ, and the regression coefficient, β. Therefore, the CFA model has df=((3x2)/2)–2=1, and can be fitted with **lavaan**. The P-values by the z-test=$\beta/SE(\beta)$ of factor mean differences quantify the group perturbance on the unknown common(s) cause(s) of the covariance of $(Y_k; Y_j)$.

Fisher's transformation z-to-r correlation method. The correlation method applies the Fisher's r-to z transformation for testing the pairwise difference between the correlation coefficients of linked genes in the input (directed or undirected) interactome, G. The transformation is done in each group, C= (1,0):

$$r_{jk}^{(0)} = \text{cor}\left(Y_j^{(0)}; (Y_k^{(0)}\right) \quad \& \quad r_{jk}^{(1)} = \text{cor}\left(Y_j^{(1)}; (Y_k^{(1)}\right)$$

$$z_{jk}^{(0)} = \log\left(\frac{1+r_{jk}^{(0)}}{1-r_{jk}^{(0)}}\right) \quad \& \quad z_{jk}^{(1)} = \log\left(\frac{1+r_{jk}^{(1)}}{1-r_{jk}^{(1)}}\right)$$

The group differences are computed by the classical test, t proposed by Fisher, 1915:

$$t_{jk} = \frac{z_{jk}^{(1)} - z_{jk}^{(0)}}{\sqrt{\frac{1}{n_1-3} + \frac{1}{n_0-3}}}$$

The P-values of t's from $N(0,1)$ distribution measure the group perturbance on the gene-gene correlations.

After P-value computing, the edge weights are defined as inverse of negative logarithm of the P-value, i.e., w= 1/[-log_{10}(P-value)]. In this way, edges with lower P-values (0 to 1) have lower weights (w >0) on a positive continuous range. Intuitively, this weight can be assumed as the perturbance acting on the relationship between two connected genes in the interactome, due to the genotype difference between groups. The lower the P-value (or the w-value), the higher the perturbance. Using the definition of perturbance, we refer to a set of adjacent perturbed edges as a *perturbation route*, originating from a source, passing through a number of connectors, and terminating in a sink node.

1.1.7 Network clustering and scoring

SEMgraph offers the possibility to define topological communities of an input graph, generating scores for each statistical unit (i.e., subject) by using data from nodes belonging to communities. Clusters can be defined using the algorithms implemented in the R package **igraph** (Csardi and Nepusz, 2006) and then they can be fitted as independent models. Among the available clustering methods, we suggest either the walktrap community detection algorithm (WTC), based on random walks and develobed by Pons and Latapy, 2005, or the edge betweenness clustering (EBC), developed by Newman and Girvan, 2004. The former tends to generate as many clusters as needed to cover the whole input network. The latter generally produces one large subnetwork and other much smaller communities or singletons. In case of trees, our implementation of the tree agglomerative hierarchical clustering (TAHC), proposed by Yu et al., 2015, is the suggested solution. Our aim here is to provide a tool yielding different (orthogonal) local models when dealing with large networks ($|V| > 100$). Beside network size, we generally recommend clustering when there are evidences of possible functional modules (i.e., subnetworks whose members are involved in a specific process).

For sample scoring in the SEM framework, we consider the general SEM with esplicitly latent variables (i.e., latent factor) defined as (Bollen, 1989, p. 395):

$$Y = \Lambda Y + E \quad \text{and} \quad F = B F + U \tag{1.23}$$

with the general model-implied covariance matrix:

$$\Sigma(\theta) = \Lambda(I-B)^{-1}\Psi(I-B)^{-T}\Lambda^T + D \tag{1.24}$$

where F is the vector of $q < p$ latent factors, E the vector error terms, Λ is the matrix ($p \times q$) of factor loadings, and D is the diagonal covariance matrix of error

terms; while, B and Ψ of dimension $(q \times q)$ are the usual path coefficient matrix and covariance matrix of the latent factors and unobserved variables of a SEM.
Setting $\Lambda = I$ and $D = 0$ the general SEM is reduced to usual SEM on the observed variables Y; while, setting $B = 0$ and $\Psi = 0$, the general SEM is the special case of the Confirmatory Factor Model (CFA)(Bollen, 1989, p. 226).
In system biology was proposed eigengene networks to describe the relationship between co-expression gene modules (clusters), where the *eigengene* is the first principal component of a given module (Langfelder and Horvath, 2007). It can be considered a rappresentative summary of the gene expression profile of a module, since in pratice, explains typically more than 50 percent of the variance of the module expressions. If we set $q = 1$ in the CFA model, sample scoring can be generated by three different *hidden* models: the latent variable model (1LV), the composite variable model (1CV), and the unobserved variable model (1UV).
The 1LV model consists in a confirmatory factor analysis (CFA) with one factor and specific error variances (Bai and Li, 2012):

$$Y_j = \lambda_j F + E_j \quad \text{with} \quad \text{var}(F) = 1 \ \text{ and } \ \text{var}(E_j) = \psi_j \tag{1.25}$$

The 1CV model consists in a CFA with one factor and equal (common) error variances, equivalent to a principal component analysis (PCA) (Bai and Li, 2012):

$$Y_j = \lambda_j C + E_j \quad \text{with} \quad \text{var}(C) = 1 \ \text{ and } \ \text{var}(E_j) = \psi \tag{1.26}$$

The 1UV model corresponds to a fixed factor analysis (FFA) model with one factor projected on the exogenous observed X set, with zero residual variance, and equal (common) error variances fixed to 1. This is equivalent to a reduced-rank regression analysis (RRA) (Davies and Tso, 1982):

$$Y_j = \lambda_j U + E_j \quad \text{with} \quad \text{var}(U) = 1 \ \text{ and } \ \text{var}(E_j) = 1 \tag{1.27}$$

$$U = \sum \gamma_k X_k \tag{1.28}$$

In every hidden model, Y_j are the random endogenous observed variables of each module, and E_j the residual errors, with $j = (1, \ldots, q)$. In the UV model, X_k represent the exogenous observed variables, with $k = (1, \ldots, r)$. Variables F, C, and U correspond to the scores assigned to each subject for each cluster, representing the latent factor, the principal component, and the unmeasured variable of the hidden model, respectively. In the UV model, the factor scores U of the endogenous variables (i.e. sink and connectors nodes of each module) are found in the space spanned by the exogenous variables X (i.e., they are projected on source nodes of each module). Factor scores U are also called *unmeasured variables*, rather than latent variables or factors, because they can be expressed as a function of the exogenous observed X. Although the underlying variables are not actually measured, the scores U are measurable (Bentler and Weeks, 1980).
To note, only 1LV, 1CV, or 1UV modules for which cluster scores represent 50 percent or more of the total variance are considered as raspresentative of the gene profile.
Cluster algorithms and cluster scoring can be applied with the `clustertGraph()` function (see help documentation: `?clusterGraph`), and the `clustertScore()` function to any graph (see help documentation: `?scoreCluster`), respectively. Scores are generated using the `factor.analysis()` function of the R package **cate** (Wang and Zhao, 2019), an efficient package for high-dimensional factor analysis models

1.2 Knowledge-based model improvement in Amyotrophic Lateral Sclerosis (ALS)

The R package **SEMgraph** offers the possibility to import, build, and fitting causal models directly leveraging on knowledge (i.e., the input graph), quantitative data, and a possible exogenous perturbation source (e.g., a phenotypical trait or a disease), see for computational details Section A.0.1 of the Supplementary Material.
In this section, we will use example data in the package to build a SEM from an available graph. Our example come from an ALS RNA-seq expression data (Cooper-Knock et al., 2015) stored in the `alsData`, a list of three objects: `alsData$graph` a subgraph with 32 nodes and 47 edges of the "Amyotrophic lateral sclerosis" pathway from KEGG database; `alsData$exprs`, a matrix of 160 subjects $\times$ 303 genes (with 139 ALS cases and 21 healthy controls), and `alsData$group`, a binary group vector of 139 ALS subjects (1) and 21 healthy controls (0).

1.2.1 SEM fitting functions

The three objects of `alsData` are the basic **SEMgraph** arguments: `graph`, `data`, and `group`. Regarding quantitative data, we always suggest to apply some kind of correction method to relax the normality assumption required by SEM. While *log*2 or *ln* transform are frequently used for count data (e.g., sequencing), we generally suggest the *nonparanormal* transform implemented in the `huge.npn()` function of the R package **huge** (Zhao et al., 2012).

```
R> library(SEMgraph)

R> # ALS example
R> summary(alsData$graph) # ALS input graph
R> dim(alsData$exprs) # ALS RNA-seq expression data
R> table(alsData$group) # {case = 1, control = 0} vector

R> # Nonparanormal transform
R> library(huge)
R> data.npn <- huge.npn(alsData$exprs)
```

In **SEMgraph**, the basic function for model assessment is `SEMrun()`:

```
R> sem0 <- SEMrun(graph = alsData$graph, data = data.npn)

@NLMINB solver ended normally after 8 iterations
deviance/df: 10.92479  srmr: 0.2858233
```

This function maps data onto the input graph (removing possible identifiers inconsistencies), converts the input graph into a SEM, and fits the model using **lavaan** syntax. For high-dimensional data, the shrinkage covariance proposed by Schäfer and Strimmer, 2005 is applied to estimate the sample covariance S, as implemented in the `cor.shrink()` function of the **corpcor** R package (Schäfer et al., 2017). Model fitting results and the output graph are saved inside the `sem` object. If the `group` argument is omitted, `SEMrun()` will only generate estimates for direct effects, as specified by the input `graph`. Object `sem0$fit` is a fitted model of class `lavaan`, from which we can simply extract direct effect estimations with `summary` or `parameterEstimates`, as follows:

```
R> est0 <- parameterEstimates(sem0$fit)
R> head(est0)
```

```
@     lhs op   rhs    est    se      z pvalue ci.lower ci.upper
1 z10452  ~ z6647  0.037 0.079  0.466  0.641   -0.118    0.192
2  z1432  ~ z5606  0.397 0.069  5.741  0.000    0.261    0.532
3  z1432  ~ z5608  0.578 0.069  8.361  0.000    0.442    0.713
4  z1616  ~ z7132  0.245 0.110  2.236  0.025    0.030    0.461
5  z1616  ~ z7133 -0.036 0.110 -0.324  0.746   -0.251    0.180
6  z4217  ~ z1616 -0.074 0.079 -0.943  0.346   -0.229    0.080
```

For gene networks, we always recommend using Entrez gene IDs, to avoid possible special characters or naming ambiguities. If the argument `group` is given, group influence is modeled as an exogenous variable acting on every node, perturbing their activity.

```
R> sem1 <- SEMrun(alsData$graph, data.npn, alsData$group)

@NLMINB solver ended normally after 23 iterations
deviance/df: 11.02558  srmr: 0.2747457
Brown's combined P-value of node activation: 7.104523e-08
Brown's combined P-value of node inhibition: 0.01068782
```

Also in this case, direct node-node effects, as well as group effects on nodes, can be inspected using `parameterEstimates()`:

```
R> est1 <- parameterEstimates(sem1$fit)
R> head(est1)

@     lhs op   rhs    est    se      z pvalue ci.lower ci.upper
1 z10452  ~ group -0.150 0.078 -1.913  0.056   -0.303    0.004
2  z1432  ~ group -0.042 0.073 -0.578  0.563   -0.186    0.101
3  z1616  ~ group  0.025 0.079  0.315  0.753   -0.131    0.181
4   z317  ~ group  0.218 0.077  2.832  0.005    0.067    0.370
5  z4217  ~ group  0.176 0.078  2.273  0.023    0.024    0.328
6  z4741  ~ group  0.343 0.076  4.530  0.000    0.195    0.490
```

Significant perturbed nodes can be viewed calling `gplot()` on the output graph, as shown below. The resulting plot is shown in Figure 1.3.

```
R> # Convert Entrez identifiers to gene symbols
R> library(org.Hs.eg.db)
R> V(sem1$graph)$label <- mapIds(org.Hs.eg.db, V(sem1$graph)$name,
+                                column = 'SYMBOL',
+                                keytype = 'ENTREZID')
R> # Graph plot
R> gplot(sem1$graph)
```

High dimensionality can be troublesome not only due to a reduced sample size. Network size (i.e., the number of its nodes, $|V|$) may dramatically increase the computational demand, mainly during model parameters estimation. For large graphs ($|V| > 100$), standard error (SE) computation will be disabled and parameter estimates will be computed through residual iterative conditional fitting (RICF) and permutation based P-values by randomization of group labels. The RICF mode is either automatically enabled when $|V| > 100$ (this limit can be changed using the `limit` argument in `SEMrun()`, to enforce standard SE estimation) or manually called using the `algo = "ricf"` argument:

```
R> ricf1 <- SEMrun(alsData$graph, data.npn, alsData$group,
                   algo = "ricf")
```

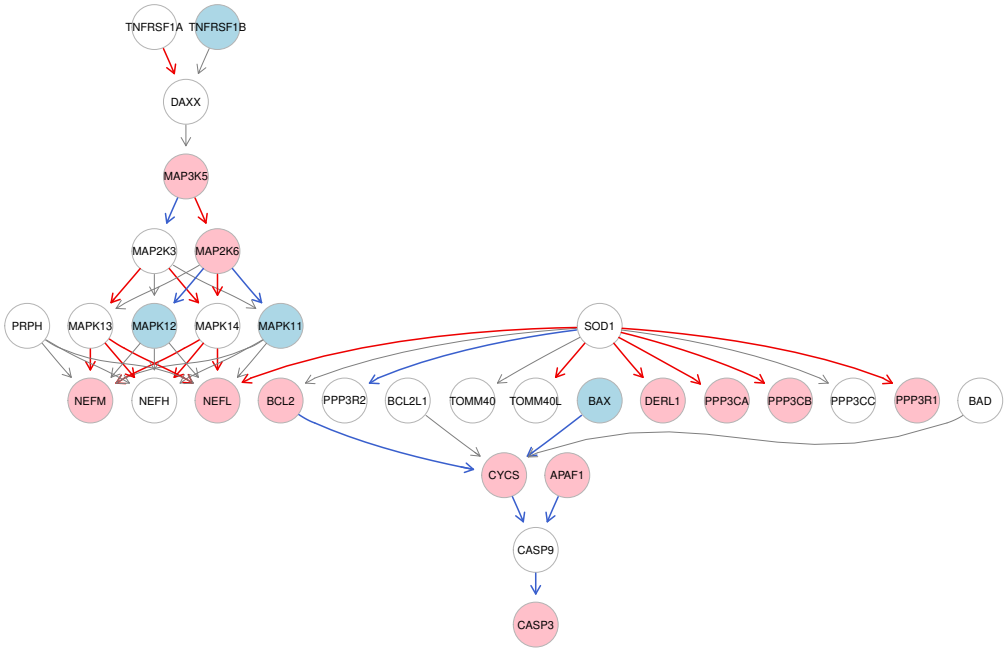

FIGURE 1.3: Estimated group effects on nodes and direct effects. The graph shows differentially regulated nodes (DRNs) as ALS-activated (pink-shaded) or ALS-inhibited (blue-shaded) variables. White nodes do not show significant variation in ALS, respect to healthy controls. "Common" to both groups significant direct effects are shown in either red (activated) or blue (inhibited), while gray direct effects are not significant.

```
RICF solver ended normally after 2 iterations
deviance/df: 11.02558  srmr: 0.2747457
Brown's combined P-value of node activation: 7.074471e-08
Brown's combined P-value of node inhibition: 0.009522031
```

As for the basic (i.e., lavaan-based) algorithm, the command `gplot(ricf1$graph)` can be used with the `gplot()` function to plot node perturbation. Both lavaan-based and RICF-based fitting show two important results. Firstly, the randomization approach leads to a perturbation estimation that is not significantly different from the asymptotic one (model fitting and overall perturbation is left unaltered by both RICF and the randomization procedure). Secondly, both functions detect significant network perturbation (mainly activation), but no acceptable fitting (see Section 1.2.3 for model refinement).
In addition, to node perturbation, **SEMgraph** enables edge perturbation estimation via the two-groups SEM implemented in `SEMrun`, setting the `fit` argument to 2 groups (see Section 1.1.4 for details):

```
R> sem2 <- SEMrun(alsData$graph, data.npn, alsData$group, fit = 2)

Estimating optimal shrinkage intensity lambda (correlation matrix):
0.4313
NLMINB solver ended normally after 30 iterations
deviance/df: 5.295486  srmr: 0.2785664
Brown's combined P-value of edge activation: 9.085395e-06
Brown's combined P-value of edge inhibition: 0.2748447
```

In accordance with node perturbation, we observe a predominant global edge activation. As for node-level testing, edge perturbation can be plotted through the

command `gplot(sem2$graph)`. The list of DRNs and DREs can be extracted from the objects `sem1$gest` and `sem2$dest`, respectively:

```
R> DRN <- sem1$gest[sem1$gest$pvalue < 0.05,]
R> nrow(DRN)
@[1] 16

> head(DRN)
      lhs op   rhs   est    se     z pvalue ci.lower ci.upper
4     317  ~ group 0.218 0.077 2.832  0.005    0.067    0.370
5    4217  ~ group 0.176 0.078 2.273  0.023    0.024    0.328
6    4741  ~ group 0.343 0.076 4.530  0.000    0.194    0.491
8    4747  ~ group 0.223 0.062 3.611  0.000    0.102    0.344
9   54205  ~ group 0.188 0.067 2.789  0.005    0.056    0.319
10   5530  ~ group 0.160 0.072 2.224  0.026    0.019    0.301

R> DRE <- sem2$dest[sem2$dest$pvalue < 0.05,]
R> nrow(DRE)
[1] 3

> head(DRE)
     lhs op   rhs d_est  d_se   d_z pvalue d_lower d_upper
28  5532  ~  6647 0.449 0.227 1.983  0.047   0.005   0.893
30  5534  ~  6647 0.584 0.229 2.547  0.011   0.135   1.034
34  5603  ~  5606 0.496 0.239 2.073  0.038   0.027   0.965
```

The current model yields 16 DRNs and 3 DREs. With increasing $|V|$, also the edge perturbation estimation could be computationally intensive. For large graphs (by default, $|V| > 100$), edge perturbation is estimated using a constrained gaussian graphical model (GGM), as implemented in the **GGMncv** package (Williams, 2020), or manually called using `algo = "cggm"` argument:

```
R> cggm2 <- SEMrun(alsData$graph, data.npn, alsData$group, fit = 2,
+                  algo = "cggm")

@GGM (constrained) solver ended normally after 0 iterations
deviance/df: 5.32122  srmr: 0.2860305
Brown's combined P-value of edge activation: 0.0004052721
Brown's combined P-value of edge inhibition: 0.0152556
```

Also in this case, with large graph ($|V| > 100$) the canonical (i.e., lavaan-based) perturbation estimation can be enforced by increasing the `limit` argument.

1.2.2 Total effect estimation

As anticipated in Section 1.1.5, total effect (TE) estimation could be a key tool to search for perturbed routes conveying information inside a complex network. Biological signaling pathways provide a paradigmatic example of this propagation inside the cell regulatory network. A ligand interacts with a cell surface receptor (*source*), starting the information flow that is propagated and modulated by second messengers, enzymes and chaperones (*connectors*) through the cytoplasm to the cell nucelous, where specific factors (*sinks*) are either activated or inhibited, regulating transcription, replication, cell development, and fate. This directional information flow can be computationally represented by a DAG, where the TE can be evaluated with a single comprehensive estimation as an average causal effect (ACE). Function `SEMace()` converts the input graph into a DAG and computes ACEs between every possible source-sink node pair, using the optimal adjustement set (O-set) procedure described in Section 1.1.5:

```
R> ace <- SEMace(graph = alsData$graph, data = data.npn,
+                            type = "optimal", effect = "source2sink",
+                            method = "BH", alpha = 0.05)
R> ace <- ace[order(abs(ace$z), decreasing = TRUE),]
R> nrow(ace)
[1] 10

R> head(ace)
    sink op source    est    se      z pvalue ci.lower ci.upper
4   4747 <-   6647  0.514 0.063  8.113      0    0.390    0.639
14   836 <-    317  0.472 0.061  7.737      0    0.352    0.592
5  79139 <-   6647  0.522 0.068  7.723      0    0.390    0.655
7   5532 <-   6647  0.521 0.068  7.700      0    0.389    0.654
10  5535 <-   6647 -0.462 0.070 -6.565      0   -0.600   -0.324
3    836 <-   6647  0.430 0.067  6.433      0    0.299    0.561
```

In this example, there are 10 significant ACEs, ordered by decreasing z scores. Function `SEMpath()` allow us to evaluate any of them as an independent model. The following code shows fitting and node perturbation estimation for the sixth directed path in the example above, connecting SOD1 (Entrez ID: 6647) and CASP3 (Entrez ID: 836):

```
R> source <- as.character(ace$source[6])
R> sink <- as.character(ace$sink[6])
R> path <- SEMpath(alsData$graph, data.npn, alsData$group,
+                  from = source, to = sink,
+                  path = "directed",
+                  verbose = TRUE)

@Path: 6647 -> 836 size- 5 4 --

NLMINB solver ended normally after 6 iterations
deviance/df: 24.52598  srmr: 0.2067487
Brown's combined P-value of node activation: 3.724367e-06
Brown's combined P-value of node inhibition: 0.9286749 @
```

Argument `path = "directed"` considers every directed path connecting the source-sink pair. This argument can be also set to `"shortest"`, to consider shortest paths only. Argument `verbose = TRUE` shows the position of the selected path within the input network. Function `pathFinder()` can be used to extract all the directed paths whose source-sink pairs share a significant ACE and evaluate each of them as an independent SEM:

```
R> paths <- pathFinder(alsData$graph, data.npn,
+                       group = alsData$group, ace = ace)
R> print(paths$dfp)
```

Argument `ace` allows the user to specify an existing data.frame of ACEs, while `group` can be skipped if one is just interested in path fitting (i.e., no node perturbation test is performed).

1.2.3 Model estimation strategies

In biological systems, curated networks rarely provide a complete explanation of data variability, often leading to a poor SEM fitting. This is exactly what happened when we fitted RNA-seq ALS data onto the ALS pathway provided by KEGG. In this case, the known ALS model is able to detect significantly perturbed nodes and edges, but a significant proportion of data variability is still unexplained, as shown

by the global fitting statistics (deviance/df and SRMR).
One of goal of **SEMgraph** is to learn the causal structure from data, applying the best tradeoff between model fitting and perturbation, improving the initial model by leveraging on both knowledge-based and data-driven procedures. The other goal, is to provide a set of causal interence tools also for users with minimal statistical expertise. To this end, we propose four preset strategies, combining `SEMdag()`, `SEMbap()`, and `resizeGraph()` functions (see help documentation: `?SEMdag()`, `?SEMbap()` and `?resizeGraph()` and Chapter 4-5 for statistical details). All strategies estimate a DAG according to the following steps:

0. **Input**: an initial suitable DAG = $G^{(0)}$ and an initial raw (or pre-processed) data $Y = Z^{(0)}$;

1. Given ($G^{(0)}$ and $Z^{(0)}$) update $Z^{(1)} = Z^{(0)}\Psi^{-\frac{1}{2}}$ by fitting the constrained matrix Ψ^{-1} after d-separation testing of $\text{cor}(Z_j; Z_k \mid \text{pa}(j) \cup \text{pa}(k)) = 0$ at a given *alpha* significance level, using the `SEMbap(`$G^{(0)}, Z^{(0)}$`)` function;

2. Given ($G^{(0)}$ and $Z^{(1)}$), update $G^{(1)}$ estimating the DAG via topological order of $G^{(0)}$ and edges penalty weighted LASSO screening at a given *beta* threshold, using the `SEMdag(`$G^{(0)}, Z^{(1)}$`)` function;

3. (optional) Remove edges/add nodes to DAG $G^{(1)}$ at a given geodesic distance d in the reference interactome G using the `resizeGraph(`$G^{(1)}, G$`)` function;

4. **Output**: Goodness-of-fit statistics (C-test, SRMR) of the DAG $G^{(1)}$ or the resized graph $G^{(2)}$ of step 3.

This procedure is implemented in the `modelSearch()` function (see help documentation: `?modelSearch`). DAG estimation can be controlled through the argument `alpha` (i.e., the significance level for the FDR correction), where 0 corresponds to no data de-correlation, and `beta` (i.e., the LASSO coefficient threshold), where 0 maintains all the edges of the input graph, and `d` the geodesic distance for the connection check in the reference network. We suggest to start with `beta = 0.1` to have a good balance between model adjustment and graph sparsity. Then `beta` could be gradually decreased (0.1 to 0) to obtain more complex models. Similarly, argument `alpha` can be decreased up to `5e-09`. A lower `alpha` level includes less hidden covariances, thus considering less sources of confounding, resulting in a lower data de-correlation. We also suggest to start with `search = "basic"` procedure, without DAG resize (`genet = NULL` and geodesic distance `d =0`).
Considering the ALS example, the model search of a DAG using the `search = "basic"` procedure has the following code:

```
R> # Model search
R> model <- modelSearch(alsData$graph,  data.npn, gnet = NULL,
+                       d = 0, search = "basic", beta = 0.1,
+                       alpha = 0.05, verbose = FALSE)

@Step1: BAP deconfounding...
Step2: DAG estimation...
Step3: DAG resize (remove edges/add nodes)...

None DAG resize for basic search !

Done.
```

```
d-separation test (basis set) of 267 edges...
    C_test  df   pvalue
1 543.7724 534 0.375395

RICF solver ended normally after 2 iterations
deviance/df: 1.755445  srmr: 0.0839307 @
```

The resulting graph is shown in Figure 1.4A. We may then evaluate model perturbation using the `SEMrun()` function, as shown in Figure 1.4B. In addition, with `SEMace()` and `SEMpath()` we can evaluate ACE, path perturbation, and fitting of specific directed paths between a souce-sink pair. As an example, Figure 1.4C shows in yellow all directed paths between genes SOD1 (Entrez ID: 6647) and NEFM (Entrez ID: 4741).

```
R> pert <- SEMrun(model$graph, model$data, alsData$group)
R> ace <- SEMace(model$graph, model$data, alsData$group,
+                method = "BH")
R> path <- SEMpath(model$graph, model$data, alsData$group,
+                  from = "6647", to = "4741", path = "directed")
```

All the steps done by `modelSearch()` are shown to standard output, and the resulting graphs are visualized in Figure 1.4 A-B-C. Following the example above, the extracted DAG model has a good fitting (deviance/df < 2, srmr near 0.08, and C-test with P-value > 0.05). The output `model` object contains model fitting as a **lavaan** object (`model$fit`), the output graph coloured according to node and edge relevance during the estimation steps (`model$graph`), and the adjusted dataset (`model$data`). With `search = "basic"`, we enabled a *data-driven* model search strategy, where model structure is based only on data and no validation against a reference network is done (i.e., the optional `resizeGraph()` function does not run). In the example above, we set `beta` to `0.1` to reduce graph density. As a result, input edges could be removed and new ones could be added, partially reshaping model architecture. The aim is to generate an improved model, achieving a good overall fitting (for DAGs, the main fitting index is the Shipley's global test P-value > 0.05), showing the best possible balance among model complexity, fitting, and perturbation.
In this example, the output model shows how the SOD1 gene deregulation is causally connected to the deregulated gene NEFM, implied in the maintainance of a physiological neuronal caliber. This indirect connection (the yellow path in Figure 1.4C), absent in the input model (Figure 1.3), is now possible thanks to the new connections BCL2-DAXX, CYCS-MAPK13, and TOMML40-MAP2K6 (red links in Figure 1.4A), showing a tight association between apoptosis and neuronal caliber regulation, both dysregulated in neurodegenerative disorders.
Conversely, we could take advantage of known interactions, importing them in our model to resize it (i.e., `gnet = kegg` and `d > 0` in `resizeGraph()`). We define them as *knowledge-based* strategies. The *outer* (`search = "outer"`) strategy relies on an external reference network , to assess the presence of possible hidden mediators (`d > 1`), including them in the output model. If one is not interested in adding new mediators from the reference, but still wants to evaluate the presence of internal hidden indirect (i.e., mediated) paths, the `search` argument can be set to `"inner"`: the reference network is still used, but only to validate the new direct and indirect paths added to the model. Both *inner* and *outer* search strategies rely on the initial estimation of a DAG, working as a causal model backbone. Finally, we can use a *direct* strategy (`search = "direct"`), where the input graph structure is improved only through direct (i.e., adjacent) link search, followed by interaction validation and import from the reference network, with no mediators (i.e., `d = 1`).

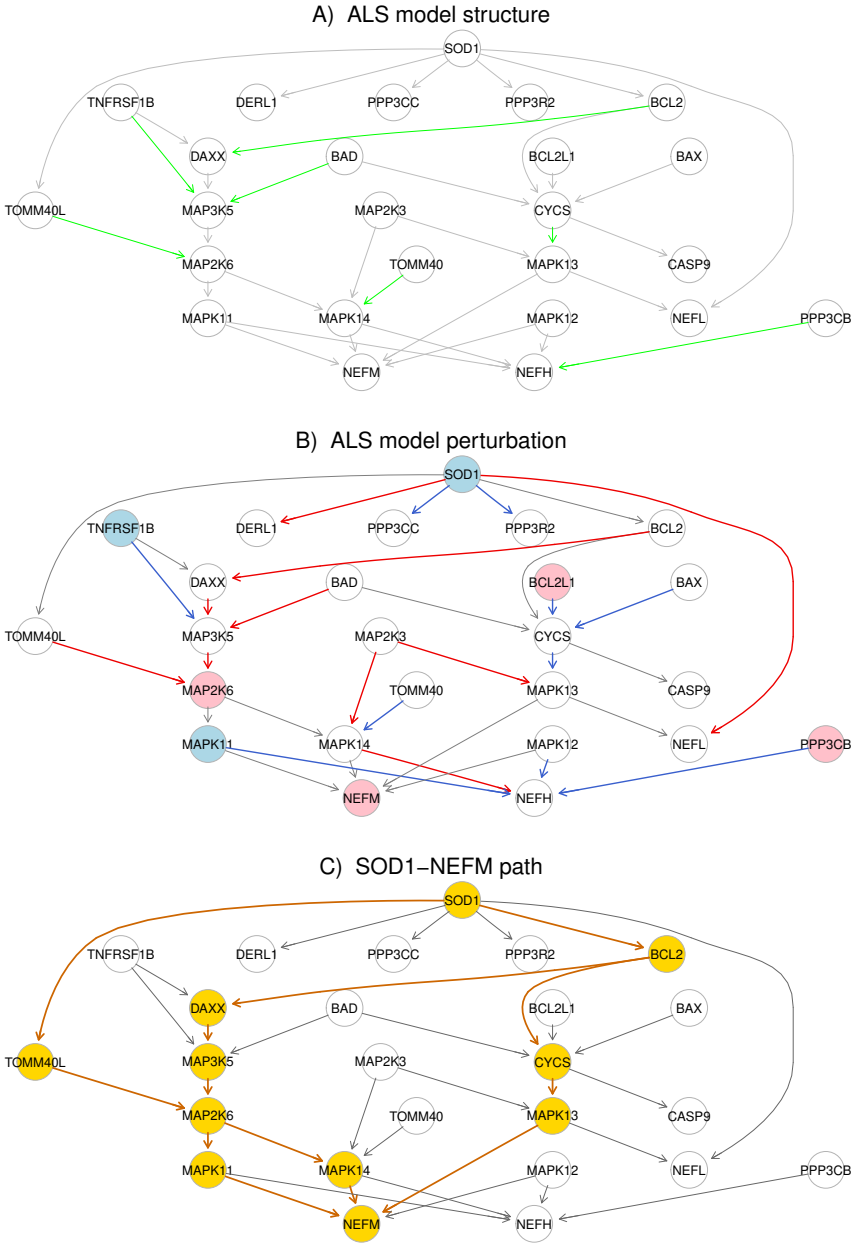

FIGURE 1.4: ALS improved model. **Panel A** shows the output model structure, as generated by `modelSearch()`. Added edges are highlighted in green, while gray edges are maintained from the input ALS graph. **Panel B** shows node-level perturbation, estimated by `SEMrun()`: pink nodes are activared, while lightblue nodes are inhibited. Edges are coloured according to their significance: significant direct effects (P-value < 0.05) are either red (estimate > 0) or blue (estimate < 0), while non-significant ones are gray-shaded. **Panel C** highlights in yellow all directed paths between genes SOD1 and NEFH, showing how `SEMpath()` may help us to clarify and evaluate causal effects between perturbed source-target pairs, within an entangled cluster.

1.2.4 Communities and factor scores

SEMgraph generates communities using `igraph()`, and cluster scores via the function `factor.analysis()` of the R package **cate** (Wang and Zhao, 2019), an efficient package for high-dimensional factor analysis models. Only modules for which cluster scores represent 50% or more of the total variance are considered. Topological clustering sintax of the `model$graph`, extracted with the `modelSearch` function, is the following:

```
R> LV <- clusterScore(model$graph, model$data, alsData$group,
+                     HM = "LV", type = "ebc", size = 5)

@modularity = 0.4816345

Community sizes
 1  3  2
 6  9 10

Model converged: TRUE
SRMR: 2.618078e-09
```

Arguments `type` and `size` set the clustering algorithm and the minimum group of nodes to generate a cluster (groups smaller than `size` are considered as singletons). The suggested `type` is the edge betweeness (`"ebc"`) community detection algorithm (see help documentation: `?clusterScore` for the other cluster algorithms) resulting in the largest number of nodes included in clusters, with a minimum cluster size of 5. Argument `HM` determines the type of *hidden model* used to generate cluster scores: latent variable model (`HM = "LV"`), composite variable model (`HM = "CV"`), and unobserved variable model (`HM = "UV"`) (see Section 1.1.7 for details).
Here the clusters have size 6, 9, 10. The global effect of the `group` on every cluster scores, defined by LV model, can be viewed using `parameterEstimates()`:

```
head(parameterEstimates(LV$fit))

@  lhs op   rhs     est    se       z pvalue ci.lower ci.upper
1 LV1  ~ group -0.471 0.238 -1.976  0.048   -0.938   -0.004
2 LV2  ~ group  0.042 0.249  0.167  0.867   -0.447    0.531
3 LV3  ~ group  0.744 0.287  2.588  0.010    0.180    1.307
4 LV1 ~~   LV1  1.037 0.116  8.944  0.000    0.809    1.264
5 LV2 ~~   LV2  1.135 0.127  8.944  0.000    0.886    1.384
6 LV3 ~~   LV3  1.507 0.168  8.944  0.000    1.176    1.837
```

Every cluster is represented by a LV and each estimate measures the global effect of the group over it. Together with the fitted hidden model `LV$fit`, `clusterScore()` returns the `data.frame` containing cluster scores (`LV$dataHM`) and a vector indicating the cluster membership for every node (`LV$membership`). Topological cluster networks (without subject scoring) can be produced independently from `clusterScore()`, using the `clusterGraph()` utility:

```
R> C <- clusterGraph(model$graph, type = "ebc", size = 5,
+                    verbose = FALSE)
```

The `clusterGraph()` arguments are equivalent to those used in `clusterScore()`. In addition, function `cplot()` generates and plots separate graphs for each cluster:

```
R> cg <- cplot(graph = model$graph, membership = LV$membership,
+              verbose = TRUE)
```

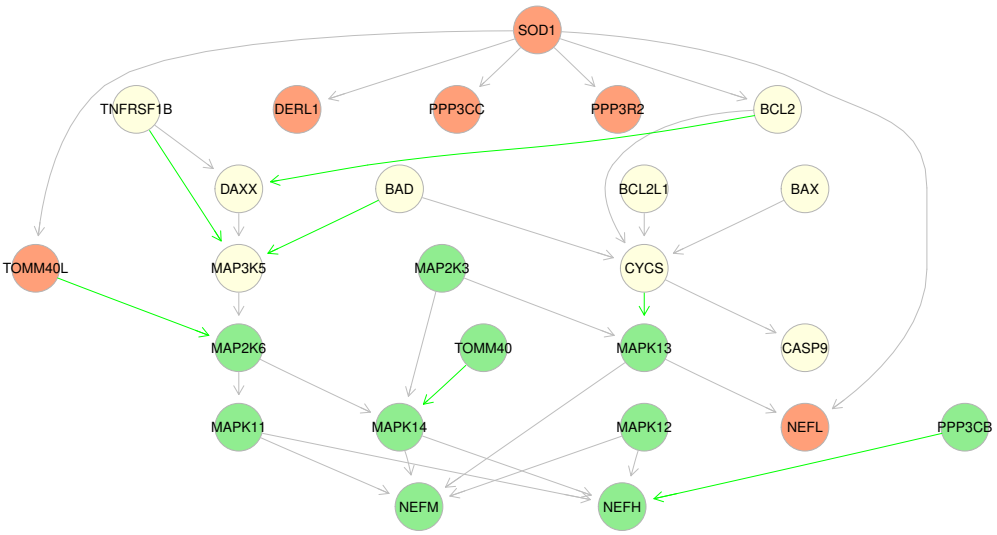

FIGURE 1.5: Colored clustering of the ALS model using `clusterScore` with the edge betweeness algorithm (`type = "ebc"`). Each cluster has its specific functional characterization: SOD1 and phosphatases module (lightsalmon), BCL2 and caspases module (lightyellow), MAPK module (lightgreen).

Arguments `graph` and `membership` correspond to the input graph and node membership, respectively. If the `map` argument is set to `TRUE`, the input graph is colored according to cluster membership (object `cg$graph`), as shown in Figure 1.5.

If we consider clusters as local models, we can extract and fit them through the function `extractClusters()`:

```
R> cls <- extractClusters(model$graph, model$data, alsData$group,
+                         membership = LV$membership)
R> print(cls$dfc)

  cluster N.nodes N.edges dev_df  srmr   pv.act   pv.inh
1     HM1       6       5  1.894 0.056 0.922706 0.043883
2     HM2      10      13  2.282 0.068 0.000000 0.011690
3     HM3       9      10  1.919 0.077 0.118250 0.108623
```

The object `cls` contains the list of clusters as separated igraph objects (`cls$clusters`) and a list of fitting results (`cls$fit`). The summary statistics shown above are stored in the object `cls$dfc`. From Figure 1.5, we observe that each cluster has its specific functional characterization: HM1: SOD1 and phosphatases module (lightsalmon), HM2: BCL2 and caspases module (lightyellow), HM3: MAPK module (lightgreen). All modules have good fit (srmr< 0.08), BCL2 module has down/up perturbation, and SOD1 module a down perturbation. Similar perturbations was observed on the cluster scores, LV1-LV3. In short, our analysis on the ALS data provides useful tools for model improvement in knowledge-based biological networks.

1.3 Network interrogation in Frontotemporal Dementia (FTD)

The aim of a standard **SEMgraph** analysis, illustrated in the previous Section, was to evaluate relevance and perturbation of every biological variable in the context of their shared interactions, extending its connectivity on the base of empirical data and possible exogenous influences, highlighting sources, effectors, causal paths connecting them, and their possible aggregation into modules in knowledge-based biological networks.
In this section, we describe three network (pathways) interrogation methods with **SEMgraph** for investigating specific type of biological questions applied in the Frontotemporal Dementia, a neurodegenerative disorder characterized by cognitive and behavioural impairments (Palluzzi et al., 2017). For this example, we will use DNA methylation data from Li et al., 2015 (GEO accession: GSE53740), stored in the **SEMdata** package as `ftdDNAme`, a list of two objects: `ftdDNAme$pc1`, a data matrix of 256 rows (subjects) and 16560 columns (genes) containing the value of the first principal component of DNA methylation levels, obtained applying a principal component analysis to methylated CpG sites within the promoter region, for each gene (genes with unmethylated CpGs in both conditions were discarded); and `ftdDNAme$group`, a binary group vector of 105 FTD patients (1) and 150 healthy controls (0).

1.3.1 Gene set analysis

Once differential expression of individual gene or genes that work together are identified, the next step in traditional genetic analysis is to infer their biological structure by pathways interrogation (or enrichment analysis) of literature-curated, gene-centric biological databases that connect genes to functional categories (terms) of Gene Ontology (Ashburner et al., 2000), or pathways of KEGG or others. A plethora of tools are available, such as overrepresentation analysis (ORA) or gene set enrichment analysis (GSEA) (Reimand et al., 2019). While older generations of approaches are still commonly used, topology-based methods (Nguyen, Mitrea, and Draghici, 2018), which incorporate the pathway structure, show superior performance in terms of improved sensitivity and specificity. Following the latter stream of literature, we propose a Gene Set Analysis (GSA) with a SEM-based procedure.
The core of the methodology is implemented in the RICF algorithm of `SEMrun()` recovering from RICF output node-specific group effect p-values, and Brown's combined p-values of node activation and inhibition (see Section 1.1.4 for details). Node-specific pvalues corrected for multiple comparisons $< \alpha$, with one of several adjustment methods, including Bonferroni or Benjamini- Hochberg procedure for controlling false discovery rate, i.e. FDR. (see `p.adjust()` function), are used for Differential Espression Genes (DEGs) identification. While, a single pvalue of the two Brown's pvalues ($P^{(+)}$: P-activation, and $P^{(-)}$: P-inhibition) combined with a Bonferroni procedure, i.e. $2 \times min(P^{(+)}; P^{(-)})$, indicates the specific-pathway perturbation.
Function `SEMgsa()` uses the RICF fitting `SEMrun(..., fit = 1, algo = "ricf")` to iteratively apply the GSA on a list of gene networks (in our example, KEGG signaling pathways). We refer the reader to the help documentation: `?SEMgsa()` and Chapter 2 for statistical details.

For computational efficiency purposes, pathways with less than 5 and more than 500 of nodes have been excluded from the analysis.

```
R> # load libraries
R> library(SEMgraph)
```

```
R> library(SEMdata)
R> library(huge)

R> # Nonparanormal transform of DNAme PC1 data
R> pc1.npn <- huge.npn(ftdDNAme$pc1); dim(pc1.npn)
R> # Set binary group classification
R> group <- ftdDNAme$group; table(group)

R> # Black list n < 5 | n > 500
R> n <- unlist(lapply(1:length(kegg.pathways),
               function(x) vcount(kegg.pathways[[x]])))
R> blacklist <- which(n < 5 | n > 500)
R> length(blacklist)
[1] 1
R> # SEM-based Gene Set Analysis (GSA)
R> pathways <- kegg.pathways[-blacklist]
R> GSA <- SEMgsa(pathways, pc1.npn, group, method = "BH",
+                alpha = 0.05)

R> # TOP 10 pathways
R> GSA$gsa[1:10, c(1:3,7)]

                     No.nodes  No.DEGs      pert          ADJP
Salmonella infection      249       25   up act 6.755414e-10
MicroRNAs in cancer       310       14 down inh 5.603214e-08
Alzheimer disease         384       29 down inh 1.139624e-07
Non-alcoholic fatty
liver disease             155       16   up act 1.244577e-07
Diabetic
cardiomyopathy            203       14   up act 1.542071e-07
Prion disease             273       20   up act 2.097764e-07
Apoptosis                 136       19   up act 2.187588e-07
Amyotrophic lateral
sclerosis                 364       14 down inh 2.995745e-07
Necroptosis               159        9   up act 3.590629e-07
```

Every pathway is listed in the `GSA$gsa` data.frame, reporting size (*No.nodes*), DEG number (*No.DEGs*), pathway perturbation column (*pert*), p-value for node activation (*pNa*), p-value for node inhibition (*pNi*) and the Bonferroni combination of them (*PVAL*). *ADJP* refer to the pathway combined p-value adjusted for multiple tests with Bonferroni correction, i.e., $ADJP = min(No.pathways \times PVAL; 1)$. In addition, the list `GSA$DEG` contains a vector of DEG IDs for each pathway, selected with p-value < `alpha` after Benjamini-Hochberg correction (`method = "BH"`). In this example, we used the `kegg.pathways` list, though any list of `igraph` network objects can be passed.

Gene set analysis methods can be evaluated in terms of their ability to rank to the top the pathways that are indeed relevant to a given condition (prioritization) and generate low p-values for the relevant pathways (sensitivity). The term Frontotemporal lobar degeneration can be associated to several pathways related to neurodegenerative disorders like Alzheimer disease, Parkinson disease and Amyotrophic lateral sclerosis. These three pathways are included in the top 20 pathways ranked according to the significance of their combined p-value. Alzheimer disease is ranked in third position with an highly Bonferroni significant p-value (1.14e-07) and overall pathway perturbation column indicates an overall reduction of node inhibition in cases with respect to control group. To summarise, results show high ability of SEMgsa both in terms of prioritization and sensitivity.

1.3.2 Fitting active disease modules

In the following, we want to build and fitting a SEM-based subgraph of the DNA methylation (DNAme) alterations caused by FTD, without an initial disease model. A common problem in network biology and medicine is to filter large models. Reducing large complex graphs by either extracting critical relationships or perturbed disease modules is key to focus relevant information into simpler subgraphs. Usually, the disease-specific subpathways (or modules) extraction process starts from a large interactome weighted with P-values, computed with some statistical methods. Then, a combinatorial, diffusion or gready algorithm is performed to search the "active" disease module (Liu et al., 2020; Ritchie et al., 2015).
Although not necessary, having a collection of known disease-associated networks is an advantageous starting point. For instance, the KEGG BRITE database allows to search for terms, including human disorders, that could be associated to one or more pathways. The term Frontotemporal lobar degeneration (an alias for FTD; KEGG ID: H00078) is associated to 6 KEGG pathways: *MAPK signaling pathway* (hsa04010), *Protein processing in endoplasmic reticulum* (hsa04141), *Endocytosis* (hsa4144), *Wnt signaling pathway* (hsa04310), *Notch signaling pathway* (hsa04330), and *Neurotrophin signaling pathway* (hsa04722). We can use the `SEMgsa()` utility to apply gene set analysis (GSA) on a collection of networks. Here, we explore the 6 selected FTD pathways.

```
R> # FTD-related pathway selection
R> ftd.pathways <- c("MAPK signaling pathway",
+                    "Protein processing in endoplasmic reticulum",
+                    "Endocytosis",
+                    "Wnt signaling pathway",
+                    "Notch signaling pathway",
+                    "Neurotrophin signaling pathway")
R> j <- which(names(kegg.pathways) %in% ftd.pathways)
R>
R> # Gene set analysis (GSA) with 5000 permutations
R> ftd.gsa <- SEMgsa(kegg.pathways[j], pc1.npn, group,
+                    n_rep = 5000)
```

From the list `ftd.gsa$DEG` of DEG IDs for each pathway, we extract a seed list of 60 DEGs to map on the KEGG interactome (`?kegg`), union of all KEGG pathways with 4242 nodes and 34975 edges, after graph weighting with the `weightGraph()` function (see help documentation: `?weightGraph`):

```
R> # Seed extraction
R> seed <- unique(unlist(ftd.gsa$DEG))
R> length(seed)
R> # KEGG interactome weighting
R> R> W <- weightGraph(kegg, pc1.npn, group, method = "r2z")
R> summary(W)
```

Here, `method="r2z"` argument set the perturbation edge information (i.e., the case-control edge difference). This method uses the r-to-z transform for testing the group difference between the correlation coefficients $r_{jk}^{(1)}$ and $r_{jk}^{(0)}$ of connected nodes j and k, by applying the Fisher's r-to-z transform: $z = 0.5 \log[(1+r)/(1-r)]$, with $t = (z^{(1)} - z^{(0)}) / \sqrt{1/(n_1 - 3) + 1/(n_0 - 3)}$ (Fisher, 1915).

We may then use the function `SEMtree()` to map on the reference graph W: (i) the user seed list (`seed = seed`), re-coded internally as binary variable, taking value 1 for a seed and 0 for a non-seed, (ii) the P-values from r-to-z Fisher's transform (`eweight = "pvalue"`), re-coded internally as inverse of negative log(pvalue).
In this way, edge weights in a positive continuous range $[0, \infty)$ and binary node

weights, are used to generate our reduced (or graph filtering) perturbed model via the `type = "ST"` method, the Steiner Tree algorithm suggested by (Kou, Markowsky, and Berman, 1981) (see help documentation: `?SEMtree` for other methods and Chapter 3 for statistical details):

```
R> # Steiner tree extraction
R> ST <- SEMtree(W, data= pc1.npn, seed=seed, type="ST",
+                eweight="pvalue")
R> summary(ST)
```

The Kou's algorithm is a very fast and efficient solution to find a connected subgraph of the input graph such that the additional nodes (called the *Steiner* or *connector* nodes) connecting seed nodes (called the *terminal* nodes) minimize the sum of the weight of every edge in the subgraph (i.e., maximizing edge perturbation between disease nodes).
The `kou` method yielded a Steiner tree `ST` of 92 nodes (44 terminal nodes on 60 input seeds and 48 Steiner nodes) and 90 directed edges and 1 bi-directed edge. We fit the filtered active (perturbated disease) module with`SEMrun()`. In addition, the SEM fitting can be improved using the deconfounding data extracted with the `SEMbap()` function, applying a grid of significance levels to select `alpha = 5E-06`, a trade-off which reduce the badness-of-fit measures and conserve the largest perturbation signal present in the original data (see Chapter 4).

```
R> # Perturbation evaluation with raw data
R> sem1 <- SEMrun(ST, pc1.npn, group)

NLMINB solver ended normally after 7 iterations
deviance/df: 5.894403  srmr: 0.4099542
Brown's combined P-value of node activation: 0.000430902
Brown's combined P-value of node inhibition: 3.476108e-13

R> # Perturbation evaluation with deconfounding data
R> adj.pc1 <- SEMbap(ST, pc1.npn, method= "bonferroni",
+                    alpha = 5E-06)$data
R> adj.sem1 <- SEMrun(ST, adj.pc1, group)

d-separation test (basis set) of 4096 edges...
Number of significant local tests: 1208 / 4096

NLMINB solver ended normally after 10 iterations
deviance/df: 1.614046  srmr: 0.07365701
Brown's combined P-value of node activation: 0.03450855
Brown's combined P-value of node inhibition: 0.00997967
```

The most dense subgraph extracted with the tree agglomerative hierarchical clustering (TAHC) method (`C=clusterGraph(ST, type="tahc")`) of inferred FTD perturbed backbone is shown in Figure 1.6 (`gplot(C$HM3, l="neato")`). The latter shows node MAPK13, connecting through Steiner nodes to others DEGs, being an hub in FTD network, showing the central role of MAPK signaling pathway in neurodegenerative diseases. MAPK13 is corroborated also after data de-correlation as key-gene (node-wise P = 0.005). This is not surprising given that MAPK signaling has been linked to a variety of neurodegenerative disorders, including FTD (Chen-Plotkin et al., 2008; Santiago, Bottero, and Potashkin, 2020). Tau, a microtubule-associated protein, is the main component of intracellular filamentous inclusions implicated in tauopathies such as Alzheimer's disease (AD), Frontotemporal Dementia with Parkinsonism-17 (FTDP-17), Pick disease (PiD), Progressive Supranuclear Palsy (PSP),

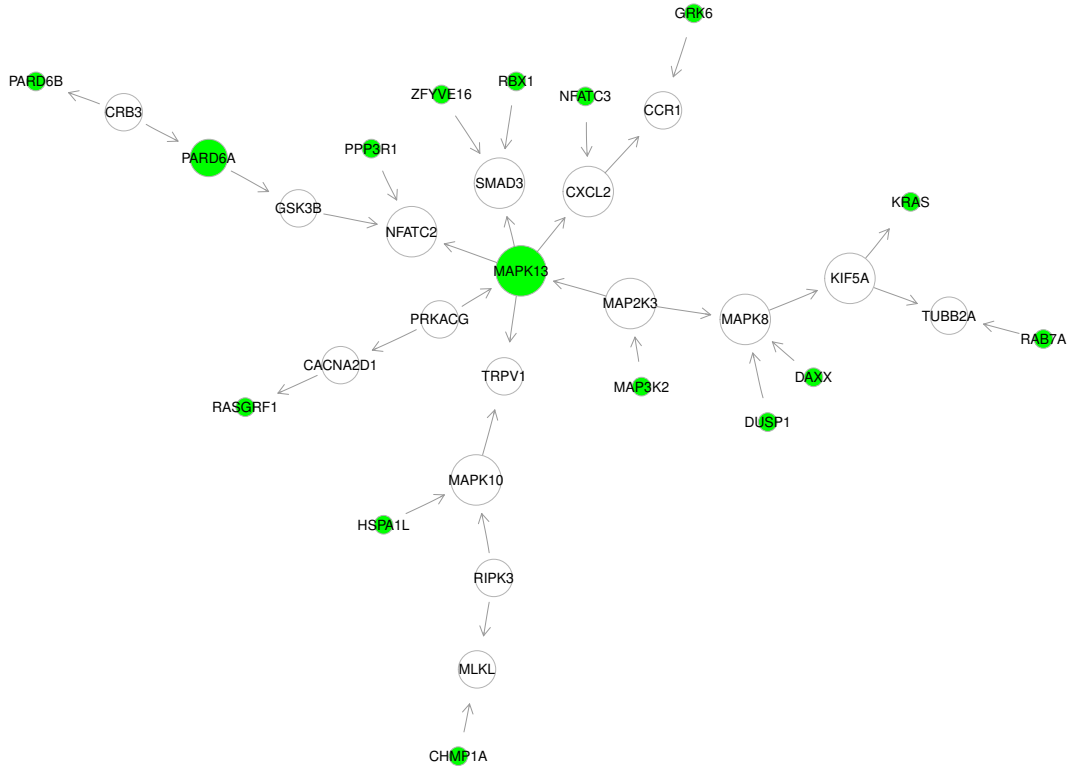

FIGURE 1.6: FTD perturbed backbone estimated from DNA methylation data. This perturbed backbone is extracted from the weighted input graph, maximizing both edge perturbation (i.e., minimizing the total weight of the tree) while traversing all the seeds (i.e., the Steiner connectors), defined as the nodes that are significantly perturbed by the diseased phenotype. The Steiner tree has 92 nodes (44 terminal nodes on 60 input seeds and 48 Steiner nodes) and 90 directed edges and 1 bi-directed edge oriented using edge directions of the KEGG interactome. For graph readability purposes, the most dense cluster derived from tree agglomerative hierarchical clustering (TAHC) method, in terms of both nodes and connections has been shown. This procedure results in a network with 32 nodes and 31 edges. Terminal nodes are coloured in green, and node sizes are proportional to node degrees.

and Corticobasal Degeneration (CBD). Tau is phosphorylated by a variety of kinases that fall into four categories. GSK3, MAPK13, and AMP-activated protein kinase have recently been discovered to have a role in in vivo Tau phosphorylation utilizing several cell lines (Gao et al., 2018). These results suggest that the hub nodes identified by SEM fitting a Steiner tree are important signatures of neurodegenerative disease.

1.3.3 Locating differentially connected genes

With the recent advances in high-throughput sequencing under different contexts and cell types, there is an increasing demand for approaches that learn changes in causal (regulatory) relationships between two related gene regulatory networks corresponding to different conditions. Inferring changes in gene regulatory networks might show, for example, that a single gene regulates various groups of target genes

under different disease states. In this section, we present a difference causal inference (DCI) procedure based on two-group constrained Gaussian Graphical Modeling (CGGM) algorithm of `SEMrun()`. The aim is to provide direct estimation of the difference regolatory graph integrating know genetic pathways into a DCI of observational data from two conditions. We focus on detecting changes in network edges. This is crucial in the study of biological systems, which investigates how the network of connected genes changes from one condition to another, to provide a deeper and more comprehensive understanding of complex disease (Ideker and Krogan, 2012). Here, we consider the new graph obtained from the union of FTD KEGG pathways in Section 1.3.2 of 586 nodes and 3572 edges.

```
R> # Input graph as the union of FTD KEGG pathways
R> gU <- graph.union(kegg.pathways[j])
R> gU <- properties(gU)[[1]]
R> summary(gU)
```

Function `SEMdci()` (see help documentation: `?SEMdci`) allows to create a network with perturbed edges obtained from the output of `SEMrun(..., fit = 2, algo = "cggm")`. To increase the efficiency of computations for large graphs, users can select to break the network structure into clusters (see `?clusterGraph`). The function `SEMrun()` is applied iteratively on each cluster to obtain the graph with the full list of perturbed edges, defined by P-value $< \alpha$, after multiple test correction. This procedure via short random walks (`type="wtc"`) algorithm with default False Discovery Rate (`method="BH"`, `alpha=0.05`) and minimum cluster size=10 leads to a graph of 111 nodes and 103 edges with four components with size > 5. For descriptive purposes, the third component (19 nodes and 18 edges) of the latter graph has been retained for confirmatory SEM fit since it shows the largest proportion of edge perturbation, although the analysis can be applied to the other network components. Group influence is first modeled as an exogeneous variable acting on every node. Then, the output graph obtained from node perturbation has been evaluated via the two-group model, with the aim of obtaining a subgraph where the nodes and edges are statistically significant in difference between the two groups of cases and controls.

```
R> gD<-SEMdci(gU, pc1.npn, group, type="wtc", method="BH",
+             alpha=0.05)

modularity = 0.6870652

Community sizes
 13  16  12   8  11  14   7   3   9   4  15   6   1   2  10   5
  2   2   6   7   7   8   9  10  13  20  26  47  97 101 114 117

#  Perturbation evaluation of the 3th component of gD
R> gC<- properties(gD)
R> gC3 <- gC[[3]]
R> sem1 <- SEMrun(gC3, pc1.npn, group, fit=1, algo="cggm")

GGM (constrained) solver ended normally after 0 iterations
deviance/df: 13.61558  srmr: 0.2878324
Brown's combined P-value of node activation: 1.268641e-11
Brown's combined P-value of node inhibition: 6.987595e-05

R> sem2 <- SEMrun(sem1$graph, pc1.npn, group, fit=2, algo="cggm")

GGM (constrained) solver ended normally after 0 iterations
deviance/df: 7.496005  srmr: 0.3115255
```

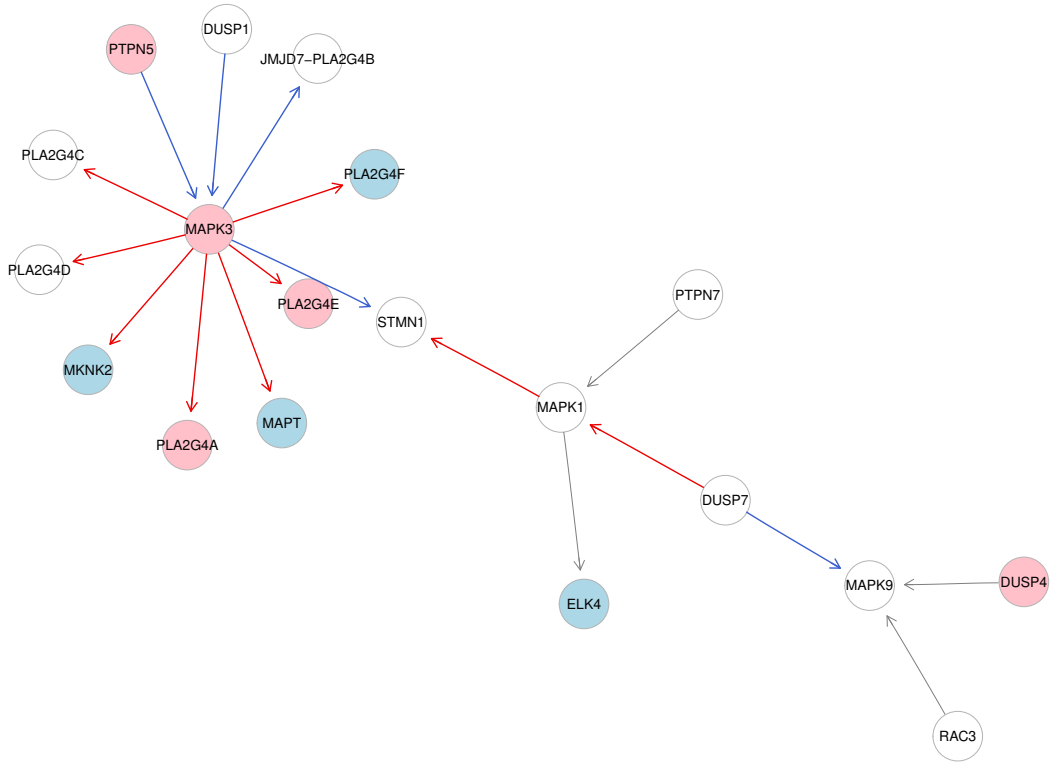

FIGURE 1.7: FTD perturbed backbone estimated from DNA methylation data via difference causal inference (DCI) procedure. Constrained Gaussian Graphical Modeling (CGGM) algorithm of `SEMrun()` outputs the difference regulatory graph between the two conditions, i.e., the edges in gene regulatory networks that appeared, disappeared or changed weight between the two conditions. Node and edge color coding follow the same rules applied in figure 1.4B.

```
Brown's combined P-value of edge activation: 1.58793e-07
Brown's combined P-value of edge inhibition: 5.573764e-10
```

The global statistics suggest a poor fitting, and de-correlation could be performed, but an extreme significant group perturbation is observed. Thus, for illustrative purpose, we display in Figure 1.7 the inferred FTD perturbed backbone obtained from DCI. As in Figure 1.6, MAPK genes represent most of the nodes in the perturbed network with MAPK3 being the receptor and emitter of perturbed edges from a large proportion of network nodes. Alzheimer's Disease (AD), Prion, Major Depressive Disorders (MDD), and Frontotemporal lobar degeneration (FTLD) all contain the MAPK3 gene, suggesting that these disorders may share a mechanism (Ge and Jakobsson, 2018). MAPK3 is a key component of the MAP signaling pathway, which transports signals from the cell surface to the nucleus. MAPK3 is one of a small handful of genes reported to exhibit different promoter use and splicing in normal versus AD brain tissue (Twine et al., 2011). These results confirm that SEM-based DCI can identify biomarkers and their interactions in terms of changes of a gene network.

Chapter 2

SEMgsa( )

2.1 Background

Biomedical research has been transformed by recent advances in high-throughput technologies, enabling extensive monitoring of complex biological systems. As a result, new methodological developments have emerged, most notably the adaptation of systems perspectives to analyze biological systems. Pathway enrichment has become a key tool in the analytic pipeline for Omics data and has been effectively used to generate novel biological hypotheses and determine if specific pathways are linked to specific phenotypes. In the literature, dozens of strategies have been developed, varying in model complexity and effectiveness.

Earlier methodologies, such as over-representation analysis (ORA) (Al-Shahrour, Díaz-Uriarte, and Dopazo, 2005) and gene set analysis (GSA) (Efron and Tibshirani, 2007; Subramanian et al., 2005), treat each pathway as a collection of biomolecules, as Khatri, Sirota, and Butte, 2012 point out in their review paper. The ORA approach used a list of differentially expressed (DE) genes as input to determine which sets of DE genes are over-represented or under-represented, being strongly reliant on the criteria used to choose the DE genes, such as the statistical tests and thresholds utilized.

A second generation of approaches was created to reduce this reliance on gene selection criteria by taking into account all gene expression values. The hypothesis behind these approaches is that small yet coordinated changes in groups of functionally related genes may be crucial in biological processes. These methods are named Functional Class Scoring Methodologies (FCS) (Ackermann and Strimmer, 2008). Such methods include Gene Set Enrichment Analysis (GSEA) (Efron and Tibshirani, 2007), Gene Set Analysis (GSA) (Jacob, Neuvial, and Dudoit, 2012) and Correlation Adjusted Mean Rank gene set test (CAMERA) (Mitrea et al., 2013) among others.

ORA and FCS methods can be referred to as the first two generations of pathway enrichment analysis approaches. However, when pathways are seen as a basic unstructured and unordered collection of genes, all the genetic connections and interactions that are supposed to capture and characterize the actual processes at hand are simply neglected.

With the aim of including all of this additional information into the analysis, topology-based (TB) approaches have been created. These methods account for interactions between biomolecules and provide better performance than standard second-generation methods (Efron and Tibshirani, 2007; Subramanian et al., 2005). A variety of tools have been implemented, such as DEGraph (Jacob, Neuvial, and Dudoit, 2012), topologyGSA (Massa, Chiogna, and Romualdi, 2010), NetGSA (Shojaie and Michailidis, 2009; Ma, Shojaie, and Michailidis, 2019; Hellstern et al., 2021), Pathway-Express (Draghici et al., 2007; Khatri et al., 2007), SPIA (AL et al., 2009)

among others. The common feature of approaches in this category is that they use prior knowledge of pathway topology information to obtain some gene-level statistic, which is then used to produce a pathway-level statistic, which is then used to rank the pathways.

The goal of pathway enrichment approaches is to compare the 'activity' of interest pathways across two or more biological situations or groups of specimens (patients, cell lines, etc.). Another technical feature that distinguished pathway enrichment methods is the type of the statistical null hypothesis being tested. The majority of approaches may be divided into two categories: those that test (I) self-contained null hypotheses and (II) competitive null hypotheses (Goeman and Bühlmann, 2007). A self-contained null hypothesis examines the activity of each pathway across biological situations (for example, normal vs. illness samples) without comparing it to the activity of other biomolecules/pathways. On the other hand, the activity of each pathway is compared to that of other biomolecules/pathways in a competitive null hypothesis. Even if the competitive null hypothesis has an interesting interpretation, assessing the significance of the competitive null is challenging, since tests based on it take into account a framework for gene sampling that treats genes as independent.

The main contribution of this chapter is the development of a self-contained topology-based algorithm developed into the framework of Structural Equation Models (SEM), called `SEMgsa()` (Palluzzi and Grassi, 2021; Pepe and Grassi, 2014). Evaluation of system perturbation is incorporated in SEM (Bollen, 1989), where the experimental condition is compared to a control one through the use of an exogeneous group variable acting on every node of the network. Statistical significance of specific-pathway score is obtained combining node activation and node inhibition statistics extracted from SEM model fitting. In addition, unlike existing methods, an overall status of pathway perturbation of genes between case and control group has been computed considering both node perturbation and up- or down- regulation of genes for gaining more biological insights into the functional roles of predetermined gene subsets.

A second objective of this study is to provide a consistent optimum solution of any given biological situation. Many topology-based methods that investigate distinct null hypothesis have been proposed in literature. We compare five popular pathway analysis approaches to `SEMgsa()`, starting from the most similar ones in terms of multivariate test and self-contained hypothesis type (DEGraph, NetGSA and topologyGSA) together with one approach of competitive hypothesis type (PathwayExpress) and the older approach of over-representation analysis (ORA). All the methods in this chapter offer a nice use interface in R.

The aforementioned methods have been applied on observed and simulated expression data. The ultimate goal of expression data application is to provide a meaningful comparison of gene set analysis methods in terms of (i) sensitivity and (ii) prioritization for observed data and (i) type I error and (ii) power within each simulation run.

The remainder of the chapter is organized as follows. Firstly, we describe `SEMgsa()` features with regard to gene expression data both in terms of inference procedure and user interface. Then, we outline the experimental setup constructed to evaluate pathway enrichment methods, including real data application and simulation design. In the end, we provide the results together with overall discussion.

2.2 Method and implementation

2.2.1 SEM framework

A Structural Equation Model (SEM) (Bollen, 1989; Shipley, 2016) is a statistical framework for causal inference based on a system of structural equations defining a *path diagram*, represented as a graph $G = (V, E)$, where V is the set of nodes (i.e., variables) and E is the set of edges (i.e., connections). The set E may include both directed edges $k \to j$ if $k \in \text{pa}(j)$ and bidirected edges $k \leftrightarrow j$ if $k \in \text{sib}(j)$. Although in the general setting of SEM latent variables and non-linear functions can be included (Bollen, 1989), we focus on the special case where the *parent* set $\text{pa}(j)$, and the *siblings* set $\text{sib}(j)$, determine a system of *linear* equations, as follows:

$$Y_j = \sum_{k \in \text{pa}(j)} \beta_{jk} Y_k + U_j \qquad j \in V \tag{2.1}$$

$$\text{cov}(U_j; U_k) = \begin{cases} \psi_{jk} & \text{if } j = k \text{ or } k \in \text{sib}(j) \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where Y_j and U_j are an observed variable and an unobserved error term, respectively; β_{jk} is a regression (path) coefficient, and a covariance ψ_{jk} indicates that errors are dependent, which is assumed when there exists an unobserved (i.e.latent) confounder between k and j.
Under such a model, the dependence structure among genes provided by pathways in biological database with directed and/or bi-directed edges (i.e., KEGG, Reactome, and many others) (Kanehisa and Goto, 2000; Jassal et al., 2020), interacting with each other to generate a single biological effect, can be included explicitly though the graph, $G = (V, E)$ and evaluated using local and global statistics.
`SEMgsa()` procedure adds a binary group (treatment or disease class) node labeled X to V, and suppose that $X = \{0, 1\}$ directly affects the set of genes in the pathway. As bonus, adding group node and group-genes edges, the pathway with several components (clusters) and singleton genes induces a connected graph (see Figure 2.1).

Thus we consider a directed graph $G = (V \cup X, E \cup E_X)$ with the linear structural equations:

$$Y_j = \beta_j X + U_j \qquad j \in V(x) \tag{2.3}$$

$$Y_j = \sum_{k \in \text{pa}(j)} \beta_{jk} Y_k + \beta_j X + U_j \qquad j \in V(y) \tag{2.4}$$

where $V(x)$ and $V(y)$ are the sets of exogenous (i.e., source and singleton genes) and endogenous (i.e., connectors and sinks) genes, respectively. The covariances, $\text{cov}(U_j; U_k)$ are assumed to be equal to Equation (2.2).
Comparing Equation (2.1) with Equation (2.3)-(2.4), we note that the added node X may affect the mean gene expression values, but not their regression paths or covariances. In the R package **SEMgraph** these coefficients can vary by experimental or disease group via two-group SEM (Palluzzi and Grassi, 2021; Pepe and Grassi, 2014), but in the following we assume additive group effects. Coefficients β_j (adjusted by the parents of the j-th node) determine the effect of the group on the j-th node, while the common path coefficients β_{jk} represent regression coefficients, adjusted by parent set and group effect.

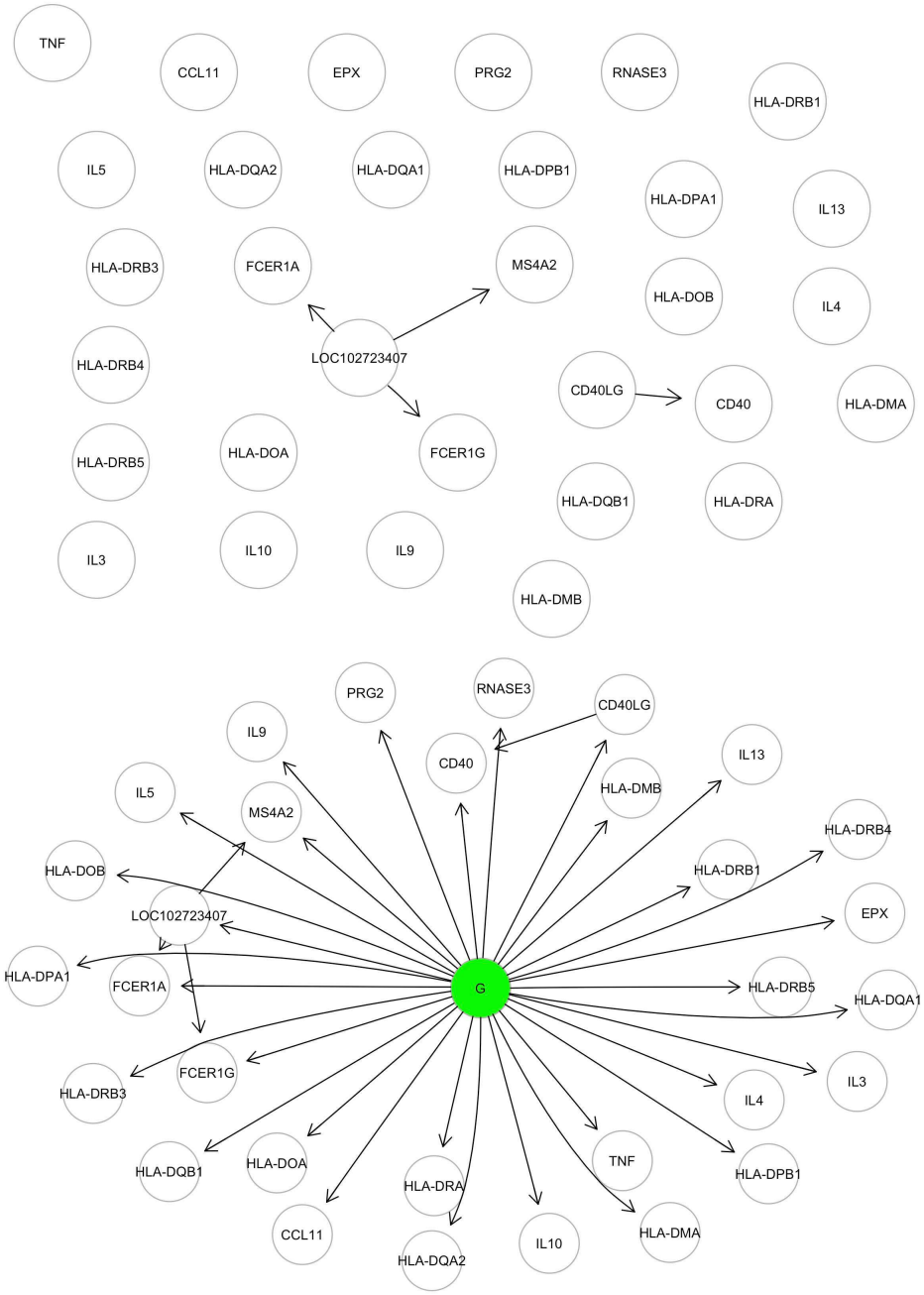

FIGURE 2.1: Visualisation of SEMgsa() procedure starting from Asthma KEGG pathway. The first graph summarise Asthma network properties, showing a pathway consisting of 31 nodes, 4 edges and 25 singletons. To maximise pathway information, SEMgsa procedure adds a binary group node (G = {0,1}) that directly affects the set of genes in the pathway. In this way, the pathway with numerous singleton genes is edged with a group node and group-genes, resulting in a linked graph.

This type of SEM enables the identification of differentially expressed genes (DEGs) if genes show a statistically significant variation in their activity (e.g., gene expression) in the experimental group respect to the control one. In other terms, a test for the null value of the path, $X \rightarrow Y$ is a test of:

$$H_0 : Y_j \perp \{X\} \quad vs. \quad H_1 : Y_j \not\perp \{X\}, \quad j \in V(x) \tag{2.5}$$

$$H_0 : Y_j \perp \{\mathrm{pa}(\mathrm{Y_j}), X\} \quad vs. \quad H_1 : Y_j \not\perp \{\mathrm{pa}(\mathrm{Y_j}), X\}, \quad j \in V(y) \tag{2.6}$$

From Equation (2.5)-(2.6) we note two different tests. Marginal tests of conventional DEGs analysis (Ackermann and Strimmer, 2008) for source and singleton genes, and conditional tests, given the parents, for connectors and sink genes. Conditioning increases power when there is a direct group effect, and reduces gene variability. So, pathway topological structure makes the inference more precise (Edwards, Wang, and Sørensen, 2012).
Maximum Likelihood Estimates (MLEs) of the paths ($X \to Y$) can be easily obtained with one of the three algorithms of **SEMgraph**. Specifically, the core of model fitting in `SEMgsa()` function relies on the Residual Iterative Conditional Fitting (RICF), an efficient iterative algorithm that can be implemented through least squares, with the advantage of clear convergence properties (Drton, Eichler, and Richardson, 2009), and permutation-based p-values for testing null hypothesis in Equation (2.5)-(2.6).
P-values of group effect ($X \to Y$) are computed by randomization of group labels comparing the estimated parameters by RICF with their random resampling distribution after a sufficiently high number of case/control labels permutations. Accurate small p-value estimations are possible with no need for a large number of permutations (`SEMgsa()` makes default = 1000 permutations), using the moment based approximation proposed by Larson and Owen, 2015. Once the empirical distribution of the permuted path coefficients is obtained, the two-sided p-values are extracted from the normal distribution with mean and standard deviation estimated from the empirical distribution.
From node-wise p-values, overall group perturbation on pathway genes can be computed based on the Brown's method for combining non-independent, one-sided significance tests (Brown, 1975). The method computes the sum of one-sided p-values: $X^2 = -2\sum_j \log(p_j)$, where the direction is chosen according to the alternative hypothesis (H_1), and the overall p-value is obtained from the chi-square distribution with new degrees of freedom f and a correction factor c to take into consideration the correlation among p-values, resulting in $\frac{X^2}{c} \sim \chi^2(f)$.
The conversion of two-sided p-values in one-sided p-values is performed according to the sign of the path ($X \to Y$) coefficient, β_j. We refer the reader to Section 1.1.4 of Chapter 1 for detailed equation reference.
Node-wise p-values $< \alpha$ (after correcting for multiple comparisons with one of several adjustment methods, including Bonferroni or Benjamini- Hochberg procedure), are used for DEGs identification. While, a single p-value of the two Brown's p-values ($p^{(+)}$: p-activation, and $p^{(-)}$: p-inhibition) combined with a Bonferroni procedure (Vovk and Wang, 2020), i.e. $2 \times min(p^{(+)}; p^{(-)})$, indicates the global pathway perturbation.
In some cases, edge weights are defined in signalling pathways with discrete values [-1, 0, 1], indicating gene-gene activity derived from biological database (e.g. KEGG). Usually they are: -1 for repressed or inactive, 0 for neutral, and +1 for enhanced or activated. For gaining more biological insights into the functional roles of prior subset of genes, the sign of the minimum p-value between node activation and inhibition has been retained to assess, in combination with pathway weights, an overall status of pathway perturbation of genes between case and control group.

In detail, node perturbation obtained from RICF fitting has been combined with up- or down-regulation of genes to obtain overall pathway perturbation classification as displayed in Table 2.1.

TABLE 2.1: Overall pathway perturbation.

Up/down regulation	Node perturbation	Overall perturbation
+1	P- (inh)	down act
-1	P- (inh)	up inh
+1	P+ (act)	up act
-1	P+ (act)	down inh

- The weighted adjacency matrix of each pathway was used to determine the up- or down-regulation of genes (taken from the KEGG database) as the column sum of weights across each source node. The pathway is marked as down-regulated if the total sum of the node weights is less than 1, and otherwise as up-regulated.
- The minimum among the p-values determines whether the node perturbation is activated or inhibited; if positive, the node perturbation is described as activated, and otherwise as inhibited.
- It is possible to determine the direction (up or down) of gene perturbation by combining these two quantities. In cases compared to the control group, an up- or down-regulated gene status that is associated with node inhibition shows a decrease in activation (or an increase in inhibition). In contrast, up- or down-regulated gene status, associated with node activation, leads to an increase in activation (or decrease in inhibition) in cases relative to control group.

2.2.2 User interface

The example code of the function `SEMgsa()` is as follows.

```
SEMgsa(g = list(), data, group, method = "BH",
       alpha = 0.05, n_rep = 1000, ...)
```

The inputs are: a list of pathways to be examined (g); gene expression data where rows represent subjects, and columns graph nodes ($data$); a binary vector with 1 for cases and 0 for control subjects ($group$). Optional inputs are the multiple testing correction method ($method$), and the significance level ($alpha$) for DEGs selection, and the number of randomization replicates for RICF algorithm (n_rep, default = 1000). The first step in `SEMgsa()` workflow is to compute the weighted adjacency matrix of each pathway, obtain the sum of node weights and flag the pathway as up- or down-regulated. This is crucial to obtain the overall pathway perturbation status at the end. Then RICF algorithm of R package **SEMgraph**, i.e. `SEMrun(graph, data, group, fit = 1, algo = "ricf")`, is applied on data, considering the group binary vector and the number of specified randomization replicates. More specifically, `SEMrun()` takes as input a single $graph$ as an $igraph$ object and has several additional inputs, including: a numeric value fit indicating the SEM fitting mode, where $fit = 1$ specifies a "common" model to evaluate group effects on graph nodes; the MLE method $algo$ is used for model fitting, in this case fitting is done via $RICF(algo = "ricf")$.

The covariance matrix could not be semi-definite positive in the situation of large dimensionality (n.variables > n.subjects), making it impossible to estimate the parameters. When this occurs, regularization of the covariance matrix is enabled. `SEMrun()` uses internally two functions of the *corpcor* R package:
the `is.positive.definite()` tests if the observed covariance matrix is positive, and if the response is equal to `FALSE`, the function `pcor.shrink()` implements the James-Stein-type shrinkage estimator (Schäfer and Strimmer, 2005) to regularized the covariance matrix.

Node-wise group effect p-values are extracted from model fitting object together with the number of DEGs obtained adjusting p-values with the chosen correction *method* while testing the specified level of $alpha$. Then, a data frame of combined SEM results is obtained putting together node-wise p-values with Brown's method and Bonferroni's correction.

The output of `SEMgsa()` is represented by a list containing two objects with the following information for each pathway in the input list:

- **gsa**, a dataframe reporting size (*No.nodes*), DEGs number (*No.DEGs*), pathway perturbation status (*pert*), Brown's combined p-value of pathway node activation (pNA), Brown's combined p-value of pathway node inhibition (pNI) and the Bonferroni combination of them ($PVAL$). $ADJP$ refer to the pathway combined p-value adjusted for multiple tests with Bonferroni correction, i.e., $ADJP = min(K \times PVAL; 1)$, where K is the number of the input pathways.
- **DEG**, a list with DEGs names for each pathway, selected with p-value $< alpha$ after the multiple correction procedure with one of the method available in R function `p.adjust()`. By default, *method* is set to "BH" (i.e., Benjamini-Hochberg correction) and the significance level $alpha$ to 0.05.

To read more about `SEMgsa()` function, in terms of description, usage, function arguments and value, refer to https://rdrr.io/cran/SEMgraph/man/SEMgsa.

2.3 Experimental design

2.3.1 Benchmark data

Coronavirus disease (COVID-19) RNA-seq expression data from Carapito et al., 2021 (GEO accession: GSE172114) together with Frontotemporal Dementia (FTD) DNA methylation data (DNAme) from Li et al., 2015 (GEO accession: GSE53740) have been used as benchmark data. Network information has been retrieved from the object `kegg.pathways` of the **SEMgraph** package as a list of 225 edge weighted *igraph* objects corresponding to the KEGG pathways extracted using the *ROntoTools* R package (Ansari et al., 2017). Edge weights are defined with discrete values [-1, 0, 1]: -1 for inactive gene-gene activity, 0 for neutral, and +1 for activated.

Coronavirus disease (COVID-19)

Coronavirus disease of 2019 (COVID-19) is a highly contagious respiratory infection that is caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Multiple probes for each Entrez gene ID were first eliminated. The empirical Bayes technique, as implemented in the *limma* R package (Smyth, 2005), was

used to fit linear models for differential expression analysis, and p-values were adjusted for multiple testing using the method of Benjamini-Hochberg (Benjamini and Hochberg, 1995). This procedure results in a matrix of 69 subjects $\times$ 14000 genes. Subjects include patients in the intensive care unit with Acute Respiratory Distress Syndrome ("critical group", N=46) defined as cases, and those in a non-critical care ward under supplemental oxygen ("non-critical group", N=23) defined as controls. The expression matrix was finally matched with the corresponding Coronavirus disease - COVID-19 (*hsa*05171) KEGG pathway according to its name. The latter is a graph with 232 nodes and 208 edges, including five components and 109 sigleton (i.e. node degree = 0). The maximum subgraph consists of 54 nodes and 83 edges. This pathway was subsequently labeled as target pathway and its p-value and rank were further investigated for assessing the sensitivity and prioritization ability of the methods (Tarca, Bhatti, and Romero, 2013; Tarca et al., 2012).

Frontotemporal Dementia (FTD)

Frontotemporal Dementia, a neurodegenerative disorder characterized by cognitive and behavioural impairments (Palluzzi et al., 2017). We will use DNAme data stored in the **SEMdata** package as `ftdDNAme`, a list of two objects: a data matrix of 256 rows (subjects) and 16560 columns (genes) containing the value of the first principal component of DNAme levels, obtained applying a principal component analysis to methylated CpG sites within the promoter region, for each gene (genes with unmethylated CpGs in both conditions were discarded); and a binary group vector of 105 FTD patients (1) and 150 healthy controls (0). Unlike COVID-19 data, FTD has not a unique KEGG pathway associated to its name. According to KEGG BRITE database, the term Frontotemporal lobar degeneration (FTD; KEGG ID:H00078) is associated to 6 KEGG pathways: MAPK signaling pathway (*hsa*04010), Protein processing in endoplasmic reticulum (*hsa*04141), Endocytosis (*hsa*4144), Wnt signaling pathway (*hsa*04310), Notch signaling pathway (*hsa*04330), and Neurotrophin signaling pathway (*hsa*04722). We can use the `SEMgsa()` function to apply gene set analysis (GSA) on a collection of networks, exploring the 6 selected FTD pathways as target ones. Thus, the ability of GSA methods will be investigated on 6 target pathways, combining results in terms of median p-values and ranks for readability purposes.

2.3.2 Data simulations

Synthetic data, based on realistic expression data (COVID-19), was used to carry out simulations following the practice in Ma, Shojaie, and Michailidis, 2019. A subset of pathways $q_1 < K$ out of total K pathways has been chosen to be dis-regulated. Next, a pre-specified number (s) of genes within each dis-regulated pathway was chosen to be altered (up or down) according to a topological measure (betweenness, community, neighbourhood) and different mean signals (± 0.5, ± 0.6, ± 0.7). Another subset of $q_0 < (K - q_1)$ pathways with no dis-regulated genes has been chosen to evaluate statistical metrics. Finally, the COVID-19 benchmark data matrix is first normalized, with mean zero and unit variance for each gene within each group (cases and controls). Then, nine data generation procedures are executed, according to topological measures and adding mean signal to the pre-specified genes in the selected dis-regulated pathways. In summary, the simulation design (3×3) with 100 randomization per design levels is reported in Table 2.2.

TABLE 2.2: Summary of simulation design (3×3) with 100 randomization per design levels.

		Mean signal		
Topology design	Gene regulation	+/- 0.5	+/- 0.6	+/- 0.7
betweenness	up/down	100	100	100
community	up/down	100	100	100
neighbourhood	up/down	100	100	100

Pathway deregulation

Each data generation procedure starts with the definition of a list of pathways to be tested. After recalling the list of `kegg.pathways` including $N = 225$ signaling pathways from the KEGG database, for efficiency purposes pathways with a minimum and a maximum number of nodes equal respectively to 30 and 300 have been filtered out for the analysis. Then, to speed up computations, the maximum component of each *igraph* (Csardi and Nepusz, 2006) object corresponding to each selected pathway has been selected. Given this choice, *igraph* objects with maximum component smaller than the 60% of the total graph size have not been considered. This filtering procedure results into a list of $K = 117$ *igraph* objects.
We have to alter $q_1 = 10$ pathway's genes in order to deregulate it within a simulation. Specifically, we consider the 9 KEGG pathways associated to Coronavirus disease - COVID-19 (*hsa*05171): Vascular smooth muscle contraction (*hsa*04270), Platelet activation (*hsa*04611), Toll-like receptor signaling pathway (*hsa*04620), NOD-like receptor signaling pathway (*hsa*04621), RIG-I-like receptor signaling pathway (*hsa*04622), JAK-STAT signaling pathway (*hsa*04630), Natural killer cell mediated cytotoxicity (*hsa*04650), Fc gamma R-mediated phagocytosis (*hsa*04666) and Leukocyte transendothelial migration (*hsa*04670).
We looked at three different methods for reflecting pathway topology in order to assign impacted genes to the deregulated pathways: betweenness, community and neighborhood, following the practice in Bayerlová et al., 2015; Ma, Shojaie, and Michailidis, 2019.
The number of all shortest paths in a directed graph that pass through a given node is known as its betweenness. The top 10 highest scoring betweenness nodes were used to choose affected genes in the *betweenness deregulation* design. According to the *community deregulation* design, we located modules with dense connections between module nodes and spare connections between module nodes. Given the division of vertices in each community, the 10 affected genes are then randomly sampled from the community with the highest proportion of members. In the *neighbourhood deregulation* procedure, the vertices not farther than a given limit from another fixed vertex are called the neighborhood of the vertex. After computing the neighbourhood of order 2, we sampled the 10 vertices from the neighbourhood with the biggest size.
Within each of the associated pathways plus the target pathway of interest, an equal number of genes s has been selected as 'dysregulated'. We decided to fix the number of affected genes to $s = 10$ for each pathway, with the aim of obtaining equal contribution from each associated disease (due to the presence of smaller pathway sizes). However, given that overlapping genes between pathways may occur, the unique genes out of the total $S \leq s \times q_1 = 10 \times 10$ have been retained as DEGs for pathway dysregulation. Thus, the number of total dysregulated genes S is not fixed, but changes according to the chosen topology design. As a note, given the random sampling within the community deregulation design, the sampled genes

change according to the specified seed. In the end, a weight representing the up- or down-regulation of genes was kept along with the Entrez gene ID of the affected genes. The weighted adjacency matrix of each pathway comprises a column regarding the sum of weights over each source node, which can be used to determine up- or down-regulation (taken from the KEGG database). The pathway is marked as down-regulated or up-regulated depending on whether the total sum of node weights is less than 1. Each gene's weight has been extracted in order to obtain an up or down mean signal, which better reflects variations in the expression of the impacted genes between the control and treatment groups.
After identifying the subset of DEGs according to the chosen topology design, pathways with a number of dysregulated genes ≤ 1 have been selected as true negatives (q_0) to evaluate type I error from simulations. Unlike the q_1 number that is fixed to 10 COVID-19 related pathways, the number of q_0 pathways changes according to the chosen subset of DEGs.
To summarise, for all topology design, the total altered genes S inside the $q_1 = 10$ pathways are differentially expressed with a mean difference varying from (± 0.5, ± 0.6, ± 0.7). Note that the magnitude of the mean signal is expressed relative to the unit variance of each gene (see Ma, Shojaie, and Michailidis, 2019).
The simulated expression matrices were directly supplied to `SEMgsa()` and DEGraph, topologyGSA, NetGSA, PathwayExpress and ORA algorithms together with the list of *igraph* objects corresponding to the chosen KEGG pathways.

Pathway enrichment methods

Table 2.3 provides an overview of the tested pathway enrichment methods in terms of null hypothesis, input requirements, pathway information and availability on R together with main papers for reference. These methods differ in two main aspects: (i) the type of null hypothesis, self-contained or competitive; (ii) input data, expression data or thresholded gene p-values. ORA and PathwayExpress test the competitive null hypothesis of whether the genes in the set of interest are at most as often DE as the genes not in the set, instead `SEMgsa()`, DEGraph, topologyGSA and NetGSA test the self-contained null hypothesis of no genes in the set of interest are DE. Another major difference among these methods regards the input requirements. There is a high sensitivity to the p-value cutoff because all techniques based on testing the competitive null hypothesis must identify DE genes based on a pre-specified threshold of corrected p-values. Without making any arbitrary decisions on the list of DE genes, all self-contained tests directly employ expression data.

For computational purposes, two main aspects have been addressed within the main analysis, mostly regarding DEGraph, topologyGSA and NetGSA:

- *Common covariance matrix*: DEGraph, TopologyGSA and NetGSA are multivariate hypothesis testing-based approaches. The vectors of gene expression levels in each (sub)pathway are assumed to be random vectors with multivariate normal distributions, $N_p(\mu_1, \Sigma_1)$ and $N_p(\mu_0, \Sigma_0)$ where the covariance matrix stores the network topology information. If the two distributions of the gene expression vectors corresponding to the two phenotypes differ significantly from one another, the network is thought to be strongly altered when comparing the two phenotypes. A multivariate hypothesis test is used to determine significance. The key distinctions between these three analytic approaches are the specification of the null hypothesis for statistical tests and the

procedures for calculating the parameters of distributions.
DEGraph was developed to perform a two-sample test of means while taking the topology of the genes into account. It considers a special case where both covariance matrices are expected to be equal, $\Sigma_1 = \Sigma_0$ and tests the null $\mu_1 = \mu_0$. DEGraph uses a modified multivariate Hotelling T^2-test hypothesis to identify significant (sub)pathways. Two groups are compared in terms of the first k components of the graph-Fourier basis (or in the original space after filtering out k high-frequency components). In our analysis, the largest component is used as a representation of the whole pathway.
TopologyGSA begins by transforming the pathway network graph into a direted acyclic graph (DAG) and then to its "moral" graph by connecting all parent nodes of a vertex and removing the edge directionality. The moral graph is then decomposed into cliques (i.e. subsets of nodes in the graph for which each pair is connected by an edge). A set of two hypothesis tests is applied to compute the statistical significance of the impact on a given graph. The first test determines if the inverses of the covariance matrices are equal. The second hypothesis test examines the equality of the distributions' means. To reduce the computational burden of this method, we consider a special case like DEGraph by assuming that the covariance matrices are expected to be equal and we employed the hypothesis test only for the mean of the distributions. When the covariance matrices are equal, the test of differential expression for the means is performed through a multivariate analysis of variance (MANOVA), equivalent to Hotelling's T^2-test.
With NetGSA, the K networks may differ, and K take into account a linear mixed effects model for each condition. The underlying biological network is encoded in the 0-1 adjacency matrix, A_k which determines the influence matrix Λ_k under each condition. The latter matrix describes the impact of each gene on all the other genes in the network and is calculated from the adjacency matrix, $\Lambda_k = (I - A_k)^{-1}$. Here we defined $k = 2$ and, to speed up computations, we assumed that the network is shared between the two conditions, allowing the computation of only one common adjacency matrix for case and controls.

- *Hotelling's T^2-test*: this test represent a a natural generalisation of the t-test for testing the difference between multivariate means of two populations. T^2 is equivalent to Mahalanobis distance: $D^2 = (\bar{y}_1 - \bar{y}_0)^T \hat{\Sigma}^{-1} (\bar{y}_1 - \bar{y}_0)$, where $\hat{\Sigma}$ is an estimation of the common covariance matrix. It is known to be consistently most effective against global mean-shift alternatives for multivariate normal distributions, but it may exhibit poor behavior in high dimensions. The T^2 test has very poor performance when the number of genes, p is close to number of samples, n; and is ill-defined when p equals or exceeds n. This is because the test statistic relies on the inverse of the estimated covariance matrix, which does not exist when p $\geq$ n and has large variation when p is close to n . We proposed to regularize the sample covariance matrix in order to to stabilize its inverse and we used the decorrelated mean difference, as test statistic: $D = (\bar{y}_1 - \bar{y}_0)^T \hat{\Sigma}^{-\frac{1}{2}} u / \sqrt{p}$, where $u = (1, 1, ..., 1)$. D are very close to D^2 (Ackermann and Strimmer, 2008), but the former is computationally more efficient, especially if randomization procedure of the null distribution is used. In this way, we solved issues regarding the computation of T^2 test for both topologyGSA and DEGraph. In addition, permutation-based p-values for testing null hypothesis are computed by randomization of group labels, as performed in `SEMgsa()` function, allowing more accurate p-value estimation with no need

of a large number of permutations.

TABLE 2.3: Overview of tested pathway enrichment methods.

Method	Null hypothesis	Gene p-value tresholding	Expression data	R/Bioconductor
SEMgsa	Self-contained	No	Yes	SEMgraph 1.1.1
DEGraph	Self-contained	No	Yes	DEGraph 1.46.0
topologyGSA	Self-contained	No	Yes	topologyGSA 1.4.7
NetGSA	Self-contained	No	Yes	netgsa 4.0.3
PathwayExpress	Competitive	Optional	No	ROntoTools 2.23.0
ORA	Competitive	Yes	No	EnrichmentBrowser 2.25.3

Evaluation measures

In the benchmark data analysis, all methods were evaluated according to (i) sensitivity and (ii) prioritization. The sensitivity refers to the ability of producing small p-values for the target pathway and prioritization refers to the ability of ranking close to the top the gene sets that are indeed relevant to a given condition.
Performance of GSA methods within each simulation run has been evaluated looking at (i) type I error and (ii) power. When a true null hypothesis is rejected, a type I error, also known as a false positive, occurs, whereas the power assesses the probability of a test successfully rejecting the null when the alternative hypothesis is true. Power comparisons are only useful if the tests have appropriate type I error control. Pathways associated with COVID-19 disease (q_1) were used to evaluate power, whereas those with a number of dysregulated genes ≤ 1 (q_0) were utilized to evaluate type-I error (Hellstern et al., 2021).

The type I errors and powers were estimated as the fraction of null hypotheses rejected across 100 simulated replications.

2.4 Results

2.4.1 Benchmark results

Significance of target pathway was detected if the adjusted p-value (after Bonferroni correction for multiple tests) didn't exceed the level of 0.05. COVID-19 pathway was identified only by `SEMgsa()` together with topologyGSA and NetGSA. The lowest p-value was reported by `SEMgsa()` together with topologyGSA (see Table 2.4). DEGraph, ORA and PathwayExpress seems not to be sensitive to mean changes between conditions in Coronavirus data. Looking at the results for Frontotemporal Dementia, the only significant p-value (in terms of median p-values of related pathways) is reported by `SEMgsa()`, confirming the high sensitivity of our method.

Table 2.4 presents also the relative ranking of target pathway reported by the different methods. Gene sets having the same p-value receive the same rank. The

TABLE 2.4: Benchmark results on Coronavirus disease (COVID-19) and Frontotemporal Dementia (FTD).

		Disease	
Metrics	Method	Coronavirus disease (COVID-19)	Frontotemporal Dementia (FTD)*
Sensitivity	SEMgsa()	$<$ **0.001**	$<$ **0.001**
	DEGraph	0.771	0.653
	NetGSA	0.011	0.563
	ORA	0.709	0.375
	PathwayExpress	0.444	0.981
	topologyGSA	$<$ 0.001	0.740
Prioritization	SEMgsa()	**10**	**39**
	DEGraph	90	39
	NetGSA	63	44
	ORA	83	56
	PathwayExpress	32	100
	topologyGSA	13	58

*Note: Since the term Frontotemporal lobar degeneration (an alias for FTD; KEGG ID: H00078) is associated to 6 KEGG pathways, sensitivity and prioritisation metrics have been aggregated by taking the median.

gene sets with the lowest p-value are ranked first, and so forth. Relative rankings are computed by dividing the absolute rank by the number of unique p-value categories and multiplied by 100 (i.e., percentile rank). Among all methods, `SEMgsa()` perform the best with a 10th position for the COVID-19 pathway and a median ranking of 39 for FTD. Same ranking for FTD is reported by DEGraph but with a p-value larger than threshold (0.05). Similar performance is reported by topologyGSA for Coronavirus data, with a ranking of 13 but with poor ranking for FTD (58). Despite the good sensitivity, NETgsa shows poor prioritization results.

ORA method has poorer performance in terms of both sensitivity and prioritization because this type of approach only works when the magnitude of mean changes between conditions is large.

2.4.2 Simulation results

We summarize the relative performance of all methods based on false positive rate and power results on the subset of pathways associated to the target disease (see Data simulations). We focused on those metrics under the betweenness, community and neighborhood dysregulation design. Metrics were evaluated on 100 simulation replications and were summarized grouping results by mean signal and dysregulation design with the aim of grasping differences in the behavior of GSA methods under different experimental conditions. Error plots (Figure 2.2-2.5) show the mean of those aforementioned metrics grouped by either mean signal or topology design together with their standard deviations across simulations. `SEMgsa()` is highlighted as red, compared to the others colored blue.

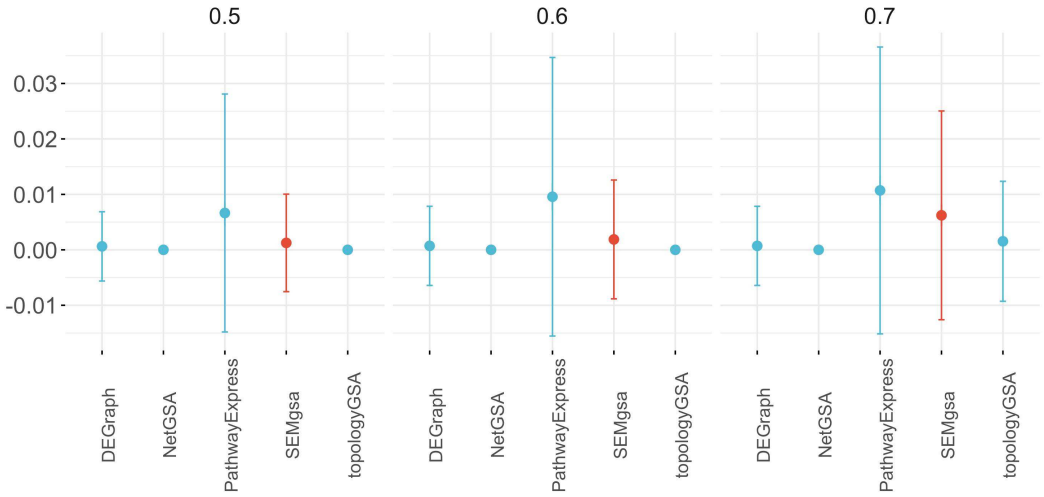

FIGURE 2.2: Average type I error on the 10 KEGG pathways grouped by method and mean signal on simulated data. Average type I error together with standard deviation across simulations is displayed for each method. Lower type I error indicates better performance. At the 0.05 significance level, all methods control the type I error rate across the 10 pathways under different level of mean signal.

Figure 2.2 shows that, at the 0.05 significance level, all methods control the type I error rate across the q_0 pathways, selected among the $K - q_1$ pathways with ≤ 1 dysregulated genes under different level of mean signal. This procedure results for betweenness topology design in $q_0 = 16$, for community in $10 \leq q_0 \leq 30$ (note that we have a range of numbers given the random sampling of genes from the community with the highest proportion of members) and neighbourhood in $q_0 = 14$. Type I error is defined as the average proportion of simulations where the method falsely rejects the null hypothesis of no enrichment. It's worth noticing that PathwayExpress' type I error rates show a wider distribution, followed by `SEMgsa()` which enlarges its error bands in correspondence of the highest mean signal. However, the error rates of the latter methods are near 0 and below the nominal threshold of 0.05. All other techniques appear to have conservative type I error rates. Type I error results have been further investigated for `SEMgsa()`, showing the ability of the proposed method to capture the signal also at lower values (smaller than 0.5). As a result, given the high sensitivity of `SEMgsa()`, the higher the value of the mean signal, the higher the rate of false positives. Same results seems not to be shared by the other methods, showing the need of higher differential expression (higher than 0.5) to achieve acceptable performance.

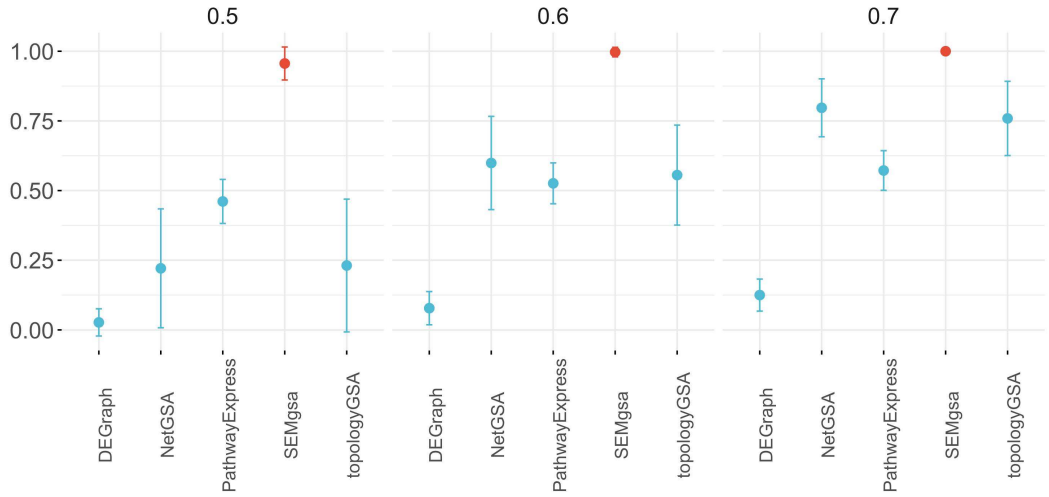

FIGURE 2.3: Average statistical power on the 10 KEGG pathways grouped by method and mean signal on simulated data. Average power together with standard deviation across simulations is displayed for each method. Higher power indicates better performance. SEMgsa stands out among all with 90%-100% power across simulation. NetGSA and topologyGSA get close to SEMgsa with about 75% statistical power only with differential mean level of 0.7.

Statistical power of different methods has been investigated in terms of average proportion of simulations where the method correctly rejects the null hypothesis of no enrichment. The higher the power, the better. Figure 2.3 shows that `SEMgsa()` stands out among all with 90%-100% power across simulations. PathwayExpress reaches a position around 50% statistical power for all the level of mean signal. The same could be stated summarizing the results by dysregulation design, with about 75% statistical power only under the betweenness design (Figure 2.5). As stated previously, the higher the mean signal, the higher the ability of the methods to correctly reject the null. NetGSA and topologyGSA get close to `SEMgsa()` with about 75% statistical power only with differential mean level of 0.7 (Figure 2.3). Slightly better results are reported from both methods with respect to the betweenness design (see Figure 2.5). DEGraph instead, is placed at the bottom of the graph for most of the comparisons (statistical power near 0).
Among the methods compared, `SEMgsa()` has the best overall performance.

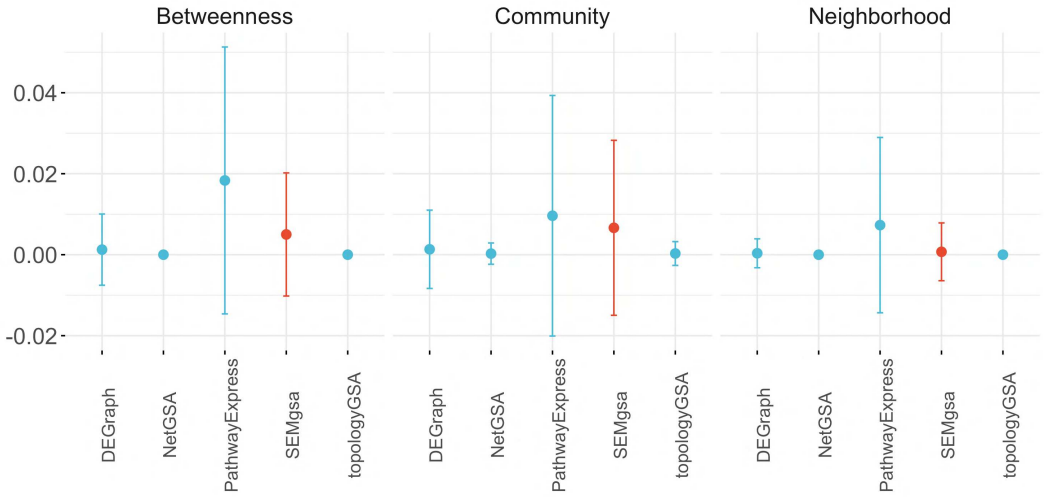

FIGURE 2.4: Average type I error on the 10 KEGG pathways grouped by method and topology dysregulation design on simulated data. Average type I error together with standard deviation across simulations is displayed for each method. Lower type I error indicates better performance. At the 0.05 significance level, all methods control the type I error rate across the 10 pathways under different topology dysregulation designs.

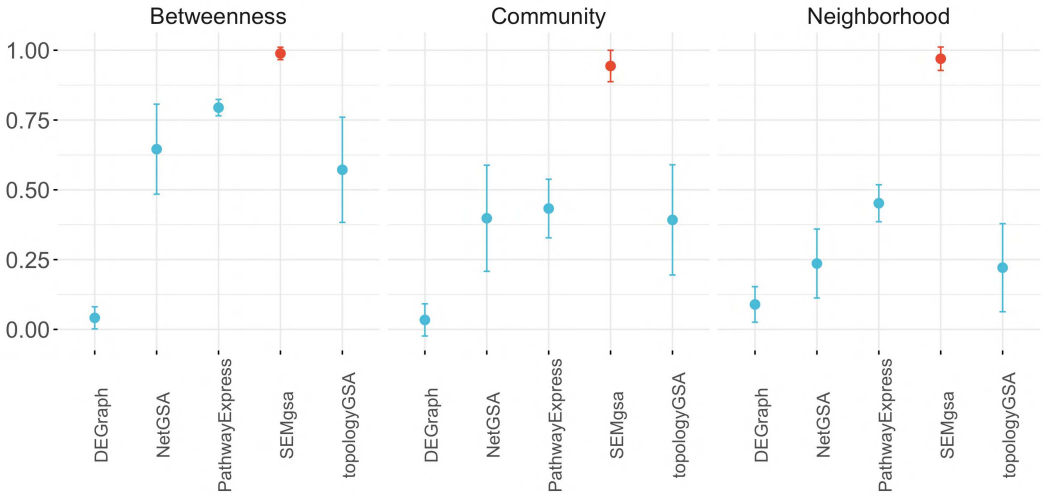

FIGURE 2.5: Average statistical power on the 10 KEGG pathways grouped by method and topology dysregulation design on simulated data. Average power together with standard deviation across simulations is displayed for each method. Higher power indicates better performance. SEMgsa stands out among all with 90%-100% power across simulation. PathwayExpress, NetGSA and topologyGSA perform slightly better under the betweenneess dysregulation design, reaching about 60%-70% statistical power.

2.5 Discussion

Topology-based approaches exhibit greater statistical power in finding pathway enrichment, according to earlier studies (Varadan et al., 2012; Jaakkola and Elo, 2015).

However, several limitations may affect user experience in terms of computational efficiency.

`SEMgsa()` represent a topological based and self-contained hypotesis method, in line with NetGSA, DEGraph and topologyGSA. Three main points make `SEMgsa()` more valuable for users than existing GSA methods:

- *Exploiting pathway information*. Existing methods have specific input requirements about pathway topology. TopologyGSA, for instance, only works for pathways whose topology is a DAG and whose size is less than the required number of samples in the two conditions/groups. If a pathway has multiple connected components, DEGraph will check to see if the means vary for each connected subgraph. Without taking into account singletons, NetGSA fits a linear mixed model for each condition. Overcoming this limitations, `SEMgsa()` accepts as input directed and/or undirected networks that define pathway interconnectedness. Inside `SEMgsa()` workflow, the function `SEMrun()` maps the expression data onto the input graph corresponding to each pathway and converts it into a SEM. Node-level perturbation is evaluated according to the specified binary group variable (i.e. case/control) by fitting a "common" model to evaluate group effects on graph nodes. In this way, adding group-nodes and group-genes edges (see Figure 2.1), the pathway with several components and singleton genes becomes a connected graph, allowing to exploit all available pathway information.

- *Higher sensitivity to pathway perturbation*: Existing methods, like ORA, save as output of GSA a list of DEGs for each pathway, recovered from the input named vector containing log2 fold-changes of the differentially expressed genes. The latter is obtained from the differential expression analysis done on the gene expression data, where the genes with an adjusted (for multiple comparisons) p-value smaller than a pre-specified cutoff are considered as DEGs. The adjustment process is highly dependent on the number of tests performed, for example Bonferroni adjusted p-values are calculated by multiplying the original p-values by the number of tests performed. `SEMgsa()` fits a SEM model for each tested pathway. For source and singleton genes, marginal tests of traditional DEGs analysis are applied while for connections and sink genes conditional testing, given the parents are used. When there is a direct group effect, conditioning increases power and reduces gene variability. As a result, the topological structure of the pathway improves the precision of the inference. Furthermore, the significance of node-level perturbation is computed within each pathway and the adjustment procedure for p-values is less stringent given the smaller number of tests. This choice allows to obtain higher sensitivity to pathway perturbation. Thus, from the output of `SEMgsa()` we can extract a seed list of DEGS for each pathway that can be useful to discover novel disease-associated interactions. A further step could be to extract a Steiner tree, mapping the DEGs on the union of the KEGG pathways and finding a connected subgraph such that the additional nodes (Steiner or connector nodes) connecting seed nodes (terminal nodes) minimize the sum of the weight of every edge in the subgraph (i.e., maximizing edge perturbation between disease nodes). Then, fitting the filtered active (perturbated disease) module with `SEMrun()`, we can obtain a perturbed backbone regarding the disease of interest, where important connectors or clusters of genes can be identified (Palluzzi and Grassi, 2021; Pepe and Grassi, 2014).

TABLE 2.5: Overall pathway perturbation of KEGG pathways related to Coronavirus disease (COVID-19) and Frontotemporal Dementia (FTD).

Disease	KEGG pathway	pert
Coronavirus disease - COVID-19	Coronavirus disease - COVID-19	up act
Frontotemporal Dementia (FTD)	Protein processing in endoplasmic reticulum	up act
	Endocytosis	NA
	Neurotrophin signaling pathway	up act
	Wnt signaling pathway	NA
	MAPK signaling pathway	up act
	Notch signaling pathway	down act

- *Index for overall pathway perturbation*: Among the existing methods, only SPIA reports the direction in which the pathway is perturbed (activated or inhibited), exploiting a posteriori pathway information obtained from hypothesis testing. Like SPIA, `SEMgsa()` outputs a column summarising overall pathway perturbation, but combining also a priori information obtained from biological databases (up- or down- regulation of genes derived from KEGG) to a posteriori information obtained from the analysis of gene expression data (node perturbation obtained from `SEMgsa()`). The combination between these flow of information allows to better define the direction of gene perturbation. Table 2.5 provides the results for perturbation index with respect to benchmark data. COVID-19 pathway is associated to an increase in activation in cases with respect to control group ("up act"); same result can be stated for two out of six pathways (Neurotrophin signaling pathway and MAPK signaling pathway) regarding Frontotemporal dementia. Notch signaling pathway is associated to a decrease in aactivation in cases with respect to control group ("down act"). Note that the NA are reported for the networks without +1 or -1 edge weights in the adjacency matrix, resulting into no calculation for the combinatorial measure.

2.6 Conclusions

We have shown that `SEMgsa()` is easily accessible to common users and provides robust results under several experimental conditions. It obtains external pathway information solving the problem common to many topology-based methods but offering better statistical power and prioritization results, while also controlling for type I error.

We believe that `SEMgsa()` can be a valuable tool for practitioners, also when undertaking complex pathway enrichment analysis.

Chapter 3

SEMtree( )

3.1 Background

The biological function on the molecular level emerges from the complex interaction of biological entities of a cell. Specifically, different types of Omics-data can interact in many various ways with each other in dependence on the tissue type and the environmental condition of an organism. The interactions among biological molecules can be broadly categorized into three types of networks: metabolic networks, transcriptional regulatory networks and protein interaction networks (Vidal, Cusick, and Barabasi, 2011). These networks need to be inferred from the experimental observations generated by different high-throughput platforms, including Next-Generation Sequencing (NGS), proteomics and microarrays.
The goal is to identify active modules, i.e., subnetworks enriched in interactions and in nodes of interest (showing condition-specific changes). Then, these active modules facilitate the investigation of the perturbed cellular responses, as functional modules are the building blocks of the cellular processes and pathways (Mitra K, 2013). To identify these subnetworks, numerous methods have been suggested. These methods can typically be divided into two categories: responsive subnetwork identification and subnetwork extraction started by seed genes (or nodes).
For the first category, a number of algorithms and tools are created by combining genome-wide measurements of signals with pre-established networks (Ideker et al., 2002; Ma et al., 2011; Beisser et al., 2010). These techniques often include a score function quantifying the alternation of a given sub-network between different conditions as well as a search strategy that aims to identify the sub-networks in the reference network that have the highest scores. Different scoring functions have imposed scores on network nodes or edges or both. Besides, high-scoring nodes were prioritized as 'disease genes' useful for generating new hypothesis (Gu et al., 2010; Zheng and Zhao, 2012).
In the second category, algorithms typically start with a set of genes as seeds to expand and extract a subnetwork from the reference network. The resultant subnetworks, which reflect the paths in which the seeds are involved, suggest the functional relationships of the seed genes and further predict additional genes that may play important roles in functional cooperation (Kleinberg and Tardos, 2006).
This class of methods has two main components: a scoring function quantifying the alternation of a given sub-network between different conditions, and a search algorithm to extract the highest scoring sub-networks. Different scoring functions have imposed scores on network nodes or edges or both. Besides, high-scoring nodes were prioritized as 'seed genes' for searching (Gu et al., 2010; Zheng and Zhao, 2012). Due to the non-deterministic polynomial-time hard (NP-hard) nature of the problem of finding the maximal-scoring connected subgraph, it can only be approached by

heuristic or approximate methods. Most approaches rely on greedy searches, simulated annealing, and genetic algorithms (see Mitra K, 2013 and Nguyen et al., 2019 for general surveys of the active module identification methods). Because of the diversity of scoring functions and searching algorithms, it is impossible to obtain identical or similar subnetworks given the same input expression profiles and PPI network.
The main contribution of this chapter is the development of a self-contained tree-based structure learning algorithm developed into the framework of Structural Equation Models (SEM), called `SEMtree()` and included in the R package **SEMgraph** (Grassi, Palluzzi, and Tarantino, 2022). To investigate the utility of our approach, we performed two sets of experiments on both observed and simulated expression data using Human Protein Reference Database (HRPD) interaction network, including 5007 proteins and 42704 interactions from KEGG database (Kanehisa and Goto, 2000). We tested the ability of our framework to evaluate plausible regulatory subnetworks of five popular subnetwork detection methods, i.e. BioNet (Beisser et al., 2010), COSINE (Ma et al., 2011), pathfindeR (Ulgen, Ozisik, and Sezerman, 2019), WalktrapGM (Petrochilos et al., 2013) and our fast Steiner Tree function to provide a meaningful comparison in terms of performance.
Regarding real data analysis, the highest-scoring subnetwork from each method has been recovered as undirected network and supplied to Causal Additive Trees (CAT) (Jakobsen et al., 2022) algorithm of `SEMtree()` to be converted in a directed tree. The latter conversion allows to compare the methods in terms of directed active subnetworks.
The remainder of this chapter is organized as follows. Firstly, we describe the `SEMtree()` features both in terms of inference procedure and user interface. Then, we outline the experimental setup constructed to evaluate subnetwork detection methods, including the real data application and simulation design. In the end, we provide the results together with the overall discussion.

3.2 Method and implementation

`SEMtree()` function includes both graph and data-driven algorithms to recover trees, $T = (V, E)$ with p nodes (V) and $p - 1$ edges (E). A tree is an undirected (or directed) graph without cycles with a *unique* path between any two nodes, where a *path* between two nodes $(j, k) \in V$ can be viewed as a sequence of edges that may have either the same or different direction with respect to neighbouring connections. The graph method refers to the Steiner Tree (ST), a tree from an undirected graph that connects "seed" (e.g., disease) with additional nodes in the "most compact" way possible based on a very fast solution provided by the Kou's algorithm (Kou, Markowsky, and Berman, 1981). The data-driven methods propose fast and scalable procedures based on the Chu-Liu–Edmonds' algorithm (CLE) (Chow and Liu, 1968) to recover a tree from a full graph. The first method, called Causal Additive Trees (CAT) (Jakobsen et al., 2022), uses pairwise mutual weights as input for the CLE algorithm to recover a directed tree (*arborescence*). The second one (Lou, Hu, and Li, 2021) applies the CLE algorithm for skeleton recovery and extends the skeleton to a *polytree* represented by a Completed Partially Directed Acyclic Graph (CPDAG). Finally, applying the Prim's algorithm (Prim, 1957), the Minimum Spanning Tree (MST) of a connected undirected graph (or a data-driven undirected full graph) can be identified. Here, we review the novel CAT method used for the conversion of undirected graphs in directed ones.

3.2.1 Causal tree recovery

A fundamental problem is learning the causal structure of a random vector $Y = (Y_1, Y_2, ..., Y_p)$ without the graph knowledge. Generally, a Directed Acyclic Graph (DAG), $G = (V, E)$ is used to understand whether Y_k causes Y_j (or vice versa), where V is the set of nodes (i.e., variables) and E is the set of edges (i.e., connections), and loops are not allowed. Causality is evaluated over *directed paths* between two nodes having causal relevance, i.e., a sequence of edges with the same direction, where node Y_k is an *ancestor* of Y_j, and Y_j is a *descendant* of Y_k. If Y_k and Y_j have a direct link ($Y_k \rightarrow Y_j$), Y_k is the *parent* of the *child* Y_j. A DAG can also be represented as a Structural Equation Model (SEM), with no confounding unobserved variables, as follows:

$$Y_j = \sum_{k \in \text{pa}(j)} \beta_{jk} Y_k + U_j, \quad \text{for all } j \in V \tag{3.1}$$

where Y_j and U_j are an observed variable and an unobserved error term, respectively; $pa(j)$ is the parent set of Y_j and β_{jk} is the regression coefficient, i.e. the weight of the direct link ($Y_k \rightarrow Y_j$). DAG models assume independent errors (no confounding), $cov(U_j; U_K) = 0$, and unequal error variances, $\sigma_j = var(U_j)$ with a Gaussian (Normal) distribution, $U_j \sim N(0, \sigma_j)$ for all $j \in V$.
For high dimensional data, recently Jakobsen et al., 2022 suggest models of reduced complexity (i.e., directed trees) as causal graphs. Their approach is known as causal additive trees (CAT). A directed tree is a connected DAG in which all nodes have a unique parent, except the *root node* (r) with none parent. The node, r is the unique node with a directed path to any other nodes in the tree. In graph theory, a directed tree is also called an *arborescence*, a *directed rooted tree*, and a *rooted out-tree*, and is a sub-class of *polytree*, that allows multiple root nodes, and nodes with multiple parents. CAT is also a SEM defined with bivariate nonlinear structural equations:

$$Y_j = f_j(Y_{pa(j)}) + U_j, \quad \text{for all } j \in V \tag{3.2}$$

where $f_j(.)$ is a non linear function of any form between the child Y_j and the unique parent $Y_k = Y_{pa(j)}$, i.e., ($Y_k \rightarrow Y_j$), and $f_j(.) = Y_k^3$, or $f_j(.) = \sin(Y_k)$, or $f_j(.) = Y_k + Y_k^2 + Y_k^3$, etc. While, the additive U_j term is assumed with a Gaussian distribution as in linear SEM.
Generally, the causal structure is not identifiable from the observational data. Common "data-driven" structure learning methods (Heinze-Deml, Maathuis, and Meinshausen, 2018) use different assumptions to ensure identifiability of the causal DAG or a list of all the equivalent DAGs (i.e., a Markov equivalence class) embedded in a CPDAG. The authors Jakobsen et al., 2022 prove that exact identification, and not just an equivalent class, is possible for systems of lesser complexity. CAT procedure consistently recovers the causal directed tree of the non linear SEM in equation [2].
The causal graph recovery problem (see Figure 3.1) is resolved finding a minimum edge weight directed spanning tree of the fully connected graph, $G = (V, E)$ with p nodes $V = Y$ and $p(p-1)$ mutual edges $E = (Y_k \rightarrow Y_j; Y_k \leftarrow Y_j)$.

CAT uses a score-based method to recover a directed tree, $T = (V, E^*)$ minimizing a suitable score function, S over all mutual edges of the full graph, that is proportional to the Gaussian log-likelihood score function, defined by:

$$S = \min_T \sum_{\substack{(k \rightarrow j) \\ (k \leftarrow j)}} w_{jk}^G = \sum \frac{1}{2} \log \left(\frac{\sigma_r}{s_r} \right) \tag{3.3}$$

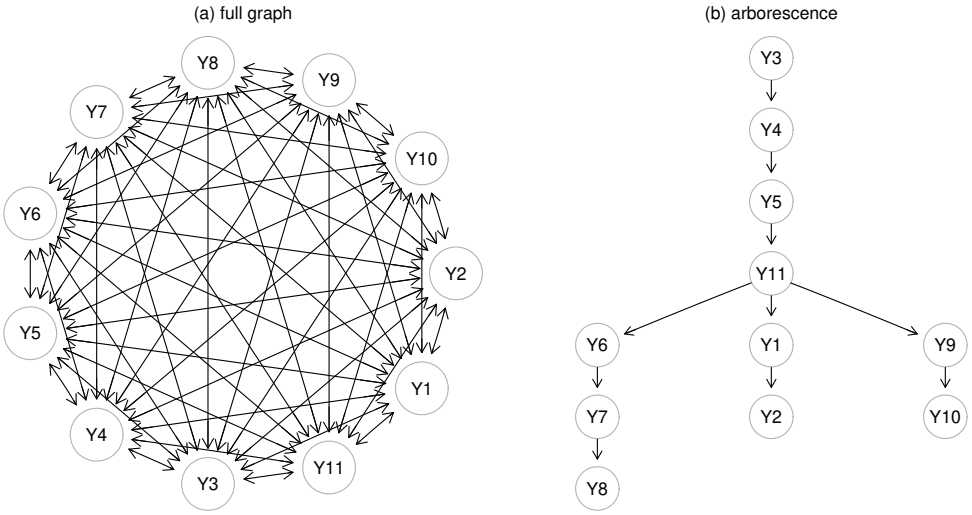

FIGURE 3.1: CAT procedure: (a) the fully connected graph with mutual edges and (b) the directed tree (an *arborescence*) minimizing edge weights with Chu-Liu–Edmonds' algorithm (CLE) algorithm, where the edge weights represents the error variance ratio and the lower the value, the better the link prediction.

where σ_r and s_r are the error variance of U_j (or U_k) and the variance of Y_j (or Y_k), respectively. The weight, w_{jk}^G represents the error variance ratio and the lower the value, the better the link prediction. It is simple to implement, computationally efficient, and only requires two steps. The mutual edge weights of the directed full graph are estimated using the residual variances of $(Y_j - f_j(Y_k))$ and $(Y_k - f_k(Y_j))$ from the (bivariate) additive regression methods in the first phase. These weights are then incorporated into the CLE algorithm to recover a directed tree with minimal edge weight in the second phase. To note, the non linearity is essential to distinguish the links $(k \to j)$ and $(k \leftarrow j)$. In linear regression with standardized variables the weights are equivalent to the negative mutual information, $-MI = \log[1 - \text{abs}(\text{cor}(Y_j; Y_k))]$, a symmetric measure that doesn't preserve directionality information.

For the implementation, `SEMtree()` function performs: (i) additive model fitting with penalized regression splines using the R-function `gam` from the R-package **mgcv**, in order to obtain estimates of $\hat{f}_{j,k}$ (resp. $\hat{f}_{k,j}$) and $\hat{\sigma}_j = \text{var}(Y_j - \hat{f}_{j,k})$ (resp. $\hat{\sigma}_k = \text{var}(Y_k - \hat{f}_{k,j})$) in the weighting phase; (ii) the R-function `edmondsOptimumBranching()` from the R-package **RBGL** for the CLE algorithm in the recovery phase.

3.2.2 User interface

The example code of the function `SEMtree()` running CAT is as follows.

```
SEMtree(graph= NULL, data, seed, type = "CAT",
        eweight = NULL, verbose = FALSE, ...)
```

The inputs are:

- a *graph* representing the network of interest as *igraph* object or *graph*=NULL, if a full graph is used;
- a gene expression data where rows correspond to subjects, and columns to graph nodes (*data*);
- a vector of user-defined seed nodes (*seed*);
- the Tree-based structure learning method, where four graph and data-driven algorithms are available (*type*" = CAT", or "CPDAG", or "ST", or "MST");
- the edge weight type for igraph object where by default the edge weights are internally computed using 1-abs(cor), otherwise are determined from the user-defined distances (*eweight*);
- the logical argument *verbose*, if TRUE allows the user to visualize and fitting (through `SEMrun()` function) the tree.

The output is the recovered tree represented by an *igraph* object. To read more about `SEMtree()` function, in terms of description and usage, refer to https://rdrr.io/cran/SEMgraph/man/SEMtree.html.

3.3 Experimental design

Workflow of the active-subnetwork search approach is display in Figure 3.2.

We selected four methods from literature for comprehensive assessment of subnetwork detection if: (i) the method is implemented within a well-maintained R package (or open source R code) and (ii) it represents diversity of methodology. Table 3.1 summarizes the selected method, highlighting the key characteristics and key differences between each method in terms (i) algorithm used to construct the subnetworks, (ii) input requirements (iii) node scoring, (iv) edge scoring (if any) and (v) statistical test for assessing the significance of the identified active subnetworks (if any).

3.3.1 Benchmark data

Coronavirus disease (COVID-19) RNA-seq expression data from Carapito et al., 2021 (GEO accession: GSE172114) have been used as benchmark data with 69 subjects $\times$ 14000 genes. Subjects include patients in the intensive care unit with Acute Respiratory Distress Syndrome ("critical group", n=46) defined as cases, and those in a non-critical care ward under supplemental oxygen ("non-critical group", n=23) defined as controls. The empirical Bayes technique, as implemented in the **limma** R package (Smyth, 2005), was used to fit linear models on the normalized RNA-seq data across the 46 case and 23 control samples. The gene P-values were adjusted for multiple testing using the method of Benjamini-Hochberg (Benjamini and Hochberg,

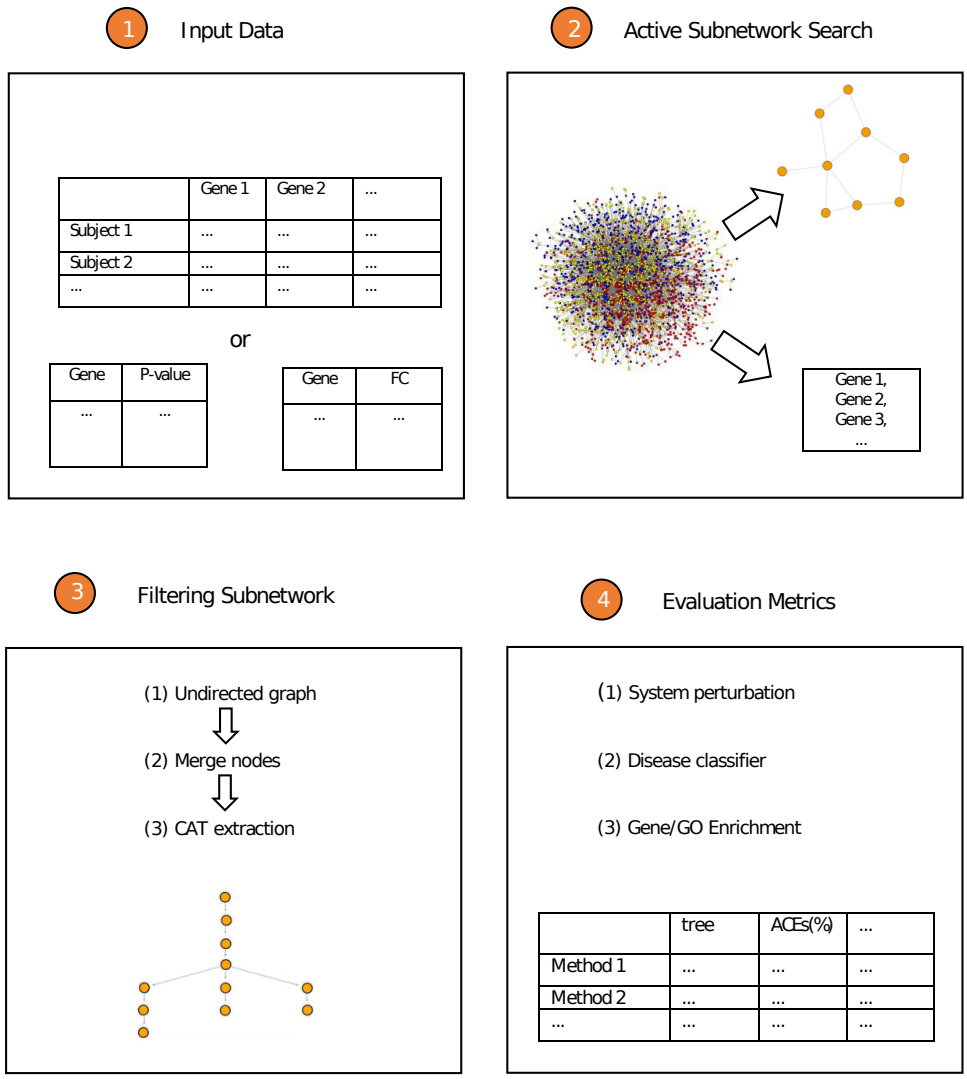

FIGURE 3.2: Overview of the active-subnetwork search approach. The required input is a gene expression data (n subjects x p genes) or a two-column table representing Gene identifiers and adjusted p-values or log-fold changes associated with differential expression data. The data are then used for active subnetwork search, starting from a KEGG PPI consisting of 3033 nodes and 19735 undirected edges. The active module is then recovered in a network format or a list of genes representing the module. Next, the identified active subnetworks are filtered according to the following steps: 1) an undirected graph is first obtained for each method; 2) group of nodes have been merged according to hierarchical clustering with prototypes; 3) an aborescence layout has been recovered with CAT algorithm. In the end, the evaluation of subnetwork detection methods has been summarized in a table in terms of: 1) system perturbation; 2) disease classifier; 3) gene/GO enrichment.

TABLE 3.1: Overview of subnetwork detection methods.

Method	Algorithm	Input Network	Input Data	Node Scoring	Edge Scoring
BioNet	Integer-Linear Programming	HPRD	p-values	p-values	-
COSINE	Genetic algorithm	HPRD	Gene expression data	F-test	ECF-test
pathfindeR	Greedy algorithm	HPRD	p-values	p-values	-
SEMtree	Fast ST algorithm (1)	HPRD	seed	seed	1-abs(cor)
	Fast ST algorithm (2)	HPRD	seed	seed	sem p-values
WalktrapGM	Random Walk algorithm (1)	HPRD	FC-values	FC-values	FC-values
	Random Walk algorithm (2)	HPRD	Gene expression data	p-values	r-to-z p-values

1995). Those P-values can be directly used as the input for subnetwork detection, be ranked to select a seed gene set, or be converted into a set of particular weights tailored to the requirement of the model.
Network information has been retrieved from the KEGG interactome object of the **SEMgraph** package as an igraph network object of 5007 nodes and 44755 edges corresponding to the union of 225 KEGG pathways extracted using the **ROntoTools** R package (Ansari et al., 2017). The latter interactome has been transformed into an *undirected* network to be suitable for fitting the already existing subnetwork detection methods. For efficiency purposes, the network has been filtered according to the genes included in the benchmark data and the largest component has been retained. This procedure results in a reference network of 3033 nodes and 19735 undirected edges.

3.3.2 Subnetwork detection methods

Table 3.1 summarizes the selected method, highlighting the key characteristics and key differences between each method in terms (i) algorithm used to construct the subnetworks, (ii) input requirements (iii) node scoring, (iv) edge scoring (if any) and (v) statistical test for assessing the significance of the identified active subnetworks (if any).
The methods analyzed here use a wide range of scoring functions to score the nodes and edges. Most of them provide a scoring function for nodes or edges, but only some of them take into account the scores of both nodes and edges. Edge-based scoring networks focus on the strength of the interaction between proteins or genes, whereas node-based scoring networks look at the relevance of one gene or protein in the context of the entire network. Node-based scoring algorithms may yield subnetworks with high scoring nodes but no significant connectivity between nodes. Edge-based scoring network may generate subnetworks with highly related genes but low network relevance. Methods that consider both node and edge scores are more likely to yield a more accurate active module. In principle, scoring functions

should be the summary statistics that capture the network perturbation, signal propagation as well as the changes between different phenotypes. Specifically:

- *Steiner tree* (ST) extracts a active subgraph from the input graph such that additional nodes (called the *Steiner* nodes) connecting "seed" nodes (called the *terminal* nodes) minimize the sum of the weight of every edge in the subgraph. ST have been tested with `SEMtree()`, that use the simple but effective heuristic approach based on the fast Kou's algorithm, as first published in 1981 by Kou, Markowsky, and Berman, 1981. We used `weightGraph()` from **SEMgraph** package to edge scoring the input graph, specifying different edge weighting methods. For comparison on benchmark data, we select the argument type="ST" proposed as default option from `SEMtree()` function, where edge weights are defined according to 1-abs(cor), and the best performing ST among `weightGraph()` options, resulting in edge weights defined by Fisher's $r - to - z$ method (Fisher, 1915). The seeds were selected with a False Discovery Rate (FDR) = 5-E06.

- *BioNet* (Dittrich et al., 2008), model the Prize-collecting Steiner tree (PCST) algorithm as a mathematical programming-based optimization that aims to find subgraphs of maximum weight using CPLEX library (a fast heuristic version in R is also implemented in the function `runFastHeinz ()`). BioNet requires multiple nominal (not adjusted) P-values derived from various sources (differential expression analysis, survival analysis, etc) and then models their combination as a Beta distribution, developing an additive scoring, where positive values are statistically significant with a FDR defined by user, and negative otherwise. Here, we use P-values derived from one source and with the threshold for FDR = 5-E06.

- *COSINE* (Ma et al., 2011)), i.e., COndition SpecIfic subNEtwok, on the other hand, uses a Genetic Algorithm (via the function `ga.bin()` in the R package **genalg**) to search for an optimal subnetwork with the highest aggregate score that jointly measures the condition-specific changes of both nodes and edges using F-statistic and Expected Conditional F-statistic (ECF-statistic), respectively. The subnetwork scoring is defined as a weighted average of nodes and edges with the parameter, λ. Thus, this method requires gene expression data as input to evaluate both the differential expression of individual genes and the differential correlation of gene pairs. Here, we use the default parameters of the genetic algorithm, and λ=0.5, i.e., equal relevance for nodes and edge weights.

- *pathfindeR* (Ulgen, Ozisik, and Sezerman, 2019) identifies active sub-networks in an unweighted reference PPI network by implementations of a greedy algorithm, a simulated annealing algorithm, or a genetic algorithm. Here, we use the greedy algorithm, a problem-solving/optimization procedure that selects locally the best option in each stage with the expectation of reaching the global optimum. The procedure start with a seed node and adds direct neighbors (d=1 by default) in each step to maximize the subnetwork score, ad so on for all seed nodes, removing a subnetwork that overlaps with a higher scoring subnetwork (at 0.5 threshold by default). Here, we use the default parameters, and the non overlapping genes list in the all extracted sub-networks is recovered.

- *WalktrapGM* (Petrochilos et al., 2013) runs a short random-walk-based community detection algorithm to identify disease modules from an edge weighted reference network (via the function `cluster_walktrap()` in the R package **igraph**). According to the node and edge weighting scheme, two different WalktrapGM algorithms have been tested. *WGM_FC* assigns gene fold change (FC) values and estimates the edge weights as a function of differential expression, taking the mean of the absolute FC-values of the two adjacent nodes of the edge (Petrochilos et al., 2013). *WGM_RWR*, a modified R function of *WalktrapGM*, assigns gene p-values to node weights and computes edge weights from Fisher's r-to-z transform for testing pairwise correlation coefficient of interacting nodes. Then, for both algorithms (WGM_FC, and WGM_RWR), module scores are used to rank high-scoring modules, comparing the module cumulative activity, i.e., the sum of node weights, against a bootstrap distribution of random differential expression values per module size. Here, the first top modules for both algorithms are recovered.

Most of the methods included in this analysis (Steiner Tree, BioNet, pathfindeR, and WalktrapGM) require the user to input a gene list (i.e., a seed list) as the significant gene set or gene P-values to serve as starting points of the algorithm. The only exceptions is COSINE that uses gene expression data and internally computes the F-test and ECF-statistic to capture node and edges changes across multiple conditions. `SEMtree()` allows the user to choose between different types of edge weights for the ST algorithm. To note, pathfindeR and WalktrapGM algorithms require node weights for ranking the sub-networks. Both followed the scoring scheme that was proposed by Ideker et al (Ideker et al., 2002). pathfindeR and WGM_FC use the unweighted sum of node z-score (i.e., the standard normal inverse of a single gene's P-value) adjusted for the size of the sub-network, and calibrated by the mean and standard deviation of a Monte Carlo simulation for each possible sub-network size. WGM_RWR applies the weighted sum of node z-score from iPINBPA (Wang, Mousavi, and Baranzini, 2015). The weights of the z-scores are obtained with a random walk with restart (RWR) method (Köhler et al., 2008) to prioritize disease-associated genes, and improve sub-network extraction.

3.3.3 Tree (CAT) extraction

The existing subnetwork detection methods (Table 3.1) differ for the class of the output in which the recovered active module is represented. Three out of five algorithms, i.e. COSINE, pathfindeR and WalktrapGM, give as output a list of genes representing the identified subnetworks, not allowing the user to visualize the full graph with the interactions between nodes. On the other side, BioNet and `SEMtree()` output the subnetwork in an undirected graph format. Therefore, we extract from the obtained gene list of COSINE, pathfindeR and WalktrapGM the undirected induced subgraphs on the reference undirected KEGG interactome.
Since in the Section Evaluation metrics a directed graph structure is required in the benchmark data analysis to evaluate the node perturbation through SEM fitting, the different type of output has been converted to a directed graph (a directed tree) by the following two steps procedure:

1. First, when all the undirected graphs representing the identified active modules have been recovered, their dimensionality has been investigated to have a a maximum number of about 200 nodes as the upper bound to retain the interpretability of the recovered modules as suggested by Petrochilos et al., 2013,

and similar to the size (232) of the KEGG "Coronavirus disease - COVID-19" pathway". Beyond this threshold, to solve this high-dimensionality problem, **SEMgraph** offers the possibility to merge groups of nodes using hierarchical clustering with prototypes from the **protoclust** R package (Minmax linkage) (Bien and Tibshirani, 2011) with `mergeNodes()` function. We therefore have a single representative data point (the prototype) for the resulting cluster for each merging of the agglomerative procedure. The `mergeNodes()` function cuts the dendrogram at height $h = 1 - \text{abs}(\rho_0)$, where ρ_0 is the Pearson's correlation coefficient, $\text{cor}(Y_j; Y_k)$. This procedure results in a merged node (and a reduced graph) in which every node in the cluster has correlation of at least ρ_0 with the prototype node. We tuned the height h to control the size of subnetworks to be approximately 200 genes.

2. Second, after merging nodes, an arborescence layout with CAT algorithm has been recovered from each method to (i) be more comparable from a structural viewpoint with a more interpretable yet visible subnetwork, (ii) to identify gene signature, i.e., significant root node, driver-gene and hub or module structure, (iii) to reduce considerably the CPU-time computation of SEM fitting.

3.3.4 Evaluation metrics

In the benchmark data analysis, the performance of the state-of-the-art approaches has been evaluated in terms of (i) system perturbation, (ii) disease classifier performance and (iii) COVID-19 gene set/GO enrichment. We also add to the seven extracted CAT modules two reference trees (after CAT conversion): (8) the KEGG "Coronavirus disease - COVID-19" pathway, and (9) the data-driven directed tree extracted from the top 200 DEGs ranking by a Random Forest variable importance procedure with the `randomForest()` function of **randomForest** R package (Breiman, 2001).

1. Evaluation of system perturbation of extracted CAT subnetworks has been evaluated via `SEMace()` and `SEMgsa()` functions of **SEMgraph**. Firstly, we compute Average Causal Effects (ACEs) between every possible source-sink node pair, using the parent adjustement set procedure, and we report (i) the number of significant paths (P $<$ 0.05 after Bonferroni correction) over the total estimated paths, and (ii) the Bonferroni combination of ACEs' p-values (P=K*min(pvalues)), where K is the total estiamate paths,the lower the value, the better the score. Then, we perform a Gene Set Analysis (GSA) on CAT modules and we report (iii) node activation and node inhibition P-values (P+ and P-, respectively) through a Bonferroni statistics ($P = 2 * min(P+; P-)$), and (ii) the number of DEGs, i.e. differential expression genes with P-values $<$ 0.05 after Benjamini-Hochberg (BH) correction.

2. Disease classifier performance was carried out by a penalized Fisher's Discriminant Analysis (pFDA) with the `PenalizedLDA()` function of **PenalizedLDA** R package Witten and Tibshirani, 2011 to identify genes in the extracted subnetworks able to discriminate between groups. Specifically, pFDA tries a discriminant projection (a discriminant variable), $a^T x = \Sigma_j a_j x_j$ in a lower dimensional space such that the ratio of between-class variance and within class variance is maximized, subject to an additional l1 (lasso)-constraint on the weights, $a's$. This constraint ensures that some discriminant weights, a_j will be estimated as

exactly zero and the corresponding variable, will not be contribute to the discriminant variable. Then, the FDA threshold $d = (a^T\overline{x}_0 + a^T\overline{x}_1)/2$ was defined to classify the patients as case if $a^T x - d > 0$ or as non-case, vice versa.
To avoid model over-performance on a specific dataset, and consequent loss of classification generality, and reproducibility, we performed a K-fold cross-validation analysis, with $K = 5$. At each iteration, K-1 partitions were merged into one and used for the learning process (training step), while the K-th left out partition (i.e., the validation set) was used to predict the outcome (i.e., the diagnostic class). The 2x2 frequency table (i.e. confusion matrix) was obtained at each iteration of the K-fold cross-validation, and the classical performance indices of the FDA classifier (sensitivity, specificity, and accuracy). Let TP be the true positives from the 2x2 confusion table, FP be the false positive, TN be the true negative, and FN be the false negative. Then, Se= TP/(TP+FN), Sp=TN/(TN+FP), and Ac=(TP+TN)/n, where n is the total sample, and the confusion table is computed both by averaging the indices of K 2x2 tables and by using the overall 2x2 table over the K iterations.

3. 3033 genes which were contained both in COVID-19 gene expression profiles and KEGG network were included in the subsequent network analysis. In addition, 245 genes related to COVID-19 disease were obtained from the collection of diseases-related genes of Feng et al., 2022. This data based comprehensively included genes collected from searches against OMIM (), KMDB/MutationView (), DisGeNET database (), and NCBI database (Tatusova et al., 2016). Non-matching genes derived from an updated version of the databases were added, resulting in 278 total genes. Among the 278 genes, 92 were included in the 3033 genes of the benchmark expression dataset. Then, we extracted the GO terms related to the 92 COVID-19 reference genes, resulting in a total of 1099 recovered GO terms.
   We perform an assessment of enrichment performance, both on benchmark and simulated data, looking at precision, recall and F1 score. To this goal, the genes (or the GO terms) are separated into two groups, Foreground Genes FG (or Foreground GO terms, FGO) and Background Genes BG (or Background GO terms). The FG (FGO) are the reference 92 COVID-19 genes (1099 GO terms), while, for simulated data, FG genes are artificially differentially expressed. Let TP and the FP the number of FG and BG present in the active modules, respectively, and the FN are the number of missing FG (i.e. the FG that were not retrieved). Then, Pre = TP/(TP+FP), Rec = TP/(TP+FN), and F1 = 2*(Prec*Rec)/(Pre+Rec) have been computed, taking the average over 100 simulation runs for simulated data.

3.3.5 Data simulations

Following the experimental setup of Ma et al., 2011, we simulated five datasets, including one "white" dataset (i.e. control) and four datasets to be compared to the control one (i.e. cases) from multivariate normal distributions. Different mean parameters (μ) and covariance matrices (with different ρ correlation coefficient) were set for each dataset, fixing the variances to 1. Each dataset consists of 500 genes and 20 samples and the condition-specific sub-network for case datasets 1, 2, 3 consisted of 50 genes while for the case dataset 4 consisted of 40 genes. Specifically:

- Control group: $\mu = \rho = 0$ for all genes (to be compared with each of the four case sets for the identification of the optimal sub-network).
- Case set 1 (both differential expression and differential correlation): Gene 1 to Gene 50 have $\mu = 0.75$, and $\rho = 0.6$ between each gene pair; the other 450 genes have $\mu = \rho = 0$.
- Case set 2 (only differential expression, no differential correlation): Gene 1 to Gene 50 have $\mu = 0.75$, and $\rho = 0$ between each gene pair; the other 450 genes have $\mu = \rho = 0$.
- Case set 3 (differential correlation and differential expression with both up and down regulation): Gene 1 to Gene 25 have $\mu = 0.75$, and $\rho = 0.6$ between each pair; Gene 26 to Gene 50 have $\mu = -0.75$ and $\rho = 0.6$ between each pair of them; $\rho = -0.6$ between any gene from 1 to 25 and any gene from 26 to 50. The other 450 genes have $\mu = 0$ and $\rho = 0$.
- Case set 4: 10 genes from each of set 2, set 3, set 4 and set 5, the other 460 genes from set 1 (mixed pattern of differential expression and differential correlation).

Given the PPI network recovered from KEGG database and the ground truth subnetwork, four gene expression data (against one control dataset) were simulated with 100 randomizations. Then we performed differential expression analysis across the 20 case and 20 control samples and we assigned to each gene an adjusted P-value representing its significance of differential expression. Gene expression data, DEGs or P-values were supplied according to the subnetwork detection method of interest. We ran 6 selected subnetwork methods 100 times for 4 case datasets. Finally, we obtained 2400 (100 randomizations $\times$ 4 case datasets $\times$ 6 methods) subnetworks. Note that, for each simulation run, the evaluation metrics (average Recall, Precision, and F1-score over 100 runs) have been computed only if an active module with more than one node has been identified.

3.4 Results

3.4.1 Benchmark results

We aim to apply `SEMtree()` on COVID-19 real data to compare its performance with existing methods and to reveal significant biological processes. The goal is to retrieve a single condition-specific sub-network composed of genes with a good system perturbation, while reporting optimal ability to discriminate between groups. In addition, the ability of each method to identify COVID-19 related genes (gene enrichment) and GO terms related to those genes (GO enrichment) has been tested.

Table 3.2 shows that the highest percentage of source-sink path perturbation and the lowest combination of path P-values (*ACEs(%)* and *PVAL(E)* respectively) is reported by ST, in line with RF_C19 and immediately followed by STr2z. pathfindeR reports the most perturbed network, with 112 DEGs (*No.DEGS*) and the lowest combination of node P-values (*PVAL(V)*), followed by BioNet, ST and STr2z. The combination of all these metrics allows to consistently identify ST as the most perturbed subnetworks among the considered ones in terms of both path and node perturbation.

In addition, Table 3.3 shows that most of the methods report high accuracy values (above 90%) in classifying patients as case or non-case, with the exception of COSINE and WGM_FC that report accuracy below 90% but still around 80%. However, according to the higher number of zero features (*no.zero*) the most parsimonious predictors (genes) are in STr2z, WGM_RW, WGM_FC and ST. BioNet reports high classification metrics but almost all the features have non-zero discriminant vector. To note, the reference modules have the poorer (KEGG_C19) and the greater (RF_C19) classification performance.

Gene and GO precision, recall and F1-score are shown in Table 3.4. ST methods show the best performance in identifying COVID-19 related genes, with the highest gene F1-score (0.12 for STr2z and 0.11 for ST) among all the considered methods. The latter methods are able to identify, respectively, 18 and 15 reference genes (see Table 3.5) over the total of 92. ST gene enrichment metrics are in line with KEGG_C19 baseline that reports a gene F1-score equal to 0.23. On the other side, pathfindeR reports the highest GO F1-score equal to 0.50, immediately followed by ST, STr2z and WGM_RW (0.44). pathfindeR is able to recover 703 reference GO terms over the total of 1099, while STr2z and ST select, respectively, 650 and 535 COVID-19 GO terms. Worst performance, both on gene and GO metrics, is reported by COSINE, with a gene F1-score of 0.05 (with a number of selected COVID-19 genes equal to 8) and a GO F1-score of 0.25 (with a number of selected COVID-19 GO terms equal to 184).

Overall, `SEMtree()` Kou's ST algorithm is able to retrieve the subnetwork of interest, with good enrichment metrics, if compared to the other methods. The module retrieved by ST together with its perturbation is reported in Figure 3.3. For tree interpretation, the `SEMtree()` recovered subnetwork can be investigated to identify significant causal paths and hub-genes with high level of graph arborescence, i.e. many edges point away from that specific node. After testing for significant ACEs ($P < 0.05$ after Bonferroni correction, see Table 3.6), a significant path consisting of 14 nodes (with only two genes not perturbed) and 13 edges (with high pairwise correlation) between source node ATG16L1 (Gene ID: 55054) and sink node CCR5 (Gene ID: 1234) has been highlighted in orange, and compared with COVID-19 literature in the legend of Figure 3.3. This perturbed route, along with others, between the virus and the host cell interaction could suggest a possible mechanism of viral pathogenesis.

In summary, trees (arborescences) are simple models, but can nevertheless provide useful biological insights and extract unrevealed knowledge-based network structures to experimentally validate new hypothesis for disease (here, COVID-19) research.

TABLE 3.2: Evaluation metrics (graph filtering and system perturbation) from the benchmark data analysis are reported in the table below. The original graph size (*graph*), the optimal height (h) to cut the minimax clustering and the direct tree (arborescence) structure (*tree*) have been firstly displayed. Then, the path perturbation of each method can be evaluated looking at the percentage of significant paths in the network together with the combination of their p-values ($ACEs(\%)$ and $PVAL(E)$ respectively). Node perturbation can be measured with the number of DEGs ($No.DEGS$) in the network and the combination of node activation and inhibition p-values ($PVAL(V)$).

			System perturbation			
method	graph	tree	ACEs(%)	PVAL(E)	No. DEGs	PVAL(V)
BioNet	(263;569)	(193;192)	19	2.70e-04	112	2.15E-08
COSINE	(241;171)	(206;205)	2	3.44e-02	57	8.71E-09
pathfindeR	(264;700)	(205;204)	0	2.86e-01	112	2.78E-11
ST	(396;395)	(192;191)	63	5.41e-06	103	4.91E-10
STr2z	(459;458)	(204;203)	22	1.55e-05	94	3.17E-13
WGM_RWR	(166;600)	(166;165)	0	4.17e-01	66	3.75E-10
WGM_FC	(155;560)	(155;154)	4	9.64e-02	49	4.77E-08
KEGG_C19	(183;113)	(183;182)	0	3.17e-01	48	1.64E-10
RF_C19	(200;199)	(200;199)	43	6.58e-03	141	2.09E-12

TABLE 3.3: Evaluation metrics (disease classifier performance) from the benchmark data analysis are reported in the table below. The ability of each method to discriminate between groups has been tested via pFDA and it has been evaluated in terms of number of zero features (*no.zeros*, with zero penalized discriminant vector) in relation to the number of recovered genes (*no.genes*) and the classical classification metrics (Sensitivity Se, Specificity Sp, Accuracy Ac).

	Disease classifier performance				
method	no. genes	no. zeros	Se	Sp	Acc
BioNet	193	2	0.96	0.87	0.93
COSINE	206	20	0.89	0.87	0.88
pathfindeR	205	42	0.96	0.87	0.93
ST	192	46	0.96	0.87	0.9
STr2z	204	87	0.96	0.87	0.93
WGM_RW	166	59	0.93	0.83	0.90
WGM_FC	155	47	0.91	0.83	0.88
KEGG_C19	183	80	0.80	0.78	0.80
RF_C19	200	0	0.96	0.91	0.94

TABLE 3.4: Evaluation metrics (gene/GO enrichment) from the benchmark data analysis are reported in the table below. Gene and GO precision, recall and F1-score are also reported (*GenePre*, *GeneRec*, *GeneF*1, *GOPre*, *GORec*, *GOF*1).

	Gene/GO Enrichment					
method	GenePre	GeneRec	GeneF1	GOPre	GORec	GOF1
BioNet	0.07	0.14	0.09	0.51	0.4	0.45
COSINE	0.04	0.09	0.05	0.47	0.17	0.25
pathfindeR	0.07	0.16	0.1	0.41	0.64	0.50
ST	0.08	0.16	0.11	0.41	0.49	0.44
STr2z	0.09	0.2	0.12	0.35	0.59	0.44
WGM_RW	0.07	0.13	0.09	0.50	0.39	0.44
WGM_FC	0.03	0.05	0.04	0.53	0.36	0.43
KEGG_C19	0.17	0.34	0.23	0.62	0.46	0.53
RF_C19	0.03	0.05	0.03	0	0	NA

TABLE 3.5: Recovered genes/GO and selected COVID-19 related genes/GO for the nine recovered subnetworks from benchmark data analysis.

method	all_genes	C19_genes	all_G	C19_GO
BioNet	193	13	862	440
COSINE	206	8	391	184
pathfindeR	205	15	1708	703
ST	192	15	1315	535
STr2z	204	18	1855	650
WGM_RWR	166	12	872	432
WGM_FC	155	5	756	397
KEGG_C19	183	31	815	507
RF_C19	200	5	0	0

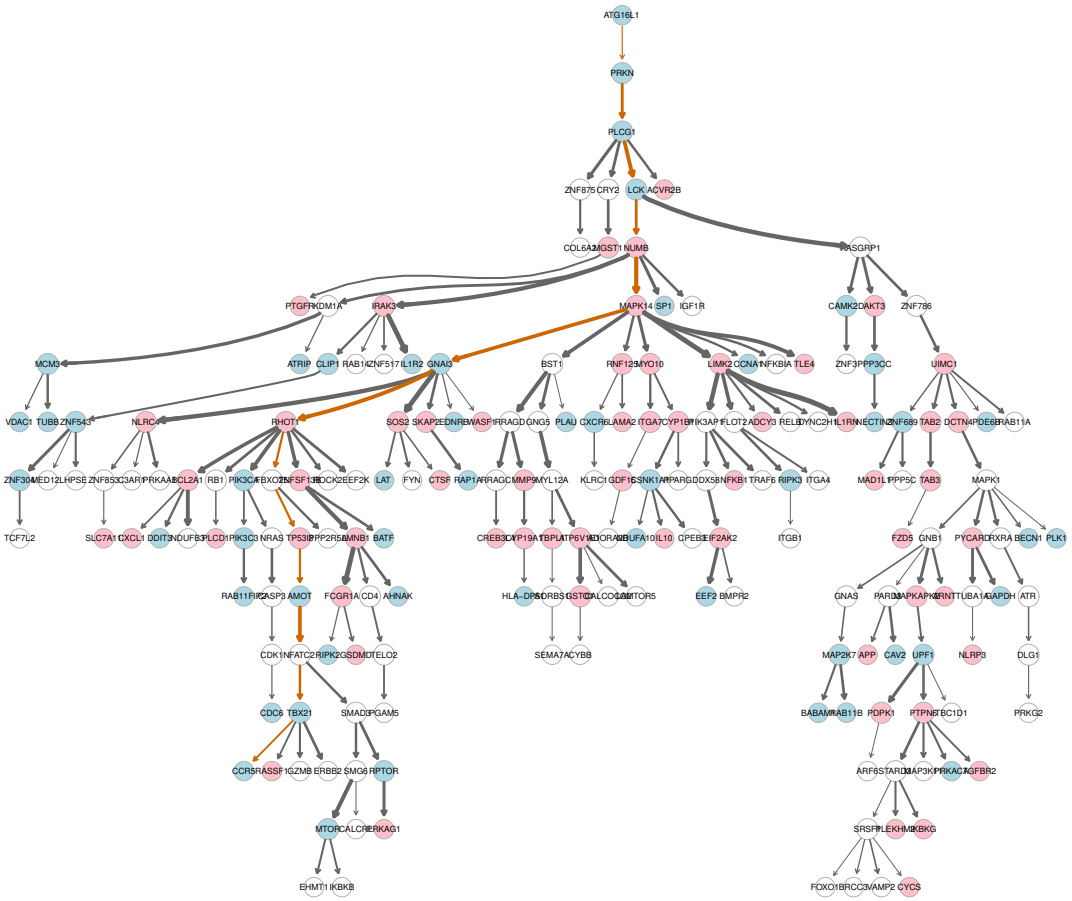

FIGURE 3.3: The graph shows the differentially regulated nodes (DRNs) as activated (pink-shaded) or inhibited (blue-shaded) variables. White nodes do not show significant variation in COVID-19, respect to healthy controls. Node size reflects node degree, the bigger the size, the higher the value of vertex degree. The width of edges shows the strength of correlation coefficient of pairs of interacting nodes. Path between source node ATG16L1 (Gene ID: 55054) and sink node CCR5 (Gene ID: 1234) has been highlighted in orange. The node ATG16L (down-regulated) gene produces a key autophagy protein that interacts with ATG5 and ATG12 to form a complex necessary for the extension of the autophagophore. Through influencing multiple components of the immune response, autophagy plays a crucial antiviral function in a variety of human illnesses (Tao et al., 2020; Ahmad, Mostowy, and Sancho-Shimizu, 2018). However, some viruses, including SARS-CoV-2, have learned how to manipulate the autophagy machinery in order to avoid their destructive destiny. On the other side, CCR5 (down-regulated) is a receptor for proinflammatory chemokines, which are implicated in host responses, particularly to viruses. Findings of (Cizmarevic et al., 2021) imply that the CCR5-32 allele may be protective against SARS-CoV-2 infection and HIV infection alike and represent a predictive biomarker for COVID-19 susceptibility, severity, and death. The activity of three hub structures along the path $MAPK14 \rightarrow GNAI3 \rightarrow RHTO1$ are altered. According to recent research reports, MAPK14 (up-regulated) stimulates regulation of inflammation that may contribute to exacerbate organ damage linked with complications of COVID-19 (Su, Rousseau, and Emad, 2021), GNAI3 (down-regulated) is a gene target predicting COVID-19—hypertension comorbidity pathway crosstalk (Barh et al., 2021), and RHTO1 (up-regulated) maps a hub protein sharing interactions with both viral baits and host baits for antiviral drug discovery (Liu et al., 2021).

TABLE 3.6: Significant average causal effects (ACEs) between source-sink pairs as obtained from SEMace function while testing for perturbation with SEMpath.

pathL	sink <- source	d_est	d_se	d_z	pvalue	d_lower	d_upper
9	DDIT3 <- ATG16L1	0.90	0.26	3.50	0	0.39	1.40
9	NDUFA10 <- ATG16L1	-0.92	0.24	-3.79	0	-1.39	-0.44
9	CPEB3 <- ATG16L1	0.88	0.24	3.67	0	0.41	1.34
8	MAD1L1 <- ATG16L1	-0.93	0.20	-4.74	0	-1.31	-0.54
9	SLC7A11 <- ATG16L1	0.79	0.22	3.66	0	0.37	1.21
6	IL1R2 <- ATG16L1	0.80	0.19	4.11	0	0.42	1.18
13	CCR5 <- ATG16L1	-0.87	0.21	-4.23	0	-1.27	-0.47
6	NFKBIA <- ATG16L1	1.02	0.22	4.65	0	0.59	1.45
6	TLE4 <- ATG16L1	1.00	0.25	4.05	0	0.51	1.48
10	EEF2 <- ATG16L1	-0.94	0.25	-3.82	0	-1.42	-0.46
10	AHNAK <- ATG16L1	-0.80	0.22	-3.68	0	-1.23	-0.37
8	PPARG <- ATG16L1	0.87	0.24	3.68	0	0.40	1.33
7	RELB <- ATG16L1	0.83	0.23	3.62	0	0.38	1.27
7	DYNC2H1 <- ATG16L1	-0.83	0.20	-4.14	0	-1.22	-0.43
7	IL1RN <- ATG16L1	0.90	0.24	3.78	0	0.43	1.36
8	FYN <- ATG16L1	-0.92	0.17	-5.44	0	-1.25	-0.59
8	CTSF <- ATG16L1	-0.69	0.17	-4.02	0	-1.03	-0.35
6	ZNF3 <- ATG16L1	-0.72	0.19	-3.74	0	-1.10	-0.34
9	PPP2R5A <- ATG16L1	0.84	0.23	3.65	0	0.39	1.29

3.4.2 Simulation results

To test the seven sub-network detection methods on the simulated data, each of the four case datasets was compared with the Control Group to identify condition specific sub-networks. The goal is to retrieve a single condition-specific sub-network composed of 50 genes while for the case dataset 4 consisted of 40 genes. Simulation results are shown in the Figure 3.5.

Compared with the other methods, `SEMtree()` ST and STr2z achieve high precision, around $90\% - 80\%$ for all the case datasets, just below the precision of BioNet. Since BioNet recovers the smallest subnetwork for all the case datasets (see Figure 3.4), its precision is the highest one compared to the other methods. `SEMtree()` recovers the smaller subnetworks immediately after BioNet and, therefore, it shares similar precision metrics with the latter. The highest network dimension is reported by WGM_RWR and WG_FC, resulting in the lower precision scores since the method selected more BG (i.e., false positives). Similar performance is reported by pathfindeR. Looking at the recall metrics (Figure 3.5), COSINE reports slightly higher results given that the higher dimensionality of its modules allows to select more genes and obtain a smaller number of false negatives. The recall values of ST and STr2z are in line with BioNet and higher than pathfindeR, WGM_RWR and WGM_FC.
Then, we calculated the F1-score to determine how good the methods are to retrieve the FG while avoiding picking BG. The F1 score for COSINE is around to 60% for all case datasets, while it is near $30\% - 40\%$ for ST, STr2z and BioNet. The latter methods are able to reach the highest F1-scores for case dataset 1 and 3, driven by the high precision values. In detail, STr2z reports F1 score around 60% for case dataset 1 and 3. For more details about simulation metrics, we refer the reader to Table 3.7.

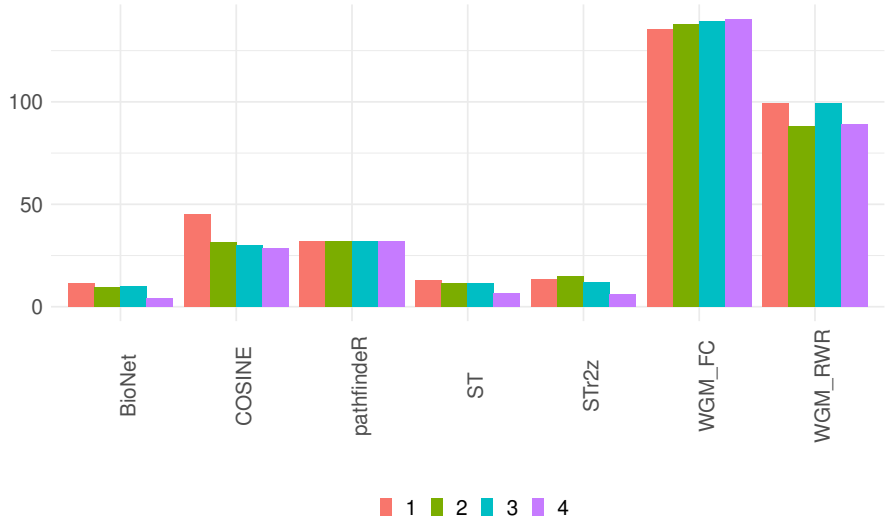

FIGURE 3.4: Average size of the recovered subnetwork for each method on simulated data. Nodes in the recovered subnetworks are coloured in yellow if they represent COVID-19 related genes while the cluster summarised by the prototype is coloured in orange if it contains at least one COVID-19 related gene (in green otherwise). The width of edges shows the strength of correlation coefficient of pairs of interacting nodes.

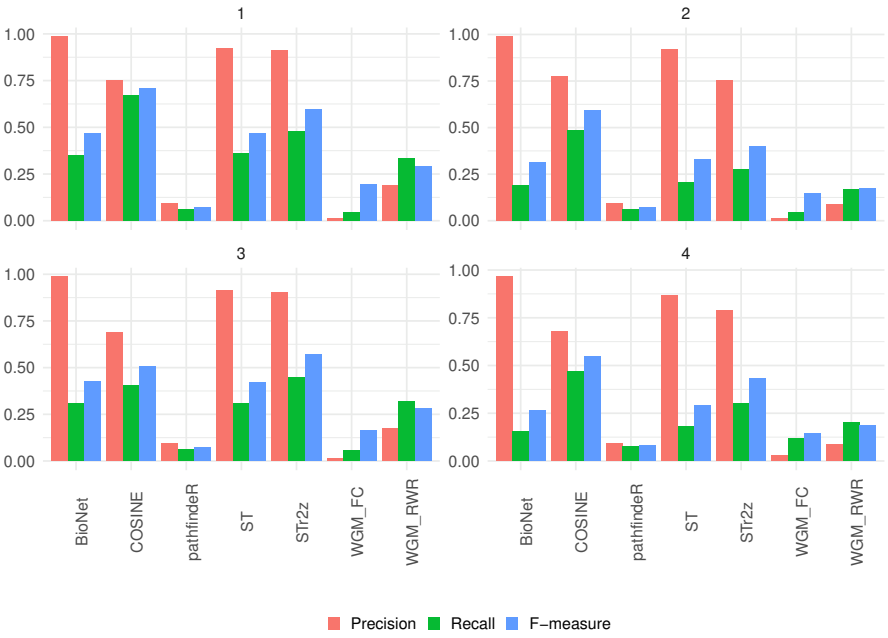

FIGURE 3.5: Precision, recall and F1-score on simulated data. The mean over 100 simulation runs of precision, recall and F1- score are displayed for each method and for each case dataset. Results show high precision score for ST and STsem, just below the precision of BioNet.

TABLE 3.7: Average simulation results (over 100 runs) for the seven subnetwork detection methods for each case dataset.

method	set	size_mean	precision_mean	recall_mean	f1_mean
BioNet	1	11.29	0.98	0.35	0.47
BioNet	2	9.30	0.99	0.19	0.31
BioNet	3	9.78	0.99	0.31	0.43
BioNet	4	3.93	0.96	0.16	0.26
COSINE	1	44.95	0.75	0.67	0.71
COSINE	2	31.61	0.78	0.49	0.60
COSINE	3	30.24	0.69	0.40	0.51
COSINE	4	28.37	0.68	0.47	0.55
pathfindeR	1	32.00	0.09	0.06	0.07
pathfindeR	2	32.00	0.09	0.06	0.07
pathfindeR	3	32.00	0.09	0.06	0.07
pathfindeR	4	32.00	0.09	0.08	0.08
ST	1	12.92	0.92	0.36	0.47
ST	2	11.24	0.92	0.21	0.33
ST	3	11.62	0.91	0.31	0.42
ST	4	6.35	0.87	0.18	0.29
STr2z	1	13.51	0.92	0.48	0.60
STr2z	2	14.77	0.76	0.28	0.40
STr2z	3	12.05	0.90	0.44	0.57
STr2z	4	5.99	0.79	0.30	0.43
WGM_RWR	1	99.13	0.19	0.33	0.29
WGM_RWR	2	87.93	0.09	0.17	0.17
WGM_RWR	3	99.44	0.18	0.32	0.28
WGM_RWR	4	89.27	0.08	0.20	0.19
WGM_FC	1	135.40	0.01	0.04	0.19
WGM_FC	2	137.97	0.01	0.05	0.15
WGM_FC	3	139.59	0.02	0.06	0.16
WGM_FC	4	140.52	0.03	0.12	0.14

3.5 Discussion

The key challenge in many disciplines is to derive networks from high-dimensional data, and numerous methods have been proposed. Despite being too simple for accurate representations of complex biological processes, trees (undirected and directed) can be used as the starting point to provide a general comprehension of the dependence structure of the network. Directed trees is an obvious choice for causal inference in high-dimensional data. Moreover, we can consider certain attributes of the chosen tree to be substitutes for related attributes of the real, underlying network. Connectivity, path length, and degree are a few attributes that can be employed in this way. All of these factors led us to design `SEMtree()`, a tree-based structure learning algorithm based on SEM. The ST approach has been chosen to be compared to the other existing methods, representative of the main algorithms dedicated to the identification of active modules: PCST (BioNet), genetic algorithm (COSINE), greedy algorithm (pathfindeR) and random walk (WalktrapGM). We have performed a comprehensive assessment of those subnetwork detection methods using COVID-19 real data and simulation data. The key conclusion in this study can be summarized as follows.
First, based on the real ans sumulation data sets, each of the approaches was asserted to be efficient in their original articles. Our results on benchmark data show high system perturbation for the ST of `SEMtree()`, while high levels of GO enrichment are reported by pathfindeR. Simulation results report high precision value for BioNet and ST, but a good F1-score around 60% for COSINE. However, worst performance on the benchmark data is reported by COSINE. As none of the methods outperformed other methods overall, users should choose an appropriate method based on the purposes of their studies.
Second, in terms of ease of use, some of the methods do not offer user-friendly interface or visualization functions for the identified subnetworks. Most of the existing subnetwork detection methods output a list of genes representing the module, not allowing the user to visualize the entire network. BioNet outputs the subnetwork in an undirected graph format.
We propose `SEMtree()` algorithm in order to overcome some limitations of existing literature. The advantages of our algorithm are summarised as follows:

1. `SEMtree()` function includes four tree-based structure learning methods implemented with graph and data-driven algorithms. Fast Kou's algorithm has been chosen for comparison with the other existing methods based on the pre-established networks (interactomes), with default edge weighting, but the users can choose one of the methods of `weightGraph()` function based on their needs.

2. `SEMtree()` utility goes beyond subnetwork detection with the graph extraction functionality. Starting from a seed list, `SEMtree()` allows the user to recover the structure of the network with data-driven algorithms. In detail, the CAT (arborescence) or the CPDAG (polytree) can be recovered from a user defined gene list or a list of differentially regulated genes, active modules or pathways.

In addition, **SEMgraph** package provides a set of utilities that have been crucial to build up the analysis of the chapter. These functions allow the user to: cluster the graph (`mergeNodes()`); apply SEM-based gene set analysis to recover the perturbation metrics (`SEMgsa()`), evaluate ACEs between source-sink pairs (`SEMace()`), evaluate SEM fitting given the recovered network and the data of interest (`SEMrun()`), and

visualize the identified module with `gplot()` function, specifying different type of layouts, and other functions illustrated in Grassi, Palluzzi, and Tarantino, 2022. As, to our knowledge, no existing method is able to fully leverage the network and data information as `SEMtree()`, allowing the user to easily recover the tree-based structure with different algorithms, extract a directed graph from a seed list and visualize the recovered module.
Given the advance in tree development, our direction for future work is also to consider the most recent proposals suggested in finance literature (Ahelegbey, Giudici, and Hadji-Misheva, 2019; Agosto, Ahelegbey, and Giudici, 2020; Giudici and Polinesi, 2021), and in machine learning (Chatterjee and Vidyasagar, 2022; Tramontano, Monod, and Drton, 2022). Specifically, the random matrix theory (Giudici and Polinesi, 2021), and the new xi-coefficient of correlation (Chatterjee and Vidyasagar, 2022) could be incorporate in `SEMtree()` as first-step filtering technique for ST and MST, and as asymmetrical edge scoring in high-dimensional ($n < p$) regime for CAT, respectively.

3.6 Conclusions

We have shown that `SEMtree()` is easily accessible to common users and provides robust results under several experimental conditions. It recovers the tree-based structure starting from the interactome and gene expression information while offering good enrichment metrics, perturbation extraction and classifier performance.
Even though trees are overly simplistic representations of biological systems, we believe that `SEMtree()` can be a valuable tool for practitioners, not only when undertaking complex subnetwork detection analysis, but also when extracting dependence (causal) structure with a direct tree (arborescence) starting from a list of genes. This simple graph can be useful as a preliminary step for visualizing observational high-dimensional data, highlighting densely connected hub nodes or neighborhoods that might be further investigated.

Chapter 4

SEMbap( )

4.1 Background

Studies of large-scale gene expression and genotype data are frequently affected by biological and technical sources of expression variation, such as batch effects, sample characteristics, and environmental influences. The ability of researchers to quantify interesting biological signals can be enhanced by recognizing and removing these potential confounders. Confounding factors might be established sources of expression variance (known covariates) or developed empirically from the expression dataset (hidden covariates).

In addition, in presence of unobserved confounding factors that impact both the predictors and the outcome, the performance of many high-dimensional regression approaches may deteriorate. Directed Acyclic graphs (DAGs) encoded in linear Structural Equation Models (SEM) assume casual sufficiency (Pearl, 2009) that requires no hidden (or latent) variables that are common causes of two or more observed variables; i.e., the covariance matrix of the unobserved terms is diagonal. This assumptions is particularly constraining, and unrealistic in most applications.

Adjusting for unobserved confounding variables is crucial, and different deconfounding techniques have been proposed for use in diverse real-world systems. Standard high-dimensional regression methods assume that the underlying coefficient vector is sparse; i.e., the response is only affected by a few predictors (Bühlmann and Geer, 2011). However, when there is confounding in a linear model, in addition to the few predictors that do in fact influence the response, there are more hidden predictors that are associated with the outcome. Some methods for relaxing the sparsity assumption represent the structure of the regression parameter as the sum of a sparse and a dense vector. The real underlying regression vector will be altered by some modest, dense perturbation if the confounding factors have an impact on a large number of predictors (Guo, Ćevid, and Bühlmann, 2022).

The approaches frequently employ some form of Principal Component Analysis (PCA) defined by Singular Value Decomposition (SVD) to estimate the confounding variables directly from the data. If dense latent factors exist, the initial main components are distinct from the others, and a two-step procedure is performed computing normalized residuals prior to downstream regression analysis. Chernozhukov, Hansen, and Liao, 2017 and Cevid, Buhlmann, and Meinshausen, 2020 suggest multiplying the response vector and the predictor matrix leftward by a well selected spectrum transformation matrix, which modifies the singular values of input data. Chernozhukov, Hansen, and Liao, 2017 propose the Lava estimator whilst Cevid, Buhlmann, and Meinshausen, 2020 suggest straightforward spectral transformation known as the "trim" transform. After that, the altered data matrix may be utilized as the input for a high-dimensional sparse regression method, of which the LASSO is a prime example.

In graphical models the goal is to estimate a concentration matrix, i.e. the inverse of the covariance matrix, of the observed variables. Chandrasekaran, Parrilo, and Willsky, 2012 address the issue of calculating the precision matrix in the presence of a few hidden confounding factors by decomposing the concentration matrix into a sparse matrix and a low-rank matrix for revealing the conditional graphical model structure in the observable variables as well as the number and impact of the hidden variables. Low Rank plus Sparse (LRpS) decomposition algorithm removes unwanted variation by using the Alternating Direction Method of Multipliers (ADMM) algorithm (Goldstein, Donoghue, and Setzer, 2014). Unlike the other methods, Chandrasekaran, Parrilo, and Willsky, 2012 decomposition regards the entire precision matrix and not only the regression coefficient. In causal structure learning, a two-step approach is suggested by Frot, Nandy, and Maathuis, 2019 which first removes the effect of the hidden variables by LRpS and then estimates the Completed Partially DAG (CPDAG) under the assumption of causal sufficiency by using the estimated sparse covariance matrix of LRpS.

Jablonski et al., 2021 introduce a novel computational approach in DAG gene expression application, known as Differential Causal Effects (DCEs), which contrasts healthy cells with cancerous cells using Average Causal Effects (ACEs); i.e. the total effect of a source-sink link of a SEM. The technique enables for the detection of specific edges in a signaling pathway that are dysregulated in cancer cells while controlling for confounding. The authors extend the linear function representing the $sink = source + parent(source)$ equation by including the first q principal components of the design matrix as additional source variables.

The main contribution of this chapter is the development of a two-stage deconfounding procedure based on Bow-free Acyclic Paths (BAP) search developed into the framework of SEM called `SEMbap()` and implemented in the R package **SEMgraph** (Grassi, Palluzzi, and Tarantino, 2022). A BAP is an acyclic graph that can have directed and bidirected edges, where the directed edges represent direct causal effects encoded by regression coefficients, and the bidirected edges represent hidden confounders encoded by pairwise covariances. The bow-freeness condition means that there cannot be both a directed and a bidirected edge on the same pair of variables. Our approach assume arbitrary latent confounding, i.e. latent variables (LVs) induce confounding dependencies among the observed variables with bow-free covariances if arbitrarily exists at least one pair of variables with covariances not equal zeros. This assumption is substantially weaker than the latent denseness, where few hidden variables have direct effect on many of the observed variables, as required from the previous cited methods.

A second objective is to provide a meaningful comparison of the state-of-the-art deconfounding methods on real and synthetic data and on a *priori* knowledge of a biological signalling pathway encoded in a DAG in terms of (i) SEM fitting, (ii) system perturbation for observed data, (iii) recovery performance metrics on simulated data.

The rest of the chapter is divided into the following sections. First, both the inference process and the user interface for `SEMbap()` features with respect to gene expression data are described. The experimental setup for assessing deconfounding techniques is then described, including real data application. Finally, we present the findings and a concluding discussion.

4.2 Method and implementation

4.2.1 SEM

A linear SEM is a set of linear equations involving the variables $Y_i = (Y_{i1}, ..., Y_{ip})^T$ and unobserved terms $U_i = (U_{i1}, ..., U_{ip})^T$:

$$Y_i = BY_i + U_i, \text{ with } \text{cov}(U_i) = \Psi \tag{4.1}$$

where $B(p, p)$ is a real matrix, and $\Psi(p, p)$ is a positive definite matrix. We consider an i.i.d. assumption across the indices $i = 1, ..., n$, and that all variables, Y_i have been standardized to mean zero and variance one. SEM has an associated graph, $G = (V, E)$ where V is the set of nodes (i.e., variables) and E is the set of edges (i.e., connections), that reflects the structure of B and Ψ. For every non-zero entry B_{jk} there is a directed edge from k to j ($k \to j$), and for every non-zero entry Ψ_{jk} there is a bidirected edge between j and k ($j \leftrightarrow k$).

The graph (or the *path diagram*), G, is also a formal tool to evaluate the hierarchical structure of a system, where we can identify *exogenous variables*, having zero explanatory variables in all structural equations, and *endogenous variables*, having at least one explanatory variable in at least one structural equation. In graph theory, exogenous variables are *source* nodes, with incoming connectivity equal to 0, whilst endogenous variables are nodes with non-zero incoming connectivity. Endogenous variables can be further divided into *connectors*, with non-zero outgoing connectivity, and *sinks*, having no outgoing connections.

We consider three special types of SEM:

- Directed Acyclic Graphs (DAGs) used in causal inference (Heinze-Deml, Maathuis, and Meinshausen, 2018), where loops are not allowed; i.e., B defines a lower (or upper) triangular weighted adjacency matrix, and all covariances are null: $\psi_{jk} = 0$ and $\Psi = diag(\psi_1, ..., \psi_p)$.
- Bow-free Acyclic Paths (BAPs), B has an acyclic structure, and bidirected connections (covariances) in Ψ are not null only if do not share any directed link: if $\psi_{jk} \neq 0$ then $\beta_{jk} = 0$; i.e., they are bow-free (Brito and Pearl, 2002).
- Latent (or Hidden) Variable Graphs (LVs), B has an acyclic structure, and U terms encode a Factor Analysis (FA) model (Bai and Li, 2012) with common latent factors and specific errors: $U_i = \Gamma F_i + E_i$, where $\Gamma(p, q)$ are the loading factors of $q < p$ LVs (i.e., new exogenous or source variables), $F_i = (F_{i1}, ..., F_{iq})^T$ and $E_i = (E_{i1}, ..., E_{ip})^T$ are idiosyncratic error terms.

The multivariate system (1) is equivalent to $Y_i = (I - B)^{-1}U_i$ that links the observed variables, Y_i only on the unobserved variables, U_i with the population covariance matrix, $\Sigma = \text{cov}(Y_i) = E(Y_i Y_i^T)$ for DAG, BAP or LV models given by:

$$\Sigma_1 = (I - B)^{-1} D_\psi (I - B)^{-T} \tag{4.2}$$

$$\Sigma_2 = (I - B)^{-1} \Psi (I - B)^{-T} \tag{4.3}$$

$$\Sigma_3 = (I - B)^{-1} (\Gamma\Gamma^T + D_e)(I - B)^{-T} \tag{4.4}$$

Considering U_{jk} an unobserved confounder between pair of observed variables, the covariance matrices $\Psi_1 = D_\psi$, $\Psi_2 = \Psi$ and $\Psi_3 = \Gamma\Gamma^T + D_e$ account for unobserved confounding, that we call *de-correlated, arbitrary* or *pervasive* confounding, respectively.

By definition, DAG is a de-correlated model, BAP model states that the LVs induce confounding dependencies between at least one pair of observed variables (Y_j, Y_k), and LV model assumes that several unobserved variables have an effect on many of the observed ones. To note, not every Y_i needs to be affected by each LV.

4.2.2 BAP deconfounding

DAGs are used for visual representations of a priori causal hypotheses making underlying relations explicit: a connection between two variables denotes causation, and variables without a clear causal relationship are left unconnected. The underling DAG may become a BAP with directed (regression weight) and bidirected (covariance) connections, adding specific missing edges hidden by unmeasured variables. Assuming arbitrary confounding and the implied population precision matrix Ψ^{-1} known, we can adjust (or de-correlate) the observed variables, Y_i in the multivariate system (1) by Mahalanobis's transformation:

$$\Psi^{-1/2}Y_i = \Psi^{-1/2}(B\,Y_i + U_i) \quad \text{s.t.} \quad Z_i = A\,Z_i + D_i \tag{4.5}$$

where $Z_i = \Psi^{-1/2}Y_i$, $A = \Psi^{-1/2}B\Psi^{1/2}$, and $D_i = \Psi^{-1/2}U_i$. The Mahalanobis's transform performs a de-correlation of the bow-free covariances: $\text{cov}(D_i) = I_p$ and Σ_4 =$\text{cov}(Z_i) = (I - A)^{-1}(I - A)^{-T}$.

For a fixed (know) Ψ^{-1}, it follows that the log-likelihood of the model with a multivariate Gaussian distribution the Mahalanobis norm, i.e. the weighted squared l2-loss, is equivalent to the SEM log-likelihood (Loh and Bühlmann, 2014):

$$\log L(B, \Psi; Y) \equiv -E(||\Psi^{-1/2}(Y - BY)||_2^2) = -||Z - AZ||_2^2 \tag{4.6}$$

Removing bow-free covariances helps to better train a DAG model, which assumes independence among error terms, and to perform Maximum Likelihood Estimate (MLE) of regression coefficients with Ordinary Least-Squares.

Z_i is an example of a whitening transformation of Y_i, that minimises the expected total squared component-wise difference between Y 's and Z's: the whitened and original variables are always positively correlated, see Kessy, Lewin, and Strimmer, 2018 for a review. This facilitates the interpretation of the whitened variables.

When the population precision matrix is unknown, the adjusted (de-correlate) variables, Z_i should be computed from data by BAP (or covariance) search. We suggest to perform d-separation (Pearl, 2009) test between all pairs of variables with missing connection in the input DAG by Shipley's basis sets (Shipley, 2000).

d-separation test. In a DAG, missing edges between nodes imply a series of independence relationships between variables (either direct or indirect). These independences are implied by the topology of the DAG and are determined through d-separation: two nodes, Y_j and Y_k, are d-separated by a set of nodes S if conditioning on all members in S blocks all confounding (or *backdoor*) paths between Y_j and Y_k (Pearl, 1998).

We need to define: (i) a path that begins with an arrow pointing to Y_k and ends with an arrow pointing to Y_j, called a confounding (or back-door) path from Y_k to Y_j ($Y_k \leftarrow ... \rightarrow Y_s$); (ii) a node $Y_s \in S$ in which two arrowheads meet Y_s ($\rightarrow Y_s \leftarrow$) called a collider; (iii) a collider along a path blocks (close) that path. However, conditioning on a collider (or any of its descendants) unblocks (open) that path; (iv) blocking a confounding path requires conditioning on any intercepted (not-collider) nodes on the path.

With these definitions out of the way, two nodes Y_k and Y_j are d-separated by S if conditioning on all members in S blocks all confounding paths between the two nodes. Shipley, 2000 find a basis set of d-separation relations (and therefore independence claims) that are implied by a DAG: $S_U = \{Y_j \perp Y_k \,|\, \text{pa}(j) \cup \text{pa}(k),\, j > k\}$, where pa() is the "parent" set; i.e., the variables with a direct effect on the response variable in a DAG.

The number of d-separation constraints in the set S_U equals the number of missing edges, corresponding to the number of degrees of freedom (df) of the model. Because the individual tests implied by the basis set, S_U are mutually independent, each one can be tested separately at a significance level of α, a using the Fisher's z-transform of the partial correlation. An edge $(j; k)$ is absent in the graph when the null hypothesis:

$$\text{H}_0 : \rho_{jk} = \text{cor}(Y_j; Y_k \,|\, \text{pa}(j) \cup \text{pa}(k)) = 0 \tag{4.7}$$

is not rejected, after multiple testing correction following a Bonferroni or False Discovery Rate (FDR) procedure.

If the graph is not very large with huge missing edges, it is possible to perform local testing of all missing edges separately. If the number of missing edges is large, only those tests where the number of conditioning variables does not exceed a given value can be performed. High-dimensional conditional independence tests can be very unreliable. We suggest to force the sparsity by: testing bow-free covariances with basis set size close to the sparsity index, $s = \sqrt{n}/log(p)$ (Janková and Geer, 2015) or, beyond 200 nodes, estimating not-zero elements of the precision matrix fixing to zero the elements of the DAG structure by gLASSO algorithm, see details in Section Gaussian Graphical Models.

CGGM. In summary, BAP search uses d-separation tests between all pairs of variables with missing connection in the input DAG. A BAP is then built by adding a bidirected edge (i.e., bow-free covariance) to the DAG when there is a significant association between them. BAP bidirected edges, encoded in the selected covariances, provides information about which part of a DAG is not supported by the observed data.

Although covariances do not indicate a specific direction of causality, they identify the local misspecification given by the structural assumptions implied by the DAG, which may substantially alter the observed data variability, and if the selected bow-free covariances, are not correctly removed, it would be difficult to analyze (fit) a causal DAG.

We propose a post-inference procedure for BAP deconfounding (or data decorrelation) based on Constrained Gaussian Graphical Model (CGGM), where the minimization of the objective function uses the solution originally described in "Elements of Statistical Learning" book (Hastie, Tibshirani, and Friedman, 2009, pg. 631), and spectral decomposition defined by:

1. fitting the constrained precision matrix, Ψ^{-1} using CGGM with null (zero) pattern corresponding to the DAG edges and null (zero) edges after the local d-separation screening, and

2. removing the latent triggers responsible for the nuisance edges by conditioning out from the observed data with the spectral decomposition of the fitted precision matrix, $\hat{\Psi}^{-1} = VLV^T$ from which we get the adjusted (de-correlate) matrix by Mahalanobis's transformation, $Z_i = (VL^{\frac{1}{2}}V^T)Y_i$.

Using Z as additional information might enhance the DAG fitting. Since the confounding correlation vanishes, we find that this de-correlation step is able to substantially increase DAG goodness-of-fit indices, applying the best trade-off between global model fitting and local statistical significance of regression coefficients.

4.2.3 PCA deconfounding

Component analysis is a common technique used in deconfounding methods to directly estimate pervasive confounding variables from the data, which appear in a number of economic and biological applications (see Price et al., 2006; Leek and Storey, 2007 for a review). In a dense confounding regime the initial principal components are different from the others, and measuring confounding proxies for hidden variables as the scores of the first q principal components, P is a possible procedure (Jablonski et al., 2021). This defines a SEM:

$$Y_i = BY_i + \Gamma P_i + U_i, \text{ with } \text{cov}(U_i) = D_\psi; \ \text{cov}(P_i) = I_q \tag{4.8}$$

Of course, we have that $\text{cov}(Y) = \Sigma_3$. The principal components are additional uncorrelated source nodes in the DAG, $G = (V = (V_p; V_y), E = (E_p; E_y))$, and the adjusted data matrix is the augmented matrix, $Z = \text{cbind}(P, Y)$.

Computational aspect uses routine software. Standard PCA learns the projections or principal components of a dataset, $Y(n, p)$ on q-dimensional orthonormal basis, $Q(p, q)$ where $q < p$. PCA issue, though non-convex, has a global minimum that can be calculated using SVD, producing the low-rank representation of the data, $\hat{Y} = PQ^T$ where $P(n, q) = YQ$ is the projected data; i.e., the principal component scores (see Section Spectral transformation).

PCA deconfounding assumes that confounding is dense, but as suggested by Jablonski et al., 2021: "not every Y needs to be affected by each confounder. However, the more Y each LV affects, the more information we have about it in the data, and thus the confounding proxies (i.e., LVs estimated by data) capture the effect of the confouders better". In addition, dense assumption ensures simply tuning of the number $\hat{q}$ of confounding proxies. In the literature, some method have been developed for selecting $\hat{q}$, see a review in Onatski, 2010. Here, we use a permutation method: it randomly permute the columns of the data matrix, and selects components if their singular values are larger than those of the permuted data (Dobriban, 2020). The permutation methodology determines a maximum number, $\hat{q}$ and visualizing the scree plot (Cattell, 1966) user can reduce the selected $\hat{q}$ by goodness-of-fit statistics of two SEM fitting (see Figure 4.12).

gLPCA. BAP or PCA consider arbitrary or pervasive patterns of confounding but we sometimes expect mixed structure. Numerous works on low-rank representation recovery have connected data manifold information in the form of a discrete graph, or its adjacency matrix, into the framework for dimensionality reduction (Jiang et al., 2013; Tao et al., 2015). The fundamental hypothesis is that high-dimensional data samples are on or near a smooth low-dimensional manifold.

Specifically, let $A(p, p)$ be the weighted symmetric matrix that encodes the adjacency information between the variables of dataset, $Y(n, p)$ and $D = diag(d_1, ..., d_p)$ be the diagonal degree matrix with $d_j = \sum_k A_{jk}$. Then, $L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$ is the definition of the normalized graph Laplacian, which describes the structure in A. The graph Laplacian, $L(p, p)$ may be used to leverage the data manifold information in A, leading to different Graph Regularized PCA models.

Graph Laplacian PCA (gLPCA) was introduced in this setting by Jiang et al., 2013, combining data cluster structures inherent in A with PCA. The model is a data representation, i.e. $\hat{Y} = PW^T$ where $P(n,q) = YW$ is the projected data on the q-dimensional orthonormal basis, $W(p,q)$ embedding the cluster structures in A. The spectral vectors, W are the eigenvectors corresponding to the first q smallest eigenvalues of the combined matrix:

$$G_\beta = (1-\beta)(I_p - Y^TY/e_1) + \beta(L/e_2 + 11^T/n) \tag{4.9}$$

where β is a tuning parameter $\in (0,1)$ weighting PCA or graph Laplacian based aspect, while e_1 and e_2 are normalized values, see Jiang et al., 2013 for details.

We use as weighted adjacency matrix the element-wise product, $A = S * C$ where S is the covariance matrix and C is the unweighted adjacency matrix (1,0) of the significant bow-free covariances selected by BAP search, to extract the projected scores, P of gLPCA. The number of components, q is determined by the number of clusters by spectral clustering through `cluster_leading_eigen()` function of **igraph** R package, and the beta parameter is fixed to $\beta = 0.75$ or $\beta = 1$, if $q > 3$. In a mixed confounding regime, we suggest to add these projected scores to the input data and its uncorrelated source nodes to DAG, as in the PCA procedure.

4.2.4 Spectral transformation

The idea of spectral transformation is to transform data, Y by applying a linear transformation, TY that only transform the singular values of the data, while keeping it singular vectors intact. It is designed to reduce the magnitude of dense confounding. In particular, when the LVs aligns with the top singular values of Y (i.e. the magnitude of LV effects are large compared with specific error terms), a SEM is designed on the transformed data (in matrix form):

$$TY = TYB^T + U, \text{ with } \text{cov}(U) = D_\psi \tag{4.10}$$

making standard DAG assumptions.

Here, we review the "trim" and "pcss" transformations. Following Cevid, Buhlmann, and Meinshausen, 2020, let $Y = PDQ^T$ be the SVD of $Y(n,p)$, where $P(n,r)$, $Q(p,r)$, $D(r,r) = diag(d_1 \geq d_2 \geq ... \geq d_r)$ are the spectral matrices with $P^TP = Q^TQ = I_r$, and $r = min(n;p)$ is the rank of Y. We use the truncated form of the SVD, which uses only non-zero singular values. The "trim" method upper-bounds each singular value to $d_m := median(d_1,...,d_r)$, and the spectral transformation is given by $T := PD_mP^T$, where D_m is diagonal with each element on the diagonal equal to $D_m(ii) := min(d_i;d_m)/d_i$. Therefore:

$$Z = TY = PD_mP^TY = (PD_mP^T)(PDQ^T) = PD_tQ^T \tag{4.11}$$

where $D_t(ii) := min(d_i;d_m)$. The spectral transformation keeps the ordering of singular values in the transformed design matrix while still shrinking the large ones, and an advantage is that we do not have to estimate the number of LVs. See Cevid, Buhlmann, and Meinshausen, 2020 for additional details.

The PCSS transformation consider the commonly used approach to extract the first $q < r$ principal component of Y. The principal components (or the projected scores), $P(n,q)$ serve as a measuring proxies for LVs; i.e., are "sufficient statistics" (pcss) for unobserved scores, if the confounders are dense or pervasive (Agrawal et al., 2022). Ideally q is equal to or slightly larger than the dimension of dense

confounders. One then adjusts data by using the partial residuals after a linear regression of Y on P:

$$Z = Y - P\hat{B} = Y - P(P^TP)^{-1}P^TY = (I_p - H)Y \tag{4.12}$$

where $H = P(P^TP)^{-1}P^T$ is the projection matrix on the principal component space. These partial residuals can be interpreted in term of a spectral transformation, TY where $T = PD_0P^T$ with $D_0 = diag(0_1, .., , 0_q, 1_{q+1}, ..., 1_r)$; i.e., the first q singular values are set to zero, and the others remain intact. Of course we have that $T = I_p - H$.

4.2.5 Gaussian Graphical Models

In Gaussian Graphical Model (GGM) (Shutta et al., 2022) statistical inference is based on conditional dependence of pairwise variables $(Y_j; Y_k)$ given the conditional set *rest* defined by all variables in the graph excluding $(Y_j; Y_k)$, and the pairwise partial correlations, ρ_{jk}=cor$(Y_j; Y_k|rest)$ are reflected in the elements of the precision matrix; i.e., the inverse of the covariance matrix, $\Theta = \Sigma^{-1}$. Specifically:

$$\rho_{jk} == \frac{-\theta_{jk}}{\sqrt{\theta_{jj}\theta_{kk}}} = 0 \iff \theta_{jk} = 0 \tag{4.13}$$

Thus the sparsity pattern of Θ contains the pairwise Conditional Independence (CI) relations encoded in the corresponding precision graph, and the problem of estimating a GGM is equivalent to the problem of estimating Θ.

With a large input graph, the CI evaluation is performed with the graphical LASSO procedure (Friedman, Hastie, and Tibshirani, 2008b), a sparse penalized maximum likelihood estimate (MLE) of the precision matrix, $\hat{\Theta}$. The LASSO (L1) penalty applied to Θ encourage sparsity and can produce shrinkage estimates equal to zero, $\hat{\theta}_{jk} = 0$.

We use the algorithm in `glasso()` function of **glasso** R package, that also include the option to estimate a graph with missing edges by specifying which edges are fixed zeroes for some elements, while regularization on the other elements is activated. The edge set of a GGM is therefore defined by the set of all pairs $(Y_j; Y_k)$ with nonzero elements in the estimated precision matrix.

Given that the confounding variables in a arbitrary regime is encoded in missing edges of a *priori* DAG, we built the GGM graph if $\hat{\theta}_{jk} \neq 0$, fixing to zero the DAG structure. Thus, the DAG edges are guaranteed to be absent in the resulting precision graph.

Successively, the adjusted (de-correlaed) data, as in BAP deconfounding, are obtained by Mahalanobis's transformation using the square root of the precision matrix estimated by glasso, $Z_i = \hat{\Theta}^{1/2}Y_i$.

LRpS. Alternatively in a pervasive regime, Low Rank plus Sparse (LRpS) (Chandrasekaran, Parrilo, and Willsky, 2012) of the GGM can be applied. LRpS learns a observed precision matrix assuming that variables are partially observed, $Y = (Y_O, Y_H)$ and the precision matrix is partitioned as:

$$\Sigma^{-1} = \begin{pmatrix} \Theta_O & \Theta_{OH} \\ \Theta_{HO} & \Theta_H \end{pmatrix} \tag{4.14}$$

The matrix, Θ_O describes the estimated precision matrix between p observed variables, the matrix, Θ_H represents the estimated effect of q hidden (latent) variables, and Θ_{OH} =cov$(Y_O; Y_H)$. By standard theory for multivariate Gaussian distributions, the conditional covariance matrix of $Y_O|Y_H$ is:

$$\Sigma_O^{-1} = \Theta_O - \Theta_{OH}\Theta_H^{-1}\Theta_{HO} = \Theta_O - L \tag{4.15}$$

For sparse DAGs, Θ_O is a sparse matrix and for $q << p$, L is low-rank with rank$(L) = q$. The Θ_O is the target matrix for inference. Hence, when the confounders are pervasive, it is possible to estimate each component through a low-rank plus sparse matrix decomposition using the Alternating Direction Method of Multipliers (ADMM) solver (Goldstein, Donoghue, and Setzer, 2014) applied to a covariance matrix in stage 1, and using the estimated covariance, $\hat{\Theta}_O^{-1}$ in stage 2, following the two-stage procedure suggested by Frot, Nandy, and Maathuis, 2019.

4.2.6 User interface

The `SEMbap()` pipeline employs the following R functions: `Shipley.test()` of **SEMgraph**, the `fitConGraph()` of **ggm**, the `glasso()` of **glasso**, and the `svd()` of **base** for bow-free covariance search, constrained estimation solver, glasso fitting and spectral decomposition, respectively.

The example code of the function `SEMbap()` is as follows.

```
SEMbap(graph, data, group = NULL, dalgo = "cggm",
       method = "BH", alpha = 0.05, hcount = "auto",
       cmax = NULL, limit = 200, ...)
```

The inputs are: an igraph object (*graph*); a matrix with rows corresponding to subjects and columns to graph nodes (*data*); a binary vector with 1 for cases and 0 for control subjects (*group*); the deconfounding method (*dalgo*, default = "cggm"); the multiple testing correction method (*method*, default = "BH"); the significance level (*alpha*, default = 0.05) and other optional inputs.

Both a graph and data input are required for methods involving BAP search, since the input graph will be used to recover the non-zero missing covariances. As the other methods involve SVD, only the data input is required.

Based on the deconfounding method that has been specified, `SEMbap()` will involve different computational steps:

- "cggm" (default) (i) BAP recovery through `Shipley.test()`; (ii) estimation of the constrained precision matrix, Ψ^{-1} through `fitConGraph()` function of **ggm** R package; (iii) obtain the de-correlated data matrix Z by multiplying the data matrix, Y rightward by the square root of the estimated precision matrix, $Z = Y\hat{\Psi}^{-1/2}$.

- "glpc": (i) BAP recovery through `Shipley.test()`; (ii) fitting gLPCA and obtain confounding proxies as the last q principal component scores; (iii) extend the DAG by including these confounding proxies and add these LV scores to the data matrix, `Z=cbind(P,Y)`.

- "pc": (i) First q principal components (projected scores) to obtain the factor scores proxies; (ii) extend the DAG by including these confounding proxies and add these LV scores to the data matrix, `Z=cbind(P,Y)`.

- "trim" or "pcss": (i) SVD of the observed data; (ii) compute spectrum transformation matrix, T of "trim" or "pcss" methods; (iii) obtain adjusted data matrix by multiplying observed data Y by the spectral transformation matrix, $Z = TY$.
- "limit=200", beyond this limit, the precision matrix is estimated by "glasso" algorithm to reduce the computational burden of the exhaustive BAP search.

A list of four objects:

- *dag*, the DAG extracted from input graph. If (*dalgo* = "glpc" or "pc"), the DAG also includes LVs as additional source nodes.
- *dsep*, the data.frame of all d-separation or CI tests over missing edges in the DAG. If (*dalgo* = "pc" or "trim"), d-separation dataframe is equal to NULL.
- *adj*, the adjacency matrix of selected covariances; i.e, the missing edges selected after multiple testing correction. If (*dalgo* = "pc" or "trim"), adjacency matrix is equal to NULL.
- *data*, the adjusted (de-correlated) data matrix or, if (*dalgo* = "glpc" or "pc"), the combined data matrix where the first columns represent LVs scores and the other columns the raw data.

To read more about `SEMbap()` function, in terms of description, usage, function arguments and value, see help documentation: `?SEMbap` or refer to https://rdrr.io/cran/SEMgraph/man/SEMbap.html.

4.3 Experiments

For testing and comparing the performance of the mentioned deconfounding methods we use simulation and real data. Our simulation set-ups consider a (2 graph dimension x 2 sample size x 6 confounding types) design with 100 randomization per design levels as reported in Table 4.1.

TABLE 4.1: Overview of the 4×6 simulation design.

		dense			sparse		
		1LV all	3LVs cluster	3LVs overlap	HDLVS sporadic	HDLVS interconnected	DAG
n=100	p=32	100	100	100	100	100	100
	p=190	100	100	100	100	100	100
n=400	p=32	100	100	100	100	100	100
	p=190	100	100	100	100	100	100

Starting from the "Amyotrophic lateral sclerosis" (ALS) pathway from KEGG database (Kanehisa and Goto, 2000), two subgraphs have been extracted to test for different dimensions of number of variables p in the simulated data. The small graph is a subgraph with 32 nodes and 47 edges whilst the larger one has 190 nodes and 259 edges. Hence, the number of variables is varied in $p \in \{32, 190\}$ (see Figure 4.1 and Figure 4.2).

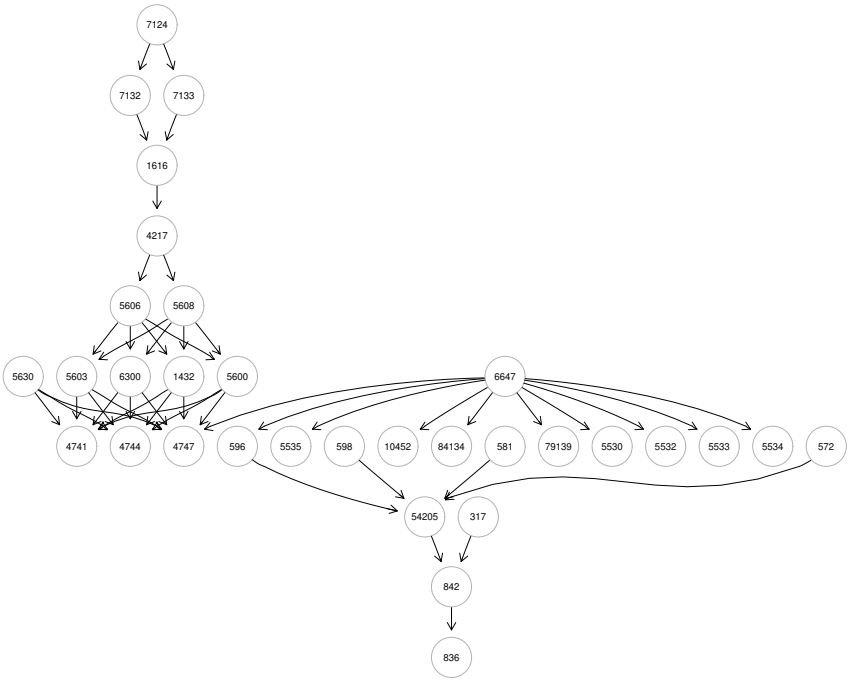

FIGURE 4.1: Small subgraph from ALS pathway for simulated data (32 nodes and 47 edges).

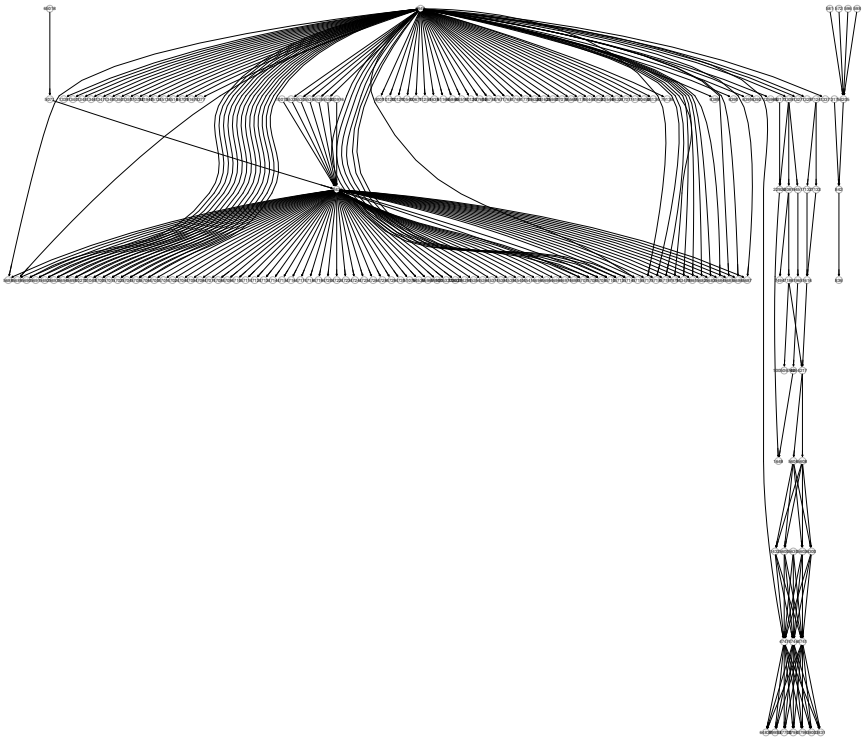

FIGURE 4.2: Large subgraph from ALS pathway for simulated data (190 nodes and 259 edges).

The number of samples is varied in $n \in \{100, 400\}$ to test for situations of, respectively, high ($p = 190 > n = 100$) and low ($p = \{32, 190\} < n = 400$) dimensionality. In the former, the covariance matrix could not be definite positive, preventing parameter estimates. When this occurs, the function `pcor.shrink()` of the **corpcor** R package implements the James-Stein-type shrinkage estimator, which enables covariance matrix regularization. However, in some cases it could happen that matrix regularization generates a near identity matrix and, as a result, misleading evaluation metrics. To avoid this issue, in these cases we also set $n = 400$. In detail, we consider six scenarios described below. For each of these, we generate n independent errors, $E(n, p)$ from a multivariate normal distribution with a mean vector, $\mu(p, 1)$ and a covariance matrix, $\Sigma(p, p)$ that has an idiosyncratic component and a component due to confounding. All edge weights (i.e., the non-zero entries of the DAG coefficient matrices, $B(p, p)$ are drawn from an uniform distribution on the interval between 0.1 and 1, whilst their signs are drawn from a Bernoulli distribution with probability 0.5. Then, data have been generated according to $Y = E(I - B)^{-1}$.

The mean vector is sampled from an uniform distribution on the interval from 0.05 to 0.75 to recreate differential expression between cases and controls for the 25% of p genes, otherwise the mean vector is equal to zero.

Error covariance matrix can be represented by (i) a random Factor Analysis (FA) model, $\Sigma = \Gamma\Gamma^T + diag(\sigma_1, ..., \sigma_p)$, where $\Gamma(p, q)$ is the matrix of factor loadings, $\Gamma\Gamma^T$ represents the shared variance in the common factor structure, and the diagonal elements σ_j are referred to as the specific variances of the variables; (ii) a random uniform distributions, $U(min; max)$; or (iii) a random small-word network generate by Watts-Strogats (WS) model (Watts and Strogatz, 1988) defined by dimension, neighborhood and rewired probability, $SW(d, nei, p)$.

Based on how the few LVs affect the observed variables, two different main confounding design have been investigated:

(i) **Dense confounding**: the effect of few LVs is "spread out" over most of the observed variables;

(ii) **Sparse confounding**: every confounding variable affects few variables in the dataset.

The six scenarios that are considered distinguish themselves by a different structure of the error covariance matrix, Σ the number of latent confounders, q and overall strength of latent confounding. The diagonal entries of Σ are defined as random uniform terms sampled from $U(0.1; 0.9)$. According to the chosen initial graph (small or high dimension), variances of source nodes is set to 1.

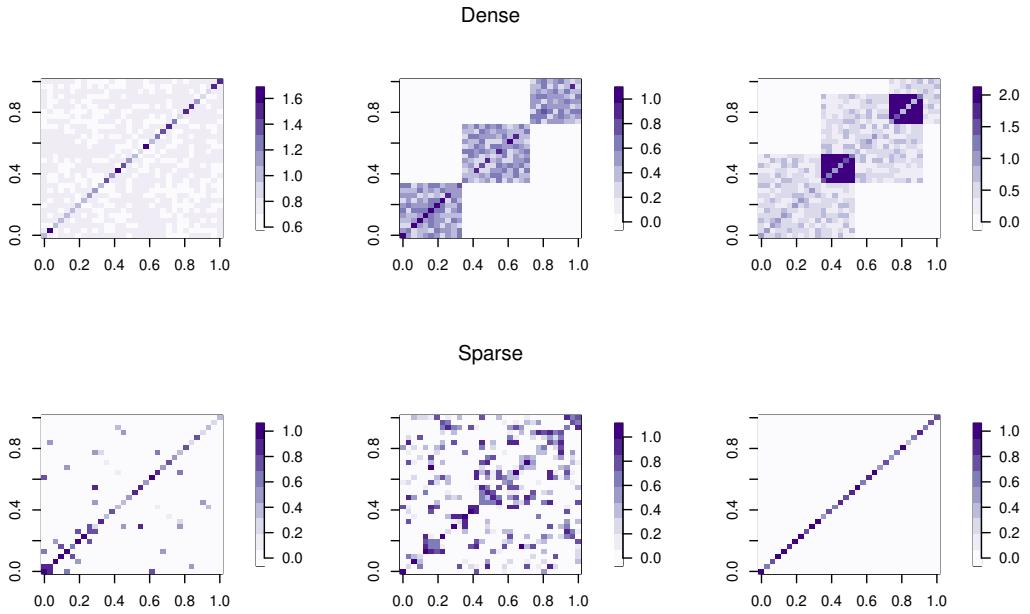

FIGURE 4.3: Example of simulated covariance matrices with n = 400 and p = 32 for each confounding design (dense and sparse). For dense confounding, three covariance patterns have been generated (starting from the left): (1) 1 LV all; (2) 3 LVs cluster; (3) 3 LVs over. The remaining sparse scenarios (starting from the left) are: (1) HDLVS sporadic; (2) HDLVS interconnected; (3) DAG.

Three scenarios regard the dense confounding design while the remaining three the sparse confounding one. Both dense and sparse scenarios can be better visualized in Figure 4.3.

The three dense scenarios can be listed as follows:

(i) **One LV all**: $q = 1$ LV affects all the observed variables. This is a FA scenario where all the covariances are non-zero, with factor loadings sampled from an uniform distribution, $U(0.64; 0.81)$, respectively from medium to high loadings according to Widaman, 2018;

(ii) **Three LVs cluster**: $q = 3$ LVs affect three (not overlapping) blocks of observed variables. This is a FA scenario where three blocks of covariances are non-zero, with factor loadings sampled from an uniform distribution, $U(0.2; 0.7)$, respectively from low to medium loadings;

(iii) **Three LVs over**: $q = 3$ LVs affect three (overlapping) blocks of observed variables. This is a FA scenario where three blocks of covariances are non-zero, with factor loadings sampled from an uniform distribution, $U(0.2; 0.7)$, respectively from low to medium loadings, with loadings larger than 0.7 if more than one LVs affect a specific variable.

The remaining three sparse scenarios are:

(i) **High Dimensional LVs (HDLVS) sporadic**: many LVs affect sporadic (isolated) observed variables. This is a scenario characterized by hidden confounding with no modularity (no groups of the nodes that are more densely connected together than to the rest of the network) but with random affected nodes which are isolated. There are many non-zero covariances sampled from an uniform distribution, $U(0, 1)$;

TABLE 4.2: Overview of the considered deconfounding methods.

Method	Algorithm	Input data	Confounding assumption	BAP	SVD
CGGM	Constrained Gaussian Graphical Model (w/ dsep search)	Gene expression and graph object	Arbitrary	Yes	No
gLPCA	Graph-Laplacian PCA (w/ dsep search)	Gene expression and graph object	Mixed	Yes	No
PCA	Singular Value Decomposition (SVD)	Gene expression	Dense	No	Yes
Trim	Spectral trasformation	Gene expression	Dense	No	Yes
PCSS	Spectral trasformation	Gene expression	Dense	No	Yes
gLASSO	Graphical LASSO	Gene expression	Arbitrary	No	No
LRpS	Low rank plus sparse decomposition	Gene expression	Dense	No	No

(ii) **HDLVS interconnected**: many LVs affect few interconnected modules of observed variables. This is a scenario characterized by hidden confounding with high modularity of affected nodes. There are many non-zero modules of covariances sampled from Watts-Strogats (WS) model, $SW(d = p, nei = 5, p = 0.9)$;

(iii) **DAG**: negative control with no hidden confounding. The covariances are 0.

While, for real data we make use of the (pre-processed) breast cancer RNA-seq dataset from TCGA project, also analyzed in Jablonski et al., 2021.

Table 4.2 provides a summary of the deconfounding methods in terms of type of algorithm employed, input requirements, confounding assumption and methodological steps together with main papers for reference. We compare the methods included in our `SEMbap()` function plus LRpS procedure provide by the R package **lrspadmm** (https://github.com/benjaminfrot/lrpsadmm), using default arguments. While for `glasso()` function activated in `SEMbap()` with `limit=1` the regolarized "rho" argument is fixed to the canonical, $\rho = \sqrt{log(p)/n}$ and the "zero" argument is fixed to the indices $(j;k)$ indicating the DAG structure.

Besides the type of algorithm, these methods differ in three main aspects: (i) the input requirements, gene expression data and graph object or only the former; (ii) the confounding assumption, arbitrary or pervasive; (iii) methodological steps, BAP search or SVD.

CGGM, gLPCA and gLASSO requires as input also a graph object, since CGGM and gLPCA algorithm involves BAP search with d-separation tests between all pairs of DAG missing edges, while for gLASSO the precision matrix was constrained fixing zero entry for DAG edges. The remaining methods only require a gene expression data as input since their approach involves a SVD on observed data or the ADMM as for LRpS algorithm. Most of the methods work under the structural assumptions regarding the sparsity of the underlying DAG and the denseness of latent effect, except for CGGM, gLPCA and gLASSO, where the confounding assumption is arbitrary or mixed. As a result, since different experimental designs have been tested within simulation runs, some methods are expected to perform better than others depending on the starting confounding assumption. Hence, the goal is to find a deconfounding method that represents an optimal solution in both situations.

4.4 Metrics

Varying the covariance matrix structure, number of samples, n dimension, p and strength of the latent confounders, we run 100 simulations of each unique parameter configuration and compute the following quantities:

1. **Recovery performance measures**. Once obtained the estimated confounding covariance matrix, $\hat{\Sigma}$ for each method, we try to recover the component due to confounding in the form of adjacency matrix ($[0, 1]$ entries) to be easily comparable with true hidden confounding matrix. Specifically, non-zero entries correspond to LVs effects. Easily for methods based on BAP search (CGGM and gLPCA), the hidden component is represented by the adjacency matrix of the BAP covariances. Differently, the other methods report the hidden component as a continuous output (being a part of the estimated covariance matrix). Specifically, the outputs obtained by each method can be listed as follows:

   - the matrix of $\hat{\Gamma}$ coefficients obtained from SEM fitting for PCA, thus the common factor covariance, $\hat{\Gamma}\hat{\Gamma}^T$;
   - the estimated inverse covariance matrix from glasso fitting;
   - the estimated dense low-rank matrix from LRpS procedure;
   - NULL covariance for SVD methods.

   These continuous matrices have been converted to binary $[0, 1]$ format by applying a reasonable threshold to the absolute values of the confounding matrix. In the same way, the true hidden confounding matrix has been recovered from the covariance matrix of the simulated data, putting 1 for the non-zero entries. In the end, the two confounding adjacency matrices have been compared to obtain the 2x2 frequency table (i.e. confusion matrix) and the classical performance indices (precision, recall and f1-score). Let TP be the true positives, FP be the false positive, TN be the true negative, and FN be the false negative. Then, Pre = TP/(TP+FP), Rec =TP/(TP+FN), and f1-score = (2*Rec*Pre)/(Rec+Pre). The higher the metrics, the better the performance. In addition, (iv) false positive rate, fpr = FP/((TP+FP) has been recovered to evaluate if, in the DAG scenario with no confounding, the methods still recognize the presence of LVs. For SVD methods with NULL confounding covariance, none performance metrics have been computed.

2. **Goodness-of-fit measures**. We obtain the adjusted data from each method, accounting for estimated hidden confounding. Then, we fit the ALS graph (small and large) via `SEMrun()` function of **SEMgraph** R package considering the unadjusted data (with hidden confounding) and the adjusted data. We obtain SEM evaluation metric using the Standardized Root Mean Square Residual (SRMR): the square root of the average of squared standardized residuals between the observed and the hypothesized covariance. This metrics has been compared with the reference cut-off suggested from SEM literature (0.08 for SRMR). The lower the value, the better the performance. In addition, it is possible to identify differentially regulated nodes (DRNs), or variables that exhibit a statistically significant difference in their activity (for example, gene expression) between the experimental and control groups, by taking into account an exogenous group variable acting over a common model. Node activation and node inhibition P-values ($P+$ and $P-$, respectively) have been combined

through a Bonferroni statistics ($P = 2min(P+; P-)$) to obtain a measure of the ability of each method to recover group perturbation of simulated data despite confounding adjustment. As the latter statistics was transformed by the negative logarithm function, nlog10(P) the higher the value, the better the performance. The ability of each method was evaluated in terms of recovery the perturbation level of not-adjusted data when removing hidden confounding. Moreover, the absolute number of nodes showing significant variation in cases with respect to healthy controls has been reported (vcountP).

4.5 Results

We report here the performance metrics on simulated data with n=400, and the goodness-of-fit measures on real breast cancer RNA-seq dataset.

Simulated data. Considering the simulated run with n=400 of Figure 4.4, we observe that CGGM/gLPCA based on BAP search recovers an high proportions of the simulated covariance representing the sparse/dense hidden confounding (f1-score around 0.9 for the large graph and around 0.75 for the small graph). PCA approximately has an f1-score of 0.7 for all dense setting, as expected. Similarly, given the pervasive confounding assumption of LrPS, it reports high f1-score (0.9) in the 1LV setting. Lowest scores are reported by gLASSO (0.3-0.5). These results are representative of the results obtained for different recovery measures.

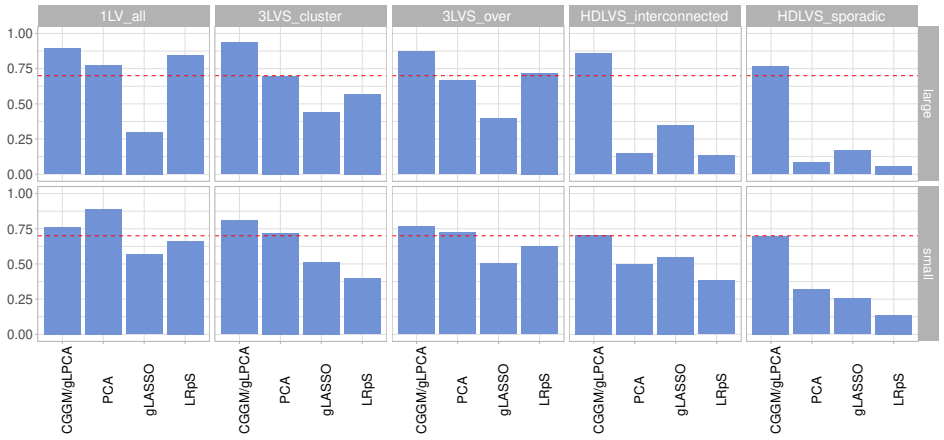

FIGURE 4.4: F1 score summarised as mean over simulations for dense/sparse confounding design with n=400. For SVD methods with NULL confounding covariance, none performance metrics have been computed.

The relative performance of all methods has been summarized under different experimental conditions on 100 simulation replications to better quantify the efficiency of each deconfounding method. Note that the results referring to the case with $n = 400$ will be discussed, since in this case we obtain more robust evaluation metrics. However, the results regarding the experimental design with $n = 100$ show similar conclusions.

Classification performance. Obviously, some methods are expected to perform better than others in some experimental set-ups based on their confounding assumption (i.e. arbitrary or dense). As the recovered adjacency matrix representing hidden

confounding is the same, classification metrics for CGGM and gLPCA have been aggregated.

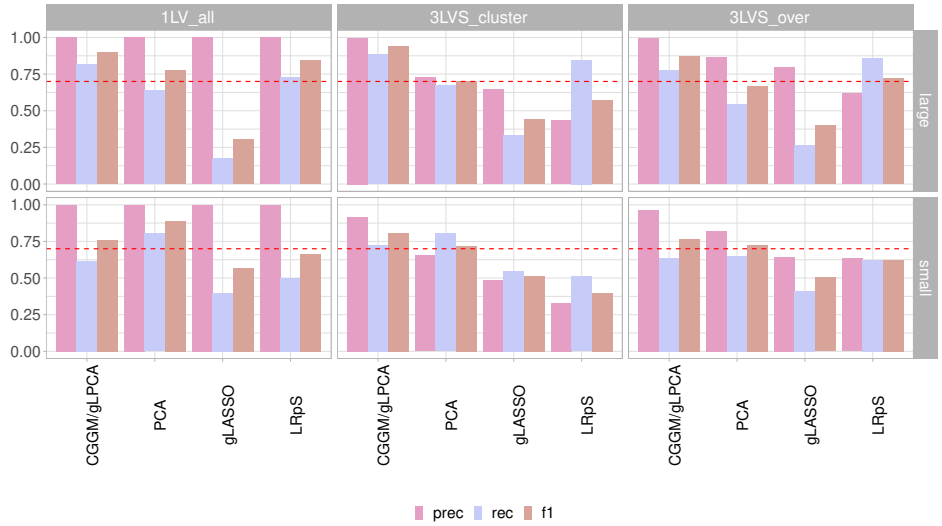

FIGURE 4.5: Precision, recall and f1 score summarised as mean over simulations for dense confounding design with n=400.

Figure 4.5 shows that, respectively, for the dense confounding design CGGM/gLPCA recovers both an high proportion of covariances compared to the true ones (recall) and an high number of correctly identified covariances over the estimated ones (precision). This result allows to reach an f1-score around 0.9 for the large graph and around 0.75 for the small graph scenario. PCA seems to reach the level of (approximately) 0.7 f1-score for all the dense confounding scenarios, except for $1LV\_all$ design in the small graph case where the method exceeds the threshold of 0.7, reaching an f1-score of 0.9. Given the pervasive confounding assumption of LRpS, the latter reports high f1-score (around 0.7-0.8) for $1LV\_all$ design and lower metrics for the other dense scenarios. Lowest scores are reported by gLASSO (0.3-0.5).

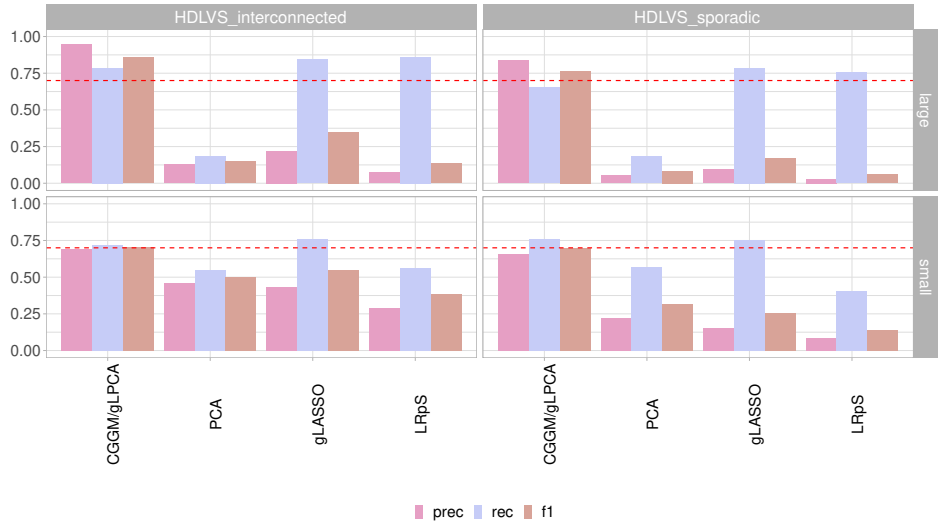

FIGURE 4.6: Precision, recall and f1 score summarised as mean over simulations for sparse confounding design with n=400.

Figure 4.6 reports also high classification metrics for CGGM/gLPCA, with an f1-score around 0.7 for almost all sparse confounding scenarios, with a maximum of 0.9 for the *HDLVS_interconnected* design. However, in this sparse scenario, PCA reports a f1-score around (or below) 0.15 for the large graph case; the latter method is able to reach the 0.5 threshold only in the *HDLVS_interconnected* design. Almost same conclusions as for PCA can be reported for gLASSO and LRpS. Generally, in most of the sparse cases, the number of the correctly identified covariances over the estimated ones was low (precision).

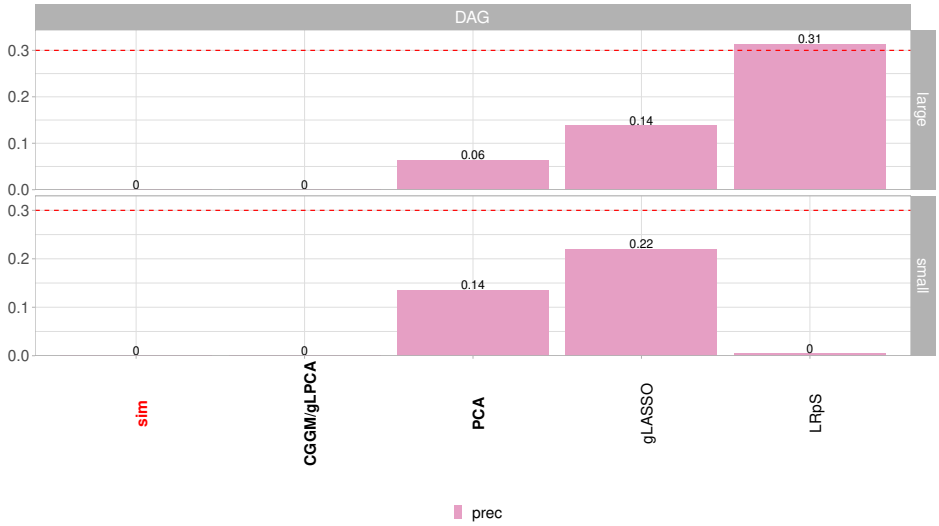

FIGURE 4.7: False positive rate summarised as mean over simulations for sparse confounding design (DAG) with n=400.

In addition, Figure 4.7, shows if the methods are able to control fpr in the DAG scenario with no hidden confounding. CGGM and gLPCA show a fpr equal to 0. Also PCA is able to control the error rate, reporting a fpr around 0.1. However, the largest scores are reported by gLASSO for the small graph case (0.22) and LRpS for the largest one (0.31). High proportion of false hidden confounding is recovered by those methods.

Goodness-of-fitting performance. As for benchmark data analysis, a good performance is also characterized by a low srmr value together with a good perturbation level. Figures 4.8 and 4.9 show the srmr for each method summarized as mean across simulations, respectively for dense and sparse confounding design.

Almost all the methods are able to lower the srmr of the simulated data but only some are able to lower it below the threshold value of 0.08. The lowest values of srmr are reported by PCA in almost all sparse and dense confounding scenarios. CGGM reports low values but around 0.1 for most of the cases whilst gLPCA has higher srmr score given that, to prevent overfitting when the number of identified clusters in the recovered subnetwork is higher than 3, the gLPCA procedure switches to full Laplacian graph. The worst srmr score, around 0.2, is reported by PCSS in both the dense and sparse confounding design (large graph case).

Figure 4.10 and 4.11 report the results with regards to the perturbation level for, respectively, dense and sparse scenario. Surprisingly, almost all the methods are able to recover the same level of perturbation of simulated data despite the adjustment for hidden confounding. Only LRpS is not able to recover enough data perturbation, consistently with benchmark data results.

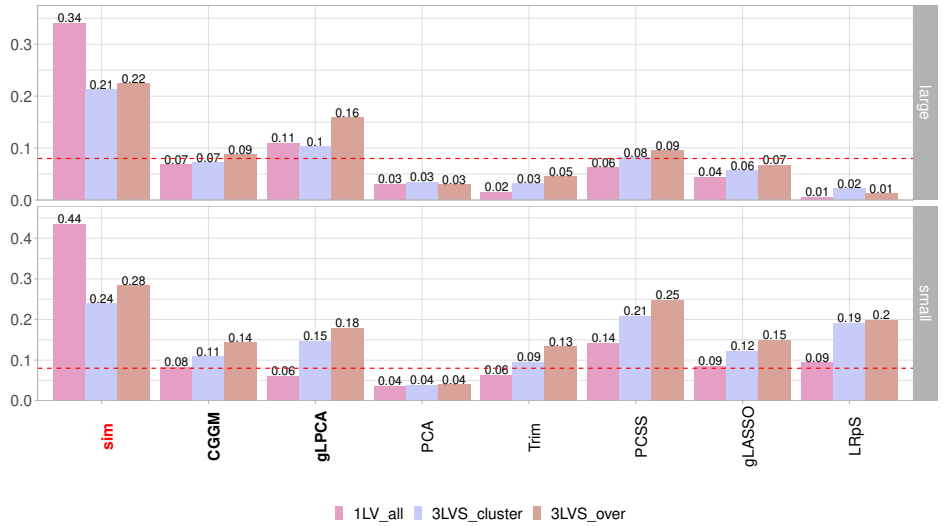

FIGURE 4.8: SRMR summarised as mean over simulations for dense confounding design with n=400.

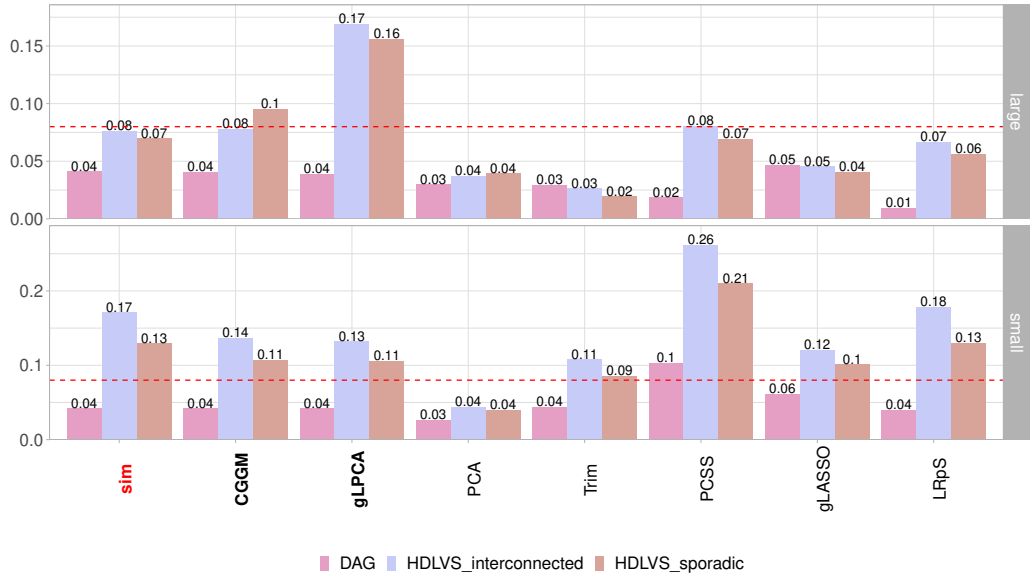

FIGURE 4.9: SRMR summarised as mean over simulations for sparse confounding design with n=400.

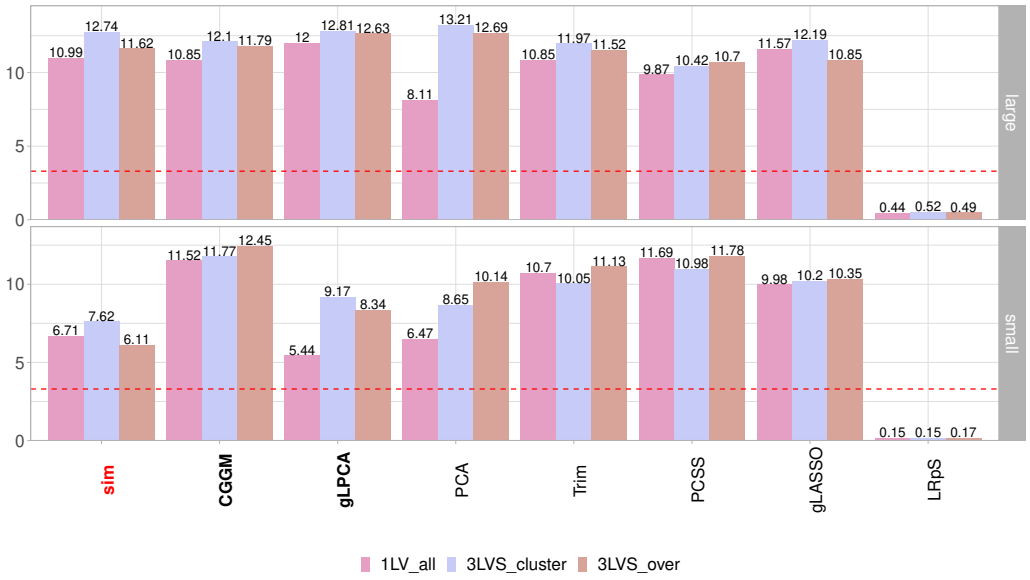

FIGURE 4.10: Perturbation (nlog10P) summarised as mean over simulations for dense confounding design with n=400.

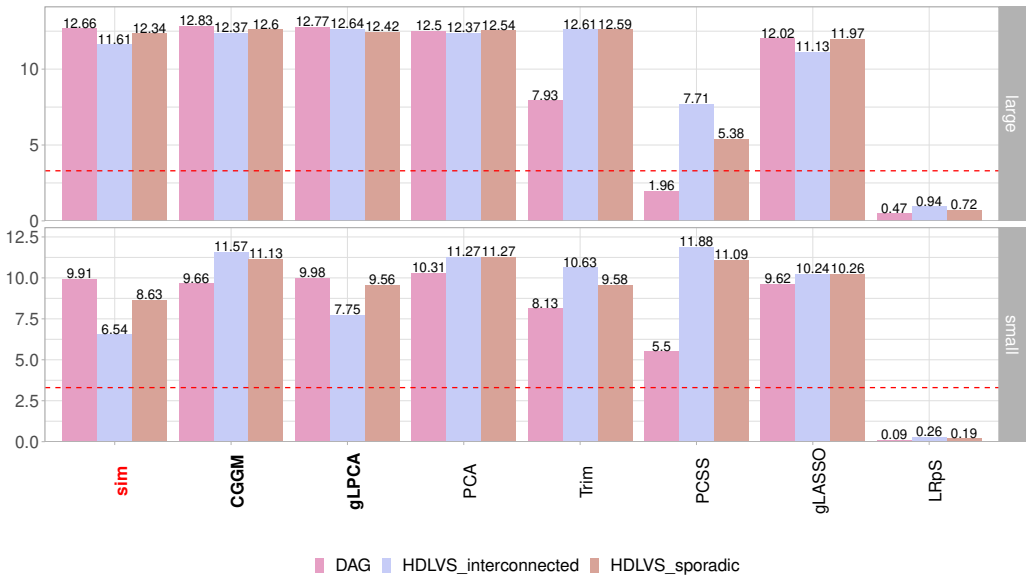

FIGURE 4.11: Perturbation (nlog10P) summarised as mean over simulations for sparse confounding design with n=400.

Benchmark data. For testing and comparing the performance of the mentioned deconfounding methods we use the (pre-processed) breast cancer RNA-seq dataset from TCGA project (Jablonski et al., 2021). The extracted data matrix has $p = 20501$ genes and $n = 224$ human samples grouped into 112 normal and 112 tumor subjects. As a knowledge graph, we retrieve the "Breast Cancer Pathway" (*hsa05224*) from KEGG which contains 147 nodes and 509 edges. There are a total of 22 transcription factors (TFs) into the pathway if matching with TRRUST reference database, which states that in the actual causal ordering the TFs should come before the others genes. Thus, we use 109 nodes and 397 edges after removing TFs and mapping on the benchmark breast cancer data.

By removing the TFs, we can evaluate how well the comparison methods concealed confounding. TFs are correlated with a lot of genes, but we analyzed only the remaining genes. Suppose that the "observed" gene, Y_j is strongly correlated with one of the "latent" TF, X_k. Since we know the true values of the TFs, we would expect a sharp decrease in the high correlation, cor$(Y_j; X_k)$ after subtracting out the TF confounding variation.

The number of degree of freedom, i.e. the number of missing edges to be tested in BAP search is 5489, and the basis set, S_U have been properly tuned at $cmax = 3$ parents. The number of confounding proxies (LVs) for PCA deconfounding methods (i.e. gLPCA, PCA and PCSS) has been determined according to a permutation method and then the scree plot has been visualized where eigenvalues are displayed against the number of the principal component. We've selected an optimal number of 3 confounding proxies that explains 41% of total variance, based on trade-off between SEM fitting and perturbation metrics.

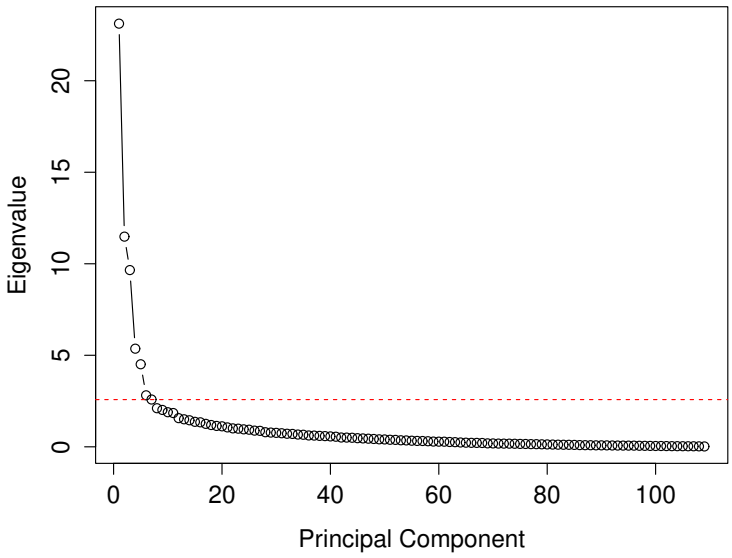

FIGURE 4.12: Scree plot of the eigenvalues against the number of the principal component in benchmark data analysis.

We want to evaluate if the methods are able to adjust the data while retaining most of data perturbation, see Table 4.3 for benchmark results. Some methods performs better than others on both sides. gLPCA reaches the lowest srmr value

TABLE 4.3: Evaluation metrics (srmr, dev/df, nlog10(P) and vcountP) from benchmark data analysis.

Method	**srmr**	**dev/df**	**nlog10P**	**vcountP**
Unadjusted	0.15	3.57	13.77	85
CGGM	0.09	2.45	12.58	55
gLPCA	0.08	2.64	13.63	71
PCA	0.08	2.59	11.90	69
PCSS	0.13	7.02	2.70	6
Trim	0.05	0.90	3.14	35
gLASSO	0.08	1.98	5.45	44
LRpS	0.02	0.10	0.17	5

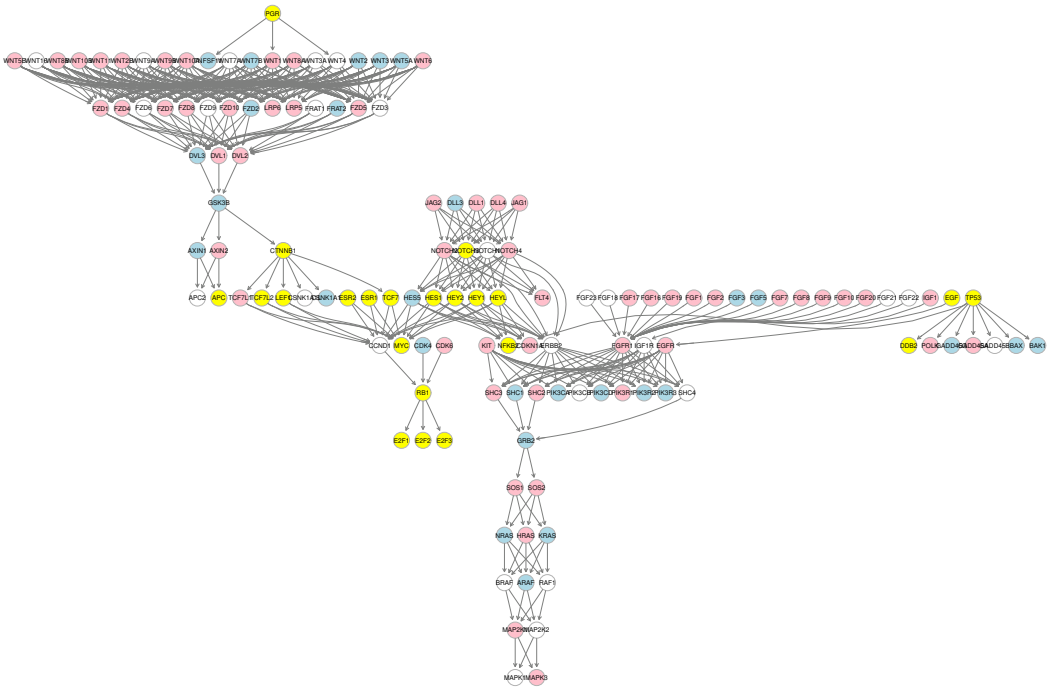

FIGURE 4.13: Perturbed original subnetwork from benchmark data analysis. Nodes in the recovered subnetworks are coloured in yellow if they represent TFs, pink-shaded if significantly activated or blue-shaded if signficantly inhibited.

(0.08) while retaining a good perturbation level (13.63) with 71 differentially regulated nodes (DRNs) over the total of 85 for the unadjusted data. On the other side, gLPCA report the same srmr value (0.08) but slightly lower data perturbation metrics (nlog10P = 11.90 and vcountP = 69). Also CGGM reports a good performance but with a lower number of DRNs (55).

Trim shows good fitting metrics but, even if has 35 DRNs, the combination of node activation and inhibition p-values (as summarised by nlog10p = 3.14) is really low if compared to the previous methods. Same conclusions as for Trim can be reported for gLASSO, with an srmr value equal to 0.08 and 44 DRNs. LRpS recovers the lowest srmr value (0.02) but has also the lowest level of retained perturbation. Thus, the method aggressively adjusts the data while losing a huge portion of information. The srmr value could be a sign of overfitting problems. Same conclusions can be reported for PCSS.

By removing the TFs, we can evaluate how well the techniques handle concealed confounding effect, visualizing the scatter plots with the transcription factor TCF7L2, i.e. EGFR and BAK1, respectively. In Figure 4.14 and 4.15, we examine the highest positively and negatively linked genes with the transcription factor TCF7L2, i.e. EGFR and BAK1, respectively. Each gene's unadjusted expression level correlates well with TCF7L2, as can be seen. We observe how the shared confounding effect of the transcription factor TCF7L2 is removed using the various deconfounding techniques. The points seem to be scattered randomly, showing no correlation (or really low values) between the two variables. Since gLPCA and PCA methods add to data the first principal components without further changes, adjusted gene expression levels have been obtained as residuals by subtracting the LVs' effect from the response variable of interest.

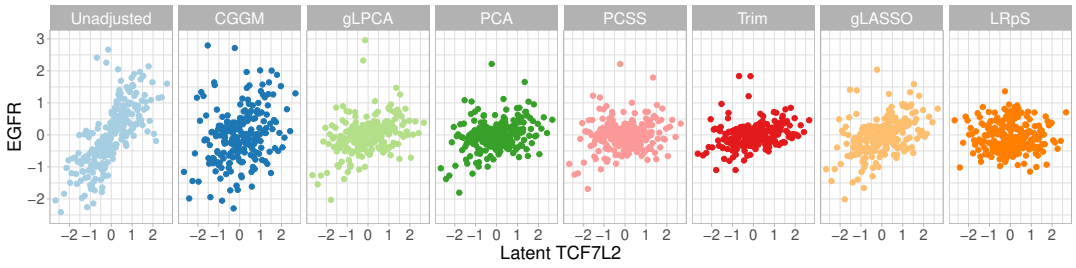

FIGURE 4.14: Gene EFGR has a positive correlation greater than 0.5 with the unobserved transcription factor TCF7L2 (Unadjusted). After subtracting out the confounding variation estimated using the different methods for each gene (denoted as "deconfounded" expression level), the genes are no longer correlated with TCF7L2.

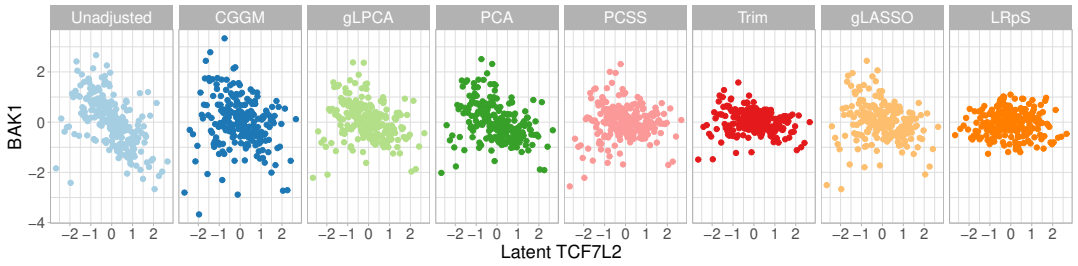

FIGURE 4.15: Gene BAK1 has a negative correlation greater than 0.5 with the unobserved transcription factor TCF7L2 (Unadjusted). After subtracting out the confounding variation estimated using the different methods for each gene (denoted as "deconfounded" expression level), the genes are no longer correlated with TCF7L2.

4.6 Discussion

We have discussed the problem of dealing with unobserved confounding factors to correctly quantify interesting biological signals. Building on existing literature (Chernozhukov, Hansen, and Liao, 2017; Cevid, Buhlmann, and Meinshausen, 2020; Chandrasekaran, Parrilo, and Willsky, 2012), a two-stage (deconfounding plus fitting) procedure based on Bow-free Acyclic Paths (BAP) search developed into the framework of SEM has been proposed. The existing deconfounding methods differ in the way they perform the first stage, i.e.:

- directly estimate confounding variables from the data as the scores of the first q principal components and simply add them to the data matrix, creating an augmented data matrix;
- transform data by applying a linear transformation that only transform the singular values of the data, while keeping it singular vectors intact;
- decompose the concentration matrix as a sum of a sparse matrix and a low-rank matrix where the latter reveals the number and effect of the hidden variables.

Instead, our approach first makes an exhaustive BAP search of missing edges with significant covariance with Shipley's independent d-separation local tests and then either (i) fit the inverse of the selected covariance matrix via CGGM and decorrelate the data matrix via Mahalanobis's trasformation or (ii) learn a low dimensional representation of the observed data matrix that incorporates graph structures and add the last q principal component scores to the data matrix.

This methodology differs from the other methods since it requires a priori graphical structure as input and makes use of both arbitrary (CGGM), mixed (gLPCA) or pervasive deconfounding assumption (based on the chosen combination of methodological steps) unlike the other approaches that operate only under dense LV regime.

After removing hidden confounding, based on the goal of the analysis, the methods can perform a second step where the modified data can be used as an input for SEM fitting, a high-dimensional sparse regression technique or for any structure learning algorithm. Since our approach starts from a knowledge-based biological network (i.e., either Breast cancer or ALS provided by KEGG database, in our examples), we aim to adjust the data for hidden confounding, map the adjusted data

matrix onto the input graph and convert it into a SEM to assess goodness-of-fit (srmr, dev/df) and perturbation recovery (nlog10P, vcountP).

In benchmark data analysis, best performances are reported by gLPCA and PCA, immediately followed by CGGM. As previously mentioned, these methods differ in their deconfounding assumption and methodological approach. To better evaluate how the methods perform in different setting, i.e. dense and sparse confounding scenario, different simulation set-ups have been generated. Simulation results confirm benchmark data analysis since report an outstanding performance of CGGM and gLPCA in both sparse and dense confounding scenarios. On the other side, as expected, PCA reports good evaluation metrics only for the dense scenario.

Note that, in the simulation data analysis, in addition to SEM evaluation metrics, the adjacency matrix representing hidden confounding has been recovered for each method and evaluated in terms of classification metrics (f1, precision, recall, tpr). Thus, we have provided three different optimal choices that can be used by the user based on its needs. PCA represents an efficient algorithm in case of dense confounding whilst CGGM and gLPCA can be implemented in case of sparse confounding or a mixture or both. However, PCA and gLPCA can be preferred over CGGM methodology because the former methods add the first (or last) principal components as additional source nodes without adjusting the existing data matrix.

In the end, `SEMbap()` is easily accessible to common users and provides several methods to deal with hidden confounding under several experimental conditions. However, the reader needs to be aware that, to obtain an optimal performance of the deconfounding methodology, the inputs of BAP search algorithms need to be properly tuned, especially with respect to:

- *cmax* (default = NULL): maximum number of parents set. In more detail, this option can only be applied to run tests where the number of conditioning variables does not go over the specified value. Conditional independence tests with a high dimensionality may not be very reliable. Our recommendation is to test bow-free covariances with basis set sizes that are near to the sparsity index, $s = \sqrt{n}/log(p)$ in order to drive the sparsity;

- *alpha* (default = 0.05): False discovery rate (FDR) significance level for Shipley's local d-separation tests. The data de-correlation process is controlled by this argument. A higher alpha level takes into account more hidden covariances, hence accounting for more confounding factors. Data de-correlation is not enabled if alpha = 0.;

- *hcount* (default = "auto"): The number of latent (or hidden) variables. By default hcount="auto", the hidden count is determined with a permutation approach where, permuting the columns of the data matrix, Y, the singular values are compared to what they would be if the variables were independent, and components are chosen if their singular values are greater than those of the permuted data (for a review see Dobriban, 2020).

Furthers studies will combine the deconfounding problem with causal discovery algorithms (Heinze-Deml, Maathuis, and Meinshausen, 2018), allowing the user to use the presented deconfounding approach not only starting from a priori knowledge-based network but also from a fully data-driven network. Moreover, once the hidden confounding has been removed, another data-driven network will be recovered to represent true data variation.

4.7 Conclusions

We have shown that `SEMbap()` is easily accessible to common users and provides several methods to deal with hidden confounding under several experimental conditions. We have introduced and validated (both on benchmark and simulated data) a two-stage deconfounding plus fitting procedure based on BAP search. Results report that CGGM and gLPCA are able to correctly identify hidden confounding whilst controlling false positive rate and achieving good fitting and perturbation metrics in both sparse and dense confounding scenarios. We believe that, both CGGM and gLPCA can valuable tools for practitioners when undertaking complex sparse confounding scenario while PCA can be used in case of pervasive confounding.

Chapter 5

SEMdag( )

5.1 Background

Causality is a complex topic with roots in many disciplines, including statistics, economics, epidemiology, computer science, and philosophy. The two primary fields of causality research are causal inference and causal discovery. The former emphasizes deriving causal knowledge directly from observable data. It is the process of evaluating whether an observed association actually reflects a cause-and-effect relationship. The latter aims to deduce causal structure from data. In other words, find a causal model that accurately reflects a dataset.

A formal representation of the interactions between the observable variables, such as a casual graph, is crucial for causal inference, or the process of quantifying the influence of a cause on its consequence. Such a basic premise for a graphical representation is so powerful when it comes to explanation. Following Spirtes, Glymour, and Scheines, 2000, it basically comes down to drawing arrows from a cause to an effect (outcome) in order to gain a qualitative description of the system under consideration. Contrasting sharply with this are black-box strategies, which rely only on data to make predictions about a result. As indicated in Bareinboim et al., 2022; Pearl, Glymour, and Jewell, 2016, these strategies are ineffective in terms of decision-making as well as explainability.

According to Lauritzen, 1996, a graphical model is a family of multivariate distributions connected to a graph, where the network's nodes stand in for random variables and the graph's connections denote permitted conditional dependency relationships between the corresponding random variables. A particular kind of graphical model called a causal graphical model interprets its edges as having direct causal effects.

In a wide range of fields, such as genetics (Sachs et al., 2005), finance (Sanford and Moosa, 2012), and social science (Newey, Powell, and Vella, 1999), a Directed Acyclic Graph (DAG) offers an elegant way to describe directional or causal structures among collected nodes. Learning the DAG structures from observable data has received a lot of attention recently from both academia and business.

Structure learning is well known to be computationally difficult, and several algorithms have been proposed to solve it, using one of three possible approaches: constraint-based algorithms (Spirtes, Glymour, and Scheines, 2000), which use conditional independence tests to learn the dependence structure of the data; score-based algorithms (Chickering, 2003; Yuan et al., 2018), which maximize some goodness-of-fit scores in the potential graph space; and hybrid algorithms, which combine both approaches (Tsamardinos, Brown, and Aliferis, 2006; Nandy, Hauser, and Maathuis, 2018). However, the majority of the aforementioned methods can only restore a DAG's Markov equivalence class. Exact DAG recovery has recently received a lot of attention. It has been demonstrated that algorithms based on correctly constructed

Functional Causal Models are capable of differentiating between various Directed Acyclic Graphs (DAGs) in the same equivalence class. This advantage is attributable to more data distributional assumptions than just conditional independence relations. Several forms of the SEM have been shown to be able to produce unique causal directions, and have received practical applications. By designing the function and noise, a group of functional causal models are proposed, such as linear model with equal error variances (EqVarDAG, Chen, Drton, and Wang, 2019), linear model with non-Gaussian error (LiNGAM, Shimizu et al., 2006), non-linear model with Gaussian error (ANM, Peters, Janzing, and Schölkopf, 2010), and causal additive model (CAM, Bühlmann, Peters, and Ernest, 2014).

The main contribution of this chapter is the development of a two-step algorithm for learning high-dimensional sub-Gaussian linear SEMs with the same error variances (Peters and Bühlmann, 2014), called `SEMdag()` and included in the R package **SEMgraph** (Grassi, Palluzzi, and Tarantino, 2022). First, a 1) a node (vertex) or layer (level) ordering of the p nodes is extracted and then 2) the DAG is estimated using penalized (L1) regressions (Shojaie and Michailidis, 2010). The estimated linear order is determined by a priori graph topological vertex (TO) or level (TL) ordering, or by using a data-driven Bottom-up (BU) approach. To investigate the utility of our approach, we used a training dataset for model training and a test dataset for evaluating classification performance. We performed four sets of experiments on Amyotrophic Lateral Sclerosis (ALS), Breast cancer (BRCA), Coronavirus disease (COVID-19) and ST-elevation myocardial infarction (STEMI). We tested the ability of our framework to discover plausible DAGs against of five popular causal discovery methods, i.e. PC (Spirtes, Glymour, and Scheines, 2000), GES (Chickering, 2003), ARGES (Nandy, Hauser, and Maathuis, 2018), directLINGAM (Shimizu, 2014), CAM (Bühlmann, Peters, and Ernest, 2014), NOTEARS (Zheng et al., 2018) to provide a meaningful comparison in terms of disease predictive performance.

The outline of the paper is as follows. Section 2 discusses the problem setting, introduces different classes of structure learning methods and, in the end, our contribution. Section 3 describes the experimental design and the evaluation scheme. Section 4 discusses about the results. We close with a brief discussion in Section 5 and the conclusions in Section 6.

5.2 Method and implementation

5.2.1 Graphical and structural equation models

A DAG is defined as $G = (V, E)$, where V is the vertex set and E is the set of directed edges. When there is an edge $(j, k) \in E$, the edge $j \leftarrow k$ is implied. The parent set and the set of children of the j-th node in the graph G are indicated, respectively, by the symbols $pa(j)$ and $sib(j)$. If $pa(j) = \emptyset$, the vertex j is a source (root) vertex in G; if $sib(j) = \emptyset$, the vertex j is a sink (leaf) vertex in G, otherwise the vertex j is a connector vertex in G.

If each variable in child set can be expressed as a linear combination of the variables in its parent set as shown below, the system of linear equation is as follows:

$$Y_j = \sum_{k \in pa(j)} \beta_{jk} Y_k + U_j, \quad j \in V \tag{5.1}$$

where Y_j and U_j are an observed variable and an unobserved (hidden) error term, respectively, while β_{jk} is a regression (path) coefficient. The error terms $U_1, ..., U_p$ are

independent with Gaussian distribution, $U_j \sim N(0, \sigma_j), j \in V$. In vectorized form model is expressed as:

$$Y = BY + U \text{ and } \text{Cov}(U) = \text{diag}(\sigma_1^2, ..., \sigma_p^2) \tag{5.2}$$

As result, the joint distribution of Y factorizes according to the following decomposition of the DAG, G: $P(Y) = \prod_{j=1}^{p} P(Y_j|pa(j))$. P is then called *Markov* w.r.t. G. Various assumptions for the model defined in Equation (2) are specified:

- *Causal sufficiency*: The absence of hidden (or latent) variables is referred to as causal sufficiency (Spirtes et al. 2000). For modeling hidden variables, there are two typical approaches: (i) they may appear as a dependence between the error terms, U or (ii) they may be explicitly modeled as nodes in the structural equations. The absence of latent confounding in equation (2) uses (i): the U terms are considered to be independent, i.e., $\text{cov}(U_j; U_k) = 0$ for all pairwise (j, k).

- *Causal faithfulness*: If there are no Conditional Independence (CI) relations other than those implied by the Markov property, the distribution of $P(Y)$ produced by Equation (2) is faithful to a DAG G. This indicates that using the so-called d-separation rule (Spirtes, Glymour, and Scheines, 2000), all CI can be read out from a DAG G if the distribution P is faithful to the DAG G. Given a set S, two nodes (k, j) are said to be d-separated if the conditional correlation between node j and k (given S) is equal to 0.

- *Acyclicity*: The DAG G needs to be acyclic, which implies that it is not feasible to start at any variable in the DAG, go ahead along the directed arrows, and then return to the same variable. Solution of structural equations requires that $(I - B)$ is invertible and can be interpreted as instantaneous feedback system that converges to a stable equilibrium. In this case the equilibrium is: $Y = (I - B)^{-1}U$.

- *Linearity and Gaussianity*: Nodes (observed variables) of the DAG G can be expressed as a linear combination of its parents plus independent Gaussian noise random variables, $U \sim N_p(0, D_{\sigma^2})$.

The different algorithm are discussed below, and their assumptions are summarized in Table 5.1 .

TABLE 5.1: The assumptions of the considered structure learning methods.

Method	Causal faithfulness	Causal sufficiency	Graph acyclicity	Model linearity	Gaussian error	Equal error variances
PC	yes	yes	yes	yes	yes	no
GES	yes	yes	yes	yes	yes	no
ARGES	yes	yes	yes	yes	yes	no
LiNGAM	no	yes	yes	yes	no	no
CAM	no	yes	yes	no	ni	no
NOTEARS	no	yes	yes	ni	yes	yes
SEMdag	no	yes	yes	yes	yes	yes

5.2.2 Structure learning methods

The problem of learning the structure of a SEM is as follows. Given an $n \times p$ data matrix, $Y := (Y_1, ..., Y_p)$ with i.i.d n rows drawn from G and a SEM $(B, \{\sigma_i^2\})$, we want to learn a $\hat{G}$ and a SEM $(\hat{B}, \{\hat{\sigma}_i^2\})$ from Y such that $G = \hat{G}$.

G is typically not identifiable from the distribution of Y, but we may determine its Markov equivalence class, or in other words, its Completed Partially Directed Acyclic Graph (CPDAG). Markov-equivalent DAGs have the same skeleton and v-structures (Frydenberg, 1990; Verma and Pearl, 1990b). A v-structure consists of the triple $u \to v \leftarrow w$, where u and w are not adjacent. Each Markov equivalence class may be represented as a CPDAG that can include both directed and undirected edges (Andersson, Madigan, and Perlman, 2000). Only when the edge $v \to w$ is shared by all DAGs in the equivalence class, a CPDAG has the edge $v \to w$. If a DAG with $v \to w$ and a DAG with $v \leftarrow w$ are both present in the class, hence the CPDAG has the undirected $v - w$.

To learn the CPDAG (assuming causal faithfulness) numerous structure learning techniques are used, which may be broadly divided into three classes (Spirtes, Glymour, and Scheines, 2000; Peters, Janzing, and Schölkopf, 2017).

Constraint-based methods. The first class is the constraint-based approach (Spirtes, Glymour, and Scheines, 2000; Kalisch and Bühlmann, 2007), which tests pairwise causal links using a local conditional independence criterion.

The PC algorithm (Spirtes, Glymour, and Scheines, 2000) carries the names of its creators, Peter Spirtes and Clark Glymour. In order to understand the structure of the underlying DAG, it does a number of conditional independence tests. In particular, it learns the CPDAG of the underlying DAG in three steps: (a) determining the skeleton, (b) determining the v-structures, and (c) determining additional edge orientations. The undirected graph created by swapping out all directed edges for undirected edges forms the skeleton of the CPDAG.

In step (a), starting with an entirely undirected graph, the PC algorithm develops its skeleton. It then evaluates the conditional independence of Y_j and Y_k given Y_s for all $S \subseteq adj(j)$ k with $|S| = r$ and for all $S \subseteq adj(k)$ j with $|S| = r$ for the $r = 0, 1, 2, ...$ and neighboring nodes j and k in the current skeleton. If a conditional independence is discovered (i.e., the independence null hypothesis was not rejected at some significance level, α), the edge is removed, and the corresponding separation set S is stored. If the degree of the graph is equal to the size of the conditioning set, step (a) comes to an end.

In step (b), all edges are replaced by $-$, and the algorithm then takes into account all unshielded triples, or triples $i - j - k$ where i and k are not contiguous. The algorithm decides whether or not to align the triple as a v-structure with $i \to j \leftarrow k$ based on the separating set that caused the removal of $i - k$.

In step (c), additional orientation criteria are applied to orient as many of the remaining undirected edges as possible, for detail see (Meek, 1995).

The PC algorithm was shown to be consistent in certain high-dimensional settings (Kalisch and Bühlmann, 2007). There are various modifications of the algorithm. We consider the stable and order-independent version (Colombo and Maathuis, 2014).

Score-based and hybrid methods. Score-based methods (Chickering, 2003; Yuan et al., 2018) rely on the fact that each DAG, $G \in \mathbb{G}$ may be scored in relation to the data, often using a penalized likelihood score, e.g, the BIC (Schwarz, 1978):

$$\hat{G} \in \underset{G \in \mathbb{G}}{\arg\min}\ S(G;Y) := -\log L(Y;G) + \lambda|E| \tag{5.3}$$

where $L(Y;G)$ is the likelihood function of the SEM mapped on the DAG G, $|E|$ represents the number of parameter (edges) in the model, and λ is a penalized parameter ($\lambda = \log(n)$ for BIC). The algorithm then look for a CPDAG that gives the best score. Greedy techniques are often utilized because the space of potential graphs, $\mathbb{G}$ is too large. One of these is the well-known two-phase approach known as Greedy Equivalence Search (GES, Chickering, 2003). Specifically, by doing a search on the space of potential CPDAGs through Markov equivalence classes, GES discovers the CPDAG of the underlying causal DAG. In the forward phase of its greedy search, it does single edge additions to maximize score improvement, and in its backward phase, it performs single edge removals. High-dimensional consistency of GES was demonstrated by Nandy, Hauser, and Maathuis, 2018.

The hybrid methods learn the CPDAG by combining the ideas of constraint-based approach and score-based methods. Among the hybrid approaches, here we consider a novel version of the GES algorithm, called adaptively restricted greedy equivalence search (ARGES), has been introduced by Nandy, Hauser, and Maathuis, 2018. ARGES use a greedy search on a restricted search space using as input the skeleton of the PC algorithm or an estimated conditional independence graph (CIG), i.e. an undirected graph with edge between Y_j and $Y_k \iff \text{cor}(Y_j; Y_k|rest) \neq 0$, derived from a preliminary search. But also changes adaptively the forward phase of GES, by restricting edge additions. Let G be the loop CPDAG and X and Y be two of its non-adjacent vertices. Then an edge connecting X and Y is acceptable if (i) X and Y are adjacent in the (estimated) skeleton of G or (ii) there is a node Z such that $X \to Z \leftarrow Y$ is a v-structure in G. At every stage of the algorithm, shields of v-structures (or unshielded triples) in the current CPDAG are allowed in addition to the CIG's (or CPDAG-skeleton's) edges. ARGES scales well to sparse graphs with thousands of variables, and as GES, the output is a consistent estimate of the CPDAG.

Order-based methods. Exact DAG recovery (without causal faithfulness assumption) has recently received a lot of attention. It has been demonstrated that algorithms based on correctly model definition are capable of differentiating between various DAGs in the same equivalence class. This advantage is attributable to more data distributional assumptions than just conditional independence relations. Different studies have emphasized that under certain conditions, such as linearity with constrained error variances, linearity with non-Gaussian errors, and non-linearity with additive errors, unique identification is achievable by topological ordering search.

The topological ordering of the variables (nodes) of a DAG G is defined as a non-unique permutation π of the nodes: $Y_1 \prec Y_2 \prec \cdots \prec Y_p$, where the relation $k \prec j$ is understood to mean that node k comes before node j (i.e., there is an acyclic route connecting node k and node j). Formally, $\pi_k < \pi_j \iff j \in de(k)$ and $k \in an(j)$, where $de(k)$ are the descendants of the k-th node, and $an(j)$ are the ancestors of the j-th node in the DAG G.

These algorithms decompose the DAG learning problem into two phases: (i) Topological order learning under certain conditions; (ii) Graph estimation depends on the learned topological order via a step-wise selection procedure of ancestor nodes. The following is a brief review of the identifiable conditions:

(1) Linearity with constrained error variances. According to Peters and Bühlmann, 2014, when the observational data are produced using a Gaussian linear SEM that captures the causal linkages and has equal error variances, the causal graph may be distinguished from the joint distribution. In addition, Ghoshal and Honorio, 2018 and Park and Kim, 2020 provide relaxed identifiability conditions with heterogeneous variances requiring an explicit order among the noise variances. In detail:

- *Equal error variance assumption* (Peters and Bühlmann, 2014):
  $\text{Cov}[U] = \text{diag}(\sigma_1^2, ..., \sigma_p^2) = \sigma^2 I$;
- *Bottom-up variance assumption* (Ghoshal and Honorio, 2018):
  the noise variance of the child node (variable) is approximately larger that of its parents (ancestors), $\sigma_j^2 > \sigma_k^2, j = \pi_m \in V, k = an(j)$;
- *Top-down variance assumption* (Park and Kim, 2020):
  the noise variance of the parent node (variable) is approximately lower that of its childs (descendants), $\sigma_j^2 < \sigma_k^2, j = \pi_m \in V, k = de(j)$.

Along these lines, numerous order-based learning techniques are put forth to determine the precise DAG structure (Ghoshal and Honorio, 2018; Yuan et al., 2018; Chen, Drton, and Wang, 2019; Park, 2020; Gao, Ding, and Aragam, 2020).

For example, the top-down algorithm runs as follow. Stage (1) infers the ordering by successively finding sources. We start with the set which contains all nodes, $R = V$ and the empty set, $S = \emptyset$. We iterate over R and S: for each node in R we calculate its conditional (error) variance given all nodes in S. We select the node with the *lowest* variance and append it to the ordering set, S and also remove it from the remaining set, R. With the updated R and S we repeat the process of finding the node with the lowest conditional (error) variance given the nodes in S, append it to the ordering set S and remove it from the remaining nodes in R, and so on until $R = \emptyset$. Lastly, the node ordering in S is returned. Once the ordering is know (estimated), in Stage (2) existing linear (or nonlinear) variable selection methods (glmnet, leaps, L0learn, etc) suffice to learn the parent set $pa(j)$ and hence the DAG G. Limitation of this procedure is that can be challenging to actually confirm assumptions of equal or ordered noise variances.

(2) Linearity with non-Gaussian errors. Recent research has demonstrated that, without requiring any prior information of the network structure, the application of non-Gaussianity may reveal the whole structure of a linear acyclic model, that is, a causal ordering of variables and the strength of their connections. The linear non Gaussian DAG, often referred to as the linear non-gaussian acyclic model (LiNGAM) (Shimizu et al., 2006), relaxes the Gaussianity condition and does not call for an additional constrained noise variance assumption for identifiability. All external unobserved errors, U are continuous random variables with non-Gaussian distributions, zero means, non-zero variances, and are independent of each other such that no hidden confounding factors exist.

As shown by Shimizu et al., 2006, the causal ordering of a linear non-Gaussian DAG may be reconstructed via an iterative search method. Permutation and linear independent component analysis (ICA) are two techniques used in this approach. However, this ICA-LiGAM algorithm has several potential problems. Thus, Shimizu et al., 2011 proposes a novel approach, called directLiNGAM, to estimate a causal ordering of variables, that ensure the validity of DAG identification in the LiNGAM model. The new technique calculates the topological (causal) order of variables by

sequentially computing residual errors from the model's input data. This procedure is carried out with a top-down procedure, starting at root nodes, followed by children of the root nodes and so on until completion. In detail:

- (a) Given the observed data matrix, Y and the order list, $\pi = \emptyset$ perform linear regressions of Y_j on Y_k and compute the residual vectors, $R_j^{(k)} = Y_j - \hat{\beta}_{jk} Y_k$ for all $(j \neq k) \in V/\pi$. Then, the root node, $Y_{(1)}$ in the order list, $\pi = Y_{(1)}$ is the most independent variable over all its residuals using a non-parametric independence (IND) test, $Y_{(1)} = \min(k \in V/\pi) \sum_{j \neq k} \text{IND}(Y_k; R_j^{(k)})$;
- (b) collect the $(p-1)$ residuals of the root node in a new data matrix, $Y := R^{(1)}$, i.e., removing the effect of the root node, perform step (a) on these residuals, and append the new root $Y_{(2)}$ in the order list, $\pi = (Y_{(1)}, Y_{(2)})$;
- (c) repeat (a)-(b) until $R^{(p-1)} = \emptyset$.

To note, non-Gaussian errors are crucial because for a Gaussian random variable, uncorrelated and independent are equivalent, so the residual are always independent of its regressors. Vice versa, when the errors are non-Gaussian, the independence of residuals and regressors can be used to select the root sequence with the independence (IND) measure.
Once the causal ordering between the variables are established, it is simple to estimate the strength of the relationships of a strictly triangular matrix B by following the order in π, using SEM covariance-based procedure such least squares and maximum likelihood approaches, pruning the non-significant ($P < 0.05$) β coefficients, or via a nodewise-based model selection procedure of ancestor nodes.

(3) Non-linearity with additive errors. Non-linear transformation is frequently used in data generation in practice, hence it should be considered as alternative to linear models. A functional causal model, called additive noise model (ANM), depicts the causal effect on each Y_j as a function of the direct causes $Y_{pa(j)}$ and some addittive unmeasurable noise, U_j (Peters and Bühlmann, 2014):

$$Y_j = f_j(Y_{pa(j)}) + U_j, \quad j \in V \tag{5.4}$$

where $U_j (j = 1, ..., p)$ are (mutually) independent with Gaussian distribution, i.e., there are no hidden variables. Generally, the function, f is a suitably restricted functional class and describes how the outcome, Y is produced from its causes, X supposed independent with noise errors, $X \perp\!\!\!\perp U$. Because the independence constraint between the noise and cause, only holds for the correct causal direction and is broken for the incorrect direction, the non-linear functional classes make it possible to identify the causal order and direction between X and Y.

The authors therefore propose a two-phase iterative procedure, called RESIT (Regression with Subsequent Independence Test). First phase, yields a topological ordering or a fully connected DAG. Second phase, visits every node and eliminates incoming edges in the full DAG until the residuals are not independent anymore. RESIT not scale well in high dimensional data, and the order in which the residual independence tests are performed may lead to different results.

Thus, Bühlmann, Peters, and Ernest, 2014 develop estimation for potentially high-dimensional on a special (and more practical) ANN, or functional SEM, with (mutually) independent and potentially misspecified Guassian errors, called Causal Additive Model (CAM):

$$Y_j = \sum_{k \in pa(j)} f_{j,k}(Y_k) + U_j, \quad j \in V \tag{5.5}$$

An important results is that if all functions $f_{j,k}(.)$ are nonlinear, the underlying DAG structure is identifiable from the observational distribution, $P(Y)$. An efficient order-based algorithm which can deal with many variables consists of three phases (stages) is proposed by Bühlmann, Peters, and Ernest, 2014. In detail:

- *Preliminary neighborhood selection*. Fit an additive model for each variable, Y_j on all other variables, $Y_{\{-j\}}$ for estimating a superset of the skeleton of the underlying DAG with K (usually $K < 10$) "possible" parents of Y_j using a boosting procedure for additive model implemented in the `gamboost()` R-function of the package `mboost`;

- *Estimating the topological order by greedy search*. Order search for the variables that starts with an empty order and iteratively adds edges between nodes that corresponds to the largest gain in negative log-likelihood score:

$$S(G^\pi; Y) := \sum_{j=1}^{p} \log||Y_j^\pi - \sum_{k=1}^{j-1} \hat{f}_{j,k}^\pi(Y_k^\pi)||_2^2 \tag{5.6}$$

  until no more edges can be added without decreasing the score $S(G^\pi; Y)$, where G^π is the full connected DAG, in which each ordered node k has an directed arrow to all j if $k \prec j$. The possible permutations, $\pi(R)$ are "restricted" taking into account that edges are compatible with the preliminary neighborhood selection sets. After $p(p-1)/2$ interactions the graph is completed to a full connected DAG, corresponding to the best restricted permutation, $\hat{\pi}(R)$ for the indices of the variables. The score is based on a generalized additive model with penalized regression splines (with 10 basis functions per variable) using the `gam()` R-function from the package `mgcv`;

- *DAG pruning by feature selection*. For pruning the full DAG, nodewise additive models can be used by applying significance testing on the covariate functions, usually with a P-value < 0.001, or with penalized additive models excluding expected non-parent variables if $\hat{f}_{j,k} = 0$.

Limitation of ANM and CAM is that in the absence of detailed knowledge about the data generating mechanism, the assumed model must be capable of capturing complex nonlinear relationships with respect to the linear model. The findings of causal discovery could be misleading if the expected functional links are too constrained to be able to match the real process of data generation.

Gradient-based methods. Aciclicity is the most common assumption in causal discovery and score-based methods uses heuristic greedy algorithms for solving non-convex optimization without feedback loops, i.e., a combinatorial problem that scales super-exponentially with the number of variables. Recent work called NOTEARS (Zheng et al., 2018) provide a new algorithmic framework for score-based learning of DAG models. The procedure is based on a new algebraic characterisation of acyclicity constraint, which recasts the score-based optimization issue as a continuous problem rather than using the conventional combinatorial technique.

In the linear situation, the matrix $B \in \mathbb{R}^{p \times p}$ properly encodes the graph G, i.e., an edge $j \leftarrow k$ in G is present if and only if $\beta_{kj} \neq 0$. The entire problem may be

expressed in terms of B. Given a score function, $S(B;Y)$ the solution of B is defined by optimizing $S(B;Y)$ subject to the continuous constraint, $h(B) = 0$: :

$$\arg\min_{B \in \mathbb{R}^{p \times p}} S(B;Y) \quad s.t. \;\; h(B) = 0 \tag{5.7}$$

where h is a non-negative non-convex differentiable function used to enforce acyclicity in the estimated graph. Some possible score functions include:

- *Least squares-EV*: $\sum_{j=1}^{p} ||Y_j - \sum \beta_{jk} Y_k||_2^2$ for linear SEM with equal error variances (Peters and Bühlmann, 2014);
- *Negative log-likelihood-EV*: $\frac{p}{2} \log \sum_{j=1}^{p} ||Y_j - \sum \beta_{jk} Y_k||_2^2$ for linear SEM with Gaussian equal error variances (Ng, Ghassami, and Zhang, 2020);
- *Negative log-likelihood-NV*: $\frac{1}{2} \sum_{j=1}^{p} \log ||Y_j - \sum \beta_{jk} Y_k||_2^2$ for linear SEM with Gaussian not-equal errors variances (Ng, Ghassami, and Zhang, 2020).

The function h quantify the "DAG-ness" of the graph, and now literature contains many different proposals:

- *The NOTEARS condition* (Zheng et al., 2018). The first differentiable aciclicity characterization of DAG: $h(B) = \text{tr}[\exp(B \circ B)] - p$;
- *A polynomial condition* (Yu et al., 2019). Proposed to ease the coding effort as the matrix exponential may not be available in all deep learning platforms: $h(B) = \text{tr}[I - (B \circ B)/p]^p - p$;
- *The DAGMA condition* (Bello, Aragam, and Ravikumar, 2022). For a non-negative matrix with spectral radius less than one that has better gradients and run faster than exp and poly conditions: $h(B) = -\log \det[I - (B \circ B)]$.

Where $\circ$ denotes the Hadamard product, $[B \circ B]_{jk} = \beta_{jk}^2$. Usually, the score function includes a sparsity (regolarized) L1-penalty, followed by a thresholding step of the estimated weighted adjacency matrix using a relatively large cut-off of 0.3.

Continuous optimization methods are pervasive in the field of deep learning, whereby highly parameterized networks are optimized using variations on well-studied gradient-based solvers (Goodfellow, Bengio, and Courville, 2016). In general, these methods are more global than other approximate greedy or 2-3 stages methods. This is because they update all edges at each step based on the gradient of the score and as well as based on the acyclicity constraint, and usually have a faster training time as optimization run is known to be highly parallelizable on GPU.

This has resulted in the confluence of black-box deep learning approaches, and causal structure discovery based on non-linear SEM with Gaussian errors in Equation (4), i.e., NOTEARS-MLP, GraNDAG, DAG-GNN, MCSL, and many others proposal can be found in the recent review (Vowels, Camgoz, and Bowden, 2022). `gCastle` Python package (Zhang et al., 2021) includes many development gradient-based methods with optional GPU acceleration. Ng, Huang, and Zhang, 2023 investigates cases of poor performance of structure learning with continuous optimization.

Table 5.2 provides a summary of the structure learning methods in terms of type of algorithm employed, category and output with main papers for reference. Besides the type of algorithm, these methods differ in three main aspects: (i) the input requirements; (ii) the category; (iii) the output. All the methods require as input a

data matrix $n \times p$ where n is the number of subjects and p is the number of genes, with the exception of `SEMdag()` that requires also a graph object. The latter can be derived from existing knowledge or can be an empty graph object if the user decide to implement a full data-driven procedure. Each method represent a different category, in order to provide a comprehensive overview of existing structure learning approaches. Then, PC, GES, and ARGES give as output a CPDAG while the others are able to recover a DAG object. The goal is to find a structure learning method that provide an optimal solution while controlling the computing time of the algorithm.

TABLE 5.2: Overview of the considered structure learning methods.

Method [REF]	R package	Algorithm	Category	Output
PC [Spirtes et al. (2000)]	pcalg	Peter & Clark algorithm	Constraint	CPDAG
GES [Chickering (2003)]	pcalg	Greedy Equivalence Search	Score	CPDAG
ARGES [Nandy et al. (2018)]	pcalg	Adaptively Restricted GES	Hybrid	CPDAG
LiNGAM [Shimizu (2014)]	CausalXtreme	Top-down order search	Order	DAG
CAM [Bühlmann et al. (2014)]	CAM	Greedy order search	Order	DAG
NOTEARS [Zheng et al. (2018)]	gnlearn	NOTEARS (linear) algorithm	Gradient	DAG
SEMdag [Grassi et al. (2022)]	SEMgraph	Knowledge-based ordering (TO/TL)	Order	DAG
		Bottom-up ordering (TO/TL)	Order	DAG

5.2.3 SEMdag algorithm

Our `SEMdag()` function uses a two-stage order-based search with prior knowledge-based or data-driven approach, under the assumption that a linear SEM with equal variance error terms is assumed (Peters and Bühlmann, 2014). After determining the vertex (node) or level (layer) order of nodes in stage (1), the DAG may be trained using penalized (L1) regressions in stage (2) (Shojaie and Michailidis, 2010).

Learning ordering

The estimated linear order is determined via a prior graph topological vertex (TO) or level (TL) ordering, or by using a data-driven node or level bottom-up (BU) procedure. In detail:

Knowledge-based ordering. Topological sorting or ordering of a directed graph's vertices is only feasible if and only if the knowledge-based graph is a directed acyclic graph, which means we must convert the graph in a DAG. At least one topological ordering exists in every DAG. For DAGs, topological vertex sorting is a linear ordering of the vertices such that vertex u occurs before vertex v for each directed edge $u \rightarrow v$. We can construct a topological sort with running time linear to the number of vertices plus the number of edges, i.e., $O(V + E)$. Examples are the Kahn's algorithm or the Depth First Search algorithm. However, there can be more than one

topological sorting for a DAG. To overcome this issue, we consider DAG topological layer (level) sorting.

Given a DAG G, define a collection of sets as follows (cf.Gao, Ding, and Aragam, 2020): L_0, denotes the set of root (source) nodes in the top layer, $L_j = \cup_{m=0}^{j} L_m$ and for $j > 0, L_j$ is the set of all source nodes in the subgraph $G[V - L_{j-1}]$ formed by removing the nodes in L_{j-1}. So, e.g., L_1 is the set of source nodes in $G - L_0$. This decomposes G into d layers, $L(G) := (L_1; ...; L_d)$ where each layer L_j consists of nodes that are sources in the subgraph $G[V - L_{j-1}]$, and L_j is an ancestral set for each j. The number d of "layers" denotes the longest possible distance from some nodes in the DAG to a root node and measures the "depth of a DAG. See Figure 5.1 for an illustration.

The idea of a topological layer enables us to transform a DAG into a distinct and unique topological structure with d levels, where a node's parents must be located in the node's upper layers and acyclicity is thus naturally ensured (Jothi et al., 2009). In particular, given a DAG G, we derive the topological structure of the DAG by allocating each node to a single layer depending on its longest distance to one of the leaf nodes. Starting with a DAG, we first create DAG^T, the transpose of the DAG, by flipping the orientation of the DAG's edges. Next, a linear hierarchical ordering of the nodes for each network is obtained by applying the iterative leaf-removal procedure to DAG and DAG^T. The leaf-removal method is a bottom-up iterative process that eliminates all leaf nodes from the network as well as the edges that incident on them with each iteration. Leaf nodes are those that have no outgoing edges. The algorithm ends when the network is entirely deconstructed. The topological ordering of the network's nodes is ultimately determined by reversing the bottom-up ordering of nodes in DAG^T (which is identical to the top-down ordering of nodes in DAG) and combining it with the bottom-up ordering of nodes in DAG. Vertex sort gives a linear ordering of nodes that encompasses all feasible solutions rather than just one by utilizing DAG and DAG^T. We refer the reader to Figure 5.1 for more details about the leaf-removal algorithm.

Any node, $j \in L_j$ have some parents in the previous layer, L_{j-1} and some childs in the next layer, L_{j+1}. Learning G is equivalent to learning the sets $L(G) = (L_1; ...; L_d)$, since any topological sort π of G can be determined from $L(G)$, and from any sort π, the graph G can be recovered via variable selection. Unlike a topological sort of G, which may not be unique, the layer decomposition $L(G)$ is always unique.

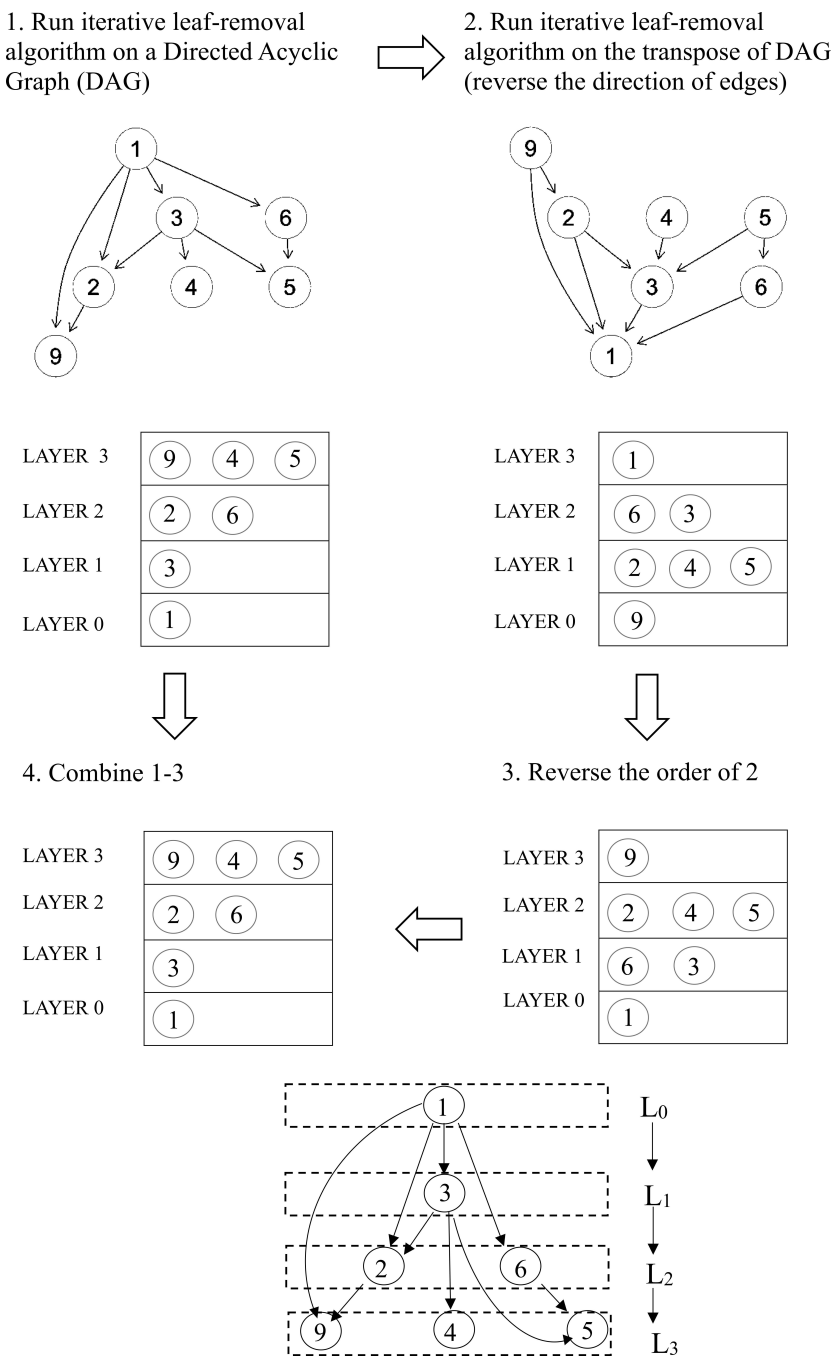

FIGURE 5.1: Application of the leaf-removal algorithm on the network of interest (1) and its transpose (2) to determine the linear ordering of the network's nodes by the combination (4) of the node structure in step (1) and (3).

Bottom-up ordering. The proposed algorithm for learning DAG sworks by constructing the DAG in a bottom-up fashion as in Ghoshal and Honorio, 2018, estimating with a backward procedure the inverse covariance matrix, $\hat{\Omega} = \hat{\Sigma}^{-1}$ of the sample covariance (correlation) matrix, $S := (Y^T Y)/n$ using the graphical lasso algorithm (Friedman, Hastie, and Tibshirani, 2008a):

$$\hat{\Omega} \in \underset{\Omega \succeq 0}{\arg\min} \ \operatorname{tr}(\Omega S) - \log\det(\Omega) + \lambda \sum_{j \neq k} |\omega_{jk}| \tag{5.8}$$

and define step by step a *reversed* causal ordering recovering the minimum precision: $\text{var}(Y_j|Y_{\{-j\}})^{-1}$, i.e., the maximum full conditional variance, from its diagonal elements. To note, if $\lambda \to 0$ and $n >> p$, then the Maximum Likelihood Estimate (MLE) is given as: $\hat{\Omega} = S^{-1}$.

In detail, each element of the ordering is approximated through the following steps:

- Start with the empty set, $P = \emptyset$ and select the node with the *highest* full conditional variance as terminal vertex, Y_p, i.e. the vertex with minimum value in the diagonal values of the precision matrix, $\hat{\omega} = \min(\text{diag}(\hat{\Omega}))$ or the terminal layer (> 1 vertices, L_d) with $\hat{\omega} \in \min(\text{diag}(\hat{\Omega})) + \eta$. The latter means that all nodes with a maximum distance of η from the precision value of the terminal vertex's can be combined to determine the terminal layer rather than just one terminal vertex.
- Append Y_p or L_d to the ordering set, P and also remove the selected node(s) from the column(s) of the data matrix. With the updated data matrix repeat the process of estimating the precision matrix and identifying the vertex (or vertices) with the lowest precision(s);
- and so on until the source node, Y_1 or the top layer, L_0 is found.

Lastly, the reverse of the node (or level) ordering in the set P is returned. The glasso procedure is run using the penalized parameter, $\lambda = 0.001$ or $\lambda = \sqrt{log(p)/n}$ for low ($n > p$) or high ($n < p$) dimensional data, respectively.

Learning parents

Finding the topological vertex (node) or level (layer) ordering, as stated by Shojaie and Michailidis, 2010 the challenge of estimating the DAG structure (edge set) may be viewed in terms of penalized likelihood. Assuming fixed the node or layer ordering from stage (1): $Y_1 \prec Y_2 \prec \cdots \prec Y_p$, or $L_0 \prec L_1 \prec \cdots \prec L_d$, the stage (2) executes parent estimations by doing LASSO (Least Absolute Shrinkage and Selection Operator) regressions of the j-th outcome variable on the predictor (ancestor) variables, $S_j := \left\{Y_k : Y_k \prec Y_j\right\}$ or $S_j := \left\{Y_k : L_k \prec L_j\right\}$ in the vertex or level order list:

$$\hat{\beta} \in \underset{\beta \in \mathbb{R}^{k \prec j}}{\arg\min} \; ||Y_j - \sum_{k \prec j} \beta_{jk} Y_k||_2^2 + \lambda_j \sum_{k \prec j} w_{jk}|\beta_{jk}| \tag{5.9}$$

It is possible to estimate the DAG adjacency matrix, $\hat{A}$ removing nodewise the beta coefficients equal zero ($A_{jk} = 0$ if $\hat{\beta}_{jk} = 0$ and 1 otherwise) or using a threshold on the beta absolute values. To allow for differential shrinkage, various penalty factors w_{jk} might be given to each beta coefficient. There is no shrinkage if $w_{jk} = 0$ for some variables, and those variables are always included in the chosen model. If the input graph is known (knowledge-based approach), weights can be based on the graph edges: 0 (i.e., edge present) and 1 (i.e., edge absent).

The λ_j parameter for each outcome variable in LASSO regression is chosen by tuning a vector of λ values, or by cross-validation ($p \leq 100$) or BIC-based ($p > 100$) lambdas selection. To further improve efficiency, some tuning-free schemes (such as λ = (N(0,1)-quantile at $\alpha/[2p(j-1)])/\sqrt{n}$, suggested in Shojaie and Michailidis, 2010, or $\lambda = \sqrt{log(p)/n}$, suggested in Janková and Geer, 2015 for graphical lasso) can also be enabled.

User interface

The example code of the function `SEMdag()` is as follows.

```
SEMdag(graph, data, LO = "TO", beta = 0, eta = NULL,
       lambdas = NA, penalty = TRUE, verbose = FALSE, ...)
```

The inputs are: an igraph object (*graph*) that can be a priori graph topological order or a graph with no edges (data-driven procedure: note that in this case it can be created with the function `make_empty_graph()` of the igraph package, specificying the number of nodes n as input); a matrix with rows corresponding to subjects and columns to graph nodes (*data*); the linear order method (*LO*, default = "TO"); the minimum absolute LASSO beta coefficient for a new direct link to be retained in the final model (*beta*, default = 0); the minimum fixed eta threshold for bottom-up search of vertex (*eta*, default = 0.05); a vector of regularization LASSO lambda values (*lambdas*, default = NA); penalty factors for differential shrinkage (*penalty*, default = TRUE).

Using a two-step order search methodology, the recovered DAG is approximated. Following the determination of the vertex (node) or level (layer) order of p nodes run the `glasso()` function of the **glasso** R package (Friedman, Hastie, and Tibshirani, 2019), in step 1) the DAG may be trained using penalized (L1) regressions with the `glmnet()` function of the **glmnet** R package (Friedman et al., 2023), in step 2).

When choosing between node or layer ordering, the user has to keep in mind the reduced computational burden in the layer-based approached compared to the node one. In detail, in step 1), the layer approach has to identify the order of $d + 1$ layers, where d represents the "depth" of the DAG, instead the node approach needs to find the order of $p + 1$ nodes, where p is the number of nodes in the DAG. As a result, an high dimensional graph could impact the computation time of the latter step in the nodewise approach. Same consideration could be done for the step 2) where the number of L1 regressions in the nodewise approach is equal to $(p - 1)$; instead, for the layer-based one, the number of regressions is equal to p - (number of layers), a smaller set compared to the latter.

The output of `SEMgsa()` is represented by a list containing four objects: *dag*, the estimated DAG; *dag.new*, new estimated connections; *dag.old*, connections preserved from the input graph; *LO*, the estimated vertex ordering.

To read more about `SEMdag()` function, in terms of description, usage, function arguments and value, see help documentation: `?SEMdag` or refer to
https://rdrr.io/cran/SEMgraph/man/SEMdag.html.

5.3 Experimental design

5.3.1 Benchmark data

For each specific disease, two different datasets have been selected: one for the training process and the other for testing the proposed modelling scheme. Before selecting the data, we've checked that each pair of datasets had the same study type (expression profiling by high throughput sequencing, i.e. RNA-seq data), the same platform and a similar number of subjects. This selection procedure resulted in 4 x 2 datasets as shown in Table 5.3.

TABLE 5.3: Description of the selected training/testing datasets for each disease.

Data (Type)	Split	GSE	n	case	control	p	KEGG pathway	vcount	ecount
ALS (RNA-seq)	Train	GSE124439	160	139	21	100	Amyotrophic lateral sclerosis	190	261
	Test	GSE153960	273	206	67	100			
BRCA (RNA-seq)	Train	TCGA	224	112	112	100	Breast cancer	133	483
	Test	GSE81538 + GSE205725	377	190	187	100			
COVID-19 (RNA-seq)	Train	GSE157103	126	100	26	100	Coronavirus disease - COVID-19	54	83
	Test	GSE152641	86	62	24	100			
STEMI (RNA-seq)	Train	GSE59867	98	84	14	100	Lipid and atherosclerosis	191	420
	Test	GSE62646	88	65	23	100			

Amyotrophic Lateral Sclerosis (ALS). A rare kind of neurodegenerative illness called amyotrophic lateral sclerosis (ALS) causes the gradual loss of motor neurons that regulate voluntary muscles. For training, we selected postmortem cortex ALS RNA-seq expression data (GSE124439) from Cooper-Knock et al., 2015 with 139 ALS cases and 21 healthy controls. For testing, postmortem cortex RNA-seq data from the NYGC ALS Consortium (GSE153960) were selected, with 206 ALS cases and 67 controls. Network information has been extracted from the KEGG pathway "Amyotrophic lateral sclerosis", consisting of 364 nodes and 333 edges. For computational purposes, the largest connected component has been retained, corresponding to 190 nodes and 261 edges.

Breast Cancer (BRCA). Breast cancer develops when cells in the breasts multiply and expand out of control, resulting in a mass of tissue known as a tumor. For training, we make use of the (pre-processed) breast cancer RNA-seq dataset from TCGA project, which has $n = 224$ human samples, comprising 112 BRCA samples and 112 control samples. For testing, two RNA-seq datasets were combined: GSE81538 (Brueffer et al., 2018) for 190 breast cancer cases and GSE205725 (German et al., 2023) for 187 healthy controls. Network information has been extracted from the KEGG pathway "Breast cancer", consisting of 147 nodes and 488 edges. For computational purposes, the largest connected component has been retained, corresponding to 133 nodes and 483 edges.

Coronavirus disease (COVID-19). The severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) is the cause of the respiratory infection known as coronavirus disease of 2019 (COVID-19), which is extremely contagious. RNA-seq data from Overmyer et al., 2021 (GSE157103) were considered for training, with a total of $n = 126$ samples with 100 COVID-19 patients and 26 non-COVID-19. Conversely, RNA-seq data (GSE152641, Thair et al., 2020) from whole blood of 62 COVID-19 patients and 24 healthy controls was considered for testing. Network information has been retrieved from the KEGG pathway "Coronavirus disease - COVID-19", consisting of 232 nodes and 208 edges. For computational purposes, the largest connected component has been retained, corresponding to 54 nodes and 83 edges.

ST-elevation myocardial infarction (STEMI). A heart attack known as a STEMI, happens when an elevation in the ST segment, often results in myocardial injury or necrosis. As training data, we made use of the RNA-seq dataset (GSE59867) from Maciejak et al., 2015 that reports a total of 98 subjects, among which 84 are cases and 14 are healthy controls. As testing data, we selected the RNA-seq dataset (GSE62646) from Kiliszek et al., 2012, where 65 subjects were cases and 23 controls. Network information has been extracted from the KEGG pathway "Lipid and atherosclerosis" (a pathway associated with myocardial disease) consisting of 215 nodes and 428 edges. For computational purposes, the largest connected component has been retained, corresponding to 191 nodes and 420 edges.

5.3.2 DAG structure recovery

The causal DAG discovery procedure implemented in this analysis is visually summarised in the first two boxes of Figure 5.3 and is better explained in this section.

- *Data filtering (gene extraction).* To reduce the computational burden of structure discovery methods, genes of the data matrix have been filtered according to Differential Expression Analysis (DEA). In detail linear models for DEA were fitted with the **limma** R package (Smyth, 2005) and p-values were adjusted for multiple testing using the method of Benjamini-Hochberg (Benjamini and Hochberg, 1995). In this way, the $p = 100$ most significant DEGs were filtered out for each dataset, implementing a fully data-driven procedure for causal structure discovery. The differential expression patterns for each pair of datasets of each specific disease is shown in Figure 5.2. Each pair of datasets share similar differential expression structure, highlighting same biological differences between healthy and diseased states. As a result, the model fitted on the training (learning) data should be well generalizable to the testing (validation) data.

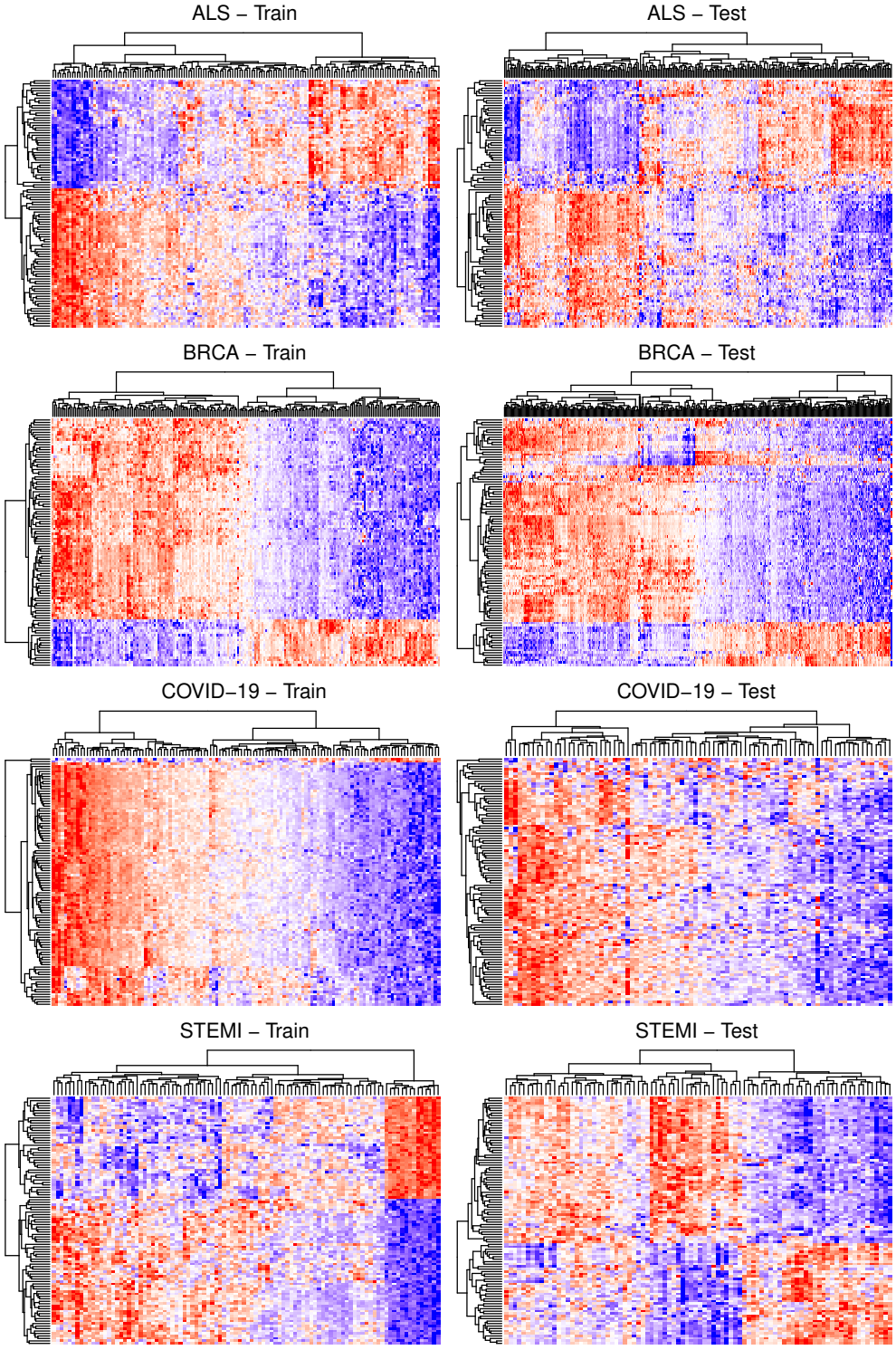

FIGURE 5.2: Heatmap of differentially expressed genes (DEGs) for each train/test dataset of each specific disease. The heatmap illustrates expression levels for all DEGs, where red indicates high expression and blue indicates low expression.

- *DAG/CPDAG structure recovery*. The considered methods recover the DAG structure in three different formats:

  (i) adjacency matrix: the functions `pc()` and `ges()` from **pcalg** R package (Kalisch and Bühlmann, 2007) estimate the connectivity matrix of a DAG specifying, as input, various possible methods (PC, GES and ARGES) have been selected. On the other side, the **causalXtreme** R package (Gnecco et al., 2021) provides wrapper functions for fitting the DirectLINGAM algorithm and obtaining an adjacency matrix output. In the end, an igraph object has been obtained from the `graph_from_adjacency_matrix()` function of the **igraph** package (Csardi and Nepusz, 2006);

  (ii) edgelist: the function `CAM()` from the **CAM** R package (Bühlmann, Peters, and Ernest, 2014) estimates the edge list of a DAG using the CAM algorithm. From the edgelist output, an igraph object has been obtained from the `graph_from_edgelist()` function of the **igraph** package;

  (iii) graph: the function `notears()` from the **gnlearn** R package (Lebrón and Varando, 2021) estimates the DAG structure as an igraph object, without the need for further refinements; the function `SEMdag()` from the **SEMgraph** package (Grassi, Palluzzi, and Tarantino, 2022) using as input gives as output the igraph object of interest.

All the methods require as input the data matrix, with the exception of `SEMdag()` that also requires a graph object. So, an empty graph with a number of genes equal to the number of selected DEGs ($p = 100$) is generated and the data-driven bottom-up (BU) search of vertex (or layer) order is performed using the vertices of the empty graph.

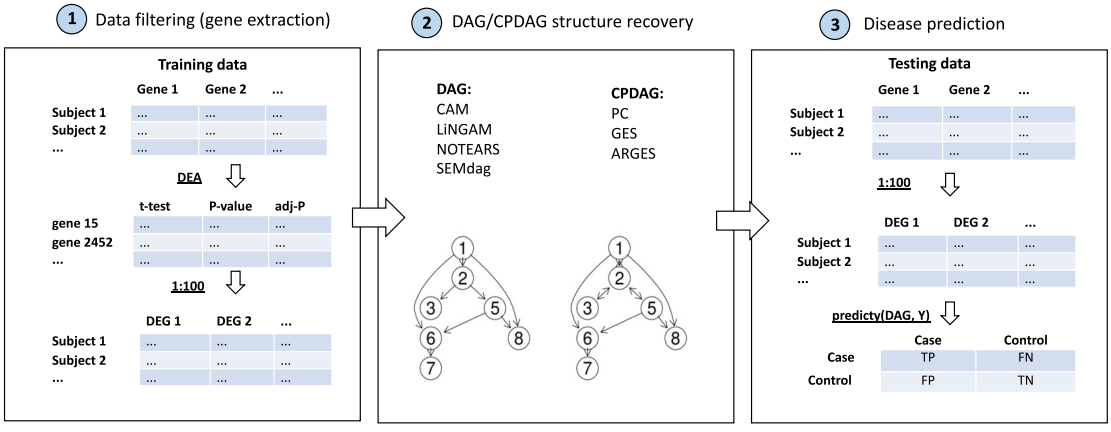

FIGURE 5.3: Experimental design scheme.

5.3.3 Disease prediction

The last step of the experimental design scheme is summarised in the last box of Figure 5.3. Our aim is to make out-of-sample predictions with a SEM-based predictive procedure (Rooij et al., 2022) taking into account the recovered graph structure from the causal discovery algorithms. Additionally, we consider the Random Forest (RF) algorithm (ref), a popular supervised machine learning method for disease prediction, that requires only a data matrix as input and is used for benchmark comparison. RF is performed with the `rfCMA()` function of the **CMA** R package (ref). In this way, comparisons are made with the most performed supervised learning algorithm and within different causal recovery methods. We briefly describe the SEM-based predictive procedure in the next subsection.

SEM based out-of-sample predictions. Suppose that the variables can be divided into two sets: one set of predictor variables and the other set of response or outcome variables. In SEM, it is assumed that the variables have a joint multivariate normal distribution where the predictor variables have mean μ_x and the response variables μ_y. The covariance matrix of predictors and responses (Σ) can be expressed as follows:

$$\Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix}$$

where Σ_{yy} is the $R \times R$ covariance matrix of the responses, Σ_{xx} is the $P \times P$ covariance matrix of the predictors, and Σ_{xy} contains the covariances between predictors and responses, and is of size $P \times R$. In detail:

- P = number of DEGs. Specifically, if the prediction is a continuous variable representing sink nodes, P = number of source nodes and mediators otherwise, if the prediction is a binary group variable, P = number of source, mediators and sink.
- R = number of sink nodes or a binary group variable. Specifically, in the latter, the prediction can be a binary vector with 1 for cases and 0 for controls.

Suppose we have fitted our SEM (i.e. the recovery graph from a causal discovery procedure) to an empirical data set to obtain the estimated parameter vector ($\hat{\theta}$) including means, and regression weights. We can extract from the fitted model the estimated model-implied mean vector ($\mu(\hat{\theta})$) and covariance matrix ($\Sigma(\hat{\theta})$). The predictive distribution, a model-implied conditional distribution of the responses given the predictors, can be used to make predictions. Suppose we have a new observation with predictor values of x_0. The predictive distribution for a fitted SEM is a (multivariate) normal distribution with the mean vector ($\mu_{y|x_0}(\hat{\theta})$) and covariance matrix ($\Sigma_{y|x_0}(\hat{\theta})$) as shown below:

$$\begin{aligned} \hat{\mu}_{y|x_0} &= \hat{\mu}_y + \hat{\Sigma}_{xy}^T \hat{\Sigma}_{xx}^{-1}(x_0 - \hat{\mu}_x) \\ \hat{\Sigma}_{y|x_0} &= \hat{\Sigma}_{yy} - \hat{\Sigma}_{xy}^T \hat{\Sigma}_{xx}^{-1} \hat{\Sigma}_{xy} \end{aligned}$$

The mean of the predictive distribution provides the most common method for obtaining a point prediction. The SEM-based prediction rule for a normal distribution is $\hat{y} = \hat{\mu}_{y|x_0}$, which can be expressed as:

$$\hat{y} = \hat{\mu}_y + \hat{\Gamma}(x_0 - \hat{\mu}_x) = \hat{\mu}_y - \hat{\Gamma}\hat{\mu}_x + \hat{\Gamma}x_0 = \hat{\alpha} + \hat{\Gamma}x_0$$

where $\hat{\Gamma} = \hat{\Sigma}_{xy}^T \hat{\Sigma}_{xx}^{-1}$. The key distinction is that, as opposed to conventional (least squares) regression, the intercept and regression weights are estimated differently by taking into consideration the structure defined in the SEM. The scores on the predictors using the model estimates from the current sample, will be used to predict the score of the output variables for new cases.

If the outcome variable is a binary group, the corresponding output value is a real value, but can be properly converted in a discrete score by choosing a suitable threshold value. The cut-off point is calculated as a half-sum of the mean of the two groups (cases and controls). In the end, the samples will be categorized as cases or controls. Predictions have been made via the function `predicty()` of the **SEMgraph** package. For more information, see the help documentation: `?predicty`.

5.3.4 Evaluation metrics

The aforementioned structure recovery methods have been evaluated with the following metrics:

- MultiDimensional Scaling (MDS): Once obtained the estimated graph structures from each method, the Structural Hamming Distance (SHD) has been computed to generate a measure of structural similarity between graphs by comparing their adjacency matrices. This might be interpreted as how many addition/deletion operations are necessary to transform the edge set of G1 into that of G2. To obtain a distance measure between 0 and 1, the measurement was related to the size (number of nodes) of each graph; the higher the number, the more distant the objects. Then, a visual representation (MDS) of distances between the obtained SHDs has been generated to identify more or less similar structures (respectively, objects with shorter or longer distances) via the `cmdscale()` function of the **stats** R package. The graph objects have been divided into k clusters, with each observation belonging to the cluster with the closest mean, using the K-means algorithm. The number of cluster (K) is selected via hierarchical clustering (`hclust()` function of the **stats** R package, with complete linkage method as default). After plotting the dendogram, the optimal height for cutting the tree has been chosen to be the one that better reflects the more distant clusters of objects, joining together the ones with really low SHD values.

- Matthews correlation coefficient (MCC): Out-of-sample predictions for each disease prediction method, categorized as positive and negative cases, have been obtained for testing datasets and compared with ground truth. The confusion matrix, also known as the error or contingency matrix, has been used to assess the diagnostic capacity of classifiers. True positives (TP) and true negatives (TN) are the positive and negative cases that have been correctly identified by the classifier. False positives (FP) are cases where the classifier mistakenly classified a negative as positive, and false negatives (FN) are situations when the classifier mistakenly classified a positive as negative. In binary classification tasks, accuracy and F1 score calculated using confusion matrices continue to be among the most often used measures. However, on unbalanced datasets, these statistical techniques can dangerously show inflated and too

optimistic outcomes. Alternatively, a more faithful statistical rate is the MCC, which yields a high score only when the prediction performed well in each of the four confusion matrix categories (TP, FN, TN, FP), proportionately to the size of the dataset's positive and negative elements (see Matthews, 1975; Baldi et al., 2000; Chicco and Jurman, 2020 for reference). As a result, DAG structure recovery methods and RF algorithm have been compared with each other using MCC. To note that a MCC = -1 denotes complete disagreement between the prediction and the observation, C = 0 is for a prediction that is no better than random, and C = 1 shows perfect agreement.

To note that, regarding the procedure with `SEMdag()` function, all four causal structure recovery strategies have been implemented: i) Knowledge-based ordering (TO/TL) based on the KEGG pathway of the disease of interest, i.e. a biologically validated network structure: SEMdag_KB_TO and SEMdag_KB_TL; ii) Data-driven Bottom-Up ordering (TO/TL) based on the empty graph with $p = 100$ nodes (DEGs): SEMdag_BU_TO and SEMdag_BU_TL.

5.4 Results

DAG/CPDAG. Table 5.4 and Table 5.5 report a descriptive analysis of the recovered graph structures in terms of graph dimension (vertex and edges), number of source and target nodes and measures of centrality as degree and betweenness.
In terms of node dimension, we have the same number of 100 DEGs for almost all methods except for SEMdag_KB. In the latter case, the node dimension depend on the largest component of the KEGG pathway of reference, matched with the nodes in the data. The largest DAGs, with the higher number of nodes together with the most dense structure of connections, are the SEMdag_KB_TL and SEMdag_KB_TO of the ALS and STEMI dataset, where the starting graphs are the largest ones compared to BRCA and COVID-19. After these two methods, the most densely connected graphs are the ones of LINGAM. Lower density graphs are reported by ARGES, GES and PC methods.
Moreover, given the source-sink prediction structure explained in Section 5.3.3, it is interesting to understand the number of source and sink nodes reported by each causal discovery method. The highest number of source-sink nodes is reported by the PC method for all datasets together with the ARGES method for the COVID-19 and STEMI datasets. Also SEMdag_TL reports an high number of source nodes. Conversely, the lowest number of source-sink nodes is shown by GES.
Degree centrality instead involves counting the number of direct connections a node has; as a result, if high, there is an high number of nodes with high degree (hub nodes). It is interesting to note that the higher mean degree is shown by `SEMdag()` methods and the lower one by ARGES, GES, PC and NOTEARS.
Betweenness centrality instead involves calculating how often a node occurs on all shortest paths between other pair of nodes. Thus, high betweenness indicate that the structure is characterised by vertices with high influence over the network. Higher betweenness values can be highlighted for CAM and GES for all datasets, and the lower ones for PC.

TABLE 5.4: Descriptive table of recovered DAG/CPDAG structures for each method, divided by ALS and BRCA data. V counts the numberf of Vertices in the network and E the number of Edges; S reports the number of Source nodes and T the number of Targer nodes; D stands for mean(Degree) and B for mean(Betweenness)

	ALS			BRCA		
method	G(V,E)	G(S,T)	G(D,B)	G(V,E)	G(S,T)	G(D,B)
ARGES	G(100,125)	G(3,22)	G(2,58)	G(100,175)	G(1,16)	G(4,124)
CAM	G(100,276)	G(1,15)	G(6,91)	G(100,335)	G(1,9)	G(7,105)
GES	G(100,148)	G(1,5)	G(3,75)	G(100,197)	G(1,5)	G(4,152)
LiNGAM	G(100,436)	G(2,19)	G(9,50)	G(100,471)	G(2,15)	G(9,47)
NOTEARS	G(100,198)	G(6,25)	G(4,58)	G(100,186)	G(6,25)	G(4,72)
PC	G(100,148)	G(18,23)	G(3,13)	G(100,175)	G(25,26)	G(4,8)
SEMdag_BU_TL	G(100,580)	G(2,19)	G(12,14)	G(100,666)	G(37,13)	G(13,13)
SEMdag_BU_TO	G(100,591)	G(1,8)	G(12,56)	G(100,727)	G(6,15)	G(15,54)
SEMdag_KB_TL	G(168,1162)	G(12,129)	G(14,18)	G(107,865)	G(28,16)	G(16,30)
SEMdag_KB_TO	G(168,1077)	G(4,21)	G(13,117)	G(107,804)	G(1,11)	G(15,52)

TABLE 5.5: Descriptive table of recovered DAG/CPDAG structures for each method, divided by COVID-19 and STEMI data. V counts the numberf of Vertices in the network and E the number of Edges; S reports the number of Source nodes and T the number of Targer nodes; D stands for mean(Degree) and B for mean(Betweenness)

	COVID-19			STEMI		
method	G(V,E)	G(S,T)	G(D,B)	G(V,E)	G(S,T)	G(D,B)
ARGES	G(100,38)	G(62,62)	G(1,0)	G(100,73)	G(27,34)	G(1,4)
CAM	G(100,99)	G(1,43)	G(2,48)	G(100,189)	G(1,17)	G(4,160)
GES	G(100,50)	G(50,50)	G(1,0)	G(100,100)	G(1,1)	G(2,488)
LiNGAM	G(100,470)	G(2,13)	G(9,55)	G(100,407)	G(2,19)	G(8,47)
NOTEARS	G(100,210)	G(4,28)	G(4,26)	G(100,211)	G(5,20)	G(4,101)
PC	G(100,144)	G(29,29)	G(3,3)	G(100,110)	G(35,35)	G(2,1)
SEMdag_BU_TL	G(100,608)	G(3,19)	G(12,17)	G(100,514)	G(4,12)	G(10,24)
SEMdag_BU_TO	G(100,610)	G(2,10)	G(12,48)	G(100,531)	G(2,8)	G(11,62)
SEMdag_KB_TL	G(49,202)	G(9,18)	G(8,12)	G(159,1069)	G(32,52)	G(13,51)
SEMdag_KB_TO	G(49,195)	G(3,9)	G(8,17)	G(159,767)	G(8,27)	G(10,61)

MDS. Figure 5.4 shows the MDS plots divided by disease (ALS, BRCA, COVID-19, STEMI). The figures give a quick overview about how the causal discovery methods are grouped together based on the SHD of the recovered graph structures. A cluster between ARGES-CAM-GES-LINGAM-NOTEARS-PC can be identified, showing similar causal structures. The most distant objects are represented by all SEM-based methods that appear to be distant from all the other ones, belonging to different classes. For ALS and STEMI, SEMdag_KB methods belong to the same classes, moving away from SEMdag_BU methods.

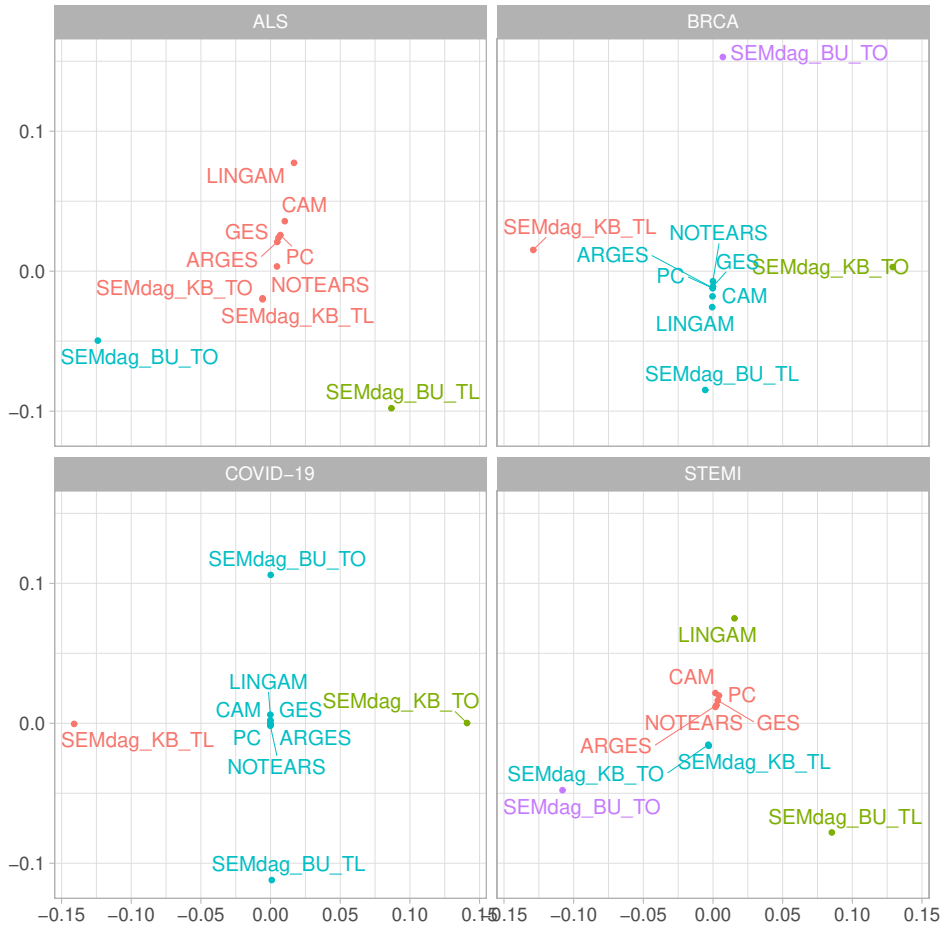

FIGURE 5.4: MDS plots divided by disease (ALS, BRCA, COVID-19, STEMI).

MCC. MCC. Figure 5.5 report the MCC score divided by disease (AD, ALS, BRCA, COVID-19). Higher MCC score is reported for BRCA dataset where MCC reaches almost the level of 1, indicating perfect agreement between the observation and prediction. However, no real differences between the methods can be assessed, only that SEMdag_KB_TO has lower performance. MCC around 0.5 is reported for COVID-19 dataset, specifically shown by RF together with GES, SEMdag_BU_TL and SEMdag_KB_TL. Regarding STEMI and ALS dataset, SEMdag_KB_TO is the only method able to almost recover the RF performance around 0.3, with a good performance also of the other `SEMdag()` methods. In that case, good score is also reported by CAM, LINGAM and PC. GES, ARGES are the methods with lower performance: for the ALS disease, their MCC scores are around 0, showing a prediction no better than random choice; for the STEMI disease, GES has a negative MCC score, showing total disagreement between prediction and observation in the latter. Overall, SEM-based methods are able to recover RF in almost all scenarios, with BU approach methods that have slightly lower MCC scores.

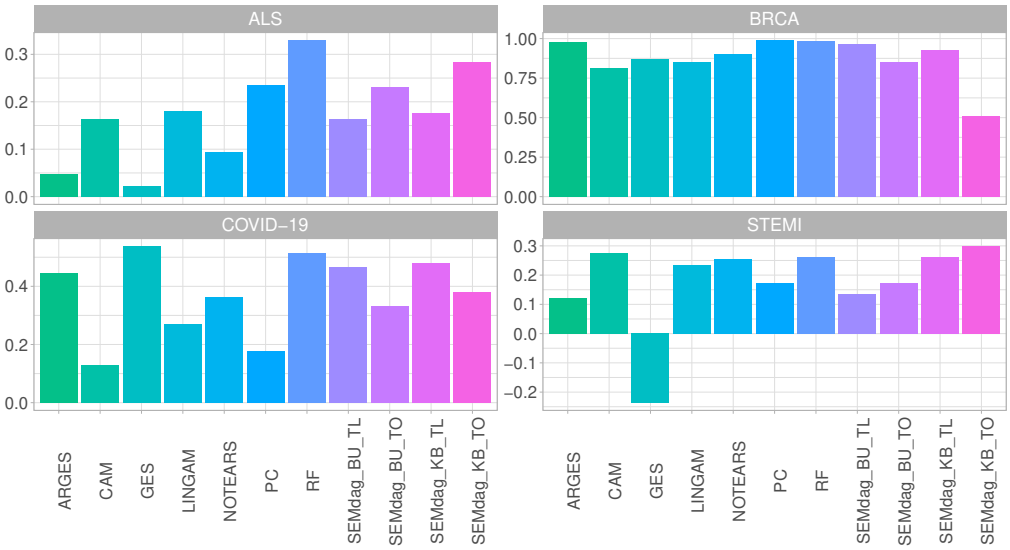

FIGURE 5.5: MCC score divided by disease (ALS, BRCA, COVID-19, STEMI).

5.5 Discussion and conclusions

Building on existing literature, we have discussed the problem of learning high-dimensional linear SEMs introducing a two-stage algorithm called `SEMdag()` and included in the R package **SEMgraph**. First, (1) the linear order is estimated via a priori graph topological vertex (TO) or level (TL) ordering, or by using a data-driven node or level bottom-up (BU) procedure; then, (2) the DAG is estimated using penalized (L1) regressions.

This methodology stands within the class of order-based methods and assumes equal variance of the error terms. `SEMdag()` differs from the other methods since it requires a graphical structure as input and makes use of different procedures for learning the ordering.

In the experimental design scheme, we performed a set of experiments on observed RNA-seq data considering a pair of training and testing dataset for four different diseases, where the latter has been used for disease predictive performance evaluation. Comparisons have been made with (i) a traditional supervised learning algorithm (RF) and with (ii) a set of structure discovery methods to find a structure learning method that provide an optimal solution while controlling the computing time of the algorithm.

If the input graph is known (knowledge-based approach), `SEMdag()` predictions are able to recover almost the same performance of RF algorithm in all four disease datasets, with slightly higher performance in the knowledge-based approach. This result was expected, since, starting from existing graph knowledge, the causal structure should better reflect the data of interest. However, even if the graph is not known, our algorithm is able to report good performance (bottom-up ordering). Unlike `SEMdag()`, the other methods are case-sensitive, having a lower or higher performance depending on the data matrix given as input, not representing a generally optimal solution.

In addition to the good results in predictive terms, unlike the existing literature, `SEMdag()` is able to: deal with high dimensional problems with reduced computational burden; allow the user to specify different structure learning procedures; recover a graph structure that well fits the data of interest.

Conclusions

Identification and characterization of the individual molecules inside a complex biological system, such as cells, tissues, or even the human body, is insufficient for understanding the system. Having a detailed understanding of how molecules and pathways interact is also essential. This is especially true when trying to grasp complex diseases. The motivation of this thesis is grounded on the growing attention on computational models to support researchers in developing hypotheses to direct the design of new experimental tests, systematically analyzing perturbations of systems, and eventually evaluating the suitability of particular molecules as novel therapeutic targets. Researchers can use mathematical models to examine the relationships between intricate regulatory mechanisms , such as metabolic processes or signaling and regulatory pathways, and how disturbances of these processes may lead to the onset of disease. This thesis aims to provide a completely automated framework for managing complex biological systems as multivariate networks embedded in structural equation models.

In *Chapter 1*, a fast and user-friendly, yet powerful R package for causal network analysis is presented, called **SEMgraph**. It conveys causal structure learning within the framework of multivariate linear networks, combining accurate data-driven discovery and confounding adjustment to model interpretability. The other chapters aim to present and validate the four main algorithms within the **SEMgraph** package.

In *Chapter 2*, we employed `SEMgsa()` to find biologically significant results in a FTD DNA methylation dataset and a COVID-19 RNA-seq dataset, and we compared its performance with various other approaches. `SEMgsa()` outperforms other software tools and exhibits low p-values (0.001) and high rankings while being very sensitive to the disease-specific pathways. To produce simulated expression data and assess the effectiveness of the approaches in terms of type I error and statistical power, three route dysregulation mechanisms were used. The best overall performance of `SEMgsa()` is supported by simulation results. `SEMgsa()` is an innovative but powerful approach to evaluate enrichment in relation to gene expression data. It uses pathway perturbation statistics and topological data to disclose biological information.

In *Chapter 3*, we used `SEMtree()` on simulated datasets with distinct differential expression patterns as well as the COVID-19 RNA-seq dataset. `SEMtree()`, as compared to other approaches, is able to capture biologically significant sub-networks with straightforward directed path visualization, effective perturbation extraction, and classifier performance. Despite the fact that trees are oversimplified representations of biological systems, we think that `SEMtree()` can be a helpful tool for practitioners when performing complex subnetwork detection analysis as well as when determining dependence (causal) structure using a direct tree (arborescence) beginning with a list of genes. By emphasizing highly connected hub nodes or neighborhoods that could be further studied, this straightforward graph might be helpful as a first step in visualizing observational high-dimensional data.

In *Chapter 4*, we applied `SEMbap()` on the BRCA RNA-seq dataset and simulated expression data. In the latter, many hidden covariance matrix configurations have been replicated. `SEMbap()` is a two-stage procedure. In the first stage, an exhaustive search of missing edges with significant covariance is performed via d-separation tests; then, in the second stage, a CGGM is fitted or a low dimensional representation of bow-free edges structure is obtained via gLPCA. In comparison to other approaches, the BAP search algorithm achieves good fitting and perturbation metrics, controls the false positive rate, and properly identifies hidden confounding. We have demonstrated that `SEMbap()` offers a number of ways to deal with hidden confounding in a variety of experimental setups.

In *Chapter 5*, we used `SEMdag()` on four sets of experiments involving pairs of observed RNA-seq data (training and testing) for ALS, BRCA, COVID-19 and STEMI diseases. In order to make a relevant comparison regarding the performance in predicting disease, we examined our framework's capacity to find plausible DAGs against six well used causal discovery techniques. In conclusion, predictions based on `SEMdag()` graph structures are able to achieve high performance in all four disease datasets, representing a generally optimal solution. Moreover, the evaluated structure discovery methods can be differentiated according to the accessibility (and adaptability) of their algorithms and the processing time required. In the end, `SEMdag()` recovers a graph structure that nicely matches the data of interest, handles high dimensional issues with less computing load, and lets the user define several structure learning techniques.

This thesis is mainly focused on methodological aspects combining network analysis and causal inference within the framework of SEM and applied on biological systems. The contribution of this thesis is in the development of new approaches to evaluate relevance and perturbation of every biological variable in the context of their shared interactions, extending its connectivity on the base of empirical data and possible exogenous influences, highlighting sources, predictors, causal paths connecting them, and their possible aggregation into modules in knowledge-based biological networks.

Future research should extend and improve the methodological frameworks presented so far. Moreover, further interesting domains of application are already under examination, such as identifying relevant genes in a co-expression network using a (cooperative) game theoretic approach (Shapley values) and a non-linear extension of linear SEM through deep neural network (DNN) algorithms enclosed in a nodewise-based model fitting.

Appendix A

Supplementary material

A.0.1 SEMgraph

Code to reproduce all results of this supplementary material, and additional code of how-to-use graph utilities (conversion and helper functions) implemented in **SEMgraph** can be found in the supplementary file available at: https://github.com/fernandoPalluzzi/SEMgraph/blob/master/replicationCode.R.
SEMgraph package is available under the GNU General Public License version 3 or higher (GPL $\geq$ 3) from CRAN repository, at https://cran.r-project.org/web/packages/SEMgraph and the latest stable version can be installed via: `install.packages("SEMgraph")`. The development version of **SEMgraph** can be installed from the GitHub repository, at https://github.com/fernandoPalluzzi/SEMgraph through:`devtools::install_github("fernandoPalluzzi/SEMgraph")`.

SEMgraph works directly with a collection of interactomes from commonly used biological databases after `igraph` conversion or any user `igraph` objects. KEGG (Kanehisa and Goto, 2000), Reactome (Jassal et al., 2020), and STRING (Szklarczyk et al., 2019) interactomes stored as `igraph` objects, so that they can be manipulated in R, are available in the **SEMdata** data package at: https://github.com/fernandoPalluzzi/SEMdata. KEGG and Reactome are also present as a list of `igraph` objects (`kegg.pathways` and `reactome.pathways`, respectively), each being a single pathway.
However, an external graph representation can be easily used as input for **SEMgraph** workflow provided that it has been transformed to an `igraph` object. The `igraph` package provides a variety of conversion tools, including `graph_from_data_frame()`, `graph_from_edgelist()`, `graph_from_graphnel()`, `graph_from_adjacency_matrix()` functions that create an `igraph` from data frames, edge list (i.e., SIF format), graphNEL graphs, adiacency matrices, respectively. In addition, **SEMgraph** includes `lavaan2graph()` and `dagitty2graph()` functions to convert `lavaan` syntax and `dagitty` graphs in `igraph` objects.

For performance details, Table A.1 shows results of run time (in seconds) and SEM fitting indices (dev/df and SRMR) for the main `SEMrun()` function on four graph objects (small, medium, large and merged), three fitting algorithms ("lavaan", "ricf", and "cggm") and three different option settings: group=NULL (sem0), group=group (sem1), and group=group + fit=2 (sem2), using full ALS RNA-seq expression data retrieved in **SEMdata** package (n = 160 subjects, and p = 17695 genes). Input graphs have been specified as follows:

1. small graph (V=32; E=47): subgraph of the KEGG pathway: "Amyotrophic Lateral Sclerosis (ALS)"

2. medium graph (V=190; E=261): "Amyotrophic Lateral Sclerosis (ALS)" KEGG pathway
3. large graph (V=366; E=1128): union of ALS with the related KEGG pathway: "Pathways of neurodegeneration - multiple diseases"
4. merged graph (V=188; E=622): merged graph of the union graph (3) with prototypes nodes (h=0.2)

The merged graph derives from the `mergeNodes()` function applied on the large graph (see documentation `?mergeNodes`). This function allows to reduce the input graph by hierarchical clustering with prototypes derived from the **protoclust** R package. By cutting the dendrogram at height $h = 1 - |r_{jk}| = 0.2$, i.e. Pearson's correlation coefficient $r = 0.8$, we obtain a merged graph that is roughly half the size of the large graph, but preserving its global structure.
Table 1 shows similar results for the three fitting algorithms, but different system time. As expected, lavaan is very slow for graph of medium or large size (>100 nodes) due to huge Hessian matrix computation for parameter Standard Error (SE). While, RICF is very fast, but computes individual parameter p-values only in the setting of group perturbation in the "common" model (sem1). CGGM is fast in all setting. Thus, SEMrun() function switch settings from algo="lavaan" to algo="ricf" (if group=group) or algo="cggm" (if group=group and fit=2) for fast fitting in case of graph (models) with >100 nodes. In addition, the merged graph can be fitted with the lavaan algorithm (and related SE computation) in relatively short time.

TABLE A.1: System time in seconds (PC: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz; RAM 32.0 GB; Windows 11Pro-21H2) and fitting global measures of `SEMrun()` with group=NULL (sem0), group=group (sem1), and group=group + fit=2 (sem2) with algo = "lavaan", "ricf", or "cggm".

		sem0		**sem1**		**sem2**	
graph	**method**	**time**	**srmr**	**time**	**srmr**	**time**	**srmr**
small	**lavaan**	0.30	0.29	0.42	0.27	0.73	0.28
	ricf	0.07	0.30	0.35	0.27	0.12	0.29
	cggm	0.08	0.29	0.10	0.27	0.18	0.29
medium	**lavaan**	59.00	0.36	86.98	0.35	249.82	0.37
	ricf	0.35	0.36	0.76	0.35	0.61	0.37
	cggm	1.10	0.36	1.07	0.35	2.08	0.37
large	**lavaan**	18 min	0.31	24 min	0.32	90 min	0.31
	ricf	1.09	0.33	2.07	0.32	2.17	0.34
	cggm	26.69	0.29	28.31	0.28	44.56	0.30
merged	**lavaan**	154.89	0.23	204.72	0.24	10 min	0.23
	ricf	0.70	0.25	1.19	0.24	1.29	0.25
	cggm	1.95	0.22	2.14	0.21	4.07	0.23

A.0.2 SEMgsa

Source code and data are available in
https://github.com/fernandoPalluzzi/SEMgraph/tree/master/SEMgsa_replication.

A.0.3 SEMtree

Code to reproduce all results of the analysis, together with the COVID-19 data used in this study can be found in the supplementary files available at:
https://github.com/fernandoPalluzzi/SEMgraph/tree/master/SEMtree.

A.0.4 SEMbap

Code to reproduce all results of the analysis, together with BRCA RNA-seq dataset used in this study are available at:
https://github.com/fernandoPalluzzi/SEMgraph/tree/master/SEMbap.

A.0.5 SEMdag

Code to reproduce all results of the analysis, together with ALS, BRCA, COVID-19 and STEMI RNA-seq dataset used in this study are available at:
https://github.com/fernandoPalluzzi/SEMgraph/tree/master/SEMdag.

Bibliography

Ackermann, Marit and Korbinian Strimmer (2008). "A general modular framework for gene set enrichment analysis". In: *BMC Bioinformatics* 10, pp. 47 –47.

Agosto, Arianna, Daniel Felix Ahelegbey, and Paolo Giudici (2020). "Tree networks to assess financial contagion". In: *Economic Modelling* 85, pp. 349–366. ISSN: 0264-9993.

Agrawal, Raj et al. (2022). *The DeCAMFounder: Non-Linear Causal Discovery in the Presence of Hidden Variables*.

Ahelegbey, Daniel Felix, Paolo Giudici, and Branka Hadji-Misheva (2019). "Latent factor models for credit scoring in P2P systems". In: *Physica A: Statistical Mechanics and its Applications* 522, pp. 112–121. ISSN: 0378-4371.

Ahmad, Liyana, Serge Mostowy, and Vanessa Sancho-Shimizu (Nov. 2018). "Autophagy-Virus Interplay: From Cell Biology to Human Disease". In: *Frontiers in Cell and Developmental Biology* 6, p. 155.

Akaike, H. (1974). "A new look at the statistical model identification". In: *IEEE Transactions on Automatic Control* 19.6, pp. 716–723.

AL, Tarca et al. (Jan. 2009). "A novel signaling pathway impact analysis". In: *Bioinformatics* 25.1, pp. 75–82.

Al-Shahrour, Fátima, Ramón Díaz-Uriarte, and Joaquín Dopazo (Apr. 2005). "Discovering molecular functions significantly related to phenotypes by combining gene expression data and biological information". In: *Bioinformatics* 21.13, pp. 2988–2993.

Andersson, Steen, David Madigan, and Michael Perlman (Feb. 2000). "A Characterization of Markov Equivalence Classes for Acyclic Digraphs". In: *Annals of Statistics* 25.

Ansari, Sahar et al. (2017). "A Novel Pathway Analysis Approach Based on the Unexplained Disregulation of Genes". In: *Proceedings of the IEEE* 105.3, pp. 482–495.

Ashburner, M et al. (2000). "Gene Ontology: Tool for the Unification of Biology. The Gene Ontology Consortium". In: *Nature Genetics* 25.1, pp. 25–29.

Babu MM Luscombe NM, Aravind L et al. (2004). "Structure and evolution of transcriptional regulatory networks". In: *Current Opinion in Structural Biology* 14(3), 283–91.

Bai, J and K Li (2012). "Statistical analysis of factor models of high dimension". In: *The Annals of Statistics* 40.1, pp. 436–465.

Baldi, Pierre et al. (May 2000). "Assessing the accuracy of prediction algorithms for classification: an overview ". In: *Bioinformatics* 16.5, pp. 412–424.

Barabási A. L., Gulbahce N. and J. Loscalzo (2011). "Network medicine: a network-based approach to human disease". In: *Nat Rev Genet* 12(1), pp. 56–68.

Bareinboim, Elias et al. (2022). "On Pearl's Hierarchy and the Foundations of Causal Inference". In: *Probabilistic and Causal Inference: The Works of Judea Pearl*. 1st ed. New York, NY, USA: Association for Computing Machinery, 507–556.

Barh, Debmalya et al. (May 2021). "Predicting COVID-19-Comorbidity Pathway Crosstalk-Based Targets and Drugs: Towards Personalized COVID-19 Management". In: *Biomedicines* 9, p. 556.

Bayerlová, Michaela et al. (2015). "Comparative study on gene set and pathway topology-based enrichment methods". In: *BMC Bioinformatics* 16.

Beisser, Daniela et al. (Feb. 2010). "BioNet: an R-Package for the functional analysis of biological networks". In: *Bioinformatics* 26.8, pp. 1129–1130.

Bello, Kevin, Bryon Aragam, and Pradeep Ravikumar (2022). "DAGMA: Learning DAGs via M-matrices and a Log-Determinant Acyclicity Characterization". In: *Advances in Neural Information Processing Systems*.

Benjamini, Yoav and Yosef Hochberg (Nov. 1995). "Controlling The False Discovery Rate - A Practical And Powerful Approach To Multiple Testing". In: *J. Royal Statist. Soc., Series B* 57, pp. 289 –300.

Bentler, P M (2016). *EQS 6 Structural Equations Program Manual*. Encino, CA. URL: http://www.mvsoft.com/.

Bentler, P. M. and D. G. Weeks (1980). "Linear structural equations with latent variables". In: *Psychometrika* 45.3, pp. 289–308.

Bien, Jacob and Robert Tibshirani (Sept. 2011). "Hierarchical Clustering With Prototypes via Minimax Linkage". In: *Journal of the American Statistical Association* 106, pp. 1075–1084.

Bollen, Kenneth A (1989). *Structural Equations with Latent Variables*. 1st. Hoboken, NJ, USA: John Wiley & Sons.

Bollen, Kenneth A. and Robert A. Stine (1992). "Bootstrapping Goodness-of-Fit Measures in Structural Equation Models". In: *Sociological Methods & Research* 21.2, pp. 205–229.

Breiman, L (Oct. 2001). "Random Forests". In: *Machine Learning* 45, pp. 5–32.

Brito, Carlos and Judea Pearl (2002). "A New Identification Condition for Recursive Models With Correlated Errors". In: *Structural Equation Modeling* 9.4, pp. 459–474.

Brown, Morton B. (1975). "A Method for Combining Non-Independent, One-Sided Tests of Significance". In: *Biometrics* 31.4, pp. 987–992.

Brueffer, Christian et al. (2018). "Clinical Value of RNA Sequencing–Based Classifiers for Prediction of the Five Conventional Breast Cancer Biomarkers: A Report From the Population-Based Multicenter Sweden Cancerome Analysis Network—Breast Initiative". In: *JCO Precision Oncology* 2, pp. 1–18.

Bühlmann, Peter and Sara van de Geer (2011). *Statistics for high-dimensional data*. Springer Series in Statistics. Methods, theory and applications. Springer, Heidelberg.

Bühlmann, Peter, Jonas Peters, and Jan Ernest (2014). "CAM: Causal Additive Models, high-dimensional order search and penalized regression". In: *The Annals of Statistics* 42.6, pp. 2526–2556.

Carapito, Raphael et al. (2021). "Identification of driver genes for severe forms of COVID-19 in a deeply phenotyped young patient cohort". In: *medRxiv*.

Cattell, Raymond B. (1966). "The Scree Test For The Number Of Factors". In: *Multivariate Behavioral Research* 1.2, pp. 245–276.

Cevid, Domagoj, Peter Buhlmann, and Nicolai Meinshausen (2020). "Spectral Deconfounding via Perturbed Sparse Linear Models". In: *Journal of Machine Learning Research* 21.232, pp. 1–41.

Chandrasekaran, Venkat, Pablo A. Parrilo, and Alan S. Willsky (2012). "Latent variable graphical model selection via convex optimization". In: *The Annals of Statistics* 40.4, pp. 1935 –1967.

Chatterjee, Sourav and Mathukumalli Vidyasagar (2022). *Estimating large causal polytree skeletons from small samples*. arXiv: 2209.07028 [stat.ME].

Chen, Wenyu, Mathias Drton, and Y Samuel Wang (2019). "On Causal Discovery with an Equal-Variance Assumption". In: *Biometrika* 106.4, pp. 973–980.

Chen-Plotkin, Alice S. et al. (Jan. 2008). "Variations in the progranulin gene affect global gene expression in frontotemporal lobar degeneration". In: *Human Molecular Genetics* 17.10, pp. 1349–1362.

Chernozhukov, Victor, Christian Hansen, and Yuan Liao (Feb. 2017). "A lava attack on the recovery of sums of dense and sparse signals". In: *The Annals of Statistics* 45, pp. 39–76.

Chicco, Davide and Giuseppe Jurman (Jan. 2020). "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.

Chickering, David Maxwell (2003). "Optimal Structure Identification With Greedy Search". In: *J. Mach. Learn. Res.* 3, pp. 507–554.

Chow, C. K. and C. N. Liu (1968). "Approximating discrete probability distributions with dependence trees". In: *IEEE Trans. Inf. Theory* 14, pp. 462–467.

Cizmarevic, Nada et al. (Dec. 2021). "Could the CCR5-Delta32 mutation be protective in SARS-CoV-2 infection?" In: *Physiological research* 70, S249–S252.

Colombo, Diego and Marloes H. Maathuis (2014). "Order-Independent Constraint-Based Causal Structure Learning". In: *Journal of Machine Learning Research* 15.116, pp. 3921–3962.

Cooper-Knock, Johnathan et al. (2015). "C9ORF72 GGGGCC Expanded Repeats Produce Splicing Dysregulation which Correlates with Disease Severity in Amyotrophic Lateral Sclerosis". In: *PLoS One* 10.5, e0127376.

Csardi, Gabor and Tamas Nepusz (2006). "The igraph software package for complex network research". In: *InterJournal* Complex Systems, p. 1695.

Davidson, E.H. (2001). *Genomic Regulatory Systems: Development and Evolution*. Academic Press.

Davies, PT and MKS Tso (1982). "Procedures for Reduced-Rank Regression". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 31.3, pp. 141–145.

Dittrich, Marcus T. et al. (July 2008). "Identifying functional modules in protein–protein interaction networks: an integrated exact approach". In: *Bioinformatics* 24.13, pp. i223–i231.

Dobriban, Edgar (2020). "Permutation methods for factor analysis and PCA". In: *The Annals of Statistics* 48.5, pp. 2824 –2847.

Draghici, Sorin et al. (Nov. 2007). "A systems biology approach for pathway level analysis". In: *Genome research* 17, pp. 1537–45.

Drton, M, D Foygel, and S Sullivant (2011). "Global Identifiability of Linear Structural Equation Models". In: *The Annals of Statistics* 39.2, pp. 865–886.

Drton, Mathias, Michael Eichler, and Thomas S Richardson (2009). "Computing Maximum Likelihood Estimated in Recursive Linear Models with Correlated Errors". In: *Journal of Machine Learning Research* 10.81, pp. 2329–2348.

Edwards, David, Lei Wang, and Peter Sørensen (July 2012). "Network-enabled gene expression analysis". In: *BMC bioinformatics* 13, p. 167.

Efron, Bradley and Robert Tibshirani (June 2007). "On testing the significance of sets of genes". In: *The Annals of Applied Statistics* 1.1, pp. 107 –129.

Emmert-Streib, Frank (Oct. 2007). "The Chronic Fatigue Syndrome: A Comparative Pathway Analysis". In: *Journal of computational biology : a journal of computational molecular cell biology* 14, pp. 961–72.

Emmert-Streib, Frank and Galina Glazko (July 2011). "Network Biology: A Direct Approach to Study Biological Function". In: *Wiley interdisciplinary reviews. Systems biology and medicine* 3, pp. 379–91.

Feng, Shanshan et al. (Jan. 2022). "Potential Genes Associated with COVID-19 and Comorbidity". In: *International journal of medical sciences* 19, pp. 402–415.

Finos, Livio et al. (2018). *flip: Multivariate Permutation Tests*. R package version 2.5.0. URL: https://CRAN.R-project.org/package=flip.

Fisher, R. A. (1915). "Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population". In: *Biometrika* 10.4, pp. 507–521.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani (2019). *Graphical Lasso: Estimation of Gaussian Graphical Models*. R package version 1.11. URL: https://CRAN.R-project.org/package=glasso.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (Aug. 2008a). "Sparse inverse covariance estimation with the graphical LASSO". In: *Biostatistics (Oxford, England)* 9, pp. 432–41.

Friedman, Jerome et al. (2023). *glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 4.1-8. URL: https://CRAN.R-project.org/package=glmnet.

Friedman, Jerome H., Trevor J. Hastie, and Robert Tibshirani (2008b). "Sparse inverse covariance estimation with the graphical lasso." In: *Biostatistics* 9 3, pp. 432–41.

Frot, Benjamin, Preetam Nandy, and Marloes Maathuis (Mar. 2019). "Robust causal structure learning with some hidden variables". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 81.

Frydenberg, Morten (1990). "The chain graph Markov property". In: *Scandinavian Journal of Statistics* 17, pp. 333–353.

Förster, Jochen et al. (Mar. 2003). "Genome-Scale Reconstruction of the Saccharomyces cerevisiae Metabolic Network". In: *Genome research* 13, pp. 244–53.

Gao, Ming, Yi Ding, and Bryon Aragam (2020). "A polynomial-time algorithm for learning nonparametric causal graphs". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 11599–11611.

Gao, Yong-Lei et al. (2018). "Tau in neurodegenerative disease". In: *Annals of Translational Medicine* 6.10.

Ge, Weihao and Eric Jakobsson (2018). "Systems Biology Understanding of the Effects of Lithium on Affective and Neurodegenerative Disorders". In: *Frontiers in Neuroscience* 12.

German, Rana et al. (July 2023). "Exploring breast tissue microbial composition and the association with breast cancer risk factors". In: *Breast Cancer Research* 25.

Ghoshal, Asish and Jean Honorio (2018). "Learning linear structural equation models in polynomial time and sample complexity". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Ed. by Amos Storkey and Fernando Perez-Cruz. Vol. 84. Proceedings of Machine Learning Research. PMLR, pp. 1466–1475.

Giudici, Paolo and Gloria Polinesi (2021). "Crypto price discovery through correlation networks". In: *Annals of Operations Research* 299.1, pp. 443–457.

Gnecco, Nicola et al. (June 2021). "Causal discovery in heavy-tailed models". In: *The Annals of Statistics* 49.

Goeman, Jelle J. and Peter Bühlmann (Feb. 2007). "Analyzing gene expression data in terms of gene sets: methodological issues". In: *Bioinformatics* 23.8, pp. 980–987.

Goldstein, Tom, Donoghue, and Simon Setzer (July 2014). "Fast Alternating Direction Optimization Methods". In: *SIAM Journal on Imaging Sciences* 7.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press.

Grassi, Mario, Fernando Palluzzi, and Barbara Tarantino (Aug. 2022). "SEMgraph: an R package for causal network inference of high-throughput data with structural equation models". In: *Bioinformatics* 38.20, pp. 4829–4830.

Gu, Jin et al. (Apr. 2010). "Identification of responsive gene modules by network-based gene clustering and extending: Application to inflammation and angiogenesis". In: *BMC systems biology* 4, p. 47.

Guo, Zijian, Domagoj Ćevid, and Peter Bühlmann (2022). "Doubly debiased lasso: High-dimensional inference under hidden confounding". In: *The Annals of Statistics* 50.3, pp. 1320 –1347.

Han, Jing-Dong et al. (Aug. 2004). "Evidence for dynamically organized modularity in the yeast protein-protein interaction network". In: *Nature* 430, pp. 88–93.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. 2nd ed. Springer Science.

Heinze-Deml, Christina, Marloes H. Maathuis, and Nicolai Meinshausen (2018). "Causal Structure Learning". In: *Annual Review of Statistics and Its Application* 5.1, pp. 371–391.

Hellstern, Michael et al. (June 2021). "netgsa: Fast computation and interactive visualization for topology-based pathway enrichment analysis". In: *PLOS Computational Biology* 17.

Hu, L. and P. M. Bentler (1999). "Cutoff Criteria for Fit Indexes in Covariance Structure Analysis: Conventional Criteria Versus New Alternatives". In: *Structural Equation Modeling: A Multidisciplinary Journal* 6.1, pp. 1–55.

Hudson, Aaron and Ali Shojaie (2022). "Covariate-Adjusted Inference for Differential Analysis of High-Dimensional Networks". In: *Sankhya A* 84, 345—388.

Ideker, Trey and Nevan J Krogan (2012). "Differential network biology". In: *BMC Systems Biology* 8.565.

Ideker, Trey et al. (July 2002). "Discovering regulatory and signalling circuits in molecular interaction networks". In: *Bioinformatics* 18.suppl$_1$, S233–S240.

Jaakkola, Maria K. and Laura L. Elo (2015). "Empirical comparison of structure-based pathway methods". In: *Briefings in Bioinformatics* 17.2, pp. 336–345.

Jablonski, Kim Philipp et al. (Dec. 2021). "Identifying cancer pathway dysregulations using differential causal effects". In: *Bioinformatics* 38.6, pp. 1550–1559.

Jacob, Laurent, Pierre Neuvial, and Sandrine Dudoit (June 2012). "More power via graph-structured tests for differential expression of gene networks". In: *The Annals of Applied Statistics* 6.2.

Jakobsen, Martin E. et al. (2022). "Structure Learning for Directed Trees". In: *Journal of Machine Learning Research* 23.159, pp. 1–97.

Janková, Jana and Sara van de Geer (2015). "Confidence intervals for high-dimensional inverse covariance estimation". In: *Electronic Journal of Statistics* 9.1, pp. 1205 – 1229.

Janková, Jana and Sara van de Geer (2015). "Confidence intervals for high-dimensional inverse covariance estimation". In: *Electronic Journal of Statistics* 9.1, pp. 1205–1229.

Jankova, Jana and Sara Van De Geer (2019). "Handbook of Graphical Models". In: Chapman Hall/CRC. Chap. 14, pp. 325–349.

Jassal, Bijay et al. (2020). "The Reactome Pathway Knowledgebase". In: *Nucleic Acids Research* 48.D1, pp. D498–D503.

Jiang, Bo et al. (2013). "Graph-Laplacian PCA: Closed-Form Solution and Robustness". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3492–3498.

Jothi, Raja et al. (Feb. 2009). "Genomic analysis shows a tight link between transcription factor dynamics and regulatory network architecture". In: *Molecular systems biology* 5, p. 294.

Jöreskog, Karl G. (1973). "Analysis of Covariance Structures". In: *Multivariate Analysis–III*. Ed. by PARUCHURI R. KRISHNAIAH. Academic Press, pp. 263–285.

Jöreskog, Karl G. and Dag Sörbom (1982). "Recent Developments in Structural Equation Modeling". In: *Journal of Marketing Research* 19.4, pp. 404–416.

Kalisch, Markus and Peter Bühlmann (2007). "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm". In: *Journal of Machine Learning Research* 8.22, pp. 613–636.

Kanehisa, M and S Goto (2000). "KEGG: Kyoto Encyclopedia of Genes and Genomes". In: *Nucleic Acids Research* 28.1, pp. 27–30.

Kessy, Agnan, Alex Lewin, and Korbinian Strimmer (2018). "Optimal Whitening and Decorrelation". In: *The American Statistician* 72.4, pp. 309–314.

Khatri, Purvesh, Marina Sirota, and Atul Janardhan Butte (2012). "Ten Years of Pathway Analysis: Current Approaches and Outstanding Challenges". In: *PLoS Computational Biology* 8.

Khatri, Purvesh et al. (2007). *A System Biology Approach for the Steady-State Analysis of Gene Signaling Networks*.

Kiliszek, Marek et al. (Nov. 2012). "Altered Gene Expression Pattern in Peripheral Blood Mononuclear Cells in Patients with Acute Myocardial Infarction". In: *PloS one* 7, e50054.

Kleinberg, Jon and Éva Tardos (2006). *Algorithm Design*. Addison Wesley.

Kou, Lawrence T., George Markowsky, and Leonard Berman (1981). "A fast algorithm for Steiner trees". In: *Acta Informatica* 15, pp. 141–145.

Köhler, Sebastian et al. (May 2008). "Walking the Interactome for Prioritization of Candidate Disease Genes". In: *American journal of human genetics* 82, pp. 949–58.

Langfelder, Peter and Steve Horvath (2007). "Eigengene networks for studying the relationships between co-expression modules". In: *BMC Systems Biology* 1.54.

Larson, Jessica L and Art B Owen (2015). "Moment based gene set tests". In: *BMC Bioinformatics* 16, p. 132.

Lauritzen, Steffen L. (1996). *Graphical Models*. Oxford University Press.

Lebrón, Ricardo and Gherardo Varando (2021). *gnlearn: Genetic Network Learning*. R package version 0.0.0.

Leek, Jeffrey T and John D Storey (Sept. 2007). "Capturing Heterogeneity in Gene Expression Studies by Surrogate Variable Analysis". In: *PLOS Genetics* 3.9, pp. 1–12.

Li, Y et al. (2015). "An epigenetic signature in peripheral blood associated with the haplotype on 17q21.31, a risk factor for neurodegenerative tauopathy." In: *PLoS Genetics* 10.3, e1004211.

Liu, Chuang et al. (2020). "Computational Network Biology: Data, Models, and Applications". In: *Physics Reports* 846, pp. 1–66.

Liu, Xiaonan et al. (Nov. 2021). "SARS-CoV-2–host proteome interactions for antiviral drug discovery". In: *Molecular Systems Biology* 17.

Loh, Po-Ling and Peter Bühlmann (2014). "High-Dimensional Learning of Linear Causal Networks via Inverse Covariance Estimation". In: *J. Mach. Learn. Res.* 15.1, 3065–3105.

Lou, Xingmei, Yu Hu, and Xiaodong Li (2021). *Learning Linear Polytree Structural Equation Models*. arXiv: 2107.10955 [stat.ML].

Love, Michael, Wolfgang Huber, and Simon Anders (Dec. 2014). "Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2". In: *Genome Biol* 15, p. 550.

Ma, Haisu et al. (Mar. 2011). "COSINE: COndition-SpecIfic sub-NEtwork identification using a global optimization method". In: *Bioinformatics* 27.9, pp. 1290–1298.

Ma, Jing, Ali Shojaie, and George Michailidis (Nov. 2019). "A comparative study of topology-based pathway enrichment analysis methods". In: *BMC Bioinformatics* 20.

Maciejak, Agata et al. (Mar. 2015). "Gene expression profiling reveals potential prognostic biomarkers associated with the progression of heart failure". In: *Genome Medicine* 7.

Malaeb, Ziad, James Summers, and Bruce Pugesek (Jan. 2000). "Using structural equation modeling to investigate relationships among ecological variables". In: *Environmental and Ecological Statistics* 7, pp. 93–111.

Marchetti, Giovanni M., Mathias Drton, and Kayvan Sadeghi (2020). *ggm: Graphical Markov Models with Mixed Graphs*. R package version 2.5. URL: https://CRAN.R-project.org/package=ggm.

Massa, Maria Sofia, Monica Chiogna, and Chiara Romualdi (2010). "Gene set analysis exploiting the topology of a pathway". In: *BMC Systems Biology* 4, pp. 121–121.

Matthews, B.W. (1975). "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2, pp. 442–451. ISSN: 0005-2795.

Meek, Christopher (1995). "Causal inference and causal explanation with background knowledge". In: *UAI*.

Mitra K Carvunis AR, Ramesh SK Ideker T. (Oct. 2013). "Integrative approaches for finding modular structure in biological networks". In: *Nat Rev Genet.* 14.10, pp. 719–732.

Mitrea, Cristina et al. (2013). "Methods and approaches in the topology-based analysis of biological pathways". In: *Frontiers in Physiology* 4.

Nandy, Preetam, Alain Hauser, and Marloes Maathuis (2018). "High-dimensional consistency in score-based and hybrid structure learning". In: *Annals of Statistics* 46.6A, pp. 3151–3183.

Newey, Whitney, James Powell, and Francis Vella (1999). "Nonparametric Estimation of Triangular Simultaneous Equations Models". In: *Econometrica* 67.3, pp. 565–604.

Newman, M. E. J. and M. Girvan (2004). "Finding and evaluating community structure in networks". In: *Physical Review E* 69.2, p. 026113.

Ng, Ignavier, AmirEmad Ghassami, and Kun Zhang (2020). "On the Role of Sparsity and DAG Constraints for Learning Linear DAGs". In: *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, 17943—17954.

Ng, Ignavier, Biwei Huang, and Kun Zhang (2023). *Structure Learning with Continuous Optimization: A Sober Look and Beyond*. arXiv: 2304.02146 [cs.LG].

Nguyen, Hung et al. (2019). "A Comprehensive Survey of Tools and Software for Active Subnetwork Identification". In: *Frontiers in Genetics* 10.

Nguyen, Tin, Cristina Mitrea, and Sorin Draghici (2018). "Network-Based Approaches for Pathway Level Analysis". In: *Current protocols in bioinformatics* 61.1, 8.25.1—8.25.24. ISSN: 1934-3396.

Oates C. J., Mukherjee S. (2012). "Network Inference and Biological Dynamics". In: *The annals of applied statistics* 6(3), 1209–1235.

Onatski, Alexei (2010). "Determining the Number of Factors from Empirical Distribution of Eigenvalues". In: *The Review of Economics and Statistics* 92.4, pp. 1004–1016.

Overmyer, Katherine A. et al. (2021). "Large-Scale Multi-omic Analysis of COVID-19 Severity". In: *Cell Systems* 12.1, 23–40.e7. ISSN: 2405-4712.

Palluzzi, Fernando and Mario Grassi (2021). *SEMgraph: An R Package for Causal Network Analysis of High-Throughput Data with Structural Equation Models*.

Palluzzi, Fernando et al. (2017). "A novel network analysis approach reveals DNA damage, oxidative stress and calcium/cAMP homeostasis-associated biomarkers in frontotemporal dementia". In: *PLoS One* 12.10, e0185797.

Park, Gunwoong (2020). "Identifiability of Additive Noise Models Using Conditional Variances". In: *Journal of Machine Learning Research* 21.75, pp. 1–34.

Park, Gunwoong and Youngwhan Kim (2020). "Identifiability of Gaussian linear structural equation models with homogeneous and heterogeneous error variances". In: *Journal of the Korean Statistical Society* 49.1, pp. 276–292.

Pearl, Judea (1998). "Graphs, Causality, and Structural Equation Models". In: *Sociological Methods & Research* 27.2, pp. 226–284.

— (2009). *Causality: Models, reasoning, and inference*. 2nd. New York, NY, USA: Cambridge University Press.

Pearl, Judea, M Maria Glymour, and Nicholas P. Jewell (2016). *Causal Inference in Statistics: A Primer*. Wiley & Sons.

Pepe, Daniele and Mario Grassi (2014). "Investigating Perturbed Pathway Modules from Gene Expression Data via Structural Equation Models". In: *BMC Bioinformatics* 15, p. 132.

Peters, J. and P. Bühlmann (2014). "Identifiability of Gaussian structural equation models with equal error variances". In: *Biometrika* 101.1, pp. 219–228.

Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf (2017). *Elements of Causal Inference: Foundations and Learning Algorithms*. Adaptive Computation and Machine Learning. MIT Press.

Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf (Jan. 2010). "Identifying Cause and Effect on Discrete Data using Additive Noise Models". In: *Journal of Machine Learning Research - Proceedings Track* 9, pp. 597–604.

Petrochilos, Deanna et al. (Oct. 2013). "Using random walks to identify cancer-associated modules in expression data". In: *BioData mining* 6, p. 17.

Pons, Pascal and Matthieu Latapy (2005). *Computing communities in large networks using random walks (long version)*. arXiv: physics/0512106 [physics.soc-ph].

Price, Alkes L. et al. (2006). "Principal components analysis corrects for stratification in genome-wide association studies". In: *Nature Genetics* 38, pp. 904–909.

Prim, Robert C. (1957). "Shortest connection networks and some generalizations". In: *Bell System Technical Journal* 36, pp. 1389–1401.

Regan, Erzsébet (Feb. 2009). "Detecting Hierarchical Modularity in Biological Networks". In: *Methods in molecular biology (Clifton, N.J.)* 541, pp. 145–60.

Reimand, Jüri et al. (2019). "Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap". In: *Nature Protocols* 14.1, pp. 482–517.

Richardson, Thomas and Peter Spirtes (2002). "Ancestral graph Markov models". In: *The Annals of Statistics* 30.4, pp. 962 –1030.

Ritchie, Marylyn et al. (2015). "Methods of Integrating Data to Uncover Genotype-Phenotype Interactions". In: *Nature Review Genetics* 16.2, pp. 85–97.

Robinson, Mark D., Davis J. McCarthy, and Gordon K. Smyth (Nov. 2009). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data". In: *Bioinformatics* 26.1, pp. 139–140.

Rooij, Mark et al. (May 2022). "SEM-Based Out-of-Sample Predictions". In: *Structural Equation Modeling: A Multidisciplinary Journal* 30, pp. 1–17.

Rosseel, Yves (2012). "lavaan: An R Package for Structural Equation Modeling". In: *Journal of Statistical Software* 48.2, pp. 1–36.

Sachs, Karen et al. (May 2005). "Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data". In: *Science (New York, N.Y.)* 308, pp. 523–9.

Sanford, Andrew and Imad Moosa (Apr. 2012). "A Bayesian network structure for operational risk modelling in structured finance operations". In: *Journal of the Operational Research Society* 63, pp. 431–444.

Santiago, Jose A., Virginie Bottero, and Judith A Potashkin (2020). "Transcriptomic and Network Analysis Identifies Shared and Unique Pathways across Dementia Spectrum Disorders". In: *International Journal of Molecular Sciences* 21.

Schäfer, Juliane and Korbinian Strimmer (2005). "A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics". In: *Stat Appl Genet Mol Biol.* 4.1, Article 32.

Schäfer, Juliane et al. (2017). *corpcor: Efficient Estimation of Covariance and (Partial) Correlation*. R package version 1.6.9. URL: https://CRAN.R-project.org/package=corpcor.

Schermelleh-Engel, K. and H. Moosbrugger (2003). "Evaluating the fit of structural equation models: tests of significance and descriptive goodness-of-fit measures". In: *Methods of Psychological Research Online* 8.2, pp. 23–74.

Schwarz, Gideon (1978). "Estimating the dimension of a model". In: *Annals of Statistics* 6.2, pp. 461–468.

Schäfer, Juliane and Korbinian Strimmer (Feb. 2005). "A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics". In: *Statistical applications in genetics and molecular biology* 4, Article32.

Shendure, Jay and Erez Lieberman Aiden (2012). "The Expanding Scope of DNA Sequencing". In: *Journal of Molecular Biology* 30.11, pp. 1084–1094.

Shimizu, Shohei (2014). "Lingam: Non-Gaussian Methods for Estimating Causal Structures". In: *Behaviormetrika* 41, pp. 65–98.

Shimizu, Shohei et al. (2006). "A Linear Non-Gaussian Acyclic Model for Causal Discovery". In: *Journal of Machine Learning Research* 7.72, pp. 2003–2030.

Shimizu, Shohei et al. (2011). "DirectLiNGAM: A Direct Method for Learning a Linear Non-Gaussian Structural Equation Model". In: *Journal of Machine Learning Research* 12.33, pp. 1225–1248.

Shipley, Bill (1999). "Testing causal explanations in organismal biology : causation, correlation and structural equation modelling". In: *Oikos* 86, pp. 374–382.

— (2000). "A New Inferential Test for Path Models Based on Directed Acyclic Graphs". In: *Structural Equation Modeling* 7.

— (2016). *Cause and Correlation in Biology*. 2st. Cambridge, England, UK: Cambridge University Press.

Shojaie, Ali and George Michailidis (2009). "Analysis of Gene Sets Based on the Underlying Regulatory Network". In: *Journal of Computational Biology* 16.3, pp. 407–426.

— (2010). "Penalized Likelihood Methods for Estimation of Sparse High-Dimensional Directed Acyclic Graphs". In: *Biometrika* 97.3, pp. 519–538.

Shutta, Katherine H. et al. (2022). "Gaussian graphical models with applications to omics analyses". In: *Statistics in Medicine* 41.25, pp. 5150–5187.

Smyth, G. K. (2005). "limma: Linear Models for Microarray Data". In: *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Ed. by Robert Gentleman et al. Springer New York, 397–420.

Spirtes, Peter, Clark N. Glymour, and Richard Scheines (2000). *Causation, Prediction, and Search*. 2nd. Cambridge, MA, USA: The MIT Press.

Su, Chen, Simon Rousseau, and Amin Emad (Dec. 2021). "Identification of transcriptional regulatory network associated with response of host epithelial cells to SARS-CoV-2". In: *Scientific Reports* 11.

Subramanian, Aravind et al. (2005). "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles". In: *Proceedings of the National Academy of Sciences* 102.43, pp. 15545–15550.

Szklarczyk, Damian et al. (2019). "STRING v11: Protein-Protein Association Networks With Increased Coverage, Supporting Functional Discovery in Genome-Wide Experimental Datasets". In: *Nucleic Acids Research* 47.D1, pp. D607–D613.

Tao, Liang et al. (2015). "Low Rank Approximation with Sparse Integration of Multiple Manifolds for Data Representation". In: *Applied Intelligence* 42.3, 430–446.

Tao, Ye et al. (2020). "The Role of Autophagy and NLRP3 Inflammasome in Liver Fibrosis". In: *BioMed Research International* 2020.

Tarca, Adi, Gaurav Bhatti, and Roberto Romero (Nov. 2013). "A Comparison of Gene Set Analysis Methods in Terms of Sensitivity, Prioritization and Specificity". In: *PloS one* 8.

Tarca, Adi et al. (June 2012). "Down-weighting overlapping genes improves gene set analysis". In: *BMC bioinformatics* 13.

Tatusova, Tatiana et al. (June 2016). "NCBI prokaryotic genome annotation pipeline". In: *Nucleic Acids Research* 44, pp. 6614–24.

Taylor, Ian W et al. (2009). "Dynamic modularity in protein interaction networks predicts breast cancer outcome". In: *Nature biotechnology* 27.2, 199—204.

Thair, Simone et al. (Dec. 2020). "Transcriptomic Similarities and Differences in Host Response between SARS-CoV-2 and Other Viral Infections". In: *iScience* 24, p. 101947.

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1, pp. 267–288.

Tramontano, Daniele, Anthea Monod, and Mathias Drton (2022). "Learning linear non-Gaussian polytree models". In: *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*. Ed. by James Cussens and Kun Zhang. Vol. 180. Proceedings of Machine Learning Research. PMLR, pp. 1960–1969.

Tsamardinos, I., Laura E. Brown, and Constantin F. Aliferis (2006). "The max-min hill-climbing Bayesian network structure learning algorithm". In: *Machine Learning* 65, pp. 31–78.

Twine, Natalie A. et al. (Jan. 2011). "Whole Transcriptome Sequencing Reveals Gene Expression and Splicing Differences in Brain Regions Affected by Alzheimer's Disease". In: *PLOS ONE* 6.1, pp. 1–13.

Ulgen, Ege, Ozan Ozisik, and Osman Ugur Sezerman (2019). "pathfindR: An R Package for Comprehensive Identification of Enriched Pathways in Omics Data Through Active Subnetworks". In: *Frontiers in Genetics* 10. ISSN: 1664-8021.

Varadan, Vinay et al. (2012). "The Integration of Biological Pathway Knowledge in Cancer Genomics: A review of existing computational approaches". In: *IEEE Signal Processing Magazine* 29.1, pp. 35–50.

Verma, Thomas and Judea Pearl (1990a). "Causal Networks: Semantics and Expressiveness". In: *Machine Intelligence and Pattern Recognition* 9.1, pp. 69–76.

— (1990b). "Equivalence and Synthesis of Causal Models". In: *Probabilistic and Causal Inference*.

Vidal, Marc, Michael Cusick, and Albert-Laszlo Barabasi (Mar. 2011). "Interactome Networks and Human Disease". In: *Cell* 144, pp. 986–98.

Vovk, Vladimir and Ruodu Wang (2020). "Combining p-values via averaging". In: *Biometrika*.

Vowels, Matthew J., Necati Cihan Camgoz, and Richard Bowden (2022). "D'ya Like DAGs? A Survey on Structure Learning and Causal Discovery". In: *ACM Comput. Surv.* 55.4.

Wang, Jingshu and Qingyuan Zhao (2019). *cate: High Dimensional Factor Analysis and Confounder Adjusted Testing and Estimation*. R package version 1.1. URL: https://CRAN.R-project.org/package=cate.

Wang, Lili, Parvin Mousavi, and Sergio Baranzini (Jan. 2015). "iPINBPA: an integrative network-based functional module discovery tool for genome-wide association studies". In: vol. 20.

Watts, Duncan and Steven Strogatz (1988). "Collective dynamics of 'small-world' networks". In: *Nature* 393, pp. 440–442.

Widaman, Keith F. (2018). "On Common Factor and Principal Component Representations of Data: Implications for Theory and for Confirmatory Replications". In: *Structural Equation Modeling: A Multidisciplinary Journal* 25.6, pp. 829–847.

Williams, Donald (2020). *GGMncv: Gaussian Graphical Models with Non-Convex Penalties*. R package version 1.1.0. URL: https://CRAN.R-project.org/package=GGMncv.

Witte, Janine et al. (2020). "On Efficient Adjustment in Causal Graphs". In: *Journal of Machine Learning Research* 21.246, pp. 1–45.

Witten, Daniela M. and Robert Tibshirani (2011). "Penalized Classification Using Fisher's Linear Discriminant". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 73.2, pp. 753–772.

Wright, S. (1921). *Correlation and Causation*.

Wright, Sewall (1934). "The Method of Path Coefficients". In: *The Annals of Mathematical Statistics* 5.3, pp. 161 –215.

Yarden, Yosef and Mark X. Sliwkowski (2001). "Untangling the ErbB signalling network". In: *Nature Reviews Molecular Cell Biology* 2, pp. 127–137.

Yu, Meichen et al. (2015). "Hierarchical clustering in minimum spanning trees". In: *Chaos* 25.2, p. 023107.

Yu, Yue et al. (2019). "DAG-GNN: DAG Structure Learning with Graph Neural Networks". In: *International Conference on Machine Learning*. Ed. by PMLR, pp. 7154–7163.

Yuan, Yiping et al. (Dec. 2018). "Constrained likelihood for reconstructing a directed acyclic Gaussian graph". In: *Biometrika* 106.1, pp. 109–125.

Zhang, Keli et al. (2021). *gCastle: A Python Toolbox for Causal Discovery*. arXiv: 2111.15155 [cs.LG].

Zhao, Tuo et al. (2012). "The huge Package for High-dimensional Undirected Graph Estimation in R". In: *Journal of Machine Learning Research* 13.1, pp. 1059–1062. URL: https://cran.r-project.org/web/packages/huge/vignettes/vignette.pdf.

Zheng, Siyuan and Zhongming Zhao (2012). "GenRev: Exploring functional relevance of genes in molecular networks". In: *Genomics* 99.3, pp. 183–188.

Zheng, Xun et al. (2018). "DAGs with NO TEARS: Continuous Optimization for Structure Learning". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc.