Extracting dynamic pathways with time-course gene expression datasets: from differential expression analysis to temporal multilayer networks

Michaël Pierrelée

NNT/NL: 2021AIXM0162/036ED62

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 21 avril 2021 par

Michaël PIERRELÉE

Extracting dynamic pathways with time-course gene expression datasets: from differential expression analysis to temporal multilayer networks

Discipline
Biologie-Santé

Spécialité
Génomique et Bioinformatique

École doctorale
ED 62 – Sciences de la Vie et de la Santé

Laboratoire
Institut de Biologie du Développement de Marseille (IBDM/CNRS UMR 7288)

Composition du jury

Andreas BEYER CECAD – Cologne	Rapporteur
Claude PASQUIER CNRS I3S – Sophia-Antipolis	Rapporteur
Laurence CALZONE Institut Curie – Paris	Examinatrice
Laurent TICHIT CNRS I2M – Marseille	Examinateur
Nathalie VIALANEIX INRAE MIA-T – Toulouse	Présidente du jury
Bianca HABERMANN CNRS IBDM – Marseille	Co-directrice doctorale
Aziz MOQRICH CNRS IBDM – Marseille	Co-directeur doctoral

Affidavit

I, undersigned, **Michaël Pierrelée**, hereby declare that the work presented in this manuscript is my own work, carried out under the scientific direction of **Bianca H. Habermann** and **Aziz Moqrich**, in accordance with the principles of honesty, integrity and responsibility inherent to the research mission. The research work and the writing of this manuscript have been carried out in compliance with both the French national charter for Research Integrity and the Aix-Marseille University charter on the fight against plagiarism.

This work has not been submitted previously either in this country or in another country in the same or in a similar version to any other examination body.

Marseille, February 15, 2021

Abstract

Dynamic pathways regulate gene expression by series of genes and proteins interacting with each other. They can be identified by first mapping dysregulated genes from differential expression analysis on biological networks. Second, subnetwork extraction identifies regions of the network enriched in dysregulated genes. However, most methods build or extract static networks and thus, dynamics of pathways are lost.

Multilayer networks have been introduced to combine multiple data types and factors. In this thesis project, I developed a method that combines time-course gene expression datasets and multilayer networks, creating so-called *temporal multilayer networks* (tMLNs). Each layer represents one time-point as a biological network with dysregulated genes. Layers are linked to each other following the time axis. To predict dynamic pathways, I adapted classic subnetwork extraction to tMLNs. I implemented this approach in the Cytoscape app **TimeNexus**. I tested it on a yeast dataset to evaluate its efficiency to extract key cell-cycle regulators, as well as on a mouse dataset to identify subnetworks involved in the inflammation of sensory neurons.

In a side project, I explored the lipid metabolism of the microalga *Chlorella* sp. HS2. Differential expression analysis showed that the overflow of metabolic co-factors is likely to induce a production of lipids under salt water.

TimeNexus is the first method to extract subnetworks from tMLNs.

Keywords: computational biology, RNA-sequencing, temporal multilayer networks, time-course datasets, pathways, interactomes.

Résumé

Les voies cellulaires dynamiques régulent l'expression génétique par des séries de gènes et de protéines qui interagissent entre eux. Elles sont déterminées en commençant par marquer les gènes dérégulés, identifiés par l'analyse de l'expression différentielle, sur des réseaux biologiques. Ensuite, l'extraction de sous-réseaux identifie des régions enrichis en gènes dérégulés. Cependant, la plupart des méthodes construisent ou extraient des réseaux statiques et donc, la dynamique est perdue.

Les réseaux multicouches peuvent combiner plusieurs types de données et facteurs. Dans ce projet de thèse, j'ai développé une méthode qui combine des données temporelles d'expression génétique à des réseaux multicouches, créant ainsi ce que l'on appelle des *réseaux multicouches temporels* (tMLNs). Chaque couche est un réseau biologique avec des gènes dérégulés à un point temporel. Les couches sont reliées entre elles en suivant l'axe temporel. Pour prédire les voies cellulaires, j'ai adapté l'extraction classique de sous-réseaux aux tMLNs. J'ai implémenté cette approche dans l'application Cytoscape **TimeNexus**. Je l'ai testée sur des données de levure pour évaluer son efficacité à extraire les principaux régulateurs du cycle cellulaire, ainsi que sur des données de souris pour identifier les sous-réseaux impliqués dans l'inflammation des neurones sensoriels.

Dans un projet parallèle, j'ai exploré le métabolisme des lipides de la microalgue *Chlorella* sp. HS2. L'analyse de l'expression différentielle a montré que le surplus de cofacteurs métaboliques induirait une production de lipides dans l'eau salée.

TimeNexus est la première méthode d'extraction de sous-réseaux à partir de tMLNs.

Mots-clés : biologie computationnelle, séquençage à ARN, réseaux multicouches temporels, séries temporelles, voies cellulaires, interactomes.

Acknowledgment

First of all, I would like to thank Bianca for her warm-hearted and trustful supervision as well as, in particular, her comprehensive proof-reading of the thesis. This PhD was a great experience. I could enjoy it as much as possible, especially the opportunities given around research, such as training and associative life. It would have been difficult to do this PhD in another team and lab. I hope my scientific contributions will help for the future of the team.

The Habermann's team was kind and friendly. Even if the last year did not allow many social interactions, I thank all of you for this time and in particular, Fabio. We survived together to the time when we were only two, with jokes and interesting debates about the definition of a meal. I am happy to know that you will become the senior PhD candidate after my departure. Thank you a lot to Maxime and Laurent to proof-read the thesis in a short time!

I also give a warm thanks to Aziz and his team. It was exciting to follow the scientific questioning and progresses. Beyond having fed me, all the team was truly supportive. I regret that we did not have more opportunities to work together. I think to Lucie who was supposed to be my co-PhD. It was not the case in practice, but in heart. *Xoxo* to Anissa and Catarina!

More generally, IBDM was a friendly lab with a specific attention to the PhD candidates.

My PhD project actually started in South Korea, during my master's internship. It ended up with the publication I present in this thesis. I have a warm thought to Giao, who helped me to survive to this travel, as well as the friendship of Jin-Ho.

Par ailleurs, je souhaiterais remercier les membres du réseau *NetBio* qui m'ont aidé pour la partie RNA-seq de mon doctorat : Nathalie Vialaneix, Guillem Rigaill and Marie-Laure Martin-Magniette. J'en profite pour remercier l'excellent tutorial sur le RNA-seq d'Ignacio Gonzalez qui m'a permis de découvrir aisément ce monde fabuleux.

J'ai eu l'opportunité d'être bénévole pour plusieurs associations, dont Hippo'Thèse et la Confédération des Jeunes Chercheurs. Elles m'ont beaucoup appris sur les autres et moi-même, j'en retire une grande expérience. Je remercie vivement les personnes avec qui j'ai eu la chance de travailler.

Je tiens à remercier ma sœur, Aurélie, pour m'avoir hébergé durant ce difficile 2e confinement.

Enfin, merci à Merveille de son soutien et son amour durant ces 3 années, malgré la distance. J'attends avec impatience le moment où je te retrouverai.

Note: I thank the online tool Excalidraw *to enable me to draw the figures of this thesis.*

Abbreviations

Bioinformatics terms	
API	Application Programming Interface
CPM	Counts Per Million
DE	Differentially Expressed
DEA	Differential Expression Analysis
DEG	Differentially Expressed Gene
FPKM	Fragments Per Kilobase of transcript per Million fragments mapped
GO	Gene Ontology
GSEA	Gene Set Enrichment Analysis
KO	KEGG Orthology
LFC	Log2-fold change
MLN	Multilayer network
PCSF	Prize-Collecting Steiner Forest
PDI	Protein-Protein interaction
PPI	Protein-DNA interaction
RLE	Relative Log-Expression
rlog	Regularized-log transformation
RPKM	Reads Per Kilobase of exon model per Million mapped reads
TC	Total Count
tMLN	Temporal Multilayer Network
TMM	Trimmed Mean of M values
VST	Variance-stabilizing transformation
Molecules	
ATP	Adenosine Triphosphate
cDNA	Complementary DNA
DNA	Deoxyribonucleic Acid
dUTP	Deoxyuridine Triphosphate
mRNA	Messenger RNA
NAD(H)	Nicotinamide Adenine Dinucleotide
NADP(H)	Nicotinamide Adenine Dinucleotide Phosphate
RNA	Ribonucleic Acid
rRNA	Ribosomal RNA
TF	Transcription Factor
Molecular techniques	
ChIP	Chromatin Immunoprecipitation
PCR	Polymerase Chain Reaction
qPCR	Quantitative PCR
Y2H	Yeast Two Hybrids
Statistics	
ANOVA	Analysis Of Variance
AUC	Area Under the receiver operating characteristic Curve
BH	Benjamini-Hochberg
FDR	False Discovery Rate
FWER	Family-Wise Error Rate
GLM	Generalized Linear Model
Log	Logarithm / Logarithmic
LRT	Likelihood Ratio Test
M	Million
NB	Negative Binomial
qCML	Quantile-adjusted Conditional Maximum Likelihood
QL	Quasi-Likelihood
ROC	Receiver Operating Characteristic

Table of contents

List of figures

Introduction

1 Pathways are dynamic

(Jacob and Monod, 1961) demonstrated that the Lactose system was not inhibited at the enzyme level, but at the gene level by a "regulator gene". This gene was producing a "cytoplasmic product" repressing the synthesis rates of the enzymes by repressing gene expression with a high specificity. They showed that this product could only be a dedicated protein, which was inactivated in the presence of a particular molecule (an "inducer"), allowing the end of the repression. The repressing protein will be latter included into the family of *transcription factors*, i.e. proteins regulating gene expression. They predicted that a type of RNA, called "messenger RNA" (mRNA), would serve as intermediate molecules between the genes and the proteins. Doing so, Jacob and Monod built the first piece of *gene regulatory networks* by showing that genes control other genes and eventually, the proteins involved in cellular pathways. This work also laid the foundation of technologies used to measure gene activity: qPCR and, the one I will explore in this thesis, the (m)RNA-sequencing. Indeed, as genes regulate protein activities by adapting their expression, mRNA levels depend on both gene- and protein-activity levels.

Genes and proteins are classified into biological processes called pathways. Networks interpret pathways as series of interactions between molecules. Endogenous pathways are related to internal processes, e.g. cell cycle, while exogenous pathways answer to environmental perturbations, e.g. inflammation. I illustrated the analysis of both pathway types in (Pierrelée *et al.*, 2020). In a network with all molecules of the cells, pathways are assumed to be subnetworks of molecules and interactions. Thereby, one can determine active pathways by searching for subnetworks showing dysregulations when compared to a reference without activity. Differential expression analysis of RNA-sequencing data is the most common approach to detect these dysregulations at the gene level.

Pathways are intrinsically dynamic: interactions are not instantaneous and processes can have multiple effects over time. This aspect should not be forgotten when exploring them. Yet, RNA-sequencing measures gene expression at a given time. It is a snapshot of the cell state. Therefore, time-course experiments give a series of snapshots, enabling to measure the individual steps of pathways.

This introduction is organized as follows (**Figure 1**). In section 2, I present how differential expression analysis from RNA-sequencing datasets enables to identify dysregulated genes. Common approaches can be used to explore the list of dysregulated genes, but they fail to determine pathways (section 3). Instead, one can build networks (section 4) and then, identify regions of interests within networks (section 5). Yet, these networks are static and do not enable to find dynamic pathways. To solve this issue, I present temporal multilayer networks and discuss how to extract subnetworks from them (section 6).

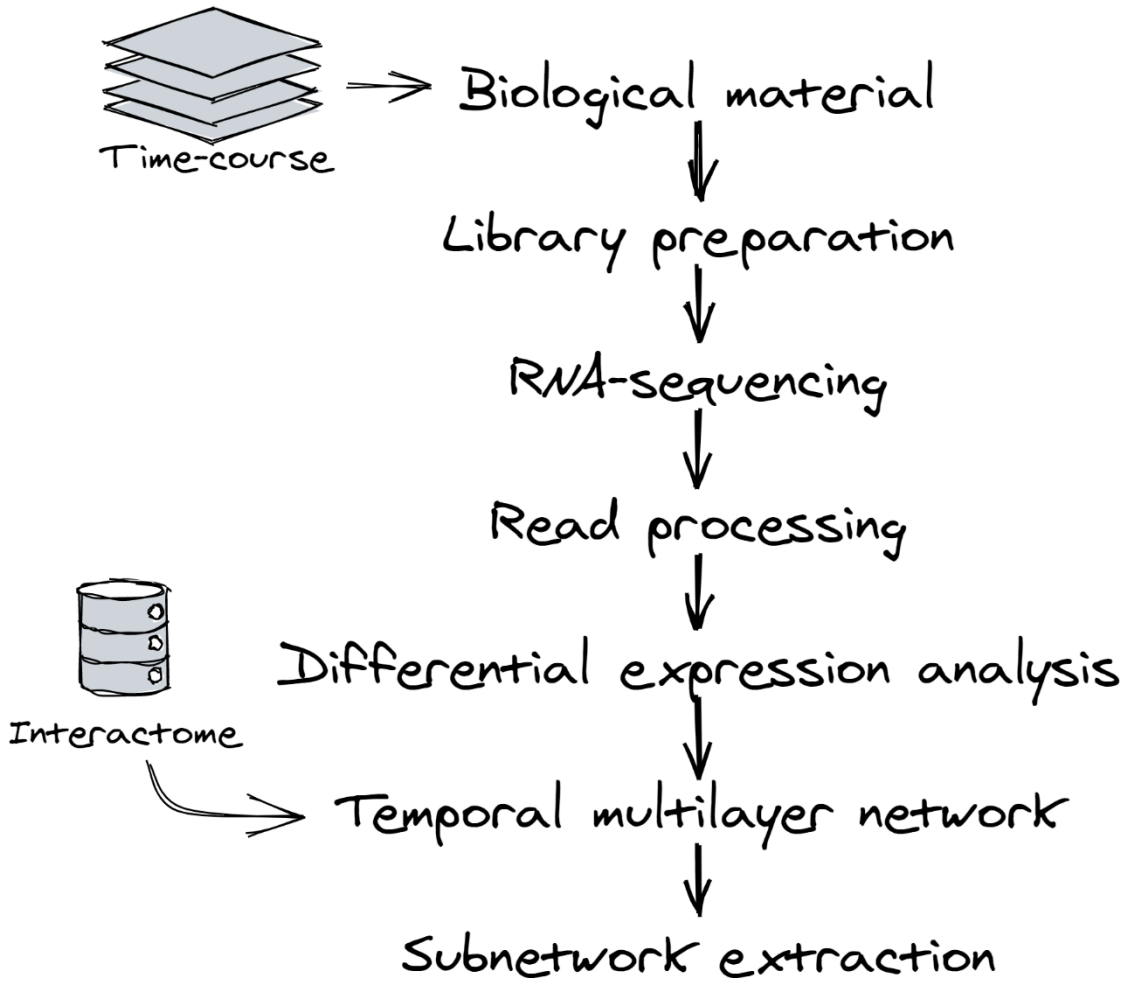

Figure 1. Workflow to extract pathways from time-course RNA-sequence dataset. *Object models from* (Tjang, 2020).

2 Differential expression analysis of RNA-sequencing datasets

2.1 Overview of RNA-sequencing

RNA-sequencing enables to measure the *relative abundance* of mRNAs in a sample (e.g. a cell culture, a tissue or an entire small organism). This technology requires a complex workflow divided in roughly 5 main steps (**Figure 2**). First, the RNA fragments are extracted from the biological material. Several RNA fragments can come from the same transcript, which carries information from a genomic sequence. *Genes* and *transcripts* are herein interchangeable. Then, the samples are prepared for sequencing. A prepared sample is called *library*. Third, the RNA fragments are sequenced to get their sequence as well as their abundance. Each sequenced fragment is called a *read*. Fourth, the data are processed to identify the transcripts related to each read. Reciprocally, counting the number of reads assigned to each transcript gives the transcript abundance. One read gives one *read count* for the associated transcript. The final step is the *differential expression analysis* (DEA) to compare read counts between samples. It identifies the genes which are *differentially expressed* (DEG) in one condition compared to another. In the next sections, I present downstream steps to explore these results.

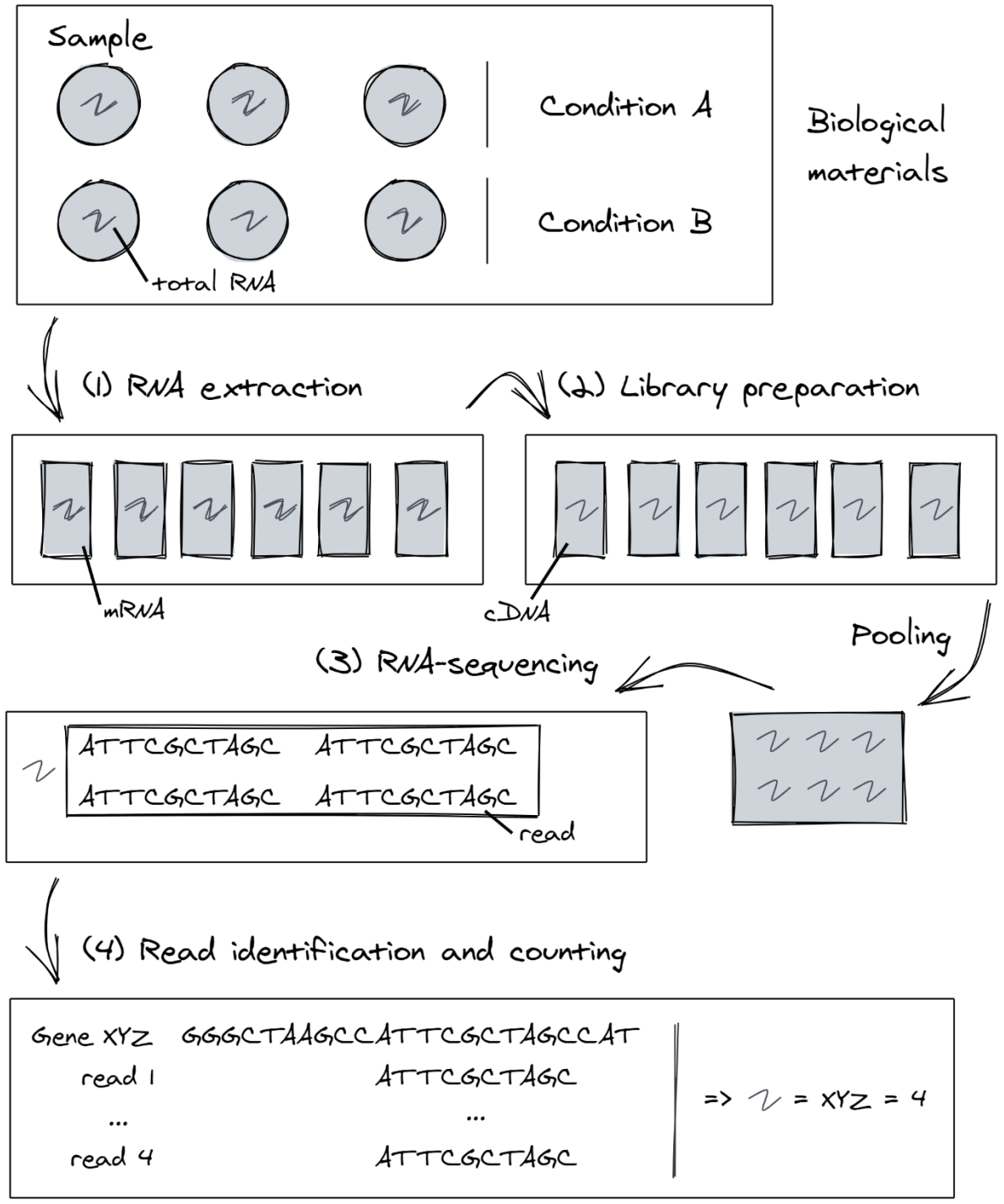

Figure 2. Workflow to measure transcript abundance from RNA-sequencing. *(Top panel) An experiment has 2 conditions with 3 samples each. (Panel 1) mRNAs are extracted from total RNAs of each sample. (Panel 2) mRNAs are converted into cDNA and prepared for sequencing. (Panel 3) Libraries are pooled together to sequence all of them at the same time. RNA-sequencing identifies the sequence of each fragment, giving one read per fragment. The panel shows one gene (transcript) which has been sequenced 4 times, giving 4 reads. (Panel 4) Data processing aligns the reads on a genome to identify the gene related to the reads. Another step counts the number of reads for each gene to have gene abundance.*

The wet-lab steps of RNA-sequencing can generate *confounding effects*. A confounding effect is an uncontrolled variable which prevents to test the causality between controlled variables and their effects during the experiment. Indeed, each sample is

independently prepared during library preparation (second step). The preparation can have different efficiencies caused by technical manipulations. Further developments are expected to solve this issue (Alpern *et al.*, 2019).

2.2 Data processing of *de novo* RNA-sequencing

The goal of the data processing is to get the gene abundance from the raw reads. For an organism with a sequenced and annotated genome, on can apply a simple workflow. In (Pierrelée *et al.*, 2020), I processed read counts for yeast and mouse datasets. To do so, I first applied **STAR** (Dobin *et al.*, 2013) to *align* the reads on the genome. It enables to assign the reads to the transcripts or the genes. Then, **featureCounts** (Liao, Smyth and Shi, 2014) *counts* the reads assigned to each gene. It returns a count table used by the differential expression analysis (see section 2.3). Both tools are popular and efficient.

If there is no annotated genome (or transcriptome) available for mapping, the workflow is more complicated. An additional step is required to build a transcriptome, so-called *de novo assembly*. I applied the workflow based on **Trinity** (Haas *et al.*, 2013) to process the microalga dataset of (Yun *et al.*, 2020). The first step is to *assemble* the reads into a transcriptome with **Trinity**. This transcriptome has thousand more *de novo* transcripts than expected; most of them are redundant or irrelevant. Biological transcripts are not always fully sequenced or too difficult to assemble, e.g. because of repetitive patterns in their sequence. Then, **Bowtie2** (Langmead and Salzberg, 2012) aligns back the reads on the transcriptome from which **RSEM** (Li and Dewey, 2011) estimates the counts at the gene level. The next step is to identify the genes by finding their orthologs from annotated species. It enables to share annotations and thus, explore the results of the differential expression analysis. This step is quite challenging. I applied the **Trinotate** pipeline as presented in (Bryant *et al.*, 2017). It calls a series of tools to annotate the genes and find the best orthologs. In particular, **TransDecoder** (Haas *et al.*, 2013) predicts protein sequences from the gene sequences. **BLASTP** searches for orthologs from the predicted proteins and **BLASTX** from the gene sequences (Camacho *et al.*, 2009). Furthermore, I converted the orthologs into **KEGG** orthologies (KOs) (Kanehisa, 2000) which are cross-species orthologous groups (see section 4.3.2.2). A gene must have at least one KO from **BLASTP** or **BLASTX**. If there are KOs for both, they should be the same. In my dataset, this was often the case. As **Trinity** produces many redundant genes, those having the same KO were merged if their dysregulation was equivalent (i.e. up- or down-regulated). Only few genes had inconsistent dysregulations. In this case, I filtered out the genes with the lowest e-value (i.e. quality of the ortholog prediction). The use of KO has many advantages. First, it enables to exploit all predicted orthologies, rather than being restricted to a given species. Second, it avoids to select one gene sequence among a group of redundant sequences at an early step. A tool such as **SuperTranscripts** (Davidson, Hawkins and Oshlack, 2017) after Trinity removes redundancies by defining a representative sequence for a group of redundant genes. Yet, a higher diversity of sequences increases the chances to get orthologs at the expense of runtime. This workflow had an efficiency estimated at 89% by counting the number of genes that the assembly is expected to find.

2.3 Differential expression analysis

2.3.1 Preparing read counts

2.3.1.1 Filtering out genes with low expression

A problem with RNA-sequencing is that the datasets have a high number of genes with zero counts or with a very low number. Consequently, the negative-binomial (NB) distributions (see section 2.3.2) cannot model well the data, biasing the estimation of parameters by the methods based on NB models. Moreover, when comparing two conditions, the statistical tests will eventually give p-values at 1 for the genes with so few counts. This is why we observe an enrichment in high p-values when we plot the distribution of raw p-values. As the procedures to control the false discovery rate (FDR) depend on the total number of computed tests (see section 2.3.3), these unnecessary tests also decrease the overall statistical power of the experiment (Bourgon, Gentleman and Huber, 2010). From the theoretical point of view, these two points justify to filter out the genes with counts under a given cutoff. The choice of the filtering strategy raised discussions among the research community.

The filtering can be useful if there are few DEGs or for technologies generating many low counts by improving the FDR control, without necessarily increasing the recall (Rigaill *et al.*, 2016). It can be either based on a minimal level of counts within few conditions (Chen, Lun and Smyth, 2016), or the sum or the mean of counts across the samples (Rau *et al.*, 2013). One can set the threshold by iteratively searching for the value optimizing the number of DEGs, without filtering more than 20% of the genes (Bourgon, Gentleman and Huber, 2010; Sha, Phan and Wang, 2015). It is worth noting that filtering does not add much if there are many or enough DEGs to explore. In this context, the filtering step can be removed to simplify the workflow.

2.3.1.2 Normalizing samples

Besides the gene expression, the read counts from RNA-sequencing have three main sources of variation: library size (Mortazavi *et al.*, 2008) from the *sequencing effects* (i.e. total number of counts in one sample), *gene length* (Oshlack and Wakefield, 2009) as well as *GC-content* (Zheng, Chung and Zhao, 2011). GC-content represents the ratio of the Guanine and Cytosine bases in the sequence, compared to the Adenine and Thymine (or Uracil for RNA) bases. Indeed, long genes and GC-rich sequences tend to accumulate more reads, all things being equal. If a gene has higher read counts, the statistical power related to this gene increases, promoting its identification as a DEG (Oshlack and Wakefield, 2009). To compare the genes and samples to each other, one should remove these sources of variations. This procedure is called *normalization*. To do that, many methods have been proposed. I will discuss a few of them here which are popular. Note that filtering should be applied after normalization (Lin *et al.*, 2016).

Within-sample normalization consists of removing biases when comparing two genes to each other. As DEA only compares samples to each other, this normalization would not be required. Yet, (Risso *et al.*, 2011) confirmed that GC-content biased the fold-changes and p-values from the DEA, with a positive correlation between DEGs and GC-content. To account for length and GC-content effect, the authors proposed a

normalization method included in the package **EDASeq**. However, to my knowledge, this entire issue seems to have been ignored afterwards.

A simple and common method for *between-sample normalization* is **RPKM** (Reads Per Kilobase of exon model per Million mapped reads) (Mortazavi *et al.*, 2008). It normalizes according to the library size, i.e. the sum of counts in a sample. The literature strongly advises against its use because it is easily biased (Bullard *et al.*, 2010; Zheng, Chung and Zhao, 2011; Dillies *et al.*, 2013; Maza *et al.*, 2013; Lin *et al.*, 2016). One of the most popular method is **TMM** (Trimmed Mean of M values) developed for **edgeR** (Robinson and Oshlack, 2010) that I applied for (Pierrelée *et al.*, 2020). It normalizes the samples by adjusting the count distribution. TMM is a robust method (Dillies *et al.*, 2013; Maza *et al.*, 2013; Rapaport *et al.*, 2013; Seyednasrollah, Laiho and Elo, 2015; Zyprych-Walczak *et al.*, 2015). Normalization works well in general, but unexpected conditions can break the assumptions used to distinguish between technical and biological effects (Bullard *et al.*, 2010; Lin *et al.*, 2012; Lovén *et al.*, 2012; Soneson and Delorenzi, 2013; Roca *et al.*, 2017; Evans, Hardin and Stoebel, 2018). Therefore, the user needs to be sure that these assumptions hold before to start the differential expression analysis.

One critical assumption of standard normalization tools is the lack-of-variation hypothesis. It assumes that most of the genes are not differentially expressed in the experiment. While acceptable in most cases, the lack-of-variation hypothesis does not hold for some datasets which require another normalization approach. It was the case in (Yun *et al.*, 2020). The studied microalgae HS2 underwent massive phenotypic and metabolic changes when grown in salt water. Therefore, I applied **SVCD** to normalize the counts (Roca *et al.*, 2017). This method normalizes the counts without assuming the lack-of-variation hypothesis. It first computes within-condition normalization factors. Then, iteratively, it applies a statistical test to identify the non-differentially expressed genes (DEGs) by comparing the average of each condition. At each iteration, it removes the genes showing the highest variations until convergence. It then computes the final normalization factors from the pool of non-DEG. Interestingly, the method does not assume any count distribution (e.g. Poisson or Negative-Binomial). The authors claimed that their approach can work for any proportion of DEGs within the gene population, as soon as there are enough genes from which we can estimate the normalization factors. Nonetheless, as for **TMM**, it still assumes that normalization factors for non-DEGs can also apply to the DEGs and the proportion of counts is not distorted.

The normalization methods produce normalization factors used by the tools for the differential expression analysis. They do not directly update the read counts. Indeed, the statistical tools of the differential expression analysis need to estimate some parameters from the raw counts, otherwise the normalized counts would bias the estimation.

2.3.2 Testing differential expression

Statistical inference generalizes results of a given experiment to any other experiments. It *estimates* statistical properties of samples for each condition, in particular the

variance. This variance can only be computed with *biological replicates*, i.e. several samples for the same condition. Contrary to *technical replicates*, biological replicates are different individuals (e.g. mouse, cell culture), but which underwent the same treatments or manipulations (i.e. factors). If individuals are more different, then results are more generalizable. In practice, individuals with the same genetic background are considered as biological replicates. The individual variability would be enough to generalize the results to the population.

Differential expression analysis (DEA) identifies the genes with variation of their expression, in a given condition compared to a reference condition, by applying statistical inference (**Figure 3**). They are called *differentially expressed genes* (DEGs). DEA measures 1) the *size effect*, expressed as a *log2-fold change* (LFC), and 2) the *significance*, evaluated from a probability called *p-value*. For a simple pairwise comparison, the fold change is the ratio of the gene abundance between a given condition and a reference condition. It is on a log2-scale. The p-value is the probability to observe a difference between conditions, while assuming the conditions are equivalent. In the context of pathways, I employ the adjective *dysregulated* when the gene is differentially expressed.

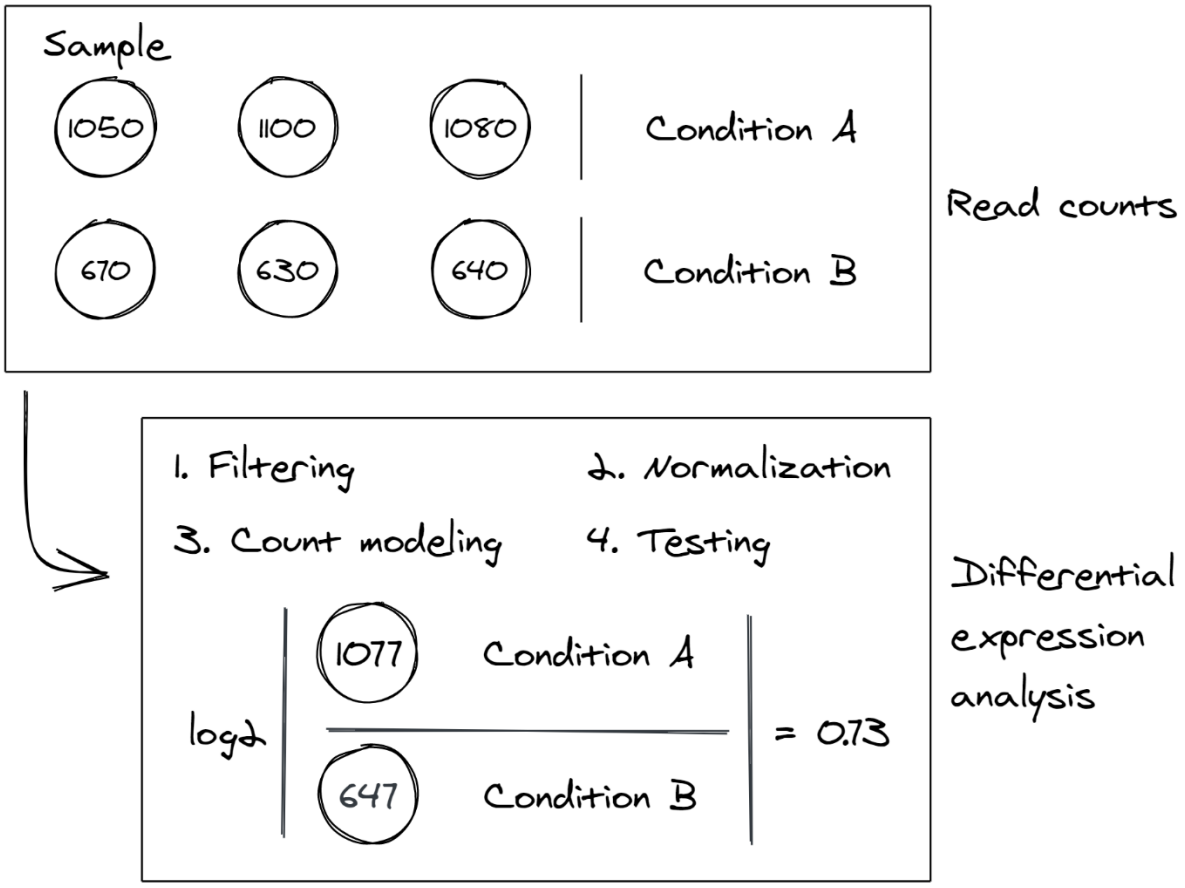

Figure 3. Differential expression analysis measures levels of dysregulation. *(Top panel) RNA-sequencing generates read counts for each gene in each sample. This example shows the reads for one gene in an experiment with 2 conditions and 3 biological replicates each. (Bottom panel) Differential expression has 4 main steps which finally give the size effect (log2-fold*

change) by comparing two conditions. Here, the gene is up-regulated with a size effect of 0.73 (i.e. a fold-change of 1.7) in condition A compared to condition B.

EdgeR is a popular tool to run differential expression analysis. It was successively improved to solve the limitations of the previous versions. The first version is the so-called **classic-edgeR** (Robinson and Smyth, 2007, 2008; Robinson, McCarthy and Smyth, 2010). It models the read count with a *negative-binomial model* which enables to define the variance as a function of the mean and a *dispersion parameter*. While a *Poisson distribution is* more natural to model the independent probability of a read being assigned to a RNA fragment, it cannot model a distribution where the variance depends on the mean. A *common dispersion parameter* is first computed from all genes using a *quantile-adjusted conditional maximum likelihood* (qCML). This enables to share information between genes. Yet, not all genes have the same dispersion. A *gene-specific dispersion parameter* is computed using a *weighted likelihood function*, but there are not enough samples to correctly estimate this dispersion. Therefore, it should be corrected. To do so, an empirical Bayesian approach "shrinks" the gene-specific dispersions toward the common dispersion. This shrinking enables to smoothly estimate the gene-specific dispersion parameters, without too strong or too weak correction. That is why the dispersion estimation is said to be *moderated*. Finally, an exact test, similar to the Fisher's exact test, compares the conditions using the estimated means and dispersions.

However, the first version of **edgeR** only manages experimental designs with a single factor. To process more complicated designs, (McCarthy, Chen and Smyth, 2012) implemented the *generalized linear models* (GLM) to **GLM-edgeR**. The principle is the same as above. A *locally weighted profile-adjusted likelihood* function, computes a dispersion trend across the genes and a second function, called *Cox-Reid profile-adjusted likelihood*, estimates the gene-specific dispersion parameters which are then shrinked toward the trend. (Zhou, Lindsay and Robinson, 2014) completed this approach to account for outliers (i.e. highly-variable genes). **GLM-edgeR** applies a *likelihood ratio test* (LRT) to correctly approximate the exact test of the **classic-edgeR**.

quasi-edgeR extended **GLM-edgeR** by developing a *quasi-likelihood* (QL) framework for the statistical test of the differential expression (Lund *et al.*, 2012). Its main advantage is to compute the uncertainties of the estimated variances and account them for the statistical test. The latter is thus more robust to errors in the modeling of the experimental design. (Phipson *et al.*, 2016) added an option to the **quasi-edgeR** QL fitting function to increase the robustness against outliers. It increases the shrinking for the main body of genes, while decreasing it for the outliers. Moreover, it decreases the shrinking for genes with null counts. Without robustness, one could call genes differentially expressed if they have very high or low dispersions across the samples. Indeed, Phipson and colleagues showed that the robust QL increased the power and the false discovery rate (FDR) control in presence of outliers.

Regardless of its version, **edgeR** is one of the best tool with low false negatives and false positives as well as an accurate FDR control (Soneson and Delorenzi, 2013; Rajkumar *et al.*, 2015; Seyednasrollah, Laiho and Elo, 2015; Rigaill *et al.*, 2016; Schurch

et al., 2016; Holik *et al.*, 2017; Lamarre *et al.*, 2018; Li *et al.*, 2020). It should however be used with its robustness feature. Therefore, I applied **quasi-edgeR** in (Pierrelée *et al.*, 2020).

Differential expression analysis aims to test thousands of genes with significant changes. By setting a significance level on the raw p-values, it is certain to have false positives among all DEGs (Bender and Lange, 2001). This is called the *multiple-hypothesis testing problem*. Solving this issue means controlling the number of false positives among the gene population for a given significance level, at the expense of false negatives. It is not possible to control for both errors, but at least controlling the false positives avoids spurious conclusions. From raw p-values, the controlling procedures produce *adjusted p-values* to select the DEGs according to the threshold of the significance level (Wright, 1992). The adjusted p-value is a *random variable*, not a probability. An *adjusted p-value* is a new value which accounts for the raw p-value with regard to the whole set of tested hypotheses. The *Benjamini-Hochberg (BH) procedure* controls the *false discovery rate* (FDR), i.e. the expected proportion of false positives among the DEGs (Benjamini and Hochberg, 1995). It gives an upper bound on the FDR to adjust the p-values. This is the most common approach in RNA-sequencing data analysis, but it generates rough FDR estimations.

2.3.3 Calling dysregulating genes

Usually, a gene is called *differentially expressed* if it meets one or two conditions: its (adjusted) p-value is lower than a FDR cutoff and/or its size effect is lower than an (absolute) LFC cutoff. On a theoretical point of view, there is no reason to justify an LFC cutoff, except if one wants to test a gene by qPCR, as qPCR is less sensitive than RNA-sequencing. Indeed, the level of dysregulation in a pathway does not depend on the LFC level (Hughes *et al.*, 2000; Ideker *et al.*, 2002; Subramanian *et al.*, 2005). Therefore, I did not apply an LFC cutoff in (Pierrelée *et al.*, 2020; Yun *et al.*, 2020).

Nonetheless, the user can apply an LFC cutoff according to a *volcano plot* (LFC vs. adjusted p-value of genes) or the distribution of LFCs. Note that if one applies an LFC cutoff, then it is equivalent to run multiple new statistical tests. These tests are said *post hoc* and can break the FDR control by inducing a *selection bias*. Therefore, it is necessary to control the FDR by considering both FDR and LFC cutoffs with a post hoc inference tool such as **cherry** (Goeman and Solari, 2011) or **sanssouci** (Blanchard, Neuvial and Roquain, 2020).

2.4 Testing methods dedicated for time-course datasets

One can take snapshots of gene expression over time by applying RNA-sequencing on successive time-points in an experiment. One can measure the expression dynamics by adapting the differential expression analysis to identify the genes varying over time. A claimed advantage is that tools dedicated to time-course datasets could use the dependencies (i.e. causality) between time-points to increase the statistical power (Spies *et al.*, 2019).

A main aspect with time-course datasets is the sampling rate, i.e. the number of sampled time-points and their frequency. For RNA-sequencing, if the frequency is

high, e.g. time-points every few minutes, then the time-points are strongly dependent on each other. It means that one can assume a direct causality between two consecutive time-points. In the other hand, it is harder to assume a causality with a sampling every day because the system evolves too much between time-points. Also, in this case, one can simply consider time as a categorical variable. A higher frequency enables to consider it as a continuous variable. (Bar-Joseph, Gitter and Simon, 2012) reviewed methods to analyze time-course datasets and gave recommendations concerning the sampling rate. A higher sampling rate will increase the number of detected DEGs varying over time or explore the kinetics of a process, but a higher number of replicates are necessary to identify precisely where the DEGs are varying. The sampling rate should also be higher shortly after a perturbation, where most of the gene expression varies. The time-points at the end of the experiment measure the permanent changes.

(Spies *et al.*, 2019) benchmarked 9 tools developed to identify differentially expressed genes over time from RNA-sequencing datasets. Among these tools, there is a popular tool for time series called **next-maSigPro** (Conesa *et al.*, 2006; Nueda, Tarazona and Conesa, 2014). This tool identifies the genes with a "non-flat profile" by using a negative-binomial model and a polynomial regression. Then, it tests the differences of expression profiles by applying an iterative regression to select the conditions with the highest changes. **next-maSigPro** is mainly distinct from **edgeR** in two aspects. First, it considers time as a quantitative variable, while the time is multifactorial in **edgeR**. This increases the number of variables and so, the false positives. Second, the tool computes the differential expression on the non-flat genes, thereby increasing the FDR control and the model fitting on gene expressions. In comparison, (Fischer, Theis and Yosef, 2018) developed **ImpulseDE2**. The tool models the gene expression profiles and the read counts with an *impulse model* and a negative-binomial model (as **edgeR**), respectively. The impulse model considers that the gene expression temporarily switches from a permanent to a transient state, before going back to the permanent state. The negative-binomial model brings "noise" to the impulse model. From the estimated models, the tool can then test the differential expression between two conditions. The authors did not consider its application to cyclic processes.

The benchmark of (Spies *et al.*, 2019) especially compared **next-maSigPro** and **ImpulseDE2** to pairwise comparisons computed by **classic-edgeR**. The authors used simulated dataset from NB models. In brief, the authors concluded that "pairwise methods" such as **edgeR** still had overall the best efficiency. **ImpulseDE2** could equal them, or outperform them with a high number of replicates, but it had lower performances with more than 8 time-points or with noise. On the contrary, **next-maSigPro** had a lower efficiency, but it was more robust to noise and could benefit from more time-points. Interestingly, selecting the DEGs by finding the overlaps between the results of 2 or 3 tools could increase the number of true positives without increasing the false positives. However, the most conservative tool will dominate the results in this approach.

In their study, (Spies *et al.*, 2019) used **edgeR** to compute pairwise comparisons and showed these tools were still competitive. However, they can enable more complex designs to test for any difference between time-points, as for ANOVA. Also, even if it

seems unused in RNA-sequencing, the authors could have tried **limma-voom** (Smyth, 2004; Law *et al.*, 2014) with a *natural regression spline*, which models time as a continuous variable, following (Smyth *et al.*, 2020, p. 49). For microarrays, **limma** showed the best performances compared to other tools dedicated to time-course datasets (Moradzadeh *et al.*, 2019).

Therefore, these dedicated methods are not useful to analyze differential expression from time-course datasets. These dedicated methods are less known and used than more general methods. To support their use, the community should have benchmarks focusing on making the best use of well-known tools such as **edgeR** and **limma**. They should include real and/or synthetic datasets to assess their performances. (Spies *et al.*, 2019) showed that the number of replicates and the sampling rate affected the performance. The ideal benchmarks should therefore test these different use-cases.

3 Analyzing dysregulated genes from time-course datasets

Differentially expression analysis (DEA) allows to select a subset of genes showing interesting behaviors. At this point, read counts were generated as well as a list of differentially expressed genes (DEG). This list can be large, making its exploration harder. A simple approach is to identify few genes of interest and after, to use them for further experiments. Instead, one can search for causes which explain the cell's phenotype by looking at the patterns within the DEG list. Grouping the genes into smaller highly-consistent subsets enables to detect such patterns. Three approaches are commonly used to do that: *clustering*, *functional enrichment* and *networks*. I will illustrate these two first approaches with time-course datasets in which time is a categorical factor. However, they work with any other categorical factor.

3.1 Clustering of time series

Clustering is an automatic method to define gene subsets from their expression profile. It can use the whole table of read counts and any factor. Many clustering methods have been developed. I will only present two of them dedicated to clustering time-course datasets from temporal RNA-sequencing or microarray data.

3.1.1 Mfuzz

(Futschik, 2003; Futschik and Carlisle, 2005; Kumar and Futschik, 2007) introduced the tool **Mfuzz** to cluster gene-expression profiles. Contrary to *hard clustering* were one gene is assigned to one cluster, Mfuzz applies a *soft clustering* algorithm where genes are shared between clusters. It enables clusters with similar profiles. Therefore, the clusters are more precise and more robust to noise. In my experience, it is difficult to find relevant clusters among all possible clusters. To decrease this number, one can only consider the DEGs. It is motivated by the fact that DEA tools can test the significant variations. Therefore, a cluster without significant variation would be less biologically relevant.

The main issue with clustering is to define the number of clusters. For **Mfuzz**, the authors suggested 3 options. Two of them consist of computing statistics decreasing when the number of clusters increases. The statistics are expected to decrease strongly after a given number. The optimal number of clusters is the highest statistics before the decrease is too strong. Yet, the user cannot apply these approaches if this drop happens just after a single cluster. This happens when the profiles are not dissimilar enough, e.g. if the number of time-points is low. On the contrary, if the drop is too soft, picking a value would be arbitrary. In practice, it is easier to determine a number of clusters when the number of genes within the dataset is low, e.g. by only considering the DEGs.

Finally, the authors suggested to apply functional enrichment on the clusters (see section 3.2). The enrichment is computed for each cluster and the clusters with the most relevant results are kept. Doing so, genes sharing the same profile are assumed to be co-regulated, i.e. a group of genes directly regulated by a transcription factor. This direct co-regulation should have roughly the same size effect and direction for each gene of the cluster. Though, two genes with the same profile are not necessarily co-regulated. This is a similar problem to the correlations used to infer networks (see section 4.2). Even if it is the case, it is harder to have clusters with few but consistent genes when there are few time-points.

The user should not focus too much on the number of clusters because it is too difficult to find a meaningful value. Instead, one should try to estimate if this number makes sense with regard to the biological context. To illustrate that, let's take an experiment with 4 time-points. In this experiment, observations suggest that the system is at a *normal state* for the first and last points, while the intermediary points are at a *perturbed state*. Thereby, one could observe genes starting with an up-regulation, then stabilizing and finally ending with a down-regulation. Also, genes can increase their expression after the 1st or 2nd time-point and then decrease it. It gives 3 clusters, more 3 others which are symmetric. In more, there are 6 clusters having permanently up- or down-regulated genes after the 1st, 2nd or 3rd time-points. There is of course a final cluster for non-regulated genes. In total, it should have 13 clusters in this experiment (**Figure 4**). One can then compute statistics for each cluster, find empty clusters and search for non-clustered genes showing unexpected variations. This method *fuzzifies* gene-expression profiles by converting their continuous values into fuzzy qualitative values. For example, -1.3 and +2.5 become "down-regulated" and "up-regulated". This greatly simplify the problem of assigning genes to clusters. Thus, it avoids to apply a more time-consuming approach proposed by Futschik and colleagues. In fact, it implies that clustering methods would be superfluous.

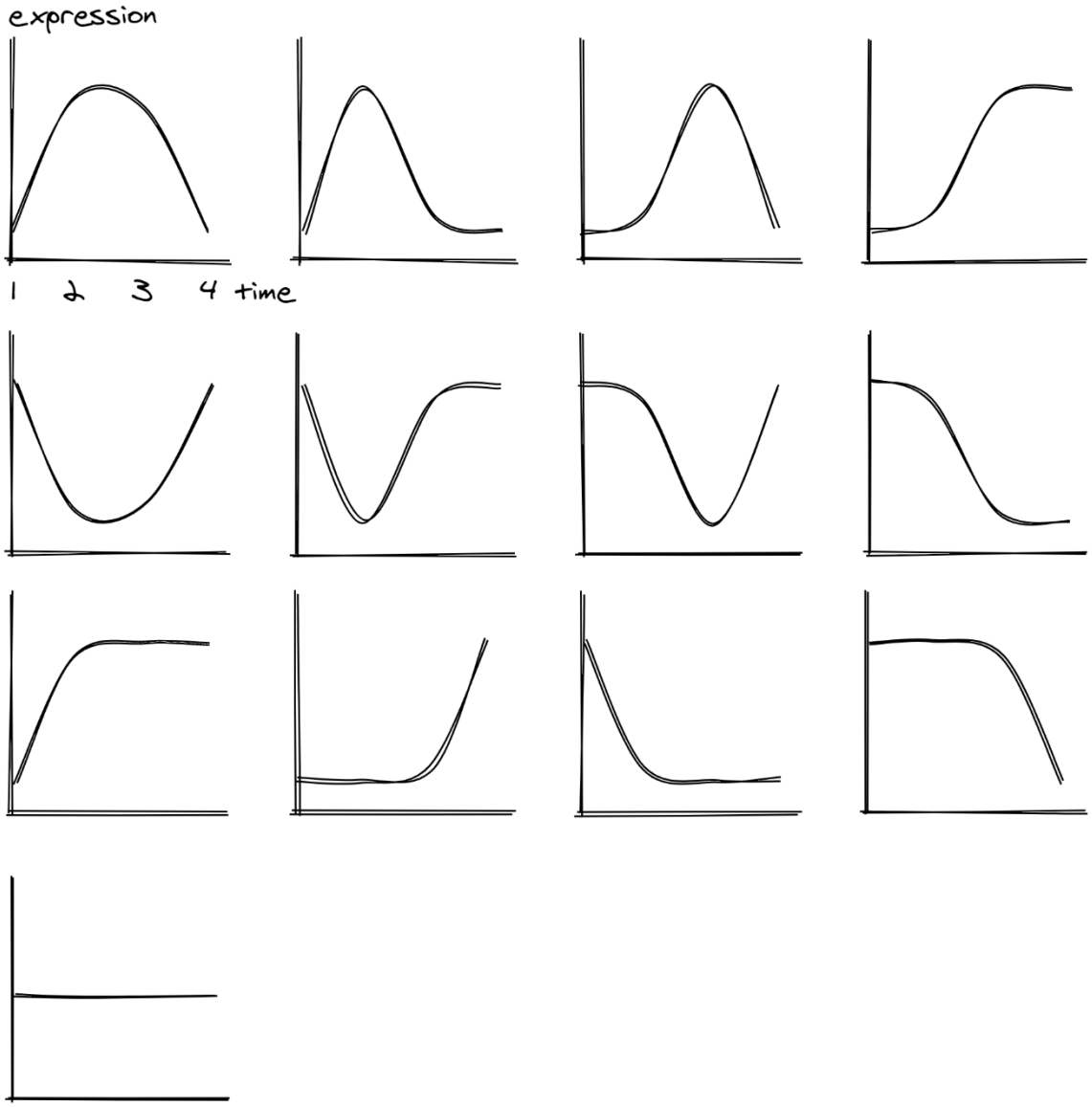

Figure 4. An experiment with a transiently perturbed system should generate 13 clusters. *Genes can be manually assigned to each cluster without a dedicated algorithm. In this example, there are 4 time-points. Only clusters with a single dysregulation (i.e. up- or down-regulation) or two opposite dysregulations (i.e. up- and down-regulations) are shown. If there are 4 time-points and 3 possible directions for dysregulations (up, down or constant), then 27 clusters are possible (3x3x3), but not all of them are always relevant. It would be more interesting to start with a smaller number as shown here and only after, increase this number if necessary.*

Mfuzz is highly demanding for the user because the user needs to tune each step without a clear aim. This is a general issue for the unsupervised clustering where experimental data are exploited to make decisions.

3.1.2 DREM

DREM is an approach developed by (Ernst *et al.*, 2007; Schulz *et al.*, 2012) for time-course datasets. The goal is to find the transcription factors causing the gene expression profiles by combining gene expression data and transcription factor (TF) binding data (i.e. sequence motifs or ChIP-sequencing). Briefly, at the first time-point,

all genes are in the same cluster. In the next time-points, if the gene expression profiles split into new branches, a statistical model assigns each branch to a new cluster. It gives a tree of nested clusters with a splitting based on the time. **DREM** assigns transcription factors to the clusters by using the TF binding data. These TF binding data constrain the number of clusters so the user does not have to set it. Contrary to **Mfuzz**, the authors recommended to filter out non-differentially expressed genes. The 2nd version of **DREM** especially enables to consider dynamic TF binding data and the expression levels of transcription factors. One limit of **DREM** is that the clusters cannot merge back after a split event. Moreover, the sampling rate should be large enough, otherwise it cannot associate the transcription factors to the clusters. Note that the authors claimed to use and build networks, but this is a misuse of language. Their method is a clustering method benefiting from two data types, while standard clustering methods only use one data type.

Above, I presented a strategy to group the genes, but the groups still lack a biological meaning. A common follow-up is to find overrepresented terms by applying functional enrichment.

3.2 Functional enrichment of gene lists

Functional enrichment aims to find gene sets with an overrepresented number of genes from an input list, for example the list of differentially expressed genes. A *gene set* is a list of genes annotated by a same *term*, such as a gene ontology (e.g. "nucleus"), a pathway (e.g. "cell cycle") or a co-regulation by the same transcription factor (e.g. "CREB"). Statistical tests assign a score to these genes sets to represent how many DEGs they have. The gene sets with the highest scores are called *enriched terms*. This approach is useful to explore consistent groups of genes within an input list, especially for large lists, or to quickly annotate the input list. Many methods were developed to build gene sets, weight the genes of the input list and test their enrichment. Even in 2009, 68 methods were already available to enrich gene sets (Huang, Sherman and Lempicki, 2009). The drawback of enrichment is that it heavily depends on the gene sets, the input lists and the algorithm (Huang, Sherman and Lempicki, 2009). Any change in the DEA method, the DEG calling or biases in RNA-sequencing will strongly affect the results of the functional enrichment.

At this point, functional relations between genes are available, but they do not enable to find cellular mechanisms that generated them. To answer this question, physical interactions between genes must be considered. This will be the topic of the next part.

4 Building biological networks

4.1 Monolayer networks

Networks model systems through their objects and the relationships between these objects (**Figure 5**). The objects are represented as *nodes* and the relationships as *edges*. Both are constitutive *elements* of the network. A network *is* not the system, but an *approximation* of it. This consideration is trivial but easy to forget in the context of networks. Indeed, they enable to intuitively represent the system under study. Such

systems are for instance mechanistic biological processes called pathways. In this context, the edges (i.e. physical interactions) are the main element of pathways because the nodes (i.e. molecules) alone cannot form a pathway. The aim is to use networks to represent pathways.

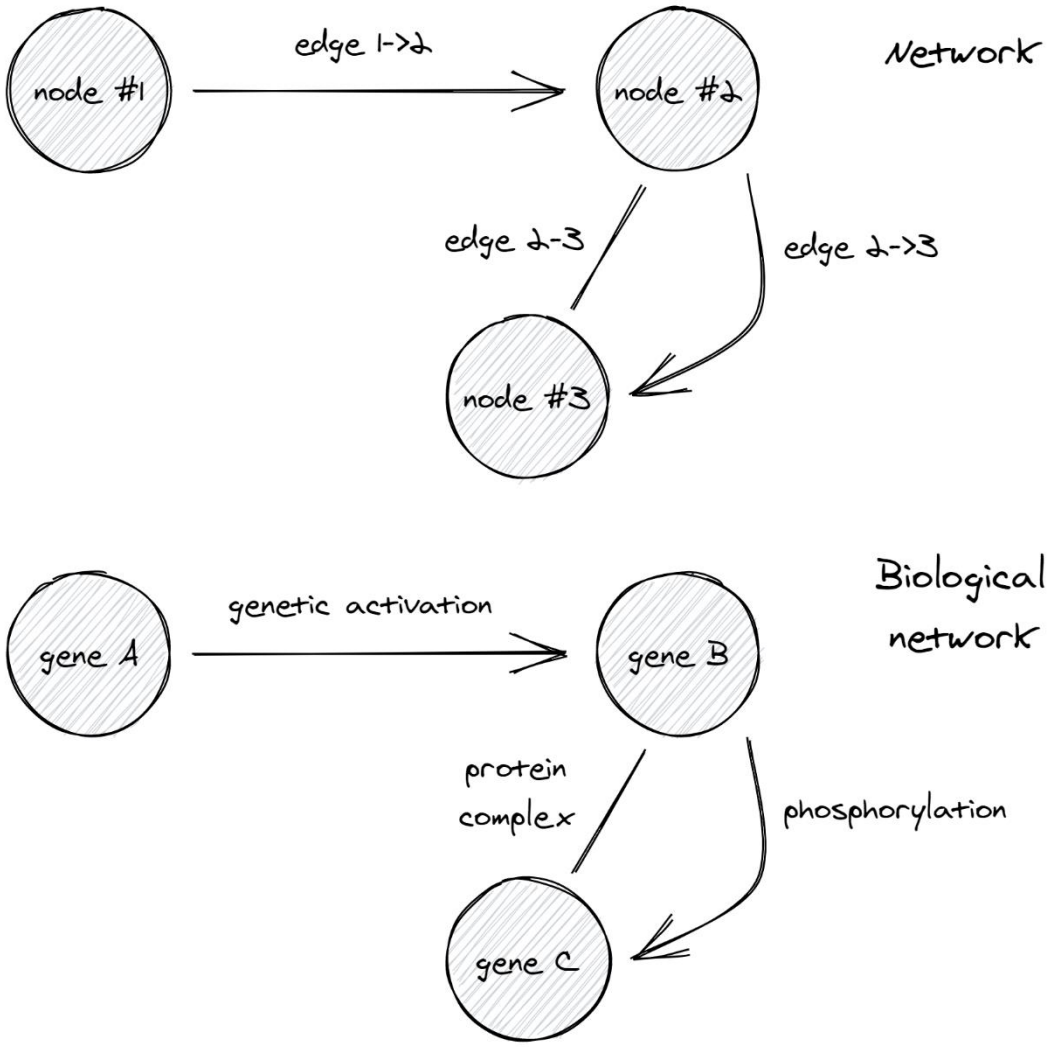

Figure 5. Networks can model any type of relationship between objects of a system. *The top network is a general network. The system's objects are represented as nodes (here 3 nodes) and the relations as edges. Edges can be directed, e.g. node #1 as a relation with #2 but without reciprocity, or undirected, e.g. nodes #2 and #3 which have a relation to each other. In this example, there are multi-edges between nodes #2 and #3 because two edges link both nodes with the same direction (i.e. an undirected edge has both opposite directions). The bottom network is a biological network. A biological process is modeled as a network. It contains one protein-DNA interaction as a genetic activation from the transcription factor "gene A" to the gene B. Biological networks generally consider a gene and its encoded protein as the same entity. This greatly simplifies the modeling. Proteins of gene B and C form a protein complex such as they bind to each other without a particular direction. Yet, gene B can catalyze a chemical reaction which phosphorylates gene C. Therefore, this edge is directed. This biological network is a complex network because its edges do not have the same type. Aggregating this network will generate a network equivalent to the top network.*

As networks approximate pathways, they include a limited number of *types* of molecules and interactions. Choosing these types mainly depends on the goal (e.g. representing genes, proteins, metabolites), the experimental data (e.g. transcriptomics, phosphoproteomics, metabolomics) and the prior knowledge (e.g. interactions). Networks of all possible physical interactions within the cell are called *interactomes* (Sanchez *et al.*, 1999). They are often made of interactions between proteins, the so-called *protein-protein interactions* (PPI). In comparison, effects of transcription factors on target genes are considered as *protein-DNA interactions* (PDI). They are the edges of *gene regulatory networks* where each node represents a gene as well as its protein. Gene regulatory networks can be inferred from *co-expression networks*. In the latter, edges are correlations between gene expression. I will present co-expression networks in section 4.2 and networks of physical interactions in section 4.3.

Common methods processing and exploring networks do not manage *complex networks* with multiple element types. Rather, networks must be homogeneous. An *aggregation* step can homogenize a complex network into a simple network, a so-called *monolayer* or *monoplex network*. Although monolayer networks are easier to analyze, information is lost during aggregation. They do not enable to combine (i.e. "integrate") multiple -omics datasets or complicated experimental designs such as time series. In the case of time-course datasets, the resulting monolayer networks are unavoidably static. Some methods claim to not require an aggregation step, but they typically ignore these multiple element types and/or produce static networks. The next sections will be dedicated to analyze such monolayer networks. I will focus on how to explore local structures of networks rather than measuring their global properties or determining general laws, in agreement with (Lima-Mendez and van Helden, 2009).

In the past decade, complex modeling approaches have increased in popularity to properly exploit complex networks by representing them as *temporal* or *multilayer networks*. In multilayer networks, each element type is represented as a *layer*, i.e. a homogeneous network. Yet, the layers can "interact". Complex networks will be the topic of the final section 6 of this introduction.

4.2 Inferring networks

Biological network can be built *de novo* by computing co-expression between genes. Such networks are called *co-expression networks*. Fundamentally, building these networks does not require to incorporate any prior knowledge, e.g. physical interactions between proteins.

By paying attention to the vocabulary employed by the papers cited below, there is an inconsistent terminology. Many do not provide any definition. I present how to build co-expression network (see section 4.2.1) and how to convert them into gene regulatory networks using reverse engineering (section 4.2.2). The first step is often called *network construction* and the second step *network inference* associated with *reverse engineering*. Sometimes, *network reconstruction* is also associated to the second step. Yet, this term is employed by authors to define a network built from physical interactions (section 4.3) or an extracted subnetwork (section 5.1). To keep away the confusion of the terms *construction* and *reconstruction* with the other approaches, I will not employ them.

Herein, I am simply using *building co-expression networks* for the first step and *reverse engineering* for the second step.

4.2.1 Building co-expression networks from gene expressions

Approaches based on network inference compute *associations* between gene-expression profiles (**Figure 6**). An association between two genes means that if the expression of one gene varies, then the expression of the other gene also varies. The association is positive or negative if both variations have the same or opposite direction (increasing or decreasing), respectively. Typical statistical methods measuring association strengths (also called *similarities*) compute *correlations*. Nodes of *co-expression networks* are genes, more exactly gene-expression profiles, and edges are association strengths. There are two main categories of approaches: *directed approaches* and *undirected approaches* (Aoki, Ogata and Shibata, 2007). Directed approaches compute correlations between all genes *versus* user-selected genes (Lisso *et al.*, 2005). For example, one can select genes related to a given pathway. These approaches enable to find novel members of pathways. In comparison, undirected approaches measure associations between all genes *versus* all genes, i.e. in a pairwise manner. They are not limited to a set of genes, but they require more computation power. They are also more often used. For reviews, see (Contreras-López *et al.*, 2018; van Dam *et al.*, 2018; Rao and Dixon, 2019; Chowdhury, Bhattacharyya and Kalita, 2020a).

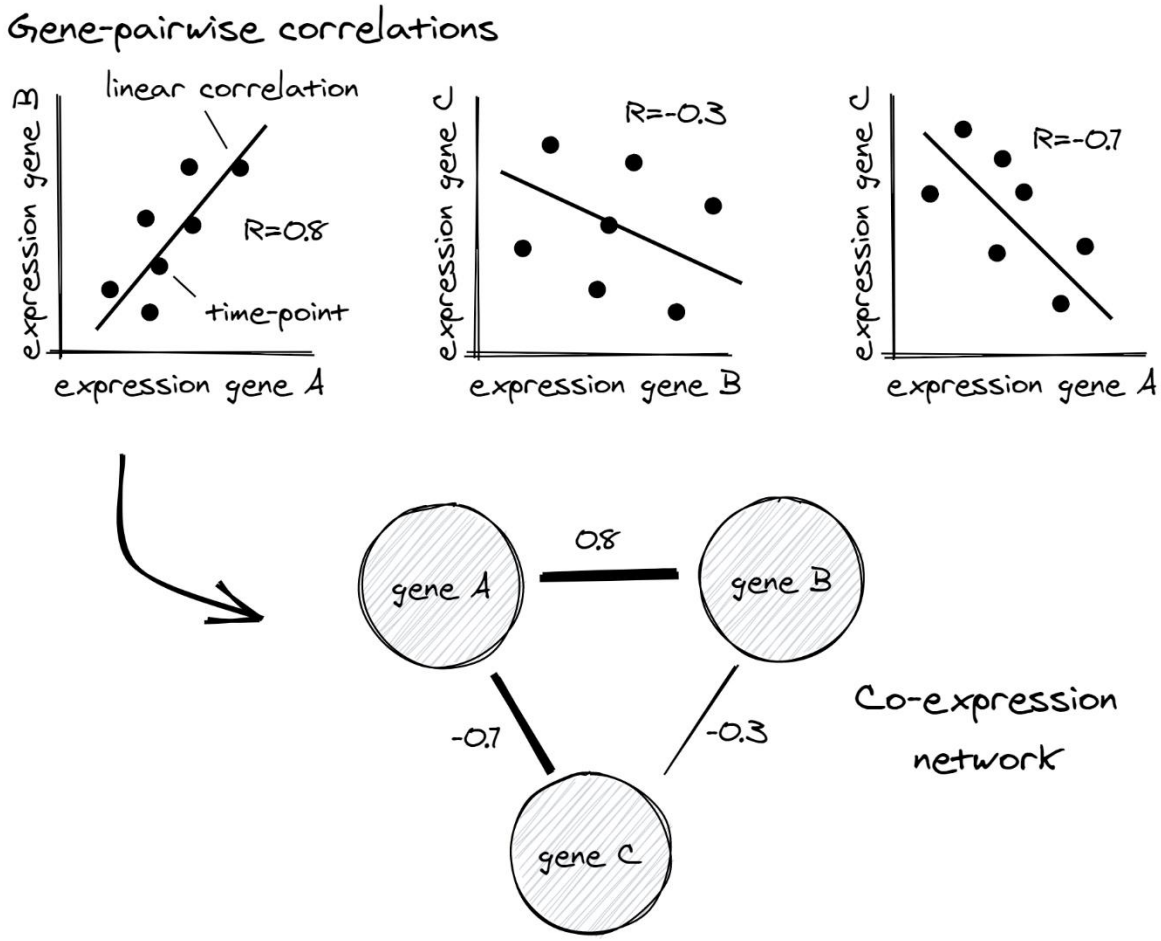

Figure 6. Computing gene-pairwise correlations enables to build co-expression networks. *First, correlations of all genes versus all genes are computed. Second, the co-*

expression network is built by representing a correlation between two nodes by an edge. The edge's weight depends on the correlation coefficient. Thereby, all nodes are linked to all nodes. The thickness of the edge depends on the coefficient. 0.8 and -0.7 are two strong coefficients, even if they imply a positive and a negative correlation, respectively. In this example, I represented linear correlations (e.g. Pearson correlation), but any other type of methods measuring associations between gene expressions are possible.

There is a great diversity of statistical methods computing association strengths. Among them, the *Spearman's rank correlation* is a simple, yet popular and efficient method (Kumari *et al.*, 2012; Ballouz, Verleyen and Gillis, 2015). (Kumari *et al.*, 2012) tested 8 methods. The authors showed non-parametric rank-based methods, such as the Spearman's correlation, were the most robust and efficient. It does not require a linear relationship or a bivariate normal distribution (i.e. both variables follow a normal law), as the widely-applied *Pearson's correlation*, for example. To compute accurate pairwise correlations, (Ballouz, Verleyen and Gillis, 2015) recommended at least 20 time-points.

To generate a co-expression network, the association strengths are converted into edge weights. It depends on whether the network is unweighted or not. For unweighted networks, a *hard-thresholding* step filters out the edges with a strength below a given threshold. The other edges get a weight of 1. In comparisons, *soft-thresholding* computes a continuous weight between 0 and 1 (Zhang and Horvath, 2005). A parameter controls the weight by pushing the lowest strengths toward 0. (Zhang and Horvath, 2005) showed weighted co-expression networks are more robust. A continuous edge weight is not an issue for the downstream steps, but visualizing the network would certainly require to cut off low-weighted edges.

4.2.2 Reverse engineering of gene regulatory networks

Gene co-expression networks enable to explore *correlations* between genes, but not *causations*. *Reverse engineering* aims to transform correlations into causations by directing edges according to causality and by removing those related to indirect associations. The result is an *inferred gene regulatory network*; the edges are *regulatory interactions*. For each edge, the type and the strength of the regulation should be known. See (Liu, 2015) for a comprehensive review of reverse engineering approaches.

For example, (Margolin *et al.*, 2006) developed **ARACNE**. The method computes the association strengths, applies a hard-filtering step and filters out unnecessary edges. The filtering applies a mathematical procedure removing the edge with the smallest weight in each group of 3 connected nodes. The authors proved that the reverse-engineering method perfectly recovers the edge directions if there are no loops in the network (i.e. feedbacks).

Dynamic Bayesian networks are another class of approaches dedicated to infer gene regulatory networks from time series (Murphy and Mian, 1999). They solve the feedback-loop issue by extending *Bayesian networks* which were introduced by (Friedman *et al.*, 2000) for gene expression datasets. A Bayesian network models gene regulation with *random variables* (i.e. the nodes) representing gene expression levels. The edges represent *conditional dependencies* between the variables. Thereby, the value

of a node depends on the values of its *parent nodes*. Random variables follow either a *continuous* or a *discrete model*. In the discrete model, a node can be up-regulated, down-regulated or normally expressed when comparing its expression level to a reference. This model enables to detect non-linear effects but information is lost. The *causal Markov assumption* considers that the level of a given gene only depends on a limited set of other genes. It enables to interpret edges as causal relationships, thus directed, and to simplify the computations. This is an *immediate* dependency; the nodes are independent from the parents of their parents. Therefore, Bayesian networks must be *acyclic*. One node cannot depend on itself through other nodes. Thus, Bayesian networks do not incorporate feedback loops.

An algorithm based on Bayesian networks searches for a model (i.e. all nodes and edges) maximizing its *posterior probability* given the dataset. The Bayes rule computes it by reverting the requirements. Thereby, the posterior probability depends on the probability of the dataset given a model and the *prior probability* of this model. (Friedman *et al.*, 2000) took a prior network with random variables following a uniform distribution, before iteratively updating the model until maximization of the posterior probability. The uniform distribution aims to not favor any edge. A powerful feature of Bayesian networks is to include any type of *prior knowledge*, such as known interactions, to decrease the false positives (Hill *et al.*, 2012). Another advantage is that the networks can also account for missing values by adding *hidden variables* to the model. For example, (Ong, Glasner and Page, 2002) added nodes representing operons. (Bacterial operons are groups of genes regulated by the same promoter and thus, transcription factor.) The dataset only contains the expression levels of the genes. The algorithm can then estimate an adequate model for this network structure.

Bayesian networks are extended by dynamic Bayesian networks (Murphy and Mian, 1999). They incorporate feedback loops by representing edges as causal relationships over time (**Figure 7**). Doing so, the nodes at a given time only depend on the node at the previous time. It assumes the regulatory interactions have a delay; their effect is only observable at the next time-point. No edges are allowed between nodes of the same time-point. Note that dynamic Bayesian networks can be considered as following a *temporal multilayer network* (see section 6.4). (Ong, Glasner and Page, 2002; Husmeier, 2003; Hill *et al.*, 2012) are example of methods implementing this approach. In particular, (Hill *et al.*, 2012) used continuous variables enabling non-additive effects, while (Husmeier, 2003) modeled discrete variables. These methods had been developed for microarrays, but they are also applicable for RNA-sequencing. For both data types, Bayesian networks are fundamentally limited by three issues. First, there are not enough time-points compared to the number of variables (e.g. genes). Second, there are hidden variables that are not included (e.g. post-transcriptional modifications). Finally, even with a perfect dataset and modeling, two models do not necessarily give the same results because there is no unique solution for the resulting causal network.

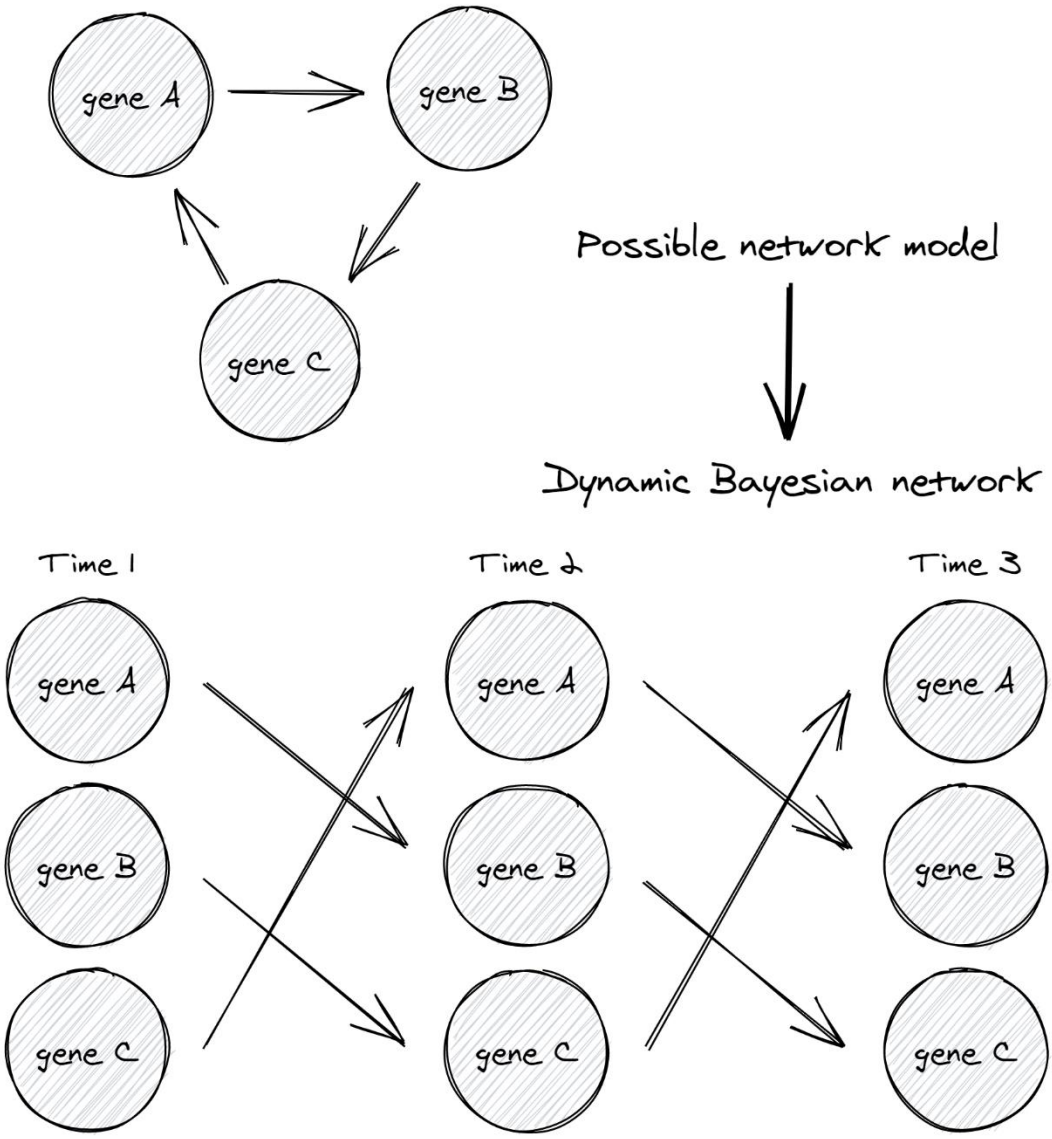

Figure 7. Dynamic Bayesian networks model a directed network over time. *This example shows how to unfold a Bayesian network (i.e. all edges are directed) into a dynamic Bayesian network. All nodes are copied at each layer representing one time-point. Edges model dependencies between nodes, but over time. The list of all dependencies is a transition matrix which is the same for each pair of consecutive layers. It enables to represent graphs with cycles (i.e. feedback loops). Therefore, no edges are allowed within a given layer. A biological or gene co-expression network is directed by identifying the optimal combination of edge directions.*

(Djordjevic *et al.*, 2014) observed that most of the edges of co-expression networks do not correspond to causal relationships. The authors concluded that reverse engineering would not be feasible with co-expression networks alone. Using mammalian developmental gene-expression datasets, a method such as **ARACNE** did not have a better recall and specificity than random networks. Therefore, they are not reliable. The authors also explored perturbation (e.g. mutations, knock-out) gene-expression datasets. These experiments enable to identify all the genes affected by a given perturbation. The authors showed that it was easier to predict more accurate networks with perturbation experiments than with time series.

Co-expression networks are built using the time component of the experiment, but they are still static. They do not enable to identify mechanistic biological processes explaining the observations. Reverse engineering overcomes this limitation by introducing causality to the network. Yet, it requires time-course datasets with a high sampling rate and frequency. The resulting networks often do not include any dynamics, for example when a node is involved in the pathway. They also do not enable to explore interactions within a given time. Representing temporal networks with multilayer networks solves these issues (see section 6). Fundamentally, co-expression networks have *genetic* interactions, i.e. *indirect* interactions. They do not represent true *physical* interactions between molecules. To predict pathways, it is necessary to revert the approach by starting from these physical interactions and then, to map gene-expression data on the network. I will present this strategy in the next section.

4.3 Building networks from the cellular interactome

4.3.1 Experimental detection of molecular interactions

Molecules physically interact with (or bind to) each other, generating the cellular activity. For the purpose of this thesis, two types of molecules are of interest: DNA and proteins. They result in two types of interactions: *protein-protein interactions* (PPI) and *protein-DNA interactions* (PDI). Proteins and genes are assumed to be equivalent so one can represent the PDI similarly to the PPI. In the network, a protein and a gene are therefore identical and interact with each other either in a PPI or a PDI. Roughly speaking, the DNA is a "passive" molecule while a protein is "active". Therefore, PPIs are bidirectional and PDIs are unidirectional. The PDI is unidirectional because we incorporate the regulatory hierarchy between the two molecules. With the same idea, one can use unidirectional PPIs to represent a hierarchical relationship such as post-translational modifications (e.g. phosphorylations).

Many approaches exist to biologically detect protein-protein interactions. They can be *high-throughput* (i.e. detecting many interactions at a time) or *low-throughput* (i.e. few interactions at a time). For example, a well-known high-throughput method is the *yeast two-hybrid* (Y2H) system introduced by (Fields and Song, 1989). It detects many protein interactors of a given protein. The idea is to use a transcription factor divided into two parts. The first part (the "bait") consists of the transcription factor's DNA-binding domain and is linked to one protein to test. The second part ("the prey") consists of the transcription factor's activating domain and is linked to a library of other proteins. If the two proteins linked to the bait and to the prey interact with each other, the transcription factor will be active and expression of a reporter gene will be induced. Using a pre-made library of prey and a screening procedure, all the interactors of a given bait protein can be identified. (Sprinzak, Sattath and Margalit, 2003) predicted between 10,000 to 15,000 PPIs in yeast. Yeast has 100,000 detected PPIs today (López, Nakai and Patil, 2020). Sprinzak and colleagues also estimated that the method produces 50% of false positives. Yet, a well-done Y2H experiment rather generates 20% false positives, e.g. (Li *et al.*, 2004). Thus, the Y2H system is not inferior to other detection approaches (Braun *et al.*, 2009). Braun and colleagues claimed that the issue was mostly a lack of sensitivity as only 25% of interactors could be found in their

benchmark with human samples. Particular protein post-translational modifications (not available in yeast) or very long proteins limit the ability of the Y2H system to detect some interactions. It should however be considered that detecting an interaction in a Y2H context does not mean that this interaction is biologically relevant.

One can computationally predict molecular interactions using transfer of knowledge across orthologs. This approach assumes orthologs conserve their interactions across distant species. To reduce the number of false-positive interactions, (Gupta *et al.*, 2020) predicted the PPIs of a network across species by applying four successive steps. First, the authors determined the orthologs and they retrieved their interactions from other species. Second, they kept the PPIs when both interactors of a PPI shared interacting protein domains. Third, they filtered out the PPIs with interactors not sharing the same subcellular localization. Finally, the isoforms from the same gene were merged into a single node. To computes a confidence score for the interactions, they measured the network modularity (i.e. capacity to form groups of well-connected nodes). They showed these steps increased the average confidence score of the final PPI network.

Strictly speaking, protein-protein interactions are physical and direct interactions. Yet, many interaction databases (see the next section) also host less reliable interactions, i.e. *associations*. These associations can be physical or functional. *Functional associations* define proteins with a functional effect on others, with or without a physical interaction. In comparison, *genetic interactions* are the associated effects of two genetic perturbations such as mutations, gene deletion or overexpression. The **HUPO PSI** consortium controls this vocabulary (Hermjakob *et al.*, 2004). To build physical interactomes for (Pierrelée *et al.*, 2020), I filtered out functional associations and genetic interactions.

Sequencing technologies enable to determine regulatory protein-DNA interactions (PDIs), e.g. (Teixeira *et al.*, 2018). They identify DNA regions on which a given transcription factor binds to. Correlating binding regions to the nearest genes give *binding evidences*. From the collection of all DNA regions, one can also get its binding profile and identify *potential binding evidences* by searching for other DNA regions where the transcription factor could be bound. Chromatin can be indeed folded and thus, not accessible to the factor at the time of the experiment. Furthermore, genetic perturbations (e.g. knock-out, knock-down) of a transcription factor give *transcription evidences*, i.e. dysregulated genes that are assumed to be regulated by the factor. The more an interaction is supported by evidences, the more it is reliable. I selected the PDIs having these three evidences (binding, potential binding and transcription evidences) in (Pierrelée *et al.*, 2020). Yet, these evidences are still indirect. Sequencing chromatin structure provides direct evidences by showing DNA regions interacting to each other. Indeed, a transcription factor binds a region, bends the DNA and links the RNA polymerase at the promoter, inducing gene expression. Therefore, it is expected that the transcription factor's binding region is spatially close to the gene's promoter.

4.3.2 Getting interactions from databases

4.3.2.1 Protein-protein interaction databases

To build networks, one needs interactions between genes. *Primary databases* collect molecular interactions from published experiments. They can have a curation process in which experts manually check and annotate the interactions. *Secondary databases*, also called *consolidated databases* by (López, Nakai and Patil, 2015), collect interactions from primary databases and re-process them (**Figure 8**). They can merge several primary databases to complete them or correct some biases. The databases can especially differ how they collect and curate their input data. Some of them contain predicted interactions.

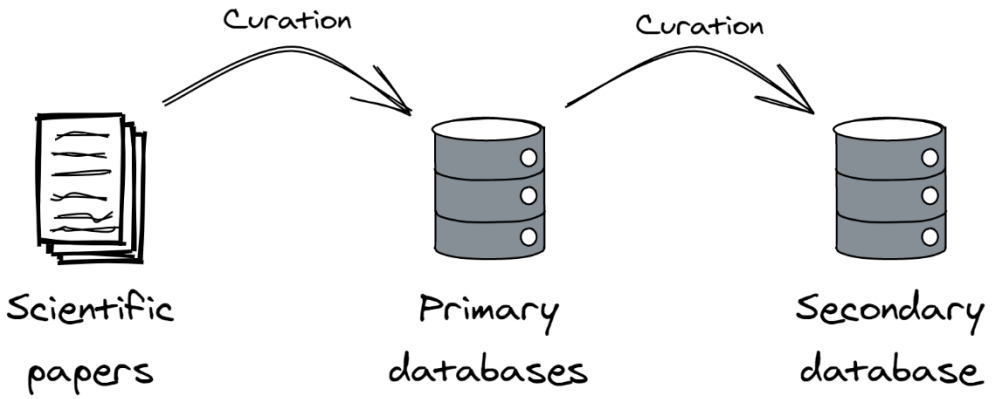

Figure 8. Databases collect and curate scientific papers testing molecular interactions. *Secondary databases do not collect interactions, but curate primary databases to clean false positives. Object models from* (Tjang, 2020).

Among the galaxy of primary database, there is the **IMEx** consortium. The consortium has 13 database members to develop curation and database standards. They work together to curate PPIs derived from publications, splitting their work based on their expertise, capabilities and interests. The members have common guidelines to find and annotate molecular interactions. In particular, they annotate them using the *HUPO PSI-MI* format (Hermjakob *et al.*, 2004). The fields of this format describe the tools used to detect an interaction as well as the references to the papers where they were published. The format includes controlled vocabulary to ensure the consistency of annotations. Within the **IMEx** dataset, more than 80% of the PPIs were not validated as physical and functional PPIs. Most experiments only test whether the proteins bind to each other. Moreover, 85% of interactions come from Yeast, Human, *E. coli* and Mouse. The **IMEx** dataset only contains experimentally-detected PPI but the member databases can also have predicted PPI, functional associations and genetic interactions.

Some **IMEx** members score the interactions based on the reliability of their detection, using a so-called *confidence score.* Different approaches exist to compute it. The **HUPO PSI** consortium recommends to use the **MIscore** approach (Villaveces *et al.*, 2015). For each interaction, it computes the final score by combining the sub scores of the detection method, the interaction type and the number of publications. The final score is the weighted sum of each sub score. It ranges between 0 and 1, with 1 equaling maximal confidence. For example, a *direct association* (i.e. molecules have direct

contact) has a sub score of 1 while a *physical association* (i.e. molecules are spatially close but not necessarily in direct contact) has a sub score of 0.66; a Y2H assay and an imaging technique-based co-detection of two proteins give sub scores of 0.66 and 0.33 respectively.

The primary **BioGRID** database (Oughtred *et al.*, 2021) includes protein, genetic and chemical interactions. It is the biggest interaction database with 800,000 genetic interactions and more than 1 million protein interactions for around 70 species, taken from more than 16,000 publications. In particular, the yeast and mouse species have 117,000 and 72,000 PPIs, respectively. **BioGRID** does not include predicted molecular interactions. Note that **BioGRID** PPIs do not include isoforms as the protein IDs are mapped to gene IDs. **BioGRID** does not belong to the **IMEx** consortium but it takes part in their development of standards for the community.

Primary databases can contain errors in their curation due to false positive interactions or curation issues. They are also incomplete or have redundant interactions. For networks, one wants physical PPIs but these databases often mix several interaction types. A secondary database such as **HitPredict** (López, Nakai and Patil, 2015) aims to solve these biases by combining primary databases and re-processing their data. It combines 4 databases which are **IMEx** members as well as **BioGRID**. In the 2020 update, there are around 800,000 PPIs for 128 species. From those, more than 126,000 come from yeast and 34,000 PPIs from mouse (López, Nakai and Patil, 2020). The curation process is an automated workflow. First, it removes inconsistent, indirect and duplicated PPIs. Then, it converts the protein IDs to have the same format and resolves the unknown IDs by mapping the protein sequences to a sequence database. To compute a confidence score, annotations from primary databases are collected and cleaned. Therefore, it removes around 30% of the initial interactions. **HitPredict** extended the confidence score by computing the geometric mean of the **MIscore** and an "annotation-based score" (**Figure 9**). The latter combines 3 sub scores: whether the protein pairs have common binding domains, whether they share gene ontology terms and whether the interaction has been observed in other species. The authors recommend a minimal confidence of 0.281 to have high quality interactions. It corresponds to have either a **MIscore** or an annotation score higher than 0.5. Therefore, **HitPredict** is much more fitted to build networks, as it reduces the number of false positives within the resulting network.

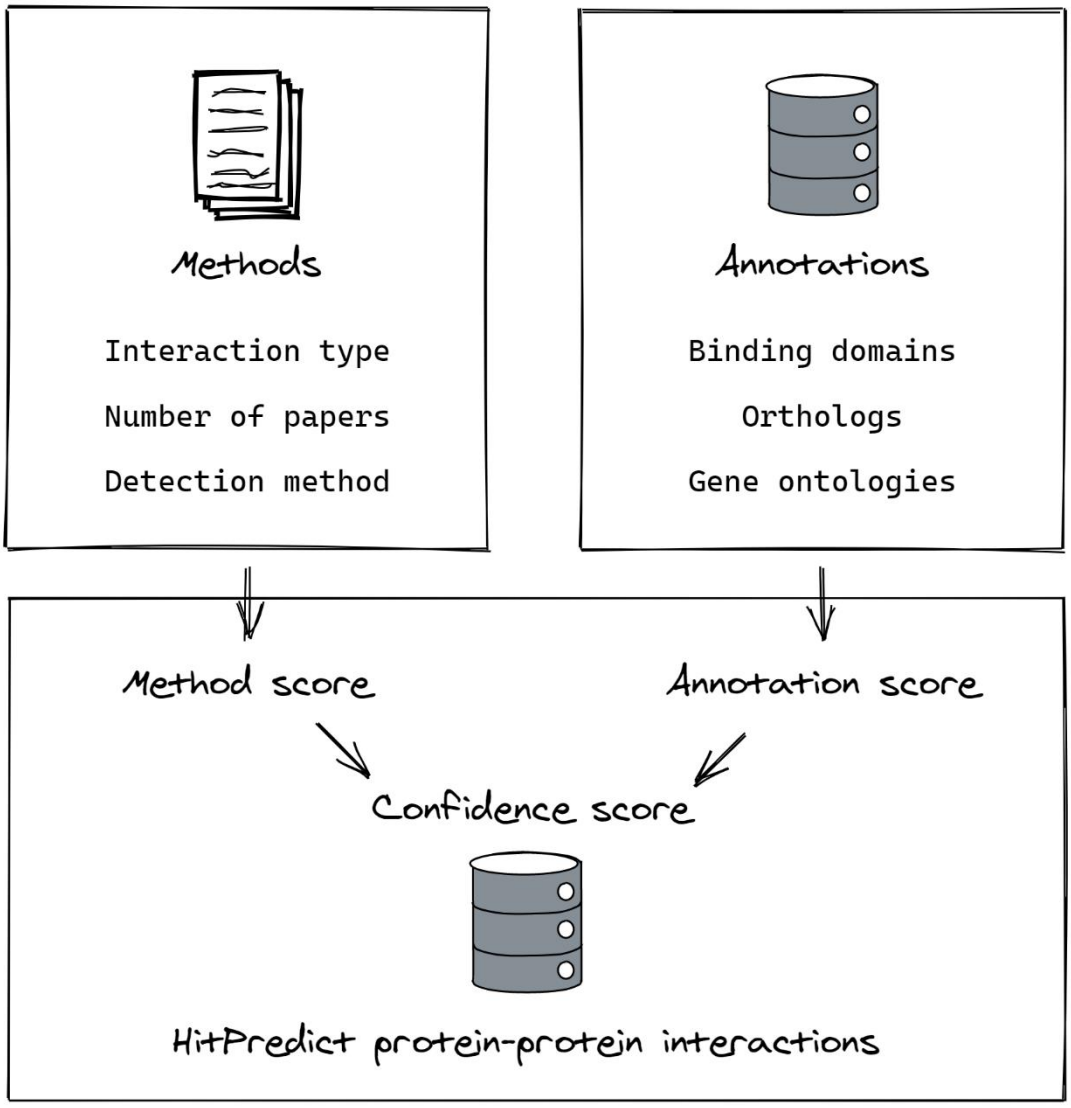

Figure 9. HitPredict computes confidence scores of interactions from the reliability of methods and annotations. *A protein-protein interaction is supported by experimental papers: what and how many methods were applied. Method scores also depend on the type of the interaction, e.g. complex interactions ("physical interactions") or phosphorylations ("direct interactions"). Moreover, HitPredict computes an annotation score whether the structures of interactors are consistent and share gene ontologies as well as whether the interaction is found in other species. Both method and annotation scores are then combined into a confidence score. Object models from* (Tjang, 2020).

4.3.2.2 Other database types

To get protein-DNA interactions, I looked at databases harboring transcription factor occupancy and regulatory data. For example, **JASPAR** (Fornes *et al.*, 2019) and **TRANSFAC** (Wingender, 2008) are well-known curated databases. Both contain the genomic binding sites of transcriptions factors, but only **TRANSFAC** provides the target genes for the transcription factors. However, **TRANSFAC** has a paywall and is limited to eukaryotes. There are no community efforts to curate cross-species PDIs from the literature as for PPIs. To my knowledge, only the primary database **YEASTRACT+** (Monteiro *et al.*, 2020) collects PDIs for yeast. It contains information on 183 transcription factors involved in 175,000 PDIs. The primary database **TRRUST**

(Han *et al.*, 2018) and the secondary database **RegNetwork** (Z. P. Liu *et al.*, 2015) provide PDIs for human and mouse but they have no recent major updates.

Another database relevant for this thesis is **KEGG** (Kanehisa, 2000), a reference database of functional classifications. It assigns a **KEGG Orthology** (KO) to each gene. A KO is a functional ortholog of a gene. If two genes from two species have the same function, share sequence similarity and identify each other in similarity searches, then they will have the same KO. **KEGG** groups the KOs into functional classes, such as pathways in the **KEGG PATHWAY** sub-database, or with hierarchies, such as in the **KEGG BRITE** sub-database. I used it to annotate genes and find relationships between them. For example in (Yun *et al.*, 2020), the **KEGG** tools annotated the predicted genes with orthologs from different species. To find dysregulated pathways, I assigned the best KO from the orthologs to the genes. Therefore, I could consistently use **KEGG** by comparing the KOs of a model microalga species to the KOs from my DEG list. **KEGG** is a highly curated database, but its goal is not to generate the whole PPI or PDI network as the aforementioned databases, but rather stores single pathways. It is therefore less complete than a network combining all known PPIs and PDIs of a species.

4.3.3 Weighting the networks

4.3.3.1 Weighting enables to favor network elements

Weighting the network enables to give a preference to some network elements, i.e. nodes and/or edges, in the downstream steps. There are two criteria to consider. First, the weighting depends on the requirements of the task. If one wants to extract regions of the interactome enriched in dysregulated genes, then it is relevant to put more weights on the nodes that are differentially regulated. For example, among the tools cited in the section 5.1, **viPEr** (Garmhausen *et al.*, 2015), **TimeXNet** (Patil and Nakai, 2014) and **PCSF** (Tuncbag *et al.*, 2013) advised to take the size effect (e.g. log-fold changes) as node weights, while **jActiveModules** (Ideker *et al.*, 2002) took as input the significance (e.g. p-value or adjusted p-value). Regarding the edge weight, methods such as **AnatApp** (Yosef *et al.*, 2011), **TimeXNet**, **PathLinker** (Gil, Law and Murali, 2017) and **PCSF** typically consider the confidence score of interactions. **CySpanningTree** (Shaik, Bezawada and Goveas, 2015) uses correlation scores of gene-expression profiles, but the authors applied this tool on a gene co-expression network. Not all tools support both node and edge weights. These include the aforementioned tools **viPEr**, **jActiveModules**, **PathLinker** and **CySpanningTree**.

Although weighting the network with experimental data such as differential expression is relevant, most tools extracting subnetworks or detecting communities of dysregulated genes also require input nodes. These nodes are given by the user by tagging those nodes which are of particular interest, such as dysregulated genes. If the weight already includes this information, these nodes will be more strongly prioritized than the others. Therefore, as a user, one should ensure that the final results are not too constrained by the dysregulated genes, as they can still be false positives and false negatives.

4.3.3.2 Penalizing hubs

Node or edge weights can be adapted to include other features of a network, such as *hub nodes* (**Figure 10**). Hub nodes have a much higher node degree than average nodes. Biological networks were often assumed to be *scale free* (Jeong *et al.*, 2000). Thus, node degrees would follow a power law which generates few hub nodes and many nodes with a low node degree. Hub nodes can make the subnetwork extraction (see section 5.1) more difficult by offering shortcuts. To avoid this, one could penalize them to extract more complex subnetworks. However, the universality of scale-free networks has been highly debated, e.g. (Lima-Mendez and van Helden, 2009; Broido and Clauset, 2019), thus questioning the disturbance that hub nodes could cause in subnetwork extraction.

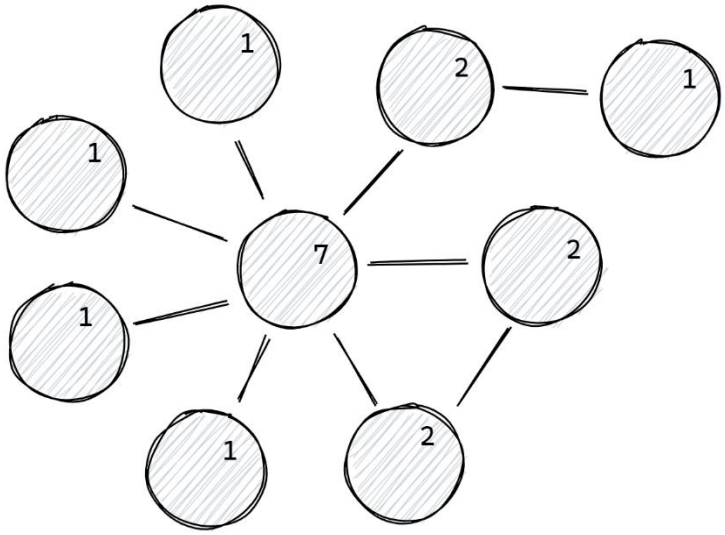

Figure 10. Hub nodes have a very high node degree. *The node degree is the number of adjacent edges to the node. For example, the node "7" has seven edges. Other edges have a much smaller degree. Node degrees of biological networks are biased because not all edges are known.*

Nonetheless, hub nodes should still be penalized because their high degree is partly due to technical and experimental facts. Well-studied proteins have a higher node degree in PPI networks, e.g. (von Mering *et al.*, 2002; Schaefer, Serrano and Andrade-Navarro, 2015). Detecting an interaction depends on the detection method as well as the abundance, function, localization and evolutionary novelty of the proteins (von Mering *et al.*, 2002). In yeast, (Sprinzak, Sattath and Margalit, 2003) found that most of detected interactions of hub nodes were false. Thus, even from a biological point of view, hub nodes are not necessarily relevant. They can be low-level metabolites (e.g. H2O, ATP, NAD) in metabolic networks or degrading proteins that bind to virtually all proteins in protein-protein interaction networks. In general, hub nodes should be considered one by one, because their importance in the network depends on their biological functions. One can merely remove those which should not contribute to any relevant result.

Penalizing the hub nodes increases the accuracy of subnetwork extraction methods (Faust *et al.*, 2010). For example, **PCSF** penalizes the hubs nodes by subtracting the node degree from the node weight (Tuncbag *et al.*, 2016).

4.3.4 Edge directions

Directed edges imply an asymmetry between two interacting nodes with one source node and one target node. On the contrary, *undirected edges* do not distinguish between them. Directed edges enable to include causality, for example a transcription factor binding in the promoter of a gene and thereby influencing gene expression or a protein affecting another between two time-points. They also reduce the number of solutions to search when using the network to e.g. extract subnetworks. A network is directed or undirected when all its edges are directed or undirected, respectively. (Faust *et al.*, 2010) showed that using directed networks enabled to find more optimal solutions than undirected networks.

Some tools can only process undirected networks, e.g. **viPEr**, **jActiveModules** and **CySpanningTree**, while others can process both, e.g. **TimeXNet**, **PathLinker** and **PCSF**. **AnatApp**, **PCSF** and **TimeXNet** can process mixed networks with both undirected and directed edges. Consequently, the input network has to match the requirements of the applied tool. This can be an issue when the network contains several *edge types*. Usually, PPIs are undirected, while PDIs are directed (**Figure 11**). The user has to choose how to *aggregate* them to build a monolayer network. It is easier when the goal is to obtain an undirected network: all edges are converted into undirected edges, where the causal information is lost. To aggregate them into a directed network, one can convert undirected edges into two opposite directed edges. This, however, dramatically increases the number of edges in the network.

Moreover, network aggregation causes two more problems: 1) how to aggregate the weights and 2) how to manage multi-edges. For the weights, I took the arithmetic mean in (Pierrelée *et al.*, 2020) to aggregate PDIs and PPIs. When converting an undirected edge to two opposite directed edges, one can assign the same weight to the two children edges as the mother edge, but this is still an assumption. Multi-edges are several edges connecting two nodes in parallel and in the same direction. Most tools do not manage them well. Therefore, they should be aggregated into at most two opposite directed edges. Yet, for example, **AnatApp** cannot process these opposite edges. This issue of multi-edges can also be raised within networks with a single edge type, if the database is inconsistent. Thus, the user should check this before using any tool on his network.

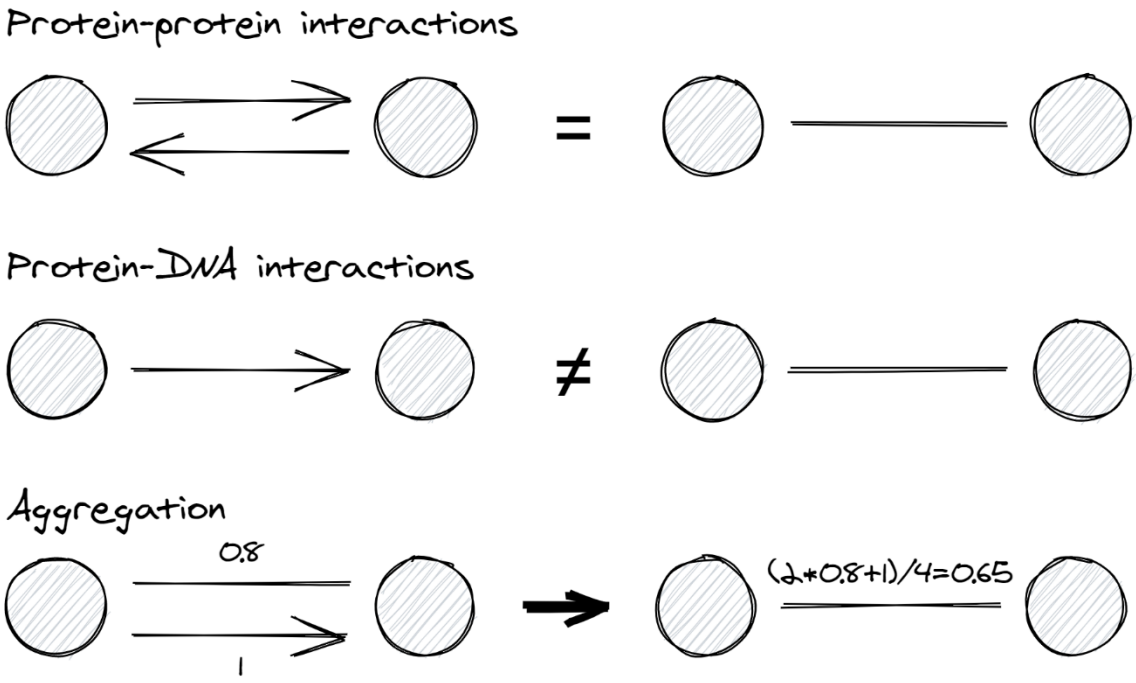

Figure 11. PPIs and PDIs are not equivalent. *A PPI can be modeled by one undirected edge or two opposite edges, while a PDI cannot. If two nodes are linked by one PPI and one PDI, then one can consider that there is one edge going in one direction and two edges going in the opposite direction. In* (Pierrelée *et al.*, 2020)*, I aggregated these edges by making the sum of their weight such as the confidence score of the PPI was the same for both opposite edges converted from the undirected edge.*

To conclude, network-analysis tools manage various types of networks, but they have precise requirements. This makes it difficult to compare two tools which do not share the same requirements.

4.3.5 Condition-specific networks

Condition-specific networks only keep the nodes and edges which are relevant to a given condition. The condition can be a treatment, a tissue or a time-point. I will call some networks being time-specific to highlight the temporal factor.

To generate a condition-specific network, a simple approach is to filter out the nodes with a low level of gene expression together with the edges which connect them to other nodes (**Figure 12**). **DyNetViewer** (Li *et al.*, 2018) provides the algorithm of (Tang *et al.*, 2011) which applies this approach. In comparison, (Park *et al.*, 2017) computed a "perturbation score" for each gene at each time-point by computing the difference between the expression score and the mean of expression across time. Park and colleagues built a time-specific network by filtering out the genes with a low perturbation score, except those which had at least one perturbed gene in their neighborhood.

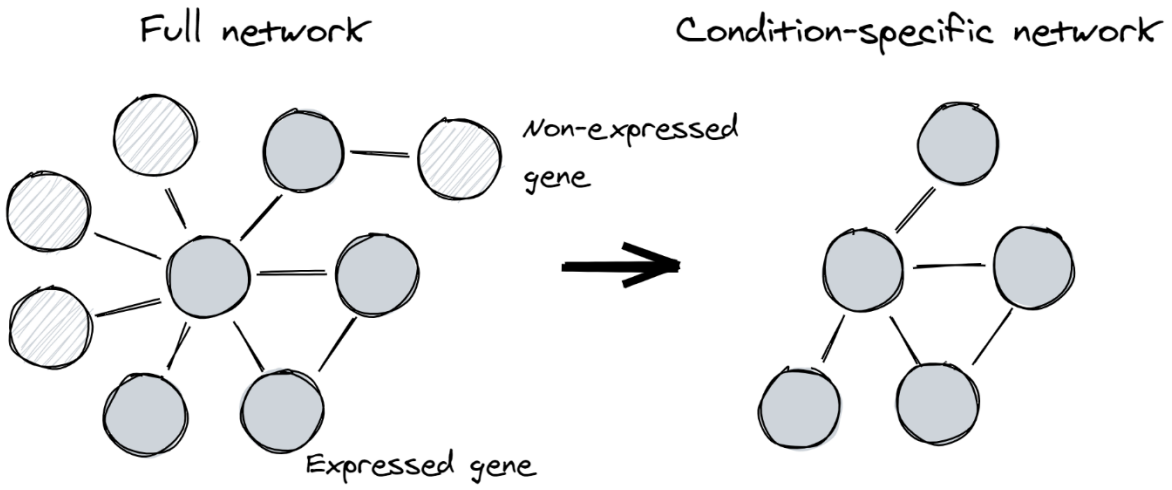

Figure 12. Non-expressed genes are removed to build condition-specific networks. *This is the most common approach in RNA-sequencing as the technology can only detect gene expression.*

In this section, I presented how to build physical networks representing a system of interest. The following sections will focus on how to extract knowledge from them.

5 Finding patterns within biological networks

5.1 Extracting subnetworks

Successive molecular interactions produce biological mechanisms, i.e. pathways. Pathways can for instance trigger the assembly of new molecules, release or capture molecules outside the cell, activate or inhibit gene expression, etc. One way to produce these effects is by dysregulating gene expression, compared to a reference or control condition where such pathways did not happen. These dysregulations can be observed through RNA-sequencing. Reciprocally, finding *paths* between dysregulated genes should identify the successive interactions and so *pathways*. A list of paths gives a *subnetwork*. *Extracting subnetworks* aims to find these paths (Ideker *et al.*, 2002). A path is the list of edges that one can go through to join a *target node* from a *source node*. This is the same analogy as going from one city to another and searching for highways on a map. Methods for subnetwork extraction can incorporate non-dysregulated genes and reject dysregulated genes from subnetworks. They enable to make predictions and reduce the noise from the differential expression analysis. More practically, a user cannot explore large interaction networks by hand. Subnetwork extraction enables a user to visualize and interpret them.

On the contrary, *community detection* looks for parts of the networks enriched in dysregulated genes (see section 5.2). These parts are not necessarily connected together and thus, community detection is less mechanistic-oriented than subnetwork extraction. Yet, the approach should allow to extract several independent subnetworks because the network does not incorporate all possible nodes and edges (Ideker *et al.*, 2002). Consequently, it cannot identify all the paths of the pathway. Several pathways can also occur in parallel and their exploration would be simpler if they were

disconnected. Furthermore, subnetwork extraction has the advantage over gene-expression clustering (see section 3.1) that the approach accounts for genes with low differential expression regardless of their direction (i.e. positive or negative log-fold change) (Ideker *et al.*, 2002).

(Nguyen *et al.*, 2019) is a recent review on subnetwork-extraction methods, although it includes *per se* community-detection tools. The authors especially grouped them into "*greedy algorithms*", "*evolutionary algorithms*" and "*diffusion-flow algorithms*". Yet, they did not mention other types of algorithms such as shortest paths computation and Steiner trees problems. I present examples of methods implementing these algorithms below. All of them take as inputs a network and list of nodes of interest (e.g. dysregulated genes). The input nodes are often split into groups of *source nodes* and *target nodes*. The goal is often to create a causal link between genes considered as effectors and others that are treated as receptors.

5.1.1 Greedy algorithm: viPEr

In the context of subnetwork extraction, greedy algorithms are methods exploring the neighbors of seed nodes to find paths between them. They blindly make the locally optimal choice at each stage of the computation, yielding locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time. More precisely, they explore all possible paths around a seed node such as the paths have a maximal length (i.e. number of nodes included in the path). For example, given a source node and a target node, the **Cytoscape** app (Shannon *et al.*, 2003) **viPEr** (Garmhausen *et al.*, 2015) starts from the target and searches for all paths with a maximal length in its neighborhood. It keeps the paths including the target node to generate the subnetwork. It scores them by aggregating the node weights of the path and the path length. **viPEr** does that for each pair of source and target nodes. Greedy algorithms tend to find suboptimal solutions and produce very large networks as they explore all possible paths independently of each other.

5.1.2 Evolutionary algorithm: jActiveModules

(Ideker *et al.*, 2002) developed **jActiveModules** which is still widely used on **Cytoscape**. (Nguyen *et al.*, 2019) grouped it within the category of evolutionary algorithms. The method applies *simulated annealing*. It computes a subnetwork by randomly adding or removing nodes from the network. At each iteration, it scores the subnetwork by summing non-customizable node weights. The node weight is computed using the p-values from the differential expression analysis. The subnetwork score is corrected by comparing it to the score for randomly-selected genes. If the score at an iteration is lower than the score at the previous iteration, then the added node is not kept. It can be included back with a certain probability that decreases at each iteration. The method can also account for multiple conditions (e.g. time-points). In this case, it computes the subnetwork score for each condition, it takes the "highest" score and finally, it corrects this score. In the paper, it is not clearly stated how the authors treated edges. **jActiveModules** can be iteratively applied to reduce the size of subnetworks. The main issue of the algorithm is the simulated annealing procedure which costs runtime.

5.1.3 Diffusion-flow algorithm: TimeXNet

TimeXNet is initially dedicated to extract subnetworks from time-course datasets (Patil *et al.*, 2013; Patil and Nakai, 2014). The method searches for paths from source to target nodes through intermediate nodes. To define these input nodes, the authors assigned the dysregulated genes to each group according to the time of their highest fold change. Only the input nodes have a weight (e.g. fold change), while all edges have a weight (e.g. confidence score). In fact, the method adds an *artificial source* and an *artificial sink* connecting all the sources and targets, respectively. To find the paths between both artificial sources and sinks, the method applies a *minimum-cost flow optimization* algorithm. It computes a *flow* for each node and edge such, that those with the highest flow are extracted. The flow goes from the artificial source to the sink through the intermediate nodes. On its way, it goes through other nodes. The algorithm computes their flow as the sum of flows from incoming edges. Yet, this flow has a *cost* when passing by the edges. Therefore, the optimization aims to reduce the total edge cost while saving as much as possible the total flow. The edge cost is computed according to the edge weight such that a higher weight gives a lower cost. However, for edges connected to the input nodes, the algorithm replaces the edge weight by a specific weight depending itself on the weight of the input node(s). It favors the edges connected to the input nodes. Finally, as **viPEr**, **PathLinker** or **AnatApp** (Yosef *et al.*, 2011; Almozlino *et al.*, 2017) (see below), **TimeXNet** takes a precise list of input nodes but divides them in 3 groups instead of 2. This division is still not general enough for time series. Thus, **TimeXNet** does not have features which could confer a theoretical advantage over the other tools.

5.1.4 Shortest path algorithm: PathLinker

(Ritz *et al.*, 2016; Gil, Law and Murali, 2017) developed **PathLinker** for **Cytoscape**. It applies an algorithm to find the first shortest paths between the source and the target nodes (**Figure 13**). A shortest path is a path with the smallest length. It minimizes the number of nodes that one crosses to go from one node to the other. The path is *loopless*, i.e. a node appears only once in the path. As for **TimeXNet**, **PathLinker** adds an artificial source connected to all source nodes and an artificial sink connected to all target nodes. It finds all the shortest paths between both artificial nodes. Removing the artificial nodes results in shortest paths between the input nodes. Then, **PathLinker** scores each path by computing the product of the edge weights (e.g. confidence score) within the path. Finally, it returns the paths with the highest score. The user has to define the number of returned paths. The subnetwork is the union of these paths. Two advantages of **PathLinker** are its speed and its API enabling to run it from another app or outside **Cytoscape**. It also enables to define the same nodes for the sources and targets.

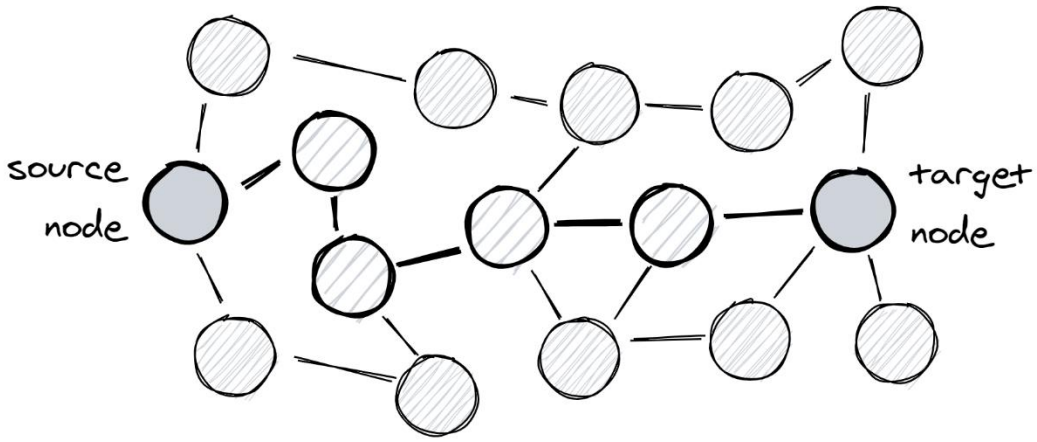

Figure 13. A shortest path is a path with the lowest number of crossed edges. *The shortest path from the source to the target node includes the nodes and the edges with the bold lines.*

5.1.5 Steiner tree algorithms: CySpanningTree, PCSF and AnatApp

To obtain a subnetwork, one can merely remove all the non-dysregulated genes. Yet, nodes of biological networks tend to have a high degree. The yeast PPI network has 14 edges per node on average. Subnetwork extraction decreases this degree by selecting edges which are likely to be involved in the dysregulated pathway. Using again the example of traveling from one city to another, the highway map connecting the cities does not need to include all possible highways, but only those of interest. Solving the *spanning tree problem* produces the minimal subset of edges connecting all nodes. In other words, it is the smallest list of interactions required to generate the pathway. This problem accounts for edge weights. The *minimal* spanning tree minimizes the sum weights while its *maximal* version maximizes the sum weights. **CySpanningTree** (Shaik, Bezawada and Goveas, 2015) implements algorithms to solve spanning tree problems within **Cytoscape**.

Spanning trees create subnetworks in term of edges, not nodes. They include all the input nodes, while some of them can be false positives. On the contrary, they do not enable other nodes of the network to be included to the subnetwork, i.e. false negatives. *Steiner trees* answer these issues by generalizing spanning trees (**Figure 14**). Among a full network, solving the Steiner tree problem aims to connect the *terminals* (i.e. input nodes). Each edge has a *cost*, so the algorithm searches to remove them as often as possible. Yet, removing an edge can disconnect a terminal. Excluding a terminal gives a *penalty*. Losing a terminal is necessary when its connection to the subnetwork costs too much compared to the penalty that it gives. In comparison, adding a *Steiner node* (i.e. non-input node) requires to account for the cost of edges, but that can serve to connect terminals to each other.

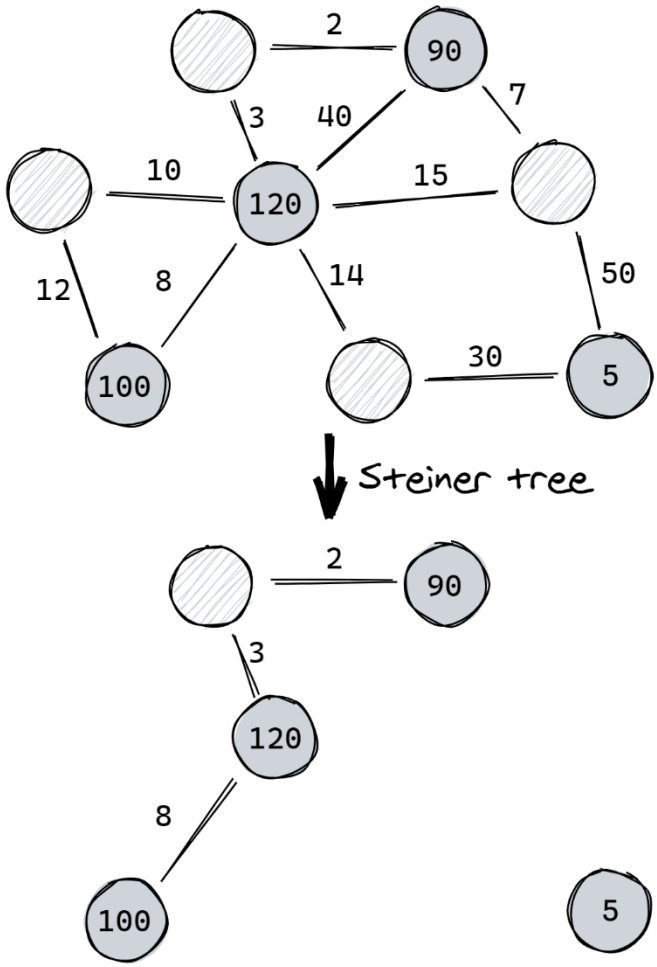

Figure 14. Steiner trees searches for an optimal subnetwork. *The algorithm tries to extract terminal nodes (grey nodes). A terminal node which is not included in the subnetwork (e.g. the node with "5") gives a penalty. On the other hand, included edges require a cost. The optimal subnetwork minimizes penalties and costs. The final subnetwork can include non-terminal nodes called Steiner nodes (white nodes) which do not have a penalty.*

This type of Steiner tree is the *prize-collecting Steiner tree* introduced by (Huang and Fraenkel, 2009) for biological interactomes. Their algorithm aims to minimize the edge costs and the node penalties. A parameter enables to tune the effect of the node penalties. Fraenkel and colleagues added multiple improvements to the algorithm efficiency in the last decade. In particular, the authors extended the algorithm to find independent subnetworks in the same network (Tuncbag *et al.*, 2013). They called it the *prize-collecting Steiner forest* problem (PCSF). It has a second parameter to enable more or less independent subnetworks. The algorithm has been implemented into the **R** package **PCSF** (Akhmedov *et al.*, 2017). Among the subnetwork-extraction tools, it seems the most flexible. It enables to find independent subnetworks, to not split the input nodes into multiple subgroups and to account node and edge weights as well as directed and undirected edges. In fact, the terminal nodes are not necessarily the input nodes defined as dysregulated genes. Terminal nodes are any nodes with a penalty higher than 0. This avoids to fix an arbitrary threshold for the p-value and/or the fold change after the differential expression analysis.

AnatApp is a **Cytoscape** app implementing both, shortest paths and Steiner trees (Yosef *et al.*, 2011; Almozlino *et al.*, 2017). Steiner tree algorithms optimizes the subnetwork by measuring the effect of adding or removing a node or an edge. They work at the *global* level. On the contrary, shortest paths returns a list of paths but without optimizing the whole subnetwork. They work at the *local* level. In (Yosef *et al.*, 2009), the authors developed an algorithm designed to combine both aspects. The trade-off between the global and the local level is set by a parameter. The authors observed that the global level had a better efficiency then the local when most of the relevant genes were defined as input nodes. This algorithm requires the input nodes to be split into a set of source and target nodes. The edges can be directed or not and the algorithm considers both node and edge weights. Moreover, **AnatApp** also includes an algorithm applying only Steiner trees as well as another one applying only shortest paths. The Steiner tree-specific algorithm does not require the input nodes to be split, but it does not take into account the node weights and the edge direction. While **AnatApp** is the only **Cytoscape** app applying Steiner trees, it is not very user-friendly because the network and the input nodes have to be written into a file with a complicated format. It also runs the algorithms on a dedicated server which requires to send the data, which is unfeasible for sensible data or slow internet connections.

5.1.6 Parameter selection

As subnetwork-extraction methods are made for data exploration, selecting their parameters is typically arbitrary. A first criterion is to reduce the subnetwork size such that it enables its exploration. Yet, changing parameters can be unpredictable. To make this choice, (Magnano and Gitter, 2019) proposed the **R** package called **Pathway Parameter Advising**. It decomposes networks into *graphlets* and compares their distribution to those of reference networks from curated databases. Graphlets are subnetworks of limited size (e.g. 4 for this approach) representing all possible connections between the nodes. They represent network patterns. For example, the authors' definition includes the graphlet with the nodes A, B, C and D and the edges A to B and B to C, such that D is isolated. This approach assumes subnetworks are reliable if they have a distribution of graphlets close to biological references. Therefore, one can test several parameters and select those with the closest distribution. In general, the authors recommend not to stick to the default parameters of software packages.

Comparing different subnetwork-extraction methods or algorithms to each other is difficult. No independent benchmarks compared the efficiency of the methods or explored their domain of applicability. It is not possible to *a priori* favor any method, except by considering practical aspects. The papers of the methods also ignore to assess their method with random data and networks. This assessment is critical for network methods as they can easily find correlations within the network structure regardless of the experimental data. Furthermore, no method is available to process time-course datasets even by considering time as a categorical variable. **TimeXNet** claims to be dedicated to time series but its approach does not differ from the others. To process time-course datasets, one can apply these static subnetwork-extraction methods at each time-point of a dynamic network (see section 6.2). For example, I connected the

Cytoscape app **TimeNexus** to **PathLinker** and **AnatApp** to use them to extract subnetworks from temporal multilayer networks (Pierrelée *et al.*, 2020).

5.2 Detecting communities

5.2.1 Vocabulary

(Hartwell *et al.*, 1999) introduced the concept of *modules*. They are groups of molecules generating a function by interacting together. Herein, there are both *topological* and *functional* modules. In a network, *community detection* methods identify communities of nodes that are considered as modules if they meet certain validation criteria (**Figure 15**). As for subnetwork-extraction methods, they assume a reciprocity between the function and the structure, which is not necessarily true (Hric, Darst and Fortunato, 2014). *Community* is a larger concept than *module*. Studies from the physics literature call communities *clusters*. In the fields of mathematics and engineering, they are called *network partitions* when the communities are non-overlapping. In RNA-sequencing, the methods search for communities of dysregulated genes. (Ravasz, 2002) showed that community structures are hierarchical, i.e. they are nested within each other, and (Lewis *et al.*, 2010) confirmed that they are functional modules at multiple scales.

Although one could apply the same method discussed above for both community and subnetwork extraction, community detection is often confused with subnetwork extraction because of two semantic issues. First, communities can be *strictly speaking* considered as a particular case of *subnetworks*. Therefore, *subunits* would be less confusing. Second, some authors only consider the functional aspect of modules when saying *module*. In this case, subnetwork extraction indeed aims to find functional modules. Yet, the goals and the assumptions of community detection and subnetwork extraction are different. The former assumes nodes of communities interact more inside the community than outside (Fortunato and Hric, 2016), while the latter does not. An subnetwork of section 5.1 can merely be a path of nodes without direct connections between the first and last nodes. This vocabulary is not conserved and often mixed up. Below, I mention a few methods to illustrate the diversity of active community detection methods. Many reviews made efforts to precise the definitions and classify the methods, e.g. (Fortunato and Hric, 2016; Batra *et al.*, 2017; Nguyen *et al.*, 2019). (Batra *et al.*, 2017) and benchmarked about 20 tools, but the study is not yet comprehensive.

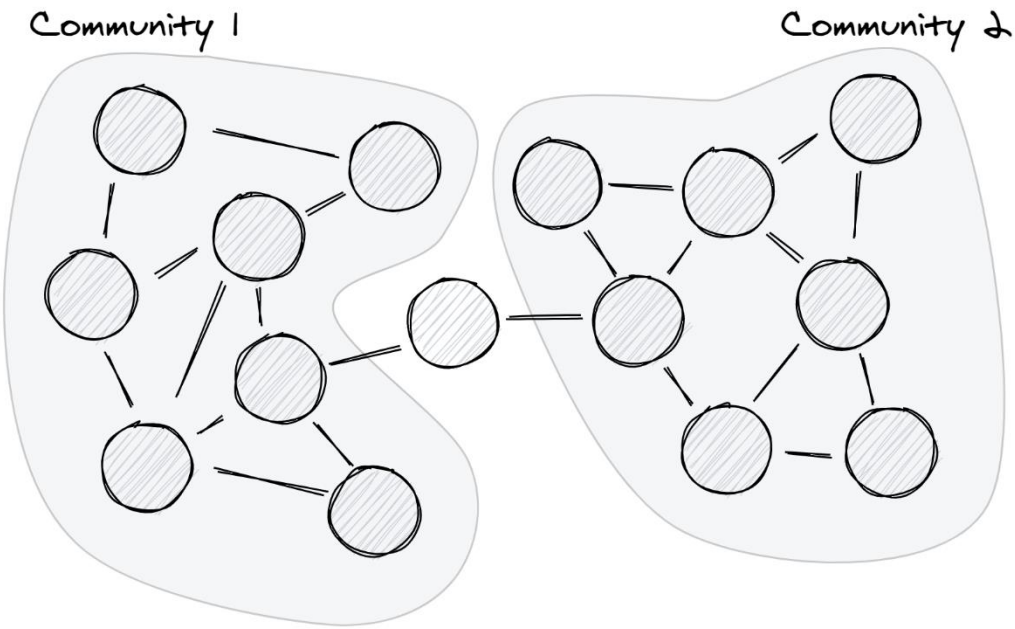

Figure 15. Community detection searches for groups of connected nodes. *This examples shows two communities of highly connected nodes. One central node links both. This node would not be returned by a community-detection tool, while it should be useful for a subnetwork-extraction tool if there are input nodes to extract in both sides.*

5.2.2 Identifying communities

(Nguyen *et al.*, 2019) reviewed 3 approaches to extract subnetworks which are actually detecting communities: the *random-walk*, *maximal-clique* and *clustering-based algorithms*. *Random-walk algorithms* are based on a "walker" jumping from a node to its neighbor. Choosing a neighbor where to jump to depends on a *transition probability*. The communities are the network regions accumulating more jumps than others. For example, **EnrichNet** starts the walkers from the input nodes and uses the edge's confidence score as transition probabilities (Glaab *et al.*, 2012). The method belongs to the class of *random walks with restart* because the walker has a given probability to go back to its node of origin. The walker's path through high-degree nodes are down-weighted to remove this bias from the community detection.

Maximal-clique algorithms search for the largest regions in which all the nodes are connected to each other, i.e. *cliques* (Luce and Perry, 1949) (**Figure 16**). (Vlaic *et al.*, 2018) developed **ModuleDiscover** to find maximal cliques enriched in dysregulated genes. It starts by finding small cliques and then, it expands them. To avoid too large cliques, several cliques can be explored at the same time such that they are competitors. Interestingly, as the nodes are not shared by the cliques, the latter can be merged into a unique "super-node" to simplify the network structure and its visualization. An advantage of **ModuleDiscover** is to generate modules which are not centered on the dysregulated genes which usually serve as input nodes. This is a drawback of many tools for subnetwork extraction and community detection.

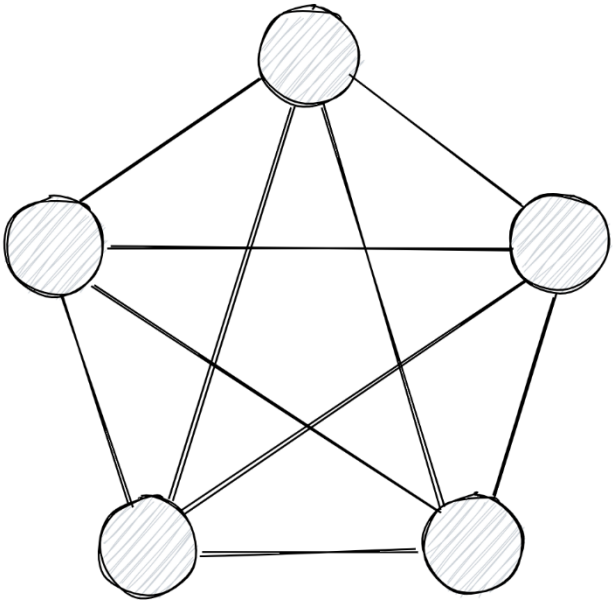

Figure 16. A maximal clique has all nodes linked to all nodes. *This examples shows 5 nodes. If there were less nodes, the clique will not be maximal. Network model from* (Pawlina, 2021).

The category of *maximal-clique algorithms* could also include the methods aiming to directly optimize *clustering coefficients*. They measure how much a community is close to be a clique at a level of one community (Watts and Strogatz, 1998) or all communities (Luce and Perry, 1949). With the same idea, *modularity* measures the clustering level by comparing the number of edges in a given community to a random community (Clauset, Newman and Moore, 2004). The latter study applied a greedy algorithm to optimize the modularity. Furthermore, (Xu *et al.*, 2007) developed **SCAN** to find small, highly-connected regions of a network. These regions are less stringent than cliques. They give communities after expanding them to their neighborhood. One interesting feature of **SCAN** is that it enables to detect hubs and outliers, which are not included in the communities. It defines the hubs as nodes connecting communities and the outliers as nodes which do not belong to a community and are not hubs.

In comparison, *clustering-based algorithms* combine clustering of gene expression profiles as well as interaction networks. To avoid a confusion with the clustering methods discussed in this section, it is more exact to call them *gene profile-clustering algorithms*, similarly to Mfuzz or DREAM but incorporating network information. **ClustEx** finds communities around node pairs with similar expression profiles (Gu *et al.*, 2010). It thus requires time-course datasets. First, it computes the correlations between gene expressions to weight the edges with the correlation coefficients. Then, it computes the distance between a node pair of dysregulated genes by taking the *length* of the shortest path. The algorithm considers the edge weights to compute the length. Third, **ClustEx** applies a hierarchical clustering to group the dysregulated genes. Fourth, it adds the nodes from the shortest paths between the dysregulated genes, as well as their neighbor nodes to the groups. Finally, it computes again the shortest paths between the nodes of the groups and it ranks them. **ClustEx** returns the modules with the shortest paths having the best scores.

5.2.3 Identifying functional modules

To evaluate modules identified by community detection methods, one common criterion is the significance of *functional enrichment*, e.g. (Mete *et al.*, 2008). If enriched terms are assigned to communities, then they are considered as functional modules. Another criterion is the *functional semantic similarity* of communities, e.g. (Zheng, Wang and Glass, 2008). This approach measures how many annotation two nodes share. It considers the probability to find a node annotated by a given annotation within the annotation database. To score a given community, the approach computes the score of each node pair in it. However, similarly to their effect on network structure, (Luecken *et al.*, 2018) showed well-studied proteins are more likely to be included into modules. On the contrary, there are less modules within regions enriched in poorly-studied proteins. This is a *selection bias*. Therefore, the authors developed **CommWalker** to remove this bias. It applies a short *random walk* (see below) on each node of modules identified by community detection tools. It then computes a *functional homogeneity* for each random walk, i.e. the level of similarity between the annotations of nodes. This gives the functional homogeneity of the network background at a local level. Next, **CommWalker** computes a score for the module by comparing its functional homogeneity to its local background. Finally, the communities with a score below a recommended cutoff are called modules.

5.2.4 Evolving communities

When time series are available, one can explore how communities evolve over time. (Spiliopoulou, 2011) reviewed the multiple definitions of *evolving communities* and the applied approaches as well as their assumptions. These approaches typically consider that networks evolve at each time-point by updating their structure, i.e. adding or removing nodes or edges. A time-point without update is ignored. This is the same input as for *temporal networks* (see section 6.1).

A first approach is to consider time as successive discrete time-points and identify communities for each of them. It enables to explore the evolution of each community by comparing them over time. For example, (Palla, Barabási and Vicsek, 2007) applied a maximal-clique algorithm on each time-point. To identify evolving communities, they applied the algorithm on the union of a pair of two consecutive time-points. This joint network contained the common communities as well as the communities unique to a time-point of the pair. Thereby, they could compare how the communities evolved over time: size increasing, decreasing, merging, splitting, birth and death.

The above approach considers the time-points as being independent. However, one can assume that a network at a given time-point depends on its state at the previous time-point. This is considered by *evolutionary clustering* algorithms. Again, several approaches exist. An interesting one is to smooth the time-independent processing of the approach by (Palla, Barabási and Vicsek, 2007). To do so, (Chakrabarti, Kumar and Tomkins, 2006) identified communities by applying algorithms such that similarity scores are optimized. Similarity scores include a local similarity, equivalent to clustering coefficients, as well as a temporal similarity. The latter measures the distance between the current time-point and the previous one.

Both approaches explore evolving communities by aggregating time-points within a time window of either one or two time-points. They model the system at each time window. However, estimating a single model from the whole time series enables to find other types of patterns, such as global or irregular patterns or those with a frequency lower than the size of the time window. With this goal, (Tantipathananandh, Berger-Wolf and Kempe, 2007) suggested a method enabling an intuitive representation of evolving communities by solving a *graph-coloring problem*. A famous application of this problem is to color the countries on a map with 4 colors such that two neighbor countries do not have the same color. The method follows the *individuals* (e.g. genes) over time; an individual at a given time is a node. The individuals form temporal communities. A community at a given time is a so-called *group*. Therefore, a node only belongs to one group. One group represents one specific community. The method focuses on how the communities evolve and so, it assumes that the groups are already known at each time-point.

Solving the graph-coloring problem aims to find the smallest number of temporal communities explaining the time series (Tantipathananandh, Berger-Wolf and Kempe, 2007) (Figure 17). In the first step, it consists of assigning a color (i.e. temporal community) to each group. The second step is to assign a color to the nodes *from* the group colors. Indeed, at a given time, an individual can have either 1) the same color as its group color or 2) a color different from its group color. An individual having inconsistent colors implies that it is an "intruder" to its current community. In other words, it belongs to a temporal community but it transiently contributes to another. Between two time-points, the individual can either 1) stay in the same community, 2) switch its membership to another or 3) become an intruder. The algorithm assigns colors to the nodes by searching for the optimal node coloring. To do so, it minimizes the total penalty of the node coloring. The penalty is increased when an individual changes its community or is an intruder. Therefore, the authors assumed that the individuals tend to keep their membership and stay in their community's groups, or switch between a limited number of communities. Splitting the graph-coloring problem into two steps simplifies the optimization by avoiding to test all combinations of node colors. Yet, testing all combinations of group colors is not possible either. The authors developed approximation algorithms for large datasets.

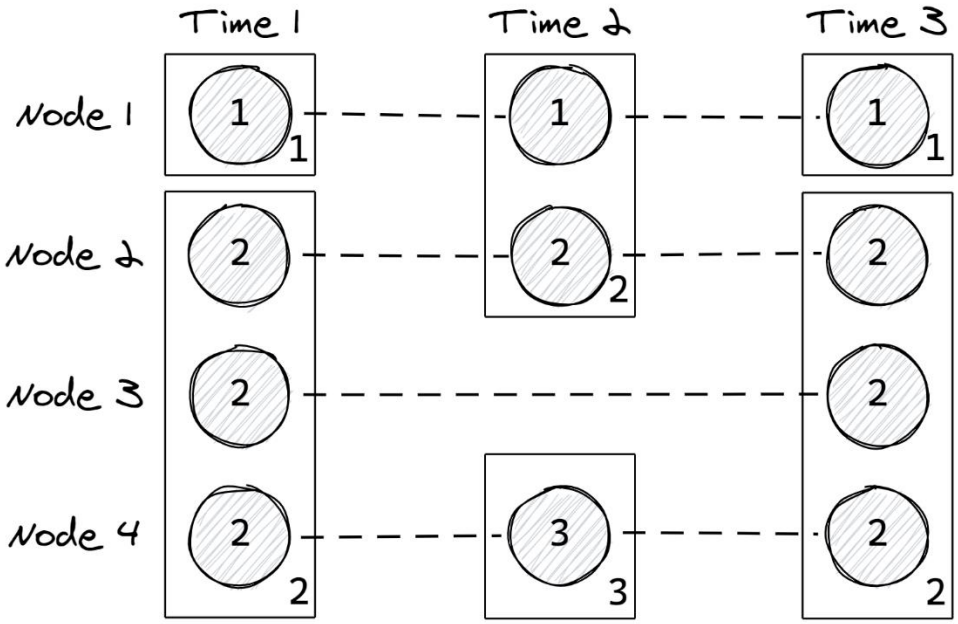

Figure 17. Solving the graph-coloring problem enables to detect temporal communities. *Each square is a group of nodes clustering together at a given time-point. The numbers indicates the temporal community assigned to a group or to a node. Indeed, at a given time-point, a node can belong to one temporal community while its group belongs to another community. For example, at time 1, the node 1 belongs to the community 1 and to a group also assigned to the community 1. Yet, at time 2, node 1 and node 2 are assigned to the same group which is assigned to the community 2. Thus, node 1 is an intruder to the community 2 at time 2. On the contrary, node 4 changed from the community 2 to 3 before to go back in 2. The node 3 does not exist at time 2. Edges within groups are not showed. Dashed lines represent evolution of nodes over time.*

The approach of (Tantipathananandh, Berger-Wolf and Kempe, 2007) is an example of *temporal networks* (see section 6.1). They enable to represent the dynamics within the network structure. The other aforementioned approaches, methods and algorithms still produce static networks and do not incorporate the dynamics into their computation. They are thus not fitted to determine and explore dynamic pathways.

5.3 Exploring dynamic networks

In the below sections, I present 3 examples of approaches to explore dynamic (sub)networks: visualizing their structure, comparing networks to identify their rewired elements or inferring causalities to determine the behavior of predicted pathways.

5.3.1 Visualizing networks

Cytoscape (Shannon *et al.*, 2003) is a popular tool for network visualization and analysis with almost 10,000 citations on PubMed (NCBI, 2021). The most prominent feature is its modularity. It enables the research community to create apps extending the capabilities of Cytoscape. For dynamic networks, **DyNet** is a popular app enabling to visualize condition- or time-specific networks (Goenawan, Bryan and Lynn, 2016). I developed the app **TimeNexus** to process temporal multilayer networks (Pierrelée *et*

al., 2020). An app store is available to easily download the apps, e.g. http://apps.cytoscape.org/apps/timenexus. Other tools exist, such as D3.js (Bostock, 2011) and BioJS (Dao, Wilzbach and Corpas, 2014) to visualize networks on web browsers.

In comparison, **KEGG Mapper** (Kanehisa and Sato, 2020) enables to visualize the maps from the **KEGG PATHWAY** database with nodes colored by the user. The user just provides a list of KEGG orthologies and specifies the species. Then, KEGG Mapper returns all the KEGG maps matching at least one KO of the list. The provided KOs are colored on the maps. This is convenient to quickly annotate an input list. I applied this tool in the paper (Yun *et al.*, 2020) to get the pathways enriched by the differentially expressed genes.

5.3.2 Rewiring

Between conditions or over time, networks do not necessarily conserve their interactions. Some molecules can start to interact while others stop their interactions. Comparing networks across conditions allows to identify these *rewired edges* (Bandyopadhyay *et al.*, 2010). This topic has been extensively studied for gene co-expression networks, see for example the reviews (van Dam *et al.*, 2018; Chowdhury, Bhattacharyya and Kalita, 2020a).

For a series of networks (e.g. time-points), **TVNViewer** merely enables to visualize the rewiring by shadowing the rewired edges (Curtis *et al.*, 2012). **PPICompare** identifies and returns the significant rewired edges by comparing a collection of networks from one condition to another, assuming all networks were built from the same original network (Will and Helms, 2017). From the pairwise comparisons of networks of both conditions, this method computes two statistics. First, it sums the number of removed and lost edges between each pair of neighbor nodes. This gives the "amount of rewiring per edge". Second, the method also computes the fraction of rewired edges in each pairwise comparison. The fraction is used to evaluate the probability of rewiring for a given edge. Further, **PPICompare** predicts rewiring events which could explain the significant rewired edges. This method could be used to compare time-specific networks from two conditions. **DyNet** also provides visualization features to represent the rewired nodes and edges (Goenawan, Bryan and Lynn, 2016). The authors defined the rewired nodes as the nodes with the highest variance for a given numerical attribute.

5.3.3 Causality inferences

Inferring causality within a network aims to direct the edges according to the time's arrow. The methods assume that the effect on a given gene at a given time-point depends on the effects of the neighbor genes at the previous time-point. In other word, a dysregulated gene at a time induces a dysregulation of its interacting partners at the next time. The edge is then directed from the former to the latter. These methods produce temporal paths as the nodes depending on time. They assume the sampling rate is high enough and that a signal can indeed propagate within the network. Also, the effect should happen on the same molecular layer (e.g. PPI network or gene regulatory network), otherwise it would confound the causality.

(Köksal *et al.*, 2018) developed **TPS** to direct the network's edges according to the causality. Their tool takes as input an undirected condition-specific network. **PCSF** generated this network by extracting a subnetwork from a PPI with the node weights being the highest fold change across time-points. **TPS** also uses the fold changes to identify which nodes are activated or inhibited at a given time. These dysregulations correspond to a temporal event. The tool then defines 3 logical constrains to the edges. First, a given edge can only have one type of dysregulation (i.e. activation or inhibition) in one direction. Second, to respect the causality, the edge has to start from the node which first underwent a dysregulation. Third, the user can define prior constrains, for example to force the direction of an edge. These constrains define all possible models for the final time-directed network. **TPS** infers it by applying a mathematical solver to not explicitly define the quasi infinite number of possible models. Note that the tool assumes the final network is loopless (i.e. no feedbacks) and a temporal event can only happen once.

(Anand and Chatterjee, 2017) implemented a simpler method to find a temporal path among a network of gene sets. To build a network at each time-point, the functional enrichment method **GSEA** is used to identify the enriched gene sets. The edges then link the gene sets with common genes. It gives as many networks as time-points. To find a temporal path, the method selects the gene set with the highest enrichment score in the first network. In the second network and the next, it takes the gene set with the highest score in the neighborhood of the last gene set of the path. The temporal path corresponds to a succession of highly enriched gene sets. The authors claim that the path identifies the propagation of dysregulation among the biological processes. However, the method has three limits. First, two consecutive enriched gene sets do not necessarily imply a causality, as a biological process can affect another without sharing any gene with it. Moreover, the method is restricted to find a unique path, while parallel propagations could happen. Finally, the enrichment score does not measure how much a gene set is dysregulated, even if **GSEA** includes gene scores (e.g. fold changes). It would be more relevant to directly consider the significance and/or the fold changes of genes among the gene sets. Nonetheless, the method is not restricted to networks of gene sets.

6 Developing temporal multilayer networks

(Lima-Mendez and van Helden, 2009; Przytycka, Singh and Slonim, 2010) advocated to combine multiple data types and consider time within networks. Multilayer and temporal networks enable these goals.

6.1 Temporal networks

Temporal networks extend static networks by redefining edges as contacts. A contact is a transient link between two nodes happening at a given time. The list of contacts defines the temporal network. (Holme and Saramäki, 2012; Holme, 2015) defined this research field by unifying various approaches exploring time series with networks under this paradigm.

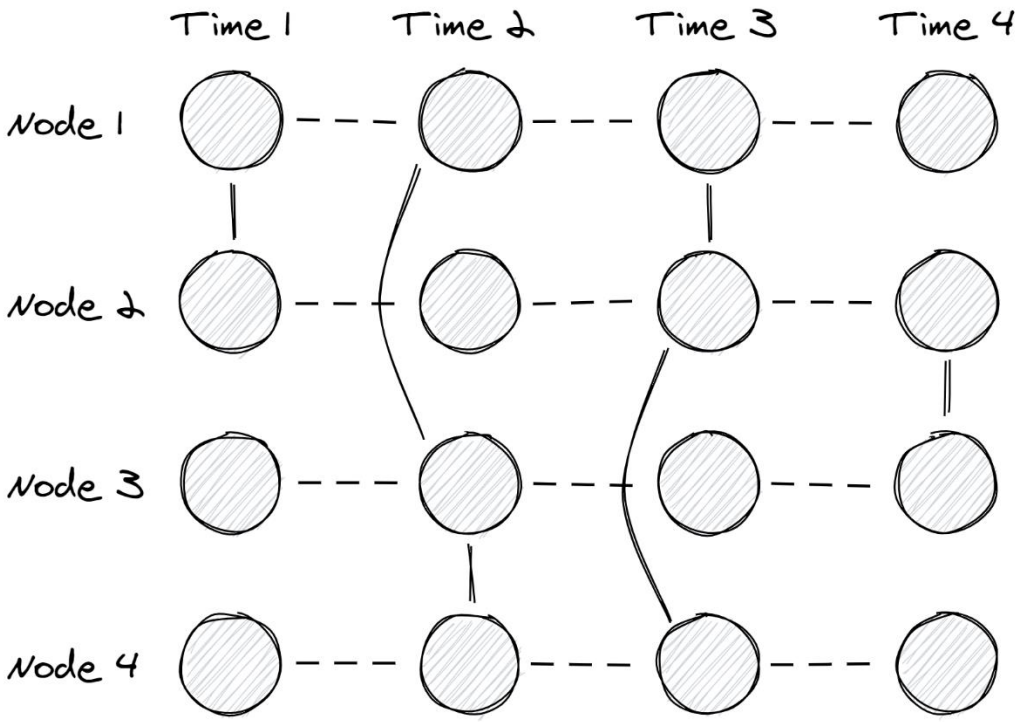

Figure 18. Temporal networks model transient links between objects. *Solid lines are contacts (i.e. transient edges). Dashed lines represent evolution of nodes over time. This represention is called "contact sequence". Dynamic and multilayer networks can also be used to represent temporal networks.*

Modeling dynamic processes with temporal networks requires to first assume that changes of the network structure can affect the dynamic process. The latter should be of the same order or last longer than the time-point frequency (called *time scale*). Temporal networks are thus not applicable to model biological processes from experiments were the sampling frequency is too low. A very fast sampling frequency would not be an issue because time-points without changes could be filtered out or aggregated. On the contrary, *evolving static networks* model static networks approximating dynamic processes when the time-point frequency is not fast enough compared to the dynamic process.

To my knowledge, no method was developed to extract temporal subnetworks following the same goal as in section 5.1. One idea would be to find the equivalent of shortest paths for temporal networks and then, apply the same strategy as **PathLinker** (Ritz *et al.*, 2016; Gil, Law and Murali, 2017), i.e. merge all the shortest paths between source and target nodes. However, it is not straightforward to define a *temporal path* or even the source and target nodes. For a start, (Bui-Xuan, Ferreira and Jarry, 2003) proposed 3 definition of temporal paths (called *journeys*). A journey depends on the nodes of departure and arrival as well the traveling time and the arrival time. The *shortest journey* is the smallest number of nodes that a walker has to go through to join two nodes. Instead, the *fastest journey* optimizes the traveling time. When traveling between cities, the country roads give shortest paths but not the quickest connections. To arrive at the earliest time, the walker searches for the *foremost journey* between two nodes. The walker can choose to go through traffic jams to arrive in time, even if that

was less efficient than leaving at a later time. Note that a journey must follow the time flow; no shortcut is possible by going back in time. Therefore, going from the departure to the arrival node is not equivalent to going from the arrival to the departure node. Furthermore, temporal paths do not guarantee that a sub-path respects the property of its path, e.g. a sub-journey of a shortest journey is not necessarily a shortest journey itself. For example, let's take 3 edges (a, b), (b, c) and (c, d) such that they are activated at the time-points 1, 2 and 3, respectively, as well as an edge (a, c) activated at time 4 (**Figure 19**). The path $\langle (a, b), (b, c), (c, d) \rangle$ is the shortest journey from a to d. Yet, the sub-journey $\langle (a, b), (b, c) \rangle$ is a not the shortest journey from a to c; it would be $\langle (a, c) \rangle$. Consequently, algorithms from static networks cannot be applied on temporal networks. This motivated the work of (Wu *et al.*, 2016) to develop algorithms to minimize these temporal paths.

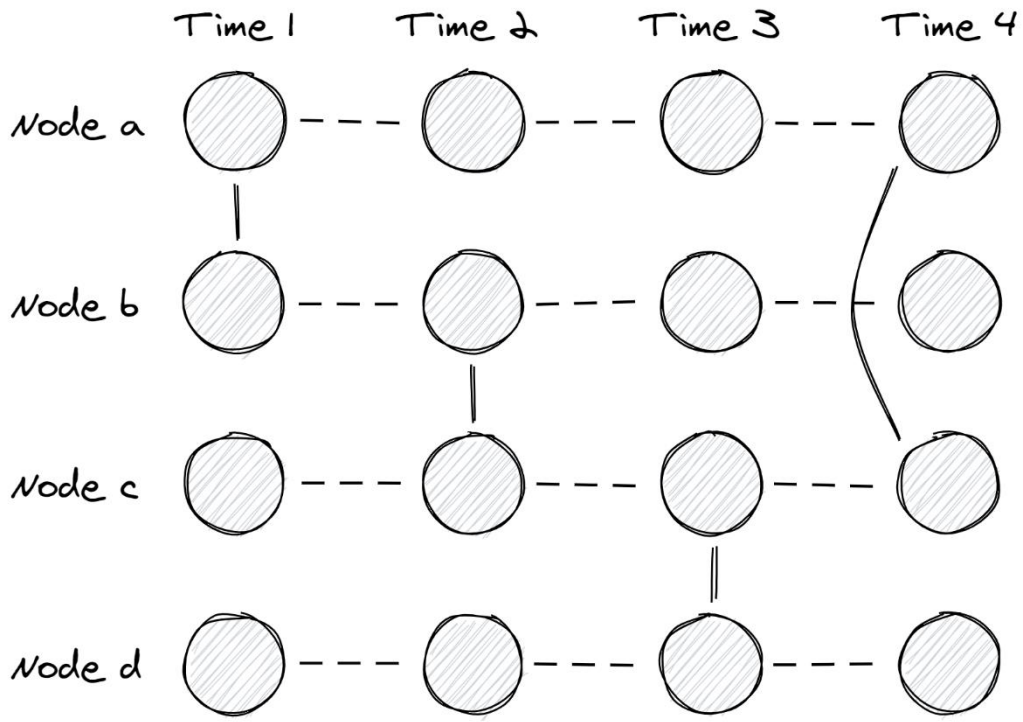

Figure 19. Shortest paths from static network are not applicable to temporal networks. *Following the contact-sequence representation, solid lines are contacts (i.e. transient edges). Dashed lines represent evolution of nodes over time. The journey from node (a) to (d) is the only shortest journey. It goes through nodes (b) and (c). Contrary to static shortest paths, subsets of shortest journeys are shortest journeys. Indeed, the journey from (a) to (c) is a subset of the shortest journey from (a) to (d), but the shortest journey is the contact from (a) to (c) at time 4 without going through (b). Other definitions of minimal temporal paths are possible. For example, the foremost journey optimizes the arrival date. The journey going from node (a) at time 1 to node (c) at time 2 is a foremost journey. A walker will arrive sooner to node (c) with this journey than using the journey between both nodes at time 4.*

Subnetwork extraction methods for temporal networks are still lacking.

6.2 Dynamic networks

Dynamic networks represent temporal networks as a collection of networks, each of which representing the network at a given time-point (so-called *snapshot network*) (Holme, 2015) (**Figure 20**). These are popular in computational biology. If there is no interaction between two proteins at a given time-point, then the snapshot network does not contain the edge between these two nodes. *Time-specific networks* are thus dynamic networks (see section 4.3.5). I already presented how to use these to detect evolving communities (see section 5.2.4). Datasets from RNA-sequencing cannot usually define whether an interaction happened or not. Therefore, methods building time-specific networks do not remove edges, but nodes. Typically, they do not include non-expressed proteins at a given time in the associated time-point network, e.g. (Tang *et al.*, 2011). This limit is intrinsic to the technology.

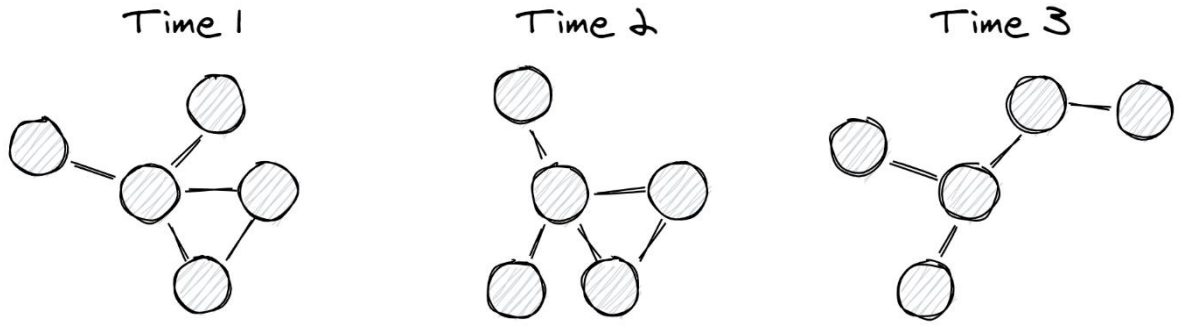

Figure 20. Dynamic networks represent temporal networks as collections of snapshot networks. *A snapshot network includes all contacts which are present at a given time-point.*

Dynamic networks enable to apply methods developed for static networks on each snapshot networks. For instance, (Luo and Kuang, 2014) aimed to search for proteins of interest within a dynamic protein-protein interaction network. The authors built a time-specific network following the same principle as (Tang *et al.*, 2011). Then, they computed static measures for each node of each snapshot network. The measures do not depend on other time-points. To have a dynamic measure for a given protein, the authors' idea was to sum the static measures of a protein's nodes and to divide the sum by the number of networks containing the protein. In other words, they computed the average of the static measure across time.

This approach does not exploit the whole dynamic network. It measures local properties of nodes. Instead, one can explore the global structure. (Masuda and Holme, 2019) developed a method to identify major events (i.e. *change-points*) affecting the network structure. The authors assumed snapshot networks can be grouped such that the transition from one group to another implies a change-point. To do so, they computed the pairwise distances between each snapshot network and applied a hierarchical clustering. For a given cutoff, they obtained clusters. The transitions between clusters are the change-points.

Temporal networks are not adapted to model biological processes with interactomes. Knowing which interactions happened and which did not is too difficult. A more

general definition of dynamic networks, which includes potential interactions, would enable to process RNA-sequencing data as I presented in the sections 5.2 and 5.3. Yet, dynamic networks lack the potential of temporal networks, as illustrated by the temporal paths, because this representation constrains the view to independent snapshots. Exploring the dynamics requires an *ad hoc* step where processing is limited or information is lost. Finding pathways and what happened at each time-point, in relation to the full time series, requires another approach.

6.3 Multilayer networks

Multilayer networks (MLN) are collections of homogeneous networks (**Figure 21**). They enable to combine multiple data types into a single model. Herein, I apply the definitions of (Kivelä *et al.*, 2014). A homogeneous network is called an *elementary layer* (abbreviated as *layer*). It has the same type of nodes and the same type of edges. Several layers can have common properties. A consistent group of layers is called an *aspect*. For example, time is an aspect shared by layers, each of them representing a time-point. One can take the statistical factors and their levels as an analogy to describe the aspects and their layers, respectively. The advantage of multilayer networks is to combine an unlimited number of aspects. Following this example, another aspect can be added to time, with one layer representing RNA-sequencing data and another layer representing phosphoproteomics data at each time-point. Thereby, a layer is the results of one of these two experiments at a given time-point.

As layers can share the same "node", a *node* represents the set of nodes specific to layers. A node within a given layer is called *node-layer tuple* as it is the coordinate of the node depending on the aspects. In (Pierrelée *et al.*, 2020), we abbreviated it as "*layer-node*". Layer-nodes are thus concrete entities in the multilayer network, while nodes are more abstract. In other words, they are the *state* of a node. The layer-nodes are connected through *intra-layer edges* within the same layer and through *inter-layer edges* between two layers. The latter enable to define how layers interact. If an inter-layer edge connects two layer-nodes related to the same node, then this inter-layer edge is so-called a *coupling edge*.

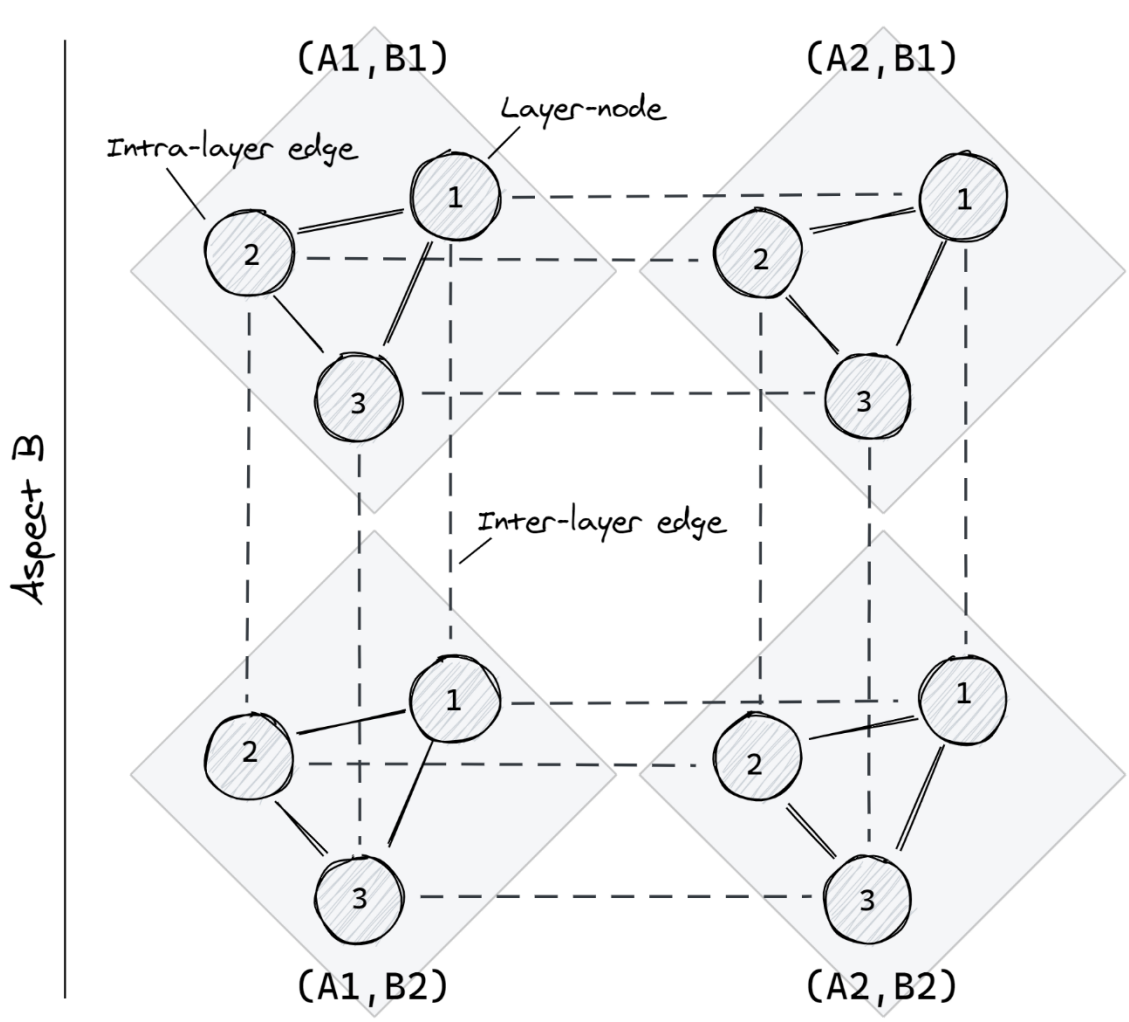

Figure 21. Multilayer networks define complex networks. *Each layer is a network with homogenous data types: one type of nodes (e.g. protein) and one type of edges (e.g. interactions). A group of layers sharing a feature is called an aspect. In this example, there are two aspects: A and B. Each aspect can have one of two versions of the feature. For example, an aspect could be the time with two time-points. Therefore, a layer is identified a tuple, e.g. (A1, B1). All layers share the same 3 nodes: (1), (2) and (3). Therefore, the multilayer network is node-aligned. A node in a given layer is called layer-node. It can have different values which are specific to the layer. Following the same principles, intra-layer edges link layer-nodes within a same layer. In comparison, inter-layer edges link layer-nodes between layers. The inter-layer edges only link layer-nodes and their counterparts. Thus, the multilayer network is diagonally coupled and it can be called multiplex network. Furthermore, this coupling is also ordinal because there are no inter-layer edges between layers (A1, B1) and (A2, B2).*

The definition of multilayer networks is general. For example, a node can be present on a single layer, i.e. it has a single layer-node, and a layer-node can connect any other layer-node. On the contrary, a popular form of a multilayer network is the *multiplex network*. Multiplex networks have *diagonal couplings*, i.e. all inter-layer edges are coupling edges. In other words, the nodes can only be connected to themselves. The diagonal coupling is *ordinal* when a layer is connected to a limited number of layers.

Often, the nodes are present in every layer. The multilayer network is thus said *node-aligned*. In section 6.4, I present multiplex networks with an ordinal coupling and one aspect corresponding to time. They are so-called *temporal multilayer network* (tMLN) (Pierrelée *et al.*, 2020). A virgin tMLN is node-aligned but this property is lost afterwards.

Multilayer networks enable to combine multiple -omics datasets without an aggregation step. For example, (Lu *et al.*, 2009) measured the fold-changes of genes on 4 layers: differential gene expression from RNA abundances, protein quantities from mass spectrometry, gene promoters with chromatin modifications and genes being transcribed by the RNA polymerase II. They determined at what layer (i.e. chromatin, transcription, post-transcription or post-translation) the cell regulates its pathways by comparing the agreement of fold-changes between layers. (Cantini *et al.*, 2015) identified communities from a multiplex network combining protein-DNA interactions (PDIs), micro-RNA interactions, protein-protein interactions (PPIs) and gene co-expression networks. They applied standard community-identification tools on each layer. Then, they clustered the layer-specific communities and computed the consensus community across layers. Yet, the methods of both studies did not use the inter-layer edges; they represented data with layers but without actually applying multilayer networks.

To consider the full potential of multilayer networks is challenging because it requires to distinguish between intra- and inter-layer edges. Most of the other properties of multilayer networks (e.g. diagonal coupling) are directly incorporated within the network structure and do not require to be considered by network measures. For example, (Cozzo *et al.*, 2015) defined clustering coefficients for multiplex networks by allowing node triangles to jump across layers. Highly clustered nodes tend to have many triangles. The triangles can have either all their sides on the same layers or on several layers. The side on another layer includes one intra-layer edge (i.e. the side itself) and one inter-layer edge (i.e. the jump from the layer to the other). Therefore, the multiplex clustering coefficient can be decomposed into the effects of "within-layer" triangles and "between-layer" triangles.

Rather than using clustering coefficients, algorithms such as random walks (see section 5.2) enable to identify multilayer communities. (Valdeolivas *et al.*, 2019) built a multilayer network including one layer with PPIs, a second layer with interactions from pathway databases, a third layer representing a gene co-expression network and a last layer with gene-disease interactions. They applied a random walk with restart to identify communities. The walker is able to jump within a layer (as usual) and between layers. Jumping from one layer to another depends on a probability set by the user. This method does not directly generate communities, but it scores the nodes to identify those with a particular interest. On the contrary, (De Domenico *et al.*, 2015) extended a common tool for community detection called **InfoMap** to multiplex networks. **Multiplex-InfoMap** applies random walks and summarizes the trajectories of the walkers. This summary results in communities where the walkers tend to be trapped. See (Huang *et al.*, 2021) for a review of community detection tools for multilayer networks.

There are few studies working on shortest paths for multilayer networks and none for subnetwork extraction. There are two main challenges: how to compare the layers and how to allow the jumps between the layers. To define shortest paths, (Magnani and Rossi, 2013) assumed walking on intra-layer edges is equivalent to walking on inter-layer edges. Yet, the authors assumed a path in one layer is not comparable to another. For example, on a multilayer network of roads where each type of road is a layer, if a path goes through 3 cities in the layer of country roads, it is not necessarily shorter than a path with 4 cities in the layer of highways. Therefore, the authors counted the number of steps (i.e. edges) in each layer for each path and compared only the steps from the same layer. A path is the shortest or one of the shortest paths if it has the lowest number of steps in each layer. For example, one path with **2** steps in layer **A** and **3** steps in layer **B** is shorter than another path with **2** steps in layer **A** but **4** steps in layer **B**. However, both paths have the same length as a path with **3** steps in layer **A** and **2** steps in layer **B**. This is the concept of *Pareto efficiency* that (Magnani and Rossi, 2013) applied to define a so-called *Pareto distance* to compute shortest paths. The Pareto efficiency is quite useful to make decisions based on multiple variables.

6.4 Temporal multilayer networks

Multilayer networks can represent temporal networks as *multilayer temporal networks*. Yet, standard –omics experiments cannot distinguish between actual and potential interactions. Moreover, the sampling frequency is often not high enough to apply temporal networks. *Multilayer temporal networks* are thus not appropriate to model biological processes because of these two practical issues.

To overcome the limits of temporal networks, I defined *temporal multilayer networks* (tMLN) (Pierrelée *et al.*, 2020) (**Figure 22**). They benefit from the flexibility of multilayer networks while representing time as an aspect. Each tMLN layer is a given time-point. This is why *temporal* is an adjective of *multilayer networks*. A temporal aspect implies the layers are ordered by time and the inter-layer edges are ordinal, such as the layer at time t is connected to the layers at times $t-1$ and $t+1$. To forbid going backward in time, all inter-layer edges are directed from t to $t+1$. Under this large definition, dynamic Bayesian networks (see section 4.2.2) can also be considered as tMLNs, but without intra-layer edges. In comparison, (Mucha *et al.*, 2010) applied a community detection method on temporal multilayer networks by computing statistics from the weights of both intra- and inter-layer edges. The statistics can then be used by monolayer-network analysis tools to detect communities across layers. Inter-layer edges are not directed but this information is not required for this approach.

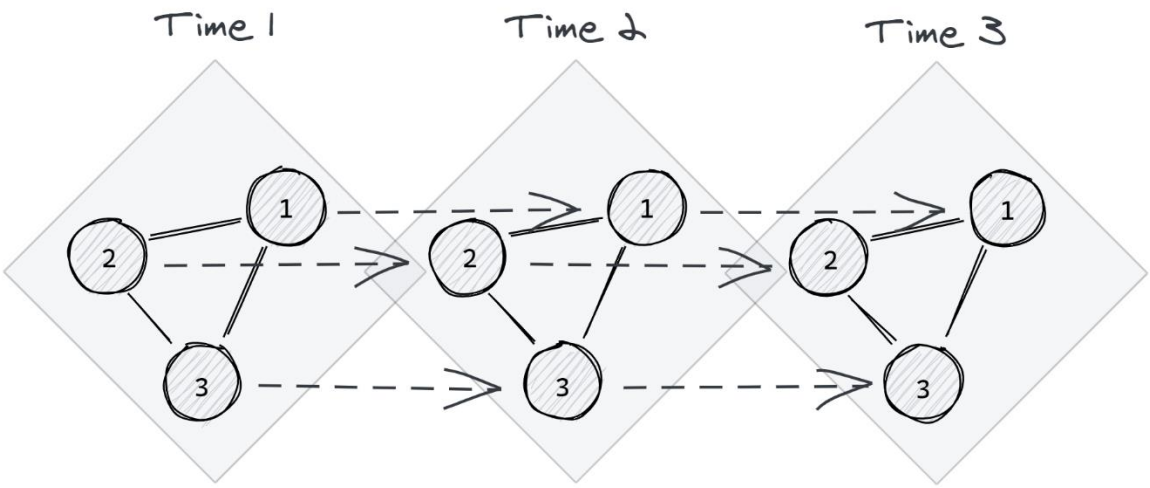

Figure 22. Temporal multilayer networks model time with one aspect. *In this example, the tMLN is a multiplex network with ordinal coupling. Inter-layer edges are directed following the time axis.*

As for temporal networks, no method was available to extract subnetworks from multilayer networks. To fill this gap, I developed **TimeNexus** to do so (Pierrelée *et al.*, 2020). TimeNexus is a **Cytoscape** app with three features: building **Cytoscape** objects representing multilayer networks, extracting subnetworks and visualizing multilayer networks; see Figure 1 of (Pierrelée *et al.*, 2020). The tool was applied to tMLNs. Yet, it is applicable for any kind of multilayer networks with a unique aspect and ordered layers as presented above. To illustrate the use of **TimeNexus**, a tMLN was built with specific properties. Protein-DNA and protein-protein interactions were aggregated into an interactome. This interactome was copied in each layer. The layer-nodes related to the same node were connected between consecutive layers. Thereby, this tMLN was a multiplex network with ordinal couplings. Intra-layer edges did not change over time as they represented potential interactions, but layer-nodes did. Intra-layer edge weights depended on the confidence score, while layer-node weights were computed from the scores of the differential expression analysis. Layer-nodes were tagged whether they were considered as dysregulated or not. As inter-layer edges connecting dysregulated nodes would be expected to be part of subnetworks, their weights depended also on the layer-node weights they were connecting.

As **Cytoscape** to not manage multilayer networks, tMLNs were represented by two monolayer networks: the *flattened network* and the *aggregated network* (**Figure 23**). Each of them represents one dimension of a multilayer network if one considers it as a 3D object. The flattened network projects the multilayer network on a flat surface such that intra- and inter-layer edges are not distinguishable. Moreover, each layer-node becomes an independent entity. The aggregated network stacks up the layer-nodes related to the same node into a single object. Intra-layer edges are stacked up as well. For a virgin multiplex tMLN, it is equivalent to reuse the structure of one layer. Therefore, the aggregated network loses the inter-layer edges. Flattened networks

enable to extract subnetworks and visualize multilayer networks. Aggregated networks are only used for visualization.

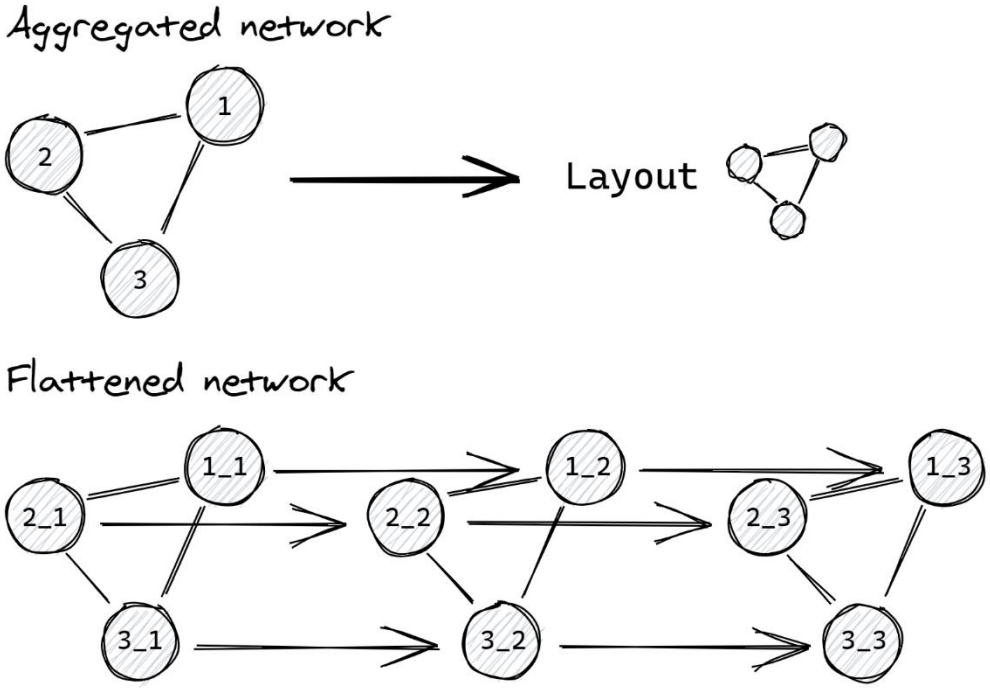

Figure 23. Aggregated and flattened networks are simple networks representing tMLNs. *In an aggregated network, all layers are merged into a single network. Layer-nodes related to the same node are aggregated into a single node. Intra-layer edges are also aggregated into one. An edge is added to the aggregated network if there is at least an intra-layer edge in one layer. Inter-layer edges are lost. On the other hand, the flattened network does not lose inter-layer edges and do not merge layer-nodes, but the layers are lost. It implies that layer-nodes are converted into simple nodes. A tag is added to node names to recognize the layer of origin. Moreover, the intra- and inter-layer edges are not distinguishable anymore. The layout of the aggregated network serves to create the layout of the flattened network.*

To extract subnetworks, I used two algorithms available in **Cytoscape**: the *extracting apps* **PathLinker** and **AnatApp** (see section 5.1.4 and 5.1.5) because they have an API enabling **TimeNexus** to directly call them within **Cytoscape**. It executes the app selected by the user on the flattened network after a processing step adapting the network format. Indeed, both apps require a different network format. The apps also need source and target nodes from the flattened network. Selecting these nodes depend on the extraction method. With the *global method* (**Figure 24**), the source and target nodes are the nodes corresponding to the layer-nodes of the first and last layer, respectively. **TimeNexus** then sends the flattened network in the correct format to **PathLinker** and **AnatApp**, gets the nodes of the extracted subnetwork and removes the nodes which are not in this list from the flattened network. This generates a new multilayer network corresponding to the active multilayer subnetwork. On the contrary, the *local method* (**Figure 25**) applies the subnetwork extraction on each layer independently. To do so, **TimeNexus** splits the flattened network according to the layers of origin and sends the parts to one of the extracting apps. Next, it combines the extracted nodes from each part into a single list, generating the new multilayer

network. The *pairwise method* (**Figure 26**) does the same but on a group of two consecutive layers. This method is thus a compromise between the global and the local methods. The former does not consider intermediate layers and the latter ignores inter-layer edges. The multilayer subnetworks generated by TimeNexus lose their node-alignment.

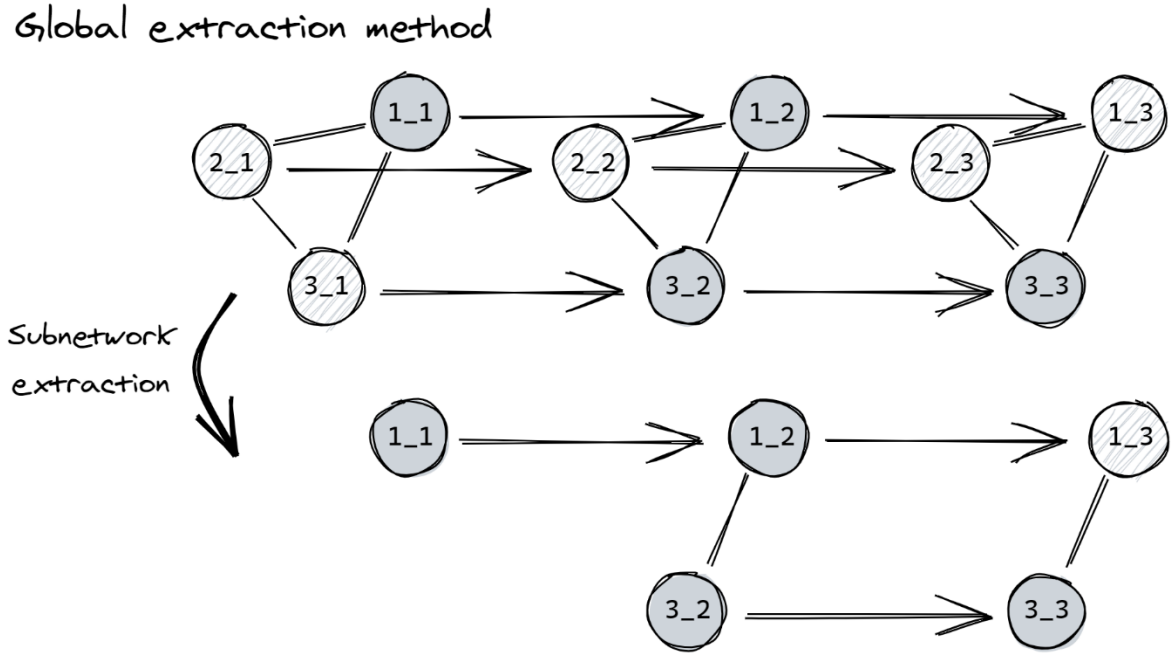

Figure 24. Global extraction method applies extraction on the full flattened network. *The top network is the flattened network from* ***Figure 23*** *and the bottom subnetwork is the extracted subnetwork. The latter is then converted back into a tMLN. Grey nodes are input nodes (e.g. dysregulated genes).*

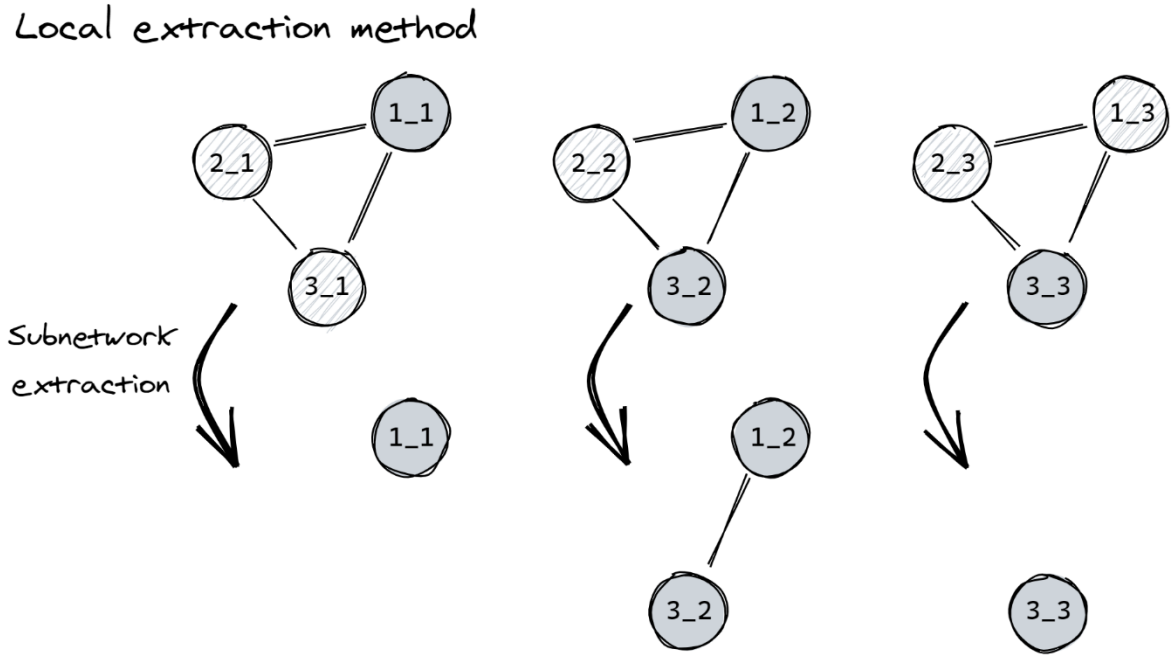

Figure 25. Local extraction method applies extraction on each layer of the tMLN. *It does not account for inter-layer edges. The subnetworks from each layer are merged and converted into a tMLN afterwards. Grey nodes are input nodes (e.g. dysregulated genes).*

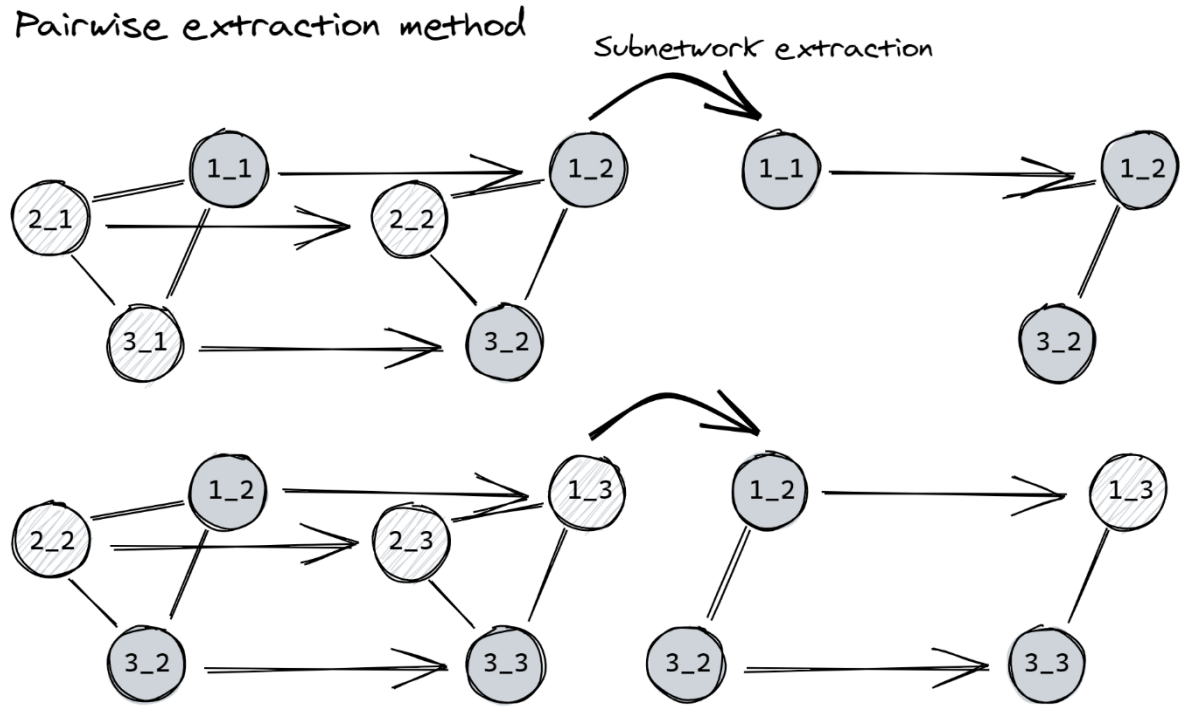

Figure 26. Pairwise extraction method applies extraction on each pair of consecutive layers. *Each pair is converted into a flattened network on which extraction is applied. The subnetworks from each pair are merged and converted into a tMLN afterwards. Grey nodes are input nodes (e.g. dysregulated genes).*

Finally, the visualization feature creates a view for the **Cytoscape** object containing the flattened network. The nodes are grouped according to their layer of origin. All groups get the same layout corresponding to the layout of the aggregated network such as the layer-nodes related to the same node are aligned. This ensures a consistent view simulating the multilayer network structure.

Chapter 1 – *Chlorella* sp. HS2 paper

Transcriptomic analysis of *Chlorella* sp. HS2 suggests the overflow of acetyl-CoA and NADPH co-factor induces high lipid accumulation and halotolerance

Jin-Ho Yun[1,†,^], Michaël Pierrelée[2,†], Dae-Hyun Cho[1], Urim Kim[1,3], Jina Heo[1,3], Dong-Yun Choi[1], Yong Jae Lee[1], Bongsoo Lee[4], HyeRan Kim[5], Bianca Habermann[2], Yong Keun Chang[6,7], Hee-Sik Kim[1,3,*]

[1]Cell Factory Research Center, KRIBB, Yuseong-gu, Daejeon, 34141, Republic of Korea
[2]Aix Marseille University, CNRS, IBDM, Marseille, France
[3]Department of Environmental Biotechnology, UST, Yuseong-gu, Daejeon, 34113, Republic of Korea
[4]Department of Microbial and Nano Materials, College of Science and Technology, Mokwon University, Seo-gu, Daejeon, 35349, Republic of Korea
[5]Plant Systems Engineering Research Center, KRIBB, Yuseong-gu, Daejeon, 34141, Republic of Korea
[6]Advanced Biomass R&D Center, 291 Daehak-ro, Yuseong-gu, Daejeon, 34141, Republic of Korea
[7]Department of Chemical and Biomolecular Engineering, KAIST, Yuseong-gu, Daejeon, 34141, Republic of Korea

*Corresponding author: hkim@kribb.re.kr
[†]These authors contributed equally to this work.
[^]Current address: Bigelow Laboratory for Ocean Sciences, East Boothbay, ME 04544 USA

Funding

This work was supported by "Carbon to X Project" and the Advanced Biomass R&D Center (ABC) of the Global Frontier Program funded by the Ministry of Science and ICT of the Republic of Korea (2020M3H7A1098291, 2016922286), by grants from Marine Biotechnology Program and Collaborative Genome Program funded by the Ministry of Oceans and Fisheries of the Republic of Korea (20150184, 20180430); and a grant from KRIBB Research Initiative Program (www.kribb.re.kr).

Abstract

Previously, we isolated *Chlorella* sp. HS2 (referred hereupon as HS2) from a local tidal rock pool and demonstrated its halotolerance and high biomass productivity under different salinity conditions. To further understand acclimation responses of this alga under high salinity stress, we performed transcriptome analysis of triplicated culture samples grown in freshwater and marine conditions at both exponential and stationary growth phases. The results indicated that the transcripts involved in photosynthesis, TCA and Calvin cycles were downregulated, whereas the upregulation of DNA repair mechanisms and an ABCB subfamily of eukaryotic type ABC transporter was observed at high salinity condition. In addition, while key enzymes associated with glycolysis pathway and triacylglycerol (TAG) synthesis were determined to be upregulated from early growth phase, salinity stress seemed to reduce the carbohydrate content of harvested biomass from 45.6 dw% to 14.7 dw% and nearly triple the total lipid content from 26.0 dw% to 62.0 dw%. These results suggest that the reallocation of storage carbon toward lipids played a significant role in conferring the viability of this alga under high salinity stress by remediating high level of cellular stress partially resulted from ROS generated in oxygen-evolving thylakoids as observed in a direct measure of photosystem activities.

Summary Statement

Allocation of storage carbon towards the synthesis of lipids seemed to play a critical role in conferring the halotolerance of a *Chlorella* isolate by remediating excess oxidative stress experienced in photosystems.

Keywords: acetyl-CoA, *Chlorella* sp. HS2, halotolerance, lipid synthesis, photosynthesis, RNA-seq

1. Introduction

Microalgae exhibit a greater biomass yield than most terrestrial crops and can be grown with excess nutrients in wastewater sources, prompting its industrial utilization as a biofeedstock for the production of nutraceuticals, pharmaceuticals, cosmetics, and biofuels (Hu et al., 2008; Quinn & Davis, 2015; Smith, Sturm, Denoyelles, & Billings, 2010; Unkefer et al., 2017; Yun, Cho, Lee, Heo, et al., 2018). However, commercial production of algal biomass is not yet considered to be economically competitive because of high energy inputs associated with biomass harvesting and downstream extraction of desirable biomolecules (Laurens et al., 2017; Stephens et al., 2010; Valizadeh Derakhshan, Nasernejad, Abbaspour-Aghdam, & Hamidi, 2015). Importantly, the productivity and operational stability of algal cultivation platforms are prone to be compromised by unpredictable meteorological conditions and culture contamination (McBride et al., 2014; Wang et al., 2016; Yun et al., 2019; Yun, Cho, Lee, Kim, & Chang, 2018; Yun, Smith, La, & Keun Chang, 2016), which has led to multifactorial efforts to develop robust algal "crops" under changing environments, just as in the case of conventional agriculture.

Of environmental conditions that determine the productivity of biomass and desirable biomolecules from industrial crops, salinity appears on the top of the list because of high crop sensitivity to presence of high concentrations of salts in the soil or waters (Flowers, Troke, & Yeo, 1977; Peng et al., 2014; Yuge Zhang & Liang, 2006). In particular, the extensive application of chemical fertilizer facilitates accumulation of salts in agricultural fields, which in turn could lead to a positive feedback loop by necessitating an increased application of synthetic fertilizer (Yuge Zhang & Liang, 2006). Notably, industrial algal cultivation platforms require continuous provision of nutrient salts with some studies demonstrating the utilization of saline wastewater sources enriched with nitrogenous and phosphorus nutrients as growth media to drive down the costs of commercial operation of algal cultivation systems (Yun, Cho,

Lee, Heo, et al., 2018; Yun, Smith, & Pate, 2015; Zhu et al., 2013). In addition, the direct application of salinity stress for algal cultivation systems has been demonstrated as an effective abiotic inducer of high lipid accumulation and an environmental barrier inhibiting the proliferation of undesirable alien invaders in cultivation systems (Church et al., 2017; Kakarla et al., 2018; Lee, Nam, Yang, Han, & Chang, 2016). Kakarla *et al.*, for instance, supplemented 60 g/L of NaCl into concentrated *Chlorella* cultures for 48 hr and reported ca. 58% increase in algal lipid productivity, supporting the possibility of deploying high salinity stress as a promising post-treatment for the cultivation systems targeting to produce algal lipids (Kakarla et al., 2018). Moreover, while high salinity stress could act as an effective method of crop protection in reducing freshwater cyanobacterial or ciliate contaminants, it was successfully demonstrated to facilitate algal harvesting by enlarging cellular diameter and increasing algal settling rates (Church et al., 2017; Lee et al., 2016; von Alvensleben, Stookey, Magnusson, & Heimann, 2013). Even though general osmosensitivity of algal crops has been acknowledged (Flowers et al., 1977), there is thus a great industrial incentive to exploit algal diversity and especially high tolerance of some algal species to highly saline environment (Yun et al., 2015).

With the apparent advantages of incorporating high salinity stress into the management of industrial algal cultivation platforms, bioprospecting halotolerant algal strains that exhibit high and reliable production of biomass and/or desirable biomolecules was the motivation of our previous study in which a halotolerant *Chlorella* sp. was isolated from a tidal rock pool (Yun et al., 2019). While the remarkable toughness of *Chlorella* under different physical and chemical stress and its recognition as one of a handful of successful industrial crops have been well documented (Fogg, 2001; Yun et al., 2019), this isolated *Chlorella* sp. HS2 (referred to hereupon as HS2) exhibited relatively high growth under a wide range of salinity conditions (i.e., 0-7% (w/v) of supplemental NaCl) compared to reference *Chlorella* strains (Yun et al., 2019). Importantly, substantial shifts in the composition of fatty acid methyl ester (FAME) and

the amount of carotenoid pigments under different salinity conditions led us to speculate that elucidating mechanisms behind relatively short-term (i.e., few days) algal acclimation to high salinity stress would enable maximizing the industrial potential of HS2 by guiding ongoing efforts in metabolic and process engineering (Oh, Chang, & Lee, 2019; Rathinasabapathi, 2000; Yun et al., 2019).

In previous studies, transcriptome analysis has served as an important tool to understand intricate algal responses to changing salinity conditions. For example, Foflonker *et al*. challenged *Picochlorum* cells with high or low salinity shock and used transcriptomic and chlorophyll fluorescence analyses to elucidate salinity-tolerance mechanisms (Foflonker et al., 2016); the authors identified photoprotective mechanisms, oxidative stress response, cell wall and membrane rearrangement, nitrogen assimilation, and diverting resources from growth and PSII repair in favor of maintaining homeostasis as the main responses against a challenging environment (Foflonker et al., 2016). Moreover, Perrineau *et al*. compared salt-acclimated and progenitor populations of *Chlamydomonas reinhardtii*, and reported downregulation of genes involved in the salt stress response (most notably, glycerophospholipid signaling) and in transcription/translation in the salt-acclimated populations, suggesting gene-rich mixotrophic algal lineages could rapidly adapt to high salinity conditions (Perrineau et al., 2014). Importantly, the survey of existing literature suggested the presence of strain-specific algal responses that could be closely associated with the phenotypic characteristics of an algal strain of interest (Erdmann & Hagemann, 2001).

Herein, we report the transcriptome of HS2 grown in freshwater and marine conditions to accomplish mechanistic understanding of algal acclimation to high salinity stress. Triplicated cultures samples were first obtained at exponential and stationary growth phases in freshwater and marine growth media for RNA-seq analysis, and the proximate analysis of harvested biomass was additionally performed along with the measure of photosystem II (PSII) activity.

Combined together with the results in our previous study, we were able to elucidate how vital metabolic pathways were shifted under high salinity stress, and an important role of allocating storage carbon towards the synthesis of lipids in conferring the viability of HS2 and remediating high oxidative stress under high salinity stress.

2. Materials and Methods

2.1. Strain selection and cultivation conditions

HS2 was previously isolated from a local tidal rock pool, and its high tolerance to a wide range of salinity conditions was acknowledged (Yun et al., 2019). While the results of HS2 cultivation in 1-L cylindrical PBRs were reported in our previous study (Yun et al., 2019), both autotrophic cultures grown in freshwater inorganic medium and in marine inorganic growth medium supplemented with 3% (w/v) sea salt were subjected to transcriptome analysis. These triplicated cultures were grown under pre-determined optimal light and temperature conditions with continuous supplementation of 5% CO_2 at 0.2 vvm and agitation at 120 rpm.

2.2. PSII activity measurement and proximate analysis

While pigment and FAME composition of harvested HS2 biomass in both freshwater and marine conditions were reported previously, photoautotrophically grown cells in exponential and stationary growth phases were subjected to measurements of the photosynthetic parameters *in vivo* using Multi-Color-PAM (Heinz Walz, Germany) (Shin et al., 2017). After adapting cells under dark condition for 20 min, the light response curves of the relative electron transport rate (rETR), the quantum yields of non-photochemical quenching (Y(NPQ)) and nonregulated excess energy dissipation (Y(NO)) were measured in biological triplicates while increasing the actinic light intensities of 440 nm LEDs with a step width of 2 min (Shin et al., 2017). In addition, proximate analysis of the biomass harvested at stationary growth phase was performed in biological triplicates to further elucidate metabolic shifts in

HS2 under high salinity stress. The lipid content of harvested biomass was first analyzed by extracting total lipids from freeze-dried biomass with chloroform-methanol (2:1 (v/v)) following a slightly modified version of Bligh and Dyer's method (Bligh & Dyer, 1959). Sample-solvent mixtures were then transferred into a separatory funnel and shaken for 30 min and the lipid fraction was separated from the separatory funnel; the solvent was evaporated using a rotary evaporator and the weight of the crude lipid obtained from each sample was measured using an analytical balance following Yun *et al* (Yun, Cho, Lee, Heo, et al., 2018). In addition, the protein content was determined using the method of Lowry using ca. 2 mg (dry weight) of the cell pellet resuspended in 0.5 ml of 1 M NaOH and boiled for 5 min (Illman, Scragg, & Shales, 2000; Lowry, Rosebrough, Farr, & Randall, 1951); the carbohydrate content was measured using the phenol sulfuric acid method of Dubois *et al*. using ca. 0.5 mg (dry weight) of the cell pellet resuspended in 1 ml of water (Dubois, Gilles, Hamilton, Rebers, & Smith, 1956; Illman et al., 2000). Finally, the ash content was analyzed gravimetrically after exposing dry biomass to 500 °C in a muffle furnace for 8 hours (Kent, Welladsen, Mangott, & Li, 2015).

2.3. RNA extraction, library construction, and Illumina sequencing

Each of salt-stressed and control PBR cultures was harvested during exponential and stationary growth phases by centrifugation at 4500 rpm for 10 min. Total RNA was then extracted using the Trizol reagent (Invitrogen, Carlsbad, CA, USA), according to manufacturer's instructions. Subsequently, the RNA samples were treated with DNase I for 30 min at 37 °C to remove genomic DNA contamination, and the quantity and integrity of the total RNA were verified using an Agilent 2100 bioanalyzer. The cDNA libraries were developed according to manufacturer's instructions (Illumina, Inc., San Diego, CA, USA), and sequenced on the Illumina HiSeq 2000 platform at Seeders Co. (Daejeon, Korea) (Liu et al., 2017). In addition, RNA-Seq paired end libraries were prepared using the Illumina TruSeq RNA Sample

Preparation Kit v2 (catalog #RS-122-2001, Illumina, San Diego, CA). Starting with total RNA, mRNA was first purified using poly (A) selection or rRNA depletion, then RNA was chemically fragmented and converted into single-stranded cDNA using random hexamer priming; the second strand was generated next to create double-stranded cDNA. Library construction began with generation of blunt-end cDNA fragments from ds-cDNA. Thereafter, A-base was added to the blunt-end in order to make them ready for ligation of sequencing adapters. After the size selection of ligates, the ligated cDNA fragments which contained adapter sequences were enhanced via PCR using adapter specific primers. The library was quantified with KAPA library quantification kit (Kapa biosystems KK4854) following the manufacturer's instructions. Each library was loaded on Illumina Hiseq2000 platform, and the desired average sequencing depth was met while performing high-throughput sequencing.

2.4. *De novo* assembly and analysis

De novo assembly was performed using Trinity 2.8.5 (Grabherr et al., 2011) using raw 100 bp paired-end reads. Assembly quality assessment was carried out with BUSCO 3.0.2 (Simão, Waterhouse, Ioannidis, Kriventseva, & Zdobnov, 2015), for which the chlorophyte database of OrthoDB 10 was employed as datasets at an e-value cutoff of 1e-5 (Kriventseva et al., 2018); high-quality reads were mapped onto genome sequences using Bowtie2 2.3.5. Thereafter, the quantification of the number of reads (i.e., counts mapped per transcripts) was performed following alignment and abundance estimation of each Trinity script using RSEM 1.3.2 and Bowtie 1.2.2, respectively (Langmead, Trapnell, Pop, & Salzberg, 2009; B. Li & Dewey, 2011). Transcripts with no count across all sampling points were removed. The matrix of counts for unigenes (i.e., a collection of expressed sequences that are aligned or located to the same position on genome) was used for downstream analyses.

2.5. DEG analysis and functional annotation

Prior to functional annotation, differential expression analysis (DEA) was performed

first to avoid determining the most relevant transcript for each unigene based on unnecessary assumptions at the early stage. In addition, given that quantitative asymmetry between up- and downregulated unigenes was strong, SVCD 0.1.0, which does not assume the lack-of-variation between up- and downregulated unigene counts (Evans, Hardin, & Stoebel, 2017; Roca, Gomes, Amorim, & Scott-Fordsmand, 2017), was used for normalization of unigenes. The mean of raw counts greater than the first quartile (i.e., 5.9 raw counts) as recommended (Roca et al., 2017) was used during normalization. To determine DEGs, we used DESeq2 1.20.0, and the DEGs between exponential and stationary growth phases were based on the adjusted p-values (i.e., DEGs were determined as unigenes with adjusted p-value < 0.01).

Functional annotation of DEGs was subsequently performed using Swiss-Prot, Pfam, and Kyoto Encyclopedia of Genes and Genomes (KEGG) databases. First, following Trinotate 3.2.0's recommendation, we predicted transcript coding regions that could be assigned to putative proteins using TransDecoder 5.5.0 (Haas et al., 2013). Thereafter, homologies were identified using in parallel BLASTp from BLAST+ 2.9.0; to identify pfam domains, *hmmscan* from HMMER 3.2.1 was used (Camacho et al., 2009; Eddy, 2011). BLASTp and *hmmscan* were run twice from the predicted proteins. SignalP 5.0b (http://www.cbs.dtu.dk/services/SignalP/) was used to determined eukaryotic signal peptides within transcripts. We also used BLASTx to find homologues, which allows to identify sequence similarities within all six reading frames of the transcript. All BLAST runs were performed against the Swiss-Prot database through DIAMOND 0.8.36 (Buchfink, Xie, & Huson, 2015) with an e-value cutoff of 1e-10. Then, KEGG cross-references associated with BLASTx or BLASTp hits were retrieved to assign each BLAST hit with a KEGG Orthology number (KO). Transcripts without a BLASTx or BLASTp hit were excluded, and a pair of transcript and coding region was removed when the KOs of corresponding transcript and coding regions were not identical. In addition, when one gene had multiple KOs, the mean of

average e-values was computed and the KO with the lowest mean was selected as the most relevant KO. Metabolic pathway maps were constructed using KEGG mapper based on the organism-specific search results of *Chlorella variabilis* (cvr) and biological objects for each KO were determined using KEGG BRITE. Enrichment was performed by implementing GSEAPreranked from Gene Set Enrichment Analysis with the conda package GSEApy 0.9.15 (Mootha et al., 2003; Subramanian et al., 2005). A term was considered to be significantly enriched when its false discovery rate (FDR) was lower than 0.25. All data generated from our transcriptome analysis are available at the NCBI GEO repository: GSE146789 at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146789.

3. Results

3.1. Phenotypic shifts of HS2 under high salinity stress

Shifts in growth, FAME and pigment composition of HS2 during autotrophy in freshwater (i.e., 0% (w/w) of supplemental sea salt) and marine (i.e., 3% (w/w) of supplemental sea salt) media were reported in the previous study (Yun et al., 2019). Briefly, the results indicated a nearly 10-fold decrease in the maximum cell density of the autotrophic PBRs in marine medium at stationary growth phase, whereas only a two-fold decrease in the average dry cell weight (DCW) was observed (Yun et al., 2019) (Supplementary Figure 1). As microscopic observation revealed, a non-proportional decrease in DCW of HS2 under high salinity stress corresponded to roughly 50% increase in cellular diameter or 3.4-fold increase in cellular volume. While previous study also reported substantial decreases in the amount of algal pigments and relative amount of polyunsaturated fatty acids under high salinity stress (Yun et al., 2019), TEM images of harvested algal cell suggested the formation of large lipid droplets under high salinity stress (Fig. 1): indeed, proximate analysis of harvested biomass indicated a significant increase in lipid content from 25.0 dw% to 62.0 dw% under high salinity

stress, contrasting a nearly three-fold decrease in the amount of carbohydrate (Figs. 1 and 2).

While relatively high amounts of carotenoid pigments (i.e., β-carotene and lutein) under high salinity stress observed in the previous study suggested their possible contribution to the protection of photosynthetic machinery (Talebi, Tabatabaei, Mohtashami, Tohidfar, & Moradi, 2013; Yun et al., 2019), the measures of relative electron transport rate (rETR), the quantum yields of non-photochemical quenching (Y(NPQ)) and non-regulated excess energy dissipation (Y(NO)) using multi-color-PAM indicated that rETR was reduced early during the exponential growth phase under high salinity stress and was recovered at later stationary growth phase. Although differences in Y(NPQ) and Y(NO) were not observed respectively at exponential and stationary phases, a significant difference in Y(NPQ) was observed during stationary phase only at high light intensities and Y(NO) of salt-shocked culture was significantly greater than that of control across all light intensities during exponential growth phase (Fig. 3).

3.2. Summary of *de novo* assembly

To determine differential transcriptomic regulation of HS2 under freshwater and marine conditions, RNA-seq was performed using Illumina Hiseq 2000 platform, followed by *de novo* RNA-seq assembly and mapping of data to the newly assembled and processed transcriptome. Alignment statistics from Trinity and Bowtie2 2.3.5 mapping results were summarized in Supplementary Table 1. Overall, 57640 unigenes were obtained out of 290 million raw reads, and the assessment of assembly quality indicated 89% of complete BUSCOs following the removal of 4870 unigenes with 0 count in any of the treatments.

3.3. Functional annotation of differentially expressed genes

To elucidate differentially expressed genes (DEGs), read normalization was first performed using SVCD normalization following standard DEGseq2 statistical test; a total of 9117 DEGs were subsequently obtained from 52770 unigenes corresponding to 39469

transcripts. While 3573 DEGs were commonly observed across all conditions, 2334 and 3120 DEGs were distinctively observed at exponential and stationary phases, respectively (Fig. 4). Overall, global observation of transcriptome changes indicated general transcriptional downregulation under high salinity stress, highlighting substantial metabolic constraints and subsequent biochemical shifts that presumably facilitated the survival of algal cells under high salinity stress. It should be also noted that a substantial difference in terms of the overall DEG expression was observed between exponential and stationary growth phases, with more transcriptional shifts towards downregulation during stationary growth phase. Finally, KO annotation of DEGs yielded 2795 DEGs (i.e., 31% of all DEGs) with 1982 unique consensus KOs, and these DEGs represented one third of genes of *Chlorella variabilis* NC64A's genome (Eckardt, 2010).

3.4. Functional enrichment of differentially expressed genes

Enrichment analysis was performed with the first and second elements of functional hierarchies of KEGG BRITE. While the terms with a p-value lower than 0.05 and a false discovery rate (FDR) equal to or lower than 0.25 were considered to be enriched, the results indicated high enrichment of ribosomal proteins (Fig. 5). In addition, papain family of intramolecular chaperones and heparan sulfate/heparin glycosaminoglycan binding proteins were enriched. Notably, even though FDR values below the cutoff were not observed, many enriched terms with a p-value lower than 0.05 were related to protein processing and membrane trafficking.

3.5. KEGG pathway analysis

To elucidate metabolic pathways associated with the acclimation of HS2 to high salinity stress, we mapped DEGs to 120 reference KEGG pathways; pathways enriched with 20 or more DEGs were summarized in Supplementary Table 2.

3.5.1. Genes involved in cell cycle and DNA replication

Upon exposure to high salinity stress, the growth of HS2 seemed to be inhibited with an apparent enlargement of cellular biovolume (see 3.1). Correspondingly, most unigenes homologous to genes identified to be involved in cell cycle were downregulated (Table 1). Additionally, DNA replication seemed to be downregulated as well, although Mcm4 of MCM complex (helicase) and DNA polymerase delta subunit 1 [EC: 2.7.7.7] were upregulated (Supplementary File 1), suggesting the inhibition of DNA replication under high salinity stress. Likewise, most of the unigenes associated with RNA degradation seemed to be downregulated under high salinity stress (Table 1), except CNOT3 (Supplementary File 1). Furthermore, most genes associated with RNA transport seemed to be downregulated under high salinity stress; and genes associated with aminoacyl-tRNA biosynthesis were downregulated, except glutaminyl-tRNA synthetase [EC: 6.1.1.18] and cysteinyl-tRNA synthetase [EC: 6.1.1.16] (Table 1 and Supplementary File 1). Although these results generally supported the impairment of both DNA and RNA processing under high salinity stress, it should be emphasized that a number of unigenes associated with repair mechanisms (i.e., nucleotide excision repair, base excision repair, mismatch repair) seemed to be upregulated (Supplementary File 1).

3.5.2. Genes involved in protein processing, MAPK signaling pathway, and ABC transporters

While salinity stress is known to substantially influence the processing and function of protein (Erdmann & Hagemann, 2001; Perrineau et al., 2014), the results indicated the downregulation of enzymes associate with protein processing in endoplasmic reticulum, except mannosyl-oligosaccharide alpha-1,3-glucosidase [EC:3.2.1.207] (GIcII), protein disulfide-isomerase A6 [EC: 5.3.4.1], and protein transport protein SEC24 (Table 1 and Supplementary File 1). Moreover, most of the ribosomal proteins were downregulated under high salinity stress: of 89 unigenes enriched on KEGG mapper's ribosome pathway, only S9, S16, and S26e of ribosomal proteins seemed to be upregulated at the exponential or stationary growth phases. In addition, while mitogen activated protein kinase (MAPK) signaling cascades are widely

recognized for their role in stress response and signal transduction in eukaryotes (Yang, Suh, Kang, Lee, & Chang, 2018), most of the genes associated with MAPK signaling pathway seemed to be downregulated, except P-type Cu^{+} transporter (RAN1) (Table 1). Although enriched unigenes indicated that all of the genes associated with protein export were also downregulated under high salinity stress, 3 protein subunits associated with the PA700 (base) of proteasome seemed to be upregulated along with an ABCB subfamily of ABC transporters (i.e., ATM) under high salinity stress (Supplementary File 1).

3.5.3. Genes involved in photosynthesis and Calvin cycle

There was a clear trend that all of the genes associated with PSII and PSI were downregulated from exponential phase under high salinity stress, corroborating with the measure of PSII activity that indicated a significant reduction in rETR during early growth phase. It should be, however, noted that these genes seemed to be less-downregulated or reverse its downregulation at later stationary growth phase (Table 1 and Supplementary File 1). Notably, there were more than 3-fold downregulation of transcripts (based on $\log_2$ fold change) associated with PSI-D, -E, -F, -H, -K and -O subunits and PSII Psb27 protein during exponential growth phase under high salinity stress; however, most of the transcripts associated with these subunits were upregulated during stationary growth phase, except those associated with PSI-K and PSII Psb27, which exhibited the downregulation with less than an absolute $\log_2$ fold change of 1.0 (Supplementary File 1). Similarly, all of the proteins associated with light harvesting complex (LHC) of HS2 seemed to be downregulated initially under high salinity stress at transcriptional level, whereas Lhcb2 and Lhcb4 were upregulated at the later growth phase.

While these results suggested an early compromise in photosynthesis, it should be pointed out that most of the enriched genes involved in carbon fixation via Calvin cycle were downregulated as well (Table 1 and Supplementary File 1). However, the upregulation of

alanine transaminase [EC: 2.6.1.2] was observed under high salinity stress and no differential expression in RUBISCO [EC: 4.1.1.39] was observed. In addition, although the results of our transcriptome analysis did not indicate differential expression of ferredoxin-$NADP^+$ reductase, an enzyme that catalyzes the reaction generating NADPH in PSI (Medina & Gómez-Moreno, 2004), malate dehydrogenase (oxaloacetate-decarboxylating) ($NADP^+$) [EC: 1.1.1.40], the third-class malic enzyme that catalyzes the oxidative decarboxylation of malate to pyruvate by the reduction of $NADP^+$ into NADPH, was upregulated during the exponential growth phase (Spaans, Weusthuis, Van Der Oost, & Kengen, 2015). Furthermore, our transcriptome analysis suggested that glucose-6-phosphate 1-dehydrogenase [EC: 1.1.1.49], one of the key enzymes involved in the generation of NADPH during the oxidative phase of pentose phosphate pathway, was substantially upregulated (Spaans, Weusthuis, Van Der Oost, & Kengen, 2015). It is thus likely that these enzymes associated with central carbon metabolism played a significant role in enhancing NADPH supply upon the induction of high salinity stress.3.5.4. Genes associated with glycolysis and TCA cycle

High salinity stress seemed to induce the upregulation of important genes associated with the conversion of glucose to acetyl-CoA (Table 1 and Supplementary File 1). In particular, pyruvate dehydrogenase E1 component alpha subunit [EC: 1.2.4.1], which is involved in the first step of converting pyruvate to acetyl-CoA was upregulated along with pyruvate decarboxylase [EC: 4.1.1.1]. Moreover, phosphoglucomutase [EC: 5.4.2.2], the enzyme involved in the first step of glycolysis, was upregulated. On the contrary, our results clearly indicated the downregulation of TCA cycle under high salinity stress: most unigenes corresponded to the known genes on TCA cycle were downregulated, suggesting the inhibition of cellular respiration (Table 1 and Supplementary File 1). In particular, 3 transcripts associated with citrate synthase [EC: 2.3.3.1], which mediates the first step of TCA cycle of converting acetyl-CoA to citrate, were substantially downregulated during both growth phases; and a

transcript associated with isocitrate dehydrogenase [EC: 1.1.1.42], which catalyzes the rate-limiting step of the oxidative decarboxylation of isocitrate to α-ketoglutarate, was downregulated during exponential growth phase (Bellou & Aggelis, 2013). Collectively, these results suggested that acetyl-CoA became more available for other cellular metabolisms, including lipid synthesis, under high salinity stress (Bellou & Aggelis, 2013).

3.5.5. Genes associated with fatty acid and TAG accumulation

Although the genes involved in the synthesis of fatty acids at upstream were downregulated, fatty acyl-ACP thioesterase A [EC: 3.1.2.14] and acyl-desaturase [EC: 1.14.19.2] were upregulated. Provided that the combined amount of C16:1, C18:0, and C18:1 was increased under high salinity stress (Yun et al., 2019), it is especially notable that these two upregulated genes are directly associated with the synthesis of these groups of monosaturated or saturated fatty acids. Moreover, while the genes enriched on KEGG mapper indicated that fatty acid elongation and the biosynthesis of unsaturated fatty acids were not upregulated, survey of the fatty acid degradation pathway indicated the inhibition of fatty acid degradation under high salinity stress (Table 1 and Supplementary File 1). Most notably, transcripts associated with acyl-CoA dehydrogenase [EC:1.3.8.7], enoyl-CoA hydratase [EC:4.2.1.17], and acyl-CoA oxidase [EC:1.3.3.6] were substantially downregulated during both exponential and stationary growth phases. Given that these enzymes facilitate fatty acid β-oxidation in mitochondria or in peroxisome (Gross, 1989; Kong et al., 2017), the results suggested their role in decreasing fatty acid turnover rate and in possibly preserving fatty acids under high salinity stress.

As the upregulation of lipid synthetic pathway in marine medium was postulated based on the increased lipid content in harvested biomass (see 3.1), the transcriptome analysis also identified that genes essential for the synthesis of triacylglycerol (TAG) were upregulated: both phosphatidate phosphate [EC: 3.1.3.4] and diacylglycerol O-acyltransferase 2 [EC: 2.3.1.20]

that both are involved in the conversion of 1,2,-Diacyl-sn-glycerol 3-phosphate to 1,2,-Diacyl-sn-glycerol and in the generation of TAG from 1,2,-Diacyl-sn-glycerol seemed to be substantially upregulated under high salinity stress during early growth phase.

3.5.6. Genes associated with carotenoid synthesis

Of 5 unigenes enriched on KEGG mapper's carotenoid biosynthesis pathway, all of the genes were downregulated, including a gene involved in the conversion of alpha-carotene to lutein (i.e., carotenoid epsilon hydroxylase [EC: 1.14.14.158]) (Supplementary File 1). In addition, two genes associated with the conversion of phytoene to lycopene, an important intermediate for the synthesis of other carotenoids, were downregulated (i.e., zeta-carotene isomerase [EC: 5.2.1.12] and zeta-carotene desaturase [EC: 1.3.5.6]) (Supplementary File 1). Interestingly, both relative and absolute amounts of lutein were increased under high salinity stress (Yun et al., 2019); these results suggest the provision of far-upstream precursors could have played an important role in lutein synthesis.

4. Discussion

Given that high salinity stress strongly influences the viability and biochemical composition of algal crops and thus the economic feasibility of entire algal biorefinery (Kakarla et al., 2018; Laurens et al., 2017; Oh et al., 2019), this study was set out to elucidate transcriptional responses that give rise to the salt tolerance of highly-productive HS2. While genetic engineering approaches have been extensively explored with an aim of obtaining robust algal crops, the results clearly indicated that halotolerant HS2 undergoes systematic acclimation responses against high salinity stress, identifying potential target pathways of interest for further genetic modifications or process optimization efforts (Ajjawi et al., 2017; Oh et al., 2019; Qiao, Wasylenko, Zhou, Xu, & Stephanopoulos, 2017). Of these acclimation responses, our results particularly identified a significant role of allocating available carbon

towards the synthesis of algal lipids.

These results support a preferential role of lipid as a carbon and energy reserve under growth-inhibiting stress in HS2. Being an energy dense biomolecule, previous studies indeed identified the role of lipids as a reserve facilitating cellular survival and growth upon the alleviation of growth inhibiting stress conditions (Juergens, Disbrow, & Shachar-Hill, 2016). Similarly, our results indicated the upregulation of enzymes associated with glycolysis and the accumulation of lipid throughout entire growth stages: these results clearly suggest that a "push" of the acetyl-CoA precursor from glycolysis towards lipid synthesis is a major driver of lipid accumulation. Accordingly, the shift in the allocation of storage carbon resulted in an increase in algal lipids and a corresponding decrease in carbohydrates from the harvested biomass. In addition, KEGG pathway analysis of carotenoid synthesis pathway and TCA cycle suggested that these competing pathways for the "pulling" of acetyl-CoA precursor were downregulated, thereby positively contributing to the redirection of acetyl-CoA towards glycerolipid synthesis (Fig. 6).

Recent studies, however, further revealed that lipid droplets are essential and dynamical components of the cellular stress response in terms of maintaining energy and redox homeostasis (Jarc & Petan, 2019), suggesting another important metabolic function of algal lipids besides simple storage reserve. In particular, the accumulation of TAG and/or starch could prevent cellular damage by utilizing excess photosynthetic energy and/or carbon inputs as postulated in the overflow hypothesis (OH) (Juergens, Disbrow, & Shachar-Hill, 2016; Neijssel & Tempest, 1975; Tan & Lee, 2016). Provided that Y(NO) of PSII represents non-regulated losses of excitation energy and thus indirectly indicate the relative amount of reactive oxygen species (ROS) (GmbH, 2012; Klughammer & Schreiber, 2008), our results suggested a strong reduction of PSII accepters and photodamage via formation of ROS during early growth phase under high salinity stress, which seemed to be subsequently resolved at stationary

phase with no substantial compromise in non-photochemical quenching (NPQ). In addition, while our results indicated no differential expression of D1 protein of HS2 under high salinity stress, overall downregulation of protein processing, including subunits of the proteasome, under high salinity stress hints at a decrease in D1 protein turnover in PSII (Andersson & Aro, 2001; Erdmann & Hagemann, 2001), which likely further contributes to the increased oxidative stress due to the inhibition of the recovery of damaged PSII and could elicite cellular remediative responses, including lipid synthesis (Zhang, Paakkarinen, van Wijk, & Aro, 2000).

Importantly, the synthesis of glycerolipid necessitates NADPH as a cofactor (Tan & Lee, 2016): being an electron donor, NADPH is synthesized along with ATP during the light reaction of photosynthesis, and has been acknowledged for its role as an oxidative stress mediator (Valderrama et al., 2006). It should be, however, noted that there was no substantial upregulation of ferredoxin-$NADP^+$ reductase in photosystems based on our transcriptome analysis. Nonetheless, the upregulation of glucose-6-phosphate 1-dehydrogenase and malate dehydrogenase (oxaloacetate-decarboxylating) ($NADP^+$) suggests that these enzymes coupled to central carbon metabolism likely made a substantial contribution to an increased NADPH pool in HS2 under high salinity stress (Spaans et al., 2015). Furthermore, given that KEGG pathway analysis suggested the downregulation of Calvin cycle under high salinity stress, the excess NADPH not utilized in carbon fixation was likely to be also directed to the high accumulation of fatty acid and/or glycerolipid, which in turn could play an important role in remediating excess oxidative stress in PSII (Fig. 6).

In addition to carbon allocation to lipid accumulation, common cellular responses under high salinity stress involve the upregulation of anti-oxidative enzymes, including catalase, superoxide dismutase (SOD), and glutathione reductase (GR) as well as the upregulation of DNA repair mechanisms and ABC transporters (Fu, Wang, Yin, Du, & Kan, 2014; Huang, Fulda, Hagemann, & Norling, 2006; Valderrama et al., 2006). Although

substantial upregulation of anti-oxidative enzymes was not observed at least at transcriptional level, the degree to which each mitigation response contributes to the overall acclimation of HS2 under high salinity stress across different growth stages remains to be elucidated. Importantly, the results also indicated upregulation of P-type Cu^{+} transporter (RAN1) on MAPK signaling pathway in HS2 – the activity of RAN1 was determined to be positively correlated with plant cold resistance; overexpression of RAN1 was further reported to increase abiotic stress tolerance in *Arabidopsis thaliana* (Xu & Cai, 2014; Xu, Zang, Chen, & Cai, 2016; Yang et al., 2018). Moreover, the increased relative proportion of saturated and mono-saturated fatty acids in HS2 under high salinity stress corresponded to the upregulation of enzymes involved in the synthesis of palmitoleate (C16:1), stearate (C18:0), and oleate (C18:1n9c) (Guo, Liu, & Barkla, 2019). Hence, the putative remediation of oxidative stress under growth-inhibiting high salinity condition could concurrently involve signal transduction and a shift in membrane fluidity (Guo et al., 2019), in addition to directing acetyl-CoA precursor and excess co-factor towards lipid synthesis.

While the orchestration of each of elucidated responses likely conferred the relatively high salt tolerance of HS2, lack of some of common algal responses under high salinity stress could offer potential targets along with the identified responses when aiming to further enhance the robustness of HS2 as an industrial algal crop. First, violaxanthin deepoxidase (VDE) and zeaxanthin epoxidase (ZEP) that are respectively involved in the synthesis of zeaxanthin and violaxanthin were not differentially expressed in HS2 under high salinity stress. Zeaxanthin, however, is known to be associated with several types of photoprotection events of the PSII reaction center (Dall'Osto et al., 2012); therefore, VDE upregulation has been acknowledged as one of common algal responses under high oxidative stress (Z. Li et al., 2016). Given that the relative amount of carotenoid pigments in HS2 was increased under high salinity stress (Yun et al., 2019), enhancing the content of zeaxanthin by either upregulating VDE or

downregulating ZEP may further enhance the halotolerance of HS2. Furthermore, although NPQ was not changed under high salinity stress, the elevation of NPQ has been denoted as one of the common algal responses under stress conditions (Cui, Zhang, & Lin, 2017). It would be, therefore, interesting to modulate the NPQ activity of HS2 as part of an effort to confer a greater halotolerance or induce a higher lipid productivity. As an example of the latter, reducing the expression levels of peripheral light-harvesting antenna proteins in PSII was demonstrated to decrease NPQ of *Chlorella vulgaris*, thereby improving biomass productivity by funneling more photosynthetic energy towards the electron transport chain (Shin, Lee, Jeong, Chang, & Kwon, 2016). A similar approach can be adapted to direct more light energy towards the electron transport chain and/or to possibly increase the available NADPH pool, although cautions should be taken to avoid the possibility of antagonistic interactions between competing metabolic pathways.

Acknowledgements

We would like to thank Dr. Carlos P. Roca for his thoughtful comments on SVCD normalization; Sujin Lee and Dr. Saehae Choi for RNA extraction; and Drs. HyunSeok Shin and Byung-Kwan Cho for their suggestions on RNA-seq analysis.

Conflict of Interest

The authors declare no conflict of interests.

Authors' Contributions

JY and HK designed, conceived, coordinated the project and wrote the manuscript; JY, MP and BH performed DEG analysis; DC, D-Y C, YJL, BL, HRK and YKC assisted manuscript preparation and contributed to data interpretation; DC and JH contributed to RNA extraction;

and DC, UK and JH performed supporting experiments. All authors have read and approved the final manuscript.

References

Ajjawi, I., Verruto, J., Aqui, M., Soriaga, L. B., Coppersmith, J., Kwok, K., . . . Xu, W. (2017). Lipid production in Nannochloropsis gaditana is doubled by decreasing expression of a single transcriptional regulator. *Nature biotechnology, 35*(7), 647.

Andersson, B., & Aro, E.-M. (2001). Photodamage and D1 protein turnover in photosystem II. In *Regulation of photosynthesis* (pp. 377-393): Springer.

Bellou, S., & Aggelis, G. (2013). Biochemical activities in Chlorella sp. and Nannochloropsis salina during lipid and sugar synthesis in a lab-scale open pond simulating reactor. *Journal of biotechnology, 164*(2), 318-329.

Bligh, E. G., & Dyer, W. J. (1959). A rapid method of total lipid extraction and purification. *Canadian journal of biochemistry and physiology, 37*(8), 911-917.

Buchfink, B., Xie, C., & Huson, D. H. (2015). Fast and sensitive protein alignment using DIAMOND. *Nature methods, 12*(1), 59.

Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., & Madden, T. L. (2009). BLAST+: architecture and applications. *BMC bioinformatics, 10*(1), 421.

Church, J., Hwang, J.-H., Kim, K.-T., McLean, R., Oh, Y.-K., Nam, B., . . . Lee, W. H. (2017). Effect of salt type and concentration on the growth and lipid content of *Chlorella* vulgaris in synthetic saline wastewater for biofuel production. *Bioresource technology, 243*, 147-153.

Cui, Y., Zhang, H., & Lin, S. (2017). Enhancement of non-photochemical quenching as an adaptive strategy under phosphorus deprivation in the dinoflagellate *Karlodinium veneficum*. *Frontiers in microbiology, 8*, 404.

Dall'Osto, L., Holt, N. E., Kaligotla, S., Fuciman, M., Cazzaniga, S., Carbonera, D., . . . Bassi, R. (2012). Zeaxanthin protects plant photosynthesis by modulating chlorophyll triplet yield in specific light-harvesting antenna subunits. *Journal of biological chemistry, 287*(50), 41820-41834.

Dubois, M., Gilles, K. A., Hamilton, J. K., Rebers, P. t., & Smith, F. (1956). Colorimetric

method for determination of sugars and related substances. *Analytical chemistry, 28*(3), 350-356.

Eckardt, N. A. (2010). The *Chlorella* genome: big surprises from a small package. In: Am Soc Plant Biol.

Eddy, S. R. (2011). Accelerated profile HMM searches. *PLoS computational biology, 7*(10), e1002195.

Erdmann, N., & Hagemann, M. (2001). Salt acclimation of algae and cyanobacteria: a comparison. In *Algal adaptation to environmental stresses* (pp. 323-361): Springer.

Evans, C., Hardin, J., & Stoebel, D. M. (2017). Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions. *Briefings in bioinformatics, 19*(5), 776-792.

Flowers, T., Troke, P., & Yeo, A. (1977). The mechanism of salt tolerance in halophytes. *Annual review of plant physiology, 28*(1), 89-121.

Foflonker, F., Ananyev, G., Qiu, H., Morrison, A., Palenik, B., Dismukes, G. C., & Bhattacharya, D. (2016). The unexpected extremophile: tolerance to fluctuating salinity in the green alga *Picochlorum*. *Algal research, 16*, 465-472.

Fogg, G. (2001). Algal adaptation to stress—some general remarks. In *Algal adaptation to environmental stresses* (pp. 1-19): Springer.

Fu, X., Wang, D., Yin, X., Du, P., & Kan, B. (2014). Time course transcriptome changes in Shewanella algae in response to salt stress. *PloS one, 9*(5), e96001.

GmbH, H. W. (2012). *MULTI-COLOR-PAM Manual*.

Grabherr, M. G., Haas, B. J., Yassour, M., Levin, J. Z., Thompson, D. A., Amit, I., . . . Zeng, Q. (2011). Trinity: reconstructing a full-length transcriptome without a genome from RNA-Seq data. *Nature biotechnology, 29*(7), 644.

Gross, W. (1989). Intracellular localization of enzymes of fatty acid-β-oxidation in the alga Cyanidium caldarium. *Plant physiology, 91*(4), 1476-1480.

Guo, Q., Liu, L., & Barkla, B. J. (2019). Membrane lipid remodeling in response to salinity. *International journal of molecular sciences, 20*(17), 4264.

Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., . . . Lieber, M. (2013). De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature protocols, 8*(8), 1494.

Hu, Q., Sommerfeld, M., Jarvis, E., Ghirardi, M., Posewitz, M., Seibert, M., & Darzins, A. (2008). Microalgal triacylglycerols as feedstocks for biofuel production: perspectives

and advances. *The plant journal, 54*(4), 621-639.

Huang, F., Fulda, S., Hagemann, M., & Norling, B. (2006). Proteomic screening of salt-stress-induced changes in plasma membranes of *Synechocystis* sp. strain PCC 6803. *Proteomics, 6*(3), 910-920.

Illman, A., Scragg, A., & Shales, S. (2000). Increase in *Chlorella* strains calorific values when grown in low nitrogen medium. *Enzyme and microbial technology, 27*(8), 631-635.

Juergens, M. T., Disbrow, B., & Shachar-Hill, Y. (2016). The relationship of triacylglycerol and starch accumulation to carbon and energy flows during nutrient deprivation in *Chlamydomonas reinhardtii*. *Plant physiology, 171*(4), 2445-2457.

Kakarla, R., Choi, J.-W., Yun, J.-H., Kim, B.-H., Heo, J., Lee, S., . . . Kim, H.-S. (2018). Application of high-salinity stress for enhancing the lipid productivity of *Chlorella sorokiniana* HS1 in a two-phase process. *journal of microbiology, 56*(1), 56-64.

Kent, M., Welladsen, H. M., Mangott, A., & Li, Y. (2015). Nutritional evaluation of Australian microalgae as potential human health supplements. *PloS one, 10*(2), e0118985.

Klughammer, C., & Schreiber, U. (2008). Complementary PS II quantum yields calculated from simple fluorescence parameters measured by PAM fluorometry and the Saturation Pulse method. *PAM application notes, 1*(2), 201-247.

Kong, F., Liang, Y., Légeret, B., Beyly-Adriano, A., Blangy, S., Haslam, R. P., . . . Li-Beisson, Y. (2017). *Chlamydomonas* carries out fatty acid β-oxidation in ancestral peroxisomes using a bona fide acyl-CoA oxidase. *The plant journal, 90*(2), 358-371.

Kriventseva, E. V., Kuznetsov, D., Tegenfeldt, F., Manni, M., Dias, R., Simão, F. A., & Zdobnov, E. M. (2018). OrthoDB v10: sampling the diversity of animal, plant, fungal, protist, bacterial and viral genomes for evolutionary and functional annotations of orthologs. *Nucleic acids research, 47*(D1), D807-D811.

Langmead, B., Trapnell, C., Pop, M., & Salzberg, S. (2009). Bowtie: An ultrafast memory-efficient short read aligner. *Genome Biol, 10*(3), R25.

Laurens, L. M., Markham, J., Templeton, D. W., Christensen, E. D., Van Wychen, S., Vadelius, E. W., . . . Pienkos, P. T. (2017). Development of algae biorefinery concepts for biofuels and bioproducts; a perspective on process-compatible products and their impact on cost-reduction. *Energy & Environmental Science, 10*(8), 1716-1738.

Lee, H., Nam, K., Yang, J.-W., Han, J.-I., & Chang, Y. K. (2016). Synergistic interaction between metal ions in the sea salts and the extracellular polymeric substances for efficient microalgal harvesting. *Algal research, 14*, 79-82.

Li, B., & Dewey, C. N. (2011). RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome. *BMC bioinformatics, 12*(1), 323.

Li, Z., Peers, G., Dent, R. M., Bai, Y., Yang, S. Y., Apel, W., . . . Niyogi, K. K. (2016). Evolution of an atypical de-epoxidase for photoprotection in the green lineage. *Nature plants, 2*(10), 16140.

Liu, K.-H., Ding, X.-W., Rao, N., Prabhu, M., Zhang, B., Zhang, Y.-G., . . . Li, W.-J. (2017). Morphological and transcriptomic analysis reveals the osmoadaptive response of endophytic fungus *Aspergillus montevidensis* ZYD4 to high salt stress. *Frontiers in microbiology, 8*, 1789.

Lowry, O. H., Rosebrough, N. J., Farr, A. L., & Randall, R. J. (1951). Protein measurement with the Folin phenol reagent. *Journal of biological chemistry, 193*, 265-275.

McBride, R. C., Lopez, S., Meenach, C., Burnett, M., Lee, P. A., Nohilly, F., & Behnke, C. (2014). Contamination management in low cost open algae ponds for biofuels production. *Industrial Biotechnology, 10*(3), 221-227.

Medina, M., & Gómez-Moreno, C. (2004). Interaction of ferredoxin–$NADP^+$ reductase with its substrates: Optimal interaction for efficient electron transfer. *Photosynthesis research, 79*(2), 113-131.

Mootha, V. K., Lindgren, C. M., Eriksson, K.-F., Subramanian, A., Sihag, S., Lehar, J., . . . Laurila, E. (2003). PGC-1α-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. *Nature genetics, 34*(3), 267.

Neijssel, O., & Tempest, D. (1975). The regulation of carbohydrate metabolism in *Klebsiella aerogenes* NCTC 418 organisms, growing in chemostat culture. *Archives of Microbiology, 106*(3), 251-258.

Oh, S. H., Chang, Y. K., & Lee, J. H. (2019). Identification of significant proxy variable for the physiological status affecting salt stress-induced lipid accumulation in *Chlorella sorokiniana* HS1. *Biotechnology for Biofuels, 12*(1), 242.

Peng, Z., He, S., Gong, W., Sun, J., Pan, Z., Xu, F., . . . Du, X. (2014). Comprehensive analysis of differentially expressed genes and transcriptional regulation induced by salt stress in two contrasting cotton genotypes. *BMC genomics, 15*(1), 760.

Perrineau, M. M., Zelzion, E., Gross, J., Price, D. C., Boyd, J., & Bhattacharya, D. (2014). Evolution of salt tolerance in a laboratory reared population of *Chlamydomonas reinhardtii*. *Environmental microbiology, 16*(6), 1755-1766.

Qiao, K., Wasylenko, T. M., Zhou, K., Xu, P., & Stephanopoulos, G. (2017). Lipid production

in *Yarrowia lipolytica* is maximized by engineering cytosolic redox metabolism. *Nature biotechnology, 35*(2), 173.

Quinn, J. C., & Davis, R. (2015). The potentials and challenges of algae based biofuels: a review of the techno-economic, life cycle, and resource assessment modeling. *Bioresource technology, 184*, 444-452.

Rathinasabapathi, B. (2000). Metabolic engineering for stress tolerance: installing osmoprotectant synthesis pathways. *Annals of Botany, 86*(4), 709-716.

Roca, C. P., Gomes, S. I., Amorim, M. J., & Scott-Fordsmand, J. J. (2017). Variation-preserving normalization unveils blind spots in gene expression profiling. *Scientific reports, 7*, 42460.

Shin, W.-S., Lee, B., Jeong, B.-r., Chang, Y. K., & Kwon, J.-H. (2016). Truncated light-harvesting chlorophyll antenna size in *Chlorella vulgaris* improves biomass productivity. *Journal of Applied Phycology, 28*(6), 3193-3202.

Shin, W.-S., Lee, B., Kang, N. K., Kim, Y.-U., Jeong, W.-J., Kwon, J.-H., . . . Chang, Y. K. (2017). Complementation of a mutation in CpSRP43 causing partial truncation of light-harvesting chlorophyll antenna in *Chlorella vulgaris*. *Scientific reports, 7*(1), 17929.

Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V., & Zdobnov, E. M. (2015). BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics, 31*(19), 3210-3212.

Smith, V. H., Sturm, B. S., Denoyelles, F. J., & Billings, S. A. (2010). The ecology of algal biodiesel production. *Trends in ecology & evolution, 25*(5), 301-309.

Spaans, S. K., Weusthuis, R. A., Van Der Oost, J., & Kengen, S. W. (2015). NADPH-generating systems in bacteria and archaea. *Frontiers in microbiology, 6*, 742.

Stephens, E., Ross, I. L., King, Z., Mussgnug, J. H., Kruse, O., Posten, C., . . . Hankamer, B. (2010). An economic and technical evaluation of microalgal biofuels. *Nature biotechnology, 28*(2), 126.

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., . . . Lander, E. S. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences, 102*(43), 15545-15550.

Talebi, A. F., Tabatabaei, M., Mohtashami, S. K., Tohidfar, M., & Moradi, F. (2013). Comparative salt stress study on intracellular ion concentration in marine and salt-adapted freshwater strains of microalgae. *Notulae Scientia Biologicae, 5*(3), 309-315.

Tan, K. W. M., & Lee, Y. K. (2016). The dilemma for lipid productivity in green microalgae: importance of substrate provision in improving oil yield without sacrificing growth. *Biotechnology for Biofuels, 9*(1), 255.

Unkefer, C. J., Sayre, R. T., Magnuson, J. K., Anderson, D. B., Baxter, I., Blaby, I. K., . . . Dale, T. (2017). Review of the algal biology program within the National Alliance for Advanced Biofuels and Bioproducts. *Algal research, 22*, 187-215.

Valderrama, R., Corpas, F. J., Carreras, A., GÓMEZ-RODRÍGUEZ, M. V., Chaki, M., Pedrajas, J. R., . . . Barroso, J. B. (2006). The dehydrogenase-mediated recycling of NADPH is a key antioxidant system against salt-induced oxidative stress in olive plants. *Plant, Cell & Environment, 29*(7), 1449-1459.

Valizadeh Derakhshan, M., Nasernejad, B., Abbaspour-Aghdam, F., & Hamidi, M. (2015). Oil extraction from algae: A comparative approach. *Biotechnology and applied biochemistry, 62*(3), 375-382.

von Alvensleben, N., Stookey, K., Magnusson, M., & Heimann, K. (2013). Salinity tolerance of *Picochlorum atomus* and the use of salinity for contamination control by the freshwater cyanobacterium *Pseudanabaena limnetica*. *PloS one, 8*(5), e63569.

Wang, L., Yuan, D., Li, Y., Ma, M., Hu, Q., & Gong, Y. (2016). Contaminating microzooplankton in outdoor microalgal mass culture systems: An ecological viewpoint. *Algal research, 20*, 258-266.

Xu, P., & Cai, W. (2014). RAN1 is involved in plant cold resistance and development in rice (*Oryza sativa*). *Journal of experimental botany, 65*(12), 3277-3287.

Xu, P., Zang, A., Chen, H., & Cai, W. (2016). The small G protein AtRAN1 regulates vegetative growth and stress tolerance in *Arabidopsis thaliana*. *PloS one, 11*(6), e0154787.

Yang, A., Suh, W. I., Kang, N. K., Lee, B., & Chang, Y. K. (2018). MAPK/ERK and JNK pathways regulate lipid synthesis and cell growth of *Chlamydomonas reinhardtii* under osmotic stress, respectively. *Scientific reports, 8*(1), 13857.

Yuge Zhang, Y. J., & Liang, W. (2006). Accumulation of soil soluble salt in vegetable greenhouses under heavy application of fertilizers. *Agr J, 1*, 123-127.

Yun, J.-H., Cho, D.-H., Heo, J., Lee, Y. J., Lee, B., Chang, Y. K., & Kim, H.-S. (2019). Evaluation of the potential of *Chlorella* sp. HS2, an algal isolate from a tidal rock pool, as an industrial algal crop under a wide range of abiotic conditions. *Journal of Applied Phycology*, 1-14.

Yun, J.-H., Cho, D.-H., Lee, B., Kim, H.-S., & Chang, Y. K. (2018). Application of

biosurfactant from *Bacillus subtilis* C9 for controlling cladoceran grazers in algal cultivation systems. *Scientific reports, 8*(1), 5365.

Yun, J.-H., Cho, D.-H., Lee, S., Heo, J., Tran, Q.-G., Chang, Y. K., & Kim, H.-S. (2018). Hybrid operation of photobioreactor and wastewater-fed open raceway ponds enhances the dominance of target algal species and algal biomass production. *Algal research, 29*, 319-329.

Yun, J.-H., Smith, V. H., La, H.-J., & Keun Chang, Y. (2016). Towards managing food-web structure and algal crop diversity in industrial-scale algal biomass production. *Current Biotechnology, 5*(2), 118-129.

Yun, J.-H., Smith, V. H., & Pate, R. C. (2015). Managing nutrients and system operations for biofuel production from freshwater macroalgae. *Algal research, 11*, 13-21.

Zhang, L., Paakkarinen, V., van Wijk, K. J., & Aro, E.-M. (2000). Biogenesis of the chloroplast-encoded D1 protein: regulation of translation elongation, insertion, and assembly into photosystem II. *The Plant Cell, 12*(9), 1769-1781.

Zhu, L., Wang, Z., Shu, Q., Takala, J., Hiltunen, E., Feng, P., & Yuan, Z. (2013). Nutrient removal and biodiesel production by integration of freshwater algae cultivation with piggery wastewater treatment. *Water research, 47*(13), 4294-4302.

Table 1. Up- and down-regulated genes within KEGG pathway at exponential and stationary growth phases.

KEGG pathway	Total	Upregulation (downregulation) at exponential phase	Upregulation (downregulation) at stationary phase
ABC transporters	16	5 (11)	5 (11)
Aminoacyl-tRNA biosynthesis	43	5 (38)	3 (40)
Base excision repair	15	7 (8)	2 (13)
Biosynthesis of unsaturated fatty acids	8	2 (6)	2 (6)
Carbon fixation in photosynthetic organisms	21	5 (16)	7 (14)
Carotenoid biosynthesis	9	3 (6)	1 (8)
Cell cycle	26	8 (18)	2 (24)
Citrate cycle (TCA cycle)	37	8 (29)	8 (29)
DNA replication	23	8 (15)	2 (21)
Fatty acid biosynthesis	15	5 (10)	4 (11)
Fatty acid degradation	16	4 (12)	1 (15)
Glycerolipid metabolism	13	9 (4)	2 (11)
Glycolysis / Gluconeogenesis	38	13 (25)	12 (26)
MAPK signaling pathway	25	4 (21)	0 (25)
Mismatch repair	11	5 (6)	0 (11)
Nucleotide excision repair	25	13 (12)	4 (21)
Photosynthesis	17	1 (16)	8 (9)
Photosynthesis - antenna proteins	12	0 (12)	7 (5)
Proteasome	32	14 (18)	6 (26)
Protein export	16	5 (16)	1 (15)
Protein processing in endoplasmic reticulum	72	20 (52)	0 (72)
Ribosome	123	3 (120)	2 (121)
RNA degradation	35	14 (21)	5 (30)
RNA transport	62	6 (56)	2 (60)

Figures

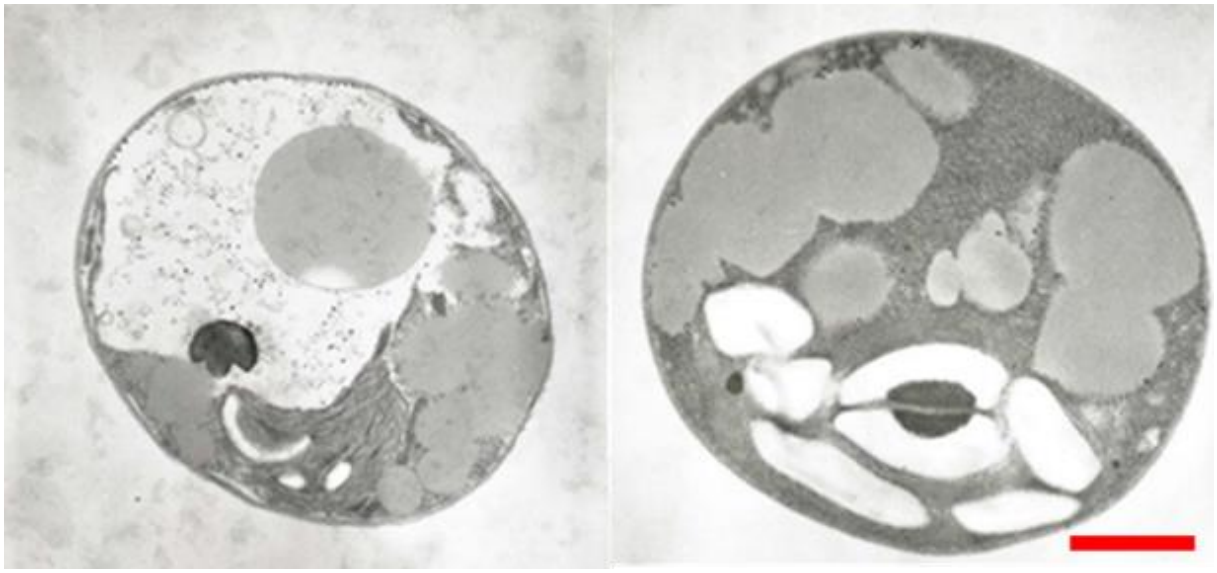

Fig 1. Electron micrographs of Chlorella sp. HS2 grown in freshwater (left) and marine (right) growth media at stationary growth phase. Scale bar denotes 1 µm.

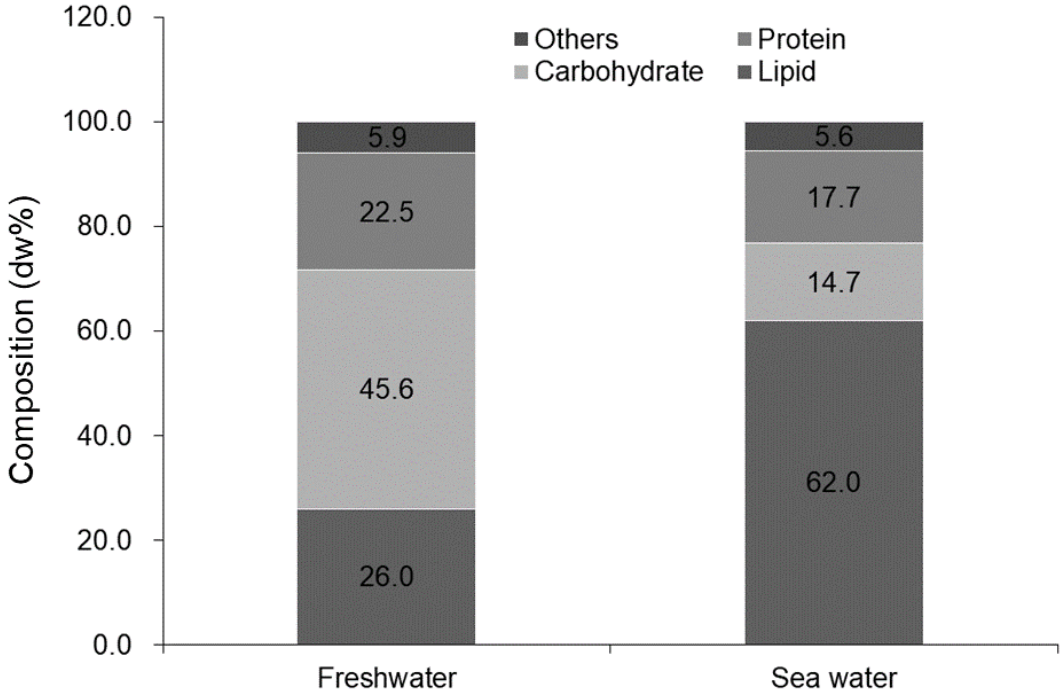

Fig 2. Proximate composition of Chlorella sp. HS2 grown in freshwater and marine growth media. Biomass harvested at stationary growth phase (n=3) was used in analysis.

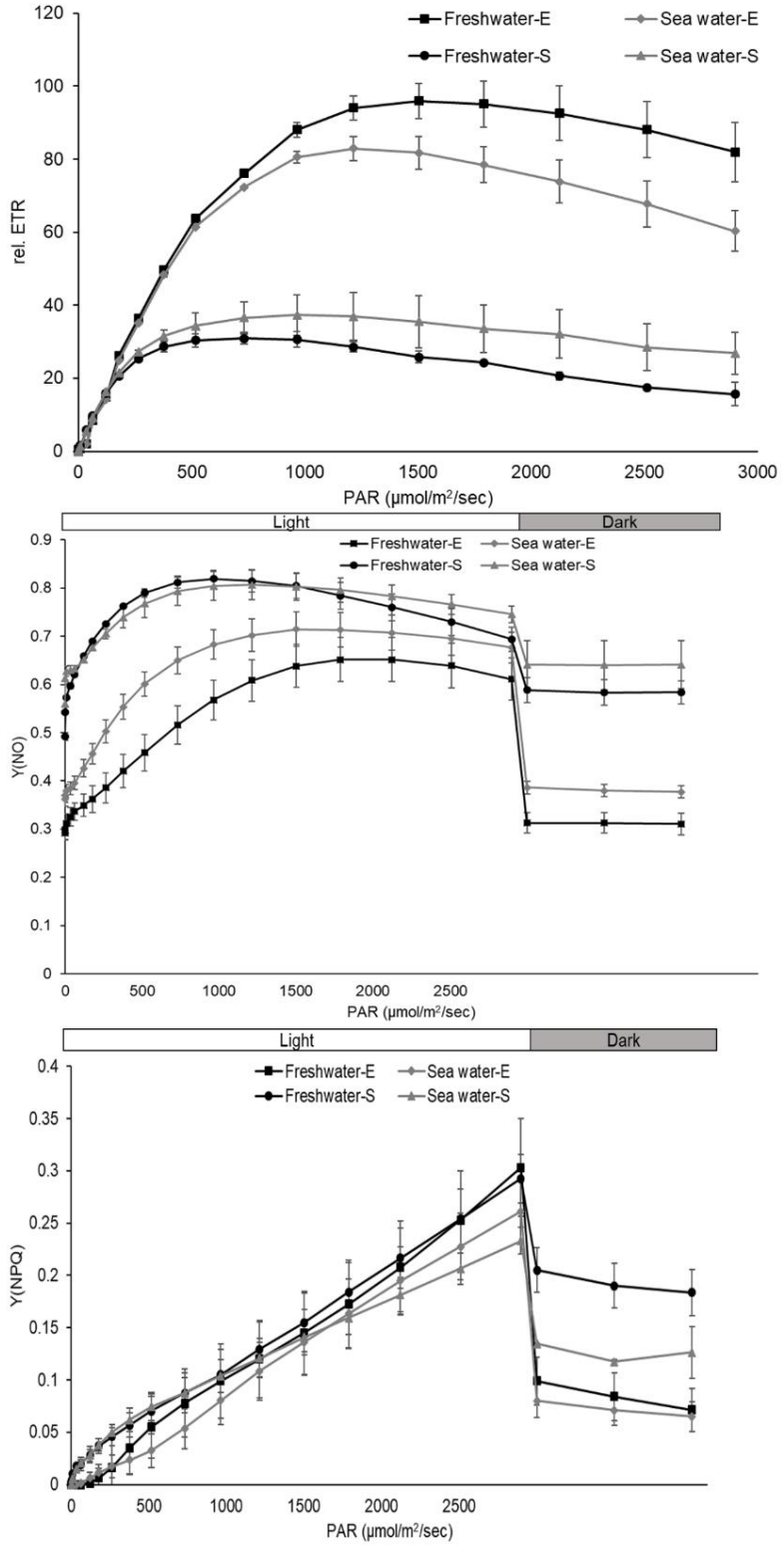

Fig 3. Measurements of parameters related to the photosynthetic activity of Chlorella sp. HS2 in freshwater and marine conditions at exponential and stationary growth phases. (a) Relative Electron Transport Rate in PSII. (b) Quantum yield of non-photochemical quenching in PSII. (c) Quantum yield of non-regulated non-photochemical energy loss in PSII. Error bars denote standard error of the mean from triplicate culture samples. E and S respectively denote exponential and stationary growth phases.

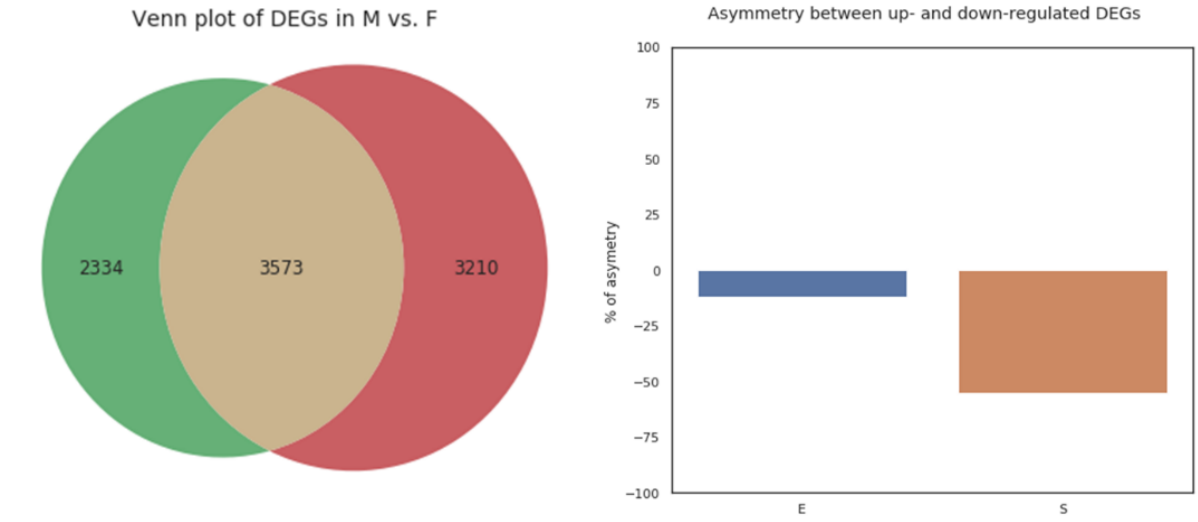

Fig 4. Analysis of DEGs. (A) Venn diagram of the DEGs with an adjusted p-value cutoff of 0.01 in marine (M) and freshwater (F) conditions. (B) Asymmetry between the numbers of up- and downregulated DEGs in exponential (E) and stationary (S) growth phases. Note that negative % asymmetry indicates more DEGs were downregulated generally.

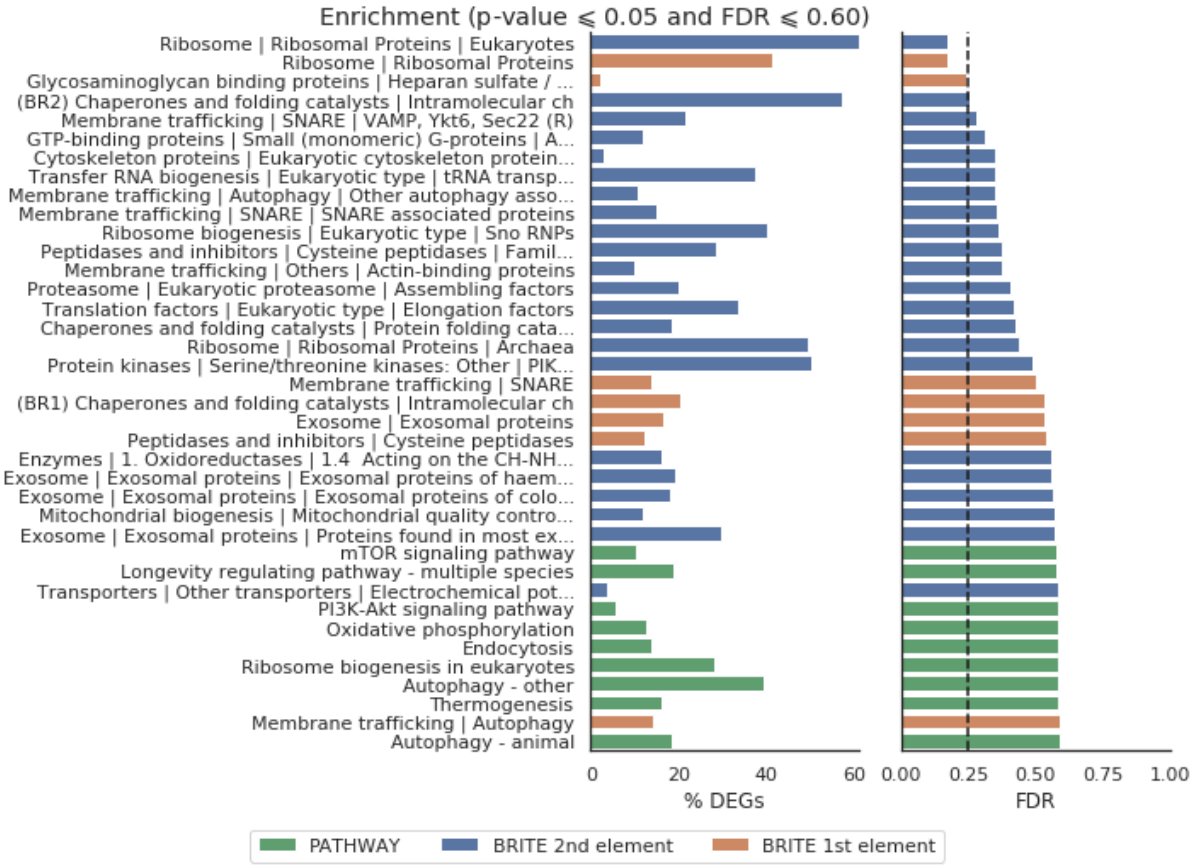

Fig 5. Functional enrichment of DEGs with KEGG pathway and BRITE databases

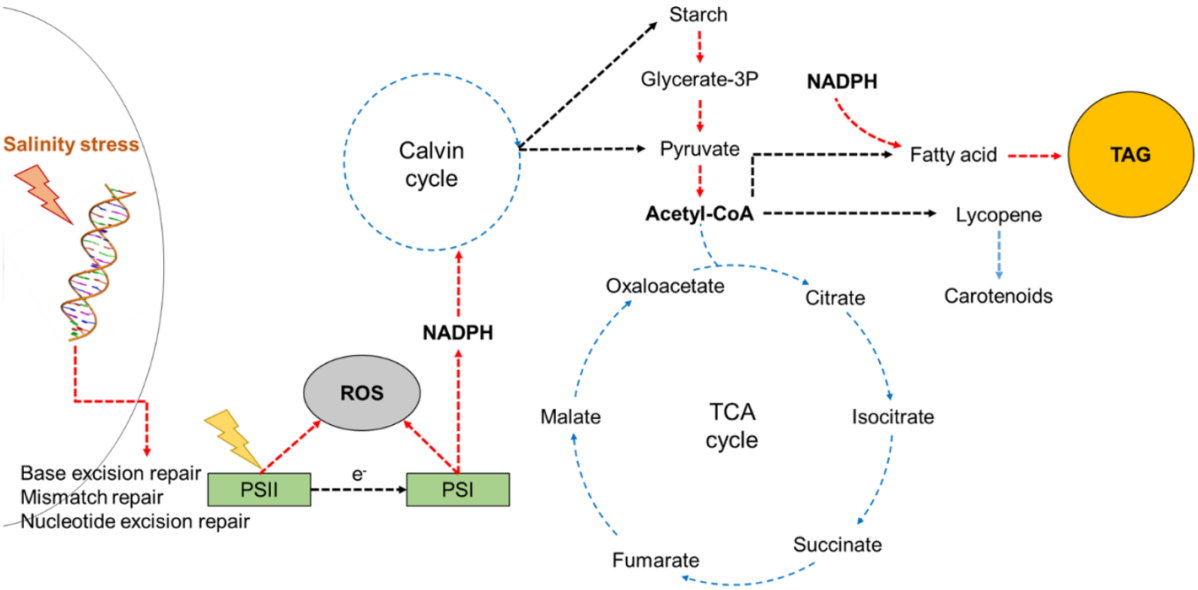

Fig 6. Simplified scheme of carbon and energy flows in Chlorella sp. HS2 for putative early responses against high salinity stress. Red and blue dashed arrows respectively indicate upregulation and downregulation of a given conversion or response based on transcriptome or phenotypic analyses. Glycerate-3p Glycerate-3-phosphate; NADPH Nicotinamide adenine dinucleotide phosphate; ROS Reactive Oxygen Species; TAG Triacylglycerol

Chapter 2 – TimeNexus paper

TimeNexus: a novel Cytoscape app to analyze time-series data using temporal MultiLayer Networks (tMLNs)

Michaël Pierrelée [1], Ana Reynders [2], Fabrice Lopez [3], Aziz Moqrich [2], Laurent Tichit [4] and Bianca H. Habermann [1]

1 Aix-Marseille University, CNRS, IBDM UMR 7288, Turing Centre for Living Systems (CENTURI), Computational Biology Team, Marseille, France

2 Aix-Marseille University, CNRS, IBDM UMR 7288, Turing Centre for Living Systems (CENTURI), Team Chronic Pain: Molecular and Cellular Mechanisms, Marseille, France

3 Aix-Marseille University, INSERM, TAGC U 1090, Marseille, France

4 Aix-Marseille University, CNRS, I2M UMR 7373, Turing Centre for Living Systems (CENTURI), Marseille, France

Corresponding Author :
Bianca H. Habermann
Aix-Marseille University, CNRS, IBDM UMR7288
Turing Center for Living Systems (CENTURI)
Parc Scientifique de Luminy, Case 907
163, Avenue de Luminy,
13009 Marseille
France
e-mail: bianca.habermann@univ-amu.fr

Abstract

Integrating -omics data with biological networks such as protein-protein interaction networks is a popular and useful approach to interpret expression changes of genes in changing conditions, and to identify relevant cellular pathways, active subnetworks or network communities. Yet, most -omics data integration tools are restricted to static networks and therefore cannot easily be used for analyzing time-series data.

Determining regulations or exploring the network structure over time requires time-dependent networks which incorporate time as one component in their structure. Here, we present a method to project time-series data on sequential layers of a multilayer network, thus creating a *temporal multilayer network* (tMLN). We implemented this method as a Cytoscape app we named TimeNexus. TimeNexus allows to easily create, manage and visualize temporal multilayer networks starting from a combination of node and edge tables carrying the information on the temporal network structure. To allow further analysis of the tMLN, TimeNexus creates and passes on regular Cytoscape networks in form of static versions of the tMLN in three different ways: i) over the entire set of layers, ii) over two consecutive layers at a time, iii) or on one single layer at a time. We combined TimeNexus with the Cytoscape apps PathLinker and AnatApp/ANAT to extract active subnetworks from tMLNs. To test the usability of our app, we applied TimeNexus together with PathLinker or ANAT on temporal expression data of the yeast cell cycle and were able to identify active subnetworks relevant for different cell cycle phases. We furthermore used TimeNexus on our own temporal expression data from a mouse pain assay inducing hindpaw inflammation and detected active subnetworks relevant for an inflammatory response to injury, including immune response, cell stress response and regulation of apoptosis. TimeNexus is freely available from the Cytoscape app store at https://apps.cytoscape.org/apps/TimeNexus.

Introduction

Time-series gene or protein expression data can give invaluable insight into the temporal dynamics of biological processes. It informs about the changes in activity of molecular pathways and key players upon a cellular stimulus or helps characterize molecular activity in cyclic processes, such as the cell cycle or the circadian rhythm. Methods and protocols exist to analyze time-series expression data and extract the dynamically expressed genes from a temporal dataset, some of which have been reviewed and compared in [1]. Results from such tools however do not provide insights into the activity of key molecules or pathways at a given time point. Clustering temporal expression of genes is another possibility to analyze time-series data [2]. Here, especially the clustering of expression profiles over time points is useful to follow the trajectory of expression dynamics of genes over time and to identify co-regulated gene groups [3-5] .

Integrating temporal expression data with protein interaction data is more challenging. Generally, the integration of -omics data with interactomes is very useful to gain deeper insight, like identifying dysregulated pathways or gene communities of interest [6-9]. Popular approaches in network analysis combined with expression data include community detection, identification of active subnetworks or of changes in general network features such as centrality measures [10-21].

However, most approaches in this type of data integration are limited to static interactomes even though the necessity of dynamic interactomes was recognized some time ago [22]. A dynamic interactome can be modeled as a temporal network. In brief, a temporal network can be described as a sequence of static networks states ordered in time, whereby each state represents the activity of the network at a given time point. Temporal networks and their usability in different scientific disciplines have been reviewed in [23,24]. In principle, the same network analysis techniques used for static networks can be applied to temporal networks, for instance extracting active subnetworks or detecting communities, identifying important nodes by centrality measures, etc. [25-27]. Yet, it should be considered that some of the standard assumptions applied to static networks are not transferable to temporal networks and so additional tools for temporal network analysis will be required [26].

Some approaches have been introduced that enable users to analyze temporal gene expression data by integrating them with an interactome in a dynamic manner. TimeXnet is a stand-alone JAVA application to identify active subnetworks in interactomes based on

time-course expression data [28]. TimeXNet assumes cellular responses to be divided into early, middle and late phases. It takes as an input a weighted interactome together with three gene lists representing the active subset of genes at the three given phases (early, middle, late). It will return the predicted active subnetwork together with the flow between the nodes (active genes) in the early, middle and late phases. The output can be directly visualized in Cytoscape in form of a network. While TimeXNet has shown promising results in mouse innate immune response [29], it allows only three phases, where each gene belongs to exactly one phase that needs to be defined *a priori* by the user. Thereby, TimeXNet cannot manage more complex dynamic systems. The Cytoscape app DyNet allows to visualize and analyze dynamic molecular interaction networks [30]. It offers interactive visualization of a temporal network as sets of state graphs, allowing re-arranging of the nodes on each state simultaneously. Moreover, network analysis functions are provided, such as comparing attributes (of nodes or edges) over two or more layers or identification of the most dynamic neighborhood by searching for the most 'rewired' nodes in the temporal network. The Cytoscape app DyNetViewer [31] is able to construct, analyze and visualize active temporal networks. It provides four different algorithms for constructing one static active subnetwork for each time point by retaining only the active nodes from a large protein interaction network at that time point. It provides in addition network analysis functions, mostly focusing on centrality measures and graph clustering algorithms of the temporal network. Furthermore, DyNetViewer enables the user to analyze and visualize the resulting active subnetwork. However, its functions are limited to handling one single layer at a time. Therefore, it does not fully apply the principles of temporal networks.

What is generally missing is an easy to use and flexible app for working with temporal data in network analysis. With TimeNexus, we introduce an approach which models a temporal network as a discrete time longitudinal network, in which the expression changes over time are projected on the layers of a multilayer network. Expression changes of one time point are projected on one layer in the form of node weights and the layers are ordered in a time-dependent manner. Other than available methods, TimeNexus uses the edges connecting the layers (*inter-layer edges*) to model transition states between nodes from one time point to the next and thus takes full advantage of the time-series data. *A priori*, all layers contain the same network (the same nodes and edges) and thus, the multilayer network initially generated by TimeNexus is a multiplex network. TimeNexus multilayer networks are not

temporal networks in the sense of [23], which assumes that edge activity varies over time. To avoid ambiguity, we refer to our networks as *temporal multilayer networks* (tMLNs, Figure 1). TimeNexus can be used to generate, manage and visualize tMLNs.

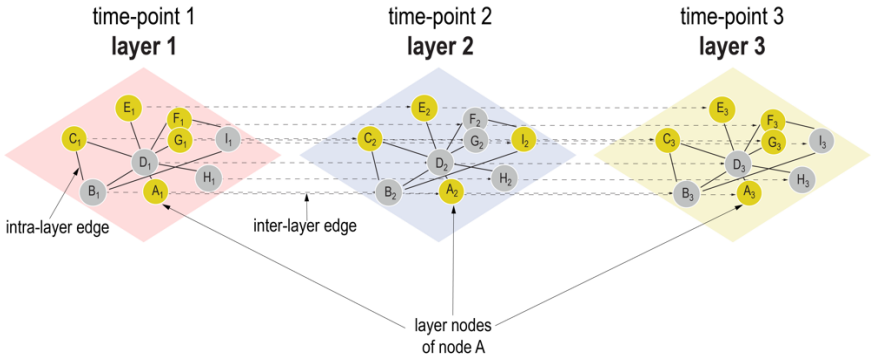

Figure 1: Basic structure of a temporal multilayer network (tMLN). Here shown is a tMLN of three layers. Each layer of the network contains the same protein-protein interaction network (PPIN). Nodes within one layer (*layer-nodes*) are connected via *intra-layer edges*, the same node between two layers is connected by an *inter-layer edge*. For example, the layer-nodes from a given node A (A_1, A_2, A_3) are successively linked by inter-layer edges (A_1->A_2->A_3). Numerical data, such as differential expression data from a time-series RNA-sequencing study, are integrated with the TimeNexus tMLN, whereby one layer represents one time point. Yellow nodes represent *query nodes*, which need to be defined a priori by the user. Query nodes can for instance be chosen based on significant differential expression of genes at a given time point versus a control. Grey nodes connecting query nodes but being themselves not significantly differentially expressed are referred to as *Steiner nodes* in extracted active subnetworks.

We wanted to use TimeNexus to extract active subnetworks from time-series data. Therefore, in the current release of TimeNexus, we provide a connection to the Cytoscape apps PathLinker [16,17] and AnatApp / the ANAT server [18-21] for active subnetwork extraction based on differential expression data, making use of their respective programmatic interfaces. PathLinker finds a user-defined number *(K)* of shortest paths between source and target nodes in a network and then creates active subnetworks by unifying these paths. It requires user-defined query nodes (source and target), and makes use of edge weights within the network to calculate scores for each path between query nodes. To identify shortest paths in large networks, it has implemented an A* heuristic version of Yen's algorithm [32]. ANAT, on the other hand, identifies 'functional networks' from a large cellular interactome. When extracting 'anchored' networks, ANAT connects a set of target proteins (nodes that were for instance identified in a large-scale screen) with 'anchor' proteins (nodes around

which the network should be constructed), and by making use of edge weights. ANAT tries to find the most probable connecting path between two nodes based on minimizing the sum of weights of all edges in the extracted active subnetwork, which is known as the Steiner tree problem [33]. As the two end-points need to be connected, the algorithm tends to include non-query nodes in the final active subnetwork, which are known as *Steiner nodes* (or Steiner points). Theoretically, TimeNexus can be extended with any network analysis app available within Cytoscape, provided that it possesses a programmatic interface, as do PathLinker and ANAT.

To test TimeNexus, we used our app together with PathLinker and ANAT on a yeast cell cycle temporal study, following gene expression dynamics of the yeast cell division cycle in synchronized cells [34]. We extracted active subnetworks of the cell cycle from a temporal multilayer network comprised of 16 temporal layers of one full cycle. We scored these active subnetworks for relevance to the process under study by looking for enriched GO-terms related to cell cycle. We also applied TimeNexus to our own data from an injury induced pain assay in mouse, following mechanosensitivity and associated transcriptional changes over 30 days. We predicted pathways relevant for this process, including immune response, stress response, apoptosis regulation and axonal growth. Although TimeNexus has been optimized for temporal and multiplex networks, it is also applicable to all other forms of multilayer networks. TimeNexus is freely available from the Cytoscape App store (https://apps.cytoscape.org/apps/TimeNexus). The source code is also available on GitLab (https://gitlab.com/habermann_lab/temporal-network-project).

Methods

Definitions

We project temporal differential gene expression data from a time-series on a multilayer network structure in the Cytoscape app TimeNexus, whereby we assign the differential expression data from each time point to the layer representing this time point in the form of node weights. We refer to this network model as a *temporal multilayer network* (tMLN, Figure 1). *A priori*, the network is the same on all layers. Therefore, the tMLN created by TimeNexus is a multiplex network. We refer to a node on an individual layer as a *layer-node*, as opposed to a node of a single-layer static network. Edges connecting nodes within one layer (A_1, B_1, C_1) are termed *intra-layer edges*; those connecting the same node between two different layers (A_1 and A_2) are called *inter-layer edges*. Weights can be added to intra- and inter-layer edges. Intra-layer edge weights will most of the time represent confidence scores on a specific interaction. Inter-layer edge scores on the other hand can contain information on changes in differential expression of one gene from one time point to the next. Thus, they represent transition weights from one layer to the next and can be used for subsequent network analysis tasks, such as active subnetwork extraction. For follow-up analysis of the tMLN, we furthermore need to define *query nodes*: a query node is a layer-node that shows significant differential expression at the given time point that is associated with that specific layer.

TimeNexus represents the tMLN by two simplified objects: the *Flattened network* and the *Aggregated network*. These two networks are complementary. Thus, in Cytoscape, a TimeNexus tMLN is represented by a network collection, which includes the Flattened and the *Aggregated network*, as well as a static network for each layer, representing the snapshot of differential gene expression at a given time. The *Flattened network* is the visual representation of the tMLN and serves for most applications, such as processing the tMLN by static network tools. In the *Flattened network*, layer-nodes become independent entities and the intra- and inter-layer edges become indistinguishable. Therefore, the Cytoscape 'create view' feature will not display this object properly as a temporal succession of layers and a dedicated viewer app is required; The *Aggregated network* represents the collapsed, single-layer network of all layers: all layer-nodes and intra-layer edges are unified in a single node and edge, respectively and all temporal information is lost.

Temporal information required for building a temporal multilayer network with TimeNexus

TimeNexus builds a tMLN by converting tables into a collection of Cytoscape networks. The conversion requires 2 types of tables: a *node table* containing attributes for each of the layer-nodes and an *intra-layer edge table* connecting the layer-nodes (Figure 2: **1. data import**). Optionally, an *inter-layer edge table* can be provided which specifies user-defined weight information for connecting the layers.

The node table must contain information on the nodes in form of gene or protein names. It must also contain the information whether a layer-node is a query node or not. The query attribute is important as it is used by the active subnetwork extracting apps to identify the layer-nodes that will contribute to the extracted active subnetwork. The query node attribute can for instance be defined based on the log2 fold change of the layer-node surpassing a selected cut-off and must either be *TRUE* or *FALSE*. Query nodes on each of the layers are thus pre-set by the user. Additional layer-specific attributes such as the weight for each layer-node in form of a numerical value can be provided, for instance reflecting the differential expression at each time point. The interactome of a tMLN is assumed to be the same at each layer. It should however be noted that TimeNexus can also handle multilayer networks that are not multiplex. In this case, the user has to provide one node table for each layer.

The intra-layer edge table contains the information to build the interactome, which is common to each layer. This table contains the edge information of the interacting nodes (proteins or genes). A weight can be given to each intra-layer edge, for instance in form of a confidence score for the interaction. The type of interaction (protein-protein interaction (PPI) or protein-DNA interaction (PDI)) can be distinguished by adding an optional attribute to each edge.

The optional inter-layer edge table is equivalent to the intra-layer edge table, but it defines the edges connecting the nodes from one layer to the next. In our example of a tMLN, the inter-layer edges connect the same layer-nodes from two different , consecutive layers. Their attributes represent weights calculated by combining the weights of the source and target nodes and thus carry information on the change in expression of that node between two time points. If the inter-layer edge table is not provided, TimeNexus will automatically create these inter-layer edge weights (see below). See Supplementary Tables S1-S3 for examples

for the node table and the intra- and inter-layer edge tables. To create a tMLN, at least 2 layers are required.

Connecting layers in TimeNexus

The layers are connected through the inter-layer edges. If the user does not provide an inter-layer edge table, the weight between a layer-node on a given layer and its counterpart on the next layer will be computed as

$$w_{\text{inter-layer edge}} = (w\_i+w\_j)/(1+w\_i+w\_j)$$

where w_i is the weight of the layer-node from the layer i and w_j the layer-node weight on the layer j=i+1. Contrary to the intra-layer edges, the inter-layer edges are directed for active subnetwork extraction with ANAT. For PathLinker, inter-layer edge directionality is removed, as this app cannot handle mixed edge types.

Building, managing and visualizing tMLNs with the Cytoscape app TimeNexus

We created the Cytoscape app TimeNexus to build, manage and visualize tMLNs and to prepare them for extracting active subnetworks (defined as the region of the interactome that connects the differentially expressed nodes over time [12] (Figure 2, see also Supplementary Figure S1)). TimeNexus was entirely implemented in Cytoscape 3.8.0 [35] and using Java 11. It is incompatible with earlier versions of Cytoscape.

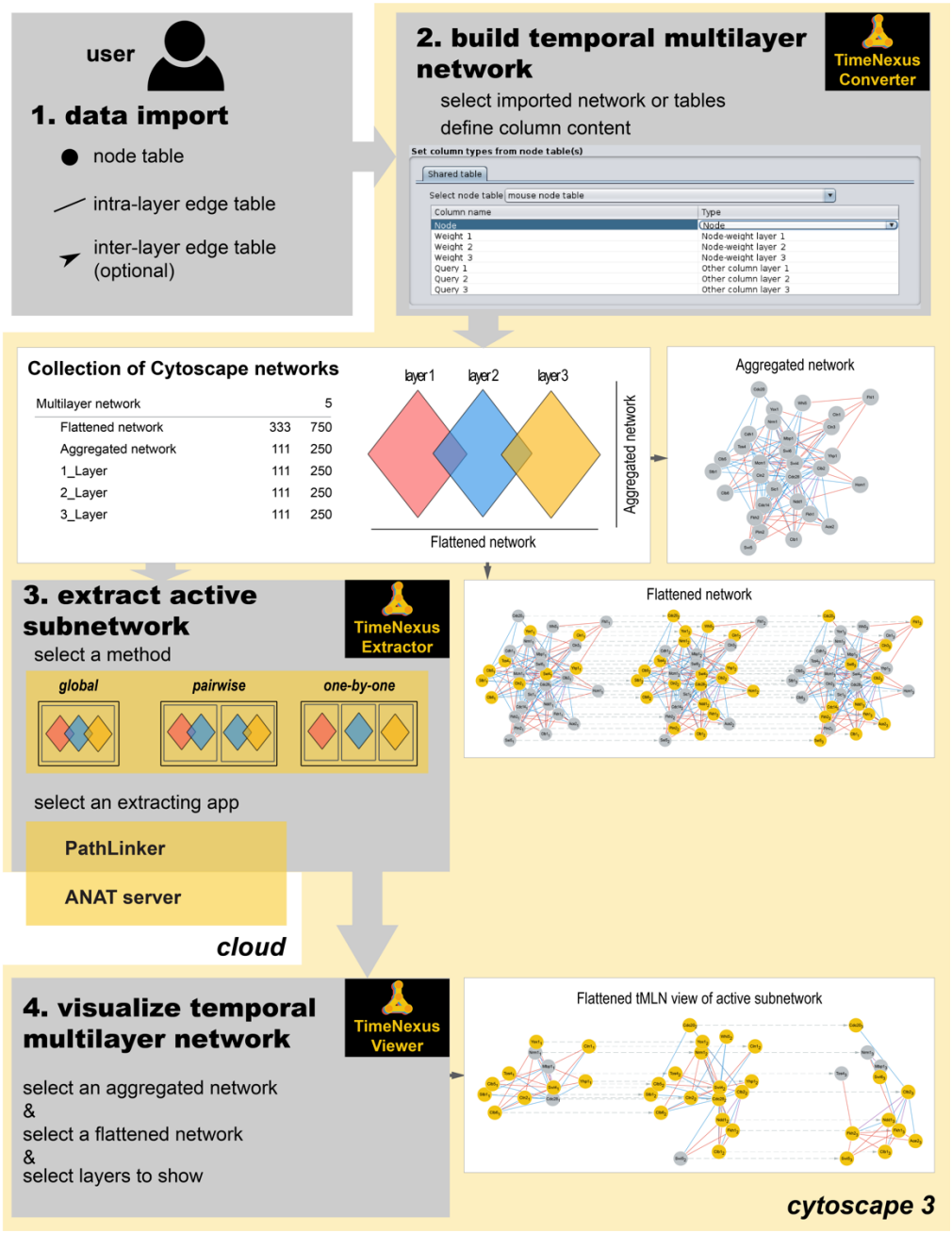

Figure 2: Workflow of the Cytoscape app TimeNexus for creating, managing and analyzing tMLNs. 1. data import: First, the elements (layer-nodes, intra-, and inter-layer edges) structuring the temporal multilayer network (tMLN) have to be imported into Cytoscape in the form of tables. **2. build temporal multilayer network**: In the second step, TimeNexus converts these data into a tMLN. For each element and for each layer, the user selects the appropriate table and specifies the attribute type of each column. Once this is done, TimeNexus represents the tMLN as a collection of Cytoscape networks (center box). It contains a *Flattened network*, an *Aggregated network* and *Layer-specific networks*. In the *Flattened network* view, each layer-node, together with the intra- and inter-layer edges are shown. In the *Aggregated network* view, the layers are collapsed into a single-layer network. **3. extract active subnetwork**: In the next step, an active subnetwork is extracted from the tMLN. First, the user has to choose the method used to extract active subnetworks. TimeNexus offers three methods: method 1 (*global*): the entire *Flattened network* is used at once, without taking into account the edge type (intra- or inter-layer edges are treated as identical); method 2 (*pairwise*): two successive layers are used to extract the active subnetwork that are then combined to the final active subnetwork; method 3 (*one-by-one*): active subnetworks are extracted in each individual layer and these are

combined to the final active subnetwork. For extraction of active subnetworks, TimeNexus offers two algorithms, PathLinker and the ANAT server. PathLinker is a Cytoscape app, while ANAT is executed on the cloud and thus requires a working internet connection. **4. visualize temporal multilayer network**: Finally, to visualize the tMLN or active subnetwork, TimeNexus creates a view of the *Flattened network*. To do so, it takes the node locations from the *Aggregated network* and transmits it on each layer. Layers are ordered in time on the X-axis from left to right.

Building the temporal multilayer network (tMLN).

TimeNexus can build a tMLN from scratch by converting tables describing the network structure, or by converting a single-layer network into a tMLN by adding a table with temporal node information. To build a tMLN from scratch, TimeNexus requires at least one node table together with one intra-layer edge table, as well as an optional inter-layer edge table (Figure 2: **1. data import**). After importing and specifying the content of the tables' columns, the **TimeNexus Converter** that is accessible from the Cytoscape Apps menu creates the tMLN (Figure 2: **2. build temporal multilayer network**; Supplementary Figure S1) which will appear as a collection of networks within Cytoscape: the *Flattened network*, the *Aggregated network*, as well as one static network for each layer (Figure 2). The *Flattened network* can be used to visualize the tLMN with the **TimeNexus Viewer.** In this view**,** the layers will be ordered on the X-axis according to time and the layer-nodes will be placed and aligned according to their position in the *Aggregated network.*

Extracting active subnetworks from tMLNs using PathLinker or ANAT

TimeNexus can be used to extract active subnetworks. To do so, the methods and apps for extracting the active subnetworks have to be chosen with the **TimeNexus Extractor** (Figure 2: **3. extract active subnetwork**, Supplementary Figure S1). First, the method for applying active subnetwork extraction on the tMLN needs to be set. There are several possible logical ways to extract active subnetworks from a temporal multilayer network: *global*, *pairwise* and *one-by-one* (Supplementary Figure S2). *Global* extracts an active subnetwork from the *Flattened network* representation of the tMLN. In the *global* method, intra-layer and inter-layer edges are not distinguished during the extraction, but are re-established for visualizing the final active subnetwork. This method only considers the queries of the first and the last layer as the source- and target- query nodes, respectively. *Pairwise* combines two adjacent layers in a single network and performs the extraction on this 2-layer *Flattened network*. Each layer is used twice, once as layer N and once as N+1. The active subnetworks are extracted as

in the *global* method for each pair of layers. Finally, all extracted active subnetworks are combined in one final active subnetwork over all time points (layers). *One-by-one* extracts active subnetworks on single layers and combines them at the final step into the final active subnetwork, again over all time points.

Second, the active subnetwork extracting app has to be selected. Currently, the **TimeNexus Extractor** (Figure 2, Supplementary Figure S1) offers the Cytoscape app PathLinker [17], which runs in the Cytoscape environment, and the ANAT server [18], which is called externally, for active subnetwork extraction. PathLinker is called by TimeNexus by its CyRest interface and performs the extraction on the user's computer. ANAT has a Cytoscape app called AnatApp, but its extraction algorithm is executed on an external server. TimeNexus directly calls this server through a SOAP interface and does not need the AnatApp to be installed to execute ANAT. We only refer to the ANAT server in this paper. When either of the extracting apps is called, TimeNexus displays the specific parameters that need to be set by the user (Supplementary Figure S1). Both apps provide default settings which can be adjusted by the user. For details on the usage and parameter choices of ANAT or PathLinker, the user should refer to the documentation of the respective chosen app. Either all layers of the tMLN or a subset of layers can be selected for active subnetwork extraction. The result of active subnetwork extraction from a tMLN is again a temporal multilayer network. It will appear in Cytoscape as a collection of active subnetworks similar to the network collection described above. It should be noted here that once active subnetworks have been extracted, the tMLN representing the active subnetworks is per definition no longer multiplex, as active subnetworks will have a different number of extracted nodes and edges on each of the layers, depending on the query nodes that have been defined for that specific layer (time point).

Visualizing temporal multilayer networks with TimeNexus

Finally, the **TimeNexus Viewer** enables users to visualize a temporal multilayer network. The tMLN can be visualized in several ways (Figure 2: **4. visualize temporal multilayer network**). In the *Aggregated network* view, all layers are collapsed into a single-layer network. The *Flattened network* shows the individual layers of the tMLN next to each other on a horizontal axis, preserving the position of a layer-node on each layer. The position of a layer-node depends on its position in the *Aggregated network* and layers are connected to each other by the inter-layer edges. Finally, each individual layer can be visualized. We provided a feature

to copy the layouts to multiple multilayer networks. It should be noted that the TimeNexus visualization is optimized for networks that have the same semantics, in our case nodes representing proteins or genes and edges interactions between those.

Yeast and mouse datasets used

Yeast cell cycle dataset

The yeast dataset from Kelliher et al. [34] was retrieved from the NCBI GEO database (GSE80474). We reprocessed the raw fastq files corresponding to the 36 first samples of the wild-type *S. cerevisiae* cultures from 0 to 175 minutes by mapping the reads to the *S. cerevisiae* S288C genome R64-1-1 with STAR aligner [36] with default parameters. Raw read counts were determined using featureCounts [37].

For all following steps, we selected 16 time points representing the first complete cell division cycle. These start at time point 25 minutes and last until time point 100 minutes as described by Kelliher according to the expression profiles of key cell cycle regulators. We renumbered these time points in our dataset to start at 0 min of the first full cycle (corresponding to 25 minutes in the original dataset) until 75 min (corresponding to 100 min in the original dataset). Using edgeR [38], lowly expressed genes were removed by the automatic function *filterByExpr* and the read counts were normalized by the Trimmed Mean of M-values (TMM normalization), resulting in normalized log-counts per million (logCPM). Then, we calculated the log2FC for each gene at time point i (t_i) versus its mean over the entire first cycle as follows:

$$\log2FC_{node}(t_i) = \log CPM_{node}(t_i) - \langle \log CPM_{node} \rangle$$

where logCPM is the log-counts per million given by edgeR and <logCPM> is the average logCPM over time for a given gene. Genes with a |log2FC| higher than or equal to 0.25 were considered differentially expressed and defined as query nodes at the respective layer where this cut-off criterion was met. As no replicates were available, we did not consider statistical significance for this dataset.

Time-resolved assay and RNA-sequencing dataset of a mouse pain assay

Pain assay

All experiments were conducted in line with the European guidelines for care and use of laboratory animals (Council Directive 86/609/EEC). All experimental procedures were approved by an independent animal ethical committee (APAFIS), as required by the French law and conform to the relevant institutional regulations of the French legislation on animal experimentation under the license number 20150702172422 62-V5#1537. All experiments were carried out according to the ARRIVE guidelines. C57/Bl6JRj male mice of 8-12 weeks of age were bought from Janvier Labs (https://www.janvier-labs.com). Mice were maintained under standard housing conditions (22°C, 40% humidity, 12 h light cycles, and free access to food and water). Special effort was made to minimize the number as well as the stress and suffering of mice used in this study.

Carrageenan-Induced Inflammation

20µl of a solution containing 1% carrageenan in H2O (weight/vol, Sigma) were injected subcutaneously into the plantar side of the left hindpaw, using a 30G needled syringe. Mechanical thresholds of the plantar surface were determined using Von Frey's filaments with the up-down method [39], prior to inflammation (D0) and one- (1d), three- (3d) and thirty-days (30d) post inflammation.

RNA extraction

Mice were deeply anesthetized with a mix of ketamine/xylazine and transcardially perfused with 5-10 mL RNA Later (Qiagen). L3 to L5 Dorsal Root Ganglia (DRG) were rapidly dissected and RNA was extracted by using RNeasy Micro Kit (Qiagen), according to manufacturer's instructions. For quality control, RNAs were loaded on an RNA NanoChip (Agilent) and processed with 2100 Bioanalyzer system (Agilent technology).

RNA sequencing

DRG RNAs were extracted in experimental duplicates from 2-3 mice each (2 pooled replicates). RNA-seq libraries were prepared using the TruSeq RNA Sample Preparation Kit (Illumina). All libraries were validated for concentration and fragment size using Agilent DNA1000 chips. Sequencing was performed on a HiSeq 2000 (Illumina), base calling performed using RTA (Illumina).

Data processing of RNA-seq datasets

Mouse sequencing data were quality controlled using FastQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/). We used cutadapt (https://cutadapt.readthedocs.io/) to trim adapter sequences. Resulting trimmed reads were mapped to the *M. musculus* genome version 10 (mm10) using STAR aligner with default parameters. Mapped data were re-analyzed with MultiQC [40]. Raw read counts were determined and filtered as described above for the yeast dataset. Differential expression analysis was done using edgeR, comparing the time points 1-day post injection (PI), 3 days PI and 30 days PI always against the 0 day control prior to injection. Finally, we showed the evolution of gene expression for the significantly differentially expressed genes of the mouse dataset by first computing the z-score of log counts per million and then splitting the significantly differentially expressed genes according to the time of their significant differential expression. Raw fastq files were submitted to the Gene Expression Omnibus database under the accession number GSE161764. We defined layer-nodes as query nodes if the associated gene had an adjusted p-value lower than 0.05 for that given time point (layer) versus the od control.

Building the *S. cerevisiae* and *M. musculus* interactomes and node tables

Both interactomes were built from the high-quality protein-protein interactions (PPIs) provided by HitPredict [41,42]. As recommended, interactions with a confidence score lower than 0.281 were removed to only keep high-quality interactions. We also removed self-loops in the network. For the yeast interactome, YEASTRACT+ protein-DNA interactions (PDIs) [43] were concatenated with the PPIs to obtain a more complete network. We used the extracting apps PathLinker and ANAT, both of which do not support multi-edges between nodes. Thus, we merged multi-edges of a given node pair by taking the mean of their confidence scores. For this, we assumed that a PPI is equal to 2 directed edges and set the confidence score of each PDI to 1. The final edge lists gave the undirected intra-layer edge tables. The nodes of the tMLN represent both, the genes and the proteins as the same entities, in case a node is both, a protein in a PPI or a regulated gene in a PDI. Nodes of genes that were not detected in the RNA-seq datasets were removed. Consequently, edges where one partner was removed were also filtered out. The weight for each individual layer-node was computed as follows:

w_{node} = -log10(p_adj) * |log2FC|

where p_adj is the adjusted p-value and log2FC the log2-fold change for the time point represented by that layer. As no replicates were available for the yeast cell cycle data, the p_adj term was ignored for this dataset. The node weight was then standardized between 0.01 to 1 (0.01 was chosen to avoid rejection of nodes with weight 0 by extracting apps). Moreover, a layer-node was tagged as a query for a layer if this layer-node had a |log2FC| $\geq$ 0.25 for the yeast cell cycle dataset; or if it had been defined as significantly differentially expressed with an adjusted p-value < 0.05 for the mouse dataset. Node names, node weights, as well as the information whether a node is a query node were contained in the node tables, enabling TimeNexus to set the correct attributes to the layer-nodes.

Extraction of active subnetworks

Active subnetworks were extracted with TimeNexus in combination with either ANAT or PathLinker from the yeast cell cycle time-series dataset. The algorithm "anchored network" with the sub-algorithm "approximation" was applied for ANAT. The network was set as "undirected" for PathLinker. For performance tests with PathLinker, we selected the optimal parameter K=750 by testing PathLinker with K-values from K50 to K=2000 and optimizing for the F1-score (Supplementary Table S4; see below for calculating performance measures). PathLinker with a K=50 was used to extract an active subnetwork for the mouse dataset, as the network size and the number of queries were both smaller. All other parameters were chosen by default.

Construction of maximum-weight, node-randomized and weight-randomized networks for robustness tests

To test the robustness of TimeNexus, we generated 3 types of multilayer networks from the yeast cell cycle tMLN. The *maximum-weight* network had intra-layer edge weights of 1 for each connection. The *node-randomized* network had node names shuffled in the node table, so the biological meaning of the network was lost. For the *weight-randomized* network, random weights were assigned to intra-layer edges following the uniform distribution

[0.01,1). 0.01 was chosen instead of 0, as ANAT removes edges with weight 0. For all networks, the node and inter-layer edge weights were not changed.

Calculating performance measures for extracted active subnetworks

We computed the extraction performances by testing if PathLinker and ANAT were able to recover the 130 genes of the KEGG yeast cell cycle pathway sce04111 [44]. In each extracted active subnetwork from the yeast cell cycle dataset, we counted the number of nodes in this active subnetwork (*# subnetwork nodes*) and the number of active subnetwork nodes overlapping with the 130 KEGG cell cycle genes (*True Positives (TPs)*). We then calculated the percentage of the active subnetwork size, the False Positives (*FPs*, as *subnetwork size minus TPs*), as well as the false negatives (*FNs,* as *# KEGG cell cycle genes minus TPs*). From these values, we computed a set of scores: the ratio of extracted nodes and the interactome size, as well as Recall, Precision, and F1-score as follows:

$$Recall = TP / (TP + FN)$$

$$Precision = TP / (TP + FP)$$

$$F1\text{-}score = 2 * ((Recall * Precision) / (Recall + Precision))$$

In addition, we performed GO enrichment analysis of active subnetworks to test for relevance of extracted nodes for the biological process 'cell cycle'. The tests were performed using modEnrichr for yeast [45]. We first extracted the expected enriched terms for 130 genes of the KEGG cell cycle pathway. A term was called "enriched" if its adjusted p-value was lower than 0.05. We then computed the percentage of these enriched terms related to KEGG cell cycle genes also found to be enriched for the nodes of the extracted active subnetwork. Finally, we also calculated this percentage of relevant terms at the first quartile (top 25% enriched terms) of the active subnetwork.

Enrichment analysis of active subnetworks extracted from mouse pain assay data

We used Enrichr [46] to calculate enrichments for active subnetworks extracted from the tMLN integrating the mouse pain assay temporal RNA-seq data and the mouse interactome. Enriched terms had an FDR <0.05 and a combined score > 100.

Results

Core functions of the TimeNexus app

TimeNexus was developed with the idea to create a versatile framework for working with temporal multilayer networks in the Cytoscape environment (Figure 2). This included a function to easily create tMLNs given tabular information on the structure of the network and its temporal dynamic – realized in the **TimeNexus Converter**. We wanted to enable users to visualize tMLNs in different ways – realized in the **TimeNexus Viewer**: in form of a *Flattened network*, which visualizes the tMLN itself, as well as an *Aggregated network*, representing the collapsed view of the tMLN. Finally, we wanted to be able to extract active subnetworks from tMLNs. We realized this by connecting TimeNexus to active subnetwork extracting apps available in Cytoscape that have a programmatic interface, PathLinker and the ANAT server. We wanted to take full advantage of the information provided by the temporal multilayer network. We therefore decided to include edge weights for the inter-layer edges of the tMLN that connect the same gene between two layers. These edge weights represent transition weights and describe the change in gene expression of a gene between two consecutive time points. The functionality for active subnetwork extraction was realized in the **TimeNexus extractor**.

We wanted to demonstrate and test the usability of TimeNexus by extracting active subnetworks from two temporal gene expression datasets: from the yeast cell cycle which offers highly resolved temporal information; and from mouse temporal gene expression data following pain response after injury with low temporal resolution.

Active subnetwork extraction using TimeNexus and PathLinker identifies relevant processes involved in early and late cell cycle events in *S. cerevisiae*

We wanted to test TimeNexus using a well-described, temporal biological system. We chose the budding yeast cell cycle as our model system. During the cell cycle, cells duplicate their content, replicate their DNA and at the end of the cycle faithfully divide into two identical cells. A cyclin-dependent kinase and its various, successive binding partners, the cyclins, drive progression of the cell cycle by precisely controlled events of phosphorylation, which is followed by the destruction of the kinase activity by the anaphase promoting complex (APC) at the onset of mitosis. Some cell cycle regulators are tightly controlled at transcriptional level. To test TimeNexus, we used time-resolved expression data from a previous study on

the transcriptional dynamics of the cell cycle [34]: in that study, *S. cerevisiae* cells were synchronized before releasing them to undergo three cell divisions. RNA was extracted each 5 minutes and subjected to RNA-sequencing to monitor the changes in gene expression during the three cell division cycles. We re-processed the raw read counts and used the normalized counts (see Methods) to calculate the log2 fold-change (log2FC) in expression for each gene of a time point versus the mean over one cycle. We created a tMLN of the first full cell cycle, representing time points 25min-100min as described in the original publication [34]. For demonstration purposes, we focused on three early time points of the cell cycle, which are characterized by cell growth and DNA replication (time points 0min, 5min and 10min representing time points 25min, 30min and 35min of the original dataset); and three late time points, which fall into the mitotic phase (60min, 65min and 70min, representing the time points 85min, 90min and 95min of the original dataset; see Supplementary Table S5). We also created a cell cycle interactome by using HitPredict and YEASTRACT+ interactions of the 130 cell cycle genes as defined by KEGG, encompassing the 130 nodes (genes/proteins) and 390 edges (interactions, see node table and intra-layer edge table in Supplementary Table S5). We used a |log2FC| cut-off of ≥ 0.25 to define a layer-node as a query node. Using **TimeNexus Viewer**, we created the *Flattened network* of the KEGG cell cycle (Figure 3 a). We used the *pairwise* method and PathLinker with default settings and a K of 150 to extract an active subnetwork from the three early and late temporal layers, respectively.

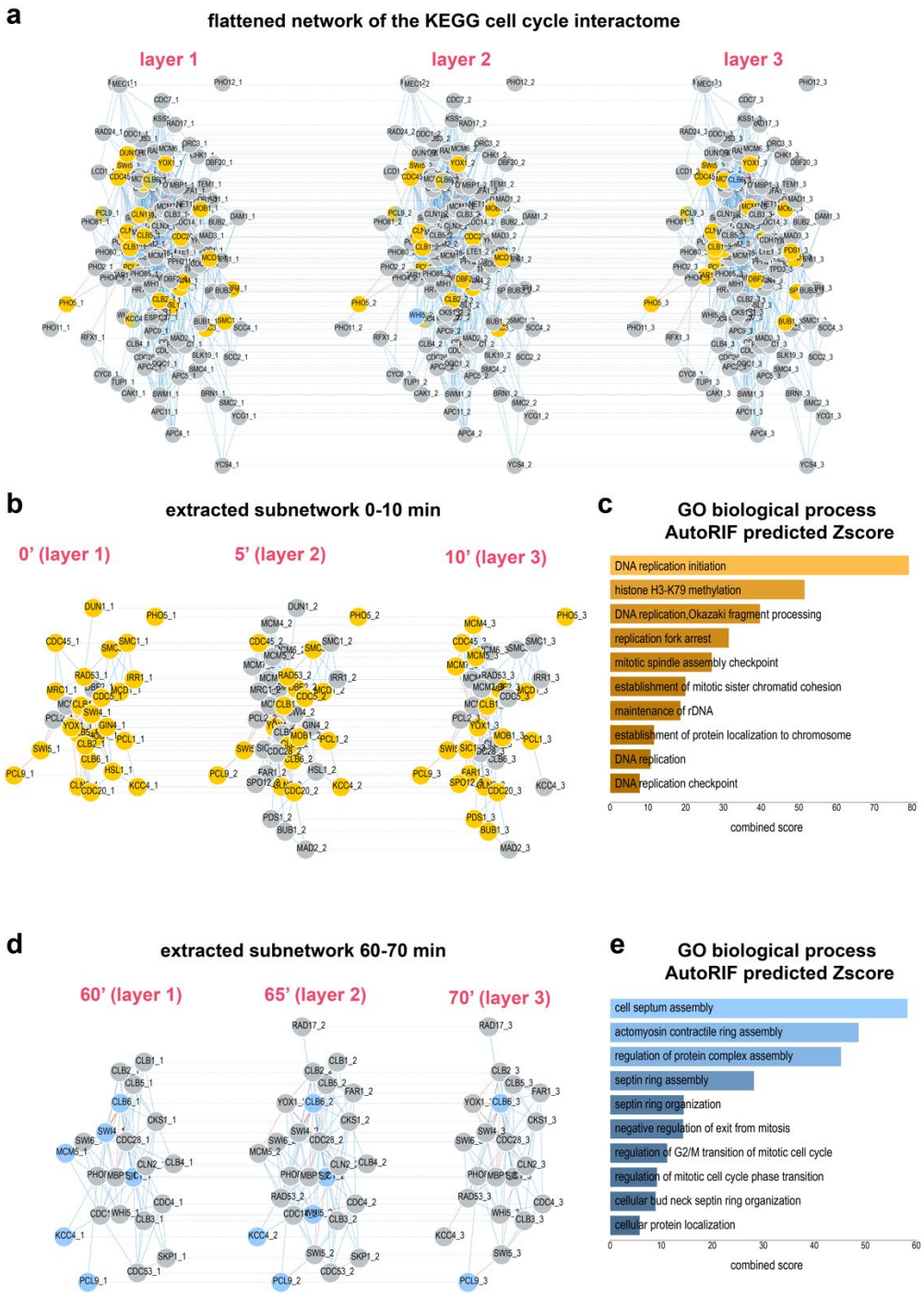

Figure 3: TimeNexus extracts active subnetworks from the yeast cell cycle interactome enriched in relevant biological terms related to cell cycle from early and late cell cycle stages. **(a)** *Flattened network* of the *S. cerevisiae* cell cycle pathway, containing core components of the yeast cell cycle as defined by KEGG. Yellow nodes are differentially expressed query nodes in the first three time points (0min, 5min, 10min) of the first full cycle in the time-series expression dataset [34], blue ones are differentially expressed query nodes in the late time points 60min – 70min; those with a gradient from yellow to blue are differentially regulated and therefore query nodes in both, early and late time points. Blue lines (edges) represent protein-protein interactions, red ones protein-DNA interactions. Dotted lines represent inter-layer edges. The interaction data were extracted from HitPredict and the YEASTRACT+ databases, respectively. **(b)** An active subnetwork was extracted from the first three time points of the yeast cell cycle (0min - 10min), containing genes differentially expressed in early phases of the cell cycle. **(c)** Enrichment analysis with genes in the early active subnetwork identified processes related to replication and active transcription. **(d)** An active subnetwork of late time points in the cell cycle (60min -

70min) was extracted. **(e)** Enrichment analysis of the genes contained in the late active subnetwork from time points 60 – 70min shown in **d** resulted in enriched pathways related to late processes in the cell cycle, such as contractile ring organization, cell septum assembly or septin ring assembly and organization. Shown in **b** and **d** are the extracted active subnetworks of core cell cycle components of the early and late phases as displayed by the **TimeNexus Viewer**. Active subnetworks were extracted using PathLinker (*pairwise* method, K=150).

The extracted active subnetwork of the early phase of the cell cycle contained 41 nodes and 134 edges (Figure 3 b). As expected, its members included proteins important for cell proliferation, DNA replication and active transcription, such as the MCM proteins MCM1 – MCM7, the cyclin dependent kinases CLB1, 2, 5, 6 and CLN2, as well as CDC28, CDC45, DBF2, SWI4, SWI5 or SIC1. To systematically identify enriched biological processes or phenotypes, we submitted the proteins of the active subnetwork to modEnrichr for yeast. We found that biological processes and phenotypes associated with early cell cycle phases were predominantly enriched (Figure 3 c, Supplementary Table S5).

To test whether extracted active subnetworks truly reflect cell cycle phases, we also used differentially expressed query genes in the three time points between 60 min and 70 min, reflecting the late stages of the yeast cell cycle, where cells prepare to undergo cell division (Supplementary Table S5). The active subnetwork extracted with PathLinker is substantially different from the one of the first three time points, with only 27 nodes and 101 edges (Figure 3 d). In accordance with the late stage in the cell cycle, genes involved in cell septum assembly, bud neck septin ring organization, actomyosin contractile ring assembly, regulation of G2/M transition and other late cell cycle events were enriched (Figure 3 e, Supplementary Table S5). Taken together, TimeNexus provides a versatile and useful platform to construct, manage and visualize tMLNs. By linking TimeNexus to active subnetwork extraction tools such as PathLinker, it is able to extract biologically meaningful, active subnetworks from the tMLN as demonstrated by analyzing time-resolved expression dynamics of the early and late yeast cell cycle.

TimeNexus performance in identifying relevant active cell cycle subnetworks from the *S. cerevisiae* interactome

We next wanted to test more rigorously the extraction performance of TimeNexus in combination with PathLinker or ANAT on tMLNs. More specifically, we were interested whether we could reliably extract the 130 genes defined by the KEGG cell cycle pathway from the yeast interactome using the time-resolved cell-cycle expression data [34]. Log2FC was

calculated as described above. The absolute log2FC was used as node weight, to compute inter-layer edge weights and to define layer-nodes as queries when their weight was equal to or higher than 0.25 (Supplementary Tables S6, S7). We built a high-quality interaction network for *S. cerevisiae* which included protein-protein, as well as protein-DNA interactions (see Methods and Supplementary Table S6). We constructed the tMLN using the **TimeNexus Converter** and extracted active subnetworks using PathLinker or ANAT. For PathLinker, the parameter K was set to 750 after optimization (Supplementary Table S4). We calculated the efficiency of both extracting apps by calculating Precision, Recall and F1 score for each time point individually, as well as over all 16 time points. Moreover, we performed GO enrichment analysis with the extracted nodes for each time point, as well as over the entire extracted active subnetwork. We scored the % enriched expected terms, so those identical to the original 226 terms enriched for the 130 KEGG-defined cell cycle genes, as well as the % top expected GO-terms in the first quartile of enriched GO-terms (Table 1 and Supplementary Table S8).

Generally, we could observe that PathLinker performed better than ANAT with our data. PathLinker extracted an active subnetwork that had 9.6% of the size of the entire yeast interactome. The overall Recall of core cell-cycle nodes was 45.4% for PathLinker, though Precision was 10.6% only, leading to an F1-score of 17.2%. 39% of expected GO-terms were found overall, and 71.7% expected GO-terms were retrieved in the first quartile of enriched GO-terms. The active subnetwork extracted by ANAT contained 12.3% of the total interactome. ANAT reached a Recall of 37.7%, a Precision of 6.9% and an F1-score of 11.7%. 35.2% expected GO-terms were found overall, and 68% within the first percentile of enriched GO-terms (Table 1). Recall, Precision and F1-score were dependent on the individual time point (layer). They peaked in the earlier phases of the cell cycle and dropped towards the end. This was not unexpected, as the number of query nodes was much lower in late phases of the cycle. Overall, expected GO-terms ranged around 50%, whereby the expected GO-terms in the first quartile seemed to be generally high throughout the entire cycle and with both extracting apps (Supplementary Table S8).

Table 1: Efficiency and robustness of TimeNexus-based active subnetwork extraction with PathLinker and the ANAT server over the entire tMLN

Robustness	Subnetwork size	Recall	Precision	F1-score	% expected GOs	% top expected GOs
TN+PL						
original	9.6	45.4	**10.6**	**17.2**	**39**	**71.7**
maximum edge weight	53.2	**93.1**	3.9	7.6	16.2	34.4
node-randomized	51.6	83.9	3.7	7.0	15.2	27.3
weight-randomized	11.9	35.4	6.7	11.3	28	47.6
TN+ANAT						
original	12.3	37.7	6.9	11.7	35.2	**68**
maximum edge weight	12.4	**41.6**	**7.5**	**12.7**	35.7	**68**
node-randomized	15.1	41.5	6.2	10.7	29	52.7
weight-randomized	12.6	40.8	7.3	12.4	**36.4**	67

We also wanted to know how sensitive active subnetwork extraction with either PathLinker or ANAT was to changes in the network structure or network attributes. To this end, we first changed the weights of all intra-layer edges to 1 (*maximum edge weight*); second, we shuffled the node names from the node table so the biological meaning of the network was lost, but its topology preserved (*node-randomized*); finally, we used random edge weights following a uniform distribution [0.01,1) for the intra-layer edges (*weight-randomized*). We observed that PathLinker was more sensitive to changes in the network structure or attributes than ANAT (Table 1): ANAT performance was overall in the same range for all extracted active subnetworks, though slightly higher performance could be observed for the maximum edge weight network. PathLinker, on the other hand showed significant differences (Table 1). The maximum edge weight, as well as the node-randomized networks resulted in very large extracted active subnetworks, both containing over 50% of the nodes of the original interactome. Consequently, Recall was very high (93.1% for maximum edge weight and 83.9% for node-randomized), and Precision very low (3.9 and 3.7%, respectively), resulting in low F1-scores (7.6% and 7.0%, respectively). The weight-randomized network showed general lower performance compared to the original one (Recall 35.4%, Precision 6.7%, F1-score 11.3%), together with lower % of enriched GO-terms relevant to cell cycle genes (28% expected and 47.6% top expected GO-terms). To conclude, TimeNexus in combination with particularly PathLinker was able to extract key cell cycle genes as defined by KEGG as an active subnetwork from the tMLN of the entire yeast interactome based on integrated

temporal cell cycle expression data, resulting in a significant enrichment of GO-terms related to cell cycle genes in the active subnetwork.

TimeNexus combined with PathLinker identifies active pathways relevant for tissue inflammation and repair in time-course expression data of pain induction in mouse

Next we tested, whether we could use TimeNexus on other systems, other model organisms and with less dense time-resolved data on differential gene expression. We used our own data from a time-resolved study of recovery from acute pain in mouse. In this experiment, Carrageenan is injected in the mouse hindpaw, inducing inflammation and mechanical hypersensitivity (Figure 4 a). The onset and the recovery from hypersensitivity can be measured by testing the ability of mice to respond to Von Frey filaments with increasing caliber. In this pain model, one day after Carrageenan injection, mice exhibit a significant decrease in their mechanical thresholds, which is a sign of inflammation-induced mechanical hypersensitivity. At day 3 post-inflammation (PI) mice recover normal mechanical sensitivity which remains steady at day 30 PI and beyond (Figure 4 a, Supplementary Table S9). In order to monitor the changes in gene expression in the pain-sensing dorsal root ganglia (DRG), we extracted RNA from these cells and performed RNA-sequencing before (0d), 1 day (1d), 3 days (3d) and 30 days (30d) after Carrageenan injection.

After differential expression analysis, we found that 60 genes were significantly differentially expressed between day 0 and day 1 PI and 38 genes showed significant differential expression between day 0 and day 3 PI (Supplementary Table S9). Finally, only 4 genes were significantly differentially expressed at day 30: Apoe (Apolipoprotein E), Itgb8 (Integrin Beta-8), Ncam2 (Neural Cell Adhesion 2) and Slc25a37 (Mitochondrial Iron Transporter 1). The temporal expression dynamics of significantly differentially expressed genes collected from the 3 comparisons showed that genes were generally upregulated between day 0 and day 1, while the majority of them was downregulated between day 3 and 30. Genes with a significant differential expression between day 30 and day 0 were few (Figure 4 b). In conclusion, acute pain induced a temporary significant differential expression of genes in DRG and the vast majority of genes returned to basal expression levels after full recovery of the mouse at day 30.

We next were interested whether we could extract active subnetworks relevant for this process from expression data integrated with the mouse PPI interactome using tMLNs. Using

TimeNexus, we built a high-quality mouse interactome using HitPredict. We created a pain node table from the differential expression data for the three time points 1d, 3d and 30d PI compared to the 0d time point prior to injection. Query nodes were defined as having an adjusted p-value lower than 0.05 (Supplementary Table S9). Using TimeNexus, we generated the tMLN for these data. We used PathLinker with K=50 and the method *pairwise* to extract active subnetworks (Figure 4 c). Layer '1d vs 0d' contained a network with 23 genes. Among those were genes involved in immune response (Stat1, Irf7, Traf6, Rsad2 and TifA), as well as cell survival and stress response (Arnt, Epas1, Hif3a, Hif1a, Mcl1, Gsk3b, Grb2 and Egfr1, as well as Stat1 and Irf7). At the second time point at 3d versus 0d, genes involved in immune response were still prevalent, as were genes involved in the regulation of apoptosis. The network is less homogenous with respect to pathways at time point 30d versus 0d. We found some genes involved in axonal growth, as well as negative regulation of apoptosis. Finally, we performed enrichment analysis of the entire active subnetwork, as well as the individual time points (1d, 3d, 30d, versus 0d control) and could confirm the enrichment of pathways and GO terms related to immune response and inflammation, regulation of apoptosis, as well as neuronal processes (Figure 4 d, Supplementary Table S9).

In conclusion, by extracting active subnetworks from the temporal multilayer network created with TimeNexus, we could identify genes involved in direct response to inflammation, cellular stress and regulation of apoptosis, as well as neuronal processes in DRG following Carrageenan-induced inflammation.

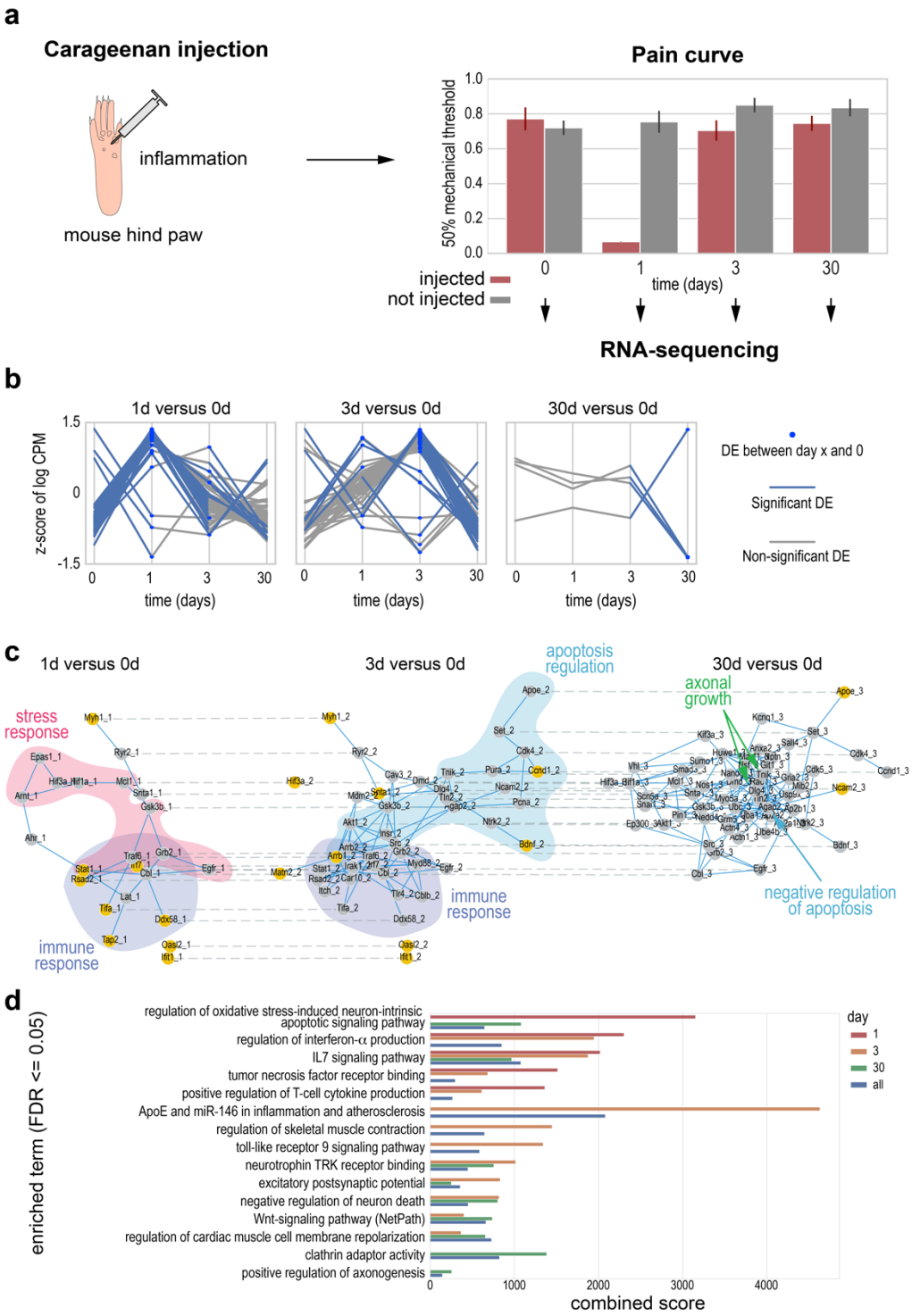

Figure 4: Identification of pathways relevant for cellular stress response, apoptosis, immune response, as well as axonal growth in mouse sensory neurons after Carrageenan-induced inflammation. (a) We injected Carrageenan in the hind paw of a C57BL/6J mouse, which induces inflammation and pain, affecting the sensory neurons. We monitored the mechanosensitivity of the paw before injection, as well as 1, 3 and 30 days after injection. We observed high mechanosensitivity up to day 1. Thereafter, we observed complete recovery of the mechanosensitivity by day 3, which persisted at least until day 30. We isolated the dorsal root ganglions at those time points and performed RNA-sequencing, identifying significant differential gene expression between time point compared to day 0 control (0d, before injection). **(b)** The 3 plots show the significantly differentially expressed genes varying over time. These genes were grouped according to their appearance in the 3 time points, day 1, day 3 or day 30 each compared against the 0d control. Consistent with the onset of injury and

inflammation, we could see strong induction of gene expression at day 1, as well as day 3 after injury, while at day 30, only few genes were significantly differentially expressed compared to the od control. Genes that are significantly differentially expressed at two time points will be present in each of the two associated plots. Blue dots indicate significant differential expression of a gene at the given time point. Blue lines indicate significant differential expression between two time points. Y-axis is plotted as the z-score of the log-transformed counts per million. **(c)** From a tMLN based on the entire mouse interactome, we extracted an active subnetwork containing 3 layers, one for each time point compared to the od control using PathLinker (*pairwise* method, K=50). We extracted an active subnetwork containing genes relevant for the pain assay: at day 1, we found genes involved in stress (red bubble) and immune response (blue bubble). At day 3, we identified genes involved in immune response (blue bubble), as well as regulation of apoptosis (cyan bubble). Finally, at day 30, a more heterogenous set of proteins was identified, including anti-apoptotic genes (cyan arrow), as well as genes involved in axonal growth (green arrows). Orange nodes represent query nodes (which showed significant differential expression at a given time point versus od control). Active subnetwork extraction returned Steiner nodes (grey nodes), i.e. nodes that are part of the network, but were themselves not significantly differentially expressed and, thus, not query nodes. Solid blue lines are protein-protein interactions within one layer (intra-layer edges), dashed lines represent inter-layer edges. **(d)** Enrichr enrichment results of WikiPathways and Gene Ontology (GO) Biological Process (BP) and Molecular Function (MF). Nodes from each of the layers (day 1, 3, and 30) as well as the layers of all nodes of the active subnetwork (all) were used for enrichment analysis. Enrichments of the first two time points included terms related to immune and stress response, encompassing signaling pathways involved in these processes. The signature changed at the later time point (day 30), where more terms related to apoptosis, as well as axonogenesis were enriched. Enriched terms had an FDR <0.05 and a combined score > 100.

Discussion

We introduced here TimeNexus, a Cytoscape app to create, manage and visualize temporal multilayer networks. TimeNexus is easy to use: tMLNs can be created either by uploading a collection of tables that contain attributes of the nodes, as well as information on edges; or by adding temporal information to a static Cytoscape network. The **TimeNexus Viewer** allows to visualize the tMLN by creating different views, enabling users to focus on the single static, as well as the dynamic features of the tMLN. This first release of TimeNexus furthermore provides a framework to extract active subnetworks from a tMLN. To this end, we create static networks of the tMLN in three different ways which are standard Cytoscape networks that can be handled by basic Cytoscape features, as well as other Cytoscape apps. These objects are created either *globally* over the entire tMLN by combining all layers in a single-layer network with layer-nodes as separate entities and by ignoring the differences between intra- and inter-layer edges; *pairwise* by creating a single-layer network from two consecutive layers similar to the *global* method, over the entire tMLN structure; or *one-by-one* by creating a single-layer network for each individual layer. The *global* method has the drawback that the network to be analyzed increases drastically, as the initial interactome is multiplied by the number of layers. Active subnetwork extraction is therefore compute-intense. Moreover, only nodes from the first and last layers will be used as source and target nodes and active subnetworks will only be extracted if they span the entire dataset. The *one-by-one* method on the other hand uses less memory, but does not consider inter-layer edges, so the nature of the temporal multilayer network is ignored. The *pairwise* method is a good compromise between both methods and therefore recommended especially with larger networks or many time points. The *global* and *pairwise* method also take full advantage of TimeNexus' unique feature to work with transition weights between layers, representing expression changes of a gene between two time points. While we have combined TimeNexus with tools to extract active subnetworks from interactomes, it should be noted that any Cytoscape app for network analysis can be combined with TimeNexus, as algorithms are applied to a classical static network structure by the *global*, *pairwise* or *one-by-one* method. The only pre-requisite is the availability of a programmatic interface for the chosen app.

We tested TimeNexus by extracting active subnetworks in combination with the Cytoscape apps PathLinker and the ANAT server. PathLinker outperformed ANAT in extracting biologically relevant, active subnetworks and worked better in our hands. It was however also

more sensitive to specific network attributes, such as intra-layer edge weights. This is not surprising, as it uses edge weights to calculate scores for paths between nodes to extract active subnetworks. The user should therefore carefully choose intra-layer edge weights in order to extract meaningful biological information from the network. We also observed that the selection of query nodes has a substantial effect on the results. In general, the overall performance of both extracting apps was mediocre, which might be owed to the test itself: we tried to extract cell cycle genes from the KEGG-defined yeast cell cycle pathway. Many of these genes are not regulated on RNA-level but rather by phosphorylation or protein degradation. While for some processes, RNA- and protein expression levels correlate quite well [47], this is not necessarily the case for cyclic processes such as the cell cycle, where a rapid activation or destruction of regulatory proteins is required and thus, protein phosphorylation as well as degradation play an important role. However, we did not want to artificially bias the test to extract differentially expressed genes, but rather wanted to know, how efficiently we could recover well-described, core cell cycle genes from the tMLN using either of the two apps, irrespective of their RNA expression dynamics. Therefore, it might not be surprising that both, Recall, as well as Precision were not high with either of the two tested apps. Furthermore, it should be noted that PathLinker and ANAT are optimized to extract active subnetworks from static single-layer networks, not from a temporal multilayer network and thus may not fully consider the information a multilayer network offers.

There are three other Cytoscape apps available for integrating temporal data with interactomes: DyNetViewer, DyNet and TimeXNet. DyNet is not able to extract active subnetworks, which excluded it from further consideration. DyNetViewer creates individual temporal layers from expression data directly, removing all nodes from an interactome that are not significantly differentially expressed. In principle, the output of the DyNetViewer could be used to create directly an active subnetwork within TimeNexus. But this app also omits transition weights from one layer to the next and therefore, is not taking full advantage of the temporal information provided. Yet, its visualization properties exceed those of TimeNexus. TimeXNet can be used for active subnetwork extraction from temporal expression data. However, it defines only three phases, representing early, middle and late genes, which could correspond to the layers in a multilayer network representation. If a higher temporal resolution is required and available, as is the case for a cyclic process such as the cell cycle, the classification in these three phases is difficult to make. Moreover, in

TimeXNet, one gene can only be part of one phase, which limits the usability of this tool for cyclic processes even further. We therefore decided not to use it for performance tests, as it would have significant disadvantages compared to TimeNexus in combination with the extracting apps PathLinker or ANAT.

We used TimeNexus in combination with PathLinker to extract active subnetworks from a time-resolved pain assay in mouse, based on expression data from the pain sensing dorsal root ganglia. While we did not find a large amount of significantly differentially expressed genes, we identified by performing tMLN analysis with TimeNexus an active subnetwork that contained genes relevant for the process of inflammation: genes involved in immune response, in cellular stress response and in anti-apoptotic signaling, as well as – at late stages – genes involved in axonal growth. Our active subnetwork contained many Steiner nodes, representing genes that were not initially identified as significantly differentially expressed. This demonstrates that integrating and analyzing temporal gene expression data together with interaction data leads to meaningful biological insights that can also help in the design of further experimental studies.

In conclusion, TimeNexus is a Cytoscape app that introduces true temporal multilayer networks within the Cytoscape environment. While we have used it to create, manage, visualize and analyze temporal data projected on a multilayer network that is multiplex, it can also handle other kinds of multilayer networks. We have combined the first release of TimeNexus with two apps for active subnetwork extraction, PathLinker and ANAT. However, TimeNexus builds native Cytoscape objects which can be handled by core Cytoscape features or other apps dedicated to network analysis. Therefore, TimeNexus can be extended with other Cytoscape apps, provided they offer a programmatic interface. Consequently, TimeNexus can be added into existing pipelines and workflows as an app for analyzing temporal multilayer networks.

Acknowledgements

Mouse and yeast interaction data from HitPredict, as well as YEASTRACT+ interactions were kindly provided by the respective authors of the resources. We thank Andy Saurin for support in RNA-sequencing of mouse pain assay samples. We thank Anais Baudot, Maxime Lucas, Friedhelm Pfeiffer and all members of the IBDM Computational Biology Team for critical reading of the manuscript. This work was supported by ANR-grant 17-CE16-0020-02 (Myochronic) awarded to AM and BHH, Aix-Marseille University and the French National Centre for Scientific Research (CNRS).

References

1. Spies, D., Renz, P. F., Beyer, T. A. & Ciaudo, C. Comparative analysis of differential gene expression tools for RNA sequencing time course data. *Brief. Bioinformatics* **20,** 288–298 (2019).
2. Spies, D. & Ciaudo, C. Dynamics in Transcriptomics: Advancements in RNA-seq Time Course and Downstream Analysis. *Comput Struct Biotechnol J* **13,** 469–477 (2015).
3. Kumar, L. & E Futschik, M. Mfuzz: a software package for soft clustering of microarray data. *Bioinformation* **2,** 5–7 (2007).
4. Kaur, S. *et al.* Temporal ordering of omics and multiomic events inferred from time-series data. *NPJ Syst Biol Appl* **6,** 22–7 (2020).
5. Hestilow, T. J. & Huang, Y. Clustering of gene expression data based on shape similarity. *EURASIP J Bioinform Syst Biol* **2009,** 195712 (2009).
6. la Fuente, de, A. From 'differential expression' to 'differential networking' - identification of dysfunctional regulatory networks in diseases. *Trends Genet.* **26,** 326–333 (2010).
7. Grimes, T., Potter, S. S. & Datta, S. Integrating gene regulatory pathways into differential network analysis of gene expression data. *Sci Rep* **9,** 5479–12 (2019).
8. Charitou, T., Bryan, K. & Lynn, D. J. Using biological networks to integrate, visualize and analyze genomics data. *Genet Sel Evol* **48,** 27–12 (2016).
9. Rhodes, D. R. & Chinnaiyan, A. M. Integrative analysis of the cancer transcriptome. *Nat. Genet.* **37 Suppl,** S31–7 (2005).
10. Nguyen, H. *et al.* A Comprehensive Survey of Tools and Software for Active Subnetwork Identification. *Front Genet* **10,** 155 (2019).
11. Mitra, K., Carvunis, A.-R., Ramesh, S. K. & Ideker, T. Integrative approaches for finding modular structure in biological networks. *Nat. Rev. Genet.* **14,** 719–732 (2013).
12. Ideker, T., Ozier, O., Schwikowski, B. & Siegel, A. F. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics* **18 Suppl 1,** S233–40 (2002).
13. Huang, S.-S. C. & Fraenkel, E. Integrating proteomic, transcriptional, and interactome data reveals hidden components of signaling and regulatory networks. *Sci Signal* **2,** ra40–ra40 (2009).
14. Tornow, S. & Mewes, H. W. Functional modules by relating protein interaction networks and gene expression. *Nucleic Acids Res.* **31,** 6283–6289 (2003).
15. Cline, M. S. *et al.* Integration of biological networks and gene expression data using Cytoscape. *Nat Protoc* **2,** 2366–2382 (2007).
16. Ritz, A. *et al.* Pathways on demand: automated reconstruction of human signaling networks. *NPJ Syst Biol Appl* **2,** 16002–9 (2016).
17. Gil, D. P., Law, J. N. & Murali, T. M. The PathLinker app: Connect the dots in protein interaction networks. *F1000Res* **6,** 58 (2017).
18. Almozlino, Y., Atias, N., Silverbush, D. & Sharan, R. ANAT 2.0: reconstructing functional protein subnetworks. *BMC Bioinformatics* **18,** 495–5 (2017).
19. Atias, N. & Sharan, R. iPoint: an integer programming based algorithm for inferring protein subnetworks. *Mol Biosyst* **9,** 1662–1669 (2013).
20. Yosef, N. *et al.* ANAT: a tool for constructing and analyzing functional protein networks. *Sci Signal* **4,** pl1–pl1 (2011).

21. Yosef, N. *et al.* Toward accurate reconstruction of functional protein networks. *Mol. Syst. Biol.* **5,** 248 (2009).
22. Przytycka, T. M., Singh, M. & Slonim, D. K. Toward the dynamic interactome: it's about time. *Brief. Bioinformatics* **11,** 15–29 (2010).
23. Holme, P. & Saramäki, J. Temporal networks. *Physics Reports* **519,** 97–125 (2012).
24. Holme, P. Modern temporal network theory: a colloquium. *The European Physical Journal B* **88,** 234 (2015).
25. Mucha, P. J., Richardson, T., Macon, K., Porter, M. A. & Onnela, J.-P. Community structure in time-dependent, multiscale, and multiplex networks. *Science* **328,** 876–878 (2010).
26. Masuda, N. & Holme, P. Detecting sequences of system states in temporal networks. *Sci Rep* **9,** 795–11 (2019).
27. Thompson, W. H., Brantefors, P. & Fransson, P. From static to temporal network theory: Applications to functional brain connectivity. *Netw Neurosci* **1,** 69–99 (2017).
28. Patil, A. & Nakai, K. TimeXNet: identifying active gene sub-networks using time-course gene expression profiles. *BMC Syst Biol* **8 Suppl 4,** S2–8 (2014).
29. Patil, A., Kumagai, Y., Liang, K.-C., Suzuki, Y. & Nakai, K. Linking transcriptional changes over time in stimulated dendritic cells to identify gene networks activated during the innate immune response. *PLoS Comput Biol* **9,** e1003323 (2013).
30. Goenawan, I. H., Bryan, K. & Lynn, D. J. DyNet: visualization and analysis of dynamic molecular interaction networks. *Bioinformatics* **32,** 2713–2715 (2016).
31. Li, M., Yang, J., Wu, F.-X., Pan, Y. & Wang, J. DyNetViewer: a Cytoscape app for dynamic network construction, analysis and visualization. *Bioinformatics* **34,** 1597–1599 (2018).
32. Yen, J. Y. Finding the K Shortest Loopless Paths in a Network. *Management Science* **17,** 712–716 (1971).
33. Winter, P. Steiner problem in networks: A survey. *Networks* **17,** 129–167 (1987).
34. Kelliher, C. M., Leman, A. R., Sierra, C. S. & Haase, S. B. Investigating Conservation of the Cell-Cycle-Regulated Transcriptional Program in the Fungal Pathogen, Cryptococcus neoformans. *PLoS Genet* **12,** e1006453 (2016).
35. Su, G., Morris, J. H., Demchak, B. & Bader, G. D. Biological network exploration with Cytoscape 3. *Curr Protoc Bioinformatics* **47,** 8.13.1–24 (2014).
36. Dobin, A. *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29,** 15–21 (2013).
37. Liao, Y., Smyth, G. K. & Shi, W. featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. *Bioinformatics* **30,** 923–930 (2014).
38. Robinson, M. D., McCarthy, D. J. & Smyth, G. K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics* **26,** 139–140 (2010).
39. Chaplan, S. R., Bach, F. W., Pogrel, J. W., Chung, J. M. & Yaksh, T. L. Quantitative assessment of tactile allodynia in the rat paw. *Journal of Neuroscience Methods* **53,** 55–63 (1994).
40. Ewels, P., Magnusson, M., Lundin, S. & Käller, M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* **32,** 3047–3048 (2016).

41. López, Y., Nakai, K. & Patil, A. HitPredict version 4: comprehensive reliability scoring of physical protein–protein interactions from more than 100 species. *Database (Oxford)* **2015,** (2015).
42. Patil, A. & Nakamura, H. Filtering high-throughput protein-protein interaction data using a combination of genomic features. *BMC Bioinformatics* **6,** 100–13 (2005).
43. Monteiro, P. T. *et al.* YEASTRACT+: a portal for cross-species comparative genomics of transcription regulation in yeasts. *Nucleic Acids Res.* **48,** D642–D649 (2020).
44. Kanehisa, M. & Goto, S. KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.* **28,** 27–30 (2000).
45. Kuleshov, M. V. *et al.* modEnrichr: a suite of gene set enrichment analysis tools for model organisms. *Nucleic Acids Res.* **47,** W183–W190 (2019).
46. Kuleshov, M. V. *et al.* Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. *Nucleic Acids Res.* **44,** W90–7 (2016).
47. Buccitelli, C. & Selbach, M. mRNAs, proteins and the emerging principles of gene expression control. *Nat. Rev. Genet.* **21,** 630–644 (2020).

TimeNexus: a novel Cytoscape app to analyze time-series data using temporal MultiLayer Networks (tMLNs)

Michaël Pierrelée [1], Ana Reynders [2], Fabrice Lopez [3], Aziz Moqrich [2], Laurent Tichit [4] and Bianca H. Habermann [1]

Supplementary Data:

Supplementary methods

Notes to the selection of query nodes for the yeast cell cycle data and the mouse pain assay

We defined the query nodes by selecting the differentially expressed genes. Genes were called differentially expressed if they were deregulated between a time-point and a reference. The reference was the average gene expression across time for the yeast dataset and a control time-point for the mouse dataset. Comparing to a control time-point is relevant for perturbation experiments, where we measure the effects of a modified factor. When monitoring periodic processes (e.g. cell-cycle) or continuous processes (e.g. morphogenesis), choosing a particular time-point would be arbitrary, so deregulation can be defined by comparing either an expression level to an average expression, two consecutive time-points or expression levels using other statistical models. These models must be able to select the interesting genes at a given time-point as query nodes to guarantee that the extraction will prioritize them using TimeNexus together with either PathLinker or ANAT. Thereby, paying attention to the sample rate of time-series experiments is equally critical, because a given process cannot be studied with the same model when the sampling rates do not have the same order (e.g. minutes vs. hours vs. days).

Supplementary Tables:

Supplementary Table S1: Exemplary node table

Node	Weight_1	Weight_2	Weight_3	Query_1	Query_2	Query_3
YKL022C	0.039733842	0.031992741	0.028613951	FALSE	FALSE	FALSE
YGL116W	0.309084735	0.39682512	0.65252339	TRUE	TRUE	TRUE
YLR103C	0.395338334	0.293985868	0.274277839	TRUE	TRUE	TRUE
YMR001C	0.524117777	0.332445277	0.178371881	TRUE	TRUE	FALSE
YPR119W	0.548604495	0.399577103	0.202970354	TRUE	TRUE	FALSE
YPR120C	0.317746605	0.218357517	0.140080492	TRUE	FALSE	FALSE
YGR109C	0.561372231	0.343806274	0.222554583	TRUE	TRUE	FALSE
YMR199W	0.281998475	0.210768255	0.21303971	TRUE	FALSE	FALSE
YPL256C	0.412229379	0.295000464	0.288928797	TRUE	TRUE	TRUE

Supplementary Table S2: Exemplary intra-layer edge table

source	target	Weight	edge type
YKL022C	YLR102C	0.380438563	PPI
YKL022C	YLR127C	0.382194976	PPI
YKL022C	YMR001C	0.250513473	PPI
YGL116W	YML027W	0.25	PDI
YGR092W	YML027W	0.25	PDI
YLR103C	YLR274W	0.379473319	PPI
YLR103C	YMR043W	0.229153333	PPI
YLR103C	YPL153C	0.320794327	PPI
YLR103C	YPR019W	0.382440191	PPI
YLR127C	YMR001C	0.267785362	PPI
YLR127C	YNL172W	0.370272197	PPI

Supplementary Table S3: Exemplary inter-layer edge table

source	target	Weight_1>2	Weight_2>3
YKL022C	YKL022C	0.066926195	0.057143418
YGL116W	YGL116W	0.413802554	0.512040048
YLR103C	YLR103C	0.408047313	0.362352138
YMR001C	YMR001C	0.461370301	0.338106537
YPR119W	YPR119W	0.486700829	0.375993519
YPR120C	YPR120C	0.349002463	0.263860409
YGR109C	YGR109C	0.4751148	0.361577509
YMR199W	YMR199W	0.330102969	0.297658094
YPL256C	YPL256C	0.414255788	0.368658674

Supplementary Table S4: PathLinker optimization tests for different K (# of paths)

# of paths PathLinker	Subnetwork size	Recall	Precision	F1-score
K50	3.5	19.2	12.5	15.2
K100	5.6	29.2	11.7	16.7
K150	6.9	33.1	10.8	16.3
K200	7.4	36.9	11.2	17.1
K250	7.6	37.7	11.2	17.2
K500	8.6	40.8	10.6	16.9
K750	**9.6**	**45.4**	**10.6**	**17.2**
K1000	10.4	46.2	10	16.4
K2000	12.2	53.9	9.9	16.7

Supplementary Table S5: Excel sheet with data for Yeast cell cycle interactome (original node tables and intra-layer edge tables for early and late cell cycle phases, as well as enrichment results for early and late tMLNs

Supplementary Table S6: Excel sheet with node table, intra-layer -, and inter-layer edge table for entire yeast interactome with 16 cell cycle layers

Supplementary Table S7: relation of query nodes to KEGG cell cycle genes in queries of the original KEGG cell cycle tMLN

Query node-layers	# query nodes	# KEGG genes in query
Layer 1	243	26
Layer 2	148	18
Layer 3	242	20
Layer 4	193	14
Layer 5	132	11
Layer 6	135	13
Layer 7	172	23
Layer 8	249	29
Layer 9	264	32
Layer 10	281	34
Layer 11	178	10
Layer 12	98	4
Layer 13	100	6
Layer 14	87	5
Layer 15	62	2
Layer 16	69	2

Supplementary Table S8: PathLinker and AnatApp/ANAT Layer-by-Layer results

TN+extracting app	Subnetwork size	Recall	Precision	F1-score	% expected GOs	% top expected GOs
TN+PL_1	3.1	20.8	15.2	17.5	50	87.76
TN+PL_2	3.5	25.4	16.2	19.7	47.44	84.48
TN+PL_3	4.0	23.8	13.2	17.2	43.64	59.32
TN+PL_4	3.7	20.8	12.6	15.7	42.06	75.47
TN+PL_5	3.3	20	13.7	16.3	48.81	83.33
TN+PL_6	3.5	25.4	16.1	19.7	54.03	88.68
TN+PL_7	3.1	20	14.7	16.9	58.33	84.44
TN+PL_8	3.8	23.1	13.8	17.2	56.84	82.98
TN+PL_9	4.1	27	14.7	19	50.42	83.05
TN+PL_10	4.5	31.5	15.8	21.1	45.49	78.26
TN+PL_11	3.3	23.1	15.6	18.6	54.23	86
TN+PL_12	2.6	13.8	11.8	12.7	48.95	55.56
TN+PL_13	2.8	14.6	11.7	13	45.11	65.22
TN+PL_14	2.4	14.6	14	14.3	45.36	63.04
TN+PL_15	1.9	10	12.9	10.9	44.03	60.61
TN+PL_16	1.8	8.5	10.8	9.5	43.97	60
TN+ANAT_1	2.0	12.3	13.7	13	61.62	92
TN+ANAT_2	2.6	13.8	11.8	12.8	51.70	81.08
TN+ANAT_3	4.7	17.7	8.5	11.4	35.98	48.48
TN+ANAT_4	3.6	11.5	7.2	8.9	29.45	46.34
TN+ANAT_5	2.5	9.2	8.2	8.7	46.08	60
TN+ANAT_6	2.7	10.8	9.1	9.9	49.32	77.78
TN+ANAT_7	3.3	17.7	12.2	14.4	58.05	86.05
TN+ANAT_8	4.7	23.8	11.3	15.3	47.66	74.58
TN+ANAT_9	5	25.4	11.5	15.8	44.44	80
TN+ANAT_10	5.2	26.2	11.3	15.8	40.80	81.33
TN+ANAT_11	3.5	13.1	8.3	10.1	38.98	54.55
TN+ANAT_12	1.8	3.1	3.8	3.4	27.42	46.67
TN+ANAT_13	2.0	5.4	5.9	5.6	31.11	45.45
TN+ANAT_14	1.9	5.4	6.4	6.9	42.65	35.29
TN+ANAT_15	1.2	1.5	2.8	2	27.08	33.33
TN+ANAT_16	1.4	2.3	3.8	2.9	32.65	33.33

Supplementary Table S9: excel table with original data on mechanical sensitivity assay, DEGs from mouse pain assay, node and intra-layer edge tables, plus GO enrichment of individual layers, we well as the entire extracted subnetwork

Supplementary Figures
Supplementary Figure S1

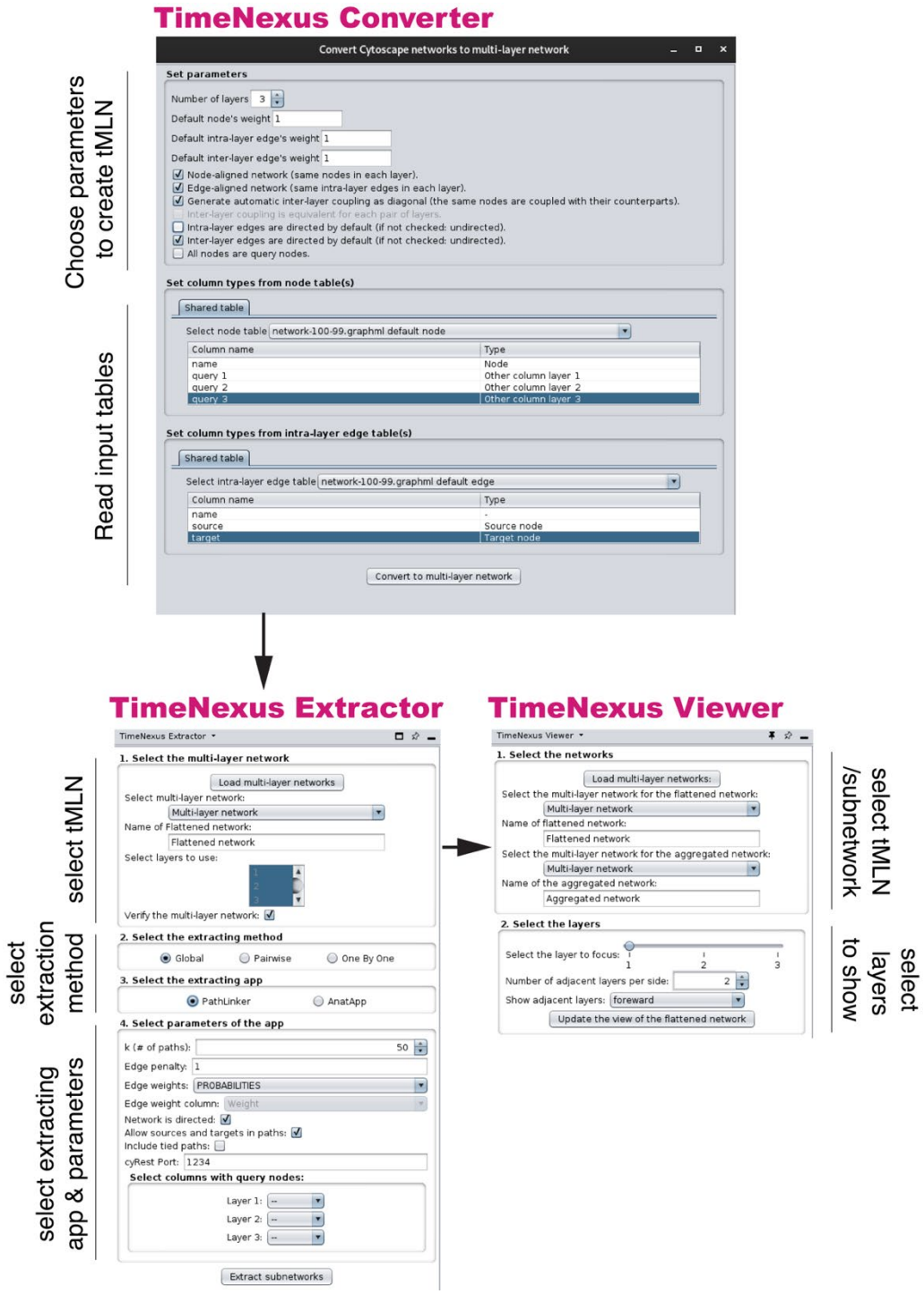

Figure S1 legend: The TimeNexus Converter, Extractor and Viewer interfaces. For creating a tMLN, first the table attributes have to be assigned to create the structure of the multilayer network using the **TimeNexus Converter**. In the **TimeNexus Extractor,** first a tMLN, second the extraction method and third the extracting app have to be chosen. The columns need to be individually assigned prior to extraction. The **TimeNexus Viewer** is needed to display a tMLN. Again, a tMLN has to be loaded and the layers to show have to be chosen.

Supplementary Figure S2

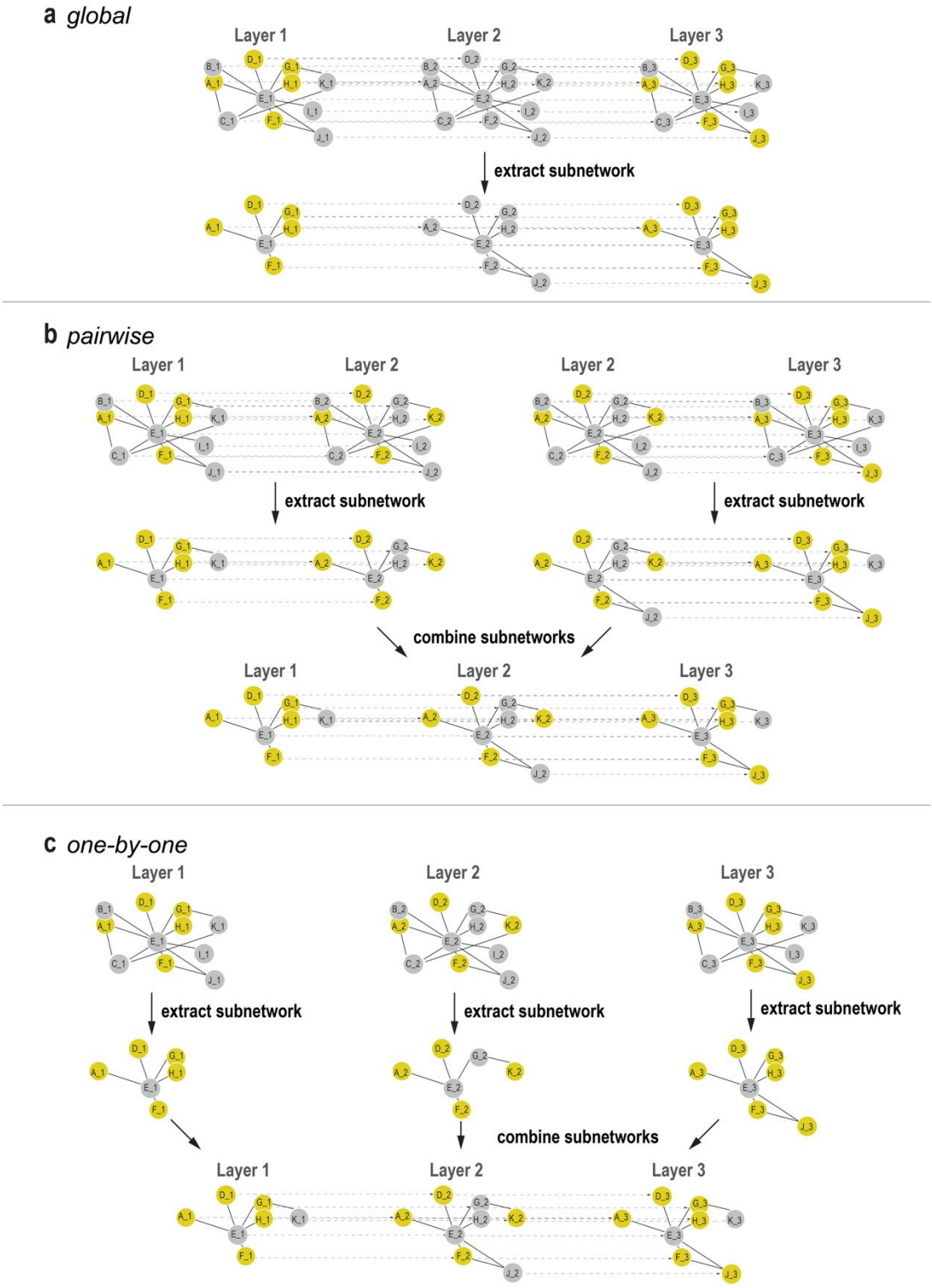

Figure S2 legend: Different extraction methods used in TimeNexus. (a) the *global* method extracts subnetworks over the entire flattened network-like structure, considering only the query nodes in the first and last layer. **(b)** in the *pairwise* method, two neighboring layers are collapsed for subnetwork extraction, whereby each layer is once layer N and once layer N+1. The subnetworks are combined to a final temporal multilayer subnetwork. **(c)** in the *one-by-one* method, one subnetwork is extracted per layer and all subnetworks are then combined to the final temporal multilayer subnetwork.

Discussion and conclusion

1 *Chlorella* sp. HS2

In (Yun *et al.*, 2020), we explored the phenotypic changes and the dysregulated pathways of the microalga *Chlorella* sp. HS2, so-called HS2, under salt stress. The microalga has a difficult but existing growth and produces high amount of lipids in salt water compared to fresh water. This makes HS2 a halotolerant and oleaginous microalga species. In both conditions, it undergoes an exponential growth phase before entering in a stationary growth phase. RNA-sequencing measured gene expression for both conditions and both phases. I applied functional enrichment and I returned the **KEGG** pathways with dysregulated genes. In particular, we observed a down-regulation of photosynthesis at the exponential phase, as previously observed for microalgae under stress, before they recover at the stationary phase. This was consistent with measures of photosynthesis efficiency. We also observed that the Krebs cycle was strongly down-regulated, while enzymes related to Acetyl co-enzyme A and NAPDH were up-regulated. Both metabolites are co-factors to build lipid molecules. We concluded that the availability of these co-factors was pushing lipid synthesis, instead of being used for classic energy production, explaining the oleaginity of HS2.

The main challenge of this bioinformatics analysis is that HS2 does not have any annotated genome or at least pre-built transcriptome. Therefore, it requires to apply *de novo* assembly and gene annotation. This quickly expands the number of steps in the workflow. Each step increases the risk of errors. I could only estimate the reliability of results by comparing the results to close species.

For this study, I applied several workflows to process data and test for differential expression and I obtained consistent results each time. The current workflow correlated well with a simpler workflow based on **SuperTranscripts** (Davidson, Hawkins and Oshlack, 2017) and **Blast2GO** (Conesa *et al.*, 2005): around 85% of differentially expressed genes (DEGs) were shared. To compute the differential expression analysis, I initially applied a normalization method called RLE (relative log expression) (Anders and Huber, 2010) and I already found more than 20% of DEGs among the gene list. This is a lot for this class of normalization methods. Knowing that HS2 underwent dramatic phenotypic changes, I applied another method, called **SVCD** (Roca *et al.*, 2017), not sensitive to the ratio of DEGs. It gave around 30% of dysregulated genes. Down-regulated genes dominated the DEG list. This result was highly consistent across the tested workflows. To annotate the DEG without being restricted to a given species, I chose to assign them a **KEGG ortholog** (KO). This resulted in more than 75% annotated DEGs. I aggregated the DEG values to the KO level. 15% of DEGs could not be directly aggregated together because they had opposite dysregulations. Some were down-regulated, while others were up-regulated. Yet, I noticed that most of these conflicts were caused by genes with high adjusted p-values, especially because of a lower expression. Therefore, I applied a stringent cutoff at 0.0001 (the usual cutoff is at 0.01) for the adjusted p-value, removing half of the conflicting DEGs. Reducing the cutoff cost 20% of KOs, but it also decreased the risk

for false positives and helped a manual exploration of the list. This stringent cutoff is also justified by the fact that the conditions are expected to have a low within-condition variability because the biological replicates are cell cultures averaging individual variability. Thus, p-values are more decreased than in other experiments.

The experiment had 3 biological replicates per condition. It should be noted than the condition of HS2 under salt stress at the stationary growth phase had a higher within-condition variability than the 3 other conditions. One replicate was indeed less similar. This difference could come from a lower efficiency during the library preparation or the sequencing. Therefore, having more replicates would enable to withdraw one replicate that would decrease the statistical power of the differential expression analysis (DEA). From a statistical point of view, 2 replicates do not enable to generalize the results to other experiments. 3 replicates per condition is the lowest possible number for statistical inference. For example, (Soneson and Delorenzi, 2013) observed that only 2 biological replicates per condition dramatically increases the false DEG detections. However, (Chen, Lun and Smyth, 2016) found 2 biological replicates acceptable if most of the genes have little biological variability and if the differential expression is high, i.e. the differences between the conditions are clear. Based on around 42 replicates per condition, (Schurch *et al.*, 2016) recommended *at least* 6 replicates. 3 replicates enabled to only find 20 to 40% of the actual DEGs. To find most DEGs (90% with an absolute log-fold change higher than 0.5), it required at least 12 replicates. More replicates increased the statistical power for detecting genes with a smaller differential expression. This recommendation was a bit exaggerated according to (Lamarre *et al.*, 2018). Based on a meta-analysis of 17 datasets, they concluded that the maximal useful number of replicates would be 10 replicates. Consequently, I would advocate to have at least 4 biological replicates, if necessary at the expense of sequencing depth and complex experimental design. As illustrated above, 4 replicates would have enabled me to withdraw one without losing the inferential capabilities of the experiment.

Differential expression analysis produces a list of DEGs to further explore. In (Yun *et al.*, 2020), I applied functional enrichment and pathway analysis. Functional enrichment was not so helpful to interpret the results of the DEA. Only few terms had a low enough adjusted p-value. The other terms with a low p-value were not informative. On the contrary, pathway analysis with **KEGG mapper** (Kanehisa and Sato, 2020) produced pathway maps showing the DEGs. These pathway maps enabled to find meaningful relationships between dysregulated genes. The relationships are interactions, so one can qualitatively estimate how a DEG is integrated in its pathway and what parts of the pathway are dysregulated. Functional enrichment does not provide such exploration. Yet, pathway analysis requires to have enough dysregulated genes in a given pathway to make a conclusion on its general dysregulation, e.g. as in our case, in which the Krebs cycle is down-regulated. The advantage of functional enrichment is indeed to filter out the pathways which are expected to have few dysregulations by chance. Both approaches, functional enrichment and pathway analysis, are strongly dependent on the input list of DEGs.

The conclusions are based on the accumulation of dysregulations. Yet, differential expression analysis can generate false negatives and false positives. The cutoff of false positives is very low, but they are still possible. We cannot ensure that pathways with a low number of dysregulated genes are not more dysregulated. This could change the final model proposed in our study. Few genes of interest could be tested to confirm hypotheses, but this approach cannot validate the whole results of a differential expression analysis. To do so, (Leek, Taub and Rasgon, 2012) suggested a statistical approach. First, the user randomly tests genes from the dataset by qPCR. Then, their approach computes the probability that the actual FDR is lower than the FDR initially defined by the user. To confirm the DEA, this probability must be much higher than 0.5. The testing must be random, otherwise it would bias the probability. For example, to have a validation probability of 0.5 for a FDR at 5%, then at least 240 genes should be tested. If there are 3 replicates, (Lamarre *et al.*, 2018) would suggest a FDR at 10%, giving 110 genes to test. A technology such as the BRB-sequencing (Alpern *et al.*, 2019) can be used as an independent validation technique as it has a lower cost than multiple qPCR. The approach of Leek and colleagues is of course not necessary if the goal of the RNA-sequencing is to determine few target genes. However, validating the DEA is required to confirm hypotheses generated from a DEG list.

The DEA workflow depends on a series of algorithms which have been benchmarked. While these benchmark studies took complementary approaches and apply them on various datasets, they are still a collection of benchmarks. There is no meta-analysis. Meta-analysis studies evaluate hypotheses by reprocessing results from previous studies. There are necessary to make general conclusions, such as if one tool is more efficient than another. Two issues limit the production of meta-analyses. First, the field has a fast development with many novel approaches and improvements. Second, no "golden standard" is available for benchmark studies, which make them hardly comparable. Furthermore, these meta-analyses – and benchmark studies in general – should not just focus on comparing tools between them. They should also enable to know whether one can apply a tool to a given dataset. Indeed, tools have domains of applicability. For example, (Soneson and Delorenzi, 2013) defined borderline cases for normalization methods by searching for a proportion of DEGs making the DEA results unreliable. These statistics could be compared to tables from meta-analysis studies in order to confirm whether a given tool would be applicable. In the absence of such studies, the DEA results should be put in question. From a practical point of view, it is difficult to rationally choose one tool or algorithm over another.

This work contributed to the community by proposing a workflow for data processing of de novo RNA-sequencing as well as hypotheses about how HS2 adapts to salt stress. Pathway analysis generated results about the relationships between dysregulated genes from known pathways. Yet, they do not enable to find relations between pathways or from unknown pathways. Network-based methods can solve this issue by exploring interactions between genes regardless known pathways. These methods thus work at a lower scale than pathways, i.e. at the scale of interactions instead of group of interactions.

2 TimeNexus

The goal of developing the **TimeNexus** app was to identify dynamic pathways using the results of the differential expression analysis from time-course gene expression datasets. It should be robust to the typical lack of power of the DEA, i.e. missing DEGs, as well as the inability of RNA-sequencing to detect dysregulation outside gene expression, e.g. post-translational modifications. Therefore, it should be able to especially return non-dysregulated genes which are predicted to take part to the pathway. It should also enable to filter out dysregulated genes which are not involved in the studied pathway, thereby removing false positives.

I developed **TimeNexus**, a **Cytoscape** app to build and visualize temporal multilayer networks (tMLN) (Pierrelée *et al.*, 2020). It extracts subnetworks by adapting the format of multilayer networks and applying the static extraction tools **PathLinker** (Ritz *et al.*, 2016; Gil, Law and Murali, 2017) and **AnatApp** (Yosef *et al.*, 2011; Almozlino *et al.*, 2017). The apps use the dysregulated genes as input nodes (i.e. source and target nodes) to search for subnetworks. To extract a subnetwork, it uses one of the three methods: global, local or pairwise. We applied the pairwise method in order to consider the dysregulated genes of each layer. We tested **TimeNexus** on time-course RNA-sequencing datasets from yeast cell-cycle and mouse sensory-neuron inflammation.

With the yeast dataset, we evaluated its efficiency to extract the cell-cycle pathway by counting the genes which are part of the cell cycle pathway. I built the yeast interactome by aggregating a protein-protein interaction (PPI) network and a protein-DNA interaction (PDI) network. In particular, the overall recall of **TimeNexus** with **PathLinker** or **AnatApp** was respectively 45% and 38%, with a precision of 11% and 7%, respectively. A minimal recall of 33% was expected as genes involved in the cell-cycle were already dysregulated. These rates were low. They could be caused by the fact that PPIs play a major role in the cell-cycle, that the DEG selection was not appropriate, that network weights were not ideal or that the extracting apps are not fitted for complex networks. We also evaluated the effect of intra-layer edge weights by replacing them with a random value or the maximal weight. The effect of the network structure was evaluated by shuffling the node names. In all cases, these changes had dramatic effects for **PathLinker** but not for **AnatApp**. **PathLinker** extracts subnetworks by only using edge weights, while **AnatApp** also considers node weights which were computed from the DEA results (size effect and significance). **PathLinker** is expected to be more affected by edge weights. However, it is surprising to observe that this app returned half of the network when the intra-layer edge weights were equal, i.e. it did not extract anything, while randomized weights generated results similar to those of **AnatApp**. **AnatApp** returned not much more than 33% of dysregulated cell-cycle genes. It implies that **AnatApp** was not able to reach the initial goal to predict non-dysregulated genes. All these observations do not have clear explanations. In fact, the behavior of both apps when extracting subnetworks is not well known. They were not tested with random networks. These evaluations are surprisingly not common in the literature of methods for biological networks. Further studies should focus on this point.

For the mouse dataset, we searched to determine a pathway for sensory neurons which would be activated under inflammation. The interactome of the mouse dataset was only the PPI network. The temporal multilayer networks had 3 layers: day 1 (after inflammation) versus day 0 (before inflammation), day 3 versus day 0 and day 30 versus day 0. This is an exogenous pathway, while the cell cycle is an endogenous pathway. Following an idea similar to that presented in section 3.1.1, we first observed 2 main groups of dysregulated genes by plotting their expression profile. A first group had most of its genes up-regulated at day 1 before returning to their basal level. The genes of the second group were up-regulated at day 2 and returned to their basal level at day 1 and day 30. Only few genes remained dysregulated at day 30. These observations were consistent with the fact that individuals appeared to have recovered from inflammation at that time point. Extracting subnetworks from the tMLN showed that parts of the interactome were successively activated by inflammation. On the first day, a group of genes related to stress response and immune response were dysregulated. On the third day, the immune response evolved and supplemented by a gene group participating in the regulation of apoptosis. The stress response genes no longer appeared at this time. A heterogeneous group of genes was extracted at the last day of the experiment, but it was centered on genes related to axonal growth and apoptosis regulation. These functional groups of genes are consistent with an inflammation process. While our conclusions represent only hypotheses, this approach illustrates the usefulness of **TimeNexus** to intuitively and qualitatively explore a biological process over time.

It should be pointed out that the yeast dataset does not have any biological replicate and the mouse dataset has only two replicates. Moreover, the latter are pooled samples, i.e. RNAs from several individuals were pooled together to give one indivisible sample. This is necessary when the total RNA amount is too low to use for a standard library-preparation protocol. Yet, (Rajkumar *et al.*, 2015) strongly advised against such an approach. Indeed, it introduces a bias due to the pooling, which averages the biological variations. It breaks the theorem of variance decomposition which demonstrates that the total variance of a gene is the sum of the within-condition variance and the between-condition variance. Yet, the DEA tests for differential expression from the estimated parameters of the model (mean and variance). By removing the within-condition variance, the statistical methods cannot properly estimate the total variance. Therefore, the results would be unreliable. The authors compared the DEA results in two cases. In the first one, the tested conditions had 8 individual RNA samples. In the other, they made 2 pools of 8 RNA samples for each condition, before starting the library preparation for the sequencing. The authors observed that the pooling experiments detected thousands more DEGs than experiments with individual samples. Thus, pooling strongly increased the number of false positives. Therefore, this strategy should be avoided as much as possible.

(Pierrelée *et al.*, 2020) introduced temporal multilayer networks (tMLNs) for networks of molecular interactions. This is the first report of an app on **Cytoscape** managing multilayer networks. **TimeNexus** represents tMLNs as standard **Cytoscape** networks to enable other apps to process them. The layers can be easily imported by dynamic-

network visualization apps, such as **DyNet** (Goenawan, Bryan and Lynn, 2016). To my knowledge, it also the first time that subnetwork extraction was adapted and applied to multilayer networks. The first limit of **TimeNexus** is that it cannot build any type of multilayer network. It only manages one aspect with inter-layer edges between consecutive layers. The second limit comes from **Cytoscape** networks used to model the flattened network. As the latter contains all layer-nodes as independent entities within **Cytoscape**, the user's computer must have enough memory to load a full interactome on multiple time-points. For example, a tMLN of the yeast dataset cannot be loaded on a standard laptop; it requires more than 10 gigabytes of RAM. The CPU requirements are lower because subnetwork extraction depends on the extracting apps. **PathLinker** does not exploit multiprocessing and **AnatApp** executes its algorithm on a dedicated server. These limits cannot be solved by using classic **Cytoscape** objects. A future version of **TimeNexus** could fully exploit the symmetries of biological tMLNs as presented here, i.e. the same nodes and edges on each layer. Indeed, a feature of **Cytoscape** can create virtual objects which are represented as physical objects on the graphical interface, but are internally linked to the same object. Nonetheless, if the multilayer network has no symmetries, there is no possibility to compress data, i.e. it would be equivalent to the current situation.

A third issue is that **PathLinker** and **AnatApp** are not optimized for multilayer networks. For example, they do not allow multiple edge types, while multilayer networks have intra- and inter-layer edges. Despite not being optimized for multilayer networks, these apps are themselves constrained by the weights (e.g. no node weight for **PathLinker**), the edge directions (e.g. either fully directed or undirected for **PathLinker**), the multi-edges (e.g. not allowed by **AnatApp**) and the input nodes. Therefore, the results from both apps are not directly comparable in (Pierrelée *et al.*, 2020) because the tMLNs used for each app are not the same. Source and target nodes are necessarily defined for each layer when using the pairwise method, otherwise the extracting apps won't process the multilayer network. Subnetwork extraction should enable either to have any number of intermediate input nodes, instead of just 1 as **TimeXNet** (Patil and Nakai, 2014), to be able to process a multilayer network or to merely have no input nodes. Moreover, input nodes are categorical variables: a node is either an input or not. Instead, one could consider a continuous variable such as a weight to model the interest in a given node. This is initially the function of layer-node weights depending on the size effect and the significance from the DEA. Developing a method to extract subnetworks requires to test its behavior according to extraction parameters and features of the network. Random networks are critical to ensure the methods are efficient and not biased.

Finally, **TimeNexus** should have an API to enable users to automatize their workflow and connect it to **Python** or **R**, following (Ono *et al.*, 2015). This API should also enable app developers to add their own extraction algorithm to **TimeNexus** using one of three extraction methods already implemented (pairwise, global or local). Yet, these methods return static networks. If the new version of **TimeNexus** includes a new format of multilayer networks, an interface should also be developed to enable third-part apps to extract subnetworks from the multilayer networks.

Appendix – About good DEA practices

I present below complementary details about limitations of standard sequencing protocols, experimental designs and differential expression analysis.

For a concise review of this field, the reader can refer to (Chowdhury, Bhattacharyya and Kalita, 2020b).

Note: references to sections below are related to this appendix, unless otherwise indicated.

1 Generating input data for differential expression analysis

See section 2.1 of the introduction for a general workflow of RNA-sequencing.

1.1 Library preparation

Following a standard library-preparation protocol such as *TruSeq* (Illumina, 2010), the RNA extraction should finish with a minimal RNA quantity of at least 0.1 μg. With lower quantity or if the RNAs were degraded, this protocol is not adapted. This quantity gives the *total RNA* amount. The library preparation starts by removing the ribosomal RNAs (rRNA) which overcome the messenger RNAs (mRNA). In the standard protocol, we purify the mRNAs by using magnetic beads attaching them. Rather than purifying mRNA, it is also possible to deplete rRNA, but it requires specific protocols. This enables in particular to sequence the non-coding RNAs and the degraded mRNAs. (Holik *et al.*, 2017) confirmed the efficiency of rRNA depletion to reach these goals. Interestingly they also showed the method increases clustering of samples and predicts better differential expression, with a boost of 20-30% in the signal, by accounting for read counts within introgenic regions and pre-mRNA. Yet, they did not recommend one method over the other. For complementary information, (Haile *et al.*, 2019) evaluated protocols for rRNA depletion for samples with degraded RNAs or in low concentrations. They concluded it was more efficient to selectively degrade rRNAs rather than to selectively remove them out of the samples.

Then, the mRNAs are fragmented to a median size of around 150 base pairs because Illumina sequencing cannot manager longer fragment sizes. As Illumina only sequences DNA, we reverse transcribe the mRNAs to cDNA. The reverse transcription is in two parts. First, it creates a DNA strand complementary to the RNA strand. Second, it degrades the RNA strand and replaces it by a DNA strand. In the standard protocol, we do not know if the cDNA molecule was a mRNA from the plus strand or the minus strand of the genomic DNA. Indeed, a gene can be on one or the other strand. Thus, we are losing information that we will need to identify the reads during the bioinformatics steps. The TruSeq protocol is said *non-stranded*. (Sultan et al., 2012) developed a *stranded protocol* to identify the genomic DNA strand from which the mRNA was transcribed by synthesizing the second cDNA strand with dUTP. The dUTP-marked strand is then degraded before the PCR amplification. For (Corley et al., 2017), the stranded protocol improved the mapping of reads to the DNA (see section 1.1.2) by resolving multi-mapped reads, i.e. reads ambiguously assigned to multiple

transcripts because of a short length. It also decreased the number of false positives and false negatives within the DEG list (see section 2).

Next, we add adapters to each cDNA according to their library of origin and we amplify the cDNAs by PCR (polymerase chain reaction). The adapters have a tag to identify the library of origin of the RNA. It enables to pool all the samples so we can sequence them at the same time. Doing so, we remove the *confounding effects* related to the sequencing (see section 1.2) (Auer and Doerge, 2010). Note that it does not removed the effects related to the RNA extraction and library preparation, i.e. *batch effects*. Indeed, the PCR amplification increases the number of reads from sequencing (see next section 1.1.1) by increasing the number of cDNA fragments. This can add *amplification biases* to the sequencing, i.e. some fragments being more amplified than others.

To correct these batch effects and decrease the RNA quantity needed by the library preparation, (Alpern *et al.*, 2019) developed a protocol called *BRB-sequencing*. This method is inspired from single-cell RNA-sequencing where the RNAs are pooled together at the beginning of the preparation. It dramatically decreases the financial costs by simplifying the library preparation. Indeed, if an experiment has a given number of samples, then the biologist needs to prepare each library in parallel. The BRB-sequencing protocol enables to prepare all samples at the same time by tagging the RNAs before to pool them. Contrary to TruSeq, the first step of the library preparation is to synthesize the first strand of the cDNA. This enables to tag the tags to the 3'-end of mRNAs. Thereby, we can pool together the samples containing mRNAs. The next steps are the synthesis of the second strand, the fragmentation of the double stranded cDNA and the PCR amplification. The BRB-sequencing loses the 5'-end after the fragmentation. Thus, this protocol cannot address the experimental questions requiring the full RNA-sequencing, such as the differential expression at the transcript-level. However, it enables to compute differential expression at the gene level, as presented herein, with the same *statistical power* as standard TruSeq. In my point of view, BRB-sequencing could replace quantitative PCR (qPCR) to validate the DEA results by testing a high number of genes (see section 2.6). That being said, *3'-enrichment technologies* are recent. We still need to evaluate their advantages and drawbacks at a large scale.

1.1.1 Sequencing

We load the pooled libraries into one or more *lanes* of a *flow cell* within the sequencer. If the libraries were not pooled, we would have one library per lane and so, confounding effects. Very briefly, the lane tethers the cDNA fragments and the sequencer synthesizes a new strand for each fragment. In the meanwhile, it monitors the base pairs added to the new strands during the synthesis. The synthesis stops after a certain number of base pairs, between 75 and 150 base pairs. This gives one read for one end of the cDNA fragment. Thereby, the cDNA fragment is not entirely sequenced. Contrary to *single-end sequencing*, *paired-end sequencing* repeats the synthesis but for the other end. It generates two paired reads for one cDNA fragments. Paired-end sequencing increases the read mapping (i.e. transcription identification) and the downstream results, but less than using a stranded protocol (Corley *et al.*, 2017). Thus,

there are 3 main sequencing parameters: the read length, the sequencing depth (total number of reads) as well as the choice between single- and paired-end sequencing. Choosing the parameters will depend on the objectives of the study and the budget. We will discuss this choice in the next section (1.2).

Besides, the sequencing depth is fixed. It means that the sequencing will assign reads to a transcript depending on the available pool of reads. This pool depends on the sequencing depth, but also on the other transcripts. For example, let's take two transcripts and a sequencing depth fixed at 100 reads. If the first transcript accumulates 70 reads for any reason, the other will have 30 reads. If we have 200 reads, each will have 140 and 60 respectively. This gives a ratio. Therefore, RNA-sequencing measures the *relative abundance* of transcripts. We do not know the exact number of molecules in the cell. It is a critical point for normalization methods (see section 2.3).

Among the reasons explaining that some transcripts accumulate more reads is their length (Oshlack and Wakefield, 2009). This is typical bias of RNA-sequencing that we do not find in the microarray technology.

1.1.2 Data processing

The goal of the data processing is to get the gene abundance from the raw reads. Before that, one can apply a pre-processing step on the reads. We observe that the sequencing biases the 3'-end of raw reads. One could *trim* this part to improve the next step, but it could bias the differential expression analysis (Williams *et al.*, 2016). I would suggest to avoid such trimming to simplify the workflow.

See section 2.2 of the introduction for more details.

1.2 General considerations about the experimental design

Herein, we consider experiments aiming to compare *conditions*. A condition is a group of homogenous samples that underwent the same *factors* (e.g. treatment, mutation, tissue, time). A factor has one or more *levels*. For example, let's take the factor "mouse mutation" with 3 levels "mutation 1", "mutation 2" and "wild-type". We can add a factor "treatment" with the levels "treated" and "not treated" to the experiment. Then, an experimental design could have 6 conditions combining a level of "mutation" and a level of "treatment". We search to explore the effects of each factor on the genes of our biological material by comparing the conditions.

To make conclusions, we need two important things. First, we have to be sure that a given level of a given factor is the cause of the observed effect. We must not mix several factors together or add uncontrolled factors to a condition. An unwanted factor is called a *confounding factor*. The measured effect is mixed up between the controlled and confounding factors. It breaks the causal chain from the factor to the effect.

Second, the experiment is not useful if we are not able to *generalize* the results of the experiment to any other experiments (Auer and Doerge, 2010). We do that using *statistical inference*. It *estimates* statistical properties from the samples for each condition, in particular the variance. This variance can only be computed if we have

biological replicates, i.e. several samples for the same condition. Contrary to *technical replicates*, the biological replicates are from different biological materials, but which underwent the same factors. From our example above, we can consider 3 treated mice with the same mutation (e.g. "mutation 1"). They give us 3 samples combining the same factors. More the mice are different, more we can generalize our results but it will be harder to see any differences between the conditions. Moreover, few biological replicates can create confounding effects, so we need to increase their number. In practice, we consider the individuals (e.g. one mouse, one cell culture) with the same genetic background as biological replicates. The individual variability would be enough to generalize the results to the population. I will present other statistics considerations for differential expression analysis (DEA) in section 2.1.

We are now seeing how to optimize the experimental design for DEA in RNA-sequencing.

1.2.1 Biological replicates and sequencing depth

Sequencing can adjust for the total number of reads to sequence, this is the sequencing depth (see section 1.1.1). It is usually between 1 and 100 million (M) reads per sample. A higher depth identifies more differentially expressed genes (DEGs), i.e. genes with a significant variation between conditions. As the experiment budget is limited, one can identify more DEGs by increasing either the sequencing depth or the replication (number of replicates in each condition). Increasing sequencing depth requires less efforts but it is not the optimal choice.

(Liu, Zhou and White, 2014) sequenced human cell lines, treated or not. Each condition had 7 replicates and 30M reads were sequenced per sample. Compared to increasing the replication, they observed that increasing sequencing depth had little effect on the capacity to identify DEGs (i.e. statistical power, see section 2.1). For example, from 10M to 30M reads per sample, with 2 samples per condition, they observed the power increased by 20%. Yet, at 10M reads with 3 samples per condition, the increase was by 40%. After 4 biological replicates, this effect was reduced. They concluded that 10M reads was enough. To increase the power, the replication should be preferred over the sequencing depth. Although some of them preferred 20M reads, this conclusion was consistent with the results of (Rapaport *et al.*, 2013; Soneson and Delorenzi, 2013; Ching, Huang and Garmire, 2014; Rajkumar *et al.*, 2015; Seyednasrollah, Laiho and Elo, 2015; Schurch *et al.*, 2016; Lamarre *et al.*, 2018). In the same way, one should prefer more replicates than using paired-end sequencing (Corley *et al.*, 2017).

See section 1 of the discussion for recommendations about the optimal number of biological replicates.

1.2.2 Pooling samples

We saw there are two confounding factors during sequencing: the *batch effect* from the library preparation and the *lane effect* from the sequencing in the flow cell. Hopefully, (Marioni *et al.*, 2008) showed that the lane effect in RNA-sequencing is very low and only few genes (0.5%) are concerned. This means there is little need for technical replicates. However, some lanes can still have issues and are anyway limited in each flow cell. After collecting data, statistical methods cannot remove confounding factors

without partially or totally losing statistical power. For example, if a lane of a cell, where the sequencing takes place, has a lower efficiency, then the sample will be less sequenced than the others. However, if all samples were in this lane, then all samples would be equally affected. Therefore, the samples should be pooled together as soon as possible to reduce the technical differences between them. Consequently, (Auer and Doerge, 2010) suggested to pool the samples by using multiplexing after library preparation. Their simulations showed this kind of approach outperforms the other kinds of experimental designs.

See section 2 of the discussion for recommendations about the optimal number of biological replicates.

RNA-sequencing is powerful, but it comes with its own biases. Today, 3'-enrichment RNA-sequencing enables to avoid pooling caused by a lack of RNAs. It also reduces the batch effects by applying multiplexing much earlier in the library preparation. Yet, the more important aspect is the replication. It is necessary to correct use to apply a differential expression analysis.

2 Identifying dysregulated genes using differential expression analysis

Assuming that the samples of the dataset have the same biological materials but in different proportions, *differential expression analysis* (DEA) identifies the genes with an unexpected variation of their expression by comparing the samples from two or more conditions. We call them *differentially expressed genes* (DEGs). They have to measures: the *size effect*, expressed as a *log2-fold change* (LFC), and the *significance*, expressed as a low *p-value*. For a simple pairwise comparison, the fold change is the ratio of the gene abundance between a given condition and a reference condition. It is on a log2-scale. In the context of pathways, we assume the genes have a constant expression in the reference condition.

2.1 About statistics and reliability of computational approaches

Among a population of elements (e.g. genes), we do not know the elements belonging to the *positive class* (e.g. differential expression) and those belonging to the *negative class* (e.g. constant expression). The statistical tests aim to predict which elements are positive and which are negative. These tests are not perfect. Sometimes, they predict that an actual positive element is negative, resulting in *false negatives*. Conversely, they predict that an actual negative element is positive, resulting in *false positives*. In practice, for each element, the statistical test computes the probability to observe its data, assuming the *null hypothesis* that the element was negative. This probability is the *p-value*. If the p-value is low enough, it is unlikely that the element is negative and thus, we predict that the element is positive. Doing so, we accepted the risk that the element is false positive. We call this risk the *significance level*, i.e. the threshold under which the p-value is low enough to reject the null hypothesis.

To measure the prediction efficiency, we compute several statistical measures by counting the number of correct and incorrect calls. In particular, we are interested in:

- the *area under the receiver operating characteristic (ROC) curve (AUC)*, informing about the capacity to detect true positive over false positive;
- the *sensitivity*, also called *recall* or *statistical power* (i.e. the rate of true positives on the actual positive class);
- the *specificity* (i.e. the rate of true negatives on the actual negative class);
- and the *precision* (i.e. the rate of true positives on the predicted positive class), which leads to the *false discovery rate* (FDR).

The *false discovery rate* (FDR) is 1 – precision, so the number of false positive divided by the size of the predicted positive class. In other words, it measures the proportion of wrongly-predicted positive elements among all predicted positive elements. If FDR increases, the number of false positive increase, so the precision decreases. In our analyses, we do not want false positives. We will search to decrease the FDR without decreasing too much the recall. *Controlling the FDR* means that if the user sets a given FDR, then, among the genes in agreement with this FDR, the actual FDR is equal to the given FDR. On the contrary, we call *conservative* the methods giving an actual FDR lower to what we wanted, while *liberal* the methods giving a higher FDR. The FDR control is incorrect in both cases. Indeed, the conservative methods are not "harmful", but they are less reliable. If one wants to have less false positives in a gene list, the best approach is to select a lower FDR, not to use another method. In the section 2.5, we will see how to estimate the FDR.

They will be enough to benchmark predictive tools presented in the following sections.

The first difficulty is obviously that we do not know the positive and negative classes in real datasets. Therefore, to benchmark a given statistical test, we have to either assume which elements are true or false within real datasets, or to simulate artificial datasets. If there are enough replicates, we can evaluate the predictions by assuming the absence of differential expression within-condition (e.g. (Seyednasrollah, Laiho and Elo, 2015; Schurch *et al.*, 2016)). In all cases, we are introducing biases by ignoring complex data structures. However, this is common to any type of modeling in empirical sciences, from statistics to networks. The solution is to stay humble on the results from a given benchmark and rather consider the *tendency* we can observe from a collection of independent benchmarks, or confirm the results with other computational and experimental approaches.

Second, often the tools assume and apply different hypotheses, parameters and methodologies, making the comparison not possible. Even if this is exact, benchmarking can group the tools according to their assumptions to allow general comparisons. Also, the interest of a tool is its *robustness*, i.e. whether it produces satisfying results outside their assumptions and despite slight changes in its parameters. It is critical to identify the limits of this robustness to identify when one cannot predict the behavior of a tool and therefore, when the results are unreliable. Note that "unreliable" does not mean "wrong".

Below, I present few popular tools applied in the context of differential expression analysis. I will particularly insist on the limits that benchmark studies identified.

2.2 Gene filtering

A problem with RNA-sequencing is that the datasets have a high number of genes with zero counts or with a very low number. Consequently, the negative-binomial (NB) distributions (see section 2.4) cannot model well the data, biasing the estimation of parameters by the methods based on NB models. Moreover, when comparing two conditions, the statistical tests will eventually give p-values at 1 for the genes with so few counts. This is why we observe an enrichment in high p-values when we plot the distribution of raw p-values (see section 2.4.4). As the procedures to control the false discovery rate (FDR) depend on the total number of computed tests (see section 2.5), these unnecessary tests also decrease the overall statistical power of the experiment (Bourgon, Gentleman and Huber, 2010). From the theoretical point of view, these two points justify to filter out the genes with counts under a given cutoff. The choice of the filtering strategy raised discussions among the research community.

(Bourgon, Gentleman and Huber, 2010) introduced the *independent filtering*. The idea is to rank the genes and remove the genes below a given rank. It is called *independent* because the filtering does not depend on the p-value computed by the final statistical test. In other words, we do not exclude a gene because it did not give a good p-value. To not loss the FDR control, the filter should also not depend on the conditions, but in overall.

For example, in (Chen, Lun and Smyth, 2016), the authors recommend to keep the genes with a log-counts per million (logCPM) above 0.5 in at least 2 samples if the minimum number of replicates per condition was 2. A logCPM of 0.5 corresponds to 10-15 counts for a library size of around 20 million reads. They implemented this strategy into the function *filterByExpr* of **edgeR**. If one uses Reads Per Kilobase of exon model per Million mapped reads (RPKM) instead of logCPM, (Mortazavi *et al.*, 2008) showed that a RPKM above 1.0 was robust and considered transcripts with more than 30 RPKMs as abundant. In their study, a RPKM of 1.0 corresponded to around 80 reads for a library size of 40M.

(Rau *et al.*, 2013) valuated filtering methods using real and simulated datasets, but they could only compute the precision and the recall with the simulated datasets. They tested 3 filtering methods: the *mean-based filters* (remove genes with an average of counts across the samples below a cutoff), *maximum-based filters* (remove genes with a maximum count across samples below a cutoff, e.g. the logCPM method) and a home-made "*global Jaccard index filter*" (compute a cutoff such as all filtered genes have counts below this cutoff, and all kept genes have counts above this cutoff). They recommended to use the maximum-based filters, in particular the logCPM method, or their home-made filter.

The filtering cutoff can be given by the user or defined by an unsupervised procedure. Using a real dataset with spike-ins and qPCR validations, (Sha, Phan and Wang, 2015) showed that first, a filtering up to 20% of the genes could increase the recall and the

precision of DEGs. Second, the filtering correlated with the number of DEGs. The authors recommended to choose the cutoff by maximizing the number of DEGs. Indeed, rather than fixing an arbitrary cutoff, **DESeq2** (Love, Huber and Anders, 2014) implemented an automatic independent filtering such as the filtering cutoff was the mean counts optimizing the number of DEGs at a given FDR threshold.

However, filtering is not always helpful. (Rigaill *et al.*, 2016) built synthetic datasets by combining one dataset without DEGs (i.e. from replicated samples) and another with many DEGs (i.e. from samples with strong differences). The latter dataset partially contained DEGs validated by qPCR. They then combined the datasets. I will describe more precisely this approach in section 2.4.3. They showed that filtering could increase the FDR control for **GLM-edgeR** and **limma-voom**, by making **limma-voom** more liberal and **GLM-edgeR** more conservative. Yet, it did not increase much the AUC and recall for the methods based on linear models (**GLM-edgeR, limma-voom, DESeq2**), but it could stabilize the scores of limma-voom. Also, they showed that filtering didn't improve computation of raw p-values. They concluded that filtering had little importance compared to other parts of the DEA, but it was interesting to decrease the number of false positives (FDR control) with a constant recall.

For (Lin *et al.*, 2016), the filtering had little effect on the final list of DEGs for the main factors of the experimental design. They compared the results between a filtering before or after normalization methods. They observed the methods were robust for the main factors of the experimental design, but this change can affect the results for the interaction terms, especially for TMM compared to RLE (see section 2.3). They advocated to filter after normalizing the data in order to change the filter cutoff without repeating the normalization.

In conclusion, the filtering can be useful if there are few DEGs or for technologies generating many low counts by improving the FDR control, without necessary increasing the recall. The filtering should be applied after the normalization. It can be either based on a minimal level of counts within few conditions, the sum or the mean of counts across the samples. One can set the threshold by iteratively searching for the value optimizing the number of DEGs, without filtering more than 20% of the genes. It is worth noting that filtering does not bring much if there are many or enough DEGs to explore. In this context, the filtering step can be removed to simplify the workflow.

2.3 Normalizing read counts

Besides the gene expression, the read counts from RNA-sequencing have three main sources of variation: library size (Mortazavi *et al.*, 2008) from the sequencing effects (i.e. total number of counts in one sample), gene length (Oshlack and Wakefield, 2009) as well as GC-content (Zheng, Chung and Zhao, 2011). Indeed, long genes tend to accumulate more reads and GC-rich and poor less reads, all things being equal. And if a gene has more reads, the statistical power related to this gene increases, promoting its identification as a DEG (Oshlack and Wakefield, 2009). To compare the genes and samples to each other, one should remove these sources of variations. This procedure is called *normalization*. To do that, many methods have been proposed, I will present few of them which are popular.

2.3.1 Methods for normalization

Total Count (TC) normalizes the read counts in a given sample by dividing them by the library size. It shifts evenly the count distribution of the sample. This is the simplest method. Doing so, it assumes the sequencing depth is the single bias and that it evenly affects all genes. Therefore, some genes with unexpectedly low or high counts will affect the whole distribution.

(Mortazavi *et al.*, 2008) introduced the **Reads Per Kilobase of exon model per Million mapped reads (RPKM)** normalization. This method normalizes the read counts by the library size as TC, but also by the gene length. A slight variation to RPKM is the **Fragments Per Kilobase of transcript per Million fragments mapped (FPKM)**, developed for the Cufflinks-Cuffdiff pipeline (Trapnell *et al.*, 2010).

With **DESeq** (Anders and Huber, 2010), the authors developed the **Relative Log Expression (RLE)** method. It first divides the counts of a given sample with the geometric mean across samples and then, it computes the median. This gives the normalization factor of the sample.

The Trimmed Mean of M values (TMM) (Robinson and Oshlack, 2010) uses another approach. By making pairwise comparisons between samples, it filters out the genes with the 30% extreme log-fold changes (LFCs) as well as those with the 5% extreme count means. From the pools of remaining genes, it computes the mean of LFCs weighted by their approximated variance. The resulting mean gives the normalization factor of the sample. The normalized counts are often exported as **log-counts per million (log-CPM)**.

2.3.2 Theoretical aspects

As normalization is at the beginning of the differential expression analysis (DEA), any issue at this stage can strongly undermine the prediction of DEGs. One should take into account 3 elements: the library size is not sufficient to normalize the samples, these methods assume the absence of differential expression and if such differential expression happens, it should be symmetrical between up- and down-regulated genes (Evans, Hardin and Stoebel, 2018).

(Robinson and Oshlack, 2010) explained that the read counts of gene depends on the composition of the RNA population. Within one sample relative to another, if some genes tend to monopolize the pool of available reads during the sequencing, the other genes have less chances to get the remaining reads. For example, (Bullard *et al.*, 2010) noted that 5% of genes monopolized 50% of reads in their experiment. This is also true when fragments contaminate the samples. Therefore, methods such as **TMM** and **RLE**, which can account for this effect, have a theoretical advantage. As DEA aims to compare genes *between samples* (conditions), rather than *within samples*, they should not need to consider within-sample effects (e.g. gene length and GC-content). In practice, (Risso *et al.*, 2011) confirmed that GC-content biased the fold-changes and p-values from the DEA, with a positive correlation between DEGs and GC-content. To account for length and GC-content effect, they proposed a normalization method included in the package **EDASeq**. However, to my knowledge, this questioning seems to have been ignored afterwards.

Normalization methods enable comparisons of samples by removing sample-specific effects. To have a correct normalization, the method should not remove any biological effect. However, it cannot itself distinguish between technical and biological effects, so they must not be confounded. Such confounding happens if the cells undergo a general transcriptional shift, for example transcription factors amplifying the transcriptional program within tumors as observed by (Lin *et al.*, 2012). Therefore, the conventional normalization methods presented above assume the *lack-of-variation hypothesis*, i.e. most genes are not differentially expressed (Roca *et al.*, 2017).

Normalization methods correcting the distribution of counts, e.g. **RLE** and **TMM**, allow to break the lack-of-variation hypothesis with moderation, as soon as there is still symmetry between up- and down-regulated genes. In other words, there are robust to a higher proportion of DEGs if there are as many up- as down-regulated genes. The higher expression of up-regulated genes balances the lower expression of down-regulated genes. In case of asymmetry, the assumption cannot hold anymore and the normalization will eventually fail.

Furthermore, even with a perfect normalization, the RNA-sequencing still measures relative abundances of transcripts. A most fundamental assumption is that each cell has the same absolute amount of RNAs. If the cells have twice more RNAs within a condition than within another, this difference cannot be observed after sequencing. Indeed, the same number of RNAs is sequenced in both condition, giving the same proportion and thereby, the same fold changes. In this case, there is a high proportion of DEGs and a strong asymmetry. If the DEG population had equivalent up- and down-regulated genes, they would counterbalance each other and the proportion of transcripts could still be comparable. The lack-of-variation hypothesis would not hold, but we could still normalize by searching for the non-DE genes among the population. This observation encouraged (Lovén *et al.*, 2012) to include *spike-ins*, i.e. known concentration of artificial RNAs, within the samples before RNA extraction. They counted the cells and added the spike-ins in proportion. Thus, they could normalize the samples according to the spike-ins. (Chen *et al.*, 2016) also advocated for a general use of spike-ins and gave advises to do so.

I would suggest that, when working within a new transcriptional context, one should first test whether the differential expression analysis would be possible with standard sequencing workflows. To have complementary details, (Evans, Hardin and Stoebel, 2018) reviewed the normalization methods and categorized them by their assumptions.

2.3.3 Benchmarks

Using real datasets with DEGs confirmed by qPCR, (Bullard *et al.*, 2010) showed the normalization methods had more effects on the DEGs than the statistical tests. Also, the authors observed highly expressed genes (i.e. outliers) strongly biased the **RPKMs**, while this method could not fully correct the read counts for the gene length. (Zheng, Chung and Zhao, 2011) confirmed the latter observation. (Dillies *et al.*, 2013) used diverse real and simulated datasets to compare the aforementioned normalization methods. They observed both **TMM** and **RLE** had similar and robust results, while

RPKM and **TC** were sensitive to outliers and produced samples with more variability. They concluded that **RPKM** and **TC** should be "definitely abounded in the context of differential expression analysis". This conclusion was in agreement with (Maza *et al.*, 2013; Lin *et al.*, 2016). Between **RLE** and **TMM**, no method outperforms the other (Dillies *et al.*, 2013; Maza *et al.*, 2013; Rapaport *et al.*, 2013; Seyednasrollah, Laiho and Elo, 2015), even if (Zyprych-Walczak *et al.*, 2015; Lin *et al.*, 2016) had a slight preference for **RLE**. On the contrary, (Li *et al.*, 2015) correlated the normalized data with qPCR data and observed that **RLE** or **TMM** did not increase the correlations, while **RPKM** did, but they did not push more their study. It was the only paper to make this conclusion, against theoretical and practical aspects we saw above.

(Robinson and Oshlack, 2010) claimed that their **TMM** method was robust up to 30% of genes differentially expressed in the same direction. However, to my knowledge, only one study evaluated the effects of the proportion of DEGs and the asymmetry between up- and down-regulated genes within a condition. (Soneson and Delorenzi, 2013) compared DEA tools by using simulated datasets from a negative-binomial (NB) model. The datasets had 0, 10 or 30% of DEGs with an asymmetry at 0% (i.e. as many up- as down-regulated genes) or 100% (i.e. only up-regulated genes). Even if it was not possible from their experiment to distinguish between the effects of the normalization and the statistical modeling, they observed that 10% of DEGs slightly decreased the AUC (area under the ROC curve), while 30% DEGs dramatically decreased it. A full asymmetry between up-and down-regulated genes enhanced this effect and increased the FDR, especially at higher numbers of replicates and DEG proportion. Interestingly, the authors noted that replication more important to increase the statistical power than the symmetry in the DEG population. (Rigaill *et al.*, 2016) had opposite results to (Soneson and Delorenzi, 2013) when considering the proportion of DEGs. However, they did not evaluate the effect of the normalization methods and they did not test the asymmetry within the DEG population. It is therefore not possible to conclude.

While acceptable in most cases, the lack-of-variation hypothesis does not hold for some datasets which require another normalization approach. For example, (Roca *et al.*, 2017) developed two methods, called **SVCD** and **MedianCD**, to normalize the counts without assuming the lack-of-variation hypothesis. Their methods first compute within-condition normalization factors. Then, iteratively, they apply a statistical test to identify the non-DE genes by comparing the average of each condition. At each iteration, they removed the genes showing the highest variations until convergence. They can finally compute the final normalization factors from the pool of non-DE genes. Interestingly, the methods do not assume any count distribution (e.g. Poisson or Negative-Binomial). The authors claimed that their approach can work for any proportion of DEGs within the gene population, as soon as there are enough genes from which we can estimate the normalization factors. Current normalization methods cannot work when the lack-of-variation hypothesis does not hold and so, they cannot estimate the actual proportion of DEGs. Standard normalization methods already gave a high proportion of DEGs with an asymmetry in favor of down-regulated genes, consistently with the literature. Therefore, I applied **SVCD** to normalize the counts.

More benchmarking should be done to test the relevancy of **SVCD** and **MedianCD**. Roca and colleagues' normalization enters in the category of "normalization by testing" described by (Evans, Hardin and Stoebel, 2018). As for **TMM** and **RLE**, it still assumes that normalization factors for non-DE genes can apply to the DE genes and the proportion of counts is not distorted.

To conclude, normalization aims to remove technical aspects of the sequencing. The simplest and common approaches are **TC** and **RPKM**. They apply library-size normalization. The literature strongly advised against their use because they are easily biased. The most popular and robust approaches are **TMM** developed for **edgeR** and **RLE** developed for **DESeq**. They normalized by adjusting the count distribution. They produce similar results but some studies have a slight preference for **RLE**. Normalization works well in general, but unexpected conditions can break the assumptions used to distinguish between technical and biological effects. Therefore, the user needs to be sure that these assumptions hold before to start the differential expression analysis. The normalization methods produce normalization factors used by the tools of the DEA. They do not directly update the read counts. Indeed, the statistical tools of the DEA need to estimate some parameters from the raw counts, otherwise the normalized counts would bias the estimation. We will find the normalized counts during the downstream analyses (see section 3).

2.4 Modeling differential expression

In this section, we will look at the core part of the differential expression analysis. Here, the tools compute a statistical inference for each gene by comparing conditions. From the raw counts and the normalization factors, they return a size effect (i.e. log-fold change) and a significance (i.e. p-value) for each gene. We will use these values in the downstream analyses to explore the list of differentially expressed genes. In the sections 2.4.1 and 2.4.2, we see the tools for the DEA and the concepts behind them. Then, we review the literature to find the presumed best tools (section 2.4.3). We quickly finish by seeing 3 statistics to consider before to move to the next step (section 2.4.4).

2.4.1 The story of edgeR and its subsequent developments

I will present the story of **edgeR** to reach 3 objectives. First, this is a popular tool and it allows us to illustrates the concepts of the differential expression analysis. Second, Smyth and colleagues successively improved their tool to overcome issues, mainly to implement complex experimental design, increase the FDR control and increase the robustness against outliers. With these changes, **edgeR** has become a complicated statistical machinery. Such changes were of course well motivated, as confirmed by the numerous benchmarks (see section 2.4.3). However, this is also interesting because as a user, we mostly see few line codes to run the tools and we can quickly forget what it is behind. After discussing with few biostatisticians, it appeared that the DEA tools still have surprising behaviors for some datasets or use cases. This section will illustrate that even if these tools are practically straightforward to use, their machinery is not.

To complete the following story provided by the canonical papers presented below, I used the reviews (Chen, Lun and Smyth, 2014, 2016; Lun, Chen and Smyth, 2016).

2.4.1.1 Classic edgeR implemented an overdispersed negative binomial model.

(Robinson and Smyth, 2007, 2008; Robinson, McCarthy and Smyth, 2010) introduced the negative binomial (NB) models for RNA-sequencing, which are now used by the most popular approaches for the DEA. Note that the 2008 paper precedes the 2007 paper. I will synthetize below how this modeling works, as implemented in the first version of edgeR published in 2010 (called ***classic edgeR***), in order to understand the context of the statistical terms that one could read across the papers.

Because we have biological replicates, the read counts of one gene in a given condition has a *mean* and a *variance*. If we assume the sequencing has the same chance to independently detect each RNA fragment, then the read counts of each gene should follow a *Poisson distribution*. This distribution implies that the mean is equal to the variance, which is not observed in RNA-sequencing. On the contrary, the *NB distribution* enables to define another *mean-variance relationship*. For a gene i in a given sample j, the gene variance $\sigma_{i,j}^2$ depends on the gene mean $\mu_{i,j}$ as well as a *common dispersion parameter* ϕ: $\sigma_{i,j}^2 = \mu_{i,j} + \mu_{i,j}^2 * \phi$. Thereby, the dispersion is the same for all genes in order to share information across them. Computing a gene-specific dispersion is not justified and, anyway, not possible with a small number of replicates. The range of values of the dispersion parameter (strictly higher than 0) is such as the model is *overdispersed*. Thus, the reads are only modeled by a gene mean and a common dispersion. The gene mean is itself the product of the *offset* of the sample, i.e. the sequencing effects (sequencing depth and library size) that the sample underwent, and the *true relative abundance* of the gene. *Testing the differential expression* means testing the null hypothesis that a gene has the same abundance between two conditions. At the end, the statistical test computes a *p-value* giving the probability that the difference of abundances between the two conditions could happen under the null hypothesis, i.e. *by chance*.

Consequently, the statistical method must estimate the mean and the dispersion. It estimates the common dispersion parameter, along with the true relative abundance, by computing a *likelihood function* from the data and then finding the dispersion maximizing this likelihood. Without going into details, *classic edgeR* implements a *quantile-adjusted conditional maximum likelihood* (qCML). With simulated datasets, they showed that the qCML decreased false predictions, but it only works for experimental designs with a single factor.

Yet, the authors recognized that the dispersion is not always the same for each gene and the outliers (i.e. genes with large variance) could bias the dispersion. Therefore, in a second step, they computed a gene-specific dispersion $\phi_{i,j}$ to replace ϕ by drawing a *weighted likelihood function* for the gene, but such as it "squeezed" (or "shrank") the dispersion toward the common dispersion parameter. The "squeezing" also depends on a weight parameter that the weighted likelihood function cannot estimate. However, this method is close to another based on an *empirical Bayesian model*, which can estimate the squeezing weight. The authors did not directly apply this model

because it does not manage well very different variances. They estimated the squeezing weight from the empirical Bayesian model at the point where the latter had the same results as the weighted likelihood function. It enables to estimate the dispersion parameters without too strong or too weak correction. That is why the dispersion estimation was said to be *moderated*.

Finally, a statistical test can compare the conditions using the estimated parameters. In short, the test can be asymptotic (e.g. *Wald's test* or *likelihood ratio test* (LRT)), but, as the replication is low, the authors developed their own *exact test*, similar to the Fisher's exact test, which controls better the FDR.

In brief, the main advantage of this method compared to the previous ones, is that it can account for sequencing effects and within-condition variability from the biological replicates. However, this **classic edgeR** could only manage simple experimental designs with a single factor. The authors still recommend to use **classic edgeR** over **GLM-edgeR** when possible.

2.4.1.2 GLM-edgeR can manage multifactor experimental designs.

(McCarthy, Chen and Smyth, 2012) implemented the *generalized linear models* (GLM) and the *likelihood ratio tests* (LRT) to **edgeR**. As we saw, **classic edgeR** cannot manage experimental designs with multiple factors, e.g. 2 yeast strains (r, s) grown on 2 mediums (A, B). Here, there are 2 factors with 2 levels, so 4 conditions. To solve this issue, the authors extended the simple NB models with *non-linear models* called *generalized linear models* (GLMs). The first step is to write the equation of the experimental design. In linear modeling, the total effect y is the sum of the effect of each factor, here strain and medium. The model can also include interaction terms describing non-additive effects. If we want to consider the combined effect of strain and medium, then the interaction is written as the product of both factors. The final equation of the experimental design can be $\text{effect} \sim \text{strain} + \text{medium} + \text{strain} * \text{medium}$. Adding more factors and interaction terms enables to model more precisely the system, but it decreases the statistical power and this choice must be biologically relevant. Also, it is not possible to solve the linear model if there are more factors and interaction terms than the number of conditions.

Finally, to compare two or more factors, we define our hypothesis by modeling it with a *contrast* and the statistical test will compute the p-value for each gene. For a given gene, the hypothesis can be whether the effect in one condition is equal to the effect in another. GLMs enable more complex hypotheses, such as whether the differences of two conditions are equal to the differences of two other conditions.

For RNA-sequencing, the equation of the experimental design means that, for one condition i, the number of counts y_i of a gene is the sum of a series of term. Each term j is the product of a *gene-specific regression coefficient* β_j and a *condition-specific predictor* x_{ij}. The β coefficient represents the number of counts brought by one level of one factor. The predictor is equal to 1 if the level of the factor contributed to the condition, 0 otherwise. This is the same principle for the interaction terms. In our example, for a given gene from the condition (r, A), we will have:

$$y_i = \text{strain}_i + \text{medium}_i + \text{strain}_i * \text{medium}_i$$
$$= (x_{ir}\beta_r + x_{is}\beta_s) + (x_{iA}\beta_A + x_{iB}\beta_B) + x_{is}x_{iB}\beta_{sB}$$
$$\Rightarrow \boldsymbol{y_{rA}} = (1 * \beta_r + 0 * \beta_s) + (1 * \beta_A + 0 * \beta_B) + 0 * \beta_{sB} = \boldsymbol{\beta_r} + \boldsymbol{\beta_A}$$

While possible, this approach can lead to some statistical issues or prevent to use some tools such as the ANOVA-like test of **edgeR**. Usually, one condition serves as a reference, averaging the expression across the condition, to which the other coefficients will add their own effect. The coefficient of the reference is β_0 and its predictor is always 1. Thus, there is no need to add the coefficients related to the reference condition, as β_0 already includes their effect. The example above gives:

$$y_i = \beta_0 + x_{is}\beta_s + x_{iB}\beta_B + x_{is}x_{iB}\beta_{sB}$$
$$\Rightarrow \boldsymbol{y_{rA}} = \beta_0 + 0 * \beta_s + 0 * \beta_B + 0 * \beta_{sB} = \boldsymbol{\beta_0}$$

We can repeat the process for each condition and add all the predictors within a *design matrix*, where the rows are the conditions, the columns are the coefficients and the cells are the predictors. The intercept, a column of 1, represents the reference condition.

Using the matrix, is it straightforward to compute the contrast modeling the desired hypothesis. For example, the null hypothesis $y_{rA} = y_{sA}$ is equivalent to $\beta_0 = \beta_0 + \beta_s$, so we want to test that $+1 * \beta_s = 0$, as the condition (r, A) is the reference. We obtain this contrast by subtracting the two related rows of the matrix.

j	β_0	β_s	β_B	β_{sB}
r, A	1	0	0	0
s, A	1	1	0	0
r, B	1	0	1	0
s, B	1	1	1	1
Contrast for H0: $(s, A) - (r, A) = 0$	0	1	0	0

Note that in the case of RNA-sequencing, the model is log-linear, i.e. the total effect and the coefficients are in log2-scale. Also, the offset of the condition is added to each y_i.

The question is how **edgeR** uses this modeling. Compared to the **classic-edgeR**, the authors changed the estimation approach to account for complex designs in ***GLM-edgeR***, but it is still similar. The approach is an iterative algorithm to fit the GLM defined by the design matrix using a *Cox-Reid profile-adjusted likelihood* function. Assuming the dispersion parameters, a method similar to the *Newton-Raphson algorithm* estimates the regression coefficients. Maximizing the likelihood function gives the gene-specific dispersion parameters, as well as the regression coefficients. At each round of the fitting, it estimates both and then, the approach squeezes the dispersion parameters toward a trended dispersion, computed by a *locally weighted profile-adjusted likelihood*. The issue is how to estimate the squeezing weight. At this

time, the amount of squeezing was a general constant. In the further developments, ***quasi-edgeR*** will apply another approach (see below). The latter tool will require the common dispersion computed in parallel by the profile-adjusted likelihood.

(Zhou, Lindsay and Robinson, 2014) completed this approach to account for the outliers by weighting the first likelihood function with the difference between the observations and the model.

Finally, to test the hypotheses in this first version of GLM-edgeR, they applied a *likelihood ratio test* (LRT), that they though to correctly approximate the exact test of the **classic-edgeR**.

They also introduced the *biological coefficient of variation*, which is the square-root of the gene-specific dispersion parameter. The coefficient includes both technical and biological variability. Yet, for highly-expressed genes in RNA-sequencing, the technical variability is low so the biological variability should dominate and stabilize. (POV) In my opinion, the average biological coefficient of variation also enables us to estimate whether we can expect a good power from the DEA. More the coefficient is high, more the experiment should have replicates to ensure the quality of the DEA.

2.4.1.3 Quasi-edgeR increases the test efficiency by accounting for the variance uncertainties and outlier genes.

(Lund *et al.*, 2012) extended **GLM-edgeR** by developing a *quasi-likelihood* (QL) framework for the statistical test of the differential expression. Its main advantage is to compute the uncertainties of the estimated variances and account them for the statistical test. The latter is thus more robust to errors in the modeling of the experimental design.

Quasi-edgeR follows and uses the results of the dispersion estimation of **GLM-edgeR**. It enables to set the squeezing weight more precisely by estimating the *prior degree of freedom* (number of independent variables). A low gene variability between conditions induce a high degree and so, a strong squeezing. Moreover, the amount of squeezing does not take into account genes with null counts, as they cannot contribute to the estimations. This extension is currently implemented within the function $glmQLFit$ of **edgeR**, which replaces $glmFit$.

To estimate the prior degree of freedom, we will estimate gene-specific QL dispersions. **Quasi-edgeR** models the variance of counts by a function depending on the gene mean, the common dispersion of the NB model and the unknown gene-specific QL dispersions. It estimates the QL means by maximizing a quasi-likelihood function based on this variance function. From these results, **quasi-edgeR** computes estimators for the QL dispersions and the prior degrees of freedom. It squeezes the QL dispersions toward the trend using the empirical Bayes method described by (Smyth, 2004), with the prior degree of freedom to set the squeezing weight.

Finally, for a given contrast, the function $glmQLFTest$ of **edgeR** computes a QL test statistic from the QL dispersions, the QL means and the estimated degrees of freedom. To compute the p-value, this statistic is compared to an F-distribution, similarly to the F-tests of standard ANOVA. The QL test is similar to the Fisher's test for a reduced

number of replicates as they are more robust and conservative. On the contrary, the LRT statistics is asymptotic, i.e. the number of replicates should then toward the infinite.

(Phipson *et al.*, 2016) added an option to the **edgeR** QL fitting function to increase the robustness against outliers (i.e. highly-variable genes). It increases the squeezing for the main body of genes, while decreasing it for the outliers, by estimating a gene-specific degree of freedom. Also, it decreases the degree of freedom for genes with null counts to reduce their squeezing. Without robustness, one could call differentially expressed the genes with very high or low dispersions across the samples. Indeed, they showed the robust QL increased the power and the FDR control in presence of outliers within simulated datasets.

The story of **edgeR** showed successive modifications to overcome the fact that count distributions from RNA-sequencing do not match well NB models. On the contrary, we will see below that **limma-voom** makes fewer assumptions, while it has similar efficiencies to **edgeR**. It will be interesting to see in the next years whether the Gaussian linear models will replace the other approaches based on NB models.

2.4.2 Few other tools for DEA

2.4.2.1 Limma-voom

Limma fits Gaussian linear model (Smyth, 2004) by following the sample principle as quasi-edgeR. It uses an empirical Bayes approach to estimate the moderated dispersion parameters and regression coefficients. From these estimators, the approach can then compute a moderated t-statistics following a t-distribution under the null hypothesis. It is said moderated because under-estimated variances do not affect its value. By comparing the t-statistics to the t-distribution, we obtain the p-value. The t-statistics are linked to the F-statistics, which are used for complex designs (more than 2 factors). **Quasi-edgeR** uses the functions of **limma** that (Phipson *et al.*, 2016) extended to increase the robustness of the approaches.

Even if **edgeR** enables generalized linear models, it makes statistical assumptions limiting its range of applicability. In general, **limma** can be applied to large-scale dataset to enables any possible experimental design (e.g. temporal effects, inter-gene dependencies, sample correlations, customized weighting of samples or genes), use techniques originally applied to microarrays and speed up the computation, while **quasi-edgeR** would be adapted for low-count datasets with few biological replicates. As we will see in section 2.4.3, they give similar results.

(Law *et al.*, 2014) introduced **voom** to RNA-sequencing by transforming the data used by **limma**. Indeed, **limma** cannot process *heteroscedastic* data, i.e. data with a variance depending on the mean. The logarithmic transformation corrects that for the genes with large counts, but not for those with small counts. **Voom** solves this issue by estimating the mean-variance relationship with a nonparametric model. Then, it communicates this information to **limma** through weights for each count value, which can remove the heteroscedasticity to produce *homoscedastic* data. **Voom** can use the normalized counts from **TMM** (Oshlack and Wakefield, 2009). To down-weight samples with a large variability, rather than removing them from the datasets, (R. Liu

et al., 2015) used the results from this first round to estimate the sample-quality weights. Then, **voom** re-computed the mean-variance relationship using these weights in a second round.

2.4.2.2 DESeq and DESeq2

DESeq (Anders and Huber, 2010) applies the same principles as **classic-edgeR** but it uses another approach. First, the authors introduced the **RLE** normalization method. Second, **DESeq** does not estimate a dispersion parameter. It also models the gene-specific variance as $\sigma_{i,j}^2 = \mu_{i,j} + f_{i,j}$. $f_{i,j}$ has the same function as the term $\mu_{i,j}^2 * \phi_{i,j}$ for **classic-edgeR**, but it is a *smooth function* computed from the abundance of counts and the mean-variance relationship. It is simpler than using a likelihood function to estimate the dispersion. Then, the authors applied an exact test similar to **classic-edgeR**. Later, (Love, Huber and Anders, 2014) proposed **DESeq2** to implement GLM and to have another method to share information across genes, as **DESeq** overestimated the variances. To do so, it estimates the gene-specific dispersions and squeezes them toward the trend defined by the smooth function using an empirical Bayes approach. In fact, this is a similar procedure to **GLM-edgeR** with a Cox-Reid profile-adjusted likelihood. Another empirical Bayes approach moderates the log-fold changes (LFC) to remove the exaggerated LFC of lowly expressed genes. **DESeq2** also removes the lowest expressed genes with independent filtering removes. Finally, a Wald's test tests for the differential expression using the squeezed the moderated LFCs and other parameters estimated by the tool.

2.4.2.3 Cuffdiff2

(Trapnell *et al.*, 2010, 2012) developed the **Cufflinks-Cuffdiff** pipeline. **Cuffdiff** estimates the transcript abundances from the assembled transcripts of **Cufflinks** and implements the differential expression analysis. Contrary to other classical DEA tools, both tools cannot be independently used. Later, (Trapnell *et al.*, 2013) updated **Cuffdiff** to **Cuffdiff2**. Using the **RLE** method, **Cuffdiff2** normalizes independently each condition first, and then normalizes the samples by considering all samples this time. It applies another distribution law to model the counts, the *beta NB model*. An advantage of **Cufflinks-Cuffdiff** is to directly test the differential expression at the transcript level, even if it can return results at the gene level. On the contrary, increasing the number of replicates does not benefit **Cuffdiff**.

2.4.3 Benchmarks

The DEA tools applied various approaches to get differentially expressed genes. They are highly competitive because one tool can often execute all the steps from the raw counts to the statistical inference. Thus, many papers evaluated the efficiency of DEA tools using real, simulated or synthetic datasets. *Real datasets* enable to evaluate them by comparing the lists of DEGs and by computing the prediction errors if qPCR validations are available (e.g. (Rajkumar *et al.*, 2015)). If there is no qPCR validation, one can estimate prediction errors from the biological replicates or by resampling them (i.e. bootstrap method, e.g. (Schurch *et al.*, 2016)). The experiment should have a high number of biological replicates to do that. Most of the time, the authors generated *simulated datasets* by simulating the count distribution from a model (e.g. negative-binomial) with the parameters from real datasets (e.g. (Soneson and Delorenzi, 2013)).

Yet, (Hawinkel *et al.*, 2020) explained that simulated datasets from NB models favor DEA tools based on NB models. This is called a *circular reasoning*. Instead, *synthetic datasets* aimed to overcome the limits of each approach by mixing two datasets (e.g. (Rigaill *et al.*, 2016)). For example, one dataset has a high chance to contain DEGs, the other a low chance. We can them assume that the former contains true positives and the latter true negatives.

Following the publication time, we will see a series of benchmark papers. It gives us the tendencies related to the continuous developments of the methods. I won't present the DEA tools that are not aforementioned, because they are either not popular or not efficient. If the reader is not interesting in the results from the benchmark approaches, it is possible to jump to the conclusive paragraph. Note that some benchmarking papers referring to **GLM-edgeR** apply the approach of the **quasi-edgeR** pipeline, as presented by (Chen, Lun and Smyth, 2014). Therefore, I will replace their naming of edgeR versions by the one I presented above. It avoids the ambiguity between the exact test of **classic-edgeR**, the LRT of **GLM-edgeR** and the QL-F test of **quasi-edgeR**. If not indicated, they do not use the robust feature recommended in the recent versions of **edgeR** developed by (Phipson *et al.*, 2016). Some researchers recommend to give the version number of the tools. Yet, for edgeR, this is not sufficient because the authors ensured the retro-compatibility by implementing the new features into new functions or parameters.

(Soneson and Delorenzi, 2013) benchmarked 11 approaches whose **classic-edgeR**, **DESeq** and **limma-voom** using *simulated data* from NB distributions. If the number of biological replicates increases from 2 to 5, then classic-edgeR controls better the false discovery rate (FDR) such the true FDR is close to the selected FDR (0.05). In all cases, the replication is still the most important factor to improve the FDR control. For 2 replicates, the best AUC among the tested tools was with **classic-edgeR**, **DESeq** and **limma-voom**. See the section 2.1 for a definition of the AUC. The authors observed that DESeq was too conservative, while classic-edgeR was not robust against outliers. Both tools were less efficient to detect true differential expression when the number of outliers increased. The authors recommended **limma-based** methods for the datasets with more than 3 biological replicates.

(Rajkumar *et al.*, 2015) tested **Cuffdiff2**, **GLM-edgeR** and **DESeq2**, as well as another tool, using *real datasets* comparing 2 conditions with 8 biological replicates per condition. They verified by qPCR the DEGs within the overlap between the DEG lists of the tools. They observed that **GLM-edgeR** had good recall and precision. On the contrary, **DESeq2** had a low recall and **Cufflink2** a low specificity. The authors preferred **GLM-edgeR** in the end.

(Seyednasrollah, Laiho and Elo, 2015) used 2 *real datasets* of 2 conditions with 10 to 28 replicates in each condition. They estimated the FDR by randomly splitting the replicates of a same condition into two artificial groups of uneven size. Thereby, comparing the two groups should result in a uniform distribution of p-values, indicating the absence of differential expression. With 8 other tools, they benchmarked **classic-edgeR**, **DESeq**, **Cuffdiff2** and **limma-voom.** They did not include **DESeq2**

because it increased the false positives compared to **DESeq**. They sorted the tools from the best to worst recall: **classic-edgeR**, **limma-voom**, **DESeq** and then, **Cuffdiff2**. However, edgeR had the highest number of false positives among these 4 tools. They observed that increasing replicates did not increase the recall of **Cuffdiff2**. **Cuffdiff2** had also the lowest precision, while **limma-voom** and **DESeq** had the highest precision. The authors concluded that **limma-voom** and **Cuffdiff2** were the most and less satisfying, respectively.

(Rigaill *et al.*, 2016) had an interesting approach. They generated a *synthetic dataset* by mixing two real datasets. The first dataset compared biological replicates of plant leaves, so it should not have any DEG. The second came from flowers buds and was compared to the plant leaves. This comparison should result in many DEGs. The authors partially validated these DEGs by qPCR. To generate the synthetic dataset, they randomly selected the non-DE genes of the first dataset and the DE genes of the second, such as this list should contain the genes validated by qPCR. They used the synthetic dataset to compare **DESeq**, **DESeq2**, **classic-edgeR**, **GLM-edgeR** and **limma-voom**. They first observed that all methods generated a non-uniform distribution of raw p-values when there were no DEGs. It implied that these tools did not properly model the counts. Nonetheless, GLM (i.e. **GLM-edgeR, DESeq2**) or linear models (i.e. **limma-voom**) can better compute raw p-value distributions, indicating the models can fit the data. These 3 methods also had a higher AUC and recall as well as better a FDR control than the tools using simple NB models (**DESeq, classic-edgeR**). Simple NB models as well as **DESeq2** were too conservative, while **GLM-edgeR** control well the FDR (i.e. expected FDR is the actual FDR). FDR control of **limma-voom** was more variable, but robust to a high proportion of DEGs and a filtering could stabilize it. The authors concluded that **GLM-edgeR**, **limma-voom** and **DESeq2** were the most satisfying with a slight preference for the **GLM-edgeR** and **limma-voom**. The 3 tools enable to take into account all experimental factors into the model and thus, they give good predictions by better estimating the mean and the variance.

(Schurch *et al.*, 2016) used a *real dataset*. They compared a wild-type and a mutant strains of the yeast, with around 42 final replicates per condition. They estimated the number of correct or wrong DEG predictions with bootstrapping. They applied 11 tools, whose **DESeq**, **DESeq2**, **exact-edgeR**, **GLM-edgeR**, **Cuffdiff2** and **limma**. They observed that **DESeq**, **DESeq2**, **limma** and both **edgeR** had the highest recall with a good specificity for any LFC cutoff and number of replicates. **DESeq2**, **limma** and **classic-edgeR** gave similar results, indicating a high efficiency. On the contrary, **GLM-edgeR** gave distinct results from them. In general, the authors recommended to use **classic-edgeR** or **DESeq2** for the experiments with less than 12 replicates. Otherwise, one should use **DESeq**.

(Holik *et al.*, 2017) prepared libraries by making technical replicates to which they added variability by degrading the samples. It enabled them to compare two conditions in which they expected to observe a given list of DEGs. They compared **classic-edgeR, GLM-edgeR, DESeq2** as well as **limma-voom** with or without the improvement of (R. Liu *et al.*, 2015)**.** They observed that the improved **limma-voom**

was the most efficient for its recall and specificity, but it was closely followed by the aforementioned other tools.

(Lamarre *et al.*, 2018) used a real dataset from which they estimated the true DEGs with a FDR lower than 0.1%. They compared **DESeq**, **DESeq2**, **classic-edgeR** and **GLM-edgeR**. For less than 5 replicates, the AUC ranking of the tool efficiency by descending order was **DESeq**, **DESeq2**, **classic-edgeR** and **GLM-edgeR**. However, the authors did not prefer one tool to another.

Finally, (Li *et al.*, 2020) reported the most contradictory results. They wanted to benchmark their new normalization method against **RLE** and **TMM** using either **classic-edgeR**, **quasi-edgeR**, **DESeq2** or **limma-voom**. I usually avoid to analyze results from benchmarks dedicated to prove that the author's tool is the best. Yet, this is a rare study comparing **quasi-edgeR** to the former tools. They used benchmark datasets developed for microarrays where the true and false positives are defined. They observed that the **classic-edgeR** detected the highest number of true positives with the lowest false positives. In comparison, **DESeq2** had less true positives and **quasi-edgeR** had more false positives. **Limma-voom** had both drawbacks. They also compared biological replicates from cancer datasets to estimate the number of false positives. In this context, **quasi-edgeR** tended to be more conservative than the **exact test**. Strangely, they observed that **classic-** and **quasi-edgeR** returned more false positives with a higher replication. It was not the case with **DESeq2** and this effect should not be observable for less than 10 replicates. The authors recommended **quasi-edgeR** for small replication. Yet, this study seems to have contradictory conclusions and to disagree with the literature; one should therefore remain cautious.

To conclude, in overall, the popular **edgeR**, **DESeq2** and **limma-voom** gave similar results. **DESeq2**, and mostly **DESeq**, tend to be more conservative than expected, but some papers still recommend them. It is not clear whether **quasi-edgeR** or **GLM-edgeR** are more efficient than **classic-edgeR**, but experiments with multiple factors constrain to use the formers. On the contrary, **Cuffdiff2** has a low efficiency and it should be avoided.

2.4.4 Quality control of the DEA

DEA is sensible to the experimental design and, in particular, to the confounding factors. The user should check them before to process the DEA results with downstream methods. (Lun, Chen and Smyth, 2016) advise to plot two statistics. First, the function *plotMDS* of **edgeR**. It clusters the samples by computing the root-mean square of the genes with the highest log-fold changes. The replicates should cluster together because we expect the within-condition variance to be lower than the between-condition variance. If the samples cluster together according to an undefined factor, then it could bias the DEA. Second, the function *plotBCV*. If the trend of the biological coefficient of variation does not stabilize for high counts, it may indicate the data underwent some confounding factors.

Furthermore, the user can check the distribution of raw p-values. It should be the sum of the distribution of p-values under the null hypothesis (no differential expression) and the distribution under the alternative hypothesis (differential expression). The

former follows the uniform distribution ("*flat-shape*"). It can eventually have an enrichment in p-values at 1 because of null counts ("*U-shape*"). On the contrary, the distribution under the alternative hypothesis is enriched in low p-values. Other types of distributions indicate that the DEA did not correctly model the experiment.

To solve this issue, one can remove the faulty samples or assign them a dedicated factor to the experimental design until the biological coefficient of variation and the p-value distributions are acceptable. It will decrease the statistical power, but the results would not be reliable otherwise. However, one should try to simplify the experimental design. If the statistical test gives too few significant p-values for a factor or an interaction term, the latter can be removed to simplify the experimental design. Moreover, if a condition is not used in the downstream analysis, one can ignore it during the DEA to improve the power.

2.5 Correcting the multiple-hypothesis testing problem

Differential expression analysis aims to test thousands of genes with significant changes. If the significance level is set at 5%, then the genes with a p-value lower than this threshold pass the test. There is reciprocally a probability of 5% that the differentially expressed genes are false positives. Few false positives would be acceptable however 5% of thousand genes would yield as many as hundreds of false positives under the null hypothesis (Bender and Lange, 2001). This is called the *multiple-hypothesis testing problem*. We correct this problem by controlling the number of false positives among the gene population for a given significance level, at the expense of false negatives. It is not possible to control for both errors, so we prefer to control the false positives which can lead to spurious conclusions.

From raw p-values of the statistical tests, the controlling procedures produce *adjusted p-values* to select the DEGs according to the threshold of the significance level (Wright, 1992). The adjusted p-value is a *random variable*, not a probability. *Adjusting the p-value* of a test means that we define a new value which accounts for the test significance with regard to the whole set of tested hypotheses. If this value is under a given significance level, then we reject the null hypothesis (no differential expression) for the test. It gives the results of the statistical inference for any significance level we choose in advance. For example, if the adjusted p-value is at 0.04, then we reject the null hypothesis at a significance level of 5%, but we do not at 1%. These values are not *absolute*; they can vary depending on the assumptions made. One's expertise can serve to assess whether the null hypothesis could be rejected even if the adjusted p-value is above the cutoff (Westfall, 2011).

A first aspect is to choose the tests to simultaneously adjust in the same *family* (Shaffer, 1995; Bender and Lange, 2001). We usually compare conditions multiple times by testing thousands of genes each time. A rigorous way would be to take all tests and control the proportion of false negatives. Yet, as we cannot decrease both the false positives and negatives, we could be too conservative. In practice, researchers tend to consider each comparison as a family and apply the controlling procedure independently for each of them. **DESeq2** does that by default and (Bender and Lange, 2001) recommended that for exploratory experiments which do not aim to prove a

defined hypothesis. Such choice should be explicit. Of course, choosing one of both approaches should depend on the final goal of the experiment. For example, if we want to have a list of DEGs where a gene is differentially expressed in at least one comparison, then we should apply the controlling procedure on the tests from all the used comparisons. We can observe that this case can be avoided by testing the appropriate hypothesis with a generalized linear model.

Without doing a comprehensive review of this field, several procedures have been proposed to correct the multiple testing problem (Goeman and Solari, 2014). An initial set of procedures aimed to compute the cumulated number of false positives within a set of tested hypotheses. For example, the *Bonferroni procedure* and its direct extension, the *Holm procedure* (Holm, 1979). Both procedures compute the *familywise error rate* (FWER), which is the probability to have at least one false positive within the DEGs. In the Bonferroni procedure, the adjusted p-values are the raw p-values multiplied by the number of tests. With a significance of 0.05 for 10,000 genes, we will have at most 500 false positives. The Holm procedure does the same, but by rejecting one by one the hypotheses from the lowest to the highest p-values. Holm proved his procedure has the same control on the FWER as Bonferroni, but with a higher power. The main drawback of these methods is that they are not well adapted when there are many tests.

On the contrary, the *Benjamini-Hochberg (BH) procedure* controls the *false discovery rate* (FDR), i.e. the expected proportion of false positives among the DEGs (Benjamini and Hochberg, 1995). This method and its derived version gives an upper bound on FDR. This is the most common approach in RNA-sequencing. Contrary to the Bonferroni and Holm procedures, it is much less conservative, but it assumes independent p-values, which is not the case in genomics and transcriptomics. The BH procedure is more adapted when many null hypotheses are rejected. Yet, too many hypotheses (genes) increase the FDR-control burden. The control procedure increases the FDR of each test because more tests induce a higher probability to have false positives. Therefore, this burden decreases the statistical power. We saw in section 2.2 that a first way is to remove the lowly expressed genes without enough counts to predict them as differentially expressed. In the same idea, using isoforms instead genes can only decrease the power (Goeman and Solari, 2014), explaining perhaps the low efficiency of **Cuffdiff2**. It could be interesting to aggregate the counts at higher level, such as basic pieces of pathways, to decrease more the number of tests.

As RNA-sequencing is more often used for exploratory researchers, the FDR-controlling procedures are more useful than the FWER procedures. However, they have two main drawbacks which are often overlooked. First, FDR is an expected proportion and it does not guarantee that the actual proportion of false positives is equal to the given significance level (Korn *et al.*, 2004). Second, FDR procedures do not allow to only consider a subset of the DEGS (Finner and Roters, 2001), for example for functional enrichment or clustering. They do not guarantee that any random subset of the DEG is not fully made of false positives. For such considerations, only the FWER-controlling procedures are usable. An another consequence of this second point is that the FDR adjusted p-values represent the whole multiple testing control, not the tested

hypothesis itself (Goeman and Solari, 2014). It is difficult to examine them to manually reevaluate the prediction of the statistical test. In practice, these drawbacks may not be strong issues in RNA-sequencing. If there are too many DEGs or if one needs to have strong conclusions to test hypotheses, switching from FDR- to FWER-controlling procedures should be considered. The latter are more interpretable and have more advantages than merely lowering the FDR cutoff.

Still, the FDR control is still too conservative. (Storey and Tibshirani, 2003) introduced a second type of procedure to estimate the FDR, instead of defining an upper bound. The introduced the term *q-value* to define the estimated proportion of false positives we would have if the q-value of a gene is lower than the significance threshold. This is a sort of adjusted p-value, but it is not computed as for the aforementioned procedures. Today, the q-value has become a synonym for the adjusted p-value. The procedure estimates the FDR by measuring the proportion of null hypotheses from the flat tail of the p-value distributions. In comparison, the BH procedures assumes that this proportion is 1. This FDR-estimating procedure does not work on U-shaped distributions. (Burden, Qureshi and Wilson, 2014) solved this issue by removing the peak of p-values at 1 and fitting the tail with a function. The Storey-Tibshirani procedure is less conservative than the BH procedure, but it still has 3 main drawbacks (Goeman and Solari, 2014). First, it assumes the absence of dependences between the p-values. The estimated FDR has a certain variability as any estimated statistics. The estimated FDR is robust to the dependency, but its variability can be unpredictable in this context. If the variability is higher, then there is no guarantee that the actual FDR is close to the estimated FDR. Second, it does not estimate the variability of the estimated FDR, so we cannot know about its reliability. Third, it does not guarantee that any subset of the DEGs have the same subset as the whole list.

This last point motivated the work of (Goeman and Solari, 2011) who developed the R package **cherry**. The procedure estimates the FWER and its variability for any subset of genes, with or without dependencies between p-values. The goal is not to get a list of DEGs by setting a significance level, but it is to select a list of interesting genes with a measure of the risk taken. For example, one can build a list of the DEGs according to a significance, a size effect and a pathway. From this list, the Goeman-Solari procedure estimates the proportion of false positives. If necessary, the user can change the criteria to build the list in order to reduce this proportion without biasing the analysis. Such selection after a statistical test, called *post hoc inference*, is rigorously not possible with the BH procedure. Similarly, (Blanchard, Neuvial and Roquain, 2020) introduced the R package **sanssouci**. The latter offers an option to quickly select the DEGs based on both FDR and LFC cutoffs using a volcano plot (LFC vs. adjusted p-value with each gene as a dot).

To conclude, we saw procedures to correct the multiple-hypothesis testing procedure by controlling and/or estimating error rates, such as the FWER (probability to one or more false positives) and the FDR (proportion of false positives among the predicted positives). In the other sections of this thesis, I only considered the FDR control as the field usually applies the BH procedure. Yet, one can replace by another procedure depending on the goal. For example, instead of changing the FDR cutoff of the BH

procedure, one could apply an FWER-controlling procedure to be more conservative, or an FDR-estimating procedure to be more liberal. This question is still open. Recent developments show an interest in correcting the multiple-hypothesis testing by including downstream hypotheses, such as the LFC cutoff (see section 2.6). However, when comparing to the benchmarks for the normalization methods or the DEA tools, I was surprised that little attention has been paid to the correcting procedures, while they enable us to properly declare a gene being differentially expressed or not. The Benjamini-Hochberg procedure is venerable, but there may be a room for improvement.

2.6 Calling a gene "differentially expressed"

Usually, we call a gene being *differentially expressed* if it meets one or two conditions: its (adjusted) p-value is lower than the FDR cutoff and/or its size effect is lower than an (absolute) LFC cutoff.

On a theoretical point of view, there is no reason to justify an LFC cutoff, except if one wants to test a gene by qPCR, as qPCR is less sensitive than RNA-sequencing. (Hughes *et al.*, 2000) explored how mutations can affect the magnitude of differential expression. They measured the expression levels of the whole genome of 300 mutated yeasts. They concluded that dysregulated pathways were mostly made of differentially expressed genes with a low size effect. Indeed, in a pathway, a combination of many little gene-expression changes can be more important than only one change (Subramanian *et al.*, 2005). (Ideker *et al.*, 2002) also showed that the transcription factors had a lower log-fold change than genes they regulates.

The user can apply an LFC cutoff according to a volcano plot (LFC vs. adjusted p-value of genes) or the distribution of LFCs. Yet, such filtering would be arbitrary. A more rational justification is to remove a second peak of LFC, after 0, which could indicate a bias. However, if one applies an LFC cutoff, then it is equivalent to run multiple new statistical tests. These tests are said *post hoc* and can break the FDR control by inducing a *selection bias*. Therefore, it is necessary to control the FDR by considering both FDR and LFC cutoffs with a post hoc inference tool such as **cherry** (Goeman and Solari, 2011) or **sanssouci** (Blanchard, Neuvial and Roquain, 2020) (see section 2.5). In all cases, the LFC filtering should not remove genes with a high number of counts, which tend to have a smaller LFC compared to the lowly-expressed genes.

With their dataset of 42 biological replicates, (Schurch *et al.*, 2016) estimated the effect of an LFC filtering (without post hoc inference) on the recall and the specificity. A higher LFC cutoff increased the number of true positives, while maintaining stable the number of false positives, by converting false negatives into DEGs. With 3 replicates per condition, 85% of DEGs had an LFC higher than 2. To have the same proportion with 6 replicates, this threshold was at 0.5. Note that 6 replicates enabled to only identify 35% of possible DEGs. From these results, the authors estimated that the optimal LFC cutoff was 0.5 with 3 replicates and 0.25 with 10 replicates because replication enables to detect smaller size effects. Therefore, this work is in favor of LFC filtering, even without post hoc inference. However, to my knowledge, Schurch and

colleagues were the only ones to study this question, but without exploring alternatives.

To find an optimal FDR cutoff, (Lamarre *et al.*, 2018) measured the ROC curves of **DESeq2** for a number of biological replicates varying from 2 to 7. They showed that an optimal cutoff for the FDR followed the relation 2^{-r} with r the number of replicates. It means that with 3 replicates, using a cutoff of 0.1 can increase the number of true positives without increasing much the false positives. A lower cutoff would be too stringent with regard to the statistical power enabled by the replication. A cutoff of 0.05 and 0.01 is relevant with 4 and 5 replicates per condition, respectively. Such work would need to be replicated to other datasets, but it could indicate that common FDR cutoffs are unnecessary too conservative.

After the differential expression analysis, the researchers usually test few DEGs by qPCR. However, this approach cannot validate the whole results of a differential expression analysis. To do so, (Leek, Taub and Rasgon, 2012) suggested a statistical approach. First, the user randomly tests genes from the dataset by qPCR. Then, their approach computes the probability that the actual FDR is lower than the FDR initially defined by the user. To confirm the DEA, this probability must be much higher than 0.5. The testing must be random, otherwise it would bias the probability. For example, to have a validation probability of 0.5 for a FDR at 5%, then at least 240 genes should be tested. If there are 3 replicates, (Lamarre *et al.*, 2018) would suggest a FDR at 10%, giving 110 genes to test. In term of work and cost, it would be interesting to think about the use of a technology such as the BRB-sequencing we saw above (Alpern *et al.*, 2019) as an independent validation technique. The approach of Leek and colleagues is of course not necessary if the goal of the RNA-sequencing is to determine few target genes. However, if one wants to analyze the DEGs, for example with clustering (section 3.2), enrichment (section 3.3) or networks (section 4), then validating the DEA is required to support their potential conclusions.

To conclude, common procedures call a gene as differentially expressed by defining cutoffs on the false discovery rate (i.e. significance), the log-fold change (i.e. size effect) or both. Common cutoffs are often arbitrary and studies suggest other approaches to optimize them or to make them statistically sound. This is necessary to strengthen the conclusions. Nonetheless, exploring a dataset requires to select a list of genes, even with a rough approach. If one needs less DEGs, lowering the FDR and increasing the LFC cutoffs are quick and easy.

In brief, DEA is powerful and widely used but its methods are not trivial. Now, I will present approaches to analyze its results.

3 Analyzing the DEG list from time-course datasets

3.1 Transformation

Genes with a low number of counts tend to have a higher variance than the other genes. Such data are said *heteroscedastic* as the mean depends on the variance. Yet,

highly-variable data have an exaggerated weight when computing statistical measures of the dataset, e.g. clustering. (Anders and Huber, 2010) proposed the *variance-stabilizing transformation* (**VST**) to have counts with a similar variance, i.e. homoscedastic data. It transforms the counts using the mean-variance relationship and the normalization factors estimated by **DESeq**. In a next paper, (Love, Huber and Anders, 2014) introduced the *regularized-log transformation* (**rlog**) which transforms the counts using the empirical Bayes approach of **DESeq2** for moderated log-fold change (LFC). It also accounts for the normalization factors. According to the authors, **rlog** is useful for datasets with less than 30 samples, but it is less efficient with a higher number (Love, 2018). It can fail if the samples do not follow well a negative-binomial distribution, for example if there are many outliers.

More generally, it can be necessary to apply a z-score standardization on the normalized or transformed counts. It enables to have each gene with a mean of 0 and standard deviation of 1. Whereby, one can have the gene variations on the same scale when plotting the gene expression profiles. For example, **Mfuzz** chose this approach (Kumar and Futschik, 2007). On these plots, one can also color the significant variations (see figure), from the results of the DEA, as not all variations can have a biological meaning. It was interesting for us to that in (Pierrelée *et al.*, 2020). We could quickly define DEG subsets correlated to phenotypic observations using expression-profile plots.

3.2 Clustering with Mfuzz

(Futschik, 2003; Futschik and Carlisle, 2005; Kumar and Futschik, 2007) introduced the tool **Mfuzz** to cluster gene-expression profiles using an algorithm based on *soft-clustering*. Futschik developed it in his thesis of 2003. They published it in 2005 and implemented it as **Mfuzz** in 2007.

Commonly, we refer to clustering as *hard clustering*. The idea is to assign one gene to one cluster. A popular method is the *hierarchical clustering* where the clusters are nested such as one cluster regroup several sub-clusters. One can show this grouping by representing the clustering results with a dendrogram. Then, the user can set the number of clusters according to the desired level of deepness *after* the clustering. Another method is the *partitional clustering*. It is less sensible to noise than hierarchical clustering. One algorithm of partitional clustering is the *k-mean clustering*. It iteratively reassigns the genes to the clusters until the within-cluster variation is the lowest. In this method, the user sets the number of clusters *before* the clustering. The authors claimed that hard clustering will find cluster even in random data contrary to *soft clustering*.

The idea of soft clustering is to assign one gene to one or more clusters. It enables clusters with similar profiles and sharing genes. Therefore, the clusters are more precise and more robust to noise. The authors implemented into **Mfuzz** a soft-clustering algorithm called *fuzzy c-means* which generalizes the k-means clustering. In this algorithm, each gene has a membership value to each cluster. The membership varies from 0 to 1 such as the gene strongly belongs to the cluster at 1, but not at 0. Hard clustering does not allow this continuity. Above 0.5, the gene was more assigned

to the cluster than to the others. The authors defined core genes such as genes with a membership higher than 0.7. Core genes enable to filter the genes after the clustering, contrary to other methods which prefer to filter before the clustering.

Genes with low counts tend to have a higher variation (heteroscedasticity). Standard approaches usually filter out these genes. However, the authors showed that **Mfuzz** was robust to noise. To avoid arbitrary thresholds, they advised against filtering before clustering. It also enables to cluster the whole dataset without being restricted to a DEG subset. Indeed, one can search for common regulatory elements shared by the genes of a cluster. The authors showed it was easier to find them with their approach, especially among core genes and stable clusters (see below). In my experience, it is difficult to find relevant clusters among all possible clusters. To decrease this number, one can only consider the DEGs. It is motivated by the fact that DEA tools can test the significant variations. Therefore, a cluster without significant variation would be less biologically relevant.

Mfuzz clusters data in Euclidian space. We should use normalized and/or transformed counts before to apply a z-score standardization. Then, we set two parameters: the number of clusters and a "m" parameter. **Mfuzz** applies an iterative algorithm. First, it assigns random membership values for each gene. Second, it computes the *centroid* of a given cluster using the counts, the membership and the m parameter. The centroid is the average expression profile of the cluster. Third, it updates the memberships from the centroid. This step decreases the within-cluster variation, i.e. the distance of gene expression to the centroid. The algorithm repeats the two last steps until convergence or if it reaches the maximal number of iterations. One can play with the number of iterations but if this value is large enough, it does not affect the results but it improves the runtime. The default value is 1000. We can follow the convergence of the algorithm using the objective function "Jm" of **Mfuzz**. The function measures the overall within-cluster variation. In all cases, it is a good practice to plot the Jm values to ensure the convergence of the algorithm. The authors warmed the algorithm is less efficient when the number of time-points is low and when the genes tend to have the similar expression profiles.

The m parameter aims to strengthen the soft clustering by increasing the gene sharing between cluster. If m is at 1, then it is equivalent to hard clustering. Higher values decrease the membership values. The clusters would have less genes with high membership but robust clusters should keep their core genes. The authors considered that the genes shared by multiple clusters have a "large noise". Instead, robust clusters should not lose their "core" genes when the m parameter changes. To automatically set this parameter, **Mfuzz** provides an option developed by (Schwämmle and Jensen, 2010). It is usually between 1 and 1.5. In my opinion, considering genes with low memberships as noise is incorrect. The user should not ignore them. The noisiness cannot only be defined by the ability of gene to cluster. For example, it could rather mean that there are not enough genes with the same profile among the population. It also depends on the number of clusters. We could increase this number to reduce the number of genes which do not have a strong membership to a cluster, but it is not necessarily biologically meaningful.

The main issue with clustering is to define the number of clusters. For **Mfuzz**, the authors suggested 3 options. First, (Schwämmle and Jensen, 2010) benchmarked many approaches and suggested the "Dmin" function. From the clustering results, the function computes the distance between the centroids of each pair of clusters. The smallest value gives the minimum centroid distance. This distance should be stable for a low number of clusters and then drop after with higher values. The drop indicates that clusters started to split without giving clusters with unique profiles. The optimal number of clusters could be the highest number before the drop. Yet, we cannot use this approach if the drop is too fast enough. This happens when the profiles are not enough dissimilar, i.e. when the number of time-points is low. On the contrary, is the drop is too soft, picking a value would be arbitrary.

Second, **Mfuzz** provides the "Cselection" function. In a cluster, if no gene has a membership higher than 0.5, then we consider the cluster has empty. Empty clusters will appear if the number of clusters is large enough. One can set this parameter by choosing the highest number for which there are no empty clusters. In practice, a large number of genes can produce a high diversity of expression profiles. No empty cluster would appear in this context.

Finally, the authors suggested to apply functional enrichment on the clusters (see section 3.3). We test the enrichment for each cluster and we keep the clusters with the most relevant results. Doing so, we assume that genes sharing the same profile are co-regulated. It implies that co-regulation should be of the 1st order, e.g. a group of genes directly regulated by a transcription factor. This direct co-regulation should have roughly the same size effect and direction for each gene of the cluster. Though, two genes with the same profile are not necessarily co-regulated. This is a similar problem to the correlations used to infer networks (see section 4.2). Even when it is the case, if there are few time-points, it is harder to have clusters with few but consistent genes.

After we set the clustering parameters, we can explore the clustering with its stability and its global structure. We evaluate the cluster stability by varying the m parameter. Stable clusters show strong structures within the data because their core is stable. In other words, core genes of stable clusters do not change while varying m. The user should explore the most stable clusters first. Furthermore, we group the clusters according to how many genes they share. It enables to remove some unnecessary clusters or to look for common features.

The user should not focus too much on the number of clusters because it is too difficult to find a meaningful value. Instead, one should try to estimate if this number makes sense with regard to the biological context. I present an intuitive method in section 3.1.1 of the introduction.

Soft clustering brings its own ambiguity. We enable the clusters to share genes, but we want clusters with unique profiles. Our approaches aim to reduce as few as possible the number of clusters. We only consider the core genes and ignore the others which benefit from the soft clustering. Therefore, we can discuss the advantages of the soft clustering compared to other approaches.

3.3 Functional enrichment of gene lists

Functional enrichment aims to find gene sets with an overrepresented number of genes from an input list, for example the list of differentially expressed genes. A *gene set* is a list of genes annotated by a same *term*, such as a gene ontology (e.g. "nucleus"), a pathway (e.g. "cell cycle") or a transcription factor (e.g. "CREB"). Statistical tests compute these gene sets and give them a score to rank according to their level of enrichment. They are called *enriched terms* when their score is higher than a given threshold. Many methods were developed to build gene sets, weight the genes of the input list and test their enrichment. (Huang, Sherman and Lempicki, 2009) Even in 2009, 68 methods were already available to enrich gene sets.

3.3.1 GSEA

(Subramanian *et al.*, 2005) developed the Gene Set Enrichment Analysis (**GSEA**). It aims to identify the gene sets which are overrepresented at the top and at the bottom of a ranked gene list. For the authors, the advantages of gene enrichment are to take into accounts genes with low size effect, to reduce the noise in the DEG identification and to enable a better overlapping of results from the experiments studying a same pathway. **GSEA** can take as an input either the matrix of read counts or a pre-ranked gene list. An advantage of the first option is to not filter out genes based on a particular threshold. Thus, the results do not depend on an arbitrary choice and the genes which are not passing the test can still contribute to the enrichment. However, the pre-ranked list is the step after the differential expression analysis which already ranked the genes. The pre-ranked list enables to customize the gene's rank score. (Xiao *et al.*, 2014) ranked the genes according to both the adjusted p-values and the log-fold change. They combined these scores with the formula $-\log_{10} pval * LFC$. The genes with high significance and size effect have higher ranks. (Zyla *et al.*, 2017) benchmarked 11 other ranking metrics for microarrays, so the study is offside.

GSEA applies an iterative algorithm. For each gene set, a "walker" reads the ranked list from the top to the bottom and increases its value when it meets a gene belonging to the set. Otherwise, if the gene is not in the set, the walker decreases its value. The increase or decrease depends on the gene's score. The *enrichment score* of a gene set is the highest value of the walker. Then, a *Kilmogorov-like statistics* computes a p-value by comparing the enrichment score to an empirical null distribution, i.e. determined from the dataset. Finally, it estimates the false discovery rate (FDR) for each enrichment score. The authors advised a FDR lower than 0.25 to call a term enriched.

As we saw, **GSEA** requires pre-made gene sets. (Liberzon *et al.*, 2015) created the Molecular Signature Database (**MSigDB**). This dataset is human-oriented. It has around 30,000 curated gene sets split into 9 collections. Especially, the "hallmark" collection is a summary of the 8 other collections, but the authors corrected 3 biases related to the enrichment. First, many enriched terms are related to the same biological process. Also, we often find many terms enriched thanks to the same genes. Third, genes from a gene set do not always have the same behavior over time.

3.3.2 Enrichr

(Chen *et al.*, 2013; Kuleshov *et al.*, 2016, 2019) introduced **enrichr**. It comes with a database of gene sets for 6 species: Human, Mouse, Fly, Yeast, Worm and Fish. For human and mouse species, more than 300,000 gene sets are available divided into 17 collections. The biggest advantage of **enrichr** over **GSEA** is that we can easily enrich a gene list from their webpage. The authors also developed a complete API (Application Programming Interface) which is quite convenient to use. The user only needs to provide a gene list.

Enrichr computes three enrichment scores. First, it computes a p-value with the exact fisher test using a binomial distribution. Contrary to **GSEA**, it assumes gene-gene independence. In parallel, enrichr computes a z-score by comparing the enrichment to random models. Then, it combines these two scores by combining the p-value and the z-score. The authors showed that this combined score was more efficient than the two other scores alone.

Contrary to **GSEA**, web service of **enrichr** cannot take into account a custom *background*. A background is a reference gene list against which the statistical test computes the enrichment (Simillion *et al.*, 2017). It enables to remove the *sample bias*. To illustrate, let's take as input a DEG list from an experiment. The experiment explores a limited number of conditions where the biological material expresses a limited number. A cell does not express all genes at the same type. Thereby, a gene cannot obviously be differentially expressed if it is not expressed in any condition. Yet, gene sets include all genes from the genome. They do not depend on the experiment. Consequently, the DEGs have a higher weight than the other genes, so gene sets having few DEGs will become more significant, even if their presence was purely random. To avoid the sample bias, the background should only contain the genes expressed during the experiment. It results in less spurious enriched terms.

For (Tamayo *et al.*, 2016), the Fisher's exact test of **enrichr** and the hypergeometric tests are competitors of the Kilmogorov-like statistics computed by **GSEA**. Tamayo and colleagues showed that the methods based on these two former tests overlook the most enriched terms and assume gene-gene independence. Thus, they increase the number of false positives. More generally, (Huang, Sherman and Lempicki, 2009) criticized the use of p-values for the functional enrichment. Various factors can strongly affect them. Instead, the order of enriched terms should be taken into account by varying the background. Moreover, (Blüthgen *et al.*, 2005) considered that the standard FDR control is not suited for gene sets because it is too conservative with regard to the huge number of gene sets to enrich. For example, **GSEA** authors recommended a FDR cutoff of 0.25. Note that **enrichr** authors recommended a cutoff at 0.05. Blüthgen and colleagues suggested a method to estimate the FDR by computing the number of false discoveries. They showed it was more exact and robust to other estimation procedures. These considerations seem to have been ignored lately.

3.3.3 Creating gene sets

The aforementioned tools provide pre-made gene sets for common model species. However, most species and in particular, the datasets from *de novo* sequencing, do not

have ready-to-use gene sets, as there are no annotations for their genome. I was confronted to this issue for the paper (Yun *et al.*, 2020) where we had only had the transcriptome of the biological material. We have to define an ortholog for each gene and fetch its annotations such as pathways. Then, we build gene sets by grouping the gene with the same annotations.

This is less straightforward for gene ontology annotations. Gene ontologies (GOs) are standard annotations of genes (Ashburner et al., 2000). A GO can be a more specialized annotation than another GO and in this case, the specialized GO is linked to the general GO. The GO annotations shape 3 independent trees starting from the root (i.e. most general) GOs "Molecular Function", "Biological Process" and "Cellular Component", respectively. All the other GOs are their direct or indirect subsets. This relationship between the GOs enables to propagate the gene assignment through the GO tree. If a gene is assigned to a given GO, then it is also assigned to the containing GOs. Therefore, we can use that to build GO gene sets. Other type of relations between the GOs exist, but they do not interest us.

Enrichr creates the GO gene sets as follows (Kuleshov et al., 2019). First, it builds the GO trees. Then, for each tree, it removes the 3 first levels starting from the root GO. Third, it assigns the genes from the specialized GOs to the general GOs far from 4 levels at most. Finally, it removes the gene sets with less than 5 genes.

In comparison, (Powell, 2014) developed **GO2MSIG** to automatically generate gene sets for **GSEA** from a GO tree and a list of annotated genes representing the background. The approach is similar to **Enrichr** with the difference that it does not remove the first levels and it does not limit the tree propagation. The user can set a minimum and maximum number of genes in each set. Moreover, the author corrected another bias resulting from gene sets with the same genes by merging them into a single set. Indeed, identical gene sets would affect the enrichment scores without adding value. I manually applied this method in (Yun *et al.*, 2020).

In experiments, we expect to get enriched terms consistent with the experiment. For example, comparing healthy tissue to a tumor should result in enriched terms related to growth and angiogenesis. This information could be taken into account when computing the enrichment. It would enable to highlight unexpected enriched terms or, on the contrary, found terms which are unexpectedly not enriched. Bayesian statistics enables that and could improve the statistics.

To conclude, functional enrichment enables to find the relationships between the differentially expressed genes. It groups the genes because, for a given enriched gene set, we obtain the DEGs which contributed to its enrichment. There are two popular tools: GSEA and enrichr. GSEA removes biases by taking into account a background and the gene dependencies. In comparison, enrichr is much easier to use. Both tools have a limited number of species and gene sets. Therefore, one has to build gene sets dedicated to non-annotated species if this information is not available.

3.4 Limits of approaches to analyze

Differential expression analysis is a long and complex workflow. It directly affects the downstream methods such as clustering and functional enrichment. If the statistical power is low, then these methods produce truncated results that could change the conclusion. Each method has its own drawbacks

Clustering finds expression patterns among a gene list. One can use the whole dataset, but it makes harder to the clustering parameters. The clusters without differential expression are not ensured to be meaningful. (Ideker *et al.*, 2002) explaining that clustering depend on the strength and the direction of the expression changes. These are stringent constrained. Moreover, it assumes that genes with similar expression profiles can be grouped. When exploring the results, one can interpret them as co-regulated genes across all conditions. This is not always true. Most clustering methods will cluster all genes, while some of them are not involved in the dysregulated pathway. On the practical side, applying clustering takes time. While it does not bring any information on the relationships, the following analyses of the results are limited.

From the DEG lists or the clustering results, one can apply functional enrichment. It gives functional hypotheses on the patterns within the gene list. Yet, enrichment heavily depends on the gene sets, the input lists and the algorithm (Huang, Sherman and Lempicki, 2009). RNA-sequencing biases the input list. For example, enriched terms include longer genes than expected because they have more chances to be called differentially expressed. Moreover, we already know that DEA does not identify most of the DEGs (e.g. (Schurch *et al.*, 2016)). Missing one gene can greatly affect the significance of gene sets with few terms. Gene sets with many terms are not necessarily meaningful either. We often observe many enriched terms with more or less the same genes, likely because the latter are highly annotated. Therefore, it is difficult to compare enriched terms from different experiments without controlled the biases, such as those presented by **MSigDB**. The incompleteness of annotations is a strong issue with functional enrichment. Even if the genes are well annotated, these annotations could have been made from a particular condition which does not enable to generalize them (Tamayo *et al.*, 2016). (Corley *et al.*, 2017) observed that even a sequencing protocol (single- or paired-end and stranded or non-stranded) affected the enrichment. The authors showed these protocols had an overlap of only 40 to 60% for the 20 most enriched GOs. The overlap was much higher when considering a larger number of enriched GOs, but it indicates that the user cannot only trust the top enriched terms. This is not tractable if there are hundreds of enriched terms. On the end, "*the notion that the enriched terms should make sense based on a priori biological knowledge of the study is the most important guideline*" (Huang, Sherman and Lempicki, 2009). Even if the enrichment analysis is long-time popular approach, there are still rooms for improvements, from building gene sets to computing enrichment statistics (Huang, Sherman and Lempicki, 2009; Tamayo *et al.*, 2016).

References

Akhmedov, M. *et al.* (2017) 'PCSF: An R-package for network-based interpretation of high-throughput data', *PLOS Computational Biology*. Edited by D. Schneidman. Public Library of Science, 13(7), p. e1005694. doi: 10.1371/journal.pcbi.1005694.

Almozlino, Y. *et al.* (2017) 'ANAT 2.0: reconstructing functional protein subnetworks', *BMC Bioinformatics*. BioMed Central, 18(1), p. 495. doi: 10.1186/s12859-017-1932-1.

Alpern, D. *et al.* (2019) 'BRB-seq: ultra-affordable high-throughput transcriptomics enabled by bulk RNA barcoding and sequencing', *Genome Biology*. BioMed Central Ltd., 20(1), p. 71. doi: 10.1186/s13059-019-1671-x.

Anand, R. and Chatterjee, S. (2017) 'Tracking disease progression by searching paths in a temporal network of biological processes', *PLOS ONE*. Edited by G. Bontempi. Public Library of Science, 12(4), p. e0176172. doi: 10.1371/journal.pone.0176172.

Anders, S. and Huber, W. (2010) 'Differential expression analysis for sequence count data', *Genome Biology*, 11(10), p. R106. doi: 10.1186/gb-2010-11-10-r106.

Aoki, K., Ogata, Y. and Shibata, D. (2007) 'Approaches for Extracting Practical Information from Gene Co-expression Networks in Plant Biology', *Plant and Cell Physiology*, 48(3), pp. 381–390. doi: 10.1093/pcp/pcm013.

Ashburner, M. *et al.* (2000) 'Gene Ontology: tool for the unification of biology', *Nature Genetics*, 25(1), pp. 25–29. doi: 10.1038/75556.

Auer, P. L. and Doerge, R. W. (2010) 'Statistical Design and Analysis of RNA Sequencing Data', *Genetics*. Genetics, 185(2), pp. 405–416. doi: 10.1534/genetics.110.114983.

Ballouz, S., Verleyen, W. and Gillis, J. (2015) 'Guidance for RNA-seq co-expression network construction and analysis: safety in numbers', *Bioinformatics*, 31(13), pp. 2123–2130. doi: 10.1093/bioinformatics/btv118.

Bandyopadhyay, S. *et al.* (2010) 'Rewiring of Genetic Networks in Response to DNA Damage', *Science*, 330(6009), pp. 1385–1389. doi: 10.1126/science.1195618.

Bar-Joseph, Z., Gitter, A. and Simon, I. (2012) 'Studying and modelling dynamic biological processes using time-series gene expression data', *Nature Reviews Genetics*. Nature Publishing Group, 13(8), pp. 552–564. doi: 10.1038/nrg3244.

Batra, R. *et al.* (2017) 'On the performance of de novo pathway enrichment', *npj Systems Biology and Applications*. Springer US, 3(1), p. 6. doi: 10.1038/s41540-017-0007-2.

Bender, R. and Lange, S. (2001) 'Adjusting for multiple testing—when and how?', *Journal of Clinical Epidemiology*, 54(4), pp. 343–349. doi: 10.1016/S0895-4356(00)00314-0.

Benjamini, Y. and Hochberg, Y. (1995) 'Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing', *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1), pp. 289–300. doi: https://www.jstor.org/stable/2346101.

Blanchard, G., Neuvial, P. and Roquain, E. (2020) 'Post hoc confidence bounds on false positives using reference families', *Annals of Statistics*, 48(3), pp. 1281–1303. doi: 10.1214/19-AOS1847.

Blüthgen, N. *et al.* (2005) 'Biological profiling of gene groups utilizing Gene Ontology.', *Genome informatics. International Conference on Genome Informatics*, 16(1), pp. 106–115. doi: 10.11234/gi1990.16.106.

Bostock, M. (2011) 'D3.js'. Available at: https://d3js.org/.

Bourgon, R., Gentleman, R. and Huber, W. (2010) 'Independent filtering increases detection power for high-throughput experiments', *Proceedings of the National Academy of Sciences*, 107(21), pp. 9546–9551. doi: 10.1073/pnas.0914005107.

Braun, P. *et al.* (2009) 'An experimentally derived confidence score for binary protein-protein interactions', *Nature Methods*, 6(1), pp. 91–97. doi: 10.1038/nmeth.1281.

Broido, A. D. and Clauset, A. (2019) 'Scale-free networks are rare', *Nature Communications*. Springer US, 10(1), p. 1017. doi: 10.1038/s41467-019-08746-5.

Bryant, D. M. *et al.* (2017) 'A Tissue-Mapped Axolotl De Novo Transcriptome Enables Identification of Limb Regeneration Factors', *Cell Reports*. ElsevierCompany., 18(3), pp. 762–776. doi: 10.1016/j.celrep.2016.12.063.

Bui-Xuan, B.-M., Ferreira, A. and Jarry, A. (2003) 'Computing shortest, fastest, and foremost journeys in dynamic networks', *International Journal of Foundations of Computer Science*, 14(02), pp. 267–285. doi: 10.1142/S0129054103001728.

Bullard, J. H. *et al.* (2010) 'Evaluation of statistical methods for normalization and differential expression in mRNA-Seq experiments', *BMC Bioinformatics*, 11(1), p. 94. doi: 10.1186/1471-2105-11-94.

Burden, C. J., Qureshi, S. E. and Wilson, S. R. (2014) 'Error estimates for the analysis of differential expression from RNA-seq count data', *PeerJ*, 2(1), p. e576. doi: 10.7717/peerj.576.

Camacho, C. *et al.* (2009) 'BLAST+: architecture and applications', *BMC Bioinformatics*, 10(1), p. 421. doi: 10.1186/1471-2105-10-421.

Cantini, L. *et al.* (2015) 'Detection of gene communities in multi-networks reveals cancer drivers.', *Scientific reports*. Nature Publishing Group, 5, p. 17386. doi: 10.1038/srep17386.

Chakrabarti, D., Kumar, R. and Tomkins, A. (2006) 'Evolutionary clustering', in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*. New York, New York, USA: ACM Press, p. 554. doi: 10.1145/1150402.1150467.

Chen, E. Y. *et al.* (2013) 'Enrichr: Interactive and collaborative HTML5 gene list enrichment analysis tool', *BMC Bioinformatics*, 14, p. 128. doi: 10.1186/1471-2105-14-128.

Chen, K. *et al.* (2016) 'The Overlooked Fact: Fundamental Need for Spike-In Control

for Virtually All Genome-Wide Analyses', *Molecular and Cellular Biology*, 36(5), pp. 662–667. doi: 10.1128/MCB.00970-14.

Chen, Y., Lun, A. T. L. and Smyth, G. K. (2014) 'Differential Expression Analysis of Complex RNA-seq Experiments Using edgeR', in *Statistical Analysis of Next Generation Sequencing Data*. Cham: Springer International Publishing, pp. 51–74. doi: 10.1007/978-3-319-07212-8_3.

Chen, Y., Lun, A. T. L. and Smyth, G. K. (2016) 'From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline', *F1000Research*. Faculty of 1000 Ltd, 5, p. 1438. doi: 10.12688/f1000research.8987.2.

Ching, T., Huang, S. and Garmire, L. X. (2014) 'Power analysis and sample size estimation for RNA-Seq differential expression', *RNA*. Cold Spring Harbor Laboratory Press, 20(11), pp. 1684–1696. doi: 10.1261/rna.046011.114.

Chowdhury, H. A., Bhattacharyya, D. K. and Kalita, J. K. (2020a) '(Differential) Co-Expression Analysis of Gene Expression: A Survey of Best Practices', *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(4), pp. 1154–1173. doi: 10.1109/TCBB.2019.2893170.

Chowdhury, H. A., Bhattacharyya, D. K. and Kalita, J. K. (2020b) 'Differential Expression Analysis of RNA-seq Reads: Overview, Taxonomy, and Tools', *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(2), pp. 566–586. doi: 10.1109/TCBB.2018.2873010.

Clauset, A., Newman, M. E. J. and Moore, C. (2004) 'Finding community structure in very large networks', *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 70(6), p. 6. doi: 10.1103/PhysRevE.70.066111.

Conesa, A. *et al.* (2005) 'Blast2GO: a universal tool for annotation, visualization and analysis in functional genomics research', *Bioinformatics*, 21(18), pp. 3674–3676. doi: 10.1093/bioinformatics/bti610.

Conesa, A. *et al.* (2006) 'maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments', *Bioinformatics*. Oxford Academic, 22(9), pp. 1096–1102. doi: 10.1093/bioinformatics/btl056.

Contreras-López, O. *et al.* (2018) 'Step-by-Step Construction of Gene Co-expression Networks from High-Throughput Arabidopsis RNA Sequencing Data', in Ristova, D. and Barbez, E. (eds) *Root Development. Methods in Molecular Biology*. 1st edn. New York, NY: Humana Press, pp. 275–301. doi: 10.1007/978-1-4939-7747-5_21.

Corley, S. M. *et al.* (2017) 'Differentially expressed genes from RNA-Seq and functional enrichment results are affected by the choice of single-end versus paired-end reads and stranded versus non-stranded protocols', *BMC Genomics*. BioMed Central, 18(1), p. 399. doi: 10.1186/s12864-017-3797-0.

Cozzo, E. *et al.* (2015) 'Structure of triadic relations in multiplex networks', *New Journal of Physics*. IOP Publishing, 17(7), p. 073029. doi: 10.1088/1367-2630/17/7/073029.

Curtis, R. E. *et al.* (2012) 'Enabling dynamic network analysis through visualization in TVNViewer', *BMC Bioinformatics*. BioMed Central, 13(1), p. 204. doi: 10.1186/1471-2105-13-204.

van Dam, S. *et al.* (2018) 'Gene co-expression analysis for functional classification and gene–disease predictions', *Briefings in Bioinformatics*. Oxford University Press, 19(4), pp. 575–592. doi: 10.1093/bib/bbw139.

Dao, D., Wilzbach, S. and Corpas, M. (2014) 'BioJS'. Available at: https://github.com/biojs.

Davidson, N. M., Hawkins, A. D. K. and Oshlack, A. (2017) 'SuperTranscripts: A data driven reference for analysis and visualisation of transcriptomes', *Genome Biology*. BioMed Central Ltd., 18(1), p. 148. doi: 10.1186/s13059-017-1284-1.

Dillies, M. A. *et al.* (2013) 'A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis', *Briefings in Bioinformatics*, 14(6), pp. 671–683. doi: 10.1093/bib/bbs046.

Djordjevic, D. *et al.* (2014) 'How Difficult Is Inference of Mammalian Causal Gene Regulatory Networks?', *PLoS ONE*. Edited by F. Emmert-Streib, 9(11), p. e111661. doi: 10.1371/journal.pone.0111661.

Dobin, A. *et al.* (2013) 'STAR: ultrafast universal RNA-seq aligner', *Bioinformatics*. Oxford University Press, 29(1), pp. 15–21. doi: 10.1093/bioinformatics/bts635.

De Domenico, M. *et al.* (2015) 'Identifying Modular Flows on Multilayer Networks Reveals Highly Overlapping Organization in Interconnected Systems', *Physical Review X*, 5(1), p. 011027. doi: 10.1103/PhysRevX.5.011027.

Ernst, J. *et al.* (2007) 'Reconstructing dynamic regulatory maps', *Molecular Systems Biology*. John Wiley & Sons, Ltd, 3(1), p. 74. doi: 10.1038/msb4100115.

Evans, C., Hardin, J. and Stoebel, D. M. (2018) 'Selecting between-sample RNA-Seq normalization methods from the perspective of their assumptions', *Briefings in Bioinformatics*, 19(5), pp. 776–792. doi: 10.1093/bib/bbx008.

Faust, K. *et al.* (2010) 'Pathway discovery in metabolic networks by subgraph extraction', *Bioinformatics*. Narnia, 26(9), pp. 1211–1218. doi: 10.1093/bioinformatics/btq105.

Fields, S. and Song, O. (1989) 'A novel genetic system to detect protein–protein interactions', *Nature*. Nature Publishing Group, 340(6230), pp. 245–246. doi: 10.1038/340245a0.

Finner, H. and Roters, M. (2001) 'On the False Discovery Rate and Expected Type I Errors', *Biometrical Journal*, 43(8), p. 985. doi: 10.1002/1521-4036(200112)43:8<985::AID-BIMJ985>3.0.CO;2-4.

Fischer, D. S., Theis, F. J. and Yosef, N. (2018) 'Impulse model-based differential expression analysis of time course sequencing data', *Nucleic Acids Research*. Oxford University Press, 46(20), pp. 1–10. doi: 10.1093/nar/gky675.

Fornes, O. *et al.* (2019) 'JASPAR 2020: update of the open-access database of transcription factor binding profiles', *Nucleic Acids Research*, 48(D1), pp. D87–D92. doi: 10.1093/nar/gkz1001.

Fortunato, S. and Hric, D. (2016) 'Community detection in networks: A user guide', *Physics Reports*. Elsevier B.V., 659, pp. 1–44. doi: 10.1016/j.physrep.2016.09.002.

Friedman, N. *et al.* (2000) 'Using Bayesian Networks to Analyze Expression Data', *Journal of Computational Biology*, 7(3–4), pp. 601–620. doi: 10.1089/106652700750050961.

Futschik, M. E. (2003) *Methods for Knowledge Discovery in Microarray Data, Chapter V: Clustering of Gene Expression Data*. University of Otago, Dunedin New Zealand. Available at: http://www.sysbiolab.eu/PhD/index.htm.

Futschik, M. E. and Carlisle, B. (2005) 'Noise-Robust Soft Clustering of Gene Expression Time-Course Data', *Journal of Bioinformatics and Computational Biology*, 03(04), pp. 965–988. doi: 10.1142/S0219720005001375.

Garmhausen, M. *et al.* (2015) 'Virtual pathway explorer (viPEr) and pathway enrichment analysis tool (PEANuT): Creating and analyzing focus networks to identify cross-talk between molecules and pathways', *BMC Genomics*. BMC Genomics, 16(1), pp. 1–13. doi: 10.1186/s12864-015-2017-z.

Gil, D. P., Law, J. N. and Murali, T. M. (2017) 'The PathLinker app: Connect the dots in protein interaction networks', *F1000Research*, 6, p. 58. doi: 10.12688/f1000research.9909.1.

Glaab, E. *et al.* (2012) 'EnrichNet: network-based gene set enrichment analysis', *Bioinformatics*, 28(18), pp. i451–i457. doi: 10.1093/bioinformatics/bts389.

Goeman, J. J. and Solari, A. (2011) 'Multiple Testing for Exploratory Research', *Statistical Science*. Institute of Mathematical Statistics, 26(4), pp. 584–597. doi: 10.1214/11-STS356.

Goeman, J. J. and Solari, A. (2014) 'Multiple hypothesis testing in genomics', *Statistics in Medicine*, 33(11), pp. 1946–1978. doi: 10.1002/sim.6082.

Goenawan, I. H., Bryan, K. and Lynn, D. J. (2016) 'DyNet: visualization and analysis of dynamic molecular interaction networks', *Bioinformatics*. Narnia, 32(17), pp. 2713–2715. doi: 10.1093/bioinformatics/btw187.

Gu, J. *et al.* (2010) 'Identification of responsive gene modules by network-based gene clustering and extending: application to inflammation and angiogenesis', *BMC Systems Biology*, 4(1), p. 47. doi: 10.1186/1752-0509-4-47.

Gupta, S. K. *et al.* (2020) 'Genome-wide inference of the Camponotus floridanus protein-protein interaction network using homologous mapping and interacting domain profile pairs', *Scientific Reports*, 10(1), p. 2334. doi: 10.1038/s41598-020-59344-1.

Haas, B. J. *et al.* (2013) 'De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis', *Nature Protocols*. Nature Publishing Group, 8(8), pp. 1494–1512. doi: 10.1038/nprot.2013.084.

Haile, S. *et al.* (2019) 'Evaluation of protocols for rRNA depletion-based RNA sequencing of nanogram inputs of mammalian total RNA', *PLOS ONE*. Edited by T. Preiss. Public Library of Science, 14(10), p. e0224578. doi: 10.1371/journal.pone.0224578.

Han, H. *et al.* (2018) 'TRRUST v2: An expanded reference database of human and mouse transcriptional regulatory interactions', *Nucleic Acids Research*. Oxford University Press, 46(D1), pp. D380–D386. doi: 10.1093/nar/gkx1013.

Hartwell, L. H. *et al.* (1999) 'From molecular to modular cell biology', *Nature*, 402(S6761), pp. C47–C52. doi: 10.1038/35011540.

Hawinkel, S. *et al.* (2020) 'Sequence count data are poorly fit by the negative binomial distribution', *PLOS ONE*. Edited by S. Kumar, 15(4), p. e0224909. doi: 10.1371/journal.pone.0224909.

Hermjakob, H. *et al.* (2004) 'The HUPO PSI's Molecular Interaction format—a community standard for the representation of protein interaction data', *Nature Biotechnology*, 22(2), pp. 177–183. doi: 10.1038/nbt926.

Hill, S. M. *et al.* (2012) 'Bayesian Inference of Signaling Network Topology in a Cancer Cell Line', *Bioinformatics*, 28(21), pp. 2804–2810. doi: 10.1093/bioinformatics/bts514.

Holik, A. Z. *et al.* (2017) 'RNA-seq mixology: designing realistic control experiments to compare protocols and analysis methods', *Nucleic Acids Research*. Oxford University Press, 45(5), pp. e30–e30. doi: 10.1093/nar/gkw1063.

Holm, S. (1979) 'A Simple Sequentially Rejective Multiple Test Procedure', *Scandinavian Journal of Statistics*, 6(2), pp. 65–70. doi: 10.2307/4615733.

Holme, P. (2015) 'Modern temporal network theory: a colloquium', *The European Physical Journal B*, 88(9), p. 234. doi: 10.1140/epjb/e2015-60657-4.

Holme, P. and Saramäki, J. (2012) 'Temporal networks', *Physics Reports*, 519, pp. 97–125. doi: 10.1016/j.physrep.2012.03.001.

Hric, D., Darst, R. K. and Fortunato, S. (2014) 'Community detection in networks: Structural communities versus ground truth', *Physical Review E*, 90(6), p. 062805. doi: 10.1103/PhysRevE.90.062805.

Huang, D. W., Sherman, B. T. and Lempicki, R. A. (2009) 'Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists', *Nucleic Acids Research*. Oxford University Press, 37(1), pp. 1–13. doi: 10.1093/nar/gkn923.

Huang, S. -s. C. and Fraenkel, E. (2009) 'Integrating Proteomic, Transcriptional, and Interactome Data Reveals Hidden Components of Signaling and Regulatory Networks', *Science Signaling*. NIH Public Access, 2(81), pp. ra40–ra40. doi: 10.1126/scisignal.2000350.

Huang, X. *et al.* (2021) 'A survey of community detection methods in multilayer networks', *Data Mining and Knowledge Discovery*. Springer US, 35(1), pp. 1–45. doi: 10.1007/s10618-020-00716-6.

Hughes, T. R. *et al.* (2000) 'Functional Discovery via a Compendium of Expression Profiles', *Cell*, 102(1), pp. 109–126. doi: 10.1016/S0092-8674(00)00015-5.

Husmeier, D. (2003) 'Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks', *Bioinformatics*, 19(17), pp. 2271–2282. doi: 10.1093/bioinformatics/btg313.

Ideker, T. *et al.* (2002) 'Discovering regulatory and signalling circuits in molecular interaction networks', *Bioinformatics*. Narnia, 18(Suppl 1), pp. S233–S240. doi: 10.1093/bioinformatics/18.suppl_1.S233.

Illumina (2010) *TruSeq RNA Sample Prep Guide (15008136 A)*. Available at: https://emea.support.illumina.com/content/dam/illumina-support/documents/documentation/chemistry_documentation/samplepreps_truseq/truseqrna/truseq_rna_sampleprep_guide_15008136_a.pdf (Accessed: 2 December 2020).

Jacob, F. and Monod, J. (1961) 'Genetic regulatory mechanisms in the synthesis of proteins', *Journal of Molecular Biology*, 3(3), pp. 318–356. doi: 10.1016/S0022-2836(61)80072-7.

Jeong, H. *et al.* (2000) 'The large-scale organization of metabolic networks', *Nature*, 407(6804), pp. 651–654. doi: 10.1038/35036627.

Kanehisa, M. (2000) 'KEGG: Kyoto Encyclopedia of Genes and Genomes', *Nucleic Acids Research*, 28(1), pp. 27–30. doi: 10.1093/nar/28.1.27.

Kanehisa, M. and Sato, Y. (2020) 'KEGG Mapper for inferring cellular functions from protein sequences', *Protein Science*, 29(1), pp. 28–35. doi: 10.1002/pro.3711.

Kivelä, M. *et al.* (2014) 'Multilayer networks', *Journal of Complex Networks*. Oxford University Press, 2(3), pp. 203–271. doi: 10.1093/comnet/cnu016.

Köksal, A. S. *et al.* (2018) 'Synthesizing Signaling Pathways from Temporal Phosphoproteomic Data.', *Cell reports*. Elsevier, 24(13), pp. 3607–3618. doi: 10.1016/j.celrep.2018.08.085.

Korn, E. L. *et al.* (2004) 'Controlling the number of false discoveries: application to high-dimensional genomic data', *Journal of Statistical Planning and Inference*, 124(2), pp. 379–398. doi: 10.1016/S0378-3758(03)00211-8.

Kuleshov, M. V. *et al.* (2016) 'Enrichr: a comprehensive gene set enrichment analysis web server 2016 update', *Nucleic acids research*, 44(W1), pp. W90–W97. doi: 10.1093/nar/gkw377.

Kuleshov, M. V. *et al.* (2019) 'modEnrichr: a suite of gene set enrichment analysis tools for model organisms', *Nucleic Acids Research*. Oxford University Press, 47(W1), pp. W183–W190. doi: 10.1093/nar/gkz347.

Kumar, L. and Futschik, M. E. (2007) 'Mfuzz: A software package for soft clustering of microarray data', *Bioinformation*, 2(1), pp. 5–7. doi: 10.6026/97320630002005.

Kumari, S. *et al.* (2012) 'Evaluation of Gene Association Methods for Coexpression Network Construction and Biological Knowledge Discovery', *PLoS ONE*. Edited by S. Horvath, 7(11), p. e50411. doi: 10.1371/journal.pone.0050411.

Lamarre, S. *et al.* (2018) 'Optimization of an RNA-Seq Differential Gene Expression Analysis Depending on Biological Replicate Number and Library Size', *Frontiers in Plant Science*. Frontiers Media SA, 9, p. 108. doi: 10.3389/fpls.2018.00108.

Langmead, B. and Salzberg, S. L. (2012) 'Fast gapped-read alignment with Bowtie 2', *Nature Methods*, 9(4), pp. 357–359. doi: 10.1038/nmeth.1923.

Law, C. W. *et al.* (2014) 'Voom: Precision weights unlock linear model analysis tools for RNA-seq read counts', *Genome Biology*, 15(2), pp. 1–17. doi: 10.1186/gb-2014-15-2-r29.

Leek, J. T., Taub, M. A. and Rasgon, J. L. (2012) 'A statistical approach to selecting and confirming validation targets in -omics experiments', *BMC Bioinformatics*. BioMed Central, 13(1), p. 150. doi: 10.1186/1471-2105-13-150.

Lewis, A. C. *et al.* (2010) 'The function of communities in protein interaction networks at multiple scales', *BMC Systems Biology*, 4(1), p. 100. doi: 10.1186/1752-0509-4-100.

Li, B. and Dewey, C. N. (2011) 'RSEM: accurate transcript quantification from RNA-Seq data with or without a reference genome', *BMC Bioinformatics*. Edited by Y. Liu. Apple Academic Press, 12(1), p. 323. doi: 10.1186/1471-2105-12-323.

Li, M. *et al.* (2018) 'DyNetViewer: a Cytoscape app for dynamic network construction, analysis and visualization', *Bioinformatics*. Edited by J. Wren. Oxford University Press, 34(9), pp. 1597–1599. doi: 10.1093/bioinformatics/btx821.

Li, P. *et al.* (2015) 'Comparing the normalization methods for the differential analysis of Illumina high-throughput RNA-Seq data', *BMC Bioinformatics*. BMC Bioinformatics, 16(1), p. 347. doi: 10.1186/s12859-015-0778-7.

Li, S. *et al.* (2004) 'A map of the interactome network of the metazoan C. elegans.', *Science*, 303(5657), pp. 540–3. doi: 10.1126/science.1091403.

Li, X. *et al.* (2020) 'Choice of library size normalization and statistical methods for differential gene expression analysis in balanced two-group comparisons for RNA-seq studies', *BMC Genomics*. BMC Genomics, 21(1), pp. 1–17. doi: 10.1186/s12864-020-6502-7.

Liao, Y., Smyth, G. K. and Shi, W. (2014) 'featureCounts: an efficient general purpose program for assigning sequence reads to genomic features', *Bioinformatics*. Oxford University Press, 30(7), pp. 923–930. doi: 10.1093/bioinformatics/btt656.

Liberzon, A. *et al.* (2015) 'The Molecular Signatures Database Hallmark Gene Set Collection', *Cell Systems*, 1(6), pp. 417–425. doi: 10.1016/j.cels.2015.12.004.

Lima-Mendez, G. and van Helden, J. (2009) 'The powerful law of the power law and other myths in network biology', *Molecular BioSystems*, 5(12), p. 1482. doi: 10.1039/b908681a.

Lin, C. Y. *et al.* (2012) 'Transcriptional amplification in tumor cells with elevated c-Myc', *Cell*, 151(1), pp. 56–67. doi: 10.1016/j.cell.2012.08.026.

Lin, Y. *et al.* (2016) 'Comparison of normalization and differential expression analyses using RNA-Seq data from 726 individual Drosophila melanogaster.', *BMC genomics*. BMC Genomics, 17(1), p. 28. doi: 10.1186/s12864-015-2353-z.

Lisso, J. *et al.* (2005) 'Identification of brassinosteroid-related genes by means of transcript co-response analyses', *Nucleic Acids Research*, 33(8), pp. 2685–2696. doi: 10.1093/nar/gki566.

Liu, R. *et al.* (2015) 'Why weight? Modelling sample and observational level variability improves power in RNA-seq analyses', *Nucleic Acids Research*, 43(15), pp. e97–e97. doi: 10.1093/nar/gkv412.

Liu, Y., Zhou, J. and White, K. P. (2014) 'RNA-seq differential expression studies: more sequence or more replication?', *Bioinformatics*. Oxford University Press, 30(3), pp. 301–304. doi: 10.1093/bioinformatics/btt688.

Liu, Z.-P. (2015) 'Reverse Engineering of Genome-wide Gene Regulatory Networks from Gene Expression Data', *Current Genomics*, 16(1), pp. 3–22. doi: 10.2174/1389202915666141110210634.

Liu, Z. P. *et al.* (2015) 'RegNetwork: An integrated database of transcriptional and post-transcriptional regulatory networks in human and mouse', *Database*, 2015, pp. 1–12. doi: 10.1093/database/bav095.

López, Y., Nakai, K. and Patil, A. (2015) 'HitPredict version 4: comprehensive reliability scoring of physical protein–protein interactions from more than 100 species', *Database*. Oxford University Press, 2015(1), p. bav117. doi: 10.1093/database/bav117.

López, Y., Nakai, K. and Patil, A. (2020) *HitPredict statistics*. Available at: www.hitpredict.org/species_interactions.txt (Accessed: 11 January 2020).

Love, M. I. (2018) *rlog transformation producing outliers with very high log-transformed counts*, *Bioconductor*. Available at: https://support.bioconductor.org/p/105334/#105336 (Accessed: 28 December 2020).

Love, M. I., Huber, W. and Anders, S. (2014) 'Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2', *Genome Biology*, 15(12), pp. 1–21. doi: 10.1186/s13059-014-0550-8.

Lovén, J. *et al.* (2012) 'Revisiting Global Gene Expression Analysis', *Cell*, 151(3), pp. 476–482. doi: 10.1016/j.cell.2012.10.012.

Lu, R. *et al.* (2009) 'Systems-level dynamic analyses of fate change in murine embryonic stem cells', *Nature*. Nature Publishing Group, 462(7271), pp. 358–362. doi: 10.1038/nature08575.

Luce, R. D. and Perry, A. D. (1949) 'A method of matrix analysis of group structure', *Psychometrika*, 14(2), pp. 95–116. doi: 10.1007/BF02289146.

Luecken, M. D. *et al.* (2018) 'CommWalker: Correctly evaluating modules in molecular networks in light of annotation bias', *Bioinformatics*. Oxford University Press, 34(6), pp. 994–1000. doi: 10.1093/bioinformatics/btx706.

Lun, A. T. L., Chen, Y. and Smyth, G. K. (2016) 'It's DE-licious: A Recipe for Differential Expression Analyses of RNA-seq Experiments Using Quasi-Likelihood Methods in edgeR', in *Methods in Molecular Biology*. Humana Press Inc., pp. 391–416. doi: 10.1007/978-1-4939-3578-9_19.

Lund, S. P. *et al.* (2012) 'Detecting Differential Expression in RNA-sequence Data Using Quasi-likelihood with Shrunken Dispersion Estimates', *Statistical Applications in Genetics and Molecular Biology*, 11(5). doi: 10.1515/1544-6115.1826.

Luo, J. and Kuang, L. (2014) 'A new method for predicting essential proteins based on dynamic network topology and complex information', *Computational Biology and Chemistry*. Elsevier Ltd, 52, pp. 34–42. doi: 10.1016/j.compbiolchem.2014.08.022.

Magnani, M. and Rossi, L. (2013) 'Pareto Distance for Multi-layer Network Analysis', in Greenberg, A. M., Kennedy, W. G., and Bos, N. D. (eds) *Social Computing, Behavioral-Cultural Modeling and Prediction*. Springer, Berlin, Heidelberg, pp. 249–256. doi: 10.1007/978-3-642-37210-0_27.

Magnano, C. S. and Gitter, A. (2019) 'Automating parameter selection to avoid implausible biological pathway models', *bioRxiv*. doi: 10.1101/845834.

Margolin, A. A. *et al.* (2006) 'ARACNE: An Algorithm for the Reconstruction of Gene Regulatory Networks in a Mammalian Cellular Context', *BMC Bioinformatics*, 7(S1), p. S7. doi: 10.1186/1471-2105-7-S1-S7.

Marioni, J. C. *et al.* (2008) 'RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays', *Genome Research*, 18(9), pp. 1509–1517. doi: 10.1101/gr.079558.108.

Masuda, N. and Holme, P. (2019) 'Detecting sequences of system states in temporal networks', *Scientific Reports*, 9(1), p. 795. doi: 10.1038/s41598-018-37534-2.

Maza, E. *et al.* (2013) 'Comparison of normalization methods for differential gene expression analysis in RNA-Seq experiments', *Communicative & Integrative Biology*, 6(6), p. e25849. doi: 10.4161/cib.25849.

McCarthy, D. J., Chen, Y. and Smyth, G. K. (2012) 'Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation', *Nucleic Acids Research*. Oxford University Press, 40(10), pp. 4288–4297. doi: 10.1093/nar/gks042.

von Mering, C. *et al.* (2002) 'Comparative assessment of large-scale data sets of protein–protein interactions', *Nature*, 417(6887), pp. 399–403. doi: 10.1038/nature750.

Mete, M. *et al.* (2008) 'A structural approach for finding functional modules from large biological networks', *BMC Bioinformatics*, 9(Suppl 9), p. S19. doi: 10.1186/1471-2105-9-S9-S19.

Monteiro, P. T. *et al.* (2020) 'YEASTRACT+: a portal for cross-species comparative

genomics of transcription regulation in yeasts', *Nucleic Acids Research*, 48(D1), pp. D642–D649. doi: 10.1093/nar/gkz859.

Moradzadeh, K. *et al.* (2019) 'Analysis of time-course microarray data: Comparison of common tools', *Genomics*. Elsevier, 111(4), pp. 636–641. doi: 10.1016/j.ygeno.2018.03.021.

Mortazavi, A. *et al.* (2008) 'Mapping and quantifying mammalian transcriptomes by RNA-Seq', *Nature Methods*, 5(7), pp. 621–628. doi: 10.1038/nmeth.1226.

Mucha, P. J. *et al.* (2010) 'Community Structure in Time-Dependent, Multiscale, and Multiplex Networks', *Science*, 328(5980), pp. 876–878. doi: 10.1126/science.1184819.

Murphy, K. and Mian, S. (1999) *Modelling Gene Expression Data using Dynamic Bayesian Networks*, *Tech. Rep. MIT Artificial Intelligence Lab*.

NCBI (2021) *Citations of Cytoscape 2003 paper*. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC403769/citedby/ (Accessed: 21 January 2021).

Nguyen, H. *et al.* (2019) 'A Comprehensive Survey of Tools and Software for Active Subnetwork Identification', *Frontiers in Genetics*. Frontiers Media SA, 10, p. 155. doi: 10.3389/fgene.2019.00155.

Nueda, M. J., Tarazona, S. and Conesa, A. (2014) 'Next maSigPro: updating maSigPro bioconductor package for RNA-seq time series', *Bioinformatics*. Oxford University Press, 30(18), pp. 2598–2602. doi: 10.1093/bioinformatics/btu333.

Ong, I. M., Glasner, J. D. and Page, D. (2002) 'Modelling regulatory pathways in E. coli from time series expression profiles', *Bioinformatics*, 18(SUPPL. 1), pp. 241–248. doi: 10.1093/bioinformatics/18.suppl_1.S241.

Ono, K. *et al.* (2015) 'CyREST: Turbocharging Cytoscape Access for External Tools via a RESTful API', *F1000Research*, 4, p. 478. doi: 10.12688/f1000research.6767.1.

Oshlack, A. and Wakefield, M. (2009) 'Transcript length bias in RNA-seq data confounds systems biology.', *Biology direct*, 4, p. 14. doi: 10.1186/1745-6150-4-14.

Oughtred, R. *et al.* (2021) 'The BioGRID database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions', *Protein Science*, 30(1), pp. 187–200. doi: 10.1002/pro.3978.

Palla, G., Barabási, A.-L. and Vicsek, T. (2007) 'Quantifying social group evolution', *Nature*, 446(7136), pp. 664–667. doi: 10.1038/nature05670.

Park, C. *et al.* (2017) 'Systematic identification of an integrative network module during senescence from time-series gene expression', *BMC Systems Biology*. BioMed Central, 11(1), p. 36. doi: 10.1186/s12918-017-0417-1.

Patil, A. *et al.* (2013) 'Linking Transcriptional Changes over Time in Stimulated Dendritic Cells to Identify Gene Networks Activated during the Innate Immune Response', *PLoS Computational Biology*. Edited by J. J. Saucerman. Public Library of

Science, 9(11), p. e1003323. doi: 10.1371/journal.pcbi.1003323.

Patil, A. and Nakai, K. (2014) 'TimeXNet: identifying active gene sub-networks using time-course gene expression profiles.', *BMC systems biology*. BioMed Central, 8(Suppl 4), p. S2. doi: 10.1186/1752-0509-8-S4-S2.

Pawlina, J. (2021) *Graphs*, *Excalidraw*.

Phipson, B. *et al.* (2016) 'Robust hyperparameter estimation protects against hypervariable genes and improves power to detect differential expression', *The Annals of Applied Statistics*, 10(2), pp. 946–963. doi: 10.1214/16-AOAS920.

Pierrelée, M. *et al.* (2020) *TimeNexus: A Novel Cytoscape App to Analyze Time-Series Data Using Temporal MultiLayer Networks (tMLNs)*, *Research Square*. PREPRINT (Version 1). doi: 10.21203/rs.3.rs-133258/v1.

Powell, J. A. C. (2014) 'GO2MSIG, an automated GO based multi-species gene set generator for gene set enrichment analysis', *BMC Bioinformatics*, 15(1), p. 146. doi: 10.1186/1471-2105-15-146.

Przytycka, T. M., Singh, M. and Slonim, D. K. (2010) 'Toward the dynamic interactome: It's about time', *Briefings in Bioinformatics*, 11(1), pp. 15–29. doi: 10.1093/bib/bbp057.

Rajkumar, A. P. *et al.* (2015) 'Experimental validation of methods for differential gene expression analysis and sample pooling in RNA-seq', *BMC Genomics*. BioMed Central, 16(1), p. 548. doi: 10.1186/s12864-015-1767-y.

Rao, X. and Dixon, R. A. (2019) 'Co-expression networks for plant biology: Why and how', *Acta Biochimica et Biophysica Sinica*. Oxford University Press, pp. 981–988. doi: 10.1093/abbs/gmz080.

Rapaport, F. *et al.* (2013) 'Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data', *Genome Biology*, 14(9), p. R95. doi: 10.1186/gb-2013-14-9-r95.

Rau, A. *et al.* (2013) 'Data-based filtering for replicated high-throughput transcriptome sequencing experiments', *Bioinformatics*, 29(17), pp. 2146–2152. doi: 10.1093/bioinformatics/btt350.

Ravasz, E. (2002) 'Hierarchical Organization of Modularity in Metabolic Networks', *Science*, 297(5586), pp. 1551–1555. doi: 10.1126/science.1073374.

Rigaill, G. *et al.* (2016) 'Synthetic data sets for the identification of key ingredients for RNA-seq differential analysis', *Briefings in Bioinformatics*. Oxford University Press, 19(1), p. bbw092. doi: 10.1093/bib/bbw092.

Risso, D. *et al.* (2011) 'GC-Content Normalization for RNA-Seq Data', *BMC Bioinformatics*. BioMed Central, 12(1), p. 480. doi: 10.1186/1471-2105-12-480.

Ritz, A. *et al.* (2016) 'Pathways on demand: Automated reconstruction of human signaling networks', *npj Systems Biology and Applications*. Nature Publishing Group, 2(1), p. 16002. doi: 10.1038/npjsba.2016.2.

Robinson, M. D., McCarthy, D. J. and Smyth, G. K. (2010) 'edgeR: A Bioconductor package for differential expression analysis of digital gene expression data', *Bioinformatics*, 26(1), pp. 139–140. doi: 10.1093/bioinformatics/btp616.

Robinson, M. D. and Oshlack, A. (2010) 'A scaling normalization method for differential expression analysis of RNA-seq data', *Genome Biology*. BioMed Central, 11(3), p. R25. doi: 10.1186/gb-2010-11-3-r25.

Robinson, M. D. and Smyth, G. K. (2007) 'Moderated statistical tests for assessing differences in tag abundance', *Bioinformatics*, 23(21), pp. 2881–2887. doi: 10.1093/bioinformatics/btm453.

Robinson, M. D. and Smyth, G. K. (2008) 'Small-sample estimation of negative binomial dispersion, with applications to SAGE data', *Biostatistics*, 9(2), pp. 321–332. doi: 10.1093/biostatistics/kxm030.

Roca, C. P. *et al.* (2017) 'Variation-preserving normalization unveils blind spots in gene expression profiling', *Scientific Reports*. Nature Publishing Group, 7(1), p. 42460. doi: 10.1038/srep42460.

Sanchez, C. *et al.* (1999) 'Grasping at molecular interactions and genetic networks in Drosophila melanogaster using FlyNets, an internet database', *Nucleic Acids Research*. Oxford University Press, pp. 89–94. doi: 10.1093/nar/27.1.89.

Schaefer, M. H., Serrano, L. and Andrade-Navarro, M. A. (2015) 'Correcting for the study bias associated with protein–protein interaction measurements reveals differences between protein degree distributions from different cancer types', *Frontiers in Genetics*, 6(Aug), pp. 1–8. doi: 10.3389/fgene.2015.00260.

Schulz, M. H. *et al.* (2012) 'DREM 2.0: Improved reconstruction of dynamic regulatory networks from time-series expression data', *BMC Systems Biology*. BioMed Central, 6(1), p. 104. doi: 10.1186/1752-0509-6-104.

Schurch, N. J. *et al.* (2016) 'How many biological replicates are needed in an RNA-seq experiment and which differential expression tool should you use?', *RNA*. Cold Spring Harbor Laboratory Press, 22(6), pp. 839–851. doi: 10.1261/rna.053959.115.

Schwämmle, V. and Jensen, O. N. (2010) 'A simple and fast method to determine the parameters for fuzzy c–means cluster analysis', *Bioinformatics*, 26(22), pp. 2841–2848. doi: 10.1093/bioinformatics/btq534.

Seyednasrollah, F., Laiho, A. and Elo, L. L. (2015) 'Comparison of software packages for detecting differential expression in RNA-seq studies', *Briefings in Bioinformatics*, 16(1), pp. 59–70. doi: 10.1093/bib/bbt086.

Sha, Y., Phan, J. H. and Wang, M. D. (2015) 'Effect of low-expression gene filtering on detection of differentially expressed genes in RNA-seq data', in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*. Milan, Italy: IEEE, pp. 6461–6464. doi: 10.1109/EMBC.2015.7319872.

Shaffer, J. P. (1995) 'Multiple Hypothesis Testing', *Annual Review of Psychology*, 46(1),

pp. 561–584. doi: 10.1146/annurev.ps.46.020195.003021.

Shaik, F., Bezawada, S. and Goveas, N. (2015) 'CySpanningTree: Minimal Spanning Tree computation in Cytoscape', *F1000Research*, 4, p. 476. doi: 10.12688/f1000research.6797.1.

Shannon, P. *et al.* (2003) 'Cytoscape: A software Environment for integrated models of biomolecular interaction networks', *Genome Research*. Cold Spring Harbor Laboratory Press, 13(11), pp. 2498–2504. doi: 10.1101/gr.1239303.

Simillion, C. *et al.* (2017) 'Avoiding the pitfalls of gene set enrichment analysis with SetRank', *BMC Bioinformatics*. BMC Bioinformatics, 18(1), p. 151. doi: 10.1186/s12859-017-1571-6.

Smyth, G. K. (2004) 'Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments', *Statistical Applications in Genetics and Molecular Biology*, 3(1), pp. 1–25. doi: 10.2202/1544-6115.1027.

Smyth, G. K. *et al.* (2020) *Linear Models for Microarray Data User's Guide*, *Bioconductor*. Available at: https://www.bioconductor.org/packages/release/bioc/vignettes/limma/inst/doc/usersguide.pdf (Accessed: 1 January 2020).

Soneson, C. and Delorenzi, M. (2013) 'A comparison of methods for differential expression analysis of RNA-seq data', *BMC Bioinformatics*, 14, p. 91. doi: 10.1186/1471-2105-14-91.

Spies, D. *et al.* (2019) 'Comparative analysis of differential gene expression tools for RNA sequencing time course data', *Briefings in Bioinformatics*. Oxford University Press, 20(1), pp. 288–298. doi: 10.1093/bib/bbx115.

Spiliopoulou, M. (2011) 'Evolution in Social Networks: A Survey', in Aggarwal, C. C. (ed.) *Social Network Data Analytics*. Boston, MA: Springer US, pp. 149–175. doi: 10.1007/978-1-4419-8462-3_6.

Sprinzak, E., Sattath, S. and Margalit, H. (2003) 'How Reliable are Experimental Protein–Protein Interaction Data?', *Journal of Molecular Biology*. Academic Press, 327(5), pp. 919–923. doi: 10.1016/S0022-2836(03)00239-0.

Storey, J. D. and Tibshirani, R. (2003) 'Statistical significance for genomewide studies', *Proceedings of the National Academy of Sciences of the United States of America*, 100(16), pp. 9440–9445. doi: 10.1073/pnas.1530509100.

Subramanian, A. *et al.* (2005) 'Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles', *Proceedings of the National Academy of Sciences*, 102(43), pp. 15545–15550. doi: 10.1073/pnas.0506580102.

Tamayo, P. *et al.* (2016) 'The limitations of simple gene set enrichment analysis assuming gene independence', *Statistical Methods in Medical Research*, 25(1), pp. 472–487. doi: 10.1177/0962280212460441.

Tang, X. *et al.* (2011) 'A comparison of the functional modules identified from time

course and static PPI network data', *BMC Bioinformatics*, 12(1), p. 339. doi: 10.1186/1471-2105-12-339.

Tantipathananandh, C., Berger-Wolf, T. and Kempe, D. (2007) 'A framework for community identification in dynamic social networks', in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*. New York, New York, USA: ACM Press, p. 717. doi: 10.1145/1281192.1281269.

Teixeira, M. C. *et al.* (2018) 'YEASTRACT: an upgraded database for the analysis of transcription regulatory networks in Saccharomyces cerevisiae', *Nucleic Acids Research*. Oxford University Press, 46(D1), pp. D348–D353. doi: 10.1093/nar/gkx842.

Tjang, Y. (2020) *Software Architecture*, *Excalidraw*.

Trapnell, C. *et al.* (2010) 'Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation', *Nature Biotechnology*. Nature Publishing Group, 28(5), pp. 511–515. doi: 10.1038/nbt.1621.

Trapnell, C. *et al.* (2012) 'Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks', *Nature Protocols*, 7(3), pp. 562–578. doi: 10.1038/nprot.2012.016.

Trapnell, C. *et al.* (2013) 'Differential analysis of gene regulation at transcript resolution with RNA-seq', *Nature Biotechnology*. Nature Publishing Group, 31(1), pp. 46–53. doi: 10.1038/nbt.2450.

Tuncbag, N. *et al.* (2013) 'Simultaneous Reconstruction of Multiple Signaling Pathways via the Prize-Collecting Steiner Forest Problem', *Journal of Computational Biology*. Mary Ann Liebert, Inc., 20(2), pp. 124–136. doi: 10.1089/cmb.2012.0092.

Tuncbag, N. *et al.* (2016) 'Network-Based Interpretation of Diverse High-Throughput Datasets through the Omics Integrator Software Package.', *PLoS computational biology*. Public Library of Science, 12(4), p. e1004879. doi: 10.1371/journal.pcbi.1004879.

Valdeolivas, A. *et al.* (2019) 'Random walk with restart on multiplex and heterogeneous biological networks', *Bioinformatics*. Edited by A. Valencia, 35(3), pp. 497–505. doi: 10.1093/bioinformatics/bty637.

Villaveces, J. M. *et al.* (2015) 'Merging and scoring molecular interactions utilising existing community standards: Tools, use-cases and a case study', *Database*. Narnia, 2015, p. bau131. doi: 10.1093/database/bau131.

Vlaic, S. *et al.* (2018) 'ModuleDiscoverer: Identification of regulatory modules in protein-protein interaction networks', *Scientific Reports*. Springer US, 8(1), p. 433. doi: 10.1038/s41598-017-18370-2.

Watts, D. J. and Strogatz, S. H. (1998) 'Collective dynamics of "small-world" networks', *Nature*, 393(6684), pp. 440–442. doi: 10.1038/30918.

Westfall, P. H. (2011) 'Discussion of "Multiple Testing for Exploratory Research" by J. J. Goeman and A. Solari', *Statistical Science*, 26(4), pp. 604–607. doi: 10.1214/11-STS356C.

Will, T. and Helms, V. (2017) 'Rewiring of the inferred protein interactome during blood development studied with the tool PPICompare', *BMC Systems Biology*. BMC Systems Biology, 11(1), p. 44. doi: 10.1186/s12918-017-0400-x.

Williams, C. R. *et al.* (2016) 'Trimming of sequence reads alters RNA-Seq gene expression estimates', *BMC Bioinformatics*. BioMed Central, 17, p. 103. doi: 10.1186/s12859-016-0956-2.

Wingender, E. (2008) 'The TRANSFAC project as an example of framework technology that supports the analysis of genomic regulation', *Briefings in Bioinformatics*, 9(4), pp. 326–332. doi: 10.1093/bib/bbn016.

Wright, S. P. (1992) 'Adjusted P-Values for Simultaneous Inference', *Biometrics*, 48(4), p. 1005. doi: 10.2307/2532694.

Wu, Huanhuan *et al.* (2016) 'Efficient Algorithms for Temporal Path Computation', *IEEE Transactions on Knowledge and Data Engineering*, 28(11), pp. 2927–2942. doi: 10.1109/TKDE.2016.2594065.

Xiao, Y. *et al.* (2014) 'A novel significance score for gene selection and ranking', *Bioinformatics*, 30(6), pp. 801–807. doi: 10.1093/bioinformatics/btr671.

Xu, X. *et al.* (2007) 'SCAN: A Structural Clustering Algorithm for Networks', in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*. New York, New York, USA: ACM Press, pp. 824–833. doi: 10.1145/1281192.1281280.

Yosef, N. *et al.* (2009) 'Toward accurate reconstruction of functional protein networks', *Molecular Systems Biology*, 5(1), p. 248. doi: 10.1038/msb.2009.3.

Yosef, N. *et al.* (2011) 'ANAT: A Tool for Constructing and Analyzing Functional Protein Networks', *Science Signaling*. American Association for the Advancement of Science, 4(196), pp. pl1–pl1. doi: 10.1126/scisignal.2001935.

Yun, J. *et al.* (2020) 'Transcriptomic analysis of Chlorella sp. HS2 suggests the overflow of acetyl-CoA and NADPH cofactor induces high lipid accumulation and halotolerance', *Food and Energy Security*, p. e267. doi: 10.1002/fes3.267.

Zhang, B. and Horvath, S. (2005) 'A General Framework for Weighted Gene Co-Expression Network Analysis', *Statistical Applications in Genetics and Molecular Biology*, 4(1). doi: 10.2202/1544-6115.1128.

Zheng, H., Wang, H. and Glass, D. H. (2008) 'Integration of Genomic Data for Inferring Protein Complexes from Global Protein–Protein Interaction Networks', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(1), pp. 5–16. doi: 10.1109/TSMCB.2007.908912.

Zheng, W., Chung, L. M. and Zhao, H. (2011) 'Bias detection and correction in RNA-Sequencing data', *BMC Bioinformatics*, 12(1), p. 290. doi: 10.1186/1471-2105-12-290.

Zhou, X., Lindsay, H. and Robinson, M. D. (2014) 'Robustly detecting differential expression in RNA sequencing data using observation weights', *Nucleic Acids Research*,

42(11), pp. e91–e91. doi: 10.1093/nar/gku310.

Zyla, J. *et al.* (2017) 'Ranking metrics in gene set enrichment analysis: Do they matter?', *BMC Bioinformatics*. BioMed Central, 18(1), p. 256. doi: 10.1186/s12859-017-1674-0.

Zyprych-Walczak, J. *et al.* (2015) 'The Impact of Normalization Methods on RNA-Seq Data Analysis', *BioMed Research International*, 2015, pp. 1–10. doi: 10.1155/2015/621690.