Universiteit Leiden

Master Computer Science

End-to-end tool for DDEG analysis and network inference using time series bulk mRNA data

Name:	Nino Verwei
Student ID:	2098695
Date:	26/05/2025
Specialisation:	Bioinformatics
1st supervisor:	Dr. K.J. Wolstencroft
2nd supervisor:	Dr. P. Ding
Daily supervisor:	Dr. R.C.X. Leong

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

1 Introduction

Cellular processes refer to the biochemical activities in a cell to sustain life. These processes, such as cellular differentiation and immune responses, inherently unfold over time. In plants, dynamic immune processes play a crucial role in defence mechanisms against pathogens. Over millions of years, plants have evolved a two-tiered immune system. The first tier, known as Pattern-Triggered Immunity (PTI), is activated at the plant cell surface level, where Pattern-Recognition Receptors (PRRs) recognise pathogen-associated molecular patterns (PAMPs) found in various pathogens. One well know example of a PAMP is flagellin found on bacteria. Once PRRs are activated, a series of intracellular signaling cascade involving kinases and reactive oxygen species occurs, leading to transcriptional reprogramming of defence responses and plant growth to tackle a pathogen infection. In numerous cases, pathogens have evolved to produce effectors which block PTI through Effector Triggered Susceptibility(ETS). In turn, plants have also evolved nucleotide-binding Leucine-rich Repeats (NLRs) that recognise directly or indirectly effectors. Upon recognition, NLRs lead to the activation of the second layer of immune response, known as Effector-Triggered Immunity (ETI). ETI activation leads to transcriptional reprogramming of the cell and a localized cell death response at the site of infection. ETI often leads to stronger and more robust resistance against pathogens[39].

While PTI is more straightforward to study due to simple receptor/ligand-based PAMP detection at the plant surface, ETI mechanisms are more complex as effectors and NLRs are often diverse and rapidly evolving. Many ETI mechanisms remain to be discovered. Research into ETI can lead to a better understanding of plant immunity, enabling researchers to identify new genes or mechanisms in plants. With more novel genes involved in ETI identified, which allows for a more complete understanding of ETI, genetic breeding of disease-resistant plants can eventually lead to environmentally friendly farming systems, as these new breeds will require fewer pesticides.

To comprehend the mechanisms of a complex cellular process like ETI in plant immunity, various methods, from microscopy-based techniques to biochemical approaches are used. One of the modern techniques, RNA sequencing, quantifies RNA in a cell, thereby generating gene expression data, and allowing the study of changes in gene expression. A Differentially Expressed Gene (DEG) is defined as a statistically significant change observed, on a gene level, between two conditions[3]. To research the influence of a treatment (treat) on a specimen, it is compared to its untreated (mock) counterpart. Through statistical analysis, the researchers ensure that the prediction of (in)significance can be trusted. Different statistical methods require a different number of replicates. Replicates are a necessity to reduce the influence of biological and technical variation.

Investigating temporal gene expression changes in biological systems is challenging as it requires analysing multiple time points simultaneously rather than identifying differences at a single time point for gene expression data. There are two analytical approaches towards Temporal Differentially Expressed Gene (TDEG) analysis, static and dynamic, with each having their pros and cons.

Static differentially expressed gene (SDEG) assumes independence between time points in the

time series. It requires the time-series datasets under different conditions to consist of the same number of time points taken at the same time. This method becomes unviable if samples are taken at different time points. This could be the case when using data from different sources. When conditions in data have the same dimensions, this method is more straightforward and uses less time and resources to determine. However, it can fail to give a robust prediction on the temporal biological process considered.

Dynamic Differentially Expressed Gene (DDEG) analysis better captures biological systems but is harder to determine. It assumes interdependence between time points which, in theory, allows for a non equal number of time points and/or samples taken at different time points. This would allow new experiments to be performed using existing data from different sources, that might not have the same size, allowing for the repurposing of previous research.

A few steps need to be followed to research the temporal gene expression changes in biological systems. First, the Temporal Differentially Expressed Gene (TDEG) must be extracted from the time series gene dataset. Using generalised temporal Gene Expression Patterns (GEP) obtained through clustering, a Gene Expression Pattern Regulatory Network (GEPRN) can be constructed. In a GEPRN, regulatory pattern-to-pattern interactions are inferred. With this GEPRN, we can discover new key regulatory genes and their downstream processes, giving us more insight into complex biological systems.

Tools for the dynamic method to analyze time series data are lacking as it requires every step of the process to be dynamic, taking the interdependence of time points into account. The steps to incorperate dynamic methods are DDEG prediction, clustering and network inference. If a single method is not dynamic the ouput of that process cannot be interpreted as DDEG. Existing tools require a certain number of replicates/time points or are built upon static methods[24]. Besides the issue related to the dynamic analysis methods, there are many options for data pre-processing. Pre-processing often requires analysing and visualising the data in multiple rounds, tweaking the pre-processing criteria slightly for every new dataset. This makes it difficult for new people to start preprocessing their data. This results in a complex landscape where every dynamic and pre-processing tool presents itself as the solution, having different applications and limitations while not covering the full end-to-end process of DDEG analysis[24]. An easy to use start-to-end pipeline is therefore crucial to assist lab scientists in their research, improving reproducibility while preventing lab scientist to figure out the complex landscape of DDEG analysis themself.

This thesis aims to address the following question: Can an end-to-end pipeline for non-replicated time series bulk mRNA data infer gene expression pattern regulatory network reliably representing complex time-dependent cellular processes such as immunity? The DDEG analysis field is convoluted by a variety of individual tools addressing only parts of the workflow. The tool created in this thesis provides researchers with an accurate, modular and easy-to-use end-to-end DDEG analysis pipeline. This is done by adapting and improving upon a previous PhD thesis of Xin He [12]. The focus of this thesis is to improve the existing methodology created by Xin He, improve usability and develop additional methodology for an easy-to-use pipeline. This will be done by training a Long Short Term Memory neural network (LSTM)[13], a machine learning model well known for its ability to handle time series data, and hierarchi-

cal clustering with a Dynamic Time Warping (DTW)[4] distance matrix. DTW is a dynamic method to calculate distance between dynamic time series data.

In this thesis, we will discuss the original pipeline of Xin He and go into the different modular parts of the improved pipeline. This includes data preprocessing, dataset labelling, neural network training, gene pattern clustering, Gene Expression Pattern Regulatory Network (GEPRN) inference, and network analysis. In the end the pipeline will be applied to two datasets resulting from ETI induction experiments on the *Aradopsis thaliana*.

2 Related work

2.1 Previous work

This thesis is based upon the PhD dissertation of Xin HE [12]. A novel analysis pipeline was presented to process and analyse time series gene expression data. The pipeline consisted of five main parts: preprocessing, DDEGs prediction, clustering, network inference and network analysis. Figure 1 shows a high-level overview of the files used by the pipeline.

The pipeline starts with pre-processing of the raw RNAseq data, which is done by filtering and normalising counts to Transcripts Per Million (TPM). From the data, two-condition time series, three different features were extracted: statistical feature, multidimensional feature and time series feature. The statistical feature contains information on the difference in mean and variance between the two conditions. The multidimensional feature contains distance metrics such as Euclidean distance, Manhattan distance and cosine distance, these distance metrics provide information on the similarity of the time series data. The time series feature is the DTW distance, this method can perform a time shift to determine the smallest distance between two time series. The amount of time the method can shift one time series depends on whether a constraint is given[30].

After preprocessing, the reduced data is put through a pre-trained XGBoost model (Extreme Gradient Boosting model) for DDEG identification[5]. An XGBoost model is an ensemble of decision trees, where each tree is trained to correct the errors of the previous ones. Instead of training a single strong model, XGBoost adds multiple weak models to itself, improving performance through boosting. The XGBoost model was trained on 200 genes. The genes were annotated by two field experts to form a dataset. Two criteria were used to annotate the genes into three groups. The first criterion was a difference in gene expression between the two conditions, and the second was similarity in temporal patterns. When both show a difference, the gene was annotated as DDEG and when both criteria show little difference, the gene was annotated as no DDEG. The final annotation was unsure, this was used to label genes that showed a difference in one of the criteria but little difference in the other.

Next a K-means algorithm was used to cluster the predicted DDEGs into generalised gene patterns, and the optimal number of clusters was determined using the elbow method [37]. The elbow method is a heuristic method to determine the number of clusters when this is unknown. To find the optimal number of clusters, the inertia is set out against the number of clusters. The name of the method stems from the shape of the curve that forms when determining the optimal number of clusters. Finally, the gene patterns were fed to the Bayesian Inference of Networks using Gaussian process dynamical models (BINGO) algorithm which infers a gene pattern network [1]. This algorithm "tries" out different network structures over many iterations and determines the likelihood of the network. Through this iterative process, the probability of a link between clusters is determined and a network is inferred.

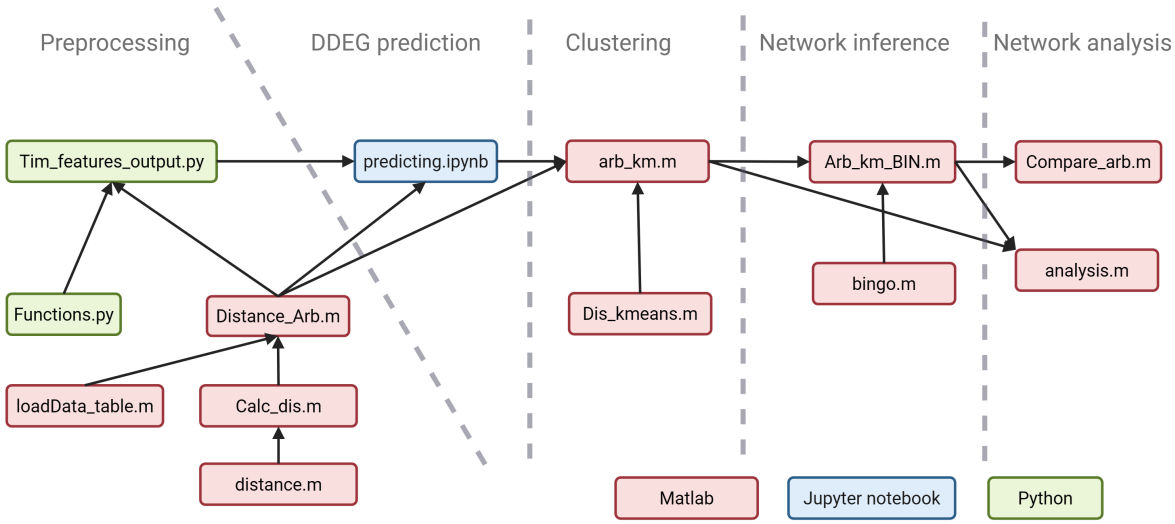

Figure 1: Schematic workflow overview of the previous pipeline created by Xin He[12]. Green represents python, blue Jupyter Notebook and red Matlab.

Adapting previous work

Taking this pipeline as the basis we can identify points of improvement. The current code spans multiple files which have to be run one-by-one. Through workflow orchestration, having a central "main" file which calls parts of the pipeline when specified, the pipeline as a whole becomes less confusing to use. The current pipeline reduces the input data into 3 features specified above before doing DDEG prediction. By reducing the data it is possible to lose crucial information making the DDEG prediction more difficult and the result less reliable. By changing the prediction model from XGBoost to an LSTM model the "raw" gene time series expression data can be input instead of its reduced form. An LSTM makes use of a memory and forget gate, and learns over iterations what to remember and what to forget. The DTW distance function currently does not have a constraint set, by setting a constraint the time warping is limited to prevent unwanted overlapping of genes. The current pipeline is trained on 200 genes, labelled by two people, with cross-validation but without test set. A new bigger dataset will be generated through multiple labelling rounds and a definition of non-statistical DDEG will be defined by analysing the labelled data. The current clustering method is k-mean clustering, this requires the clusters to be known before clustering. As there is no current idea what the ideal number of clusters there should be it is approached by using the elbow method. This can be improved upon by changing the clustering method to hierarchical clustering. Here a cluster distance can be set and over time the range can be optimised resulting in a better approximation of the number of clusters. The network inference method BINGO is currently run with 20000 iterations, the BINGO paper recommends at least 50000 iterations [1]. As a higher iteration number results in more accurate network inference, 300000 iterations will be used.

2.2 Related tools

There is no established standard for data processing or downstream analyses in the dynamic analysis of time-series gene expression data. Researchers often create their own methods for their analysis, with each method having its own requirements and limitations, making it difficult to compare findings across studies. Some of the created tools are built upon static tools, others are outdated or written in another programming language. This creates a complex landscape where a researcher has to take into account replicates, conditions, timespan and goal when looking into tools for their analysis[25].

Taking a recent analysis pipeline like NetSeekR[33], it relies on SDEG identification tools like edgeR[29]. It also uses older tools such as Weighted Correlation Network Analysis(WGCNA)[18] and Dynamic Regulatory Events Miner(DREM)[31] for network inference. There are newer network building tools, like BINGO, which can better capture the dynamic nature of time-series data. Mathematical approaches such as lmms[34] and GPrank[36] among many assume the data distribution and/or reduce the data to a simpler form. This is confounded when the number of sampling time points is limited. This is likely to result in not being able to capture all the complexities of the dynamic gene expression system being modelled. Additionally. Mathematical approaches based on statistics are dependent on the number of replicates, i.e. higher number of replicates might result in a lower p value. This may result in false positives in a dataset just by increasing the number of replicates. While there are multiple different parametric/statistical based tools in the field, they might not be able to find all hidden connections in the dynamic data. While nonparametric tools are becoming more prevalent, they still make up only a small portion in the field.

3 Methods

This chapter describes the methods for both developing the pipeline and executing the pipeline on research data. The section on pipeline development delves into the parts of the pipeline and what methods are used. The section applying pipeline defines the data and variables which are inputted into the pipeline.

Images in this thesis are outputted from the pipeline or made in BioRender.

3.1 Pipeline development

3.1.1 Programming languages and packages

Python 3.8, R 4.2.2 and matlab R2023a were used in this thesis. Table 1 shows the installed packages, versions, and part of the pipeline in which they were used.

Table 1: Packages, their versions and part in the pipeline where they were used.

Package	**Version**	**Used in**
Pandas[22]	1.5.3	Main, preprocessing, clustering
Numpy[11]	1.24.3	Main, DDEG identification, clustering
Tensorflow[19]	2.13.0	Main, DDEG identification
Optuna[2]	3.6.1	DDEG identification
Seaborn[38]	0.13.2	Main, preprocessing, clustering
Matplotlib[14]	3.7.5	Main, preprocessing, clustering
Rpy2[10]	3.5.17	preprocessing
Scikit-learn[27]	1.3.2	Preprocessing, clustering
Tslearn[35]	0.6.3	Clustering
Goatools[16]	1.4.12	Clustering
Gprofiler[17]	1.0.0	Clustering
Umap-learn[21]	0.5.7	Clustering
pyvis[28]	0.3.2	Network analysis
Dash[15]	2.18.2	Network analysis

3.1.2 Data

In the making of the pipeline, two time series gene expression datasets were used[8]. Both consist of bulk mRNA-seq data of a Super-ETI(SETI) line from the *A.thaliana* plant. One dataset consisted of plants which were injected with 50 µM β-estradiol, dissolved using DiMethylSulfOxide (DMSO), to activate the ETI response in the SETI plants, this is the treat dataset. The other dataset consisted of plants injected with only DMSO solution, the mock. With the only difference between the two groups being the β-estradiol, the effect of this ETI inducer can be studied. The experiment setting is visualised in Figure 2. Both time series consist of 13 time points, taken at half-hour intervals between 0 and 4 hours, then at 1-hour intervals Between 4 and 8 hours.

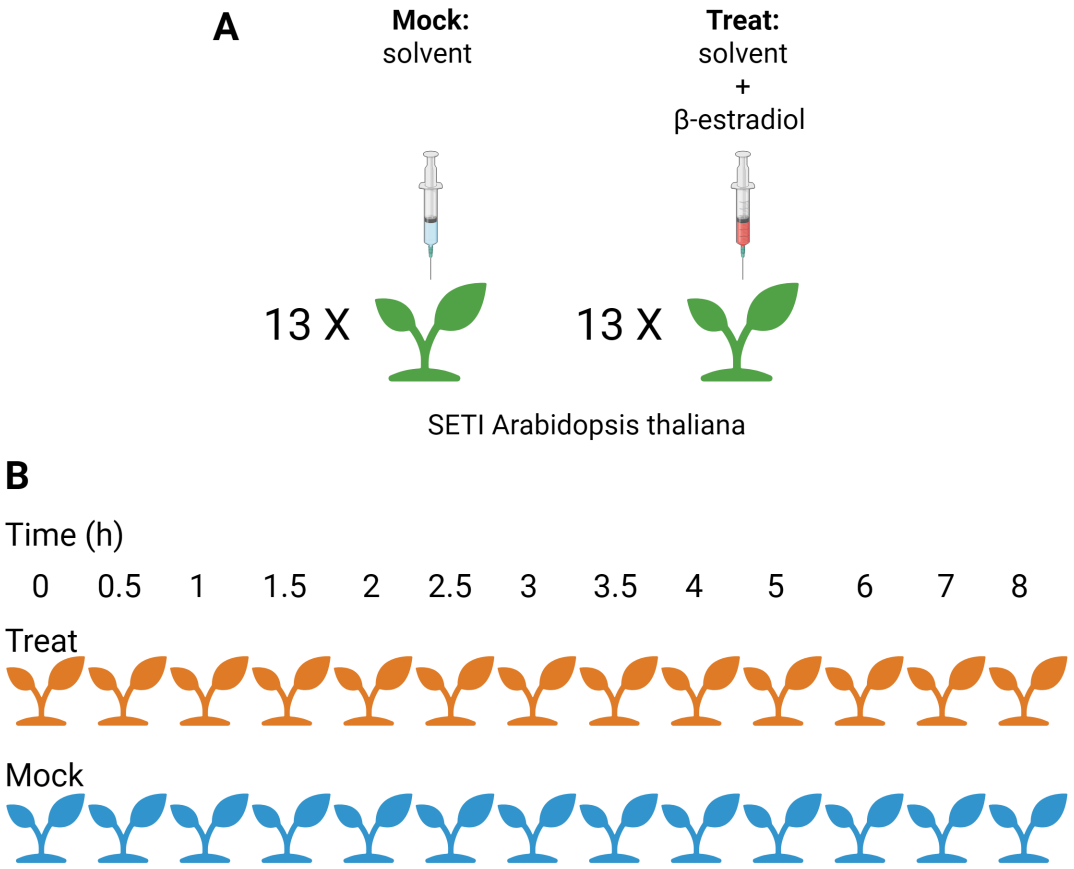

Figure 2: A) 13 plants were injected with a DiMethylSulfOxide(DMSO) solvent forming the mock group and the other with a solvent and ETI inducer forming the treat group. The plants used in this experiment are from a SETI line of Arabidopsis thaliana plants. B) At specified times, the mock and treat plants were harvested after injection.

3.1.3 Galaxy

The sequence files were provided in FASTQ format. Using a workflow created in Galaxy[6], the files were converted into a gene counts file. The workflow can be found through the following link: `https://usegalaxy.org/u/nino_ver/w/ddeg-aligment-tximport`. The galaxy workflow converts FASTQ data into gene counts by using Salmon mapping[26] and tximport[32] to convert the files into read count data. The workflow is shown in Figure 3. The following inputs are required: a paired list data collection containing all the FASTQ files of the data, a transcript reference file for the salmon aligner and a transcript-to-gene file for tximport.

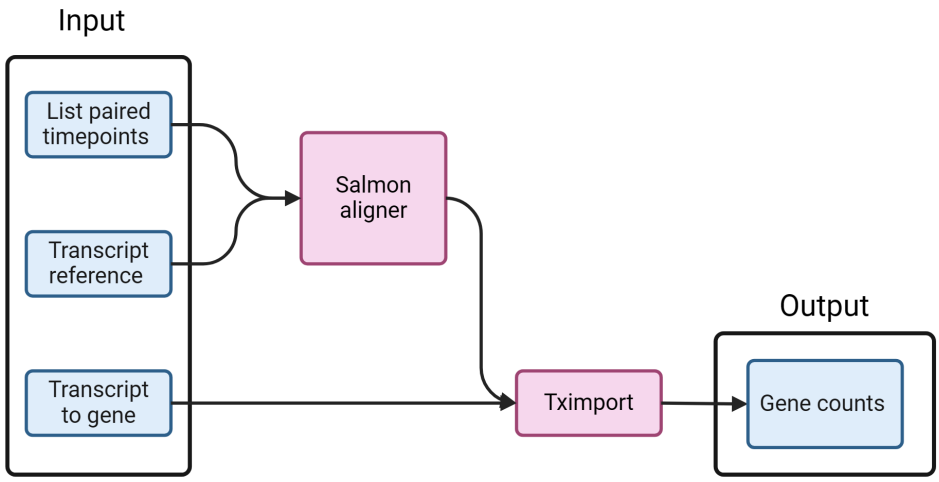

Figure 3: Galaxy workflow visualization. Blue represents a type of data and red shows the tools.

3.1.4 Normalisation methods

After the galaxy workflow, the resulting gene count data is provided to the preprocessing step. The main idea of preprocessing is to make samples comparable, filter out low-expressed genes and reduce variation in the data. There are two types of variation: biological and technical. Biological variation can be reduced by repeating the experiment and creating replicates. Technical variation can be reduced by normalising the data. Through preprocessing the gene counts are converted into gene expression values. Normalisation is applied during preprocessing and is important to make either within-sample or between-sample comparisons.

Normalisation combats technical variation, there are 3 well-known: library size, gene length, and RNA composition[40]. For between-sample comparison, the number of reads per sample must be roughly equal to that of the other samples, so normalising library size is important. Outlier genes can distort the library size of the samples, so normalising for RNA composition is important as well. For within-sample comparison, normalising for gene length is important to be able to compare genes with each other. Depending on the approach and aim of the researcher, both types of normalisation methods can be used interchangeably.

Counts Per Million (CPM), Fragments Per Kilobase Million(FPKM), and Transcripts Per Million (TPM) are examples of within-sample normalisation methods, while Trimmed Mean of M-values (TMM) is an example of between-sample normalisation. In this pipeline TMM normalisation will be used.

3.1.5 Preprocessing

Preprocessing consists of 5 steps that convert and filter FASTQ files into Gene expression values, which are visualised in Figure 4. The first step uses the galaxy workflow described previously and converts FastQ files to raw gene counts(1. alignment). This is followed by a filter step which filters out genes with average counts, over the time series, below a set threshold and genes in which one sample contributes more than a set fraction of the total count(2. Low count filter). The filtered gene counts are then normalised using the TMM normalisation method, leading to gene expression values(3. TMM normalisation). The last step is the repeating of the filter step noted earlier(4. Low gene expression filter). After the galaxy alignment, the data is visualised for each step as shown in Figure 4(5. Data visualisation).

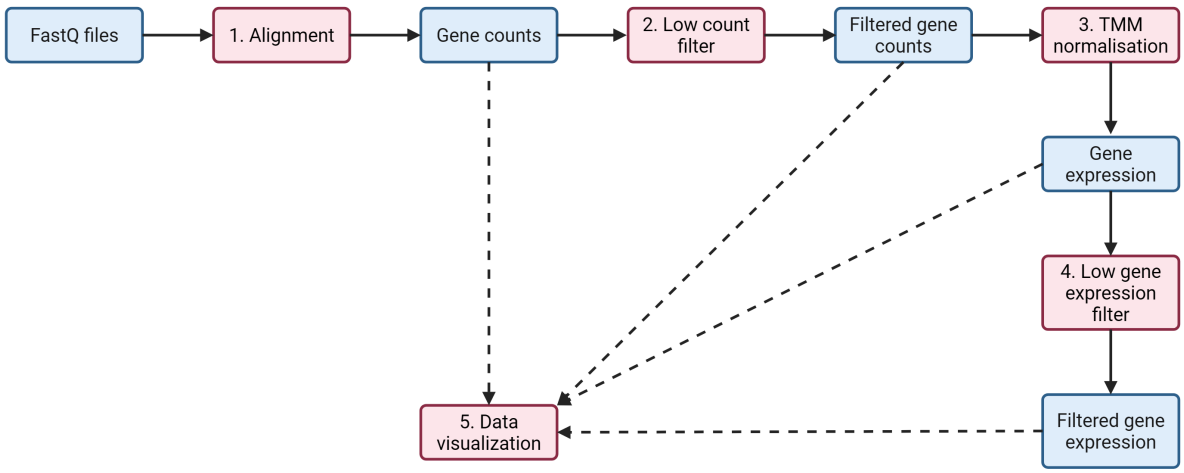

Figure 4: Schematic overview of preprocessing workflow. Red boxes indicate a performing step and blue boxes indicate a form of data.

To visualise the influence of each step during processing a figure consisting of multiple subplots was made to show different aspects of the data. Table 2 shows an overview of the different kinds of subplots.

Table 2: Information of subplots and their axis of preprocessing visualisations.

Subplot	Plot	Axis
A	Time series gene data, cluster 0, of 2-means clustering	Log2-Fold change set out against time
B	Time series gene data, cluster 1, of 2-means clustering	Log2-Fold change set out against time
C	PCA of genes coloured by 2-means clustering	PCA component 1 set out against PCA component 2
D	PCA of genes coloured by 4-means clustering	PCA component 1 set out against PCA component 2
E	PCA of samples with transparency as time	PCA component 1 set out against PCA component 2
F	Heatmap of the number of samples per gene falling below the set threshold	Samples per gene in mock set out against samples per gene in treat
G	Histogram of mean expression per gene over mock and treat	Gene count set out against mean expression
H	Distribution of contribution sample counts of mock and treat aggregated	Sample counts set out against fraction of contribution
I	Correlation plot of samples ordered by condition	Samples set out against samples
j	Correlation plot of samples ordered by time	Samples set out against samples
K	Boxplot of gene expression per sample of mock	gene expression set out against mock samples
L	Boxplot of gene expression per sample of treat	Gene expression set out against treat samples

3.1.6 Dataset Annotation

To determine whether a gene is differentially expressed, statistical analysis is used to determine if there is a significant change in expression. Since the data in this thesis contains no replicates, we cannot determine the statistical significance of the genes in the dataset. The current goal for the pipeline is to create an annotated dataset with the data provided and lay the groundwork for identifying statistical DDEGs when multi-replicate data is used.

The data was manually annotated by multiple domain experts and consisted of randomly chosen genes. The domain experts were provided with different types of visualized data of the genes before annotation. The final annotation was determined through the majority or complete agreement of the annotators. After each round, the annotation criteria and gene visualisation were altered to improve annotation. Table 3 shows an overview of the different subplots during annotation, and Figure 5 shows an example of the data provided for a gene during annotation. In total, there were four rounds of annotating. An overview of the annotation criteria can be seen in Table 4. In the final round, the set of instructions were given to annotate the final training dataset.

Table 3: Information of subplots of the figure provided to the annotator(Fig 5).

Subplot	Content	Axis
A	Lineplot of mock and treat gene expression	Gene expression set out against time
B	NCSRM fitted line plot of mock and treat gene expression	Gene expression set out against time
C	Line plot of foldchange with guidelines for labelling	Log2foldchange set out against time
D	Slope line plot of gene expression (mock and treat) and foldchange	Acceleration set out against time
E	Text containing properties of gene	None

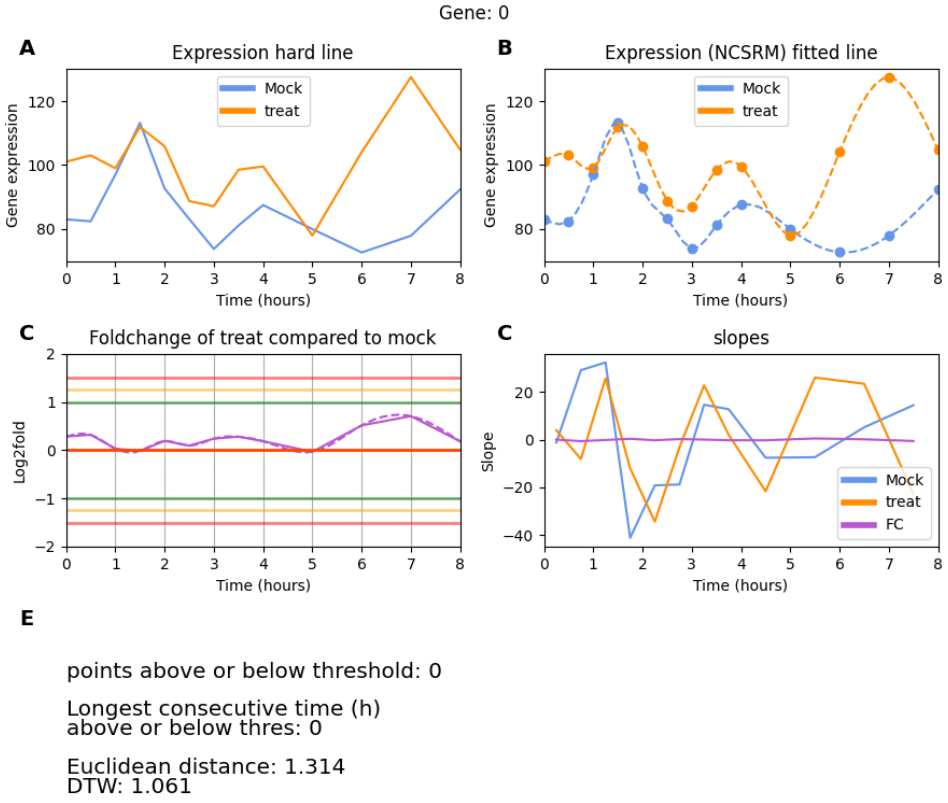

Figure 5: Example of images annotators see during annotation. Information on subplots are given in table 3.

Table 4: Instructions during each round of annotating.

Round	**Instructions**
1	- Using only expression plots of mock and treat determine DDEG or no DDEG - Take expression shape and distance into account with possible shift
2	- Take expression shape and distance into account with possible shift - More than 3 points above threshold and/or 2 consecutive hours above threshold ->likely DDEG - DTW distance below 0.6 and/ or euclidean distance below 0.6 ->likely no DDEG
3	- instructions with example images were shown of expression range, shape and log 2 fold - Instructions from previous rounds
4	- instructions with example images were shown of expression range, shape, fitted line artefacts and log 2 fold change - instructions of previous rounds

3.1.7 Neural network

To predict which genes are DDEGs, a Long Short Term Memory neural network (LSTM)[13] was trained. This model specialises in handling time series data, since an LSTM unit contains a cell state(memory) and forget gate, the node learns through training which patterns to remember or forget. When an LSTM processes time series data, it goes through it time point by time point. At every step, it 'decides' what to remember and what to forget.

The LSTM was trained using empirical experimentation and a hyperparameter optimisation package called Optuna[2]. The dataset used for this part consisted of 1004 samples. The data was split into 2 parts: 1 part, of 251 samples, for testing and the other part, 753 samples for a 3-fold cross-validation training.

The model was made using TensorFlow. In the exploration part of hyperparameter optimisation the model contained one or two LSTM layers, one or two dense layers and one or two dropout layers. All the layers were initialised as is unless specified: The first LSTM layer had return sequences set to False and the final dense layer had a sigmoid activation function. A binary cross entropy loss was used with the adam optimiser.
For simplification, only the last Optuna study and the final model will be discussed in depth. The exploration resulted in a preference for a small model with one layer of each type with a low learning rate of 0.00025.

The final optimisation round was an Optuna study of 50 trials with a TPESampler set at seed 4124. The settings are shown in Table 5. This resulted in two settings with a 0.280 and 0.286 loss. On inspection of the loss and false count figures, the curves are unstable. The decision was made to lower all hyperparameters to increase curve stability.

Table 5: The setting of trial parameters for final Optuna optimization. The trial was performed using a TPESampler with a trial budget of 50.

setting	(min, max, step)
Nodes LSTM layer	(5, 160, 5)
Nodes dense layer	(5, 80, 5)
Dropout rate	(0, 0.2, 0.01)

3.1.8 Clustering

Hierarchical clustering with a custom distance matrix was used. This distance matrix was made using log2-Fold Change(FC) time series data. By setting a value for the parameter clustering distance, clusters are obtained. By averaging the mock, treat and log2-FC time series data we can obtain their respective centroids representing gene patterns. The genes for each cluster are put into Gprofiler and GOATTOOLS for cluster annotation. The gene patterns are min-max normalised before being put into the network inference step.

3.1.9 Network inference

To infer a GEPRN from the data the BINGO algorithm is used [1]. It takes min-max normalised centroids of clustering as input, together with an iteration parameter. The higher this number is the more accurate the predicted network should be, this does lead to a longer running time.

3.1.10 Network analysis

In this step the inferred network is combined with a reference gene-to-gene network. The reference network is taken from ChIP-Hub[9]. This gives an overview of known gene interactions significant to the treatment and gene hubs of interest.

When data from 2 experiments is available it is possible to compare GEPRN networks in a visualisation. This can be done in four different ways: up/down-regulation, perturbation, GO pie chart and inter-network similarity.

3.2 Applying pipeline

3.2.1 Experiments data

The pipeline is applied to data from two experiments, in both the ETI pathways is induced by an effector. These effectors are AvrRps4(S4) and AvrRpt2(T2). The goal is to compare the networks and discover shared and unique genes of interest in the ETI pathway. The machine learning model was trained using S4 Data. The data of T2 was generated in the same experimental setting as S4 [8].

3.2.2 Pipeline variables input

Table 6. shows the variable inputs for experiments in this thesis.

Table 6: Variables and their value for two experiments: S4 and T2.

Variable name	**S4**	**T2**
To_run	[1,2,3,4,5,6]	[1,2,3,4,5,6]
Mean_threshold	10	10
Fraction_threshold	0.95	0.95
Make_DTW_matrix	True	True
Constraint_radius	1	1
Cluster_distance	5.5	5.5
Bingo_iterations	300000	300000
Link_threshold	0.8	0.8
Perturbation_threshold	0.8	0.8
Reference_network	True	True
Second_network	True	True
Input_file_name_1	AvrRps4	AvrRps4
Input_file_name_2	AvrRpt2	AvrRpt2
Input_file_name	AvrRps4	AvrRpt2
DDEG_model_name	lstm_model.keras	lstm_model.keras

3.2.3 DDEGs comparisson

After DDEG analysis the DDEG are overlapped with the DDEGs from 2 different tools: the original pipeline of Xin He[12] and the tool SplineTimeR[23]. The overlap is visualised on a Venn diagram.

4 Results

This chapter describes the results of pipeline construction and execution. The first section shows the structure of the pipeline and the output files produced when running, as well as labelling results and neural network training. The second section delves into the biological aspect when applying the pipeline on two ETI inducers on the *A. thaliana*. Here files outputted by the pipeline will be discussed.

4.1 Pipeline development

The main pipeline consists of six main parts: sample sequencing, data preprocessing, DDEG analysis, GEP clustering, GEPRN inference, and GEPRNs analysis. Except for sample sequencing each part is modular and can be exchanged for another, improved, method if needed. Each part consists of 3 main actions: load data, perform function, and save data and/or visualisation. Two parts of the pipeline are not in the main pipeline: Data annotation and LSTM training. These two parts are only performed when there is no excisting trained model for the data size/requirement. A schematic overview of the pipeline is shown in Figure 6 and an overview of files and folders is shown in Figure 7.

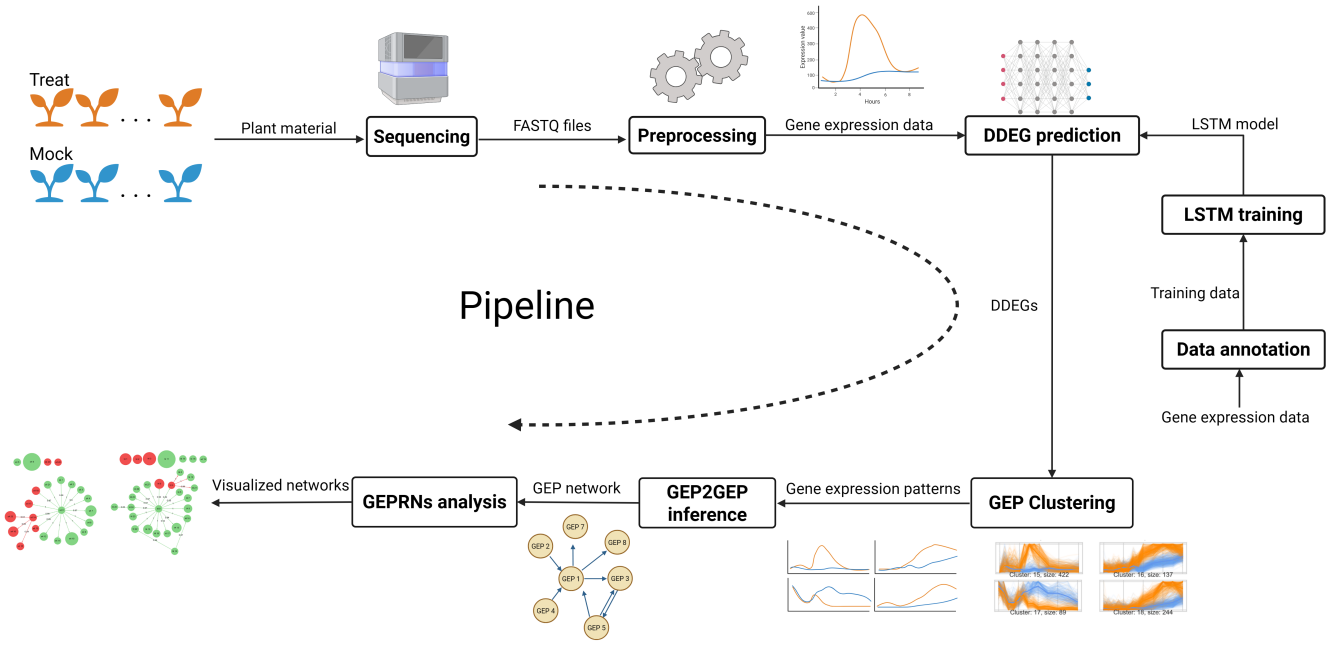

Figure 6: A schematic overview of the created pipeline.

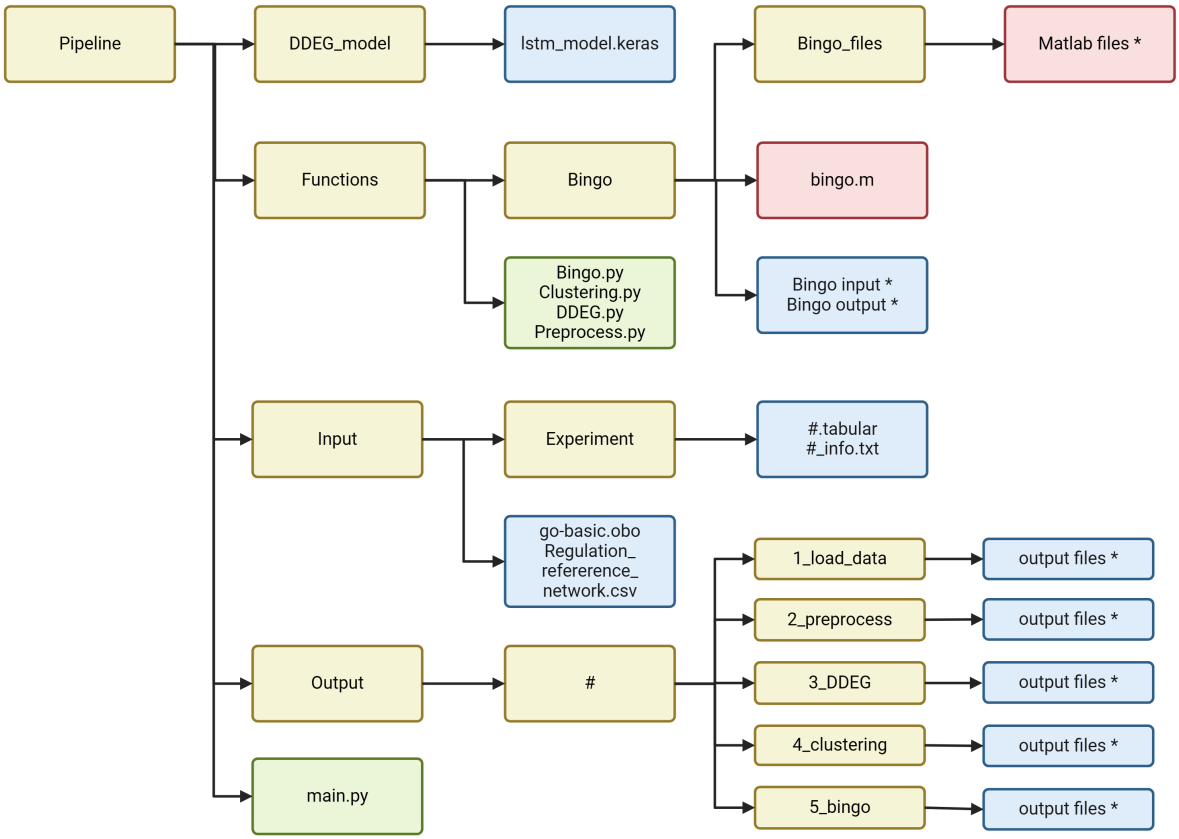

Figure 7: Schematic overview of directory of pipeline and files. Yellow indicates folders, green python files, red indicates matlab files and, blue indicates other files. # indicates experiment name and * indicates multiple files.

4.1.1 Input variables

Table 7 shows the variables the pipeline requires together with the input type and an explanation of the variable.

Table 7:

Variable name	**Input type**	**Explanation**
To_run	List of integers	What part of the pipeline do you want to run? Please include all the numbers in the 'to run' list. 1 ->Load data 2 ->Preprocess data 3 ->DDEG 4 ->Clustering 5 ->Network inference 6 ->Network analysis
Mean_threshold	Integer	Gene mean values below this threshold are filtered out.
Fraction_threshold	Float (0-1)	Sample contribution to the mean above this threshold are seen as outliers.
Make_DTW_matrix	Boolean	If DTW matrix needs to be (re)made, for the first time always set to True.
Constraint_radius	Integer	DTW sakoe_chiba radius constraint.
Cluster_distance	Float	Distance that determines the amount of clusters.
Bingo_iterations	Integer	Number of iterations used in the network inference algorithm.
Link_threshold	Float (0-1)	Acceptance threshold for cluster-to-cluster link.
Perturbation_threshold	Float (0-1)	Threshold for perturbation identification.
Reference_network	Boolean	Overlap inferred network with reference network.
Second_network	Boolean	A Second network is present for network comparison analysis.
Input_file_name_1	String	Name of the first experiment for network comparison analysis.
Input_file_name_2	String	Name of the second experiment for network comparison analysis.
Input_file_name	String	Name of the experiment.
DDEG_model_name	String	Name of the DDEG identification model.
inter_cluster_link_threshold	float (0-1)	Theshold for simmilarity between network cluster for comparison analysis.

4.1.2 Data preprocessing

Raw data in the form of FASTQ files are used as input for preprocessing. This resulted in different files as output: visualisations in PNG form, Figure 9, and data in CSV form. Figure 8 shows a schematic of preprocessing steps and their output files, and Table 8 shows the files and their content.

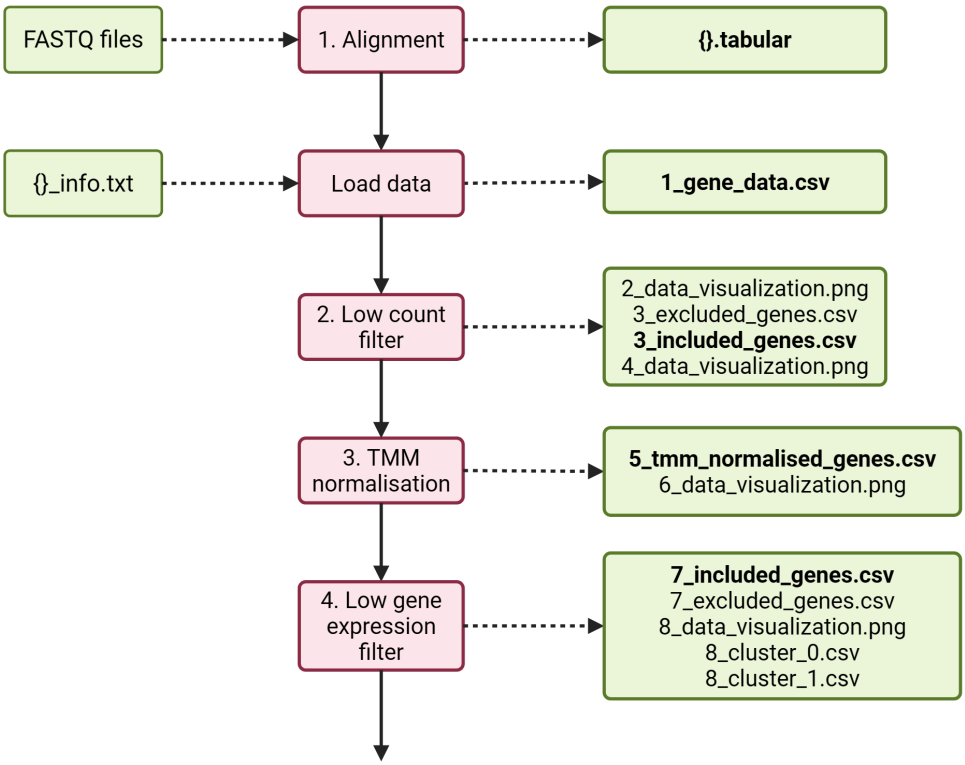

Figure 8: Schematic overview of preprocessing with input, output and variables. The files in bold are used as input in their representative next step.

Table 8: Filename and content of preprocessing output.

File name	**Content**
1_gene_data.csv	Time series count for genes
2_data_visualization.png	Visualizing data pre-filter
3_excluded_genes.csv	Filtered out genes with their time series counts
3_included_genes.csv	Filtered genes with their time series counts
4_data_visualization.png	Visualization data post filter
5_tmm_normalised_genes.csv	Time series gene expression time series data
6_data_visualization.png	Visualization data post normalization

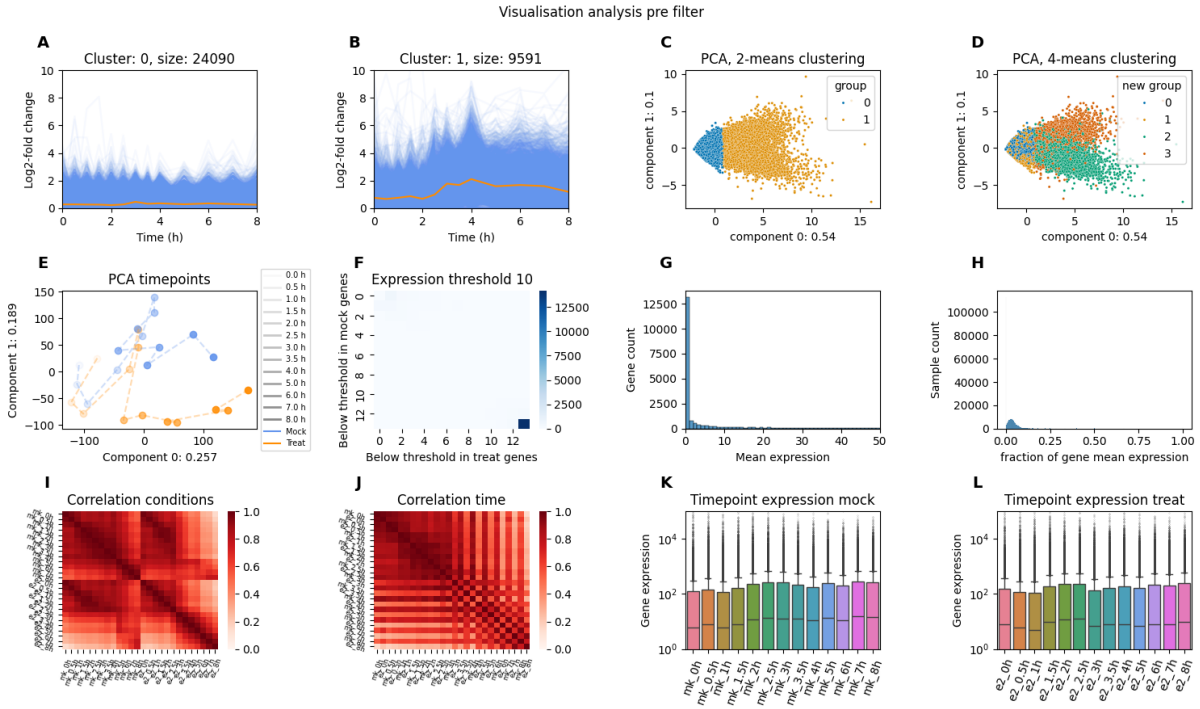

Figure 9: Data visualisation at the start of preprocessing. Information about the subplots can be found in Table 2.

4.1.3 Labelling

Table 9 shows the results from the different annotating rounds. In the final round 1004 genes were annotated, of which 530(52,8%) were DDEG. In this round also the lowest kappa score, from all annotation rounds, was found between 2 annotators, 0.48 which is weak agreement [20]. The highest kappa score in this round, 0.79, relates to a strong agreement. Since the lowest score relates to weak agreement and we used a majority vote for the final annotation the quality of the annotations is acceptable. Figure 10 shows the distributions of the annotators who voted DDEG on each gene. The majority of genes had a complete inter-annotator agreement for either DDEG or no DDEG. For around 100 genes the genes were difficult to classify as 3 or 4 labellers voted differently than the rest. Figure 11 subplot A shows a difference in the mean for DDEG and non-DDEG for DTW and Euclidean distance. The absolute average log2-FC in subplot B is higher for DDEG than no DDEG which is closer to zero. Figure 12 shows a clear distinction between labelled DDEG (y) and non-DDEG (n). It is more clearly visible when looking at subplot B where the absolute log 2 fold change violin plots are shown. From 3 hours to 8 hours, the mean log2-FC of labelled DDEG is higher than non-DDEG. This overlaps with the ETI induction which start at 2 hours.

Table 9: Results of labelling showing properties of each labelling round.

Round	Labellers	Genes (unique)	Agreement type	Kappa Cohen (max - min)	labelled DDEG	Labelled no DDEG
1	3	50	Complete (0.78)	0.8 - 0.65	23	16
2	3	100	Complete (0.71)	0.62 - 0.58	29	42
3	5	200	Majority	0.81 - 0.56	47	153
4	7	1004	Majority	0.79 - 0.48	530	474

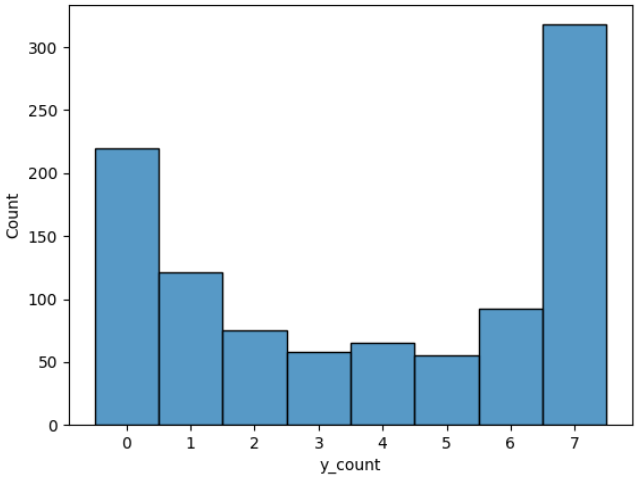

Figure 10: Gene count set out against DDEG votes by labelers.

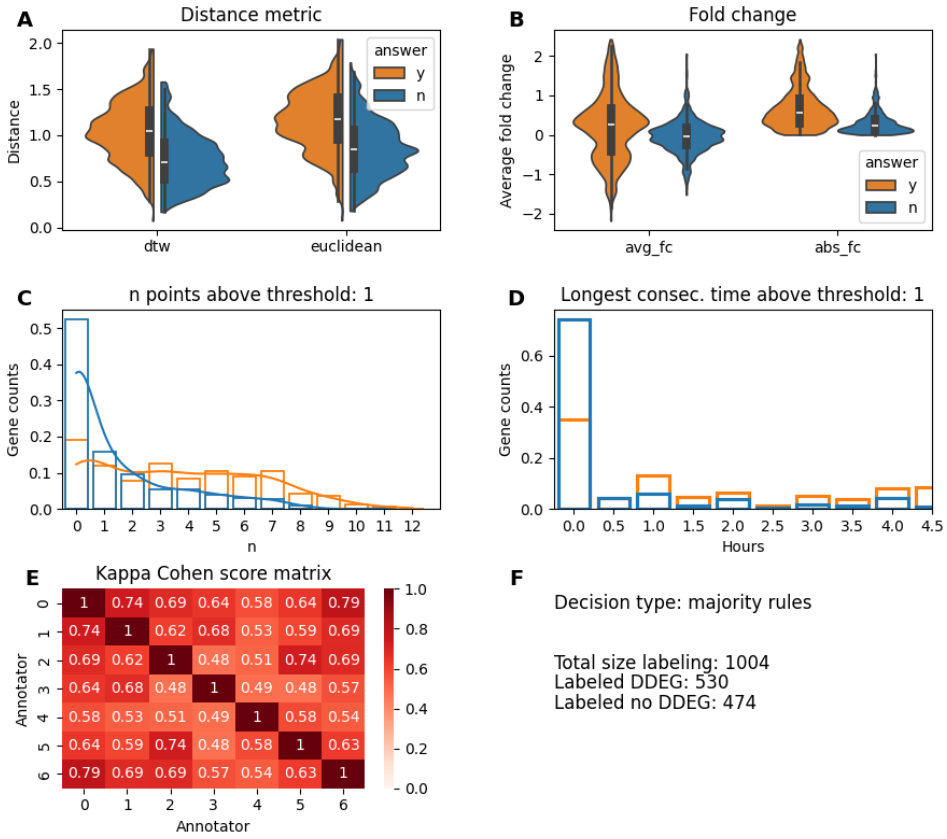

Figure 11: Results from the final from the round.

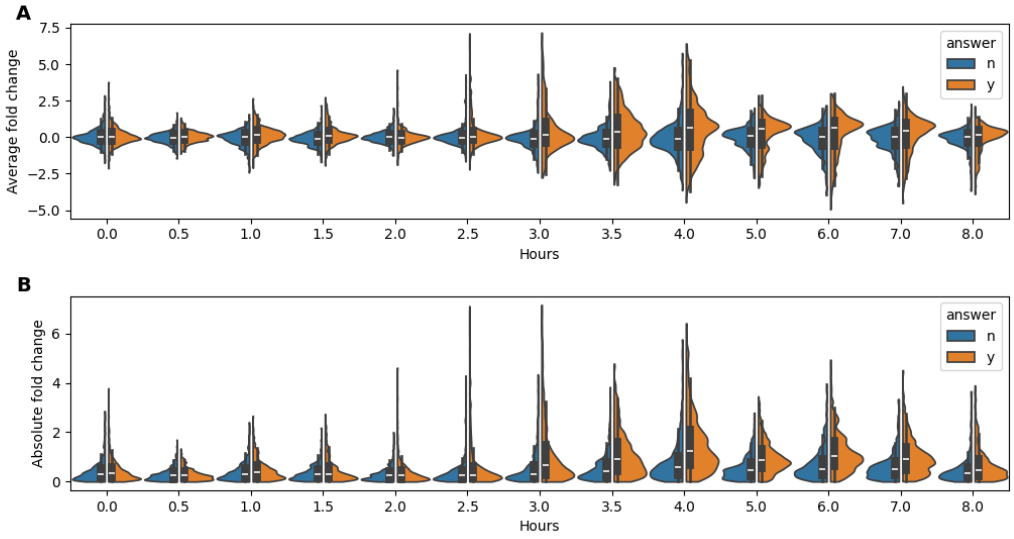

Figure 12: Violin plots of log 2 fold change for each timepoint. A) shows log2 fold-change, B) shows absolute log 2 fold change.

4.1.4 Training neural network

Figure 13 shows the final model architecture with hyperparameter settings. Figure 14 A shows the loss and accuracy of the final training. The model was then used on the test set, where a loss of 0.506 was obtained with an accuracy of 0.876. Figure 14 B shows the false predicted curves for the train and validation dataset.

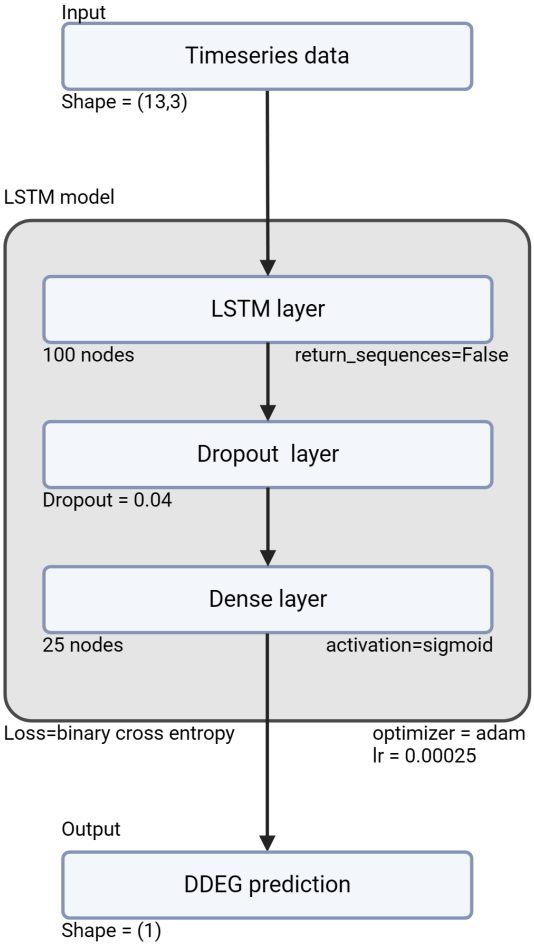

Figure 13: Architecture of final LSTM model.

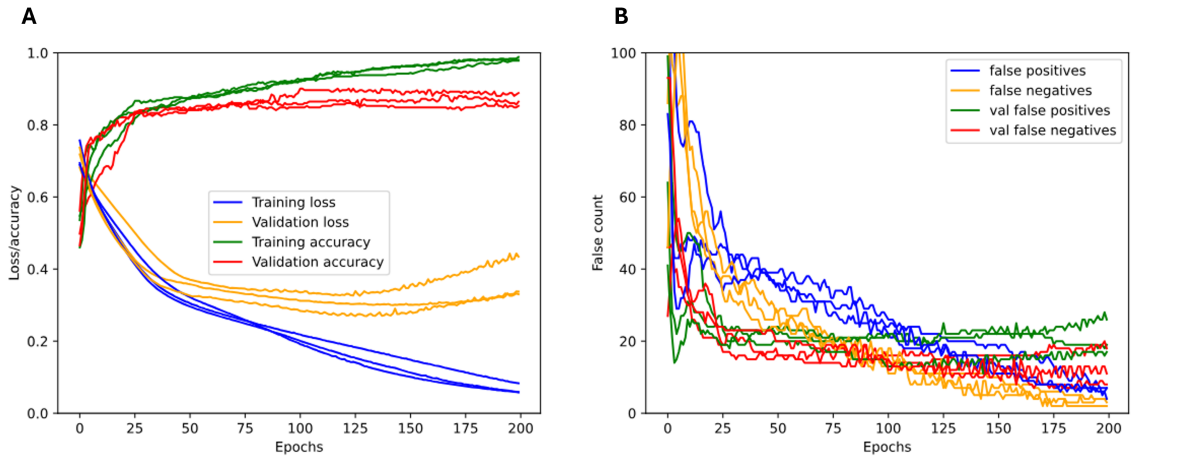

Figure 14: A) False counts plotted against the epochs for a 3-fold training. Blue and yellow lines represent training loss and validation loss respectively while green and red represent the training accuracy and validation accuracy. B) False counts plotted against the epochs for a 3-fold training. Blue and yellow lines represent false positive and false negative counts respectively on the train dataset while green and red represent the false counts on the test dataset.

4.1.5 DDEG prediction

The final pre-processed dataset is pulled through a pre-trained LSTM model, resulting in a DDEG and no-DDEG gene set. The DDEG analysis outputs 2 files: one file contains the gene names with their label (label_predictions.xlsx) and a txt file showing the number of genes in each group (label_occurrence.txt).

4.1.6 GEP clustering

Extracting gene patterns is done through hierarchical clustering. Using the complete linkage method and setting a distance cutoff clusters are obtained. Table 10 shows the files outputted during clustering.

Table 10: Clustering output files and their content

File name	Content
DTW_matrix.pickle	DTW distance matrix of genes
cluster_*.png	Timeseries foldchange plot of cluster of cluster genes and centroid indicates cluster number
cluster_*_go.csv	Gene cluster GO annotation file from gprofiler indicates cluster number
all_clusters.png	All foldchange centroids and genes clusters plotted in subplots
all_clusters_BP.png	All foldchange centroids and genes clusters plotted in subplots with a table of top 5 biological process annotations
all_clusters_CC.png	All foldchange centroids and genes clusters plotted in subplots with a table of top 5 cellular components annotations
all_clusters_MF.png	All foldchange centroids and genes clusters plotted in subplots with a table of top 5 molecular functions annotations
all_clusters_dtw.png	All foldchange centroids and genes clusters plotted in subplots with a corresponding subplot of foldchange, mock and treat DTW distribution of the time series gene data to their corresponding centroid
clustering_labels.csv	Gene names with their foldchange time series and cluster label
fc_centroids.csv	Foldchange time series data of cluster centroids
mock_centroids.csv	Mock time series data of cluster centroids
treat_centroids.csv	Treat time series data of cluster centroids
heatmap_centroids.csv	Foldchange heatmap of centroids against time
tsne_fc.png	T-sne plot of genes labelled according to clusters
umap_fc.png	Umap plot of genes labelled according to clusters
dendrogram.png	Dendrogram of genes with horizontal line set as cluster distance
dendrogram2.png	Dendrogram of genes with horizontal line set as cluster distance with different settings
heatmap_rela_BP.png	Heatmap cluster and biological processes GO annotations
heatmap_rela_BP_filter.png	Filtered heatmap cluster and biological processes GO annotations and parent annotations color labelled
heatmap_rela_CC.png	Heatmap cluster and cellular components GO annotations
heatmap_rela_CC_filter.png	Filtered heatmap cluster and cellular components GO annotations and parent annotations color labelled
heatmap_rela_MF.png	Heatmap cluster and molecular functions GO annotations
heatmap_rela_MF_filter.png	Filtered heatmap cluster and molecular functions GO annotations and parent annotations color labelled

4.1.7 Network inference

The BINGO algorithm takes the min-maxed normalised centroids of mock and treat to infer GEP regulatory network.

Table 11: BINGO output files and their content.

File name	Content
confidence_matrix_*.csv	File containing prediction by BINGO algorithm for cluster-to-cluster
inferred_network_*.html	Visualization of network
link_matrix.csv	Filtered confidence_matrix_*.csv to construct network
perturbation_posterior.png	Histogram predicting chance of cluster being in direct influence of perturbation
posterior_distribution.png	Histograms showing link summed link probabilities found by BINGO

4.1.8 Network analysis

In network analysis, we combine different types of data. The inferred network is combined with the ChIP-Hub reference gene-to-gene network to create a network highlighting known genes of interest for the experiment[9]. If a second network is available, a network comparison can be made. Table 12 shows the outputted files of this step.

Table 12: GEPRN analysis output files and their content.

File name	**Content**
reference_combined_networ.html	combined network of existing and inferred interactions.
networks_cluster_simmilarity.csv	Cluster-to-cluster similarty between two GEPRNs.
simmilarity_matrix.png	Cluster-to-cluster simmilarity visualized in a heatmap.

4.2 Applying pipeline

This chapter applies the pipeline to the S4 dataset, the images shown in this part are direct or edited files to visualise the results from pipeline.

4.2.1 Preprocessing

Figure 15 shows the log2-FC plots of 2-means clustering at each step of preprocessing. The smaller of the 2 clusters becomes more distinct after each step of the preprocessing, showing a prolonged increase compared to the other cluster starting at the 2 hour mark until after 8 hours.

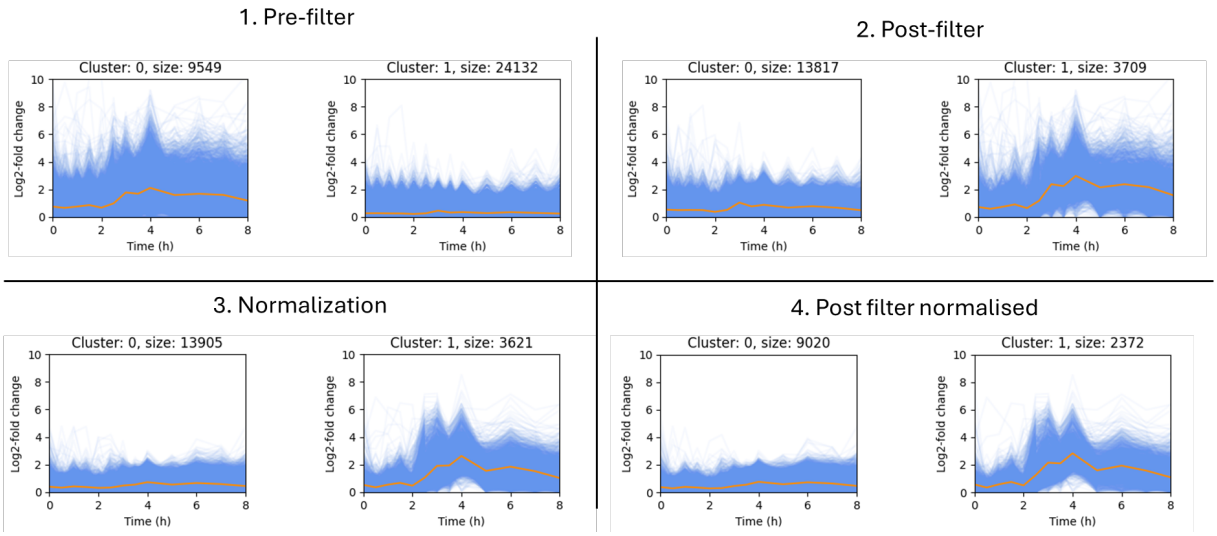

Figure 15: The 2-means clustering plots during every stop of preprocessing.

Figure 16 shows PCA plots with 2, types of k-means clustering. The plots of 2-means clustering shows in each part of the preprocessing a fairly straight vertical line. The 4-means clustering adds to this a rough horizontal line, leading to a rough separation into 4 quarters.

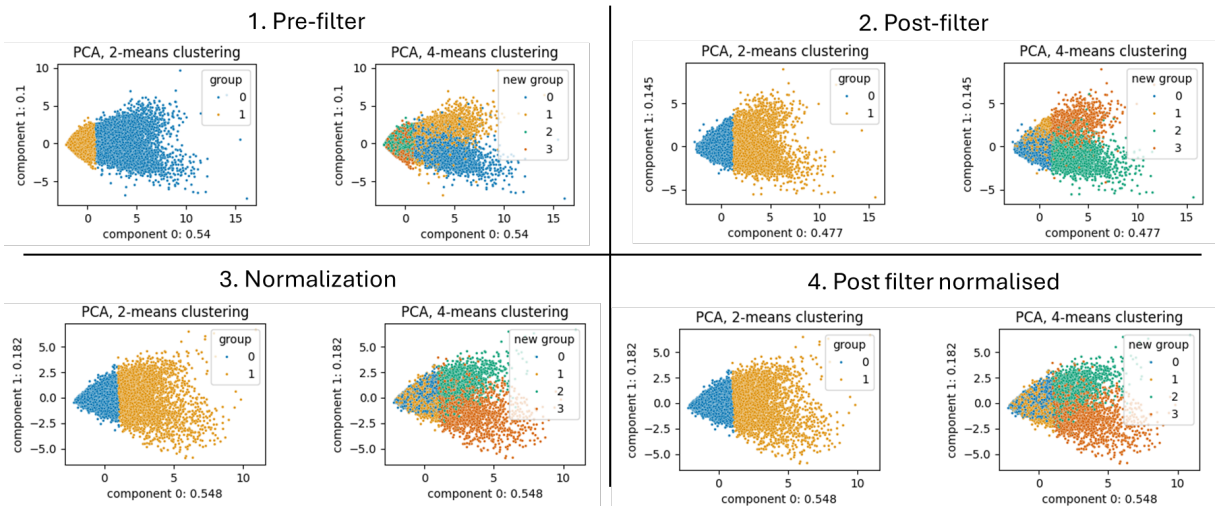

Figure 16: 2 and 4 means clustering during every step of preprocessed visualised in PCA plots.

The histogram plots in Figure 17 shows the largest change after filter steps as these figures are mainly focussed on visualising the influence of filtering.

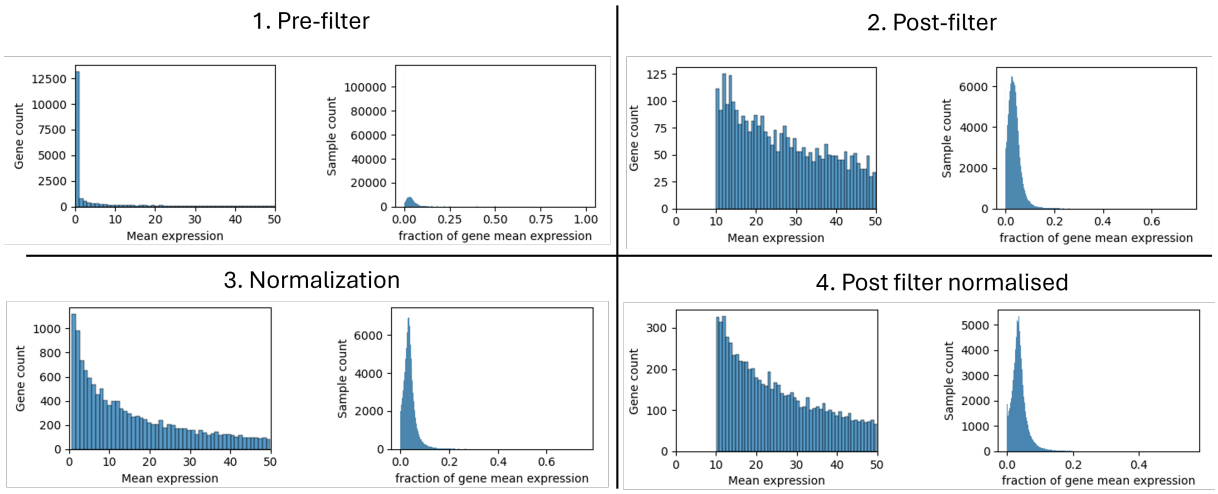

Figure 17: Overview of mean expression and fraction mean contribution of samples. After every step a large change is visible and shows the influence of the filter and normalisation steps.

Figure 18 shows no change between steps. In the correlation plot ordered by time, we see a high correlation between mock and treat in the earlier hour and a decreasing correlation after 2.5 hours. In the conditions ordered correlation map mock has a high correlation with itself, except with the hours before 4 hours and after 4 hours. Treat has a high correlation with itself between 0 and 2 hours, after 2 hours it only correlates with local neighbours.

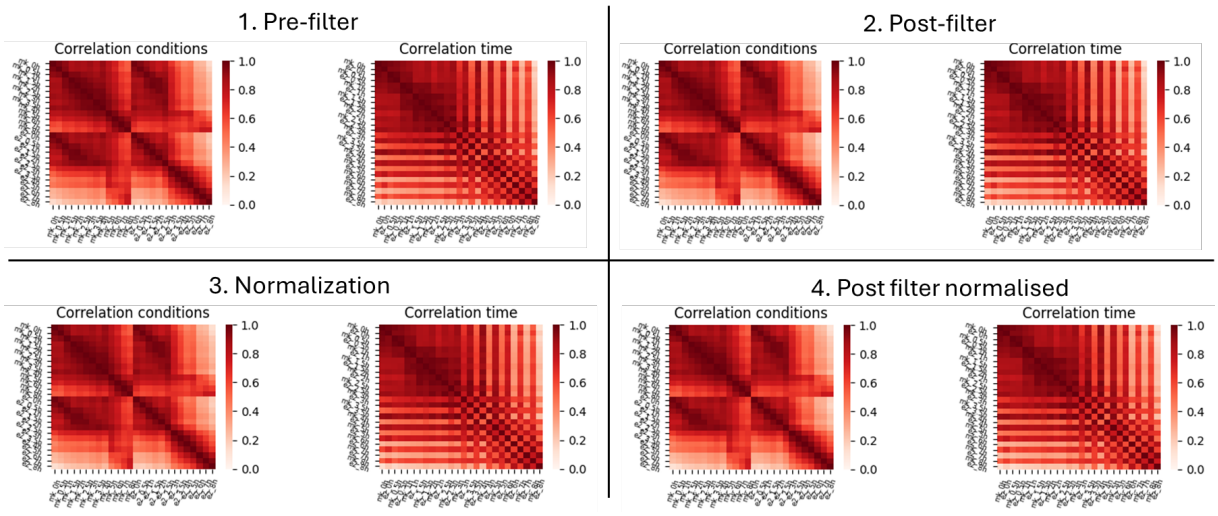

Figure 18: Expression boxplots of mock and treat. Normalisation equalizes the mean of the boxplots.

Figure 19 shows The lower range of the boxplot being removed after the first filtering, means getting aligned after normalisation and again the lower range removed after the second filtering.

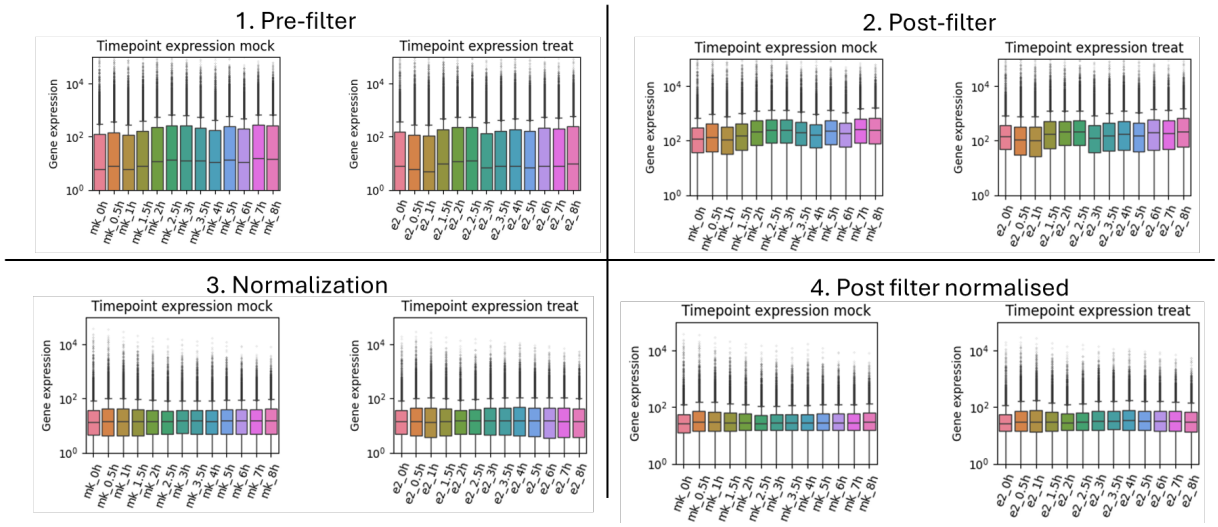

Figure 19: Boxplots of gene expression for all samples.

Figure 20 shows PCA plots of samples, a noticeable change occurs when normalising. The heatmap shows the distribution of samples under an expression range, after each step a change occurs.

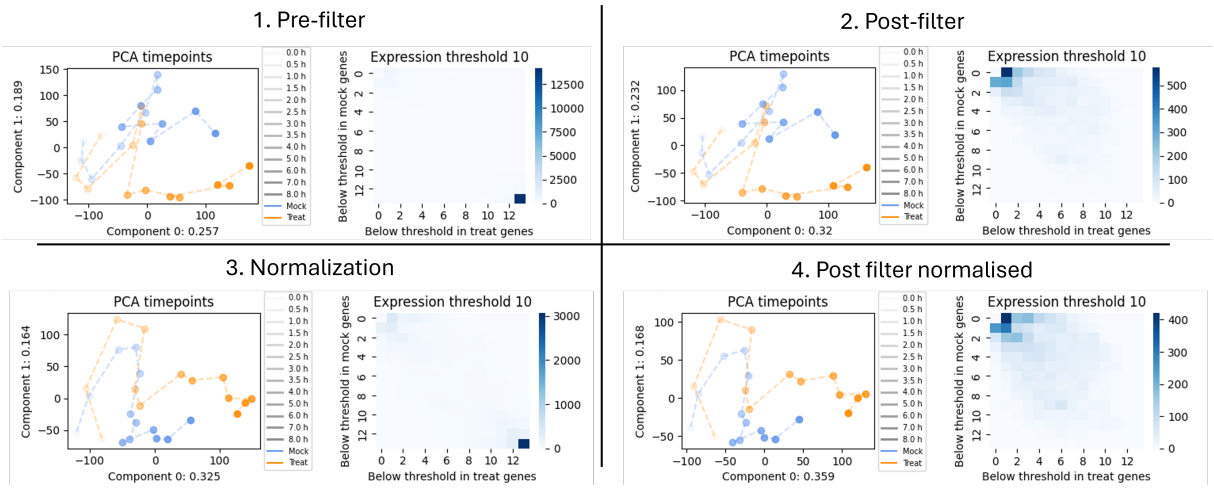

Figure 20: PCA plot of samples and expression filter heatmap

4.2.2 DDEG

The preprocessed genes are put into the trained LSTM model to predict DDEGs. Of the 11392 genes, 5816 were predicted to be DDEGs. By using the old pipeline on the same data and pulling the data through an SDEG tool (SplineTimeR) we can see an overlap of DDEGs. Figure 21 shows this overlap, 1209 genes are predicted to be DDEGs by all tools, 948 are found to be DDEGs by the current and original pipeline. 1862 genes are predicted to be DDEGs by the pipeline and SplineTimeR. The current pipeline has 1797 genes, predicted DDEGs, which are not found to be DDEGs by the other two tools.

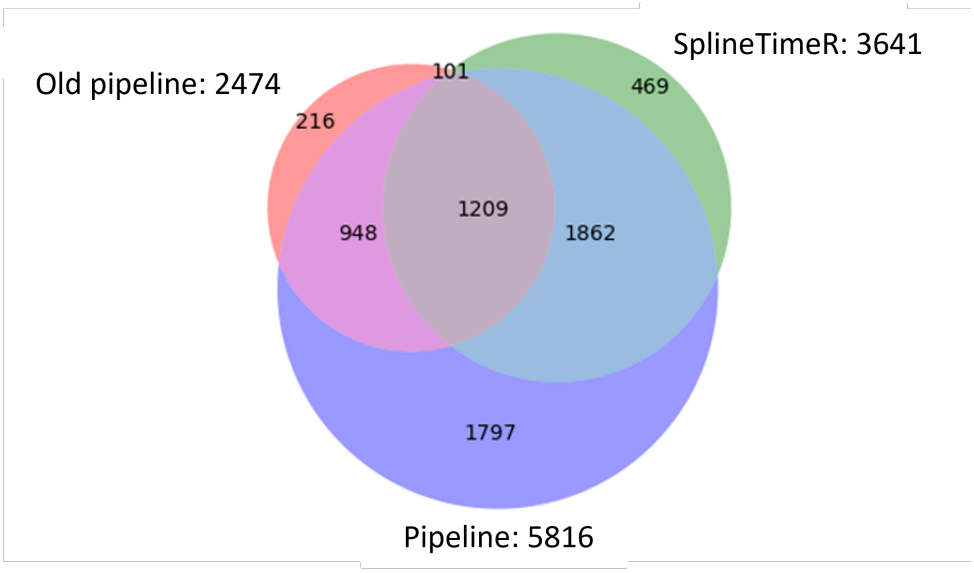

Figure 21: Venn diagram of gene overlap between 3 tools: the (new) pipeline, old pipeline, and the tool splinetimeR. The total DDEG's found are next to the tool names.

4.2.3 Clustering

After hierarchical clustering, the genes are coloured by cluster and are visualised into a T-SNE and a UMAP plot (Figure 22). In both groups, the colours group together well and 2 main groups can be seen. The foldchange GEP resulting from clustering are shown in figure 23. Visible is that clusters 0 to 12 seem to show upregulation and clusters 13 to 23 seem to have an overall downregulation.

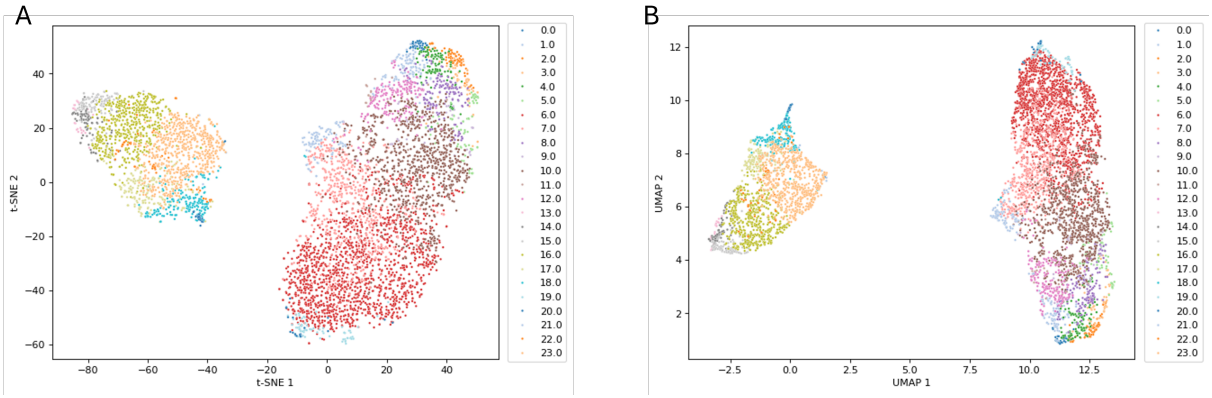

Figure 22: T-SNE (A) and UMAP (B) of DDEGs colour labelled by clustering of S4.

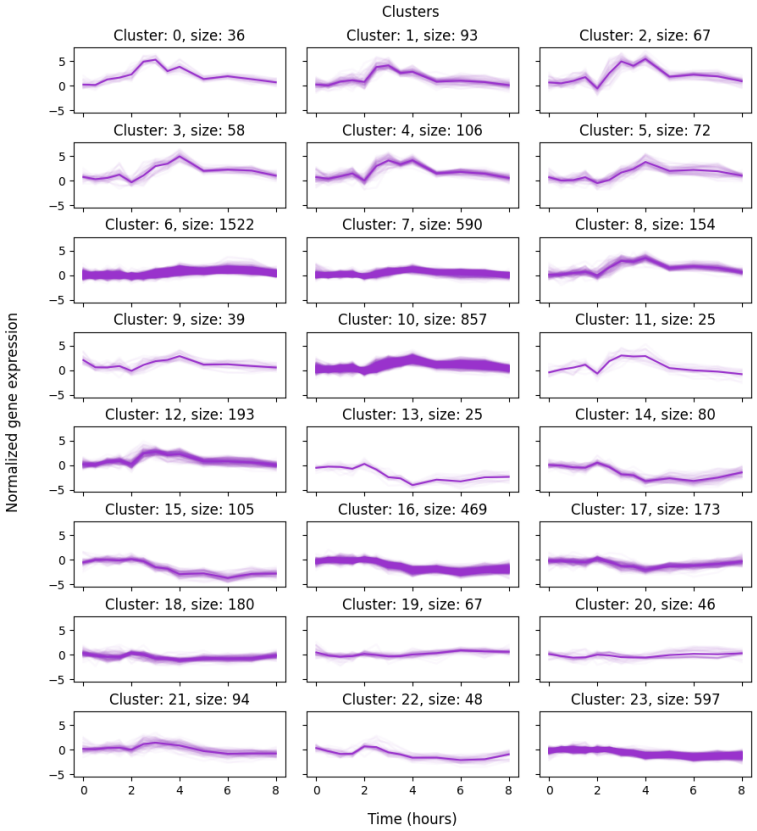

Figure 23: Subplots containing log2-FC time series genes and their averaged centroids for S4.

GO annotation was performed on each cluster and visualised in three heatmaps relating to their families: biological processes (Figure 24), molecular function (Figure 25) and cellular component (Figure 26). Visible from the figures are four types of cluster annotation. the first type of clusters does not contain enough genes to be annotated. The second type of clusters is spread out over multiple second-layer GO functions, this can be contained within the first layer of GO functions or spread out into different first-layer GO functions. The last type of cluster has a singular GO function in the second layer.

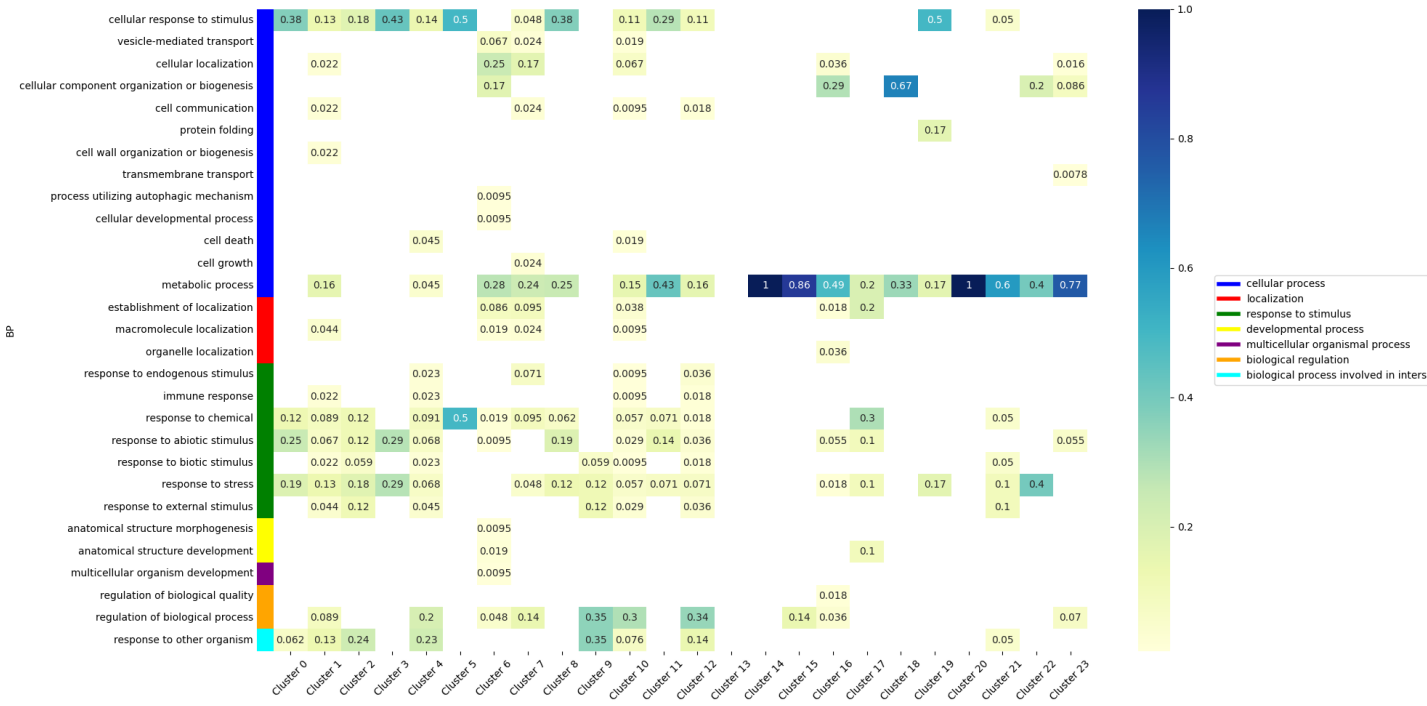

Figure 24: Heatmap setting out the second layer of GO terms under biological processing annotation against clusters, colour labelled by first layer GO term. The numbers and colours in the heatmap represent the fraction of all GO terms, found for that cluster, falling under the specified GO term.

The GO heatmap of biological processes, Fig 24, shows a lot of clusters enriched in metabolic processes. Using Fig 23 we can see that after cluster 12, most clusters are downregulating. During the immune reaction metabolic processes seem to slow down to focus on the immune response. The clusters that are upregulated seem to be enriched in GO terms related to stimulus and stress. We also see GO terms more directly related to immunity such as immune response, cell death and response to other organism.

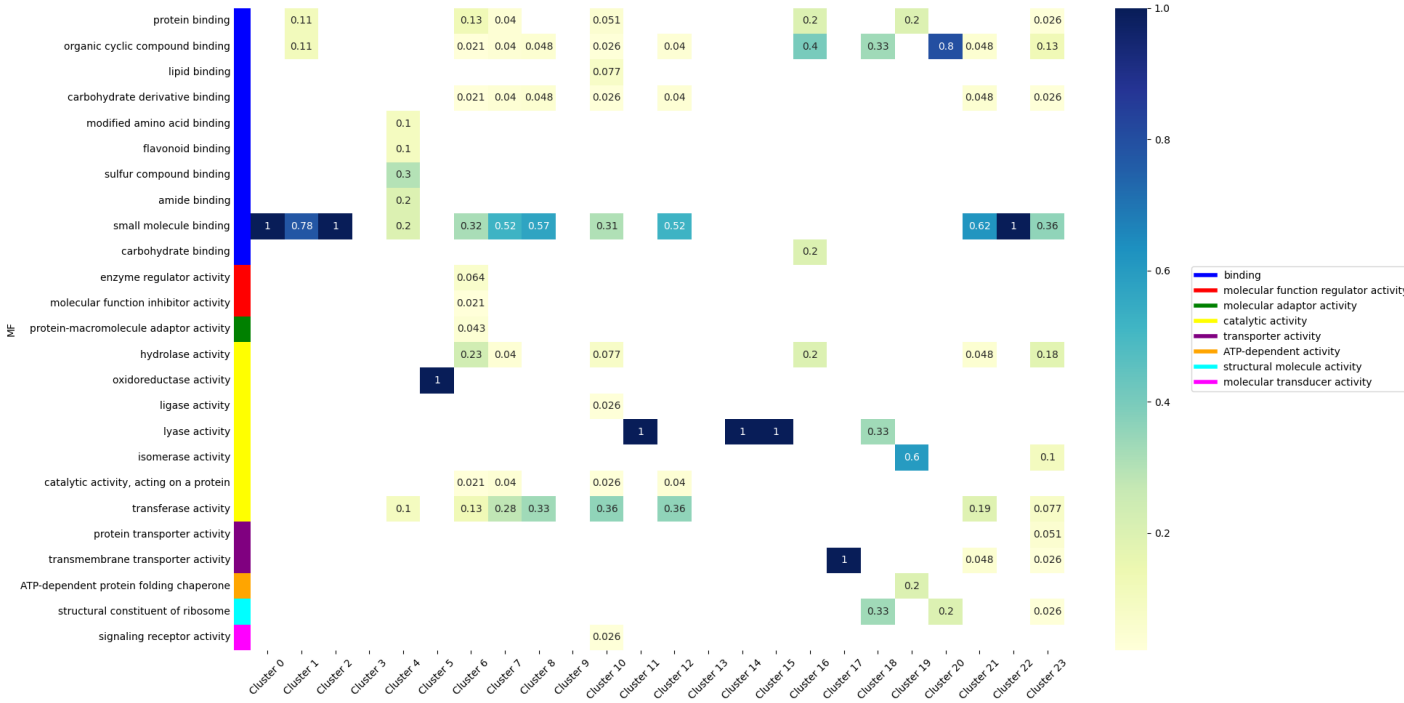

Figure 25: Heatmap setting out the second layer of GO terms under molecular function annotation against clusters, colour labelled by first layer GO term. The numbers and colours in the heatmap represent the fraction of all GO terms, found for that cluster, falling under the specified GO term.

The GO heatmap of molecular function, Fig 25, shows a lot of clusters fully enriched in a singular GO term. Most clusters fall under the group of binding and catalytic activity. Small molecule binding is the GO term most clusters are enriched in. This also account for clusters 0, 1 and 2 which have the response to other organism GO term in biological processes.

Figure 26: Heatmap setting out the second layer of GO terms under cellular component annotation against clusters, colour labelled by first layer GO term. The numbers and colours in the heatmap represent the fraction of all GO terms, found for that cluster, falling under the specified GO term.

The GO heatmap of cellular component, Fig26, shows some clusters that did not shows any known enrichment for the GO terms. Cluster 6 is enriched in the most different GO terms while cluster 2 is enriched in only cell periphery. With cluster 2 being enriched in response to other organism, small molecule binding and cell periphery it is highly likely cluster 2 plays an early role in immunity.

4.2.4 Network inference

The BINGO algorithm predicts two things: the likelihood of a link existing between two clusters and the likelihood of a gene being directly influenced by the perturbation. In Figure 27 we can see the distribution of all links visualised on the left, as we are only interested in the highly likely links, a zoom in can be seen on the right. Around 18 links have a likelihood of 0.8 or higher. Figure 28 shows cluster 0 with full certainty being directly influenced by the perturbation.

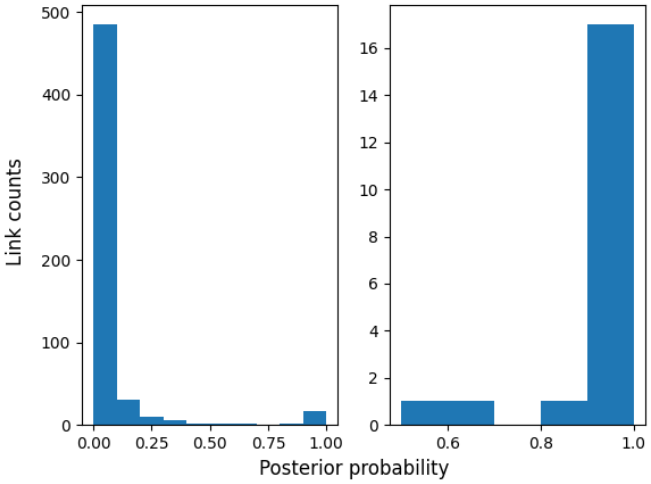

Figure 27: Link counts set out against posterior probability from the BINGO algorithm for AvrRps4. The right plot is a zoomed-in version of the left plot.

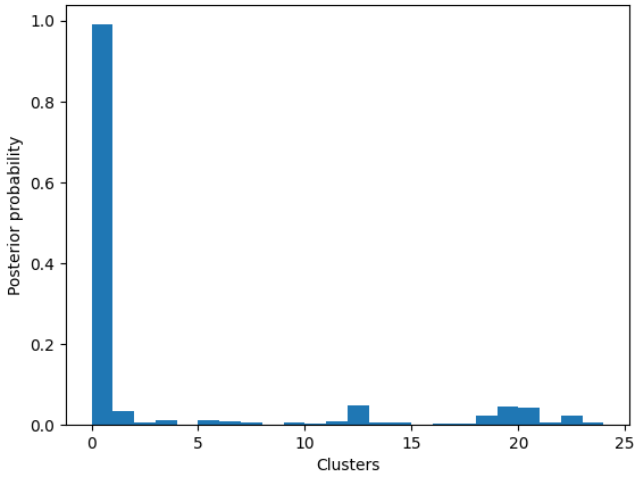

Figure 28: Pertubation probability set out against cluster numbers for AvrRps4. Indicates the likelihood of a cluster being directly influenced by the perturbation.

Combining both predictions from the BINGO algorithm allows us to visualise the network as shown in Figure 29.

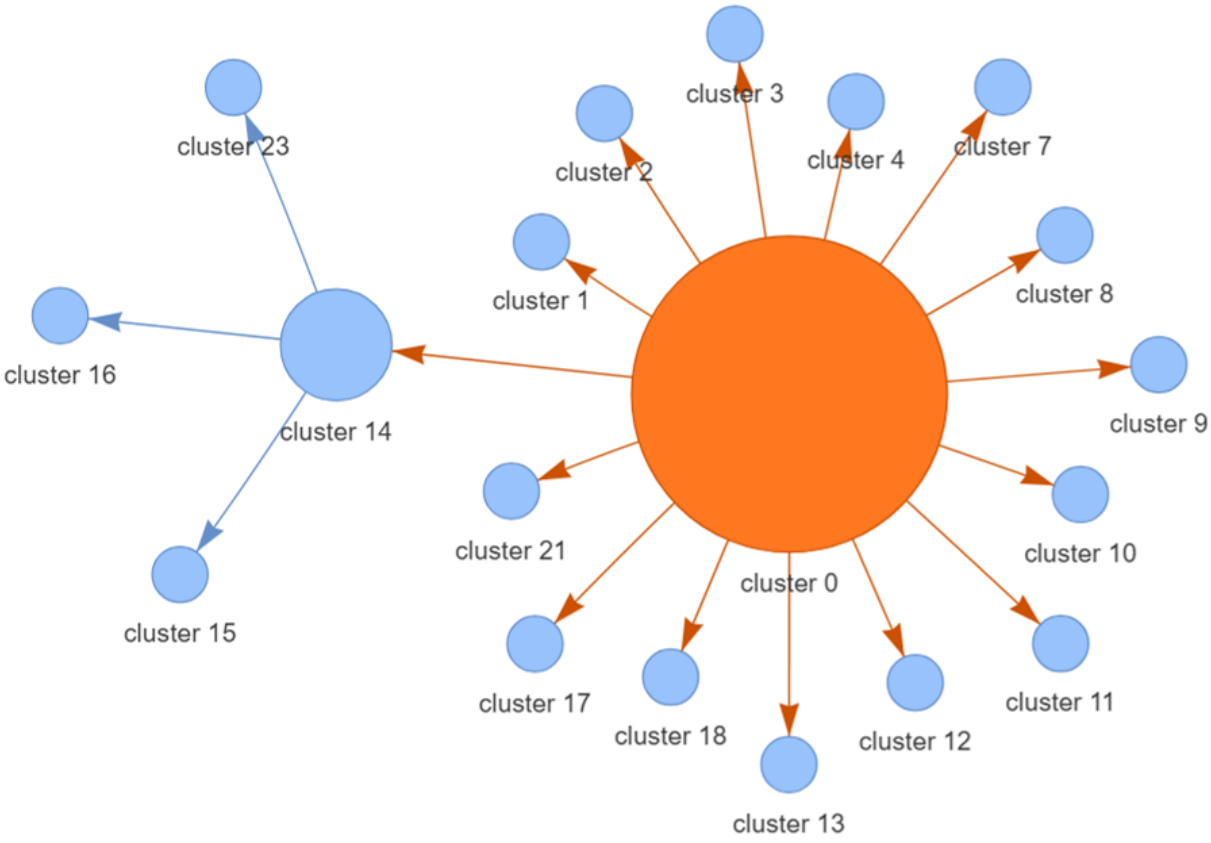

Figure 29: Inferred network for S4. Cluster numbers relate to the clusters in Figure 23. An orange node is under direct influence of the treatment, a blue node is not under direct influence.

To make a network comparison a second dataset T2 was ran. The link likelihood distribution and pertubation likelihood can be found in Figure 30 and 31. The inferred model can be found in 32.

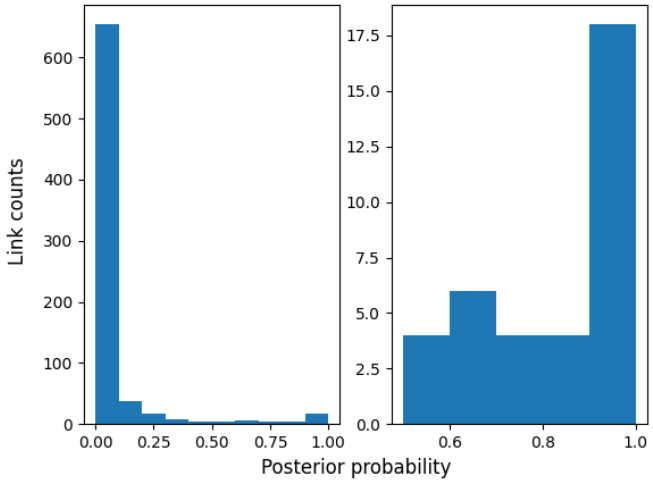

Figure 30: Link counts set out against posterior probability from the BINGO algorithm for T2. The right plot is a zoomed-in version of the left plot.

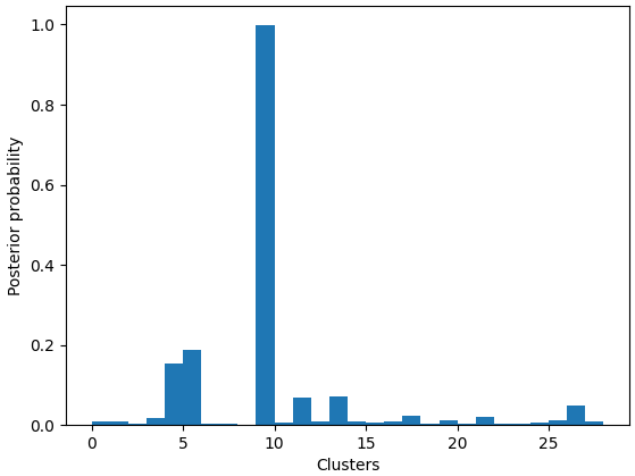

Figure 31: Pertubation probability set out against cluster numbers for AvrRpt2. Indicates the likelihood of a cluster being directly influenced by the perturbation.

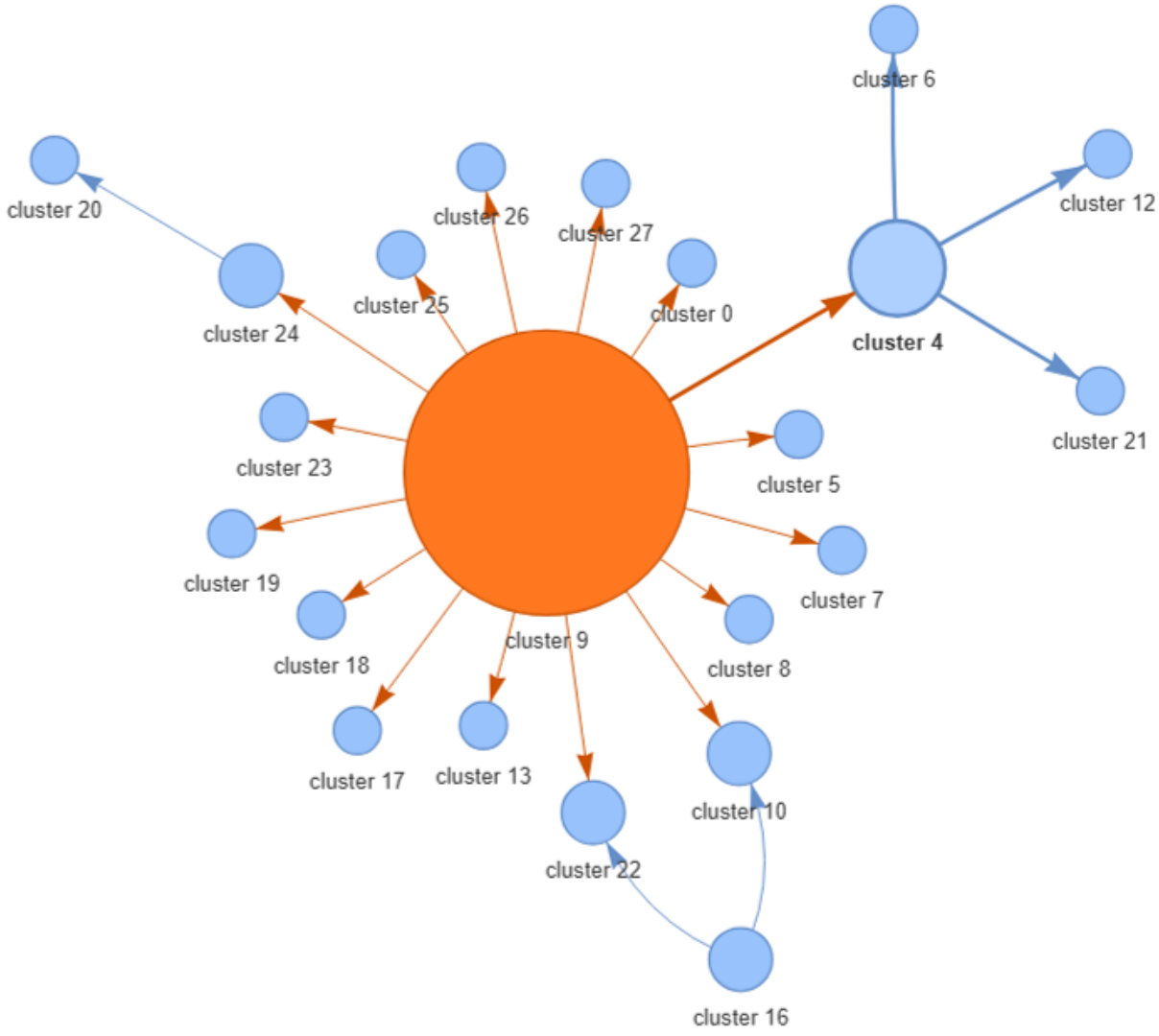

Figure 32: Inferred network for AvrRpt2. Cluster numbers relate to the clusters in Figure 45.

4.2.5 Network analysis

By overlapping the predicted network with a reference gene-to-gene network we can obtain 2 things: a validation of our inferred network as in lesser or larger extent we should find overlap between the networks and a highlight of known genes that play an important role in the perturbation. If overlapping the networks results in a to small network or no network at all, it could indicate that the parameter settings of the pipeline need to be changed. When there is alot of overlap forming a network we can focus on hub nodes, which could be genes previously thought to have no influence on a defence response.

Figure 33 shows the overlap of the inferred S4 network, Figure 29, with the reference network. The network of S4 shows 3 genes that form hubs(Table 13). The overlap network for T2 can be seen in 34 and the hub nodes in 14. The overlap of both inferred networks with the reference network results in hub genes that are mainly transcription factors.

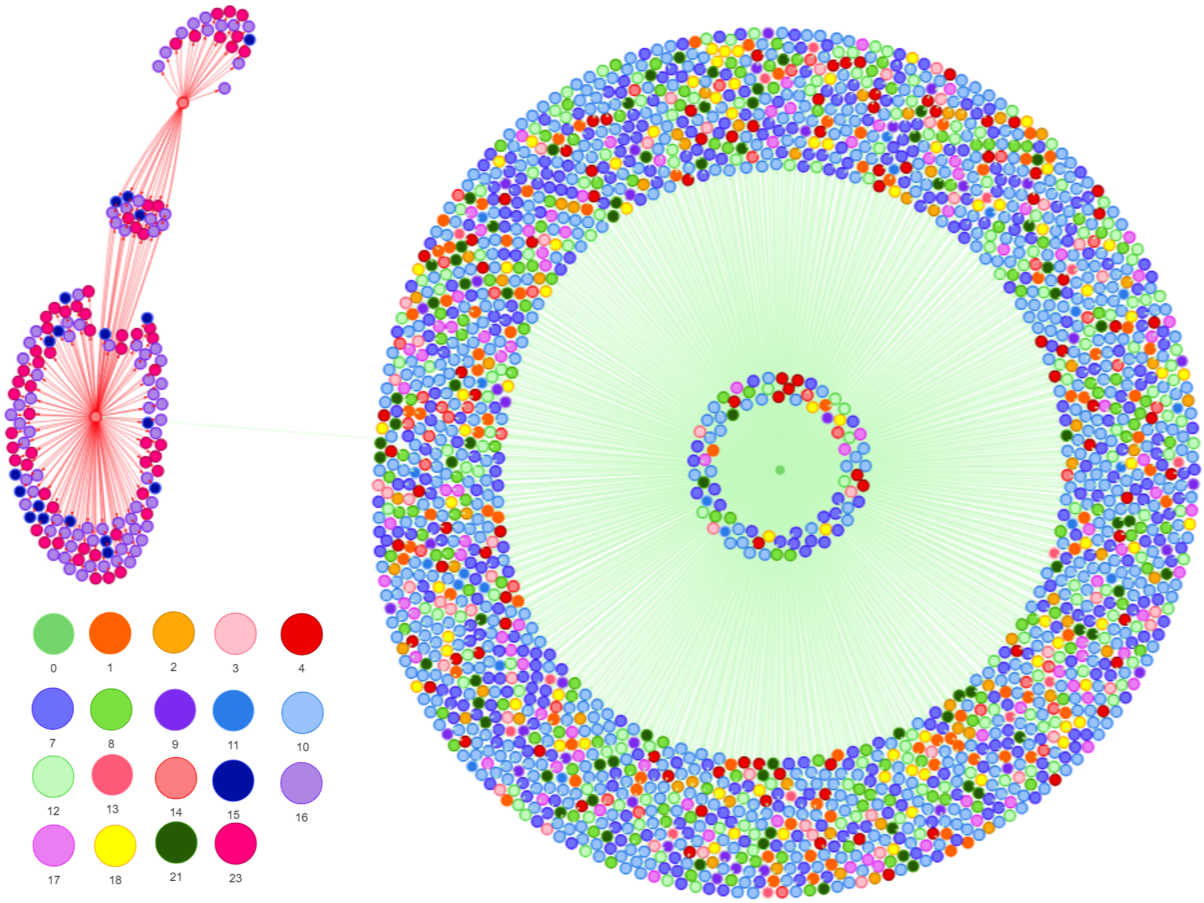

Figure 33: Inferred network for AvrRps4 overlapped with the reference network. Numbers relate to clusters in figure 33.

Table 13: Hub genes, their names, cluster and description of S4.

Cluster	**Hub gene(code)**	**Hub gene(name)**	**Description[7]**
0	AT2G38470	WRKY33	Transcription factor regulating defense pathways
14	AT2G33860	ARF3	Transcription factor that regulates plant growth
14	AT4G32980	ATH1	TRanscript factor that controls floral competency

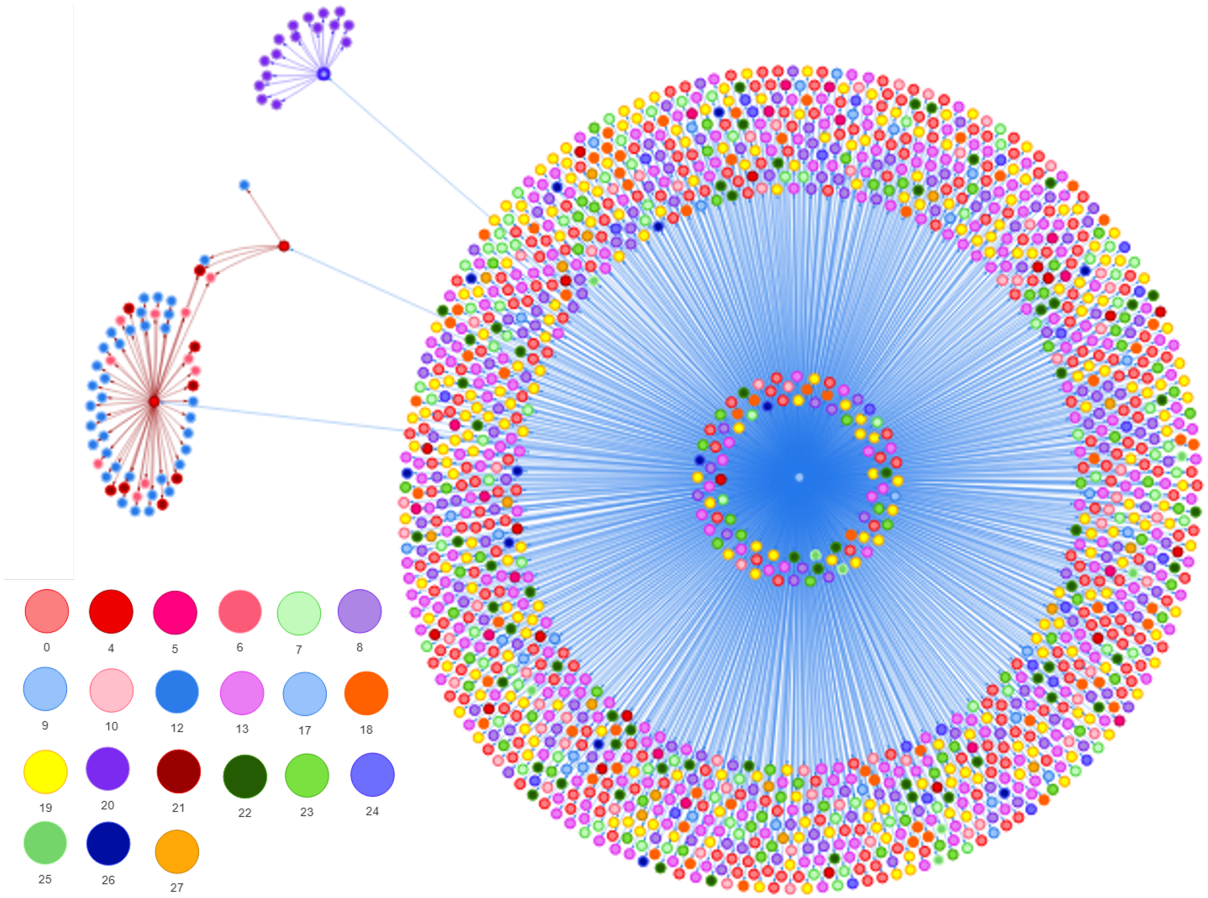

Figure 34: Inferred network for T2 overlapped with the reference network.

Table 14: Hub genes, their names, cluster and description of T2.

Cluster	Hub gene (code)	Hub gene (name)	Description
9	AT5G04340	ZAT6	Transcription factor regulating root development and homeostasis
24	AT4G31800	WRKY18	Transcription factor regulating defense repsonse
4	AT2G02080	IDD4	Transcription factor regulating photosynthetic gene expression.
4	AT5G24470	PRR5	pseudo-response regulator regulating circadian rhythm events.

When 2 experiments have been performed with the same mock a network comparisson can be performed to find overlap and differences between pertubations. This can be done by taking a look at the cluster simmilarity between the two networks. Figure 35 shows such an overview, where we can identify some simmilar clusters combinations. Cluster 6 from S4 overlaps the most with cluster 11 from T2 with a value of around 0.4.
Figure 36 shows both networks with on the left S4 and on the right T2. Both have a hub cluster under the direct influence of the perturbation. In Figure 37 shows for both network the up and down regulation of the clusters, the majority in both networks are upregulated. Figure 38 shows the similarity from S4 to T2 based on a similarity threshold. While some cluster of T2 shows similarity with one or two cluster(s) of T2 most do not pass the similarity threshold.

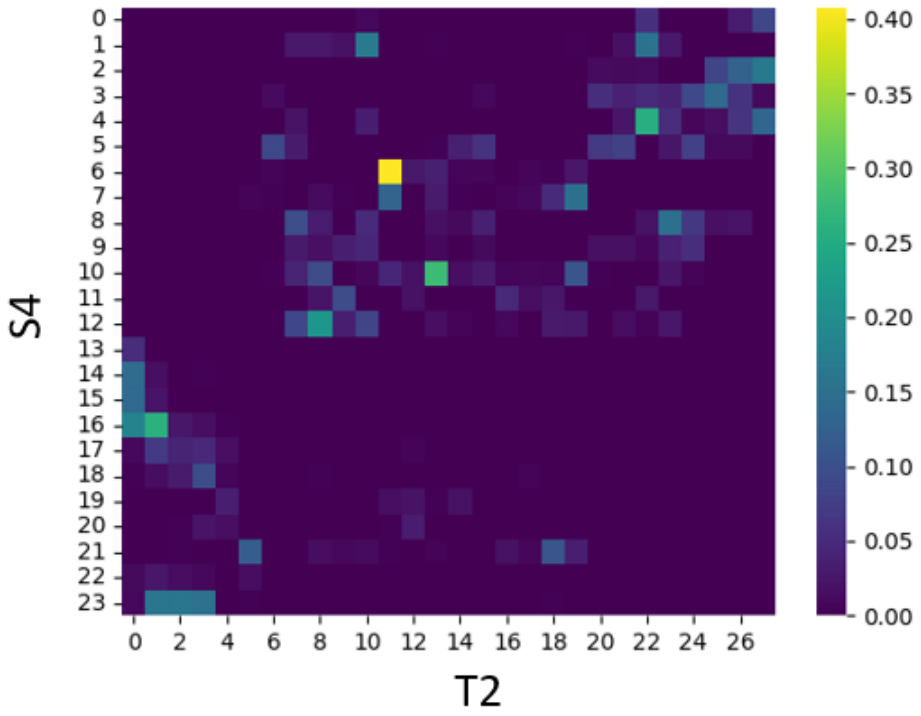

Figure 35: Inter network cluster similarity between S4 and T2.

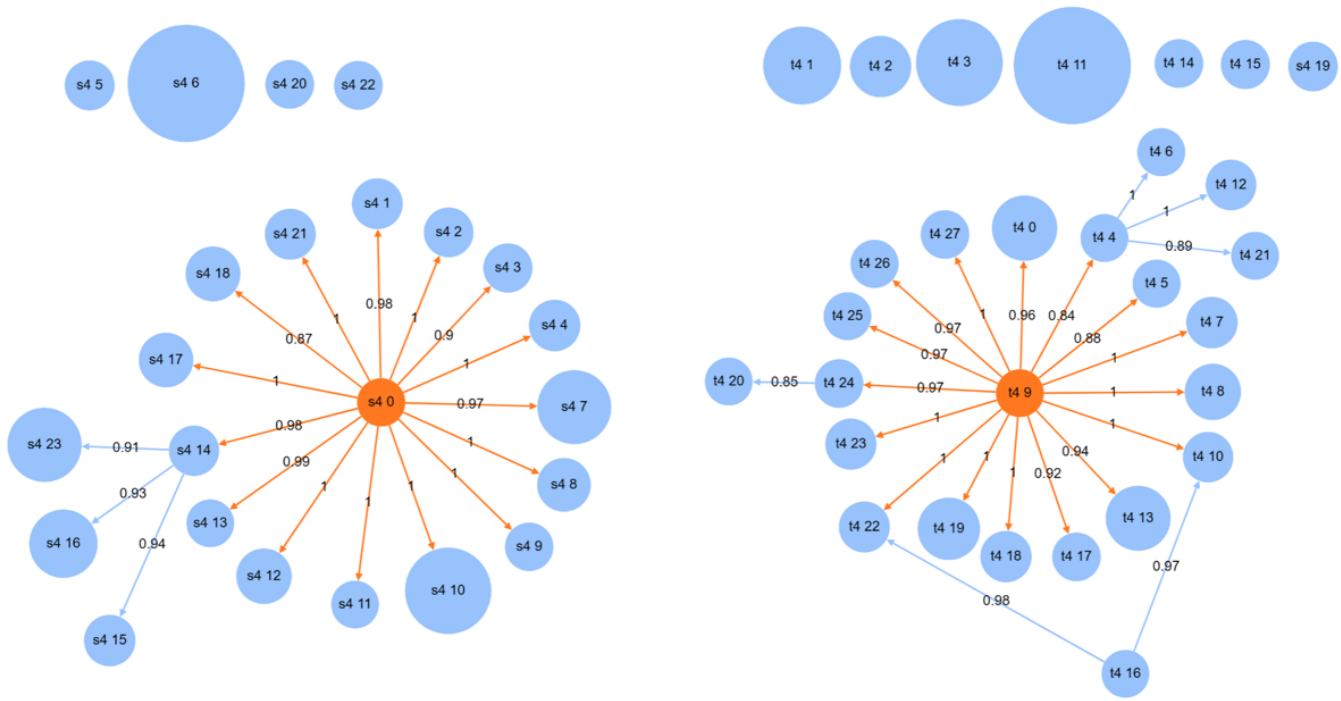

Figure 36: Inferred networks for S4 and T2. Cluster numbers relate to the clusters in Figure 23 and 45 for S4 and T2 respectively. An orange node is under the direct influence of the treatment, a blue node is not under the direct influence.

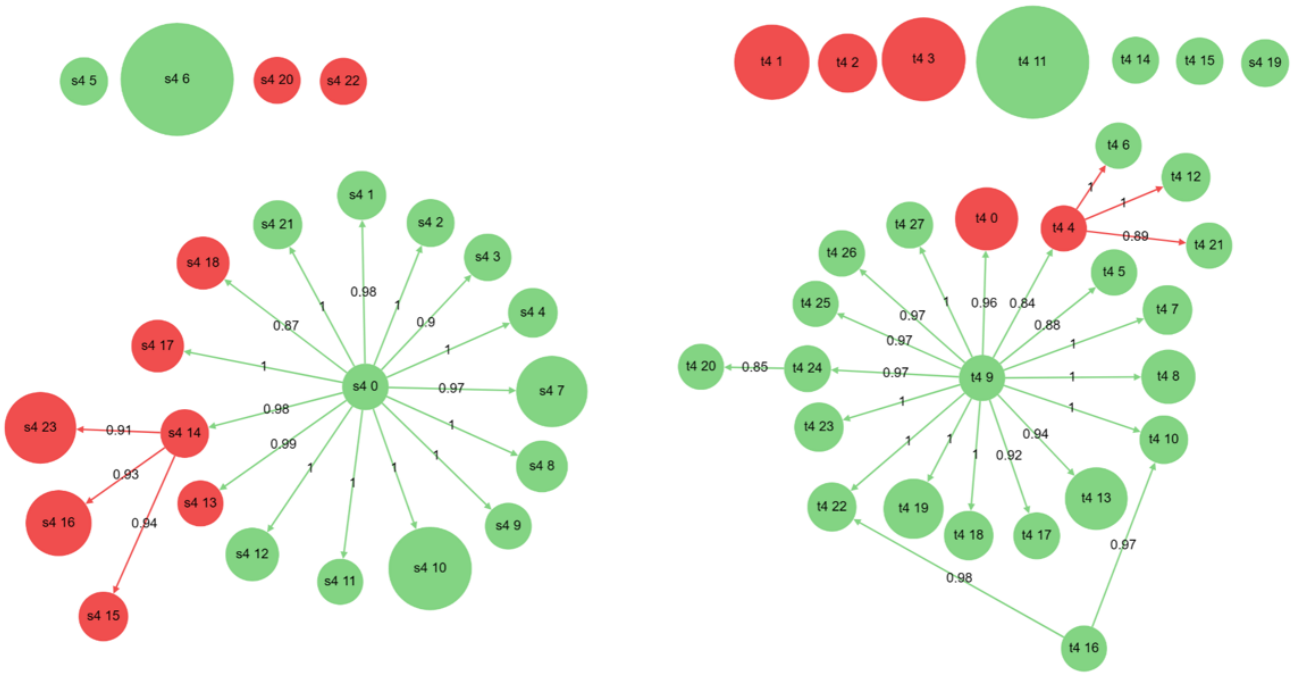

Figure 37: Inferred networks for S4 and T2. Cluster numbers relate to the clusters in Figure 23 and 45 for S4 and T2 respectively. Green nodes have an average upregulation and red nodes an average downregulation of their gene expression pattern.

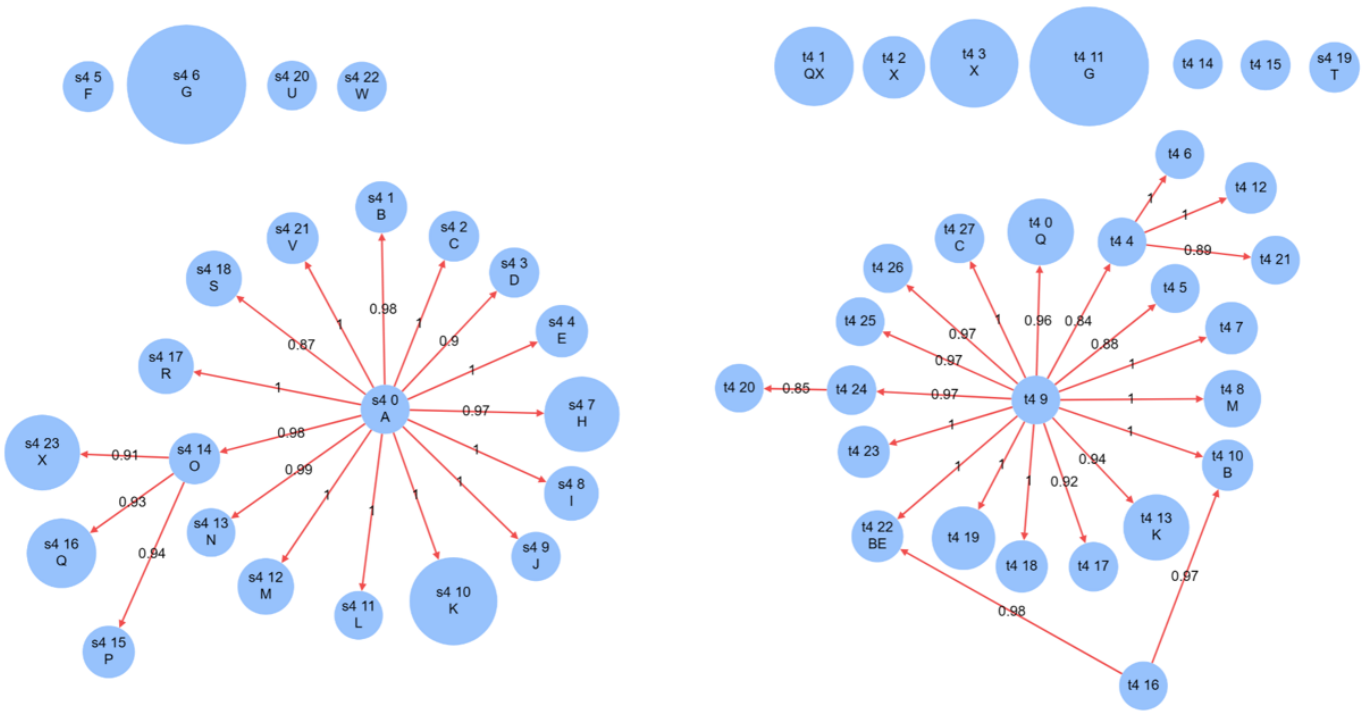

Figure 38: Inferred networks for S4 and T2. Cluster numbers relate to the clusters in Figure 23 and 45 for S4 and T2 respectively. Similarly from S4 to T2 is shown through letters. corresponding clusters with the same letters have a similarity of 0.3 or higher.

5 Discussion

This thesis aimed to adress the following question: Can an end-to-end pipeline for non replicated time series bulk mRNA data infer gene expression pattern regulatory network reliably representing complex time dependent cellular processes such as immunity? This resulted in an easy-to-use end-to-end tool for time series DDEG analysis, allowing for new insights into cellular processes through gene expression pattern analysis leading to the discovery of new gene functions. The first focus point of this thesis was to improve upon a previously created pipeline for time series FastQ bulk mRNA data into a GEPRN. By improving every step of the pipeline this has been achieved(changing clustering algorithm, training a new machine learning model, and annotating DDEG dataset). The second focus point was to improve usability and reproducibility. By running the code from a main file, a clear overview of parameters and output is maintained. Only the BINGO algorithm needs to be run separately due to it being written in MATLAB. This allows for easy reproducibility by others and ease of use for researchers starting with this pipeline.

Preprocessing is the first step of the pipeline, which processes FastQ files into gene expression, allows the user to fine-tune the amount of filtering on their dataset. Through visualisations(Figure 9), the user can interpret and change the parameters. A labelled dataset was created to train a machine-learning model for DDEG prediction. Here a non-statistical definition was made using data analysis (Fig 10. 11. 12.). The size of the dataset, 1004 genes, is sufficient for the current model, as cross-validation is possible compared to the original pipeline[12]. An LSTM model was trained for DDEG prediction, it is trained through 3-fold cross-validation using an Optuna study for hyperparameter optimisation. The study shows almost equal performance for a lot of hyperparameter combinations, showing that the current task is fairly easy to solve, it also shows that the loss is fairly high. This could be a result of labelling indicating that the training data is inconsistently labelled. The hierarchical clustering algorithm makes use of a DTW distance matrix, which allows for dynamic clustering23. The one-time calculation for this matrix is a big improvement compared to using another clustering algorithm like k-nn, which would greatly increase running time. While the parameter which extracts the number of clusters from the algorithm is not known beforehand, it is most likely to fall between a range that, with increased usage of the pipeline, can be determined. During the GEP inference step the gene patterns are put into the BINGO algorithm, which then determines the likelihood of different GEPRN configurations. The run time of the model greatly increases as more GEPs are provided or the iterations are increased. Since the GEPs from clustering, through changing the method, dropped from 43 in the old pipeline to 26 it was possible to increase iterations to get more accurate inference predictions. The threshold for perturbation posterior probability was increased from 0.3 to 0.8 compared to the old pipeline.

Since the experiment which provided the data in this thesis was not repeated, no replicates were available. Due to this the biological variation could not be determined and/or reduced. Since this pipeline consists of multiple parts which each their own error and points of optimisation this greatly affects the result of each step of the pipeline and could have an avalanche effect: biological variation with a possible error in each part of the pipeline can lead to an inaccurate inferred network at the end. The current pipeline is limited to the 13 time points with specific sample times for both mock and treat. This is due to the training data consisting of only those number of samples and sample times. It is advised to improve the pipeline in 3 maim ways. the

first is increasing the training dataset size and variety: sample times between mock and treat not alligning, and a variable number of samples. the second improvement is by improving every part of the pipeline individually. The last improvement in performing a study with replicates and their influence. It is expected that increasing the number of replicates improves the inferred model, but the exact difference between no and one replicate is at this point unknown.
Preprocessing can be improved upon by finetuning the filter method, which depends however on the desired output by the user. Currently, the DTW function used in multiple parts of the pipeline has a constraint of 1, meaning it can take direct neighbours into account when calculating its distance. Since the time series does not have constant time between its points (half-hour and 1-hour) this is most likely to have a negative influence on the pipeline predictions/output. By making a weighted DTW function, taking the time difference between each compared point into account, performance will be increased and is strongly recommended. It is strongly recommended to further increase the size of the dataset, incorporate data from different experiments and further finetune a more accurate non-statistical DDEG analysis. While labour-intensive it is crucial for the performance of the pipeline. The current model is, through the training data, limited to 13 time points with specific time intervals and requires both conditions to be of the same size and time interval.It is expected that increasing the dataset and fine-tuning a better representing DDEG definition could lower the achievable loss. When making the input data more flexible, different time differences between time points and different sizes, it is expected that the training becomes more complex. However, it is expected that this will not form an issue if the above-named quality of the training data increases. The number of clusters is currently decided by a set variable, the "best" range of this variable is likely to differ per experiment or datatype and should be finetuned each time, however, it is most likely there is a general "best" range that can be found through empirical evidence. By using the pipeline the user will find out what gives the best result, since the distance matrix only needs to be calculated once this should not form an issue as clustering only takes minutes. For The Bingo algorithm it is recommended to do an indepth study to what number of iterations is ideal, as with the increase in iterations, the results should be more stable when repeated. A balance between a stable and accurate result against running should be established. The similarity measurement of clusters between two networks should be finetuned now the similarity in GO, DTW and genes contributes equally to the final cluster-to-cluster similarity, it is expected that a more refined approach can give a more accurate similarity.

The pipeline was applied to two datasets resulting from infecting the A. thaliana plant with tho ETI inducers: AVRPS4(S4) and AVRPT2(T2). Thirteen samples were taken at intervals between 0 and 8 hours, forming a time series dataset. The aim was to apply the pipeline to both datasets, infer GEPRNs for both ETI inducers and make a comparison between both networks.

For AvrRps4 the gene count at the beginning of the preprocessing was 33681 and after preprocessing 11392. From 2 means clustering we can see that the genes can be divided into two groups: higher and lower log2 fold-change (Figure 15 and 16). The PCA plots of the samples (Fig 20 show that at the earlier time points until 2.5 hours mock and treat show similar expression patterns, after this time they diverge as expected.

DDEG analysis of the 3 tools: original pipeline, current pipeline and SplineTimeR show a large

overlap in predicted DDEG (1209 genes)(Figure 21.). Interestingly the original pipeline and SplineTimeR show a large number of genes that do not overlap. The current pipeline overlaps best with both tools but has a large number of predicted DDEGs that the other models do not find. This enforces our idea that the current pipeline definition of A non-statistical DDEG is fairly loose. However, this is not of the most importance as the following steps can filter out some of the less clear DDEGs from network inference. it is however recommended to get a more strict definition as it can influence the pipeline performance.

Figure 22. shows a clear separation between 2 large groups for both T-SNE and UMAP. This is up- and down-regulation of GEP. Within these 2 groups, a fairly clear separation of clusters is seen as the genes coloured by cluster group together.

To calculate the DTW matrix we use logfold of mock and treat but no normalisation, this allows for 2 patterns to be alike but stay separate clusters. Figure 23. shows an overview of the log foldchange GEP for AvrRps4. Different size clusters can be seen with the smallest containing 25 while the largest containing 1522 genes, this would indicate that a large number of genes follow a very strong pattern.

Cluster GO term annotation provided a heatmap of go term with higher tier GO labelling was created for CC, BP and MD (figure 24, 25 and 26). Clusters 0, 1 and 2 seem to be involved in the start of the immune response, they are enriched in GO terms related to response to stimulus, response to other organism and small molecule binding. Alot of the downregulated clusters are erniched in GO terms related to metabolic processes, meaning during an immune response these procceses slow down to focus on the immune response. This could indicate they are involved in the effector binding which forms the start of the immune response. From these heatmaps, we can see that some clusters are highly specific to a single or few GO terms while others have less common ancestors. A factor in this could be clustering quality, spread-out GO terms for clustering could indicate that a cluster should be split, while a single GO term cluster might indicate that too many clusters are generated. through analysis and parameter optimisation, a researcher should aim for an optimal distribution of clusters where the cluster can be annotated to import GO terms.

The Bingo algorithm was run for 300000 iterations, 15 times more than in the original pipeline. Visible in 27 is a good number of links found with a likelihood above 0.8. the set cut-off could be increased to 0.9. Figure 28 shows a single cluster with almost complete certainty of perturbation, cluster zero. looking at the cluster GEP in figure 23 shows one of the first clusters to increase in its log2-FC expression, reinforcing the expectation of being under the direct influence of the perturbation.
Network analysis is split into two parts, the first is the overlap with the reference network. For both S4 and T2 an overlap network is created and 3 to 4 hub genes are found per network. Almost all genes are transcription factors, which is promising as the ETI response leads to transcriptional reprogramming. This means that the pipeline can find cluster-to-cluster relations in which their genes have known interactions and the pipeline is able to find gene hubs consisting of transcription factors.
The second part of network analysis comapres inferred models from two experiments. Here T2 is compared to S4. From the similarity matrix we identify some clusters which are similar

between the networks however, a lot of clusters have little or no similarity to clusters from the other network. The highest similarity is just above 0.4, Since the method for calculating this cluster-to-cluster similarity is novel, it is highly likely that improvements need to be made. The visualisation of the networks allows for a quick and easy overview.

6 Conclusion

Cellular processes inherently unfold over time. A complete analysis workflow specialised in handling and processing time series data needs to be followed to research such processes as the Effector-Triggered Immunity(ETI). The pipeline created in this thesis fills an existing hole in the field of Dynamic Differentially Expressed Gene(DDEG) analysis. While many tools are available, they are limited in their use due to replicate requirements, written in different languages and methods which are built upon static methods. The created pipeline provides an easy-to-use start-to-end analysis workflow consisting of preprocessing, DDEG prediction, Gene Expression Pattern(GEP) clustering, GEP network inference, and network analysis. Since the is modular, researchers can implement their own methods or swap them out while maintaining a clear overview of the process. While multiple parts of the pipeline can be improved, the pipeline is able to infer a network and highlight transcription factors for plant data.

References

[1] Atte Aalto et al. "Gene regulatory network inference from sparsely sampled noisy data". en. In: *Nature Communications* 11.1 (July 2020), p. 3493. ISSN: 2041-1723. DOI: 10.1038/s41467-020-17217-1.

[2] Takuya Akiba et al. "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[3] Arfa Anjum et al. "Identification of Differentially Expressed Genes in RNA-seq Data of Arabidopsis thaliana: A Compound Distribution Approach". In: *Journal of Computational Biology* 23.4 (Apr. 2016), pp. 239–247. ISSN: 1066-5277. DOI: 10.1089/cmb.2015.0205.

[4] Donald J Berndt and James Clifford. "Using Dynamic Time Warping to Find Patterns in Time Series". en. In: ().

[5] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. arXiv:1603.02754 [cs]. Aug. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785. URL: http://arxiv.org/abs/1603.02754.

[6] The Galaxy Community. "The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update". In: *Nucleic Acids Research* 52.W1 (July 2024), W83–W94. ISSN: 0305-1048. DOI: 10.1093/nar/gkae410.

[7] The UniProt Consortium. "UniProt: the Universal Protein Knowledgebase in 2025". In: *Nucleic Acids Research* 53.D1 (Jan. 2025), pp. D609–D617. ISSN: 1362-4962. DOI: 10.1093/nar/gkae1010.

[8] Pingtao Ding et al. "Chromatin accessibility landscapes activated by cell-surface and intracellular immune receptors". In: *Journal of Experimental Botany* 72.22 (Dec. 2021), pp. 7927–7941. ISSN: 0022-0957. DOI: 10.1093/jxb/erab373.

[9] Liang-Yu Fu et al. "ChIP-Hub provides an integrative platform for exploring plant regulome". en. In: *Nature Communications* 13.1 (June 2022), p. 3413. ISSN: 2041-1723. DOI: 10.1038/s41467-022-30770-1.

[10] Laurent Gautier. *rpy2*. Version 3.5.17. 2024. URL: https://github.com/rpy2/rpy2/releases/tag/RELEASE_3_5_17.

[11] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[12] Xin HE. "Identification of fundamental dynamics from large-scale time-series data for causal interaction networks". English. PhD Thesis. Unilu - University of Luxembourg [The Faculty of Science, Technology and Medicine], Esch-sur-Alzette, Luxembourg, Oct. 2023.

[13] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[14] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[15] Plotly Technologies Inc. *Dash: A Python framework for building reactive web applications*. https://dash.plotly.com/. 2017.

[16] D. V. Klopfenstein et al. "GOATOOLS: A Python library for Gene Ontology analyses". en. In: *Scientific Reports* 8.1 (July 2018), p. 10872. ISSN: 2045-2322. DOI: 10.1038/s41598-018-28948-z.

[17] Liis Kolberg et al. "g:Profiler—interoperable web service for functional enrichment analysis and gene identifier mapping (2023 update)". en. In: *Nucleic Acids Research* 51.W1 (July 2023), W207–W212. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkad347.

[18] Peter Langfelder and Steve Horvath. "WGCNA: an R package for weighted correlation network analysis". eng. In: *BMC bioinformatics* 9 (Dec. 2008), p. 559. ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-559.

[19] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[20] Mary L. McHugh. "Interrater reliability: the kappa statistic". In: *Biochemia Medica* 22.3 (Oct. 2012), pp. 276–282. ISSN: 1330-0962.

[21] Leland McInnes, John Healy, and James Melville. "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction". In: arXiv:1802.03426 (Sept. 2020). arXiv:1802.03426 [stat]. DOI: 10.48550/arXiv.1802.03426. URL: http://arxiv.org/abs/1802.03426.

[22] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[23] Agata Michna et al. "Natural Cubic Spline Regression Modeling Followed by Dynamic Network Reconstruction for the Identification of Radiation-Sensitivity Gene Association Networks from Time-Course Transcriptome Data". en. In: *PLOS ONE* 11.8 (Aug. 2016), e0160791. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0160791.

[24] Vera-Khlara S. Oh and Robert W. Li. "Temporal Dynamic Methods for Bulk RNA-Seq Time Series Data". In: *Genes* 12.3 (Feb. 2021), p. 352. ISSN: 2073-4425. DOI: 10.3390/genes12030352.

[25] Vera-Khlara S. Oh and Robert W. Li. "Temporal Dynamic Methods for Bulk RNA-Seq Time Series Data". In: *Genes* 12.3 (Feb. 2021), p. 352. ISSN: 2073-4425. DOI: 10.3390/genes12030352.

[26] Rob Patro et al. "Salmon: fast and bias-aware quantification of transcript expression using dual-phase inference". In: *Nature methods* 14.4 (Apr. 2017), pp. 417–419. ISSN: 1548-7091. DOI: 10.1038/nmeth.4197.

[27] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[28] Giancarlo Perrone, Jose Unpingco, and Haw-minn Lu. "Network visualizations with Pyvis and VisJS". In: arXiv:2006.04951 (June 2020). arXiv:2006.04951 [cs]. DOI: 10.48550/arXiv.2006.04951. URL: http://arxiv.org/abs/2006.04951.

[29] Mark D. Robinson, Davis J. McCarthy, and Gordon K. Smyth. "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data". eng. In: *Bioinformatics (Oxford, England)* 26.1 (Jan. 2010), pp. 139–140. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp616.

[30] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (Feb. 1978), pp. 43–49. ISSN: 0096-3518. DOI: 10.1109/TASSP.1978.1163055.

[31] Marcel H. Schulz et al. "DREM 2.0: Improved reconstruction of dynamic regulatory networks from time-series expression data". In: *BMC Systems Biology* 6.1 (Aug. 2012), p. 104. ISSN: 1752-0509. DOI: 10.1186/1752-0509-6-104.

[32] Charlotte Soneson, Michael I. Love, and Mark D. Robinson. "Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences". eng. In: *F1000Research* 4 (2015), p. 1521. ISSN: 2046-1402. DOI: 10.12688/f1000research.7563.2.

[33] Himangi Srivastava, Drew Ferrell, and George V. Popescu. "NetSeekR: a network analysis pipeline for RNA-Seq time series data". In: *BMC Bioinformatics* 23.1 (Jan. 2022), p. 54. ISSN: 1471-2105. DOI: 10.1186/s12859-021-04554-1.

[34] Jasmin Straube et al. "A Linear Mixed Model Spline Framework for Analysing Time Course 'Omics' Data". In: *PLoS ONE* 10.8 (Aug. 2015), e0134540. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0134540.

[35] Romain Tavenard et al. "Tslearn, A Machine Learning Toolkit for Time Series Data". In: *Journal of Machine Learning Research* 21.118 (2020), pp. 1–6. URL: http://jmlr.org/papers/v21/20-091.html.

[36] Hande Topa and Antti Honkela. "GPrank: an R package for detecting dynamic elements from genome-wide time series". In: *BMC Bioinformatics* 19.1 (Oct. 2018), p. 367. ISSN: 1471-2105. DOI: 10.1186/s12859-018-2370-4.

[37] Edy Umargono, Jatmiko Endro Suseno, and S.K Vincensius Gunawan. "K-Means Clustering Optimization Using the Elbow Method and Early Centroid Determination Based on Mean and Median Formula". In: *Proceedings of the 2nd International Seminar on Science and Technology (ISSTEC 2019)*. Atlantis Press, 2020, pp. 121–129. ISBN: 978-94-6239-168-0. DOI: 10.2991/assehr.k.201010.019. URL: https://doi.org/10.2991/assehr.k.201010.019.

[38] Michael L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[39] Minhang Yuan et al. "PTI-ETI crosstalk: an integrative view of plant immunity". In: *Current Opinion in Plant Biology*. Biotic interactions 62 (Aug. 2021), p. 102030. ISSN: 1369-5266. DOI: 10.1016/j.pbi.2021.102030.

[40] Jure Zmrzlikar et al. "Normalizing RNA-seq data in Python with RNAnorm". en. In: ().

7 Supplementary materials

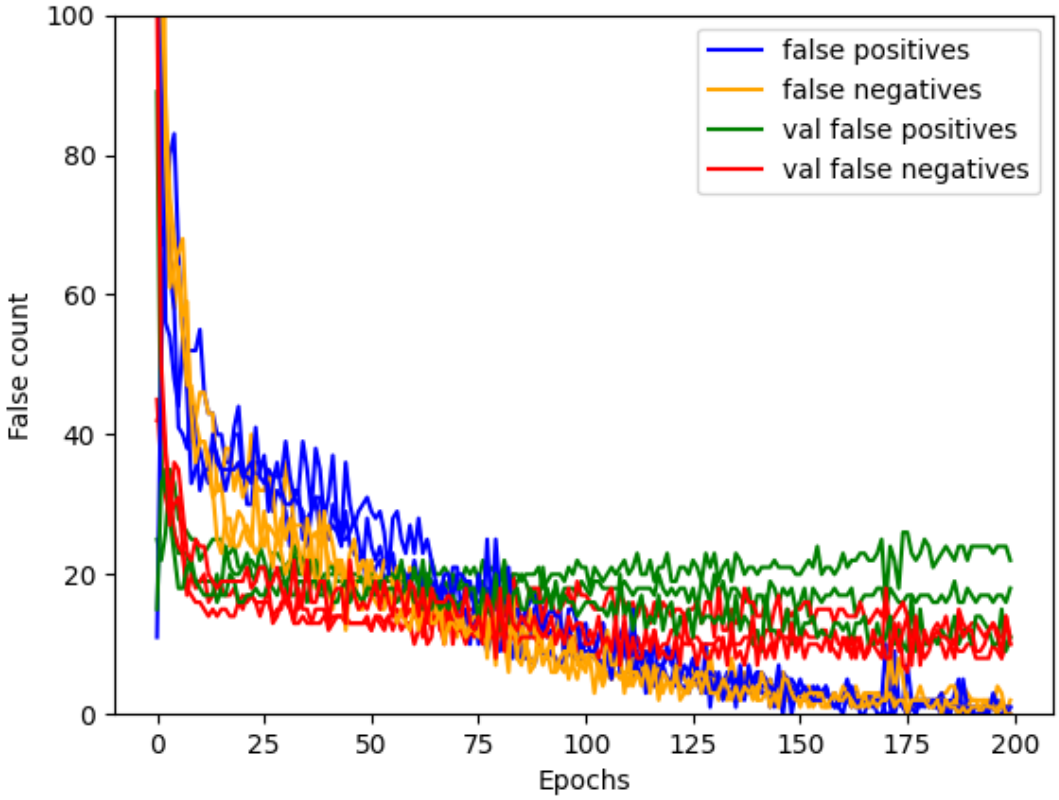

Figure 39: False counts plotted against the epochs for a 3-fold training for one of the best settings 1 after optuna study. Blue and yellow lines represent false positive and false negative counts respectively on the training dataset while green and red represent the false counts on the test dataset.

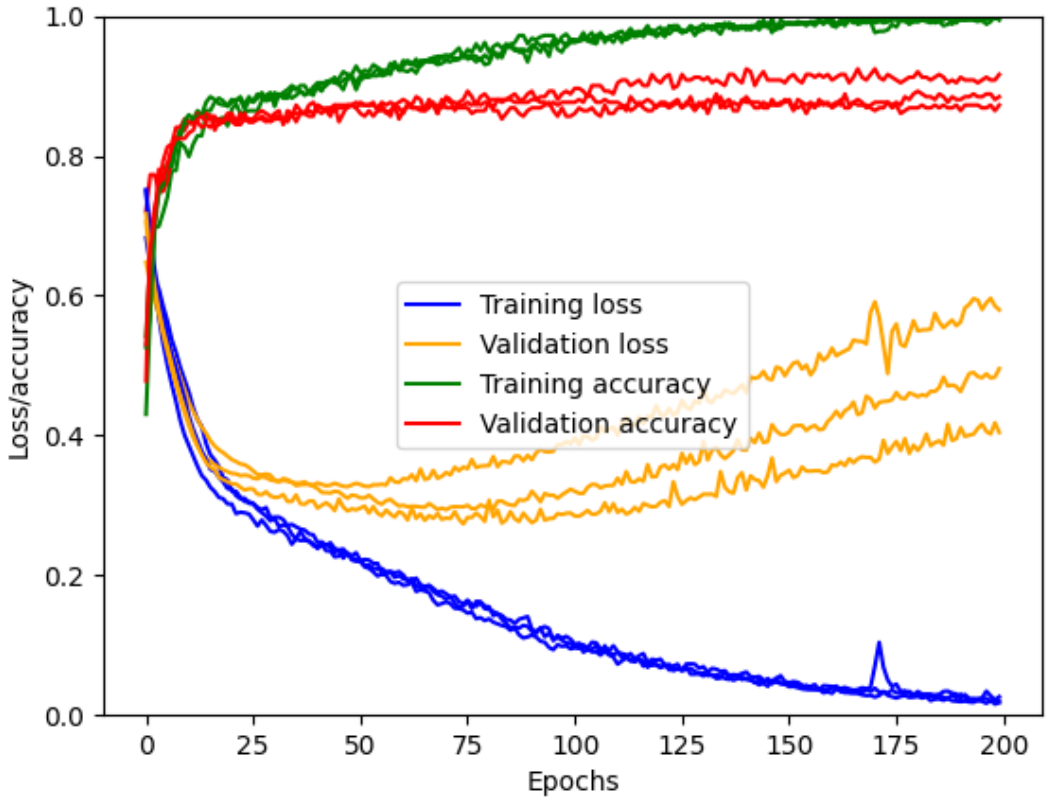

Figure 40: loss and accuracy plotted against the epochs for a 3-fold training for one of the best settings 1 after optuna study. Blue and yellow lines represent train and validate loss respectively while green and red represent the train and validate accuracy.

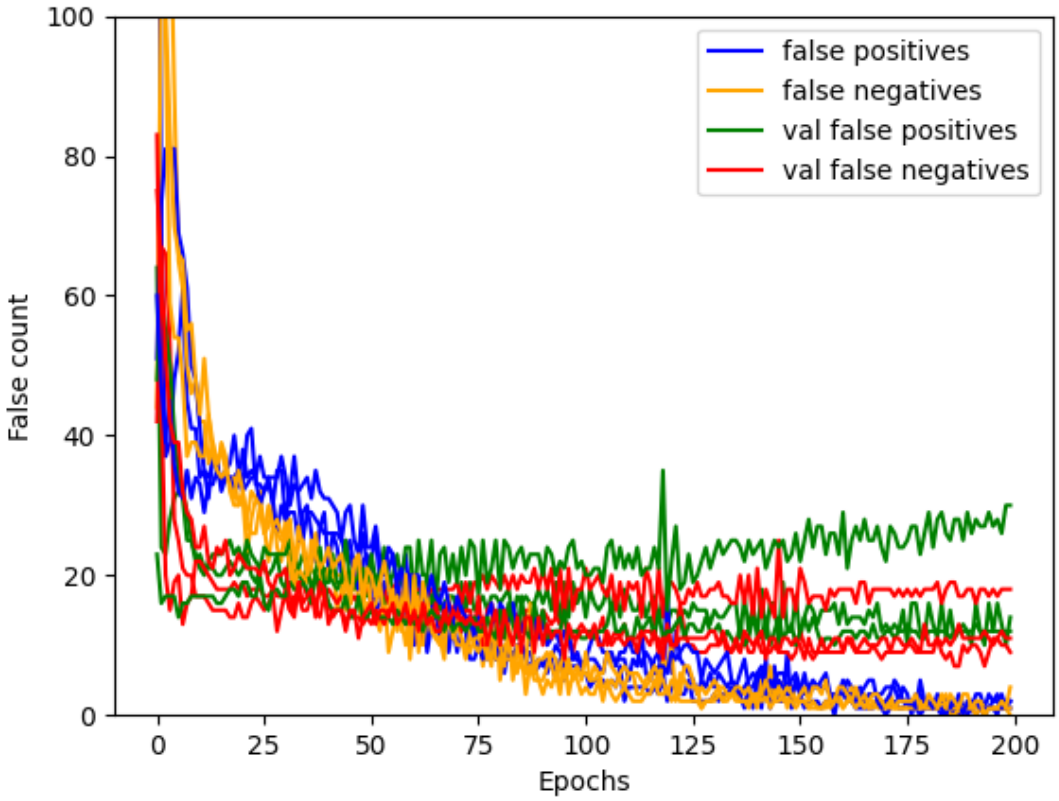

Figure 41: False counts plotted against the epochs for a 3-fold training for one of the best settings 2 after optuna study. Blue and yellow lines represent false positive and false negative counts respectively on the training dataset while green and red represent the false counts on the test dataset.

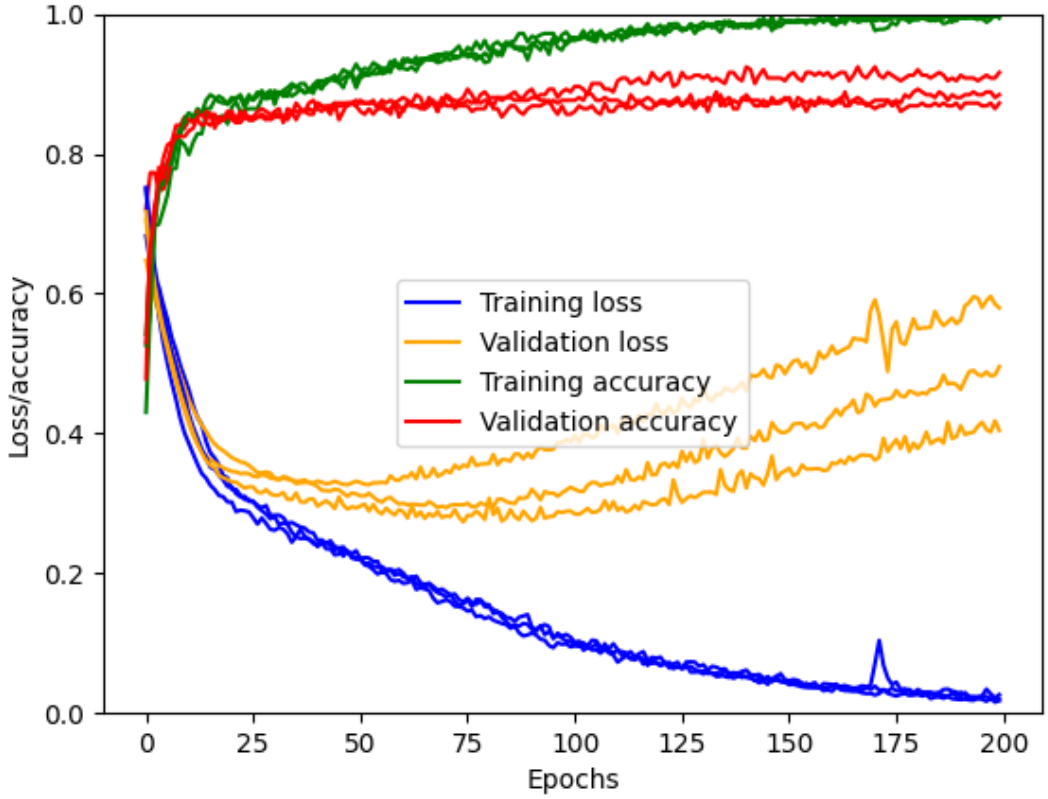

Figure 42: loss and accuracy plotted against the epochs for a 3-fold training for one of the best settings 2 after optuna study. Blue and yellow lines represent train and validate loss respectively while green and red represent the train and validate accuracy.

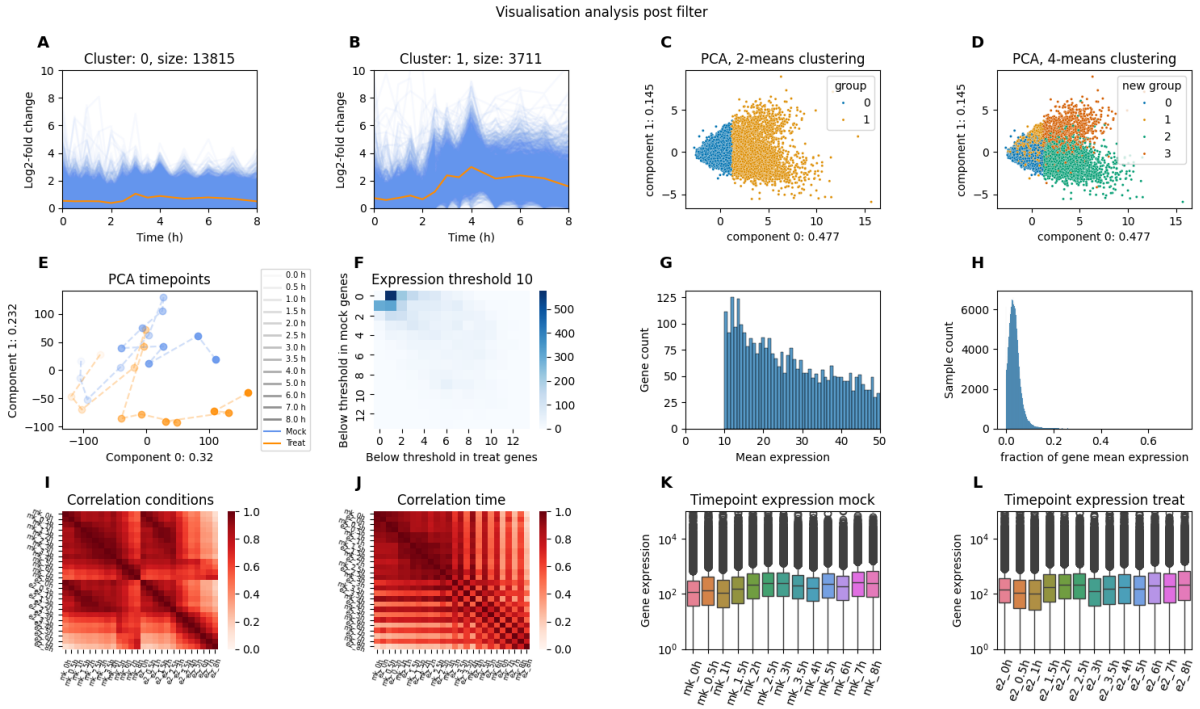

Figure 43: Data visualisation after filtering is applied. Information about the subplots can be found in Table 2.

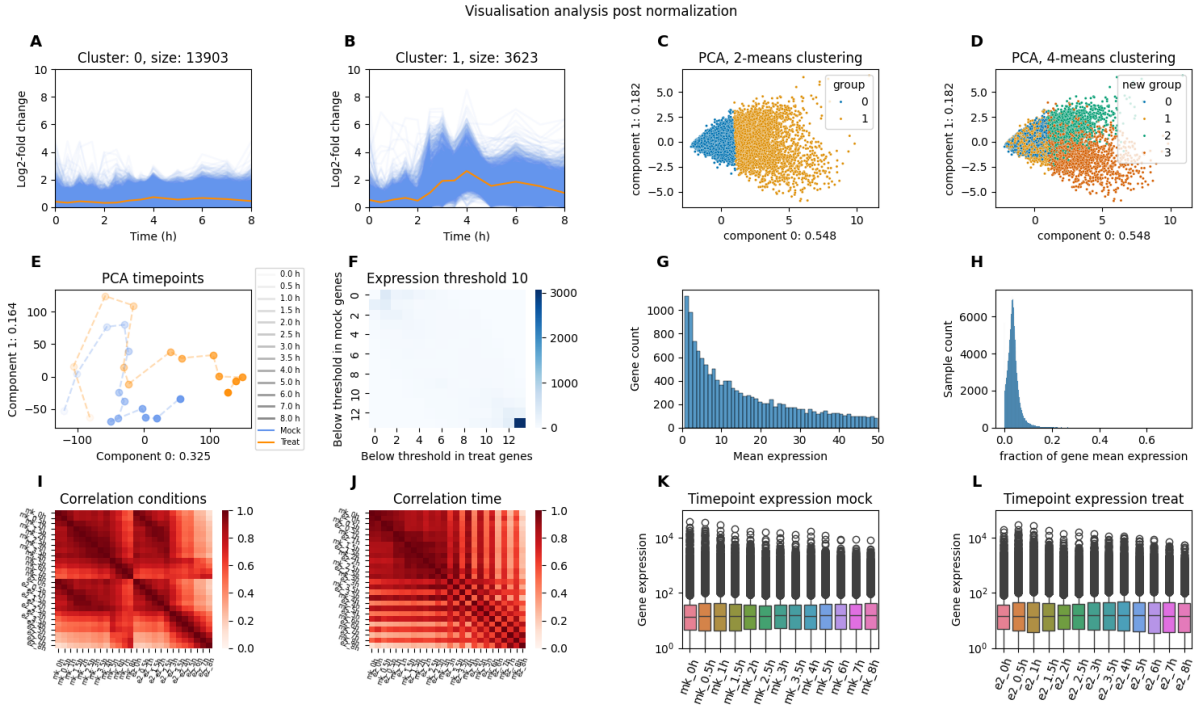

Figure 44: Data visualisation after normalisation is applied. Information about the subplots can be found in Table 2.

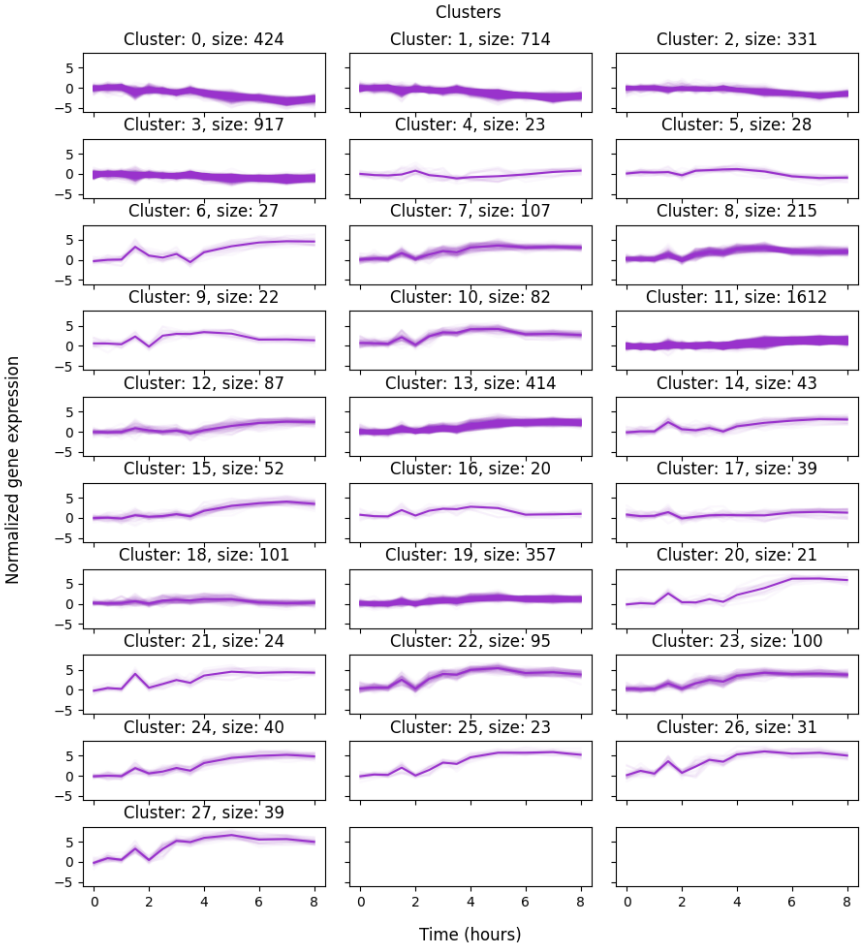

Figure 45: Subplots containing log2-FC time series genes and their averaged centroids for T2.

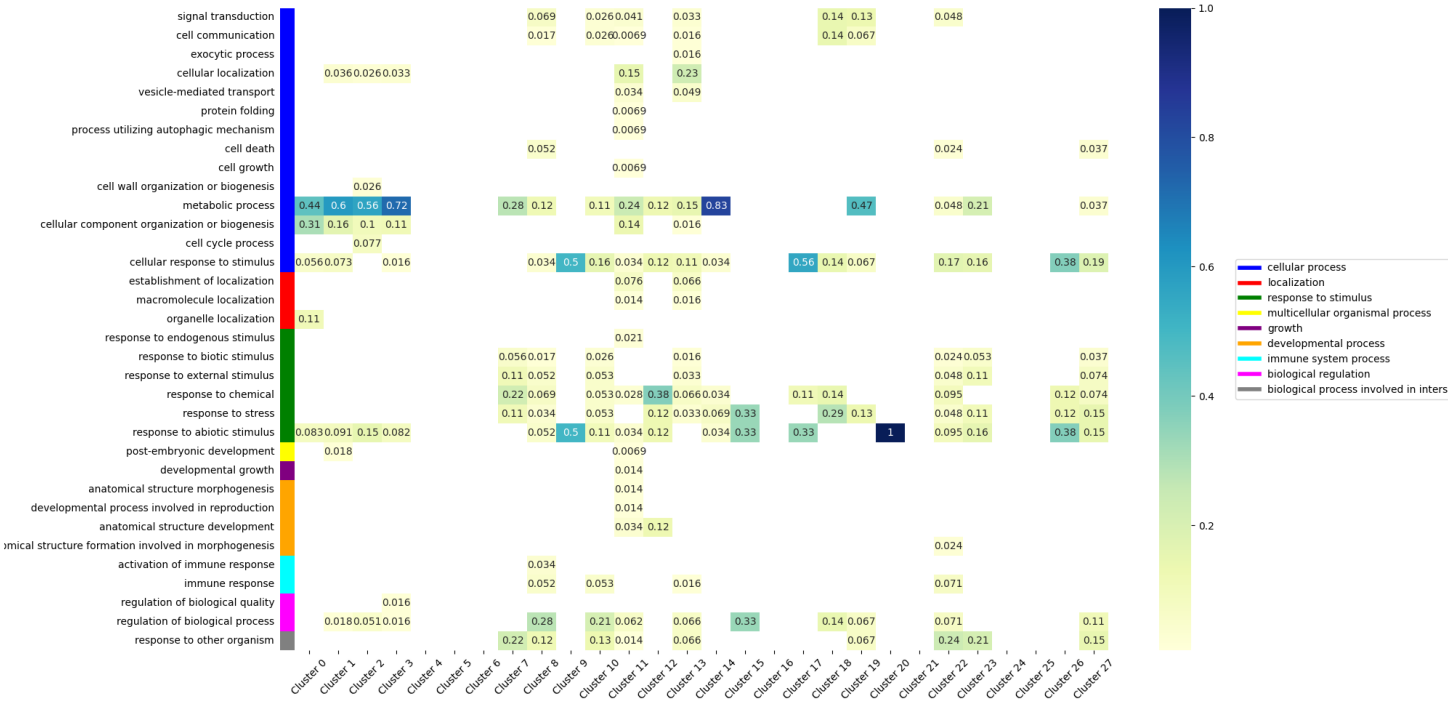

Figure 46: T2 heatmap setting out the second layer of GO terms under biological processing annotation against clusters, colour labelled by first layer GO term.

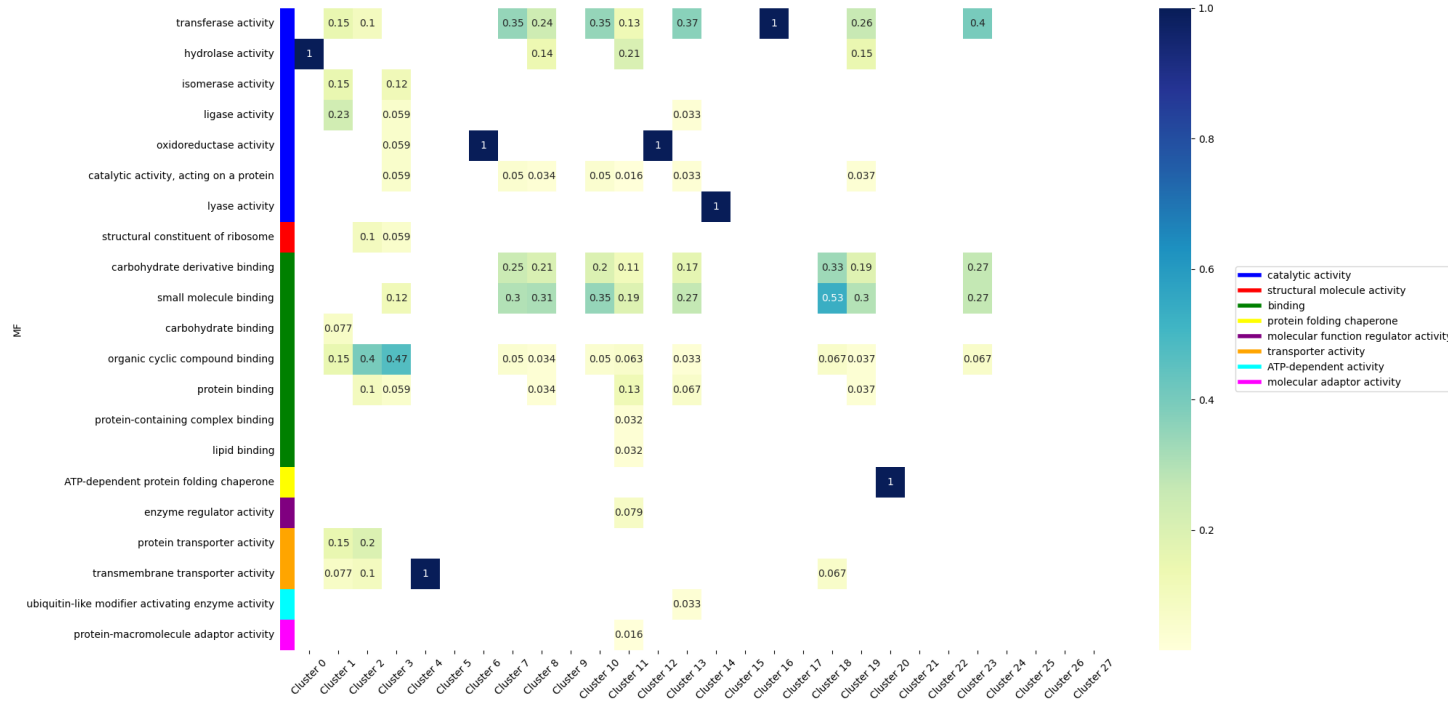

Figure 47: T2 heatmap setting out the second layer of GO terms under molecular function annotation against clusters, colour labelled by first layer GO term.

Figure 48: T2 heatmap setting out the second layer of GO terms under cellular component annotation against clusters, colour labelled by first layer GO term.