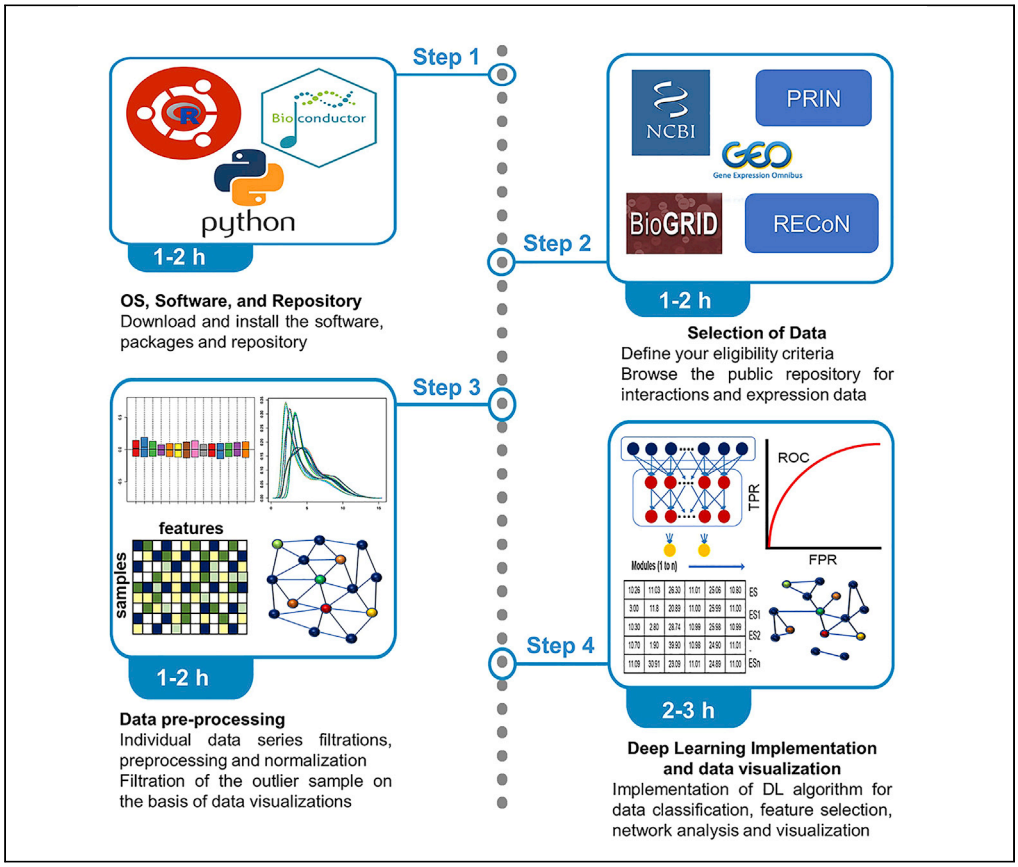
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Deep learning based protocol to construct an immune-related gene network of host-pathogen interactions in plants



Investigating network behavior from host-pathogen interactions is challenging. Here, we present the deep-learning-based protocol to construct an immune-related gene network and list the genes involved in the defense response of host to specific biotic stress. The protocol includes the steps to pre-process the interaction pairs and expression profile of plants treated with pathogen/control, feed as input for DLNet algorithm to rank genes based on their contribution to data classification. The top-ranked genes are subjected to module and enrichment analysis.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Ravi Kumar, Vishal Acharya

vishal@ihbt.res.in

Highlights

Downloading the plant data treated with pathogen(s), filtering interaction pairs

Ranking of genes by DLNet algorithm, contributing to host defense against pathogen

Constructing host gene network modules from top-ranked extracted features

Gene set enrichment to select the significant modules and network visualization

Kumar & Acharya, STAR Protocols 4, 101934
March 17, 2023
https://doi.org/10.1016/j.xpro.2022.101934

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Deep learning based protocol to construct an immune-related gene network of host-pathogen interactions in plants

Ravi Kumar[1,2,3] and Vishal Acharya[1,2,4,*]

[1]Functional Genomics and Complex System Lab, Biotechnology Division, CSIR-Institute of Himalayan Bioresource Technology (CSIR-IHBT), Palampur, Himachal Pradesh, India

[2]Academy of Scientific and Innovative Research (AcSIR), Ghaziabad 201002, India

[3]Technical contact: ravisaroch@gmail.com

[4]Lead contact

*Correspondence: vishal@ihbt.res.in
https://doi.org/10.1016/j.xpro.2022.101934

SUMMARY

Investigating network behavior from host-pathogen interactions is challenging. Here, we present the deep-learning-based protocol to construct an immune-related gene network and list the genes involved in the defense response of host to specific biotic stress. The protocol includes the steps to pre-process the interaction pairs and expression profile of plants treated with pathogen/control, feed as input for DLNet algorithm to rank genes based on their contribution to data classification. The top-ranked genes are subjected to module and enrichment analysis.

For complete details on the use and execution of this protocol, please refer to Kumar et al. (2022).[1]

BEFORE YOU BEGIN

The protocol below details the step-by-step processes for generating and analyzing gene networks under specific stress conditions on *Oryza sativa* (rice) Affymetrix microarray data. The protocol's advantages include the deep learning approach for identifying the top-ranking genes that participate in the classification of data of two specific conditions (Treatment/Control), which may contribute to the plant defense and analysis of the network hub genes for downstream study. The protocol facilitates the end-to-end gene expression profile analysis, integrated with the gene network for identifying the potential target genes.

Necessary operating system and hardware

Timing: ~1 h (for step 1)

This section includes the minimal hardware, operating system and software requirements.

1. Install Ubuntu 16.04 or greater unless such an operating system is already available. The minimum requirement for the computer hardware, operating system, and software recommendations:

Ubuntu OS ≥ 16.04 LTS, minimum RAM 16 GB, python version ≥ 3.6, pip version ≥ 19.0 and, R package ≥ 3.5.

Note: Ubuntu 20.04 LTS operating system installed on a 20-core, 256 GB RAM, and 2TB ROM machine to confirmed the data reproducibility of our published work.[1]

Install required software and libraries

Timing: ~1 h (for step 2)

This section includes the installation of all the software's and libraries of R and Python packages.

2. After installing the recommended operating system, the software must be installed using pip with sudo privilege.
   a. Install the list of software by executing the following commands on the Ubuntu terminal:

```
>sudo apt-get install python3-pip && sudo apt-get install r-base

>sudo apt-get install mcl
```

   b. Install the python libraries by executing the following commands on the Ubuntu terminal:

```
>pip install tensorflow && pip install -U scikit-learn && pip install pandas

>pip install xgboost && pip install matplotlib
```

3. Install all the R packages. Execute the R as root privilege in the Ubuntu terminal:

```
>sudo R
```

4. Execute the following commands in the Ubuntu terminal under the R environment:
   a. Install the workflow packages from Bioconductor: To install the packages with Bioconductor, a user must need to install the BiocManager version as compatible with R (troubleshooting 1).

```
>if (!require("BiocManager", quietly = TRUE))

>install.packages("BiocManager")

>BiocManager::install()
```

   b. Install the workflow packages from GitHub: To install the GitHub packages user must need to install devtools:

```
>if(!require(devtools)){

>install.packages("devtools")

>}

>devtools::install_github()
```

   c. Install the required R packages via BiocManager.

The following packages are needed for the data pre-processing: affy,[2] simpleaffy,[3] and affyPLM.[4] Download these packages with other packages that are required for downstream analysis.

```
>BiocManager::install(c("GEOquery","Biobase","BiocGenerics","simpleaffy","affy",
"genefilter","gcrma","affyPLM","preprocessCore","RColorBrewer","ggplot2","biomaRt",
``limma", "inSilicoMerging"))
```

⚠ CRITICAL: if some of the packages are not available for R specific version (preprocessCore), then use them using devtools (troubleshooting 2).

```
>devtools::install_github("bmbolstad/preprocessCore")
```

5. Now type the below snippet in the Ubuntu terminal R environment to exit the R.

```
>quit()
```

⚠ CRITICAL: The below packages are required to install GEOquery. Install these packages by copying the commands on the Ubuntu terminal if not installed previously:

```
>sudo apt-get install libxml2-dev

>sudo apt-get install r-base-core libssl-dev libcurl4-openssl-dev
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Raw expression data	Barrett et al.[5]	https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GPL2025
Protein-protein interactions	BioGRID PRIN	https://thebiogrid.org/ http://bis.zju.edu.cn/prin/
Under abiotic stress, co-expression interactions pairs	RECoN	https://plantstress-pereira.uark.edu/RECoN/
Supplemental information	Mendeley Data	https://doi.org/10.17632/4yh3d8xmbg.1
Affymetrix sample (CEL)	Present Study, Kumar et al.[1]	Table S1, https://www.cell.com/iscience/fulltext/S2589-0042(22)00818-5
Code availability	Present study	https://gitlab.com/ravisaroch/star-protocols/
Software and algorithms		
Python3	Python v3.8	https://www.python.org/
NumPy	NumPy v1.21	https://numpy.org/
Pandas	Pandas v1.4	https://pandas.pydata.org/
Sklearn	Sklearn v1.0	https://scikit-learn.org/stable/
Tensorflow	Tensorflow v2	https://www.tensorflow.org/
SciPy	SciPy v1.7	https://scipy.org/
Matplotlib	Python Library	https://matplotlib.org/
Mcl	Enright et al.[6]	https://micans.org/mcl/
Random Forest	Sklearn v1.1	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier/
XGBoost	Sklearn v1.1	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier/
R	R v3.5	https://cran.r-project.org/
Affy	Gautier et al.[2]	https://www.bioconductor.org/packages/release/bioc/html/affy.html, https://rdrr.io/bioc/affy/

(Continued on next page)

Continued		
REAGENT or RESOURCE	SOURCE	IDENTIFIER
simpleaffy	Miller[3]	https://www.bioconductor.org/packages//2.7/bioc/html/simpleaffy.html, https://rdrr.io/bioc/simpleaffy/
affyPLM	Brettschneider et al.[4]	https://www.bioconductor.org/packages/release/bioc/html/affyPLM.html, https://rdrr.io/bioc/affyPLM/
inSilicoMerging	Taminau et al.[7]	https://www.bioconductor.org/packages//2.10/bioc/html/inSilicoMerging.html, https://rdrr.io/bioc/inSilicoMerging/
biomaRt	Durinck et al.[8,9]	https://www.bioconductor.org/packages/release/bioc/html/biomaRt.html, https://rdrr.io/bioc/biomaRt/
Other		
Intel® Xeon(R) Silver 4114 CPU @ 2.20 GHz × 40 Ubuntu 20.04.5 LTS		

STEP-BY-STEP METHOD DETAILS

Here we describe the step-by-step methods for download the repository, download all-necessary data and their filtering process, procedure to use the repository (DLNet) from generating the score for each gene that participate to differentiate the data of two conditions (control/treatment), and construct the gene network for the identification of the target genes.

Downloading the repository

Timing: ~5–10 min (for step 1)

This section includes the different steps to download the repository from the GitLab.

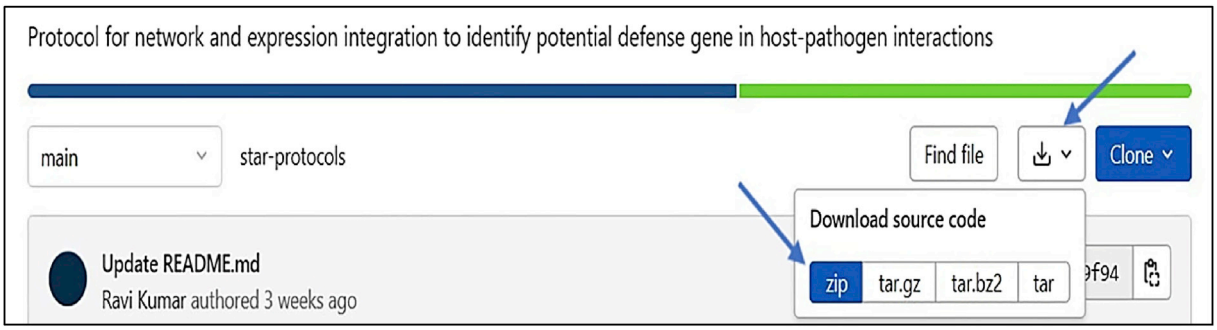
1. If the git command is not previously installed in your Debian/Ubuntu system, run the following:

```
>sudo apt-get install git-all
```

Otherwise, download the repository by executing the below command on the Ubuntu terminal:

```
>git clone https://gitlab.com/ravisaroch/star-protocols.git
```

2. The repository can be added directly by going to https://gitlab.com/ravisaroch/star-protocols and clicking on the download button, then selecting "zip" to download the pipeline to your system.



Data collection

Timing: ~1–2 h (for step 3)

Here we describe the various steps for downloading gene expression and network data from different open-source databases.

3. Collection of PPI interactions, gene co-expressed interactions and case-control gene expression datasets.
   a. Collect the experimental and predicted protein-protein interaction (PPI) data of particular species from interaction data databases:
      i. Biological General Repository for Interaction Datasets (BioGRID; https://thebiogrid.org).[10]
      ii. Search Tool for the Retrieval of Interacting Genes/Proteins (STRING; https://string-db.org).
      iii. In case of rice, download the interaction data from Predicted Rice Interactome Network (PRIN) databases.[11]
   b. The gene co-expression profile at different development and stress conditions from plant species-specific databases.

   Note: In the case of rice, download the dataset from the Rice Environment Coexpression Network (RECoN)[12] via following link: https://zenodo.org/record/3257604#.YzLB_3ZBy3B.

   c. Download the gene expression profile data for particular stress condition and their respective control from the Gene Expression Omnibus (GEO) (www.ncbi.nlm.nih.gov/geo) (Microarray Affymetrix chip data) of the National Center for Biotechnology Information.
   d. Please visit steps 5 and 6 to detail information about downloading the Affymetrix data.

⚠ CRITICAL: If the user wishes to analyze the gene interaction network under bacterial pathogen biotic stress conditions, the plant gene co-expression data under bacterial biotic stress conditions should not be included in the interaction data file. Similarly, if someone wants to examine the gene network under drought stress, try not to include co-expression data of drought stress in the interaction data file.

Filter the PPI and gene co-expression interaction data

Timing: ~1 h (for step 4)

This section includes the different steps to extract the high-quality protein-protein interaction pairs and filtration process to extract the confidence gene co-expression pair.

4. The PPI and gene co-expression data undergo a filtering process to obtain high-quality pair of interactions.
   a. The authenticity of interaction evaluates using the gene ontology term. The interactions consider important if genes in the pair have at least one common gene ontology (GO)[13] and store the filter pair in a space-separated .txt file (i.e., Interaction_PPI). of the interaction of the genes.
      i. To begin, download all genes and associated gene ontologies from the tools mentioned above and retain just the columns containing locus ids and gene ontology terms.

```
>cat GO_plantGSEA | awk '{print $5, $2}' >raw && mv raw GO_plantGSEA

>cat GO_agriGO | awk '{print $5, $2}' >raw && mv raw GO_agriGO
```

Note: Please visit the file once downloaded and select the column cautiously.

ii. Now, concatenate the two files.

```
>cat GO_plantGSEA GO_agriGO >interaction_GO
```

iii. Compare your original interaction file with the interaction_GO file. If the locus ids in the interaction pair contain at least one similar GO term, keep them; otherwise, discard them from the file.

Note: agriGo v2.0,[14] plantGSEA[15] web tools were used to extract the Gene Ontology (GO) of the pair.

b. Filter the gene co-expression profile downloaded from different databases.

i. Analyze the interaction pair and retain the pairs with a Pearson correlation coefficient (PCC) $\geq$ 0.7 because PCC values between 0.7 and 1.0 show strong linear relations and are identified as potential candidates for the interaction pair.

```
>awk '$4 >= 2' filename | awk '{print $1, $2}' > FCO
```

Note: "filename" is the file name in which interaction was stored (step 4a).

ii. Analyze the output files (FCO) having PCC of the pair $\geq$ 0.7 and retain the interaction pairs having one of the genes in the pair corresponding to the transcription factor (TF). To do this, download the list of the TF of species of interest (*Oryza sativa* in the present case) from the Plant Transcription Factor Database (http://planttfdb.gao-lab.org/).

```
>gunzip *_TF_list.txt.gz

>mv *_TF_list.txt TF_file

>awk -F"\t" '{print $2}' TF_file | awk '!a[$0]++' >list_TF
```

Note: The above commands decompress all the files that include a list of transcription factors from various families. The next command moves the list of transcription factors from separate files to a single concatenated file, and the final command contains each transcript factor once to remove data duplication.

iii. Extract the filter pair having at least one TF in the gene pair, and copy the below awk script to the Ubuntu terminal.

```
>awk ' FNR==NR{a[$0]=$0

> next

>}{

> for (i=1; i<=NF; i++) {

>  if ($i in a) {

>  print a[$i],$0 | "sort -k1"

>    }

>  }

>}' list_TF FCO | awk '{print $2,$3}' | awk '!a[$0]++'

>Filtered_Interaction
```

iv. Concatenate the file of filtered PPI (Interaction_PPI) and the file of filtered co-expression (Filtered_Interaction) and extract the unique pair of genes by copying the below command on the Ubuntu terminal:

```
>cat Interaction_PPI Filtered_Interaction | awk '!(a[$1FS$2]++ || a
[$2FS$1]++)'>Interaction
```

Note: "Interaction" is the file that contains the final pair of interactions after the filtering process.

Download the Affymetrix expression profile data

Timing: ~ 1–2 h (depending on the number of samples) (for step 5)

This section includes in details for downloading the gene expression data (Treatment/Control) from the NCBI database through command-line or graphical user interface.

5. Download the expression data profiles from GEO (NCBI) using the R library GEOquery[16] or download directly by accessing the GEO database.

Note: This protocol's main aim is to identify genes expressed during the given stress condition and the network architecture of these genes in the same condition. As such, the microarray data samples are required from treatment and control conditions for the species of interest (in this case, one of the model plant organisms is *Oryza sativa*).

a. Download the raw .CEL files of the specific series using the GEOquery package use the below command:

```
>sudo R

>library(GEOquery)

>getGEOSuppFiles("GSE69235")
```

The above command creates the directory GSE69235, which contains another directory GSE69235 RAW.tar.

b. Use the following command to decompress this directory:

```
>tar -xvf GSE69235/GSE69235_RAW.tar
```

It downloads all the .CEL files of the GSE69235 series. Keep relevant .CEL data files and eliminate the remainder by manual selection.

c. Now, using the commands listed below, decompress the retained files.

```
>gzip -d GSE69235/ GSE69235_RAW/*CEL.gz
```

d. To download the sample directly from a browser, we look for rice data samples treated with the bacterial pathogen *Xanthomonas oryzae* and their respective control samples in the GEO database.[5]
   i. GEO datasets may be found at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi.
   ii. Select platforms from the Browse Content menu.
   iii. In the search field, insert Affymetrix Rice Genome Array.

Note: You may readily get the data set by inputting the accession number at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi.

iv. In the GEO accession search box, enter the accession number (In the case of rice, this is GPL2025).
v. There are several GSE series providing experiment descriptions. To view the details of the control and treatment samples, click on the GSE series. To get the expression profile, click on the RAW.tar file for each series at the bottom of the page.

Pre-processing and normalization of the Affymetrix data

Timing: ~ 1–2 h (for step 6)

This section provides the detailed steps for pre-processing and normalization of the raw Affymetrix (.CEL) data, graphical representation of the individual data set, and identification of the MSU locus id for each affy id of the data set.

6. Download the multiple series of experiments of interest.

Note: The zipped directory first needed to unzip the raw samples (.CEL files) data, and only the CEL files relevant to the research were maintained. In contrast, the remaining CEL files were deleted from the specified directory.

Note: In the current study, we employed different series of experiments for the treatment and control samples, including GSE69235, GSE19844, GSE3341, GSE34192, GSE43050, GSE36272, GSE49242, GSE61832, GSE61833, GSE53940, and GSE34192. A complete list of samples is provided in Table S1. Using GSE69235 as an example, describe the reading, normalization, and downstream steps below.

7. Open the Ubuntu terminal and run the snippet below to launch the R environment with sudo privileges.

```
>sudo R
```

8. To access the various R libraries, use the commands listed below in the R environment.

```
>library(GEOquery)
>library(Biobase)
>library(BiocGenerics)
>library(parallel)
>library(affy)
>library(genefilter)
>library(gcrma)
>library(simpleaffy)
>library(affyPLM)
>library(preprocessCore)
```

```
>library(RColorBrewer)

>library(ggplot2)

>library(biomaRt)

>library(devtools)

>library(inSilicoMerging)
```

9. Downstream analysis for GSE69235 experimental series data.

Note: Use the simpleaffy package, read the specific file define the phenotypic dataset of the CEL file. The format for the covdesc file for the GSE69235 series data is provided in the link described as: https://gitlab.com/ravisaroch/star-protocols/-/blob/main/RAW_FILTERED_DATA/covdesc_GSE69235.

a. Read the manually generated covdesc file and the raw .CEL file from the directory (trouble-shooting 3).

```
>GSE69235_raw <- read.affy("covdesc_GSE69235", path = "/path/to/CEL/directory/GSE69235_RAW")
```

⚠ CRITICAL: Sample directory's covdesc files were utilize to read the data as data frame. It is a white space delimited file with the first column contain the name or entire path of the .CEL file in the directory and the second column containing the experimental factor. These are the components of the phenotypic experiment.

b. Create the expression matrix using Robust Multi-array Average (RMA) method.

```
>GSE69235_rma <- call.exprs(GSE69235_raw, "rma")

>GSE69235_eset <- exprs(GSE69235_rma)
```

c. Save the normalized data in an external text file.

```
>write.exprs(GSE69235_rma, file="GSE69235_rma")
```

d. Perform the quality control test for the identification of the outlier (if available).

```
>GSE69235_distance <- dist(t(GSE69235_eset), method = "maximum")

>GSE69235_cluster <- hclust(GSE69235_distance)
```

e. Visualization of the series data for the quality control analysis.
   i. Cluster formation of the GSE series data for the identification of the outlier samples (Figure 1A)

```
>png("GSE69235_cluster.png")

>plot(GSE69235_cluster)

>dev.off()
```

   ii. Heatmap of all CEL files of individual series (Figure 1B).

```
>png("GSE69235_Heatmap.png")

>par(oma=c(7,0,1,0))
```

```
>heatmap(exprs(GSE69235_rma[1:200,]), main = "Heatmap of top 200 Genes")
>dev.off()
```

iii. A standardized way of displaying the distribution of pre-normalized and normalized data using a boxplot (Figures 1C and 1D).

```
>png("GSE69235_boxplot.png")
>par(mar=c(12,4,4,2)+0.1)
>par(mfrow=c(1,2))
>boxplot(GSE69235_raw, las = 2, col = brewer.pal(6,"Set1"), main="Pre Normalized Data")
>boxplot(GSE69235_rma, las = 2, col = brewer.pal(6,"Set1"), main="Normalized Data")
>dev.off()
```

iv. Frequency distribution of data using a histogram (Figures 1E and 1F).

```
>png("GSE69235_Histogram.png")
>par(mfrow=c(1,2))
>hist(GSE69235_raw, col=brewer.pal(9, "Set1"), main = "Pre Normalized Data")
>hist(GSE69235_rma, col=brewer.pal(9, "Set1"), main = "Normalized Data")
>dev.off()
```

v. Identification of the variation in the different CEL files individual series (Figures 1G and 1H). Unwanted data fluctuation may be pretty harmful. Box plots of Normalized Unscaled Standard Errors (NUSE)[4] and Relative log expression (RLE)[17] are two effective tools for visualizing variations in high-dimensional data and have been used in recent studies.[18] The corrected arrays may be identified in the boxplot by being centered at 0 in the RLE plot and at 1 in the NUSE plot.

```
>GSE69235.qc <- fitPLM(GSE69235_raw)
>png("GSE69235_RLE.png")
>RLE(GSE69235.qc, col = brewer.pal(9,"Set1"), main = "RLE_GSE69235")
>dev.off()
>png("GSE69235_NUSE.png")
>NUSE(GSE69235.qc, col = brewer.pal(9,"Set1"), main = "NUSE_GSE69235")
>dev.off()
```

Note: Similarly performed step 9 to all data series for the downstream analysis.

10. Removal of Non-biological experimental variation.

The steps include removing the batch effect using Combining Batch Effect When Batches of Gene Expression Microarray Data (Combat),[19] which is part of inSilicoMerging packages.[7]

Note: Microarray data is frequently subject to significant variability due to batch effect across the multiple microarray experiments, and the task of combining the experiment to downstream analysis is difficult. So, removing the batch effect is necessary for the statistical significance of the combined data. The benefit of integrating batches of genomic data to increase

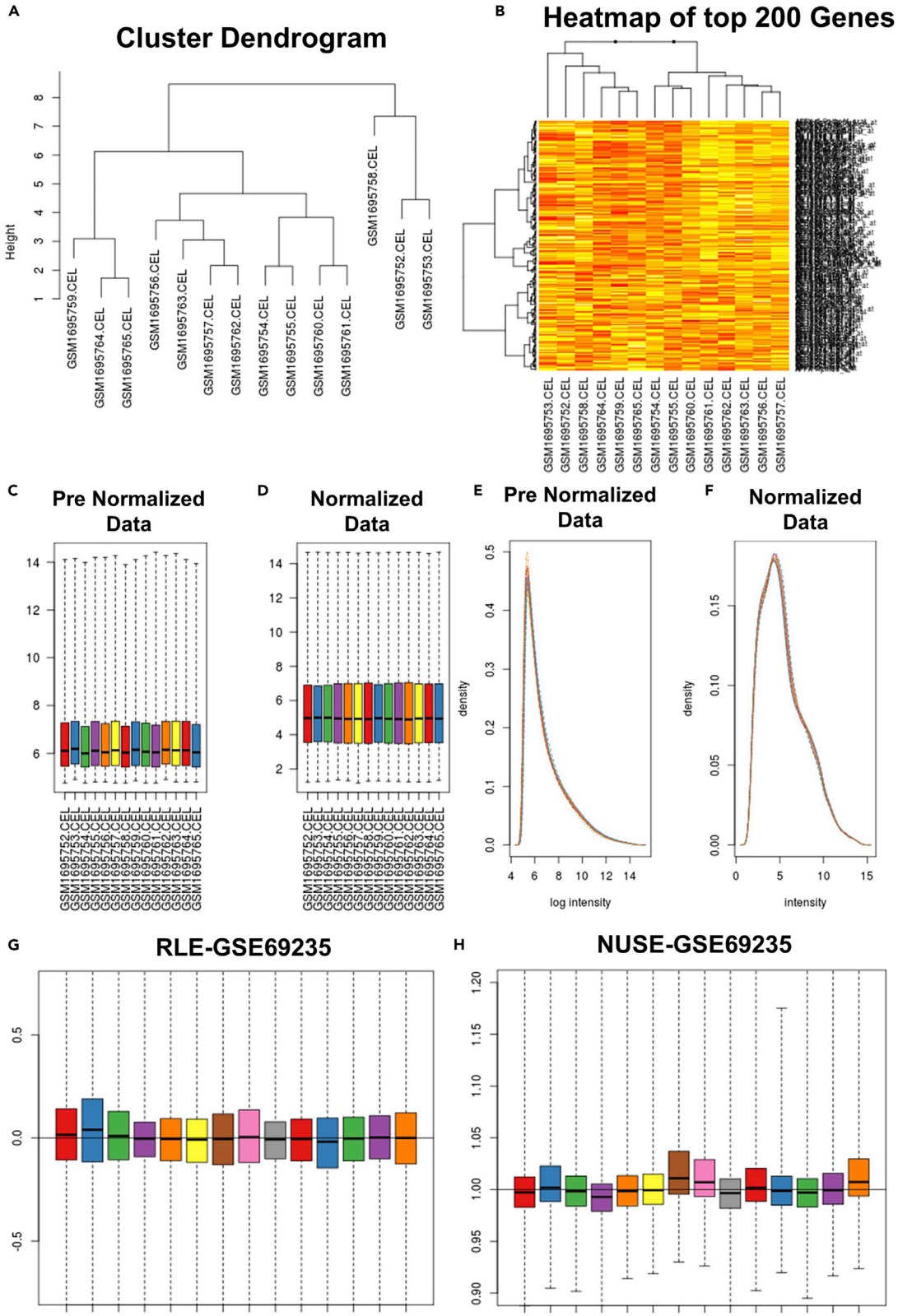

A
Cluster Dendrogram
Height
GSM1695759.CEL
GSM1695764.CEL
GSM1695765.CEL
GSM1695756.CEL
GSM1695763.CEL
GSM1695757.CEL
GSM1695762.CEL
GSM1695754.CEL
GSM1695755.CEL
GSM1695760.CEL
GSM1695761.CEL
GSM1695758.CEL
GSM1695752.CEL
GSM1695753.CEL
B
Heatmap of top 200 Genes
C
Pre Normalized Data
D
Normalized Data
E
Pre Normalized Data
density
log intensity
F
Normalized Data
density
intensity
G
RLE-GSE69235
H
NUSE-GSE69235

Figure 1. Graphical representation of pre-processing of the data of GSE69235
(A) The cluster dendrogram represents the relationship within the CEL data files of GSE69235.
(B) Heatmap of the top 200 genes to show the expression of the genes in different samples of series.
(C and D) Boxplot shows the distribution of pre-normalized and normalized data.
(E and F) Histogram representation of pre-normalized and normalized data.
(G and H) Relative Log Expression (RLE) and Normalized Unscaled Standard Errors (NUSE) data distribution.

statistical power is often hindered by batch effects or unwanted variation in data caused by differences in technical factors across batches.

a. Firstly, make a list of all rma files generated from the above step 9.

```
>eset_rma <- list(GSE69235_rma, GSE19844_rma, GSE3341_rma, GSE34192_rma, GSE43050_rma,
GSE36272_rma, GSE49242_rma, GSE61832_rma, GSE61833_rma, GSE53940_rma, and GSE34192_rma)
```

b. Merged the file without and with the CombBat method.

```
>merged_rma_none <- merge(eset_rma, method = "NONE")

>merged_rma_COMBAT <- merge(eset_rma, method = "COMBAT")

>colnames(pData(merged_rma_none))

>table(pData(merged_rma_none)[,"sample"])

>table(pData(merged_rma_none)[,"Target"])

>table(pData(merged_rma_none)[,"Experiment"])
```

c. To calculate the expression of the data set merged with ComBat method.

```
>merged_rma_COMBAT_exprs <- exprs(merged_rma_COMBAT)

>write.table(merged_rma_COMBAT_exprs, "merged_COMBAT_exprs")
```

⚠ CRITICAL: z-score transformation method can be used to remove the batch effect instead of combat, which is also part of inSilicoMerging packages.

d. Visualization of the dataset without merging with specific methods (without batch effect removed) and combined with Combat methods.
   i. Interpret the data with the multidimensional scaling method (Figures 2A and 2B).

```
>png("MDS Plot_none.png")

>plotMDS(merged_rma_none, colLabel="Target", symLabel="Experiment", main = "No Trans
formation")

>dev.off()

>png("MDS Plot_COMBAT.png")

>plotMDS(merged_rma_COMBAT, colLabel="Target", symLabel="Experiment", main = "COMBAT Trans
formation")

>dev.off()
```

   ii. Relative log expression plot without or with the ComBat method (Figures 2C and 2D).

```
#No Transformation

>png("No Transformation.png", width=1000, height=400)
```

```
>plotRLE(merged_rma_none, colLabel="Target")

>dev.off()

#Combat Transformation

>png("COMBAT Transformation.png", width=1000, height=400)

>plotRLE(merged_rma_COMBAT, colLabel="Target")

>dev.off()
```

iii. Histogram distribution of data before and after batch effect removal using that Combat method (Figures 2E and 2F).

```
>png("Histogram Before_Batch_Effect_Remove.png")

>hist(merged_rma_none, col=brewer.pal(9, "Set1"), main = "Histogram Before Batch Effect Remove")

>dev.off()

>png("Histogram After_Batch_Effect_Remove.png")

>hist(merged_rma_COMBAT, col=brewer.pal(9, "Set1"), main = "Histogram After Batch Effect Remove")

>dev.off()
```

11. Filtration of the low-intensity reads from the dataset matrix and saving the remaining data to the external file.
    a. Calculate the raw median intensity of the data.

```
>library(limma)

>merged_median <- rowMedians(exprs(merged_rma_COMBAT))

>write.table(merged_median, file="merged_median")
```

b. Generate the histogram of the median intensity of the expressed genes.

```
>hist_res <- hist(merged_median, 100, col = "white", freq = FALSE, main = "Histogram of the Median intensities", border = "brown", xlab = "Median intensities")

>threshold <- 4

>png("merged_median.png")

>plot(merged_median)

>dev.off()
```

c. Set the cut-off threshold straight line.

```
>png(" Cutoff.png")

>hist_res <- hist(merged_median, 100, col= "white", freq = FALSE, main = "Histogram of the Median intensities", border = "brown", xlab = "Median intensities")

>abline(v = threshold, col = "coral4", lwd = 2)

>dev.off()
```

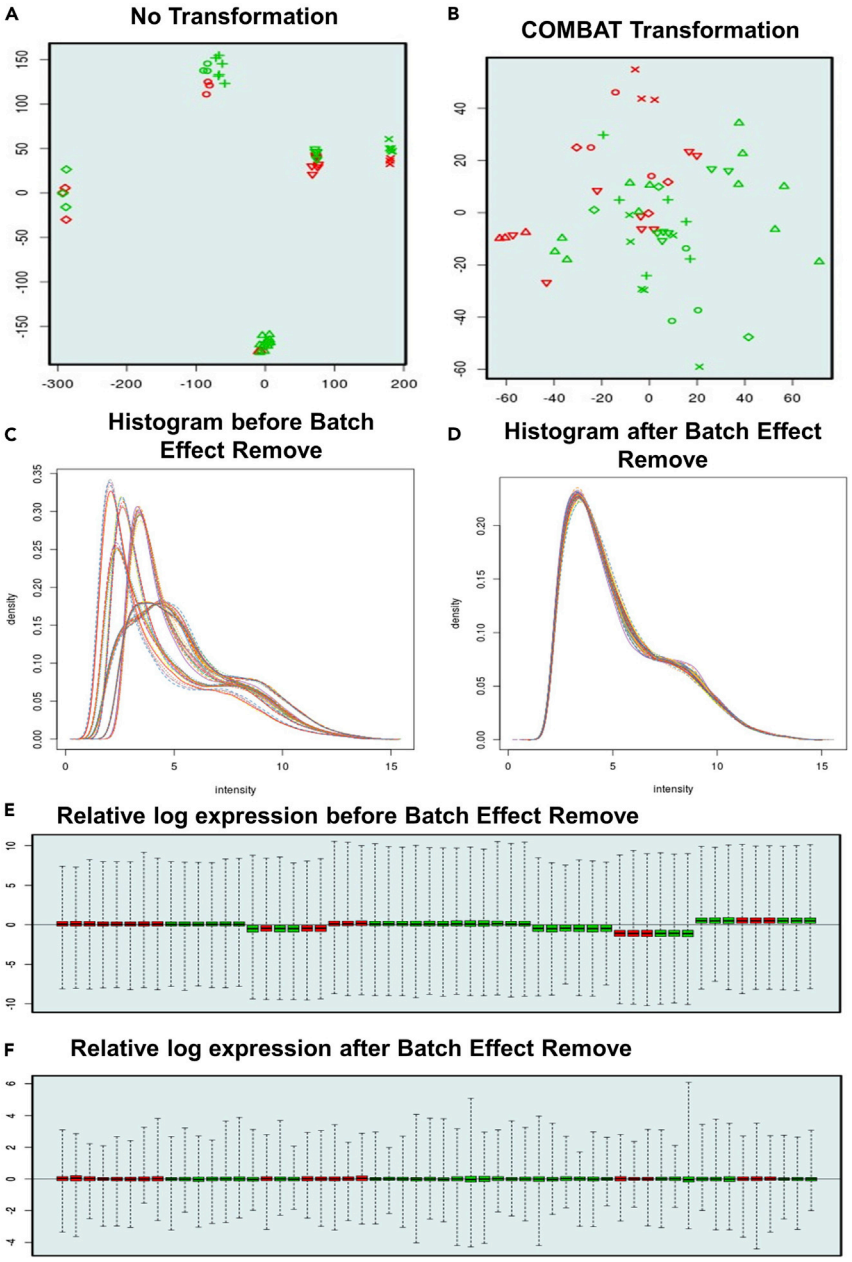


Figure 2. Graphical representation of the merging of the different series data

(A) Multidimensional scale (MDS) plot of two series of data merging without (without removing the batch effect) any specific technique. Huge inter-study biases can be observed, hindering further analysis of this combined data set.

(B) MDS plot showing the data of two series merging with batch effect remove using comBat method. The Combat transformation samples are more clustered based on the target biological variable of interest (red: control, green: treated with the pathogen) instead of the study they originate from. The scale in the combat transformation is also reduced as the method tries to move the samples toward the central.

(C and D) Data distribution in the histogram without and with the batch effect remove.

(E and F) Relative log expression distribution of the normalized data of two series before batch effect removal and after batch effect removal.

Pause point: Cut-off threshold may vary and depend on the quality dataset. The limma package of R gives the median intensity plot of the expressed genes.

d. Visualize the intensity plot based on median intensity and density, set the threshold and extract the genes with median intensity greater than a threshold value.

```
>no_of_samples <- table(paste0(pData(merged_rma_COMBAT)>Target))

samples_cutoff <- min(no_of_samples)

>idx_threshold <- apply(merged_rma_COMBAT, 1, function(x){sum(x >threshold) >= samples_
cutoff})

>table(idx_threshold)

>write.table(idx_threshold, "idx_threshold")

>COMBAT_filtered<- subset(merged_rma_COMBAT, idx_threshold)

>COMBAT_filtered_exprs <- exprs(COMBAT_filtered)

>write.table(exprs(COMBAT_filtered), file="COMBAT_filtered")
```

12. Replace the Affymetrix ids with the gene or locus ids as interaction data is available in the Gene of locus ids.

Note: The biomaRt,[8,9] a Bioconductor package, can annotate the wide range of gene product identifiers with comprehensive gene symbol and Gene ontology information. It extracts the Rice Annotation Project (RAP) ids from the Affymetrix probe ids.

```
>mart <- useEnsembl("plants_mart", "osativa_eg_gene",host = "plants.ensembl.org")

>annot<-getBM(c("affy_rice","ensembl_gene_id","external_gene_name"),"affy_rice",fea-
tureNames(COMBAT_filtered), mart)

>annot2 <- data.frame(PROBEID = featureNames(COMBAT_filtered), annot[match(featureNames(-
COMBAT_filtered), annot[,1]),])

>write.table(annot2, "biomart_annot2")
```

13. Now exit the R environment by pasting the below command in the R terminal environment.

```
>quit()
```

14. Open the terminal in a new window or tab and paste the below command to extract the RAP ids corresponding to Affymetrix probe ids. The Affymetrix ids having no corresponding RAP ids are discarded.

```
> awk '{print$0}' biomart_annot2_eti | awk '{print$2"\t"$4}' | sed 's/"//g' | sed '/NA/d' | awk
'!a[$2]++' > probe_rap_raw
```

15. Identification of the Genome Annotation project (MSU) locus ids corresponding to the RAP ids.

```
>awk '{print$0}' RAP-MSU_2019-03-22.txt | sed '/None/d' | sed '/none/d' | sed 's/\./\t/g' | awk '{print$1"\t"$2}' > RAP-MSU_2022-09-22

> awk 'NR==FNR{a[$2]=$1; next} $1 in a' probe_rap_raw RAP-MSU_2022-09-22 | awk '{rec[NR]=$0; key[NR]=$2; cnt[$2]++} END{for (i=1; i<=NR; i++) if (cnt[key[i]] == 1) print rec[i]}' > RAP_MSU

> awk 'NR==FNR{a[$1]=$2; next} $2 in a {print$0, a[$2]}' RAP_MSU probe_rap_raw > probe_rap_msu_ids
```

Note: The data MSU ids corresponding to the RAP id were downloaded from www.ricechip.org on 2022-09-22.

16. Replace the Affymetrix ids with MSU locus ids in the original data file.

```
>awk '{print$3}' probe_rap_msu_eti_ids > msu_ids

>awk 'BEGIN {for (i=1; i<= 17989; i++) print i}' > number_total
```

Pause point: Here total number of ids in the MSU_ids (Rice Genome Annotation Project) files is 17989. The number may vary with other datasets.

```
>awk '{print$1"\t"$3}' probe_rap_msu_ids > probe_msu_ids && paste number_total probe_msu_eti_ids > number_probe_msu_ids

>sed 's/"//g' COMBAT_filtered >COMBAT_filtered_without_quotes

>awk 'FNR==NR{a[$1]; next}; ($1 in a)' probe_rap_msu_eti_ids COMBAT_filtered_without_quotes > probe_intensity_values

>paste msu_ids probe_intensity_values | sed 's/\t/ /g' > probe_locus_intensity_values

>paste msu_ids probe_locus_intensity_values | awk '!($2="")' | cut -d " " -f 1,3- > locus_intensity_values

>awk '{$1=""; print $0}' locus_intensity_values | sed 's/^ //g' > Input_Data_File
```

Note: 'Input_Data_File' (https://gitlab.com/ravisaroch/star-protocols/-/blob/main/RAW_FILTERED_DATA/expression_with_gene_and_samples_names) is the file containing the MSU_ids and their intensity in the different samples.

17. Prepare the files as input for the deep learning algorithms (DLNet) to identify the potential gene target.
    a. Transpose the file matrix 'Input_Data_File' as algorithms use the file as sample name should be in a row and feature name in columns. Use the below code to transpose the file:

```
>awk -F, '
>{
>  for (i=1; i<=NF; i++) {
>    a[NR,i] = >i
>  }
>}
>NF>p { p = NF }
```

```
>END {
>  for(j=1; j<=p; j++) {
>    str=a[1,j]
>    for(i=2; i<=NR; i++){
>      str=str" "a[i,j];
>    }
>    print str
>  }
>}' Input_Data_File >Raw_file
```

b. Add the last column as a class of the data (0 and 1).
c. At last, remove the first row (feature or locus ids) and column (sample name) from the file and save in comma-separated format. Please visit the given link for more detail (https://gitlab.com/ravisaroch/star-protocols/-/blob/main/expression).

⚠ CRITICAL: The number of features should be the same in the expression and interaction data files.

18. Now we have three files require for the implementation of the deep learning algorithm:
    a. Genes: Containing a list of the genes or locus_ids.
    b. Interaction: File containing the pair of the interaction of the above genes.
    c. Expression: File 'expression' is a comma-delimited file with p features and n samples containing the expression value of each feature in individual samples and the last column containing the class of the individual samples (0 or 1).

Implementation of the DLNet (deep learning algorithm)

Timing: ~2–3 h (for step 19)

In this section, we go over each step of implementing DLNet algorithm in filtered data, tuning the algorithm parameters to obtain the best classification model, and identification of the best host target genes in response to biotic stress.

19. Apply the DLNet algorithms on the filtered data files.

Note: The study's primary goal is to identify the potential target gene and analyze the network architecture. Downloaded repository in step 1, containing the above three files (expression, genes, and interaction), and for script required to execute the below steps to calculate the feature importance score with 10-fold (Number of the fold change may be different as per requirement) cross-validation. Markov cluster method (MCL) is used for discerning quality functional modules from biological networks.[1,6]

a. Open the terminal, change the directory to the folder where the above repository is downloaded, and paste the above three files explained in steps 17 and 18.
b. Now paste the below command into the current directory (troubleshooting 4).

```
>chmod 755 dlnet.sh
>./dlnet.sh expression RF Genes
```

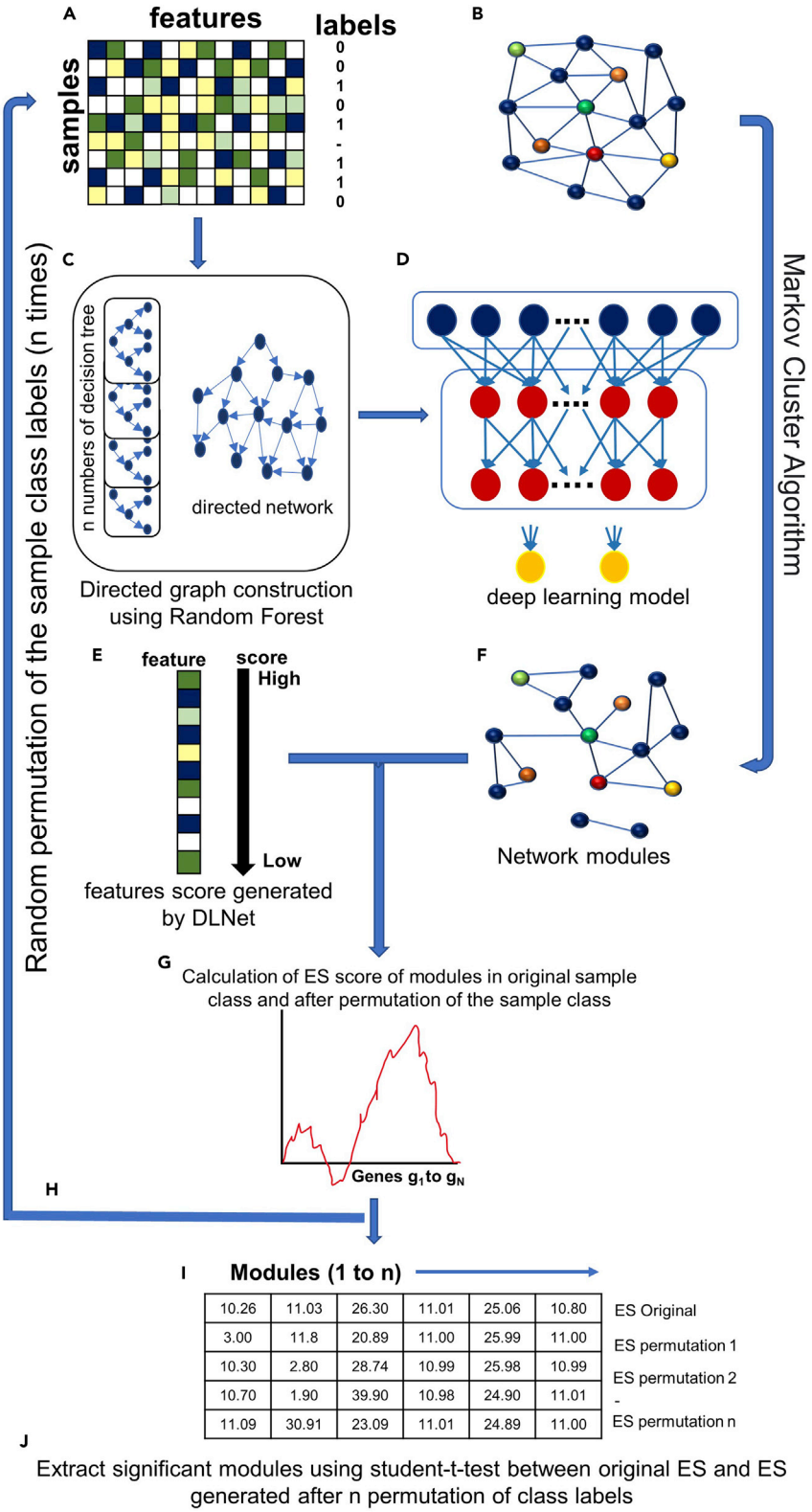

A
features
labels
samples
0
0
1
0
1
-
1
1
0
B
C
n numbers of decision tree
directed network
Directed graph construction
using Random Forest
D
deep learning model
Markov Cluster Algorithm
Random permutation of the sample class labels (n times)
E
feature
score
High
Low
features score generated
by DLNet
F
Network modules
G
Calculation of ES score of modules in original sample
class and after permutation of the sample class
Genes g1 to gN
H
I
Modules (1 to n)
10.26 11.03 26.30 11.01 25.06 10.80 ES Original
3.00 11.8 20.89 11.00 25.99 11.00 ES permutation 1
10.30 2.80 28.74 10.99 25.98 10.99 ES permutation 2
10.70 1.90 39.90 10.98 24.90 11.01 -
11.09 30.91 23.09 11.01 24.89 11.00 ES permutation n
J
Extract significant modules using student-t-test between original ES and ES
generated after n permutation of class labels

Figure 3. The complete workflow of the DLNet to construct the host genes immune related network in response to pathogen

(A and B) (A) The pre-analyzed mRNA expression profile and (B) filtered protein-protein interaction and gene co-expression data are input in the DLNet algorithm.

(C) The algorithm's feature extraction part selects the most useful *p* features from the expression profile and then builds the directed feature graph with the help of the Random Forest. This algorithm-generated graph differs significantly from the previously created filtered network, including protein-protein interaction and gene co-expression profiles.

(D) The directed features graph is used as input by the algorithm's deep learning part. The part consists of hidden layers corresponding to hidden neurons. The hidden neuron in the first layer is the same as the features in the directed features graph. The other three hidden layers consist of 64, 16, and 2 hidden neurons, respectively. The algorithm created each feature's gene score in accordance with its participation in the data classification.

(E) All the genes present in the network are arranged in descending order based on their score in the data classification. The running sums were computed for all genes starting from top to bottom ranked.

(F) The Markov Cluster (MCL) algorithm is used to construct the modular network from the filtered network (B).

(G) The ranked genes present in the module increase the module's enrichment score (ES); when a gene is absent, the ES for that module falls, and the running sum generates the final ES. Similarly, the ES is generated for the "n" number of modules.

(H) The phenotypic class labels (0, 1) were permutated randomly to calculate the background ES for all modules.

(I) Repeat all the steps (A to G) and generate the background ES for n number of modules.

(J) Perform a student-t-test between the original and background-generated ES and extract the significant modules based on *p value* < 0.05.

Where dlnet.sh is the repository shell script, expression is the expression profile file, RF is the random forest method, and Genes is the list of features in the same order as in the expression profile. The detailed protocol is described in Figure 3 and our previous study Kumar et al.[1]

Pause point: Several parameters are used in the script and may change per requirement. Parameters for hyperparameters and model training are supplied via a Python script (dlnetRank.py). The cross-validation parameters and background enrichment scores for significant module extraction can be found in the (dlnet.sh) script (troubleshooting 5).

c. After running the above command, the information is shown in the Ubuntu terminal to enter the name interaction file.

```
>Please Enter your interaction data file:
```

d. Now enter the filename, hit enter, and wait for the analysis to complete.

```
>Please Enter your interaction data file: Interaction
```

20. After implementation of the DLNet algorithm, the following files are generated: Avg_feature_Imp (Table S2), AUC_FPR_TPR (Table S3), MCL_modules (Table S4), Significant_modules (Table S5), and ROC.png (Figure 4).
21. The total module constructed using the mcl algorithm is then subjected to a statistical student's t-test. Deep learning-based enrichment analysis is carried out to get functional annotations for each module using plantGSEA.[15] It will be then transferred to Cytoscape or any other program for network visualization (troubleshooting 6).

EXPECTED OUTCOMES

The output of this protocol provides the number of potential target genes that participate in data classification or can be network hub genes and play an essential role in disease-resistant crop development. The protocol created the figures (Figures 1 and 2); based on these figures, the data sample may be filtered and used as input for the deep learning algorithms. Following the implementation of the DL algorithm, the following tables were generated: 1) Avg_feature_Imp (Table S2), the table comprises a list of genes and their scores; the high score of the genes indicates that these genes participate in the data

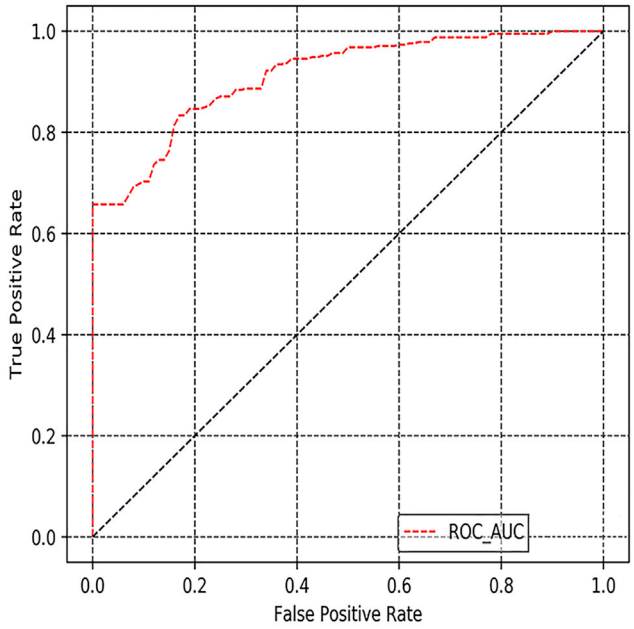


Figure 4. The area under the ROC curve plot of the DLNet model for classification of the data (Treatment vs Control)

classification (Treatment/Control) and may become potential target genes. 2) AUC_FPR_TPR (Table S3) represents the true positive rate (TPR) to the false positive rate (FPR). 3) MCL modules (Table S4) provide a list of genes in each module, their Gene Ontology, and detailed descriptions. The manually filtered significant modules (Table S5) and list of genes that participate in the significant modules (Sig_Module_Gene) files are required as input for gene network visualization using various network visualization tools and analysis of network hub genes that may be employed for next-generation disease-resistance crops. Figure 4 represents the TPR for the FPR, describing the model's accuracy.

QUANTIFICATION AND STATISTICAL ANALYSIS

The "before you begin" and "step-by-step method details" sections explain the operating system, software, and packages used in the present protocol. The AUROC measures how well a model can distinguish between distinct classes. Its statistics AUROC plot the true positive rate (TPR) against the false positive rate (FPR).

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{TN + FP}$$

Permutation, 100 null hypothesis models and the student t-test were used to build the network modules.

LIMITATIONS

The present protocol employs a deep learning algorithm, DLNet, that considers microarray data as input, and the data-driven from the microarray has less sensitive than existing RNA-seq technologies. Although, co-expression studies utilizing microarrays and RNA-seq produce a commensurate gene co-expression network.[18,20] This protocol overcomes the problem of having fewer samples than features since past deep learning methods needed a substantial number of samples.[1] Still, in the current protocol, at least 25–30 samples will be necessary to conduct the investigation. This requirement is suited not only for model species but also for other non-model species. So, additional

steps (such as homology gene search) may be necessary to accomplish the downstream analysis in other species.

TROUBLESHOOTING

It is important to examine the critical statements at the end of some steps in following this protocol. It is also essential to check the software version and package usage for the protocol implementation.

Problem 1

Package 'BiocManager' is not available for the R version in step 4a "before you begin".

Potential solution

One reason is that the R version is too old, and BiocManager is incompatible with the same. Please check the R version and upgrade if needed to avoid this problem. The BiocManager works with R versions 3.5 and higher.

Problem 2

The package is not available in Bioconductor or for the current R version in step 4c "before you begin".

Potential solution

Some packages are unavailable in the Bioconductor version and must be installed manually using devtools. The command needed to install the packages using devtools is:

```
>sudo R

>library(devtools)

>install_github("https://github.com/theislab/destiny", build_vignettes = FALSE, dependencies = TRUE)
```

Problem 3

When using the simpleaffy package to read the file, you may see the following error:

```
>In the function file(file, "rt"): unable to access the file './covdesc,' there is no such file or directory in step 9
```

Potential solution

The readaffy package needs the covdesc file to read the phenotypic information of the data. In step 9, we explain the correct syntax to use the covdesc file. Below is an example of the covdesc file used in the above step.

```
Target Experiment

GSM1695752.CEL Control GSE69235

GSM1695753.CEL Control GSE69235

GSM1695754.CEL Control GSE69235

GSM1695755.CEL Control GSE69235

GSM1695756.CEL Control GSE69235

GSM1695757.CEL Control GSE69235

GSM1695758.CEL Control GSE69235

GSM1695759.CEL Control GSE69235
```

```
GSM1695760.CEL Treatment GSE69235
GSM1695761.CEL Treatment GSE69235
GSM1695762.CEL Treatment GSE69235
GSM1695763.CEL Treatment GSE69235
GSM1695764.CEL Treatment GSE69235
GSM1695765.CEL Treatment GSE69235
```

Problem 4

When running the deep learning algorithm 'dlnet.sh' you may get the 'ImportError or ModuleNotFoundError for TensorFlow' in step 19b.

Potential solution

This error is because you may have failed to install TensorFlow correctly or downloaded it in the wrong python environment. The possible solution is to install the Tensorflow using the below commands.

```
>python -m pip install tensorflow
```

The better solution is to install Tensorflow using a conda environment using.

```
>conda create -name tensorflow-env python=3.8 pip
>conda activate tensorflow-env
>pip install tensorflow
```

Note: You may change the python version accordingly.

Problem 5

The accuracy of the model classification is significant low in step 19b.

This may be due to the hyperparameter used in the model classification.

Potential solution

Change the parameters accordingly in the hyperparameters section of the "dlnet_Rank.py" python script of the repository, and re-run the model for classification accuracy and feature selection.

Problem 6

The pipeline takes an excessive amount of time to finish the operation in step 21.

Potential solution

Although pipelines take around half day to accomplish the process in 256 GB RAM and 40 processors machine with 10-fold cross-validation and crucial network module selection with 100 Enrichment background score, the statistical test is performed. The end user can adjust the fold change value and also reduce the number of background enrichment scores generated for the statistical test in the pipeline's "dlnet.sh" shell script of the repository.

RESOURCE AVAILABILITY

Lead contact

Further information and resource requests should be directed to the lead contact, Vishal Acharya (vishal@ihbt.res.in).

Materials availability

This study did not generate new unique reagents or materials.

Data and code availability

The raw data under stress conditions and their related controls utilized in the protocol were made accessible to the public and are included in Table S1. In addition, a link to the datasets resources is provided in the key resources table.

The protocol's source code and raw data are accessible on GitLab at https://gitlab.com/ravisaroch/star-protocols.

The supplemental information are accessible on Mendeley Data at https://data.mendeley.com/datasets/4yh3d8xmbg.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2022.101934.

ACKNOWLEDGMENTS

The work was carried out under The Himalayan Center for High-throughput Computational Biology (HiCHiCoB), a BIC supported by the Department of Biotechnology, India, and CSIR, India (MLP-201 & MLP-0178). This manuscript represents CSIR-IHBT communication number 5183.

AUTHOR CONTRIBUTIONS

R.K. and V.A. designed and carried out the experiments, analyzed the data, wrote the manuscript, and approved the final manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Kumar, R., Khatri, A., and Acharya, V. (2022). Deep learning uncovers distinct behavior of rice network to pathogens response. iScience *25*, 104546. https://doi.org/10.1016/j.isci.2022.104546.

2. Gautier, L., Cope, L., Bolstad, B.M., and Irizarry, R.A. (2004). Affy—analysis of Affymetrix GeneChip data at the probe level. Bioinformatics *20*, 307–315. https://doi.org/10.1093/bioinformatics/btg405.

3. Miller, C. (2018). Simpleaffy: very simple high-level analysis of Affymetrix data version 2.66.0 from Bioconductor. https://rdrr.io/bioc/simpleaffy.

4. Brettschneider, J., Collin, F., Bolstad, B.M., and Speed, T.P. (2012). Quality assessment for short oligonucleotide microarray data. Technometrics *50*, 241–264. https://doi.org/10.1198/004017008000000334.

5. Barrett, T., Wilhite, S.E., Ledoux, P., Evangelista, C., Kim, I.F., Tomashevsky, M., Marshall, K.A., Phillippy, K.H., Sherman, P.M., Holko, M., et al. (2013). NCBI GEO: archive for functional genomics data sets - update. Nucleic Acids Res. *41*, D991–D995. https://doi.org/10.1093/nar/gks1193.

6. Enright, A.J., Van Dongen, S., and Ouzounis, C.A. (2002). An efficient algorithm for large-scale detection of protein families. Nucleic Acids Res. *30*, 1575–1584. https://doi.org/10.1093/nar/30.7.1575.

7. Taminau, J., Meganck, S., Lazar, C., Steenhoff, D., Coletta, A., Molter, C., Duque, R., de Schaetzen, V., Weiss Solís, D.Y., Bersini, H., and Nowé, A. (2012). Unlocking the potential of publicly available microarray data using inSilicoDb and inSilicoMerging R/Bioconductor packages. BMC Bioinf. *13*, 335–339. https://doi.org/10.1186/1471-2105-13-335.

8. Durinck, S., Spellman, P.T., Birney, E., and Huber, W. (2009). Mapping identifiers for the integration of genomic datasets with the R/Bioconductor package biomaRt. Nat. Protoc. *4*, 1184–1191. https://doi.org/10.1038/nprot.2009.97.

9. Durinck, S., Moreau, Y., Kasprzyk, A., Davis, S., De Moor, B., Brazma, A., and Huber, W. (2005). BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. Bioinformatics *21*, 3439–3440. https://doi.org/10.1093/bioinformatics/bti525.

10. Oughtred, R., Stark, C., Breitkreutz, B.-J., Rust, J., Boucher, L., Chang, C., Kolas, N., O'Donnell, L., Leung, G., McAdam, R., et al. (2019). The BioGRID interaction database: 2019 update. Nucleic Acids Res. *47*, D529–D541. https://doi.org/10.1093/nar/gky1079.

11. Gu, H., Zhu, P., Jiao, Y., Meng, Y., and Chen, M. (2011). PRIN: a predicted rice interactome network. BMC Bioinf. *12*, 161–213. https://doi.org/10.1186/1471-2105-12-161.

12. Krishnan, A., Gupta, C., Ambavaram, M.M.R., and Pereira, A. (2017). RECoN: rice environment co-expression network for systems level analysis of abiotic-stress response. Front. Plant Sci. *8*, 1640–1715. https://doi.org/10.3389/fpls.2017.01640.

13. Patil, A., and Nakamura, H. (2005). Filtering high-throughput protein-protein interaction data using a combination of genomic features. BMC Bioinf. *6*, 100–113. https://doi.org/10.1186/1471-2105-6-100.

14. Tian, T., Liu, Y., Yan, H., You, Q., Yi, X., Du, Z., Xu, W., and Su, Z. (2017). agriGO v2.0: a GO analysis toolkit for the agricultural community, 2017 update. Nucleic Acids Res. *45*, W122–W129. https://doi.org/10.1093/nar/gkx382.

15. Yi, X., Du, Z., and Su, Z. (2013). PlantGSEA: a gene set enrichment analysis toolkit for plant community. Nucleic Acids Res. *41*, W98–W103. https://doi.org/10.1093/nar/gkt281.

16. Davis, S., and Meltzer, P.S. (2007). GEOquery: a bridge between the gene expression omnibus (GEO) and BioConductor. Bioinformatics *23*, 1846–1847. https://doi.org/10.1093/bioinformatics/btm254.

17. Gandolfo, L.C., and Speed, T.P. (2018). RLE plots: visualizing unwanted variation in high dimensional data. PLoS One *13*, e0191629. https://doi.org/10.1371/journal.pone.0191629.

18. Zogopoulos, V.L., Malatras, A., and Michalopoulos, I. (2022). Gene co-expression analysis in Arabidopsis thaliana based on public microarray data. STAR Protoc. *3*, 101208. https://doi.org/10.1016/j.xpro.2022.101208.

19. Johnson, W.E., Li, C., and Rabinovic, A. (2007). Adjusting batch effects in microarray expression data using empirical Bayes methods. Biostatistics *8*, 118–127. https://doi.org/10.1093/biostatistics/kxj037.

20. Obayashi, T., Aoki, Y., Tadaka, S., Kagaya, Y., and Kinoshita, K. (2018). ATTED-II in 2018: a plant coexpression database based on investigation of the statistical property of the mutual rank index. Plant Cell Physiol. *59*, e3. https://doi.org/10.1093/pcp/pcx191.