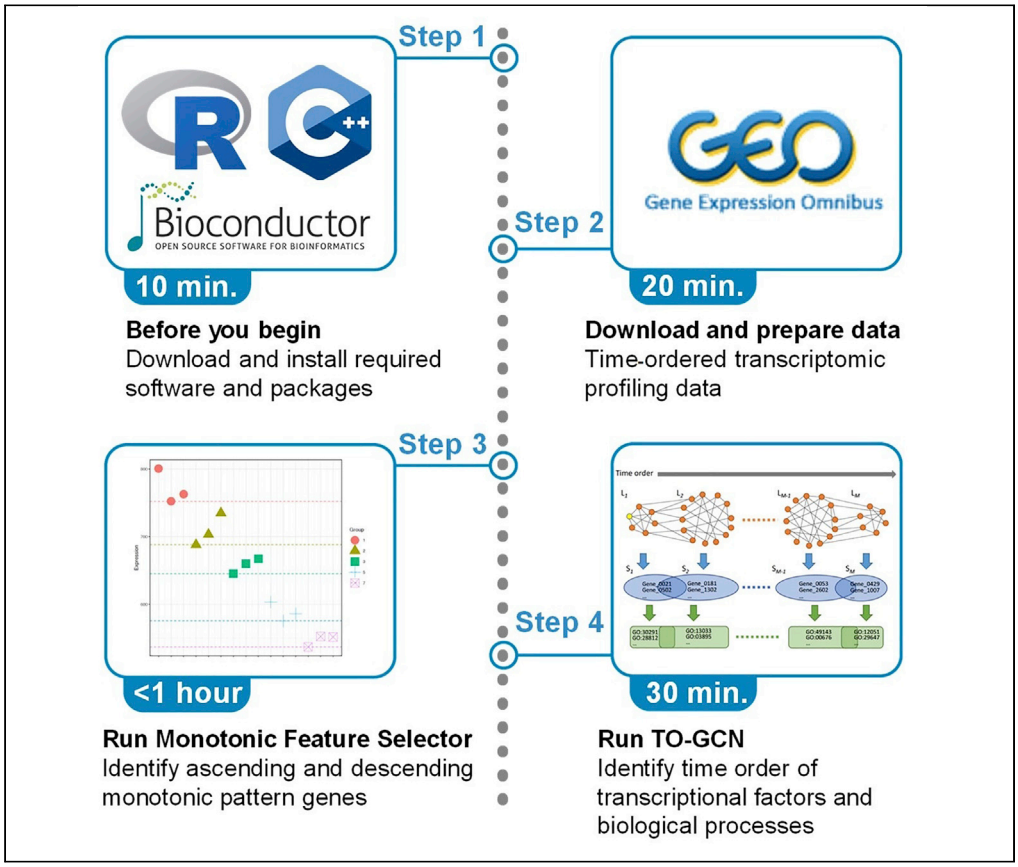
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Generating transcriptional regulatory networks from time-ordered stem cell differentiation RNA sequencing data



We describe steps to 1) identify ascending and descending monotonic key genes from time-ordered stem cell differentiation expression data, 2) construct time-ordered transcriptional regulatory networks, and 3) infer the involvement of transcription factors along the differentiation process.

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

Yu-Shuen Tsai, Yao-Ming Chang, Yang-Mooi Lim, Soon-Keng Cheong, I-Fang Chung, Chee-Yin Wong

ifchung@nycu.edu.tw (I.-F.C.)
wongcy@1utar.my (C.-Y.W.)

Highlights

A protocol to identify ascending and descending monotonic pattern genes

Temporal gene regulation in development

Pipeline can be adapted to other time-series data sets

Tsai et al., STAR Protocols 3, 101541
September 16, 2022
https://doi.org/10.1016/j.xpro.2022.101541

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Generating transcriptional regulatory networks from time-ordered stem cell differentiation RNA sequencing data

Yu-Shuen Tsai,[1,6,7] Yao-Ming Chang,[2,6,7] Yang-Mooi Lim,[3] Soon-Keng Cheong,[3] I-Fang Chung,[1,4,5,*] and Chee-Yin Wong[3,8,*]

[1]Center for Systems and Synthetic Biology, National Yang Ming Chiao Tung University, Taipei, Taiwan

[2]Institute of Biomedical Sciences, Academia Sinica, Taipei, Taiwan

[3]Faculty of Medicine and Health Sciences, Universiti Tunku Abdul Rahman, Selangor, Malaysia

[4]Institute of Biomedical Informatics, National Yang Ming Chiao Tung University, Taipei, Taiwan

[5]Preventive Medicine Research Center, National Yang Ming Chiao Tung University, Taipei, Taiwan

[6]These authors contributed equally

[7]Technical contact: yushuen.tsai@gmail.com (Y.S.T.); petitming@ibms.sinica.edu.tw (Y.M.C.)

[8]Lead contact

*Correspondence: ifchung@nycu.edu.tw (I.-F.C.), wongcy@1utar.my (C.-Y.W.)
https://doi.org/10.1016/j.xpro.2022.101541

SUMMARY

We describe steps to 1) identify ascending and descending monotonic key genes from time-ordered stem cell differentiation expression data, 2) construct time-ordered transcriptional regulatory networks, and 3) infer the involvement of transcription factors along the differentiation process.

For complete details on the use and execution of this protocol, please refer to Wong et al. (2020).

BEFORE YOU BEGIN

Download R version 4.2.0 and Rtools 4.2

Timing: 10 min

1. As described on the official website (https://www.r-project.org/), "R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS."
   a. To download and install R and Rtools, visit https://www.r-project.org/ (R Core Team, 2020) and follow the instructions. We prepare the protocol with R 4.2.0 and Rtools 4.2.

Note: Rtools are required to build R packages from source R code on Windows. In this protocol, we use the MFSelector package to identify key monotonic genes. Since the precompiled MFSelector package was built with R < 3.6.2, which is incompatible with R 4.2.0, we have to install and build the package from the R source code. It is possible to build an independent computing environment to use the precompiled MFSelector package with Conda (https://docs.conda.io/projects/conda/en/latest/), though we do not recommend to use the old version of R. See troubleshooting: problem 1.

Download the required packages in R

Timing: 10 min

2. Download the required packages (listed in the key resources table) through `BiocManager::install()`.
   a. Get version 3.15 of Bioconductor by starting R and entering the commands.

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

>install.packages("BiocManager")

>BiocManager::install(version = "3.15")
```

   b. Install the required packages with the command in the R console.

```
>BiocManager::install(c("ggplot2", "gplots", "gridExtra",

"RColorBrewer", "rtracklayer", "GEOquery", "biomaRt"))
```

Note: These packages are deposited in two separated repositories: CRAN and Bioconductor. Bioconductor has a repository and release schedule that differs from R. Bioconductor version 3.15 is the release for R version. 4.2.0. Visit http://bioconductor.org/install/ for detailed information.

3. Download and install the MFSelector package through http://microarray.bmi.nycu.edu.tw:8080/tools/module/MFSelector/index.jsp?mode=support (Wang et al., 2015).
   a. Download the R source code to the working directory and install MFSelector with the commands on the R console.

```
>download.file("http://microarray.bmi.nycu.edu.tw:8080/tools/module/MFSelector/content/support/multicore/
MFSelector_1.0.tar.gz", "MFSelector_1.0.tar.gz")

>install.packages("MFSelector_1.0.tar.gz", repos = NULL, type = "source")
```

Note: This website is hosted by the original group that published MFSelector. We recommend installing the package from the R source code, which is also supported for parallel computing with multiple cores. If there is a need to run MFSelector with multiple cores in parallel on a Windows machine, see troubleshooting: problem 2.

Download the TO-GCN package

Timing: 2 min

4. To download TO-GCN software, visit https://github.com/petitmingchang/TO-GCN_STAR-Protocol and download the package from the website.
   a. Once you have downloaded the package in zip format, unzip it and you will get a folder named TO-GCN_STAR-Protocol-main.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
A time-ordered mesenchymal stem cell differentiation RNA sequencing dataset	NCBI Gene Expression Omnibus	NCBI GEO: GSE140914 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE140914)
MFSelector_doSNOW.r	(Tsai, 2022)	https://github.com/yushuen/MFSelector_STAR-Protocol (https://doi.org/10.5281/zenodo.6637018)

(Continued on next page)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Human TF gene list	(Hu et al., 2019)	http://bioinfo.life.hust.edu.cn/static/AnimalTFDB3/download/Homo_sapiens_TF
Software and algorithms		
R software	(R Core Team, 2020)	https://www.r-project.org/
TO-GCN package	(Chang et al., 2019; Chang, 2022)	https://github.com/petitmingchang/TO-GCN_STAR-Protocol
MFselector package	(Wang et al., 2015)	http://microarray.bmi.nycu.edu.tw:8080/tools/module/MFSelector/index.jsp?mode=home
ggplot2 package	(Wickham, 2016)	https://cran.r-project.org/package=ggplot2
gplots package	(Warnes et al., 2016)	https://cran.r-project.org/package=gplots
RColorBrewer package	(Neuwirth, 2014)	https://cran.r-project.org/package=RColorBrewer
parallel package	(R Core Team, 2020)	https://cran.r-project.org/web/views/HighPerformanceComputing.html
foreach package	(Microsoft and Weston, 2020)	https://cran.r-project.org/package=foreach
doSNOW package	(Microsoft Corporation and Weston, 2020)	https://cran.r-project.org/package=doSNOW
GEOquery package	(Davis and Meltzer, 2007)	https://bioconductor.org/packages/release/bioc/html/GEOquery.html
biomaRt package	(Durinck et al., 2005, 2009)	https://bioconductor.org/packages/release/bioc/html/biomaRt.html

STEP-BY-STEP METHOD DETAILS

Download and prepare the data matrix for analysis

Timing: 1 min

Stem cell differentiation is determined by the underlying gene regulatory network during the process of development, which leads the stem cells into their particular terminal cell phenotype. During stem cell differentiation, the stemness biomarkers of stem cells will descend over time while the characteristics and functions of terminal cells will ascend. The proposed protocol is designed for time-ordered transcriptomic profiling data. First, we use the MFSelector package to identify ascending and descending monotonic key genes. Then, we use the TO-GCN software to identify the time order of transcriptional factors and biological processes. Based on the official MFSelector tutorial, it accepts data in a tab-delimited text file with the first column containing gene identifiers. It is first developed for gene expression microarray data, but it is also compatible for RNA sequencing data. The expression values can be either normalized read counts (i.e., counts per million, CPM) or other quantitative measures, such as TPM (transcripts per million), and FPKM (fragments per kilobase of transcript per million fragments mapped). TO-GCN recommends to use normalized quantitative measures, e.g., TPM but also accepts normalized read counts. In this protocol, we use an RNA sequencing dataset from our previous study (Wong et al., 2020) as an example. It is the dataset of mesenchymal stem cells differentiated into mesangial cells. This dataset has been deposited in the Gene Expression Omnibus (GEO) database: GSE140914 (https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE140914). This dataset has been normalized with TMM (Trimmed Mean of the M-values) and and processed CPM values after batch effect removal. These processed data are analysis-ready. Only a simple process is required to combine the data of all samples.

1. Download the processed data (GSE140914_RAW.tar) from GEO with GEOquery package.
   a. Once the file is downloaded, use the command to extract the file.

```
>library(GEOquery)

>getGEOSuppFiles("GSE140914", makeDirectory = F)

>untar("GSE140914_RAW.tar")

>list.files(pattern = "gz$") |> lapply(gunzip)
```

b. Import and combine the data with R.

```
# Import data
>data_list <- list.files(pattern="^GSM") |> lapply(read.delim, header = TRUE)
# Combine data
>data_mat <- sapply(data_list, \(x) x[, 3])
```

Note: We import the data of 15 samples from 15 decompressed files. Each file contains three columns, which are Ensembl gene identifiers, HGNC gene symbols, and normalized read counts. The genes of all files are in the same order. We can combine these data without additional mapping efforts.

c. Process the sample names.

```
# Get sample names, batch information, group information
>sample_names <- sapply(data_list, \(x) colnames(x)[3])
>sample_names <- gsub("X", "", sample_names)
>batch_id <- gsub("\\.\\S+", "", sample_names)
>sample_group <- gsub("\\S+\\.", "", sample_names)
>sample_code <- paste0("B:", batch_id, "_D:", sample_group)
```

Note: The third column name is the sample name which is named as "batch (integer)"-"days after differentiation started (integer)". When importing file into R, sample names will automatically be converted, for example, '1-1' will become "X1.1". Therefore, we include a few steps in our example code to deal with this.

d. Process the gene identifiers.

```
>gene_id <- data_list[[1]][, 1]
>gene_names <- data_list[[1]][, 2]
>gene_labels <- paste(gene_id, gene_names, sep = ":")
```

Note: We use these commands to create a merged gene identifier for each gene which is composed of the Ensembl gene identifier and the HGNC gene symbol. We use the merged gene identifiers in next step to obtain readable outputs of MFSelector.

e. Generate the input data for MFSelector.

```
# Assign column names and row names
>colnames(data_mat) <- sample_code
>rownames(data_mat) <- gene_id
# Generate the input data for MFSelector
>data <- cbind(gene_labels, as.data.frame(data_mat))
```

f. Visualize the overall relationship between the samples with a multidimensional scaling (MDS) plot with R (Figure 1).

```
# Define a function to generate MDS plot
>get_MDS_plot <- function(prefix, data_mat, sample_group){
  cmd <- t(data_mat) |> dist() |> cmdscale()
  df <- data.frame(Sample = colnames(data_mat), Dimension_1 = cmd[, 1],
  Dimension_2 = cmd[, 2], Group = sample_group)
  p1 <- ggplot(df, aes(Dimension_1, Dimension_2)) +
  geom_point(aes(colour = Group), size = 4) +
  scale_color_brewer(palette="Paired") +
  ggtitle(paste(prefix, "MDS Plot"))
  print(p1)
}
# Load required package
>library(ggplot2)
# Generate a MDS plot for this dataset -> Figure 1
>get_MDS_plot("GSE140914", data_mat, sample_group)
>ggsave("GSE140914_MDSplot.tiff", dpi = "retina"
```

Identification of ascending and descending monotonic key genes by Monotonic Feature Selector

Timing: 30 min

MFSelector is an R package which is designed to identify genes either with increasing or decreasing expressions across different time ordered stages, i.e., ascending and descending monotonic key genes. It is suitable for the transcriptomic profiling of cell differentiation investigations. The options for using MFSelector can be referred to the MFSelector Tutorial shown in the official site: http://microarray.bmi.nycu.edu.tw:8080/tools/module/MFSelector/index.jsp?mode=support. The key options for different datasets are the number of data points in each stage, the name represented for each stage, and the type of monotonic genes. We develop the protocol with the multicore version of MFSelector.

2. Prepare the inputs with R.
   a. Load required package.

```
>library(MFSelector)
>library(parallel)
```

   b. Prepare MFSelector key input arguments.

```
>nsc <- table(sample_group)
>stageord <- order(sample_group)
>stagename <- sample_group |> unique() |> sort()
```

c. Detect the number of total cores in the current machine.

```
>total_cores <- detectCores()

>half_cores <- total_cores/2
```

3. Identify candidate descending monotonic key genes with MFSelector. The outputs are a table of candidate genes saved in a tab-delimited text file and the scatter plots of all candidate genes saved in a PDF file.

```
>mfselector(data, nsc, stageord = stageord, stagename = stagename,

type = 1, nline = T, dline = T, pdf = 1:100, cmp = 0, permut = 100, svdenoise = 0.03, svdetimes = 4,
cores = half_cores)
```

4. Rename the output files.

```
>mf_outputs <- list.files(pattern="^mfselector")

>mf_outputs_type1 <- gsub(".*\\.", "MFSelector_Type1.", mf_outputs)

>mapply(file.rename, mf_outputs, mf_outputs_type1)

>rm(mf_outputs, mf_outputs_type1)
```

5. Identify candidate ascending monotonic key genes.

```
>mfselector(data, nsc, stageord = stageord, stagename = stagename,

type = 2, nline = T, dline = T, pdf = 1:100, cmp = 0, permut = 100,

svdenoise = 0.03, svdetimes = 4, cores = half_cores)
```

6. Rename the second output files.

```
>mf_outputs <- list.files(pattern="^mfselector")

>mf_outputs_type2 <- gsub(".*\\.", "MFSelector_Type2.", mf_outputs)

>mapply(file.rename, mf_outputs, mf_outputs_type2)

>rm(mf_outputs, mf_outputs_type2)
```

Note: For an alternative solution for running parallel MFSelector on a Windows machine, see troubleshooting: problem 3.

Export high quality plots (complying with most publication standards)

Timing: 4 min

7. Define an R function to identify genes that have fulfilled the given criteria.

```
>candidate_genes_parser <- function(input_file, DE = NULL, SVDE = NULL, nline = NULL, out_col = NULL, ...){
    input_tab <- read.delim(input_file)
    rii <- 1:nrow(input_tab)
    if(!is.null(DE)){
      sii <- which(as.numeric(input_tab[, "DE"]) <= DE)
      rii <- intersect(rii, sii)
    }
    if(!is.null(SVDE)){
      sii <- which(as.numeric(input_tab[, "SVDE"]) <= SVDE)
      rii <- intersect(rii, sii)
    }
    if(!is.null(nline)){
      sii <- which(input_tab[, "with.N.1.distinct.lines"] == TRUE)
      rii <- intersect(rii, sii)
    }
    if(is.null(out_col)){
      candidate_genes <- as.character(input_tab[rii, 1])
    }else{
      candidate_genes <- as.character(input_tab[rii, out_col])
    }
    return(candidate_genes)
}
```

8. Get candidate monotonic key genes.

```
>MF_type1_genes <- candidate_genes_parser("MFSelector_Type1.txt", DE = 4)
>MF_type2_genes <- candidate_genes_parser("MFSelector_Type2.txt", DE = 4)
```

9. Find the indices of candidate genes.

```
>mii_type1 <- match(MF_type1_genes, gene_labels)
>mii_type2 <- match(MF_type2_genes, gene_labels)
```

10. Generate MDS plots with these candidate genes (e.g., Figure 2).

```
>get_MDS_plot("GSE140914_Type1", data_mat[mii_type1, ], sample_group)
>ggsave("GSE140914_Type1_MDSplot.tiff", dpi = "retina") # Figure 2
>get_MDS_plot("GSE140914_Type2", data_mat[mii_type2, ], sample_group)
>get_MDS_plot("GSE140914_Type1+2", data_mat[c(mii_type1, mii_type2), ], sample_group)
```

11. Generate heatmap plots with these candidate genes.
    a. Define a function for the Z-score transformation.

```
>z_transformation <- function(x){(x-mean(x))/sd(x)}
```

    b. Define a function to perform hierarchical clustering with the Ward's method.

```
>ward_hclust <- function(d){hclust(d, method = "ward.D")}
```

    c. Define a function to generate a heatmap.

```
>get_heatmap<-function(prefix, data, sample_group, gene_symbols = NULL){
    sample_col_fac <- as.factor(sample_group)
    n_group <- sample_group |> unique() |> length()
    dataZ <- apply(data, 1, z_transformation) |> t()
    my_col_palette <- brewer.pal(11, "RdBu") |> rev()
    levels(sample_col_fac) <- brewer.pal(n_group, "Paired")
    col_colors <- as.character(sample_col_fac)
    if(!is.null(gene_symbols)){
         rownames(dataZ) <- gene_symbols
    }
    heatmap.2(dataZ, col = my_col_palette, ColSideColors = col_colors,
      Colv = TRUE, dendrogram = "both", hclustfun = ward_hclust,
      margins = c(5, 7), trace = "none", keysize = 1.2,
      lhei = c(1.5, 9.5), lwid = c(1.5, 6.5))
}
```

    d. Load the required packages for generating heatmap.

```
>library(gplots)
>library(RColorBrewer)
```

    e. Generate a heatmap plot with type 1 (descending) monotonic candidate genes (Figure 3).

```
>tiff("GSE140914_Type1_Heatmap.tiff", width=10, height=12, unit="in", res = 320)
>get_heatmap("GSE140914_Type1", data_mat[mii_type1, ], sample_group,
gene_symbols = gene_labels[mii_type1])
>dev.off() # Figure 3
```

Note: The default graphics device is 7 inches square, which is not large enough for a heatmap with a long gene list. Therefore, we create a larger TIFF device to save the heatmap.

12. Generate scatter plots with these candidate genes.
    a. Define a function to generate the scatter plot with ggplot2.

```
>get_scatter_plots <- function(my_gene_id, data, gene_id, sample_group, mf_tab){
    mii1 <- which(gene_id == my_gene_id)
```

```
    mii2 <- which(mf_tab[, 1] == my_gene_id)
    if(length(mii1) > 0 & length(mii2) >0){
      DE <- mf_tab[mii2, 2]
      PVAL <- round(mf_tab[mii2, 3], 2)
      QVAL <- round(mf_tab[mii2, 4], 2)
      SVDE <- mf_tab[mii2, 5]
      TITLE <- paste(my_gene_id, "\nDE =", DE, " p-value =", PVAL,
           " q-value =", QVAL, " SVDE =", SVDE)
      expression <- as.numeric(data[mii1, ])
      ordered_idx <- order(sample_group)
      df <- data.frame(Samples = 1:length(expression),
          Expression = expression[ordered_idx],
          Group = as.factor(sample_group[ordered_idx]))
      group_min_df <- data.frame(
      Min = tapply(df$Expression, df$Group, min),
      Group = levels(df$Group))
      p <- ggplot(data = df, aes(x = Samples, y = Expression)) +
        geom_point(aes(color = Group, shape = Group),
          size = 4) + scale_color_brewer(palette="Paired") +
        scale_x_continuous(breaks = 1:length(expression),
          labels = colnames(data)[ordered_idx]) +
        geom_hline(aes(yintercept = Min, color = Group),
        group_min_df, lty = "dashed") + ggtitle(TITLE) +
        theme(plot.title = element_text(hjust = 0.5)) +
        theme_bw() + theme(axis.text.x = element_text(angle = 45,
          vjust = 1, hjust = 1))
      print(p)
    }
}
```

b. Import the output text files of MFSelector.

```
>mf_tab_1 <- read.delim("MFSelector_Type1.txt", header = TRUE)
>mf_tab_2 <- read.delim("MFSelector_Type2.txt", header = TRUE)
```

c. Generate scatter plots with these candidate genes (Figure 4 as an example). If ''parallel'' package is available in your system, use ''mclapply()'' instead of ''lapply()''.

```
>get_scatter_plots(MF_type1_genes[1], data_mat, gene_labels, sample_group, mf_tab_1)
>ggsave("Figure_4.tiff", dpi = "retina") # Figure 4
```

```
>pdf("Scatter-Plots_Type1.pdf")
>lapply(MF_type1_genes, get_scatter_plots, data_mat, gene_labels, sample_group, mf_tab_1)
>dev.off()
>pdf("Scatter-Plots_Type2.pdf")
>lapply(MF_type2_genes, get_scatter_plots, data_mat, gene_labels, sample_group, mf_tab_2)
>dev.off()
```

Prepare the input data for TO-GCN

Timing: 5 s

13. Prepare the input files for TO-GCN.
    a. Compute the mean CPM values for each group.

```
>sample_group_fac <- as.factor(sample_group)
>get_group_mean <- function(x, sample_group_fac){
  tapply(x, sample_group_fac, mean)
}
>group_mean_cpm <- apply(data_mat, 1, get_group_mean, sample_group_fac) |> t()
```

    b. Define a function to set negative values as zero.

```
>neg_to_zero <- function(x){ (abs(x)+x)/2 }
```

    c. Filter out lowly expressed genes (CPM <= 1).

```
>check_mat <- neg_to_zero(group_mean_cpm - 1)
>mode(check_mat) <- "logical"
>check_ans <- apply(check_mat, 1, any)
>group_mean_cpm_subset <- group_mean_cpm[check_ans, ]
```

    d. Import Ensembl gene biotype annotation. See problem 4 for details.

```
>gene_biotype <- read.delim("gene_biotype.txt", header = T)
```

    e. Check if there are gene identifiers without Ensembl gene biotype annotations.

```
>bii <- match(rownames(group_mean_cpm_subset), gene_biotype$ensembl_gene_id, nomatch = 0)
>nii <- which(bii!=0)
```

    f. Select only protein coding genes.

```
>feature_type <- gene_biotype[bii[nii], 2]
>pii <- which(feature_type == "protein_coding")
>group_mean_cpm_pro <- group_mean_cpm_subset[nii[pii], ]
```

g. Download and import full list of human transcript factors from AnimalTFDB3.0.

```
>download.file("http://bioinfo.life.hust.edu.cn/static/AnimalTFDB3/download/Homo_sapiens_TF",
"Homo_sapiens_TF.txt")

>human_tf <- read.delim("Homo_sapiens_TF.txt", header = T)
```

h. Identify the indices of genes annotated as transcription factors.

```
>tii <- match(rownames(group_mean_cpm_pro), human_tf$Ensembl, nomatch = 0)

>tf_vec <- as.logical(tii)
```

i. Split the mean TPM table into two tables for transcription factor coding genes and non-transcription factor coding genes respectively.

```
>tf_group_mean_cpm <- group_mean_cpm_pro[tf_vec, ]

>non_tf_group_mean_cpm <- group_mean_cpm_pro[!tf_vec, ]
```

j. Generate the input files for TO-GCN.

```
>write.table(tf_group_mean_cpm, "TF_gene_matrix.tsv", row.names = T, col.names = F, sep =
"\t", quote = F)

>write.table(non_tf_group_mean_cpm, "Non-TF_gene_matrix.tsv", row.names = T, col.names =
F, sep = "\t", quote = F)
```

Note: Here, we identify protein coding genes based on Ensembl annotations. An example of how to retrieve this annotation with biomaRt is all provided. See troubleshooting: problem 4.

Run the TO-GCN analysis

Timing: 8 min

Unlike the previous R scripts steps, the TO-GCN analysis is composed of three major parts (by three tools): (1) estimates the cutoff value (by 'Cutoff'), (2) constructs the time-ordered gene co-expression network (by "TO-GCN"), and (3) generates a list of co-expressed gene for each level in the TO-GCN (by 'GeneLevel'). You can directly run the precompiled executable files or create the executable files by C++ compiler for your operating systems. In the first part, the 'Cutoff' tool will give you a range of cutoff suggestions for the following analysis by checking all Pearson's correlation coefficient (PCC) values of expression profiles between all pairs of TF genes. In the second part, the 'TO-GCN' tool will construct a gene co-expression network (GCN) with multiple levels that correspond to the up-regulation time order of the TF genes. To obtain a reasonable and meaningful TO-GCN, you need to choose some (at least one) of the TF genes that may be up-regulated earliest as the initial seed. The candidates for the initial seed can be found in the list of descending monotonic key genes (`MF_type1_genes`) generated by the MFSelector. The output of 'TO-GCN' can be visualized by the Cytoscape (https://cytoscape.org). In the last part, the 'GeneLevel' tool will give you a list of genes that correspond to each time-ordered level. You can perform the GO (Gene Ontology) term or KEGG pathway enrichment test to obtain dynamic biological processes over the time points.

14. Estimate the cutoff value of Pearson correlation coefficient for constructing a gene co-expression network. The input file contains a matrix of TF gene IDs and their expression level (can be TPM, FPKM, or CPM, etc.) at each time point.
    a. Go to the downloaded TO-GCN folder (using Windows as an example).

```
$ cd TO-GCN_STAR-Protocol-main\precompiled_files\Windows
```

b. Prepare input files (either from the output of step 13 or the example_data folder in package).

```
$ copy ..\..\example_data\TF_gene_matrix.tsv .

$ copy ..\..\example_data\Non-TF_gene_matrix.tsv .
```

c. Get the range of cutoff values for reference with TF gene matrix.

```
$ Cutoff.exe 5 TF_gene_matrix.tsv

No. of TFs: 1122

No. of time points: 5

Cutoff value for your reference: 0.86 ~ 0.90
```

15. Construct a time-ordered gene co-expression network (TO-GCN) of TF genes. Two input files are required in this step. The first input file contains the TF gene matrix used in the step 14, while the second input file contains a list of TF gene IDs for initiating the BFS (Breadth-First Search) algorithm to determine the time order for TF genes in the network.
    a. Create a text file of the initial TF gene (use ENSG00000175592 as an example).

```
$ echo ENSG00000175592 > initial_seed.txt
```

b. Construct the network with two input files and the PCC cutoff value.

```
$ TO-GCN.exe 5 TF_gene_matrix.tsv initial_seed.txt 0.9

No. of TFs: 1122

No. of time points: 5

No. of initial TF seeds: 1

Cutoff: 0.90

Assigning levels for nodes in GCN by Breath-First-Search (BFS) method...

Done!
```

Optional: Use Cytoscape to visualize the network with two output files, Node_relation.csv and Node_level.tsv, for the input of edges and nodes respectively.

16. Generate sets of gene lists for each time-ordered level. This step requires three input files, a TF gene matrix, a non-TF gene matrix, and one of the output files (Node_level.tsv) from step 15.

```
$ GeneLevel 5 TF_gene_matrix.tsv Non-TF_gene_matrix.tsv Node_level.tsv 0.9

No. of TFs: 1122

No. of non-TF genes: 11118

No. of time points: 5

Cutoff: 0.90

Done!
```

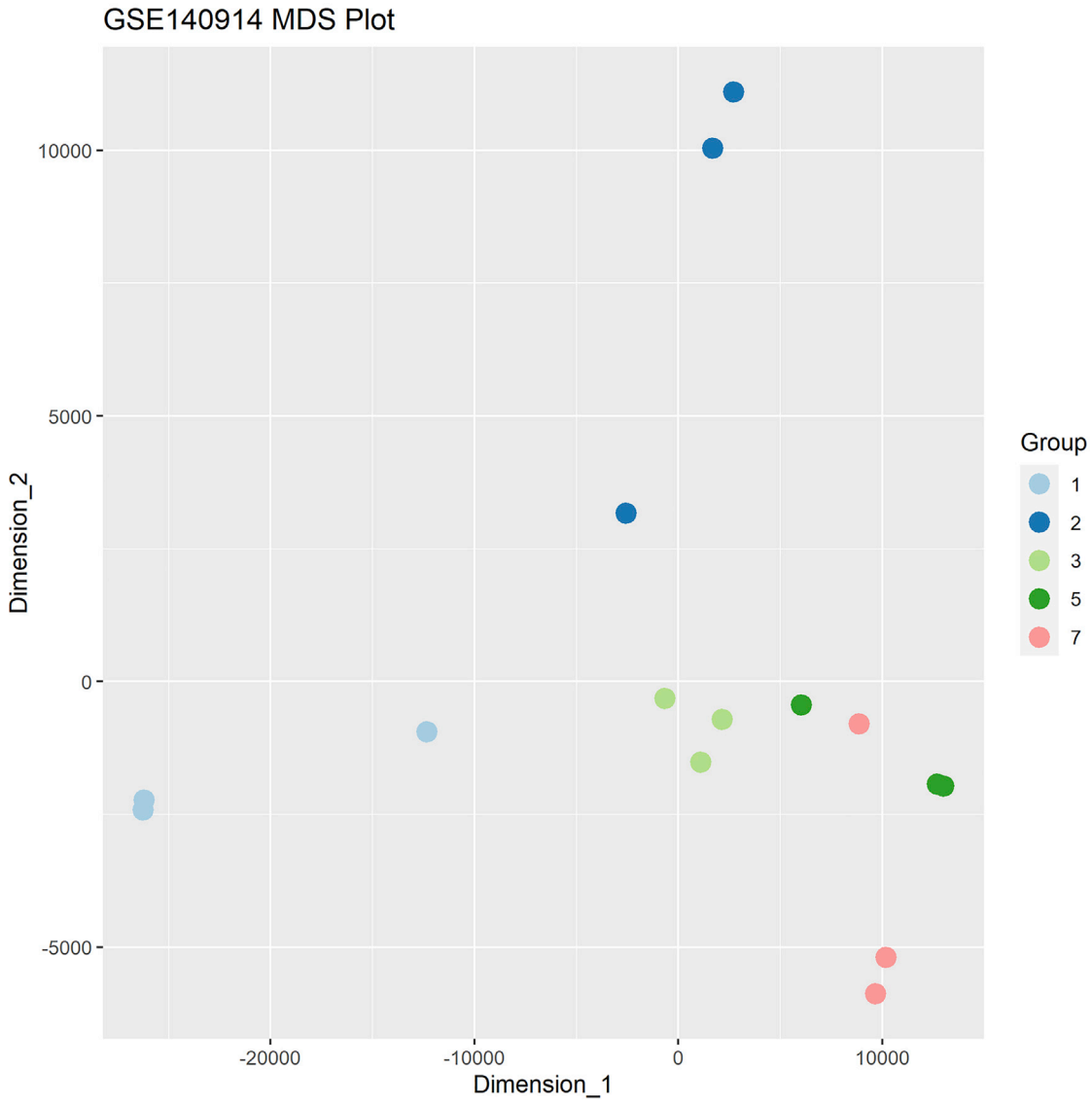


Figure 1. A multidimensional scaling (MDS) plot based on the entire gene expression profile

Optional: Conduct the functional enrichment test with the list of genes in output files. These genes were output in a list (Gene_list_in_each_level.csv) and a matrix (Gene_level_matrix.csv).

Note: Although precompiled executable files were provided for Linux, MacOS, and Windows, you can still use any C++ compiler to generate the executable files with files in the "source_-code" folder, problem 5.

The suggested cutoff value in step 14 is from the top 5% value (p-value < 0.05) in all PCC values of the expression profiles between all pairs of TF genes. It is worth noting that a higher cutoff value will generate more levels in a TO-GCN which means higher resolution of time order level, and vice-versa. However, a very high cutoff value will filter out too many likely co-expression relationships between TF genes and will obtain very few numbers of TF genes for analysis. Therefore, we provide a range of cutoff suggestions to build the TO-GCN.

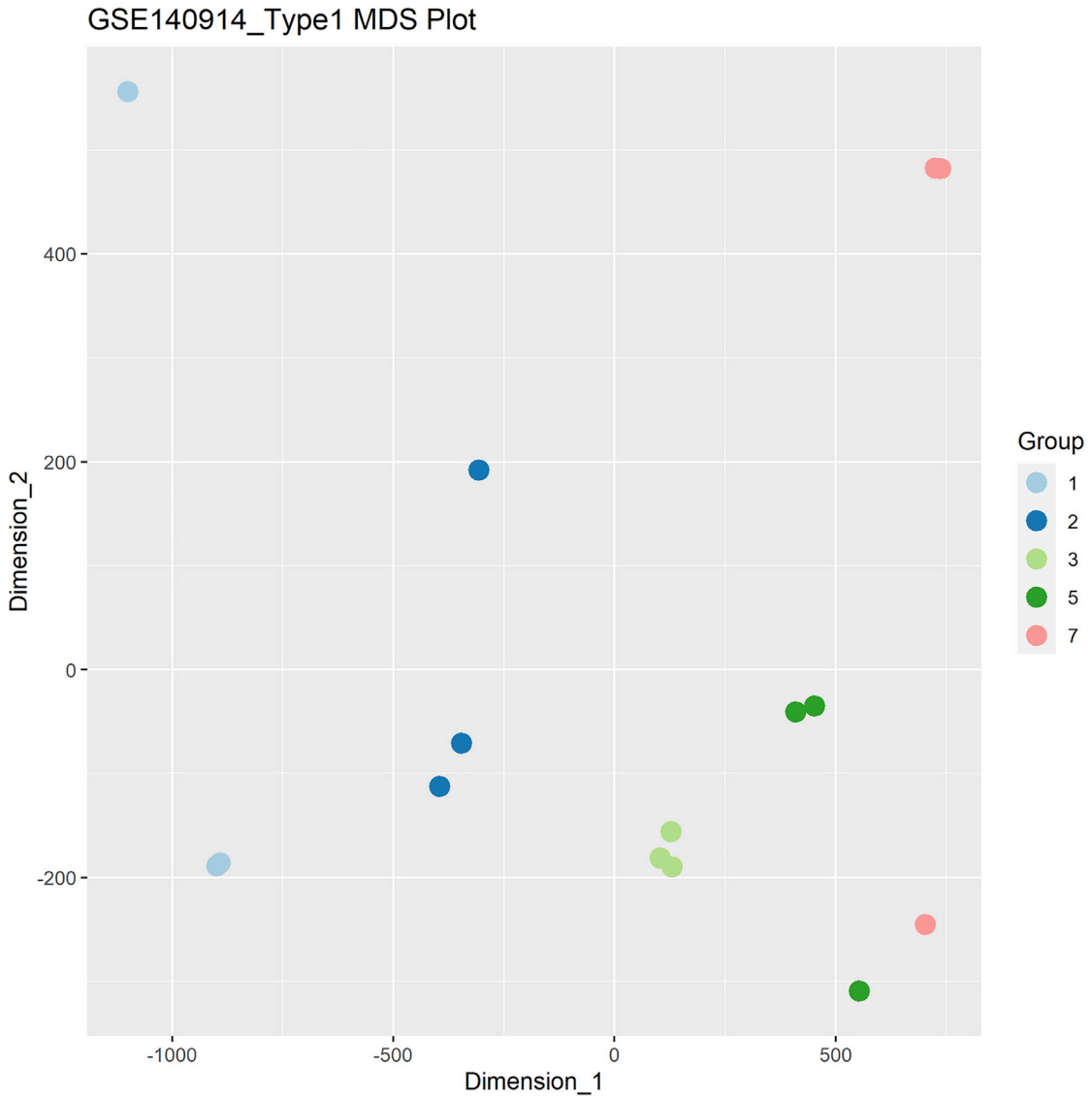


Figure 2. A multidimensional scaling (MDS) plot based on the gene expression profile of candidate descending monotonic key genes

EXPECTED OUTCOMES

MFSelector and TO-GCN were the two methods of data analysis used in this workflow. These methods worked synergistically to provide a deeper understanding of stem cell differentiation into terminal cells. In this article, the differentiation of mesenchymal stem cells into mesangial cells was used as example. MFSelector determined the degree of monotonicity for all genes during the differentiation process and provided an estimation of the expression behavior of the gene during differentiation. TO-GCN used co-expression relationship to connect TF genes as pairs, in which they have similar expression patterns (i.e., significantly high PCC) over time. It inferred expression time orders for all TF genes in the network with the starting TF in the strongest descending pattern identified by MFSelector. By applying this method to time-series experiments, TO-GCN provided the time order information of gene regulations in developmental processes. The data obtained from both methods were further used to identify the TF-key genes at specific time points to the TO-GCN at different levels. This helped to elucidate the network interaction between TF-TF and TF-key genes at each level of TO-GCN.

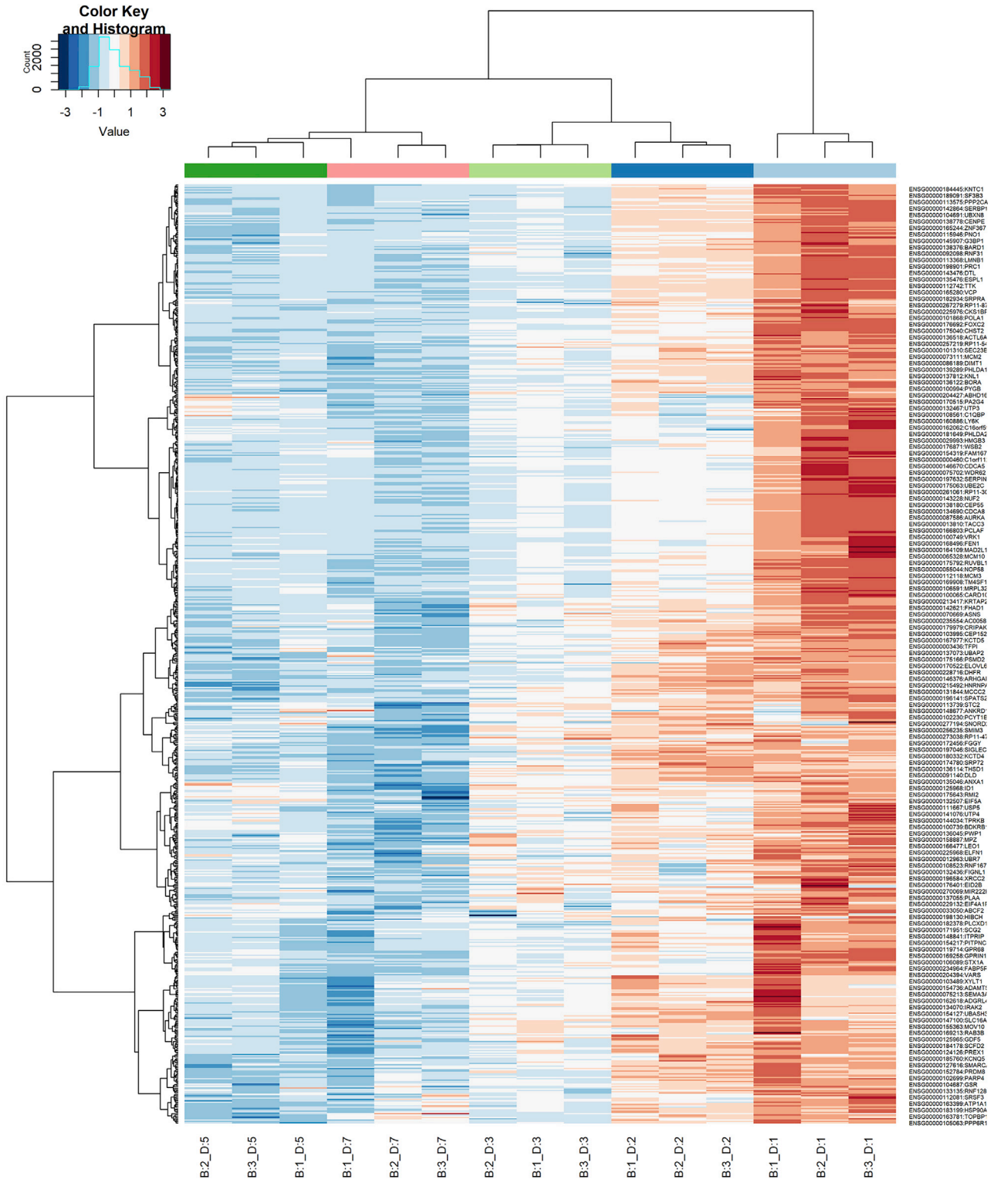


Figure 3. A heatmap based on the gene expression profile of candidate descending monotonic key genes

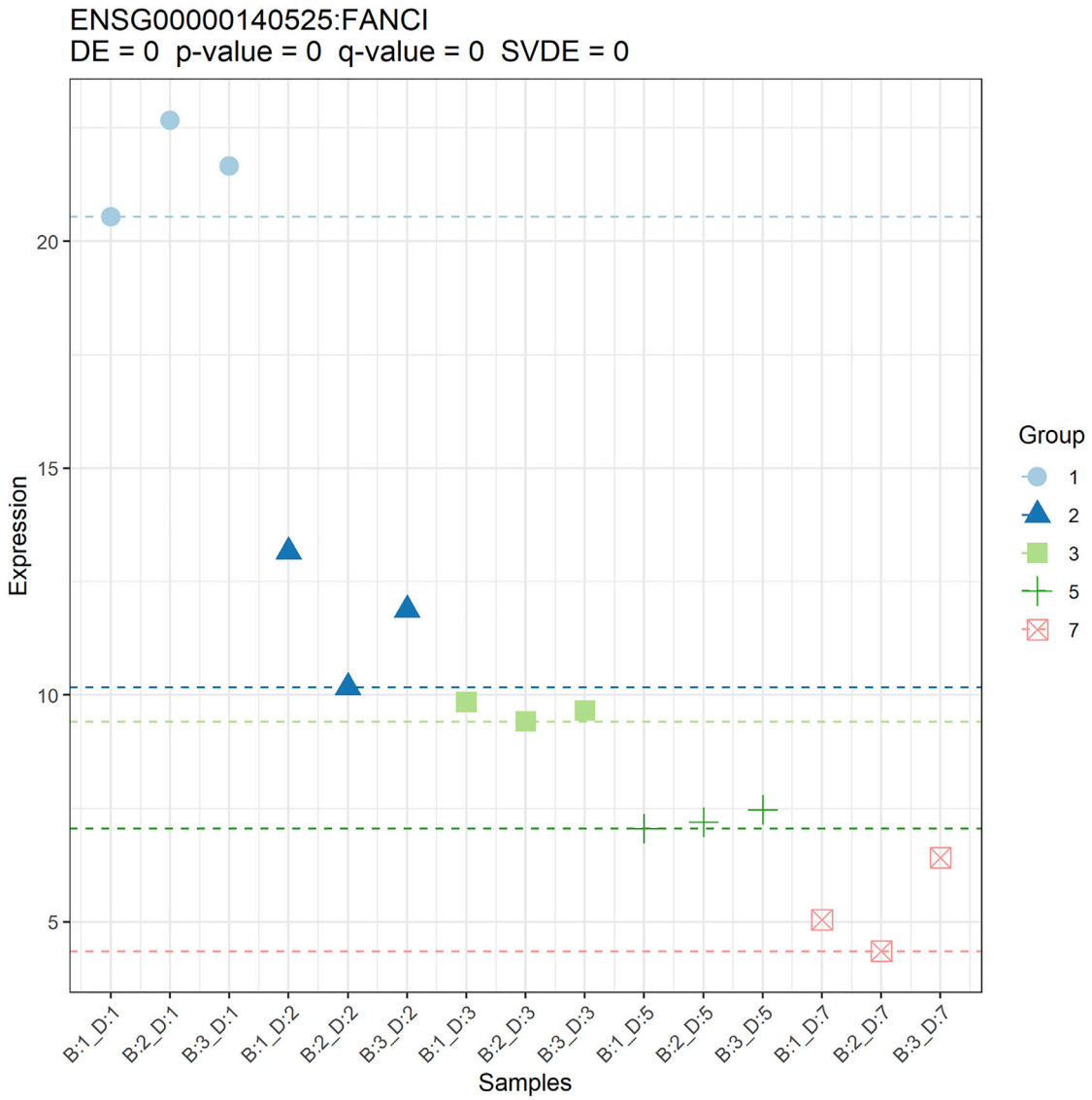


Figure 4. A scatter plot of a selected descending monotonic key gene. DE and SVDE are the discriminating error and the sample variance for the discriminating error respectively. These concepts were proposed and used by the authors of MFSelector. The statistical significance was assessed with a permutation process and the defaulted outputs such as p- and q-values were reported. Visit MFSelector official website for more details.

In Figure 1, we used different colors for different groups and different shapes for samples generated in different batches. The same illustration strategy has been used to generate MDS plots based on each set of candidate monotonic key genes. For example, Figure 2 is based on the gene expression profile of candidate descending monotonic key genes with total discriminating errors less than or equal to four (DE=4). We have also provided a clustering analysis to generate a heatmap with samples labeled in consistent colors as the ones we used in the MDS plot. We used the same set of candidate genes to generate Figure 3 as an example. In addition to the scatter plots generated by MFSelector, we have provided another strategy to create scatter plots in the preferred style (Figure 4). The MFSelector also outputs a table for each run. Users can select candidate genes according to the different criteria listed in the table.

Two major outputs of the TO-GCN package are the time-ordered levels of TF genes in the network and their co-expressed genes for each level. Figure 5 is made with the Cytoscape tool with two

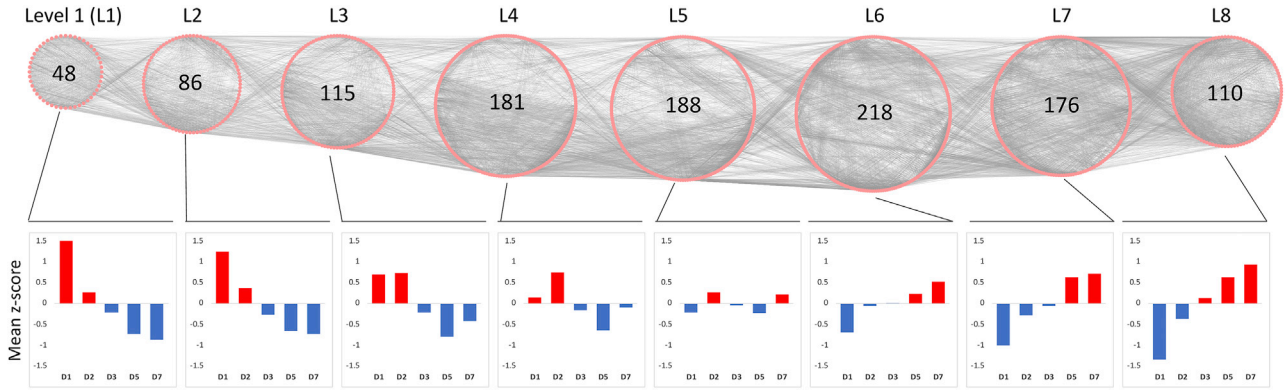


Figure 5. Time-ordered levels of TF genes in the network and their co-expressed genes for each level

Each pink circle represents a TF gene and each gray line between pink circle represents their co-expression relationship (PCC ≥ cut-off value). The number of TF genes for each level is shown in the center. The bar chart below the network demonstrates the up-regulated time order of TF genes at each level. For example, all TF genes at level 1 were up-regulated at D1 (the first time point: Day1) and gradually downregulated at the following time points.

output files, Node_level.tsv (for node information) and Node_relation.csv (for edge information), in step 15. In addition to the network, two output files in step 16, Gene_list_in_each_level.csv and Gene_level_matrix.csv, list all co-expressed genes at each level where the former provides a single list of genes with level IDs and the latter provides a gene ID by level ID matrix. A further functional enrichment test can be conducted with these gene IDs to understand the dynamic biological processes over time-ordered levels.

LIMITATIONS

This protocol relies on a properly defined grouping of samples and time-ordered data. For datasets without well-defined groups, e.g., samples of different groups that are not differentiated from the others physiologically, the MFSelector may not be able to identify monotonic key genes. It should be noted that MFSelector is designed to identify genes with continuously decreased or increased expression levels across time-ordered stages. It fits only a specific scope of research. A minimum of five-time-point time series samples are required to get the meaningful correlation value when applying TO-GCN. Another limitation is that oscillatory time series data are not suitable for TO-GCN because the correlation value may fail to infer the order between genes. We strongly suggest clarifying the research scope before adapting this protocol.

TROUBLESHOOTING

Problem 1

It is necessary to build an independent computing environment to run the precompiled MFSelector.

Potential solution

As described on the official Conda website (https://docs.conda.io/projects/conda/en/latest/), Conda is an open-source package management system and environment management system that runs on Windows, MacOS, and Linux. It can quickly install, run, and update packages and their dependencies. It can create, save, load, and switch between environments on your local computer. To download and install Conda, visit https://docs.conda.io/projects/conda/en/latest/user-guide/install/download.html and follow the instructions. Once Conda is available on your local computer, you can create an independent computing environment for R version 3.6.2 with these commands.

```
$ conda create -n R-3.6.2

$ conda activate R-3.6.2

$ conda install -c conda-forge/label/cf202003 r-base
```

Be sure to activate the environment every time before running this pipeline with the activate command.

```
$ conda activate R-3.6.2
```

Also, be sure to deactivate the environment after running it.

```
$ conda deactivate
```

Problem 2

It is necessary to run MFSelector with multiple cores in parallel on a Windows machine.

Potential solution

Install the required packages with the command in the R console to enable the use of multiple cores in parallel on Windows.

```
>BiocManager::install(c("foreach", "doSNOW"))
```

Next, download the modified MFSelector R script through https://github.com/yushuen/MFSelector_STAR-Protocol. Use the command on the R console to read the R code from the file.

```
>source("MFSelector_doSNOW.r")
```

Then, MFSelector is set to run with parallel processing in the following analysis.

Problem 3

Run MFSelector in parallel on Windows with our solution.

Potential solution

Load the required packages and import MFSelector with the R commands.

```
# Load required package

>library(foreach)

>library(doSNOW)

# To ensure the original MFSelector has been detached

>detach("package:MFSelector", unload=TRUE)

# Read R code of Monotonic Feature Selector

>source("MFSelector_doSNOW.r")

# Run the same commands as what we provided in the main contents
```

Problem 4

It is necessary to identify protein-coding genes in the dataset.

Potential solution

We use the biomaRt package to identify protein coding genes in the dataset. Here are the commands to retrieve the ''gene biotype'' annotation for genes included in the dataset and to generate an output table.

```
# Load package

>library(biomaRt)

# Check which databases are available

>listMarts()

# Select the database and create the ensembl object

>ensembl <- useEnsembl(biomart = 'ENSEMBL_MART_ENSEMBL', host = "uswest.ensembl.org")

# Check which datasets are available

>listDatasets(ensembl)

# Select the dataset and update the ensembl object

>ensembl <- useDataset(dataset = "hsapiens_gene_ensembl", mart = ensembl)

# Check which attributes are available

>listAttributes(ensembl)

# Get gene biotype

>mart_export <- getBM(attributes = c('ensembl_gene_id', 'gene_biotype'), filters = 'ensembl_gene_id', values = gene_id, mart = ensembl)

# Generate an output table

>write.table(mart_export, "gene_biotype.txt", row.names = F, col.names = T, sep = "\t", quote = F)
```

Problem 5

To generate executable files on your own, you need a C++ compiler in your operating system.

Potential solution

Except for Linux, both MacOS and Windows require installing a command-line C++ compiler tool first. On MacOS, you can download the Apple Xcode and install it from Apple Developer website. On Windows, you can download the open-source C++ compiler and follow the installation instruction from the CygWin website.

```
$ cd TO-GCN_STAR-Protocol-main\source_code

$ g++ Cutoff_STAR.cpp -o ..\precompiled_files\Windows\Cutoff

$ g++ TO-GCN_STAR.cpp -o ..\precompiled_files\Windows\ TO-GCN

$ g++ GeneLevel_STAR.cpp -o ..\precompiled_files\Windows\ GeneLevel

$ cd ..\precompiled_files\Windows
```

RESOURCE AVAILABILITY

Lead contact

More information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Chee-Yin Wong, wongcy@1utar.my.

Materials availability

This study did not generate new unique reagents.

Data and code availability

The accession number for the RNA sequencing data used as an example in this paper is NCBI GEO: GSE140914. The Monotonic Feature Selector R function program can be downloaded from http://microarray.bmi.nycu.edu.tw:8080/tools/module/MFSelector/index.jsp?mode=home. The Time-Ordered Gene Co-expression Network analysis program can be downloaded from https://github.com/petitmingchang/TO-GCN_STAR-Protocol. The parallel computing solution of MFSelector for Windows and all R commands used in the protocol can be downloaded from https://github.com/yushuen/MFSelector_STAR-Protocol.

ACKNOWLEDGMENTS

This work was supported by the Malaysia Toray Science Foundation, Malaysia (Grant number: 14/G44) and the Ministry of Science and Technology, Taiwan (Grant number: MOST110-2221-E-A49A-511).

AUTHOR CONTRIBUTIONS

Conceptualization, Y.M.L., S.K.C., I.F.C., and C.Y.W.; Methodology, Y.S.T., Y.M.C., I.F.C., and C.Y.W.; Software, Y.S.T., Y.M.C., and I.F.C.; Formal analysis, Y.S.T. and Y.M.C.; Investigation, Y.S.T., Y.M.C., and I.F.C.; Data Curation, Y.S.T. and Y.M.C.; Writing – Original Draft, Y.S.T. and Y.M.C.; Writing – Review & Editing, Y.S.T., Y.M.C., Y.M.L., S.K.C., I.F.C., and C.Y.W.; Funding Acquisition, Y.M.L., I.F.C., and C.Y.W.; All authors have read and agreed to the published version of the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

Chang, Y.M. (2022). petitmingchang/TO-GCN_STAR-protocol: version 1.0. https://doi.org/10.5281/zenodo.6634755.

Chang, Y.M., Lin, H.H., Liu, W.Y., Yu, C.P., Chen, H.J., Wartini, P.P., Kao, Y.Y., Wu, Y.H., Lin, J.J., Lu, M.Y.J., et al. (2019). Comparative transcriptomics method to infer gene coexpression networks and its applications to maize and rice leaf transcriptomes. Proc. Natl. Acad. Sci. USA *116*, 3091–3099. https://doi.org/10.1073/pnas.1817621116.

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.

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.

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.

Hu, H., Miao, Y.R., Jia, L.H., Yu, Q.Y., Zhang, Q., and Guo, A.Y. (2019). AnimalTFDB 3.0: a comprehensive resource for annotation and prediction of animal transcription factors. Nucleic Acids Res. *47*, D33–D38. https://doi.org/10.1093/nar/gky822.

Microsoft, and Weston, S. (2020). foreach: provides Foreach looping construct. R package version 1.5.1. https://cran.r-project.org/package=foreach.

Microsoft Corporation, and Weston, S. (2020). doSNOW: Foreach parallel adaptor for the 'snow' package. R package version 1.0.19. https://CRAN.R-project.org/package=doSNOW.

Neuwirth, E. (2014). RColorBrewer: ColorBrewer palettes. R package version 1.1-2. https://CRAN.R-project.org/package=RColorBrewer.

R Core Team (2020). R: A Language and Environment for Statistical Computing (R Foundation for Statistical Computing). https://www.R-project.org/.

Tsai, Y.S. (2022). yushuen/MFSelector_STAR-protocol: v1.0.0. https://doi.org/10.5281/zenodo.6637018.

Wang, H.W., Sun, H.J., Chang, T.Y., Lo, H.H., Cheng, W.C., Tseng, G.C., Lin, C.T., Chang, S.J., Pal, N.R., and Chung, I.F. (2015). Discovering monotonic stemness marker genes from time-series stem cell microarray data. BMC Genom. *16*, S2. https://doi.org/10.1186/1471-2164-16-S2-S2.

Warnes, G.R., Bolker, B., Bonebakker, L., Gentleman, R., Liaw, W.H.A., Lumley, T., Maechler, M., Magnusson, A., Moeller, S., Schwartz, M., and Venables, B. (2016). Gplots: various R programming tools for plotting data. R package version 3.1.1. https://CRAN.R-project.org/package=gplots.

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis (Springer-Verlag)978-3-319-24277-4. https://ggplot2.tidyverse.org.

Wong, C.Y., Chang, Y.M., Tsai, Y.S., Ng, W.V., Cheong, S.K., Chang, T.Y., Chung, I.F., and Lim, Y.M. (2020). Decoding the differentiation of mesenchymal stem cells into mesangial cells at the transcriptomic level. BMC Genom. *21*, 467. https://doi.org/10.1186/s12864-020-06868-5.