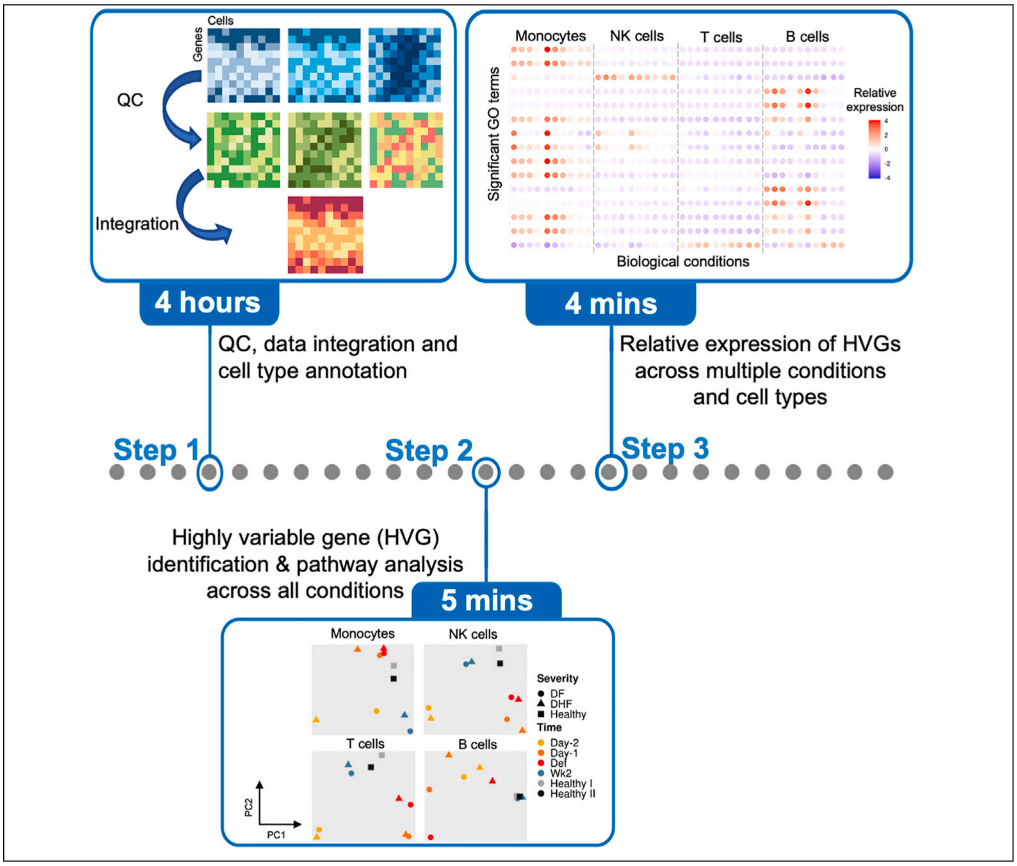
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Computational workflow for investigating highly variable genes in single-cell RNA-seq across multiple time points and cell types



Here, we present a computational approach for investigating highly variable genes (HVGs) associated with biological pathways of interest, across multiple time points and cell types in single-cell RNA-sequencing (scRNA-seq) data. Using public dengue virus and COVID-19 datasets, we describe steps for using the framework to characterize the dynamic expression levels of HVGs related to common and cell-type-specific biological pathways over multiple immune cell types.

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

Jantarika Kumar Arora, Anunya Opasawatchai, Sarah A. Teichmann, Ponpan Matangkasombut, Varodom Charoensawan

st9@sanger.ac.uk (S.A.T.)
ponpan.mat@mahidol.edu (P.M.)
varodom.cha@mahidol.ac.th (V.C.)

Highlights

Highly variable genes (HVGs) as an alternative to DEGs for time-series scRNA-seq data

Visualizing dynamic expression patterns of HVGs over multiple cell types in one figure

Exploring biological pathways with common and cell-type-specific expression dynamics

Arora et al., STAR Protocols 4, 102387
September 15, 2023 © 2023 The Authors.
https://doi.org/10.1016/j.xpro.2023.102387

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Computational workflow for investigating highly variable genes in single-cell RNA-seq across multiple time points and cell types

Jantarika Kumar Arora,[1,2,9] Anunya Opasawatchai,[3,4,7,9] Sarah A. Teichmann,[5,*] Ponpan Matangkasombut,[6,7,*] and Varodom Charoensawan[2,4,7,8,10,*]

[1]Doctor of Philosophy Program in Biochemistry (International Program), Faculty of Science, Mahidol University, Bangkok 10400, Thailand

[2]Department of Biochemistry, Faculty of Science, Mahidol University, Bangkok 10400, Thailand

[3]Department of Oral Microbiology, Faculty of Dentistry, Mahidol University, Bangkok 10400, Thailand

[4]Integrative Computational BioScience (ICBS) Center, Mahidol University, Nakhon Pathom 73170, Thailand

[5]Wellcome Sanger Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SA, UK

[6]Department of Microbiology, Faculty of Science, Mahidol University, Bangkok 10400, Thailand

[7]Systems Biology of Diseases Research Unit, Faculty of Science Mahidol University, Bangkok 10400, Thailand

[8]School of Chemistry, Institute of Science, Suranaree University of Technology, Nakhon Ratchasima 30000, Thailand

[9]Technical contact: jantarikaarora@gmail.com; anunya.opa@mahidol.edu

[10]Lead contact

*Correspondence: st9@sanger.ac.uk (S.A.T.), ponpan.mat@mahidol.edu (P.M.), varodom.cha@mahidol.ac.th (V.C.)

https://doi.org/10.1016/j.xpro.2023.102387

SUMMARY

Here, we present a computational approach for investigating highly variable genes (HVGs) associated with biological pathways of interest, across multiple time points and cell types in single-cell RNA-sequencing (scRNA-seq) data. Using public dengue virus and COVID-19 datasets, we describe steps for using the framework to characterize the dynamic expression levels of HVGs related to common and cell-type-specific biological pathways over multiple immune cell types.

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

BEFORE YOU BEGIN

Overview

Investigation of differentially expressed genes (DEGs) between two biological conditions or paired samples (e.g., control vs.' treatment, or healthy vs.' disease samples) is one of the key steps in genome-wide gene expression analyses, such as transcriptomic and proteomic studies. With single-cell RNA-sequencing (scRNA-seq) technologies, we are now able to look into transcriptomic profiles of individual cells, and hence perform DEG analyses on different cell types and suppopulations. Alternatively, one can also characterize a set of genes that are differentially expressed between cell types, frequently referred to as marker genes. However, most current computational methods were intended for the comparison between two conditions of interest and/or on specific cell type at the time,[2–9] but not for investigating gene expression dynamics across more than two conditions, such as in a time-course dataset.

Here, we present a framework for identifying and investigating highly variable genes (HVGs) that demonstrate highly dynamic expression patterns across several time points (or biological conditions). Using our workflow, one can visualize relative expression levels of HVGs associated with biological pathways of interest in multiple time points and across different cell types in one analysis. Using publicly available time-course scRNA-seq datasets of dengue[1] and SARS-CoV-2[10] infections, we have demonstrated how the protocol can be used to extract HVGs, their biological processes, and gene expression patterns that are unique to certain cell types, or shared across multiple populations of cells.

Install tools/packages

1. Installation of R (https://www.r-project.org/), and optionally, RStudio (https://www.rstudio.com/).
2. Installation of the relevant R packages.
   a. All the R packages required from the analyses are listed under the *Software and Algorithms* section of the key resources table.
   b. Users can also download the Docker Hub, URL: https://hub.docker.com/r/jantarika/rstudio_denguetimecourse, that contains all the R packages used in this protocol.

Download scRNA-seq datasets

3. Download the datasets, which was used as an example in this protocol.
   a. Download the datasets mentioned in the *Deposited Data* section of the key resources table.
   b. Alternatively, the complete datasets used for this analysis are also deposited to Mendeley Data: https://data.mendeley.com/datasets/6ry3x7r8hf/3.

Institutional permission

The data described in this article were originally published by Arora and Opasawatchai and colleagues,[1] which was approved by the Institutional Review Boards of Faculty of Medicine Vajira Hospital (No.015/12), Faculty of Tropical Medicine Mahidol University (TMEC 13041) and Faculty of Medicine, Ramathibodi Hospital, Mahidol University (MURA2019/603).

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Dataset: Raw sequencing data of 4 time-points from DF and DHF patients and a healthy donor	Arora et al.[1]	ArrayExpress: E-MTAB-9467
Dataset: 4k PBMCs from a healthy donor	10x Genomics Single Cell Gene Expression Datasets	https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k
Complete datasets used for this protocol	Mendeley Data	https://data.mendeley.com/datasets/6ry3x7r8hf/3
Dataset: Processed scRNA-seq datasets of COVID-19 PBMC samples	Schulte-Schrepping et al.[10]	http://fastgenomics.org
Algorithms and computer codes, and the version record	Arora et al.[1]	https://github.com/vclabsysbio/scRNAseq_DVtimecourse https://doi.org/10.5281/zenodo.7968936
Software and algorithms		
RStudio 4.0.2	RStudio	https://www.rstudio.com/
Seurat v3.1.2	Stuart et al.[2]	https://satijalab.org/seurat/
SoupX v1.0.1	Young and Behjati.[11]	https://github.com/constantAmateur/SoupX
DoubletFinder v2.0.3	McGinnis et al.[12]	https://github.com/chris-mcginnis-ucsf/DoubletFinder
gProfier2 v0.1.8	Raudvere et al.[13]	https://biit.cs.ut.ee/gprofiler/
ggplot2 v3.3.2	Wickham.[14]	https://ggplot2.tidyverse.org/
tidyverse v.2.0.0	Wickham et al.[15]	https://www.tidyverse.org/packages/

MATERIALS AND EQUIPMENT

Bioinformatic analyses

All the bioinformatic analyses presented here were tested on an in-house server (Intel Core Processor (Broadwell, IBRS), 39 CPUs, and 480 GB RAM.), running on Ubuntu 16.04.6 LTS.

STEP-BY-STEP METHOD DETAILS

Before identification and investigation of time-course HVGs and their dynamic expression profiles, the scRNA-seq data must be pre-processed and quality controlled, which are summarized briefly here, and are described in details through GitHub (https://github.com/vclabsysbio/scRNAseq_DVtimecourse). Note that these steps have also been comprehensively described elsewhere.[16–18]

QC and filtering single-cell RNA-seq data

Timing: ~ 1–2 h/sample (for steps 1 to 3)

The quality control and filtering scRNA-seq data were performed separately on individual samples, before data integration and cell type annotation (Figure 1). The QC stage comprises correcting of ambient RNAs, exclusion of dead and low quality cells and potential doublets, in prior to the downstream bioinformatic analyses. Raw and filtered expression matrices generated by the *cellranger count* function are required as the inputs of this step - see key resources table.

Note: Figure 1 illustrates the overview of the bioinformatic pipeline described in this protocol.

Note: R packages and codes required for the QC and filtering step are provided in Github URL: https://github.com/vclabsysbio/scRNAseq_DVtimecourse. Complete datasets used for this step are deposited in Mendeley Data: https://data.mendeley.com/datasets/6ry3x7r8hf/3.

1. Correct ambient RNAs using SoupX.[11]

   Note: Check for the expression levels of ambient RNAs in your datasets. If an excessive amount is observed, users can apply several ambient RNA removal tools such as SoupX[11] to correct their expression levels. For more details of predicting and correcting the expression values of ambient RNAs, please refer to the SoupX tutorial.[11]

2. Exclude the cells expressing excessive mitochondrial genes, in this example, more than 10% in the total transcript counts.

   Note: Check the distribution of percent mitochondrial genes (percent.mt) to determine suitable cut-offs in your scRNA-seq datasets. For details, please refer to the Seurat toolkit.[2]

3. Predict and discard doublets using doubletFinder.[12]

   Note: For droplet-based scRNA-seq, we recommend excluding "doublets" and "multiplets". For more details about the optimal cut-offs in each parameter, please refer to doubletFinder.[12]

Data normalization, integration, and cell type annotation

Timing: ~ 2 h (for steps 4 to 7)

After filtering out low quality cells and correcting the expression values of ambient RNAs, the data are used as inputs for normalization, integration, and cell type annotation. Seurat objects of individual samples after QC and filtering step are used as the inputs of this step.

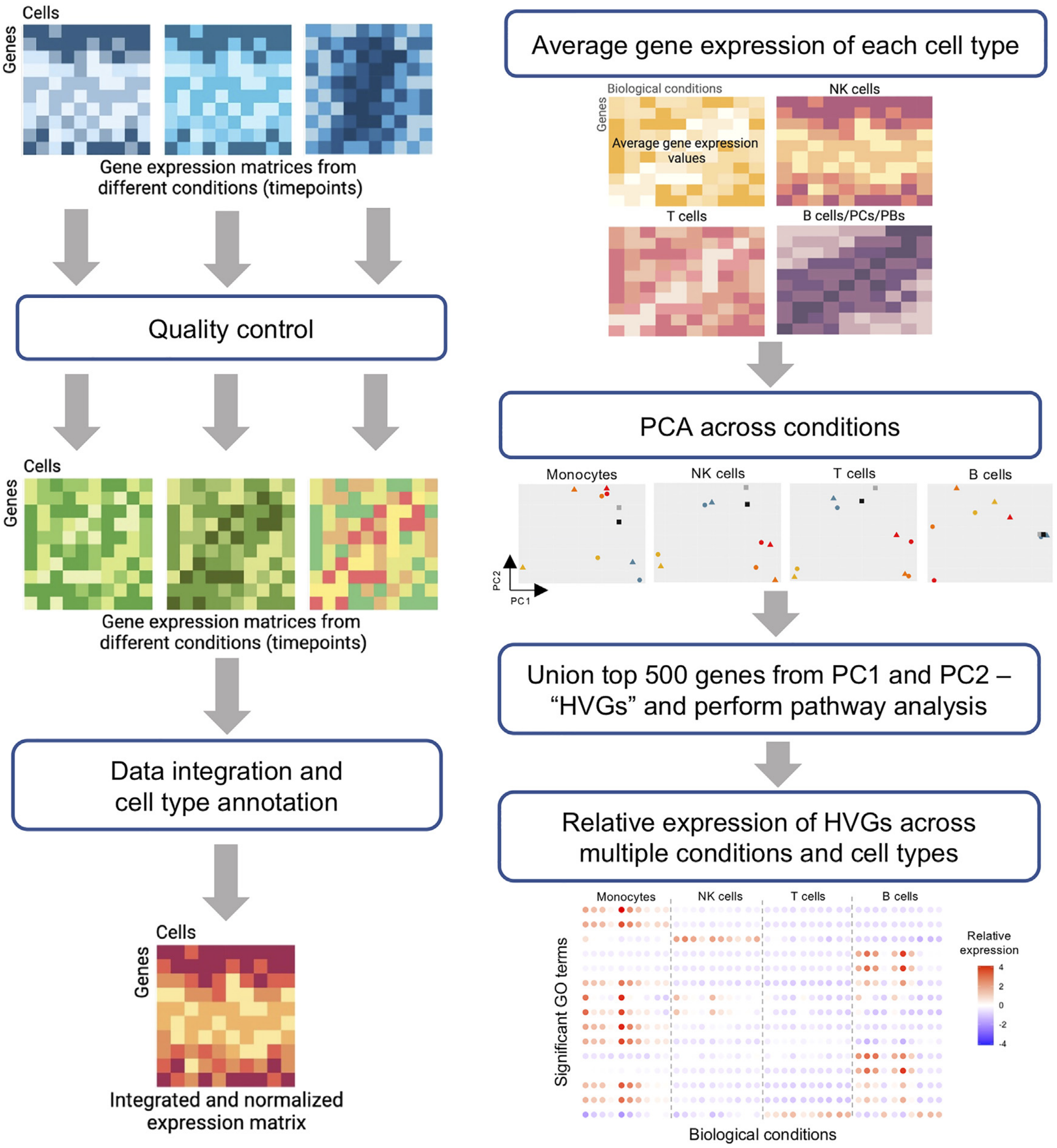

Cells
Genes
Gene expression matrices from different conditions (timepoints)
Quality control
Cells
Genes
Gene expression matrices from different conditions (timepoints)
Data integration and cell type annotation
Cells
Genes
Integrated and normalized expression matrix
Average gene expression of each cell type
Biological conditions
Genes
Average gene expression values
NK cells
T cells
B cells/PCs/PBs
PCA across conditions
Monocytes
NK cells
T cells
B cells
PC2
PC1
Union top 500 genes from PC1 and PC2 – "HVGs" and perform pathway analysis
Relative expression of HVGs across multiple conditions and cell types
Monocytes
NK cells
T cells
B cells
Significant GO terms
Relative expression
4
2
0
-2
-4
Biological conditions

Figure 1. Overview of bioinformatic pipeline of scRNA-seq data analysis described in this protocol

HVGs, highly variable genes.

4. Run the standard preprocessing workflow for each individual sample.
   a. Import and create a list of individual Seurat objects after the QC and filtering step.
   b. Run the *SCTransform* function, a regularized negative binomial regression.[19]
   c. Apply the first 30 principal components (PCs) for cell clustering.
   d. Identify cell clusters by the shared nearest neighbor (SNN) method using the Louvain algorithm with multilevel refinement.
5. Integrate all the samples. We used the following settings in our examples here: three thousand (3,000) features, a Louvain algorithm with multi-level refinement for clustering and the resolution of three. Other parameters are as default.
6. Normalize the transcript counts of each cell using the *NormalizeData()* function.

7. Identify major immune cell types using known canonical marker genes (as described in Table S1).

   Note: For cell type annotation, in our examples we first labeled different clusters of T cells with the same name in order to group them together as a single "T cells" cluster, which was subsequently re-clustered into subpopulations.

   Note: High heterogeneity of PBMCs, especially at the subpopulation levels, might be difficult to identify using the known marker genes (Table S1). Alternatively, reference-based for cell type annotation such as SingleR,[20] Azimuth,[21] and ScType[22] can be applied.

Highly variable gene (HVG) identification and pathway enrichment analysis across multiple time points – Dengue case study

⏱ Timing: ~ 10–15 min (for steps 8 to 14)

Here, we describe the process for obtaining HVGs, which represent the variations in transcriptional levels across several biological conditions (typically >2 conditions, in our case, four time points from two DENV-infected patients and two healthy controls) in multiple immune cell types of interest. The inputs of this step are the integrated scRNA-seq Seurat objects from the previous pro-processing and cell type annotation described above (Figure 1).

Note: We first performed a pseudo-bulk expression analysis by calculating the average transcription levels of all the genes. Then, principal component analysis (PCA) of the averaged gene expression in each cell type was performed (Figure 2A). This PCA of each cell type was used to extract the genes that demonstrate the largest variation across biological conditions, – to be referred to as "Highly Variable Genes (HVGs)". In this example, the most apparent differences are between the four time points, whereas those between the two DENV patients are relatively small. Finally, we investigated the biological pathways associated with the HVGs that show common and unique expression dynamics among different cell types and biological conditions (Figure 2B). The workflow is described step-by-step below.

Note: Processed scRNA-seq dataset used in this step is deposited in Mendeley Data: https://data.mendeley.com/datasets/6ry3x7r8hf/3

8. Subset the integrated Seurat object according to their major cell types into monocytes, NK cells, T cells, and B cells/plasma cells/plasmablasts - the main immune cell types of interest in this study.

```
# Load libraries

library(Seurat)

library(gprofiler2)

library(ggplot2)

# Set your working directory, pointing to the folder where all your input and output files will be
saved

setwd("PATH/TO/YOUR/WORKING/DIRECTORY")

# Load integrated data

sc_integrated <- readRDS(file = "PATH/TO/YOUR/WORKING/DIRECTORY/sc_integrated.rds")

# Subset each cell type
```

```
Idents(sc_integrated) <- "Cell_Types"

each_celltype_list <- list()

each_celltype <- c("Monocytes" , "NK cells" , "T cells" , "B cells")

for (RN in each_celltype) {

  each_celltype_list[[RN]] <- subset(sc_integrated , idents = RN)

}

names(each_celltype_list) <- each_celltype
```

9. Calculate the average gene expression levels in each cell type across biological conditions of interest.
   a. Set the object's identity class based on condition of interest using the *Idents()* function.
   b. Calculate the average gene expression using the *AverageExpression()* function.

```
# Calculate average gene expression in each cell type across severity and time

Avg_expression_list <- list()

for (RN in 1:length(each_celltype_list)) {

  Idents(each_celltype_list[[RN]]) <- "ST"

  Avg_expression_list[[RN]]  <-  AverageExpression(each_celltype_list[[RN]]  ,  assays  =
"RNA" , slot = "data")

  Avg_expression_list[[RN]] <- as.data.frame(Avg_expression_list[[RN]]$RNA)

}

names(Avg_expression_list) <- names(each_celltype_list)
```

10. Construct the principal components (PCs) of the average gene expression levels in each cell type using the *prcomp()* function in R (Figure 2A).

```
# Construct Principal Components (PCs) in each cell type

pca_out <- list()

pca_perc <- list()

df_pca <- list()

for (RN in 1:length(Avg_expression_list)) {

  pca_out[[RN]] <- prcomp(t(Avg_expression_list[[RN]]))

  pca_perc[[RN]] <- round(100*pca_out[[RN]]$sdev^2/sum(pca_out[[RN]]$sdev^2),1)

  df_pca[[RN]] <- data.frame(PC1 = pca_out[[RN]]$x[,1], PC2 = pca_out[[RN]]$x[,2], sample =
colnames(Avg_expression_list[[RN]]))

  # Add metadata can be differences in each dataset

  df_pca[[RN]]$Severity <- c(rep("DF" , 4) , rep("DHF" , 4) , "Healthy" , "Healthy")

  df_pca[[RN]]$Time <- c("Day-2" , "Day-1" , "Def" , "Wk2" , "Day-2" , "Day-1" , "Def" , "Wk2"
,"Healthy I" , "Healthy II")
```

```
  df_pca[[RN]]$Time <- factor(df_pca[[RN]]$Time , levels = c( "Day-2" , "Day-1" , "Def" , "Wk2" , "Healthy I" , "Healthy II"))

}

names(pca_out) <- names(Avg_expression_list)

names(pca_perc) <- names(Avg_expression_list)

names(df_pca) <- names(Avg_expression_list)
```

Note: Please add metadata based on your datasets.

11. Visualize the PCA results using the ggplot2 package.[14]

```
# Visualize PCA results

for (RN in 1:length(df_pca)) {

  pca_plot<- ggplot(df_pca[[RN]], aes(PC1,PC2, color = Time))+ geom_point(aes(shape = Severity ), size=6 , stroke = 1.4)+ labs(x=paste0("PC1 (",pca_perc[[RN]][1],"%)"), y=paste0("PC2 (",pca_perc[[RN]][2],"%)")) + scale_color_manual(values=c("darkgoldenrod2", "#ff7400","#ff1218", "#47849c" , "darkgrey" , "gray6")) + theme(axis.text = element_text (size = 17 , face="bold" , colour = "black") , axis.title.y = element_text(color="black", size=15, face="bold") , axis.title.x = element_text(color="black", size=17, face="bold") , legend.title = element_text(face = "bold" , size = 17) , legend.text = element_text(size = 16) , legend.key.size = unit(1, "cm") , legend.key.width = unit(0.5,"cm") , legend.key = element_rect(fill = "white") ) + ggtitle(names(df_pca[RN]))

plot(pca_plot)

}
```

Note: Color and shape can be manually adjusted based on your data.

12. Union the top 500 genes from the first and second PCs of each immune cell type – referred to as "HVGs" herein.

```
# Union top 500 genes from PC1 and PC2 from all cell types

HVGs_each_celltype <- list()

for (RN in 1:length(pca_out)) {

  HVGs_each_celltype[[RN]] <- union(rownames(data.frame(sort(abs(pca_out[[RN]]$rotation [,"PC1"]), decreasing=TRUE)[1:500])) , rownames(data.frame(sort(abs(pca_out[[RN]]$rotation[,"PC2"]), decreasing=TRUE)[1:500])))

}

names(HVGs_each_celltype) <- names(pca_out)

# unique HVGs based on number of cell types

HVGs <- unique(c(HVGs_each_celltype[[1]] , HVGs_each_celltype[[2]] , HVGs_each_celltype [[3]] , HVGs_each_celltype[[4]]))
```

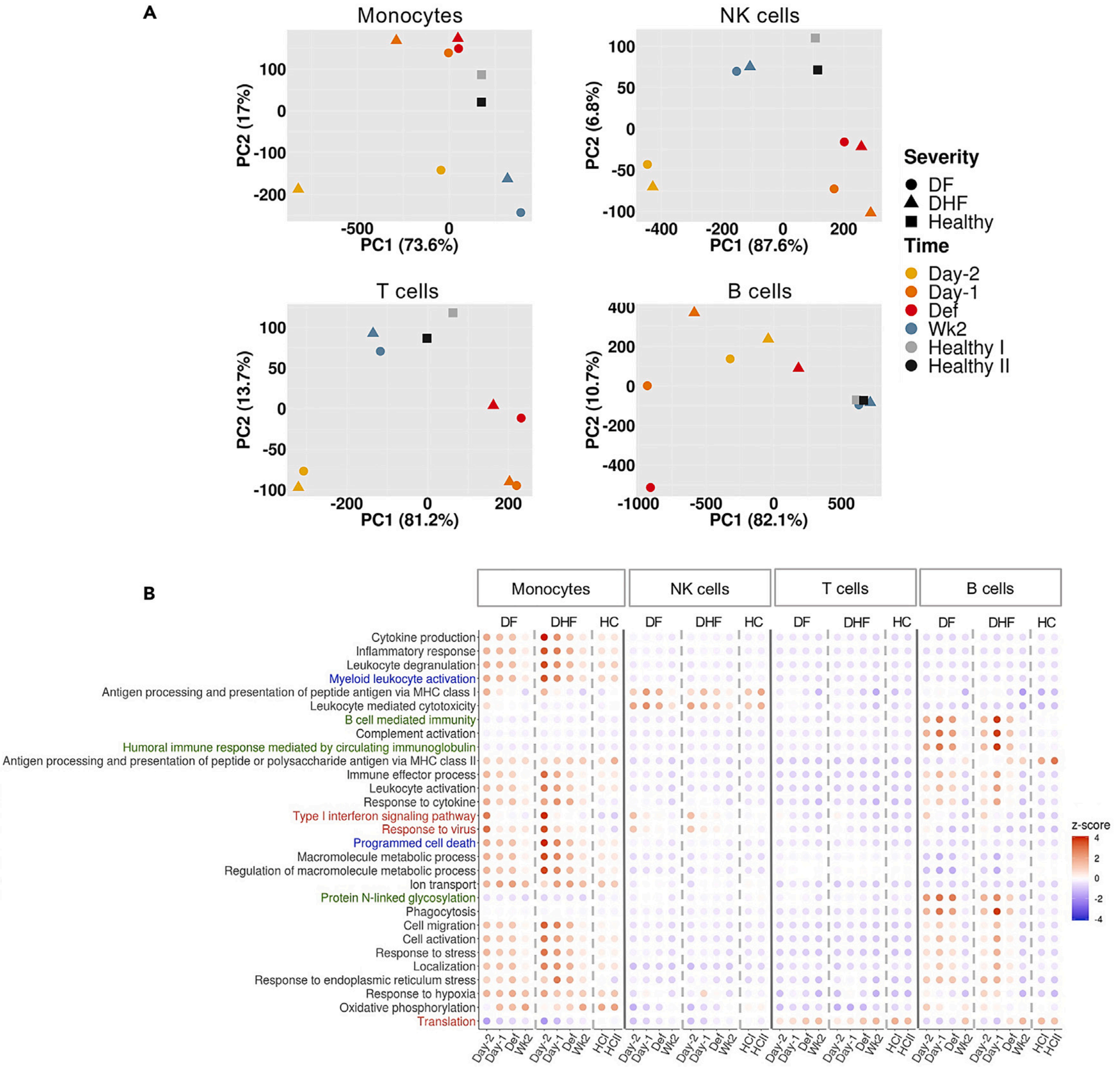


Figure 2. The relative expression levels of highly variable genes (HVGs) in each biological process (BP) of interest, across the four time points and major immune cell types, from two dengue-infected patients and two healthy controls

(A and B) Figures were modified from Arora et al., 2022[1] (A). PCA of average gene expression in each major immune cell type of interest. (B). Dotplot representing relative expression levels of HVGs. The BPs of the HVGs across the four major immune cell types of interest here are highlighted in red texts, monocyte-specific BPs are in blue, and B cell-specific BPs are in green.

⚠ CRITICAL: Edit the *unique()* function based on the numbers of cell types in your dataset. In this case, we investigated four major immune cell types: monocytes, NK cells, T cells, and B cells.

Note: To find the optimal numbers of top genes that exhibit high variations in PCs, users can construct and investigate a histogram plot where the y-axis represents PC loading calculated from the *prcomp()* function in R, and the x-axis shows the numbers of genes (Figure S1).

Example code.

```
# Extract PC loading values calculated from prcomp()

PC1_mono <- sort(abs(pca_out[[1]]$rotation[,"PC1"]), decreasing=TRUE)
```

```
PC1_NK <- sort(abs(pca_out[[2]]$rotation[,"PC1"]), decreasing=TRUE)

PC1_Tcells <- sort(abs(pca_out[[3]]$rotation[,"PC1"]), decreasing=TRUE)

PC1_Bcells <- sort(abs(pca_out[[4]]$rotation[,"PC1"]), decreasing=TRUE)

# Plot

plot(density(PC1_mono)$y, density(PC1_mono)$x,type="l" , col = "orange" , ylab = "PC1
loading values", xlab = "Number of genes" , main = " " , ylim = c(0,0.02) , xlim = c(0, 2000))

# Add lines

lines(density(PC1_NK)$y, density(PC1_NK)$x,type="l" , col = "red" , lwd=1)

lines(density(PC1_Tcells)$y, density(PC1_Tcells)$x,type="l" , col = "blue" , lwd=1)

lines(density(PC1_Bcells)$y, density(PC1_Bcells)$x,type="l" , col = "green" , lwd=1)

# Add a legend

legend(1550, 0.020, legend = c("Monocytes", "NK cells", "T cells" , "B cells"), fill=c( "or-
ange","red","blue","green" ) , box.lty=0 )

# Add vertical dashed blue line at x = 500

abline(v=500, col="blue" , lty = "dashed")
```

13. Perform a pathway enrichment analysis of the HVGs of all cell types using *gProfiler2*.[13]
    a. All the genes in the genome are used as the background gene set.
    b. Perform the pathway analysis using the gost() function.

```
# Pathway enrichment analysis

# Extract all genes that will be used as the background for pathway analysis

bg <- rownames(Avg_expression_list[[1]])

# Pathway enrichment analysis using gProfiler2

GO_out <- gost(query = HVGs , organism = "hsapiens" , correction_method = "fdr" , custom_bg = bg
, significant = TRUE , user_threshold = 0.05 , evcodes = TRUE , sources = "GO:BP")
```

> ***Note:*** Beside *gProfiler2*,[13] alternatively, functional enrichment analysis can be performed using several computational/web tools such as clusterProfiler,[23] Gene Ontology Consortium,[24] Database for Annotation, Visualization and Integrated Discovery (DAVID),[25] Kyoto Encyclopedia of Genes and Genomes (KEGG),[26] and Reactome.[27]

14. Save your outputs (optional).

```
# Save outputs

saveRDS(Avg_expression_list, file = "PATH/TO/YOUR/WORKING/DIRECTORY/Avg_expression_list.
rds")

saveRDS(GO_out , file = "PATH/TO/YOUR/WORKING/DIRECTORY/GO_out.rds")

write.csv(HVGs , file = "PATH/TO/YOUR/WORKING/DIRECTORY/HVGs.csv", row.names = F)

write.csv(bg , file = "PATH/TO/YOUR/WORKING/DIRECTORY/bg.csv", row.names = F)
```

Investigating dynamic expression patterns of HVGs across all time points and cell types

Timing: ~ 5–10 min (for steps 15 to 23)

In this section, we describe the normalization method used to obtain relative expression levels of HVGs associated with biological pathways of interest, across four different time points as well as cell types (Figure 2B). The integrated Seurat object, average gene expression levels in each cell type, and the selected GO:BP (biological process gene ontology) terms from the pathway enrichment analysis are used as the inputs of this step.

15. Import the integrated Seurat object, average gene expression levels in each cell type, and biological pathways of interest into R.

```
# Load libraries

library(Seurat)

library(gprofiler2)

library(ggplot2)

library(tidyverse)

# Set your working directory, pointing to the folder where all your input and output files will be
saved

setwd("PATH/TO/YOUR/WORKING/DIRECTORY")

# Load objects

sc_integrated <- readRDS(file = file = "PATH/TO/YOUR/WORKING/DIRECTORY/sc_integrated.rds")

Avg_expression_list <- readRDS(file = "PATH/TO/YOUR/WORKING/DIRECTORY/Avg_expression_
list.rds")

selected_GO_out <- readRDS(file = "PATH/TO/YOUR/WORKING/DIRECTORY/selected_GO_out.rds")

selected_GO_out <- selected_GO_out$result
```

16. Calculate the mean value of each gene across all cells from the integrated Seurat object.

```
# Calculate the mean value of each gene from all cells

exp_matrix <- GetAssayData(sc_integrated, slot = "data", assay = "RNA") %>% data.frame() %>%
rownames_to_column()

rownames(exp_matrix) <- exp_matrix$rowname

exp_matrix$rowname <- NULL

exp_matrix$Mean <- rowMeans(exp_matrix)

exp_matrix$Gene <- rownames(exp_matrix)

Mean_all_cells <- exp_matrix[c("Gene", "Mean")]
```

17. Create a list of HVGs in each "Gene Ontology: Biological processes (GO:BP)".

```
# Create a list of HVGs in each GO:BP
genes_each_GO <- list()
for (RN in 1:nrow(selected_GO_out)) {
  temp <- strsplit(unique(paste(as.character(selected_GO_out$intersection[RN]) , sep =
",")) , ",")
  genes_each_GO[[RN]] <- temp[[1]]
}
names(genes_each_GO) <- selected_GO_out$term_name
```

18. Extract the average gene expression values of HVGs in each selected GO:BP.

```
# Extract average gene expression values of HVGs
Avg_expression_each_GO <- list()
temp_list <- list()
for (RN in 1:length(Avg_expression_list)) {
  for (JA in 1:length(genes_each_GO)) {
    temp_list[[JA]] <- Avg_expression_list[[RN]][rownames(Avg_expression_list[[RN]]) %in
% genes_each_GO[[JA]],]
    names(temp_list)[[JA]] <- names(genes_each_GO)[[JA]]
  }
Avg_expression_each_GO[[RN]] <- temp_list
}
names(Avg_expression_each_GO) <- names(Avg_expression_list)
```

19. Normalize each gene by adding a pseudocount of 1 (to avoid undefined results from a zero denominator – in unexpressed genes), and then divide by the mean value from all cells.

```
# Normalise each gene by adding a pseudocount of 1, and divide by mean value from all cells.
Avg_expression_each_GO_norm <- list()
for (RN in 1:length(Avg_expression_each_GO)) {
  temp <- Avg_expression_each_GO[[RN]]
  for (JA in 1:length(temp)) {
    temp_1 <- Mean_all_cells[Mean_all_cells$Gene %in% rownames(temp[[JA]]),]
    temp_1
    temp[[JA]] <- (temp[[JA]] + 1) / (temp_1$Mean + 1)
  }
  Avg_expression_each_GO_norm[[RN]] <- temp
}
names(Avg_expression_each_GO_norm) <- names(Avg_expression_list)
```

20. Calculate the sum of the normalized average gene expression of all HVGs belonging to each GO:BP of interest.

```
# Sum of each gene in each GO:BP
Sum_avg_expression_each_GO <- list()
for (RN in 1:length(Avg_expression_each_GO_norm)) {
  temp <- Avg_expression_each_GO_norm[[RN]]
  for (JA in 1:length(temp)) {
    temp[[JA]][names(Avg_expression_each_GO_norm[RN]),] <- colSums(temp[[JA]])
  }
  Sum_avg_expression_each_GO[[RN]] <- temp
}
names(Sum_avg_expression_each_GO) <- names(Avg_expression_each_GO_norm)
```

21. Create a big dataframe, where each row represents each GO:BP and each column represents each cell type in each condition.

```
# Create list of summation each GO:BP in each cell type
Sum_exp_each_GO_celltype <- list()
for (RN in 1:length(genes_each_GO)) {
  Sum_exp_each_GO_celltype[[RN]] <- rbind(tail(Sum_avg_expression_each_GO[[1]][[RN]] , 1) ,
    tail(Sum_avg_expression_each_GO[[2]][[RN]] , 1) ,
    tail(Sum_avg_expression_each_GO[[3]][[RN]] , 1) ,
    tail(Sum_avg_expression_each_GO[[4]][[RN]] , 1))
}
names(Sum_exp_each_GO_celltype) <- names(genes_each_GO)
# Create a big dataframe, each row represents each GO:BP, and each column represents each cell type in each condition.
# matrix_size = 1 : (no_of_cell_types x biological_conditions)
no_cell_types <- 4
no_samples <- 10
matrix_size <- no_cell_types * no_samples
Input_dot_plot <- as.data.frame(matrix(1:matrix_size, nrow = 1, ncol = matrix_size))
colnames(Input_dot_plot) <- c(paste("Monocytes" , names(Sum_exp_each_GO_celltype[[1]])) , paste("NK" , names(Sum_exp_each_GO_celltype[[1]])) , paste("T" , names(Sum_exp_each_GO_celltype[[1]])) , paste("B" , names(Sum_exp_each_GO_celltype[[1]])))
for (RN in 1:length(genes_each_GO)) {
  temp <- Sum_exp_each_GO_celltype[[RN]]

  temp <- cbind(temp[1,],temp[2,] , temp[3,] , temp[4,])

  colnames(temp) <- c(paste("Monocytes" , names(Sum_exp_each_GO_celltype[[1]])) , paste("NK" , names(Sum_exp_each_GO_celltype[[1]])) , paste("T" , names(Sum_exp_each_GO_celltype[[1]])) , paste("B" , names(Sum_exp_each_GO_celltype[[1]])))

  Input_dot_plot <- rbind(Input_dot_plot , temp)

}

Input_dot_plot <- Input_dot_plot[-1,]

rownames(Input_dot_plot) <- names(genes_each_GO)
```

22. Calculate the z-score across all samples and cell types for visualization (Figure 2B).

```
# For visualization, calculate z-score by row

Input_dot_plot_zscore <- t(apply((Input_dot_plot[,1:length(Input_dot_plot)]), 1, function(x){

  mean <- mean(x)

  SD <- sd(x)

  Z_score <- (x-mean)/SD

  Z_score

}))

Input_dot_plot_zscore <- as.data.frame(Input_dot_plot_zscore)

# Convert to long format for plot

forplot <- gather(Input_dot_plot_zscore %>% rownames_to_column("GO"), key = "ST" , value = "Expression" , -GO)

forplot <- forplot %>% separate(col = "ST" , into = c("Cell_Types" , "Patients" , "time"),sep = " " , remove = F)

forplot$time <- factor(forplot$time , levels = c("Day-2" , "Day-1" , "Def" , "Wk2" , "I" , "II"))

forplot$Cell_Types <- factor(forplot$Cell_Types , levels = c("Monocytes" , "NK" , "T" , "B"))

forplot$Patients <- factor(forplot$Patients , levels = c("DF" , "DHF" , "Healthy"))
```

23. Visualize the results using the ggplot2 package.[14]

```
# Visualize the relative expression of HVGs over all samples and cell types

ggplot(forplot, aes(x=time, y=GO , color= Expression , size = Expression)) + geom_point(alpha = 1.5) + theme_classic() + scale_colour_gradient2( low = "blue", mid = "white", high = "red", space = "Lab" , limits = c(-max(forplot$Expression),max(forplot$Expression)) ) + xlab("") + scale_size_continuous(range = c(5,5)) + facet_grid(~Cell_Types+Patients , scales = "free", space = "free") + labs(color = paste("z-score")) + theme(axis.text.x = element_text(angle = 60, hjust=1), axis.text=element_text(size=20) , axis.title.y = element_
text(size=25), strip.background = element_rect(colour = "white", fill = c("gray","darkorange3", "gray")), legend.text = element_text(size = 15), legend.title = element_text(size =
20),legend.key.size = unit(1, "cm")) + ylab("Significant GO terms")
```

HVG identification and pathway enrichment analysis across early and late time points in immune cell types - COVID-19 case study

Timing: ∼ 10–15 min

In addition to the dengue data published in our earlier study,[1] here we also demonstrate the application of this framework on another time-course scRNA-seq study. We retrieved processed scRNA-seq data from a COVID-19 study,[10] which was deposited on FASTGenomics (https://www.fastgenomics.org/). We selected two COVID-19 patients (“cohort 1”) with the donor IDs "C19-CB-0009” and "C19-CB-0012”, whose samples were obtained at the same “days after symptom onset”; together with two healthy donors, "P15F” and "P17H". We used the same cell type annotation as initially characterized by the authors of the study.[10]

> *Note:* Using our framework to analyze this COVID-19 dataset, we observed distinct overall transcriptome profiles between early and late phases of the infection in each of the cell types of interest, while the differences between the two patients were relatively small (Figure 3A). Moreover, the overall expression patterns of the two healthy donors were nicely grouped together and clearly separated from the COVID-19 patients at all the time points (Figure 3A). We next identified HVGs and virtualized their dynamic expression patterns, for different groups of biological pathways that the HVGs are associated with (Figure 3B). Interestingly, we observed clear up-regulation of the HVGs in the early infection associated with “response to type I interferon”, “interferon-mediated signaling pathway”, and “type I interferon-mediated signaling pathway” in monocytes (Figure 3B, labeled in blue), which have also been described in the original paper,[10] but using a different analytical framework.[10] In addition to this, our analysis and visualization also revealed cell-type-specific expression dynamics of the HVGs associated with certain biological pathways, such as “natural killer cell mediated cytotoxicity” HVGs being specifically upregulated in NK cells; “T cell receptor V(D)J recombination” HVGs being expanded in CD4+ and CD8+ T cells, and several B cells-related pathways such as “B cell activation”, “humoral immune response”, and “B cell mediated immunity” being up-regulated specifically in B cells at the early infection (Figure 3B), all of which have not been mentioned in the original study.[10]

EXPECTED OUTCOMES

We propose an approach for identification of highly variable genes (HVGs) and investigation of their expression patterns across multiple time points (or biological conditions). Similarly to differentially expressed genes (DEGs) that are commonly used to represent the genes that might be activated or repressed differentially between two biological conditions, such as between patients vs.’ healthy controls, or treatment vs.’ normal controls, HVGs depict the genes that show the highest dynamics across multiple time points, and their functions might be linked to temporally specific biological processes over the course of the studies.

The overall variations between multiple transcriptomes obtained in different biological conditions can be consolidated using Principal Component Analysis (PCA) and then visualized in a two-dimensional plot. With the scRNA-seq data, we can readily obtain and investigate the PCA plots of different cell types and subpopulations (Figures 2A and 3A). HVGs of different cell types are also identified during this PCA process, and used as the inputs of pathway

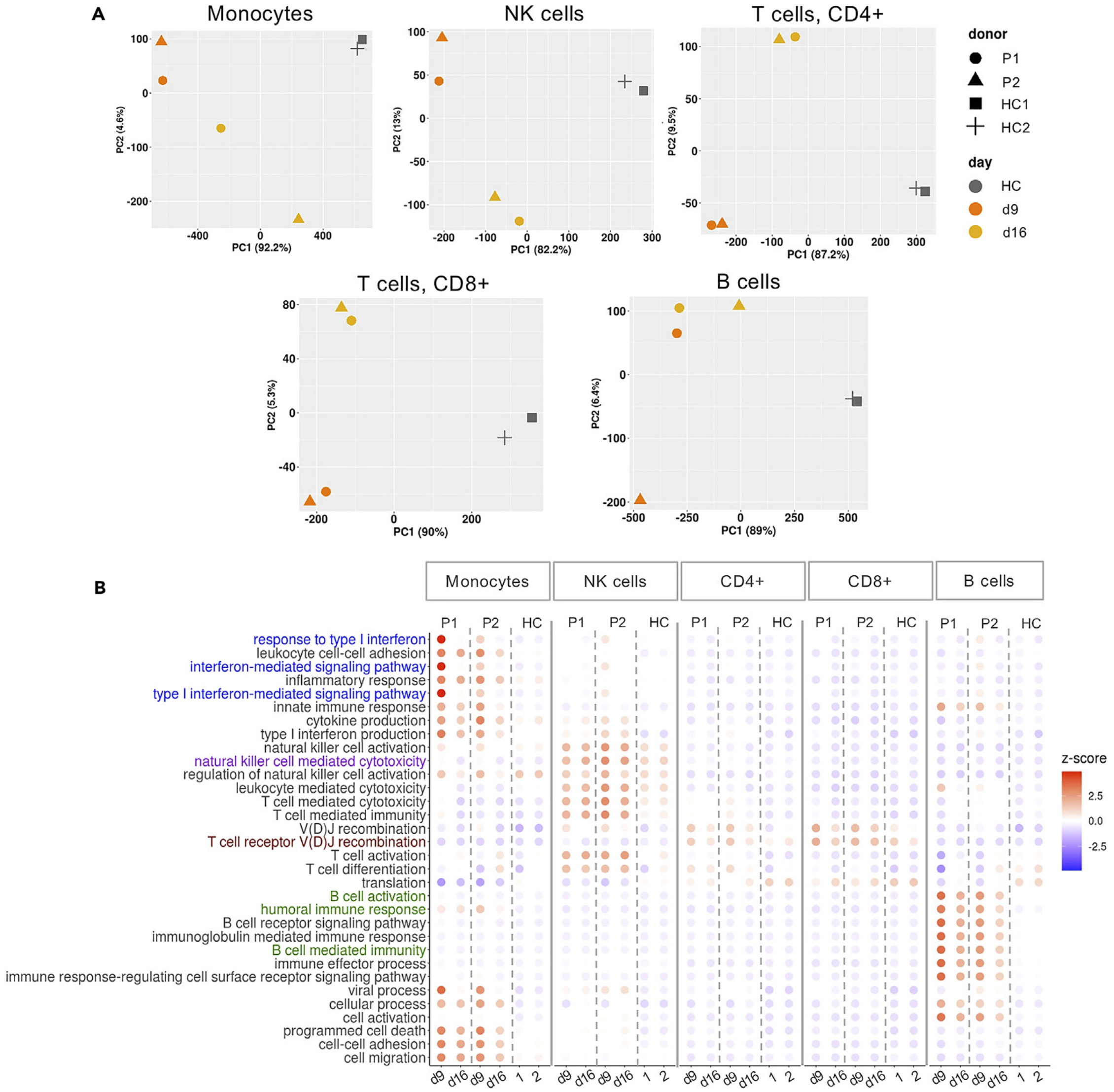


Figure 3. The relative expression values of HVGs in each biological process (BP) across two COVID-19 patients during the early time point (d9; 9 days after symptom onset) and late time point (d16; 16 days after symptom onset), together with two healthy controls

(A) PCA of average gene expression in each cell type. Different colors represent the days after symptom onset, while different shapes represent individual samples.

(B) Dotplot showing the relative expression levels of HVGs related to each BP. The time-specific BPs described in the original paper[10] are labeled in blue. The NK-specific BP is highlighted in purple, T-specific BP is in dark red, and B-specific BPs are in green. P1 = covid-19 patient 1; P2 = covid-19 patient 2; HC = healthy controls. d9 = 9 days after symptom onset; d16 = 16 days after symptom onset.

enrichment analysis. Relative expression patterns across time points of HVGs associated with biological processes of interest are then computed and visualized across different cell types in a single figure, as shown in Figures 2B and 3B. As demonstrated using the dengue[1] and COVID-19[10] case studies, we can identify the biological processes whose HVGs demonstrate cell-type-specific expression dynamics across the time points, and hence suggest the links between time-point specific gene expression regulation and relevant biological processes in each of the cell types of interest. In addition to scRNA-seq, we expect that the framework will be adaptable for the investigation of temporal or longitudinal gene expression datasets obtained using other high-throughput and omic technologies, such as time-course bulk RNA-seq or proteomic studies.

LIMITATIONS

Batch effect can occur when the samples obtained using different technologies or platforms. In the protocol described here, all the samples were processed using the same versions of the 10x genomics single-cell kits. Hence, we did not observe technical batch effects. If multiple samples used in the analysis are generated by different techniques, it is advisable to inspect the batch effect by PCA before starting the HVG analysis. Library preparation techniques could appear as the factor most affecting the grouping of the samples. In such cases, it may not be possible to obtain HVGs that truly represent the most variances from biological conditions using our described protocols. It might be possible; however, to regress such batch effects using tools such as Harmony,[28] Mutual Nearest Neighbors (MNN),[29] and Seurat Canonical Correlation Analysis (CCA).[30]

TROUBLESHOOTING

Problem 1

Can this protocol be used with a .h5ad input file for the HVG identification (related to step 9)?

Potential solution

A .h5ad file can be converted into a Seurat object, and the average gene expression can be calculated using the codes below.

Convert a .h5ad file to a Seurat object.

```
library(zellkonverter)

sce <- readH5AD("PATH/TO/file.h5ad", verbose = TRUE)

seurat_obj <- as.Seurat(sce, counts = "X", data = NULL)

Calculate average gene expression levels in each cell type:

Avg_expression_list <- list()

for (RN in 1:length(each_celltype_list)) {

  Idents(each_celltype_list[[RN]]) <- "ST"

  Avg_expression_list[[RN]] <- AverageExpression(each_celltype_list[[RN]] , assays =
"originalexp" , slot = "data")

  Avg_expression_list[[RN]] <- as.data.frame(Avg_expression_list[[RN]]$originalexp)

}

names(Avg_expression_list) <- names(each_celltype_list)
```

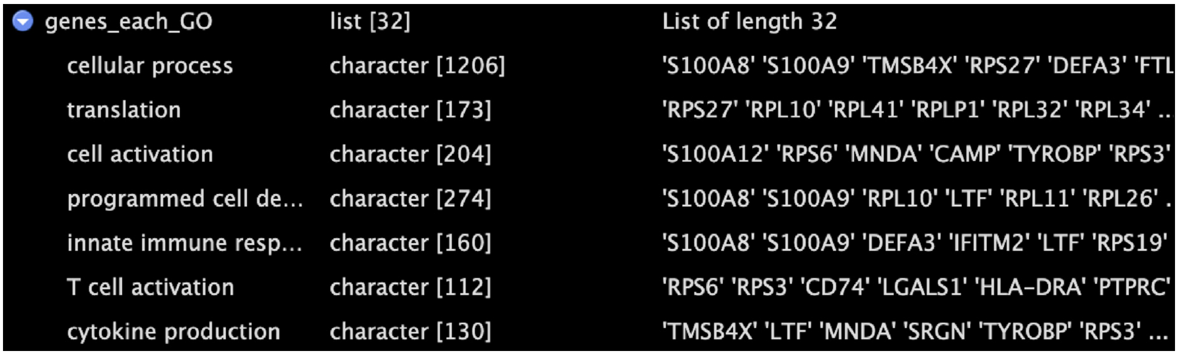
Problem 2

The code described in step 17 is specific for the output generated from the *gProfiler2* package.[13] If different functional enrichment tools are used, the outputs might be different from the *gProfiler2* one, which could result in an error when running step 17.

Potential solution

Users can alternatively create a list of GO terms themselves, as shown below. Each GO term in a list should contain genes that are related to that particular GO term. After creating this list, users should be able to continue with step 18.

An example of a list,



```
genes_each_GO          list [32]          List of length 32
  cellular process     character [1206]   'S100A8' 'S100A9' 'TMSB4X' 'RPS27' 'DEFA3' 'FTL
  translation          character [173]    'RPS27' 'RPL10' 'RPL41' 'RPLP1' 'RPL32' 'RPL34' ..
  cell activation      character [204]    'S100A12' 'RPS6' 'MNDA' 'CAMP' 'TYROBP' 'RPS3'
  programmed cell de...  character [274]  'S100A8' 'S100A9' 'RPL10' 'LTF' 'RPL11' 'RPL26' .
  innate immune resp...  character [160]  'S100A8' 'S100A9' 'DEFA3' 'IFITM2' 'LTF' 'RPS19'
  T cell activation    character [112]    'RPS6' 'RPS3' 'CD74' 'LGALS1' 'HLA-DRA' 'PTPRC'
  cytokine production  character [130]    'TMSB4X' 'LTF' 'MNDA' 'SRGN' 'TYROBP' 'RPS3' ...
```

Problem 3

The "ambiguous" gene names are sometimes not recognized by the *gProfiler2* package[13] (related to step 13).

Potential solution

We encourage users to use the Ensembl IDs as default inputs.

Problem 4

Large R, Rmd scripts, or R objects sometimes cannot be loaded and/or run on Rstudio (related to HVG identification and pathway enrichment analysis and investigating dynamic expression patterns of HVGs across all time points and cell types sections).

Potential solution

We encourage users to run the scripts using .R via R or RServer with less Graphic User Interface (GUI) instead.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead contact, Varodom Charoensawan (Email: varodom.cha@mahidol.ac.th).jantarikaarora@gmail.com, anunya.opa@mahidol.edu

Materials availability

This study did not generate new unique materials.

Data and code availability

- The complete dataset used for this protocol is deposited to Mendeley Data; https://data.mendeley.com/datasets/6ry3x7r8hf/3
- Analyses were conducted in R; all the coded described in this protocol have been deposited as GitHub data: https://github.com/vclabsysbio/scRNAseq_DVtimecourse, and the version record can be found on Zenodo: https://doi.org/10.5281/zenodo.7968936.
- Further additional information is also available through personnel contacts with the Lead and/or Technical contacts upon request.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2023.102387.

ACKNOWLEDGMENTS

J.A. is supported by the Royal Golden Jubilee (RGJ) Ph.D. Programme (PHD/0091/2559), through the National Research Council of Thailand (NRCT), Thailand Science Research and Innovation (TSRI), and Mahidol University. A.O. is supported by the Research Grant for New Scholars by the Office of the Permanent Secretary, Ministry of Higher Education, Science, Research and Innovation (Grant No. RGNS 65–217). V.C. laboratory is supported by the mid-career researcher grant from National Research Council of Thailand (NRCT) and Mahidol University (NRCT5-RSA63015-24); Mahidol University's Basic Research Fund: fiscal year 2021 (BRF1-017/2564); National Research Council of Thailand (NRCT) (N35A640274). V.C. and P.M. are jointly supported by the Program Management Unit for National Competitiveness Enhancement (PMU-C) (C10F650132). The original data set was part of the projects funded by the Newton Advanced Fellowship to V.C. and S.A.T., through the Royal Society (grant number: NA160153) and the Thailand Research Fund (DBG60800003); and the Newton Fund Research grant to P.M. and S.A.T. through the Wellcome grant (206194), the Medical Research Council (MR/R020868/1).

AUTHOR CONTRIBUTIONS

Conceptualization, J.A., A.O., P.M., V.C.; research design and methodology, J.A., A.O., P.M., S.A.T., V.C.; data analysis and interpretation, J.A., A.O., V.C.; writing – original draft, J.A., A.O., V.C.; writing – review & editing, J.A., A.O., P.M., S.A.T., V.C.; supervision, P.M., S.A.T., V.C.; funding acquisition, P.M., S.A.T., V.C.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Arora, J.K., Opasawatchai, A., Poonpanichakul, T., Jiravejchakul, N., Sungnak, W., DENFREE Thailand, Matangkasombut, O., Teichmann, S.A., Matangkasombut, P., and Charoensawan, V. (2022). Single-cell temporal analysis of natural dengue infection reveals skin-homing lymphocyte expansion one day before defervescence. iScience *25*, 104034.

2. Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W.M., 3rd, Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. Cell *177*, 1888–1902.e21. https://doi.org/10.1016/j.cell.2019.05.031.

3. Miao, Z., Deng, K., Wang, X., and Zhang, X. (2018). DEsingle for detecting three types of differential expression in single-cell RNA-seq data. Bioinformatics *34*, 3223–3224. https://doi.org/10.1093/bioinformatics/bty332.

4. Love, M.I., Huber, W., and Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biol. *15*, 550. https://doi.org/10.1186/s13059-014-0550-8.

5. Robinson, M.D., McCarthy, D.J., and Smyth, G.K. (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics *26*, 139–140. https://doi.org/10.1093/bioinformatics/btp616.

6. McCarthy, D.J., Chen, Y., and Smyth, G.K. (2012). Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. Nucleic Acids Res. *40*, 4288–4297. https://doi.org/10.1093/nar/gks042.

7. Chen, Y., Lun, A.T.L., and Smyth, G.K. (2016). From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline. F1000Res. *5*, 1438. https://doi.org/10.12688/f1000research.8987.2.

8. Finak, G., McDavid, A., Yajima, M., Deng, J., Gersuk, V., Shalek, A.K., Slichter, C.K., Miller, H.W., McElrath, M.J., Prlic, M., et al. (2015). MAST: a flexible statistical framework for assessing transcriptional changes and characterizing heterogeneity in single-cell RNA sequencing data. Genome Biol. *16*, 278. https://doi.org/10.1186/s13059-015-0844-5.

9. Cao, J., Spielmann, M., Qiu, X., Huang, X., Ibrahim, D.M., Hill, A.J., Zhang, F., Mundlos, S., Christiansen, L., Steemers, F.J., et al. (2019). The single-cell transcriptional landscape of mammalian organogenesis. Nature *566*, 496–502. https://doi.org/10.1038/s41586-019-0969-x.

10. Schulte-Schrepping, J., Reusch, N., Paclik, D., Baßler, K., Schlickeiser, S., Zhang, B., Krämer, B., Krammer, T., Brumhard, S., Bonaguro, L., et al. (2020). Severe COVID-19 is marked by a dysregulated myeloid cell compartment. Cell *182*, 1419–1440.e23.

11. Young, M.D., and Behjati, S. (2020). SoupX removes ambient RNA contamination from droplet-based single-cell RNA sequencing data. GigaScience *9*, giaa151. https://doi.org/10.1093/gigascience/giaa151.

12. McGinnis, C.S., Murrow, L.M., and Gartner, Z.J. (2019). DoubletFinder: doublet detection in single-cell RNA sequencing data using artificial nearest neighbors. Cell Syst. *8*, 329–337.e4.

13. Raudvere, U., Kolberg, L., Kuzmin, I., Arak, T., Adler, P., Peterson, H., and Vilo, J. (2019). g:Profiler: a web server for functional enrichment analysis and conversions of gene lists (2019 update). Nucleic Acids Res. *47*, W191–W198. https://doi.org/10.1093/nar/gkz369.

14. Wickham, H. (2016). Data analysis. In ggplot2 (Springer), pp. 189–201.

15. Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., et al. (2019). Welcome to the tidyverse. J. Open Source Softw. *4*, 1686.

16. Hwang, B., Lee, J.H., and Bang, D. (2018). Single-cell RNA sequencing technologies and bioinformatics pipelines. Exp. Mol. Med. *50*, 1–14. https://doi.org/10.1038/s12276-018-0071-8.

17. Luecken, M.D., and Theis, F.J. (2019). Current best practices in single-cell RNA-seq analysis: a tutorial. Mol. Syst. Biol. *15*, e8746. https://doi.org/10.15252/msb.20188746.

18. Andrews, T.S., Kiselev, V.Y., McCarthy, D., and Hemberg, M. (2021). Tutorial: guidelines for the computational analysis of single-cell RNA sequencing data. Nat. Protoc. *16*, 1–9.

19. Hafemeister, C., and Satija, R. (2019). Normalization and variance stabilization of single-cell RNA-seq data using regularized

negative binomial regression. Genome Biol. *20*, 296–315.

20. Aran, D., Looney, A.P., Liu, L., Wu, E., Fong, V., Hsu, A., Chak, S., Naikawadi, R.P., Wolters, P.J., Abate, A.R., et al. (2019). Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. Nat. Immunol. *20*, 163–172. https://doi.org/10.1038/s41590-018-0276-y.

21. Hao, Y., Hao, S., Andersen-Nissen, E., Mauck, W.M., 3rd, Zheng, S., Butler, A., Lee, M.J., Wilk, A.J., Darby, C., Zager, M., et al. (2021). Integrated analysis of multimodal single-cell data. Cell *184*, 3573–3587.e29. https://doi.org/10.1016/j.cell.2021.04.048.

22. Ianevski, A., Giri, A.K., and Aittokallio, T. (2022). Fully-automated and ultra-fast cell-type identification using specific marker combinations from single-cell transcriptomic data. Nat. Commun. *13*, 1246. https://doi.org/10.1038/s41467-022-28803-w.

23. Wu, T., Hu, E., Xu, S., Chen, M., Guo, P., Dai, Z., Feng, T., Zhou, L., Tang, W., Zhan, L., et al. (2021). clusterProfiler 4.0: a universal enrichment tool for interpreting omics data. Innovation *2*, 100141.

24. Gene Ontology Consortium (2021). The Gene Ontology resource: enriching a GOld mine. Nucleic Acids Res. *49*, D325–D334.

25. Dennis, G., Sherman, B.T., Hosack, D.A., Yang, J., Gao, W., Lane, H.C., and Lempicki, R.A. (2003). DAVID: database for annotation, visualization, and integrated discovery. Genome Biol. *4*, P3–P11.

26. Kanehisa, M., Furumichi, M., Tanabe, M., Sato, Y., and Morishima, K. (2017). KEGG: new perspectives on genomes, pathways, diseases and drugs. Nucleic Acids Res. *45*, D353–D361. https://doi.org/10.1093/nar/gkw1092.

27. Jassal, B., Matthews, L., Viteri, G., Gong, C., Lorente, P., Fabregat, A., Sidiropoulos, K., Cook, J., Gillespie, M., Haw, R., Loney, F., May, B., Milacic, M., Rothfels, K., Sevilla, C., Shamovsky, V., Shorser, S., Varusai, T., Weiser, J., Wu, G., Stein, L., Hermjakob, H., and D'Eustachio, P. (2020). The reactome pathway knowledgebase. Nucleic Acids Res. *48*, D498–D503. https://doi.org/10.1093/nar/gkz1031.

28. Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.R., and Raychaudhuri, S. (2019). Fast, sensitive and accurate integration of single-cell data with Harmony. Nat. Methods *16*, 1289–1296. https://doi.org/10.1038/s41592-019-0619-0.

29. Haghverdi, L., Lun, A.T.L., Morgan, M.D., and Marioni, J.C. (2018). Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. Nat. Biotechnol. *36*, 421–427. https://doi.org/10.1038/nbt.4091.

30. Butler, A., Hoffman, P., Smibert, P., Papalexi, E., and Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat. Biotechnol. *36*, 411–420.