Analysis of PBMC Transcriptome in COVID-19 Symptomatic vs. Uninfected Patients

Yuchen Sun

2023-04-22

Contents

Introduction

The coronavirus disease-2019 (COVID-19), caused by the severe acute respiratory syndrome coronavirus (SARS-CoV-2), was first detected in Wuhan, China, back in December 2019, and developed to a critical global pandemic over the past three years. There has been many studies towards the nature of SRAS-CoV-2 as well as the mechanism of viral invasion, so that more effective treatments can be developed. It is faily understood that the virus invades human cells by attaching the Spike structure to the ACE2 receptor on human cells and further hijacking the host cells' ability of replication and transcription[1]. However, the more complicated situation which is not fully studied at this point is the gene regulation behind the viral invasion, and there is not yet a comprehensive answer to the question regarding the difference of gene expression in the healthy people versus that in a COVID-19 patient.

Given that human peripheral blood mononuclear cells (PBMC) has crucial function in human immune system, it is very valuable to study the PMBC transcriptomic difference in healthy versus infected people and to extract potentially differentially expressed genes, as these genes might provide more insight into the underlying pathway of COVID-19 associated severe symptoms as well as shedding light on better potential targets for pharmaceutical treatments against COVID-19. Hence, this project collects human PBMC transcriptome dataset and conducted differential gene analysis, focusing on significant differential expression of genes related to crucial cytokine storm pathways such as NF-κB pathway.

Methods

Data download

The data used in this project is linked to this paper[1]. The data description metadata is retrieved from the SRA Run Selector, whereas the actual sequencing data with `.fastq.gz` format is retrieved from the ENA. The original data contains five groups: uninfected (22), recovering (15), symptomatic (13), re-detectable positive (RP) (12), and asymptomatic (8). However, in this project, only data from uninfected and symptomatic patients is used. A bash script is used to download data by first retrieving the the corresponding sample SRR ID, which is then used to create an `ftp` connection with the specific ENA web address hosting the paired-end read files of that sample, and using `wget` to download the files. Each read file is automatically put into the corresponding directory (uninfected or symptomatic) after the download is complete.

Read preprocessing

After downloading all read files, a FastQC run is used to determine the quality of the raw reads. It was discovered that most of the paired-end reads has high adapter content that failed the quality check. Hence, Trim-Galore was used to trim the raw reads by the command

```
trim_galore --illumina --paired \
            --output_dir $trim_out_dir \
            --stringency 13 $file $file2
```

Notice that the parameter `--illumina` and `--paired` was added to the command given that the original raw reads are produced by illumina paired-end sequencing, and that the parameter `--stringency 13` is set as it is a length threshold more likely to prevent potentially wrong adaptor trimming due to small overlaps (this threshold was chosen according to previous runs of Trim-Galore on the same data). `$trim_out_dir` represents the directory that holds the trimmed reads output of Trim-Galore, whereas `$file` and `file2` represents the two paired-end reads. After running Trim-Galore, another FastQC run is used to determine the quality of the trimmed reads, which are all proven to have low adatper content and are available for further processing. All scripts for this part of the preprocessing is put into a bash script to enable automatic FastQC run before and after Trim-Galore trimming.

Sequence alignment

In this project, STAR alignment tool is used to align the trimmed reads to the genome. Before running actual alignments, an index for STAR alignment is created by the command

```
STAR --runMode genomeGenerate \
     --runThreadN 1 \
     --genomeDir $ref_dir \
     --genomeFastaFiles $genome_seq \
     --sjdbGTFfile $genome_annot \
     --sjdbOverhang 149
```

Notice that the parameter `--sjdbOverhang 149` was added to the command instead of the default value (99) in order to reflect the fact that the original raw reads are produced by 2 × 150bp paired-end sequencing protocal, given that the best value for this parameter is usually the sequence length - 1. `--genomeDir $ref_dir` represents the directory that holds the STAR index output, where as `--genomeFastaFiles $genome_seq` and `--sjdbGTFfile $genome_annot` represents the actual genome sequence file (in `FASTA` format) and the genome annotation file (in `GTF` format), respectively. Human genome `hg38` is used in this project, and a bash script is used to download both genome files (`hg38.fa.gz` and `hg38.ncbiRefSeq.gtf.gz`) from the UCSC Genome Data website using `wget`. The script also contains the commands to execute the STAR index creation after verifying that the required genome files are downloaded and gzipped in the correct directory.

After creating the index, the STAR alignment is performed by the command

```
STAR --runMode alignReads \
     --runThreadN 1 \
     --genomeDir $ref_dir \
     --readFilesIn $file $file2 \
     --readFilesCommand zcat \
     --outFileNamePrefix $out_dir \
     --outSAMtype BAM SortedByCoordinate
```

Notice that parameter for file input `--readFilesIn` is provided with two files (`$file` and `$file2`) in order to simultaneously align the two paired-end reads from each sample. `--genomeDir $ref_dir` represents the directory that holds the STAR index just created, whereas `--outFileNamePrefix $out_dir` represents the directory that holds the STAR alignment output files. `--readFilesCommand zcat` was also added in order to directly access read files in a gzipped format. Finally, `--outSAMtype BAM` is used to change the alignment output file to `BAM` format in order to save memory usage, and `SortedByCoordinate` parameter was added to force the alignment results to be sorted by their coordinate in the reference genome.

After performing STAR alignment, `samtools` was also used to index the alignment output (in `BAM` format) by the command

```
samtools index $out_file
```

A separate bash script containing the STAR alignment commands and the samtools commands is used to allow automatic alignment, sorting, and indexing procedure for each sample used in this project.

With these steps, the sequence alignment in this project would produce an indexed and sorted by coordinate `BAM` file for each sample as the alignemnt output.

Quality control

This project conducts FastQC run as one method of quality control, as introduced in the previous section (see Read preprocessing), in order to monitor the quality of the raw reads and the trimmed reads after performing Trim-Galore. FastQC is performed by the command

```
fastqc $file --noextract --outdir $fastqc_out_dir
```

On the test run of the read files from 12 samples, the raw reads of these samples all fails the adapter content check by FastQC. However, after performing Trim-Galore, all 12 trimmed reads has low adapter content that passes the adapter content test. This continues to be true for all trimmed reads from other samples after performing Trim-Galore. However, it is worth noticing from the FastQC result that both of the paired-end reads of sample SRR15058644 contains higher overrepresented sequences level, causing a warning to be raised.

This project also conducts QoRTs run as another qualtity control in order to monitor the quality of the alignment results. QoRTs is performed by the command

```
qorts -Xmx16G QC --generatePlots \
      --maxPhredScore 45 --maxReadLength 150 \
      $file $gtffile $outdir
```

The plots for QoRTs are generated by running first a bash script that creates the decoder for plotting, then a bash script with the command above to perform QoRTs, and finally an R code that reads in the decoder as well as the QoRTs result, returning the plots in a PDF format. It was discovered that the cumulative gene assignment diversity appears to be of the correct trend, and that the gene-body coverage is roughly uniform across the gene body for all samples. Some concerns are raised from examing the QoRTs result: one sample from the symptomatic group contains high percentage of reads from MT chromosome, and all samples from symptomatic group seems to have higher percentage of intron coverage. The extra gene content from the MT origin can be processed in loading the preprocessing the feature counts (which can remove gene counts from mitochondrial genes), but it is admittedly unclear whether the higher intron coverage of the symptomatic group is due to genetic reasons specific to the group or some type of contamination.

Feature counts

The feature counts for this project is created by the `featureCounts` with the command

```
featureCounts -p --countReadPairs -a $gtf -t exon -g gene_id -o $fc_dir $bam_dir*/*.bam
```

Notice that the parameter `--countReadPairs` was added in order to specify the paired-end nature of the reads. A bash script containing the command above is used to automatically run feature counts for all alignment results (in `BAM` format) and store the output feature counts information in the corresponding directory. The feature counts is then read into `R` with the DESeq2 object to perform preprocessing and quality control steps.

```r
# load read counts and format the data table
readcounts <- read.table(paste0(wd, "feature_counts.txt"), header = T)
orig_names <- names(readcounts)
# preserve the SRR ID only
names(readcounts) <- gsub(".*(symptomatic|uninfected).(SRR)([0-9]{8}).*",
    "\\2\\3", orig_names)
# generate the countData and colData for DESeq2 from the
# formatted read count
row.names(readcounts) <- make.names(readcounts$Geneid)
readcounts <- readcounts[, -c(1:6)]
sample_info <- data.frame(condition = gsub(".*(symptomatic|uninfected).*",
    "\\1", orig_names)[-c(1:6)], row.names = names(readcounts))
```

```r
# make the DESeq dataset from formatted countData and
# colData
DESeq.ds <- DESeqDataSetFromMatrix(countData = as.matrix(readcounts),
    colData = sample_info, design = ~condition)
# remove samples with poor reads
keep_samples <- !(colnames(DESeq.ds) == "SRR15058644")
DESeq.ds <- DESeq.ds[, keep_samples]
# remove genes with 0 reads
keep_genes <- rowSums(counts(DESeq.ds)) > 0
DESeq.ds <- DESeq.ds[keep_genes, ]
# remove mitochondrial genes
nonmt_genes <- !(grepl("^MT", rownames(DESeq.ds)))
DESeq.ds <- DESeq.ds[nonmt_genes]

DESeq.ds
```

```
## class: DESeqDataSet
## dim: 39934 15
## metadata(1): version
## assays(1): counts
## rownames(39934): TRNP TRNT ... WASH7P DDX11L1
## rowData names(0):
## colnames(15): SRR15058638 SRR15058641 ... SRR15058634 SRR15058636
## colData names(1): condition
```

In preprocessing and quality control steps, after the feature counts table is read into R, only the SRR ID is kept as the name for each sample. However, a table containing the information of each sample and its corresponding group (either uninfected or symptomatic) is kept. The genes are also renamed by the gene ID. After the read counts table is added to a DESeq2 dataset object, samples with poor reads (SRR15058644 due to higher overrepresented sequence level), mitochondrial genes, and genes with 0 reads are removed from the dataset.

```r
# estimate size factors
DESeq.ds <- estimateSizeFactors(DESeq.ds)
# log normalize the dataset
assay(DESeq.ds, "log.norm.counts") <- log2(counts(DESeq.ds, normalized = T) +
    1)

# perform r log on the dataset
DESeq.rlog <- rlog(DESeq.ds, blind = T)
```

The dataset is then normalized to eliminate the non-biological difference on the dataset, adding the log normalized counts to the dataset as a separate assay. In order to reduce the dependence of the variance on the mean, an `rlog` is also performed on the dataset, which then generates a new object for the dataset.

Differential gene analysis

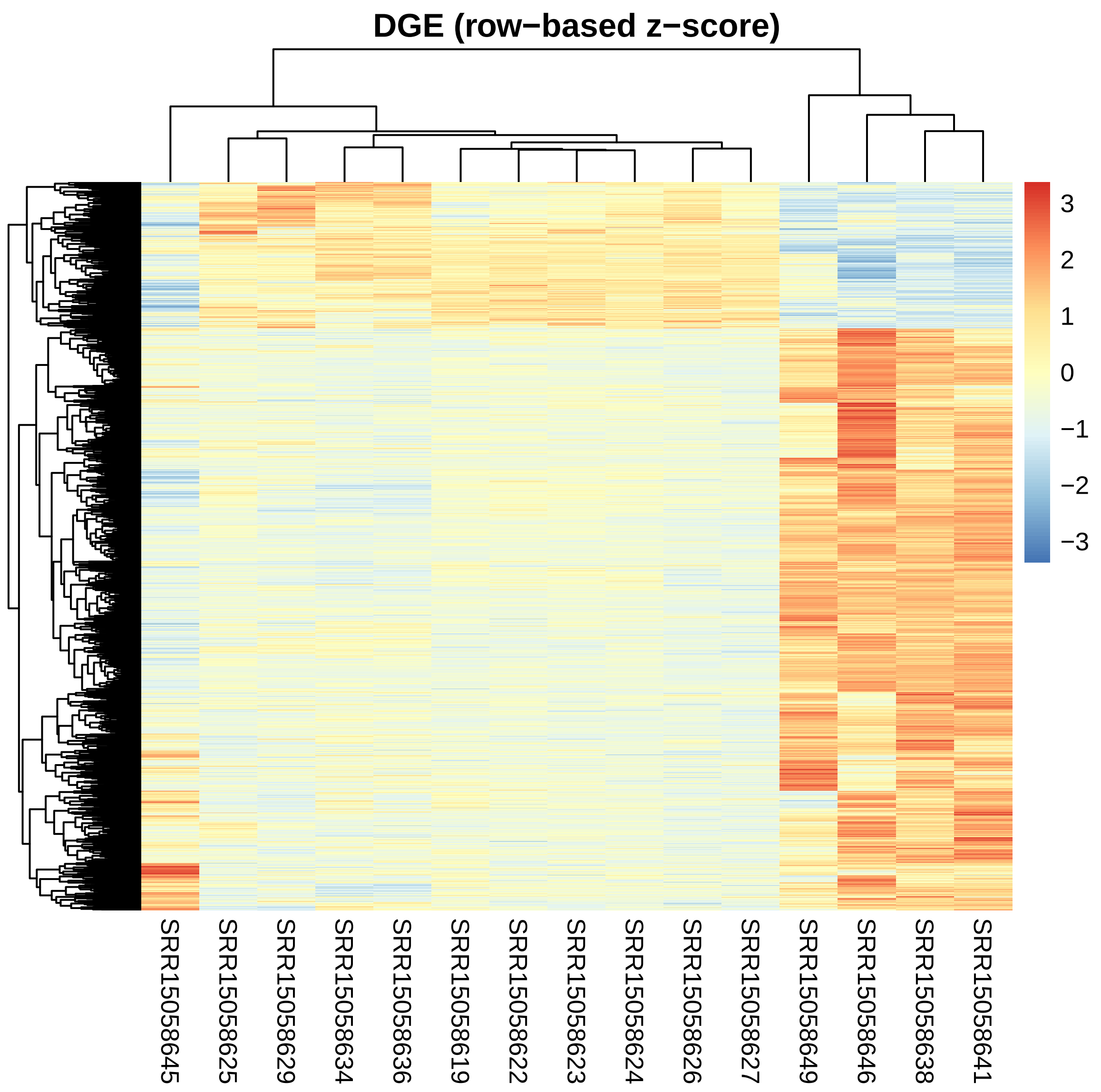
In this project, the differential gene analysis is performed by using DESeq2 function `DESeq()` on the pre-processed and log normalized dataset. The results of differential gene analysis is then adjusted for multiple hypothesis testing correction and cleaned by removing genes with an `NA` in either p-value or adjusted p-value. The alpha for significance is chosen to be 0.05, so that a gene is deemed as differentially expressed only when it has an adjusted p-value of less than 0.05. In order to cope with the unreliable fold change results for genes with noisy expression values, and to better integrate the solution for plotting, the log FC shrink is also performed on the dataset to generate a more valid result.

```r
# perform differential expression analysis
DESeq.ds$condition %<>%
    relevel(ref = "uninfected")
DESeq.ds %<>%
    DESeq()
# retrieve raw and sorted DGE results
DGE.results.raw <- na.omit(results(DESeq.ds, independentFiltering = T,
    alpha = 0.05))
DGE.results.raw.sorted <- DGE.results.raw %>%
    order(.$padj)[]

# retrieve lfcShrink-ed and sorted DGE results
DGE.results.shrnk <- lfcShrink(DESeq.ds, coef = 2, type = "apeglm")
# remove genes associated with RNA component and
# pseudogenes
clean_gene <- !(grepl("(RNA|LOC|SNOR)", rownames(DGE.results.shrnk)))
DGE.results.clean <- na.omit(DGE.results.shrnk[clean_gene, ])
DGE.results.clean.sorted <- DGE.results.clean %>%
    order(.$padj)[]

# subset significant genes
DGE.genes <- subset(DGE.results.clean.sorted, padj < 0.05)
DGE.genes <- subset(DGE.genes, abs(log2FoldChange) > 1)
DGE.genes.names <- rownames(DGE.genes)
```

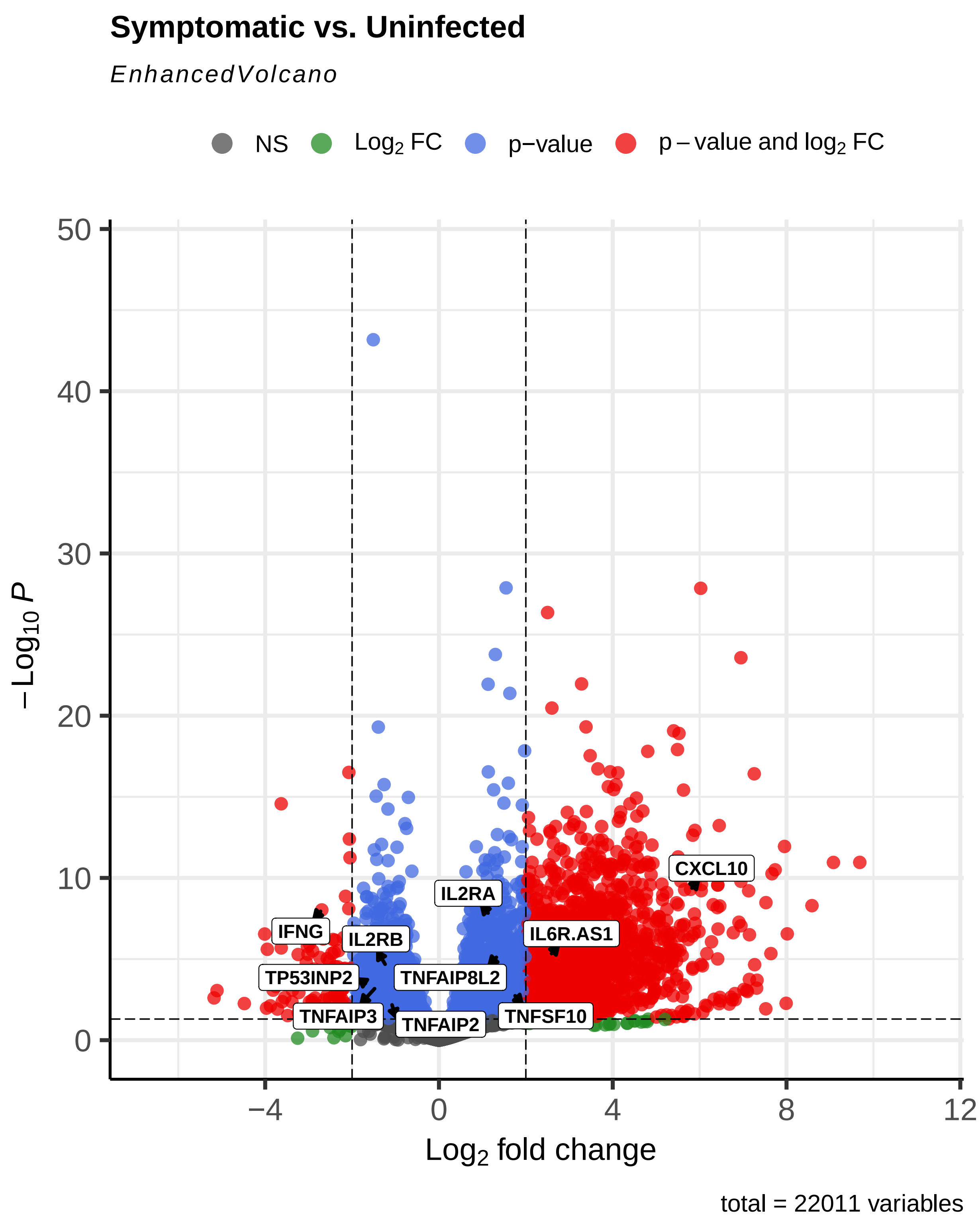
Results

Significant differentially expressed genes

```r
# generate heat maps of differentially expressed genes
rlog.dge <- DESeq.rlog[DGE.genes.names, ] %>%
    assay
pheatmap(rlog.dge, scale = "row", show_rownames = F, main = "DGE (row-based z-score)")
```

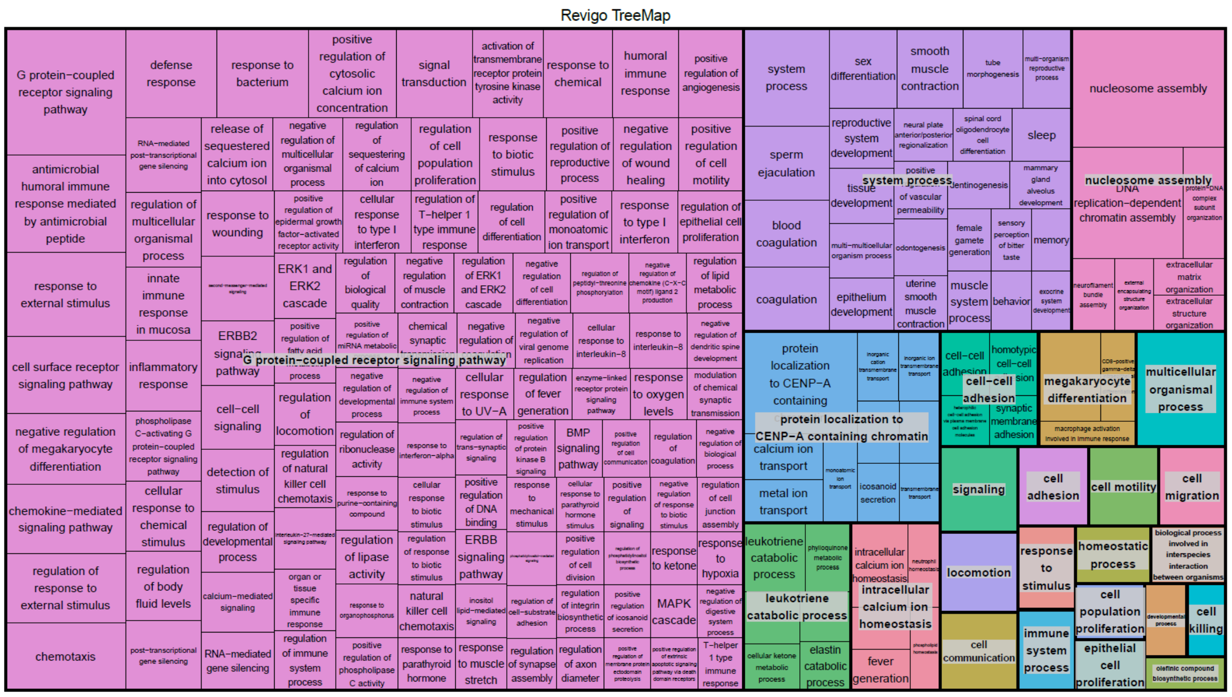


```
# select key genes
key_genes <- "^(IL2R|IL6R|TNFAIP|CCL2&|CXCL10|TNFSF10|TP53INP|TAB2|IFNG)"
DGE.genes.names.key <- DGE.genes.names[grep(key_genes, DGE.genes.names)]
# generate volcano plot of differentially expressed genes
EnhancedVolcano(DGE.results.clean, lab = rownames(DGE.results.clean),
    selectLab = DGE.genes.names.key, boxedLabels = T, labCol = "black",
    labFace = "bold", drawConnectors = T, widthConnectors = 1,
    colConnectors = "black", pointSize = 3, labSize = 4, colAlpha = 0.75,
    x = "log2FoldChange", y = "padj", pCutoff = 0.05, FCcutoff = 2,
    title = "Symptomatic vs. Uninfected")
```


Symptomatic vs. Uninfected
EnhancedVolcano
NS
Log$_2$ FC
p−value
p − value and log$_2$ FC
50
40
30
20
10
0
−Log$_{10}$ P
−4
0
4
8
12
Log$_2$ fold change
CXCL10
IL2RA
IFNG
IL2RB
IL6R.AS1
TP53INP2
TNFAIP8L2
TNFAIP3
TNFAIP2
TNFSF10
total = 22011 variables

GO term analysis

```
# construct named vector of 0 and 1 for DEG
gene.vector <- row.names(DGE.results.clean) %in% DGE.genes.names %>%
    as.integer
names(gene.vector) <- row.names(DGE.results.clean)
# quantify length bias
pwf <- nullp(gene.vector, "hg19", "geneSymbol", plot.fit = F)
# test for enrichment of GO terms
GO.wall <- goseq(pwf, "hg19", "geneSymbol")
# retrieve GO categories of each gene
go_genes <- getgo(rownames(DGE.results.clean), "hg19", "geneSymbol") %>%
    stack
sig_GOs <- subset(GO.wall, over_represented_pvalue < 0.01)
# export file for REVIGO plotting
write.table(sig_GOs[, c("category", "over_represented_pvalue")],
    file = "final_GOterms_goseq.txt", quote = F, row.names = F,
    col.names = F)
# include Revigo treeplot R script
source("final_REVIGO_treeplot.R")
```



Discussion

IL2, IL6, IL7, IL10, GSCF, IP10, MCP1, MIP1A, and TNFA CCL2, CXCL2, CCL8, CXCL1, IL33, CCL3L1 CXCL10, TNFSF10, TIMP1, C5, IL18, AREG, NRG1, IL10 TP53

TRAF2, TRAF5, TRAF6, TRADD, FADD, CARD11, TIRAP, TRAM, TRIF, RIP1 P50, P65, TAK1, TAB2/3, IKK have IL2R (A, B, G, BG internalized and degraded, A recycled) - RA upregulate, RB,RG downregulate IL6R upregulate TNFAIP (8 positive regulation of apoptotic process, 2,3 inhibit NFKB, involved in inflammatory response), 8 upregulate, 2,3 downregulate CCL2 upregulate CXCL10 upregulate (cytokine storm regulator) TNFSF10 upregulate (induce apoptosis) IL18 downregulate ?? TP53 upregulate TAB2 upregulate (induce NFKB)

Code Availability

All scripts related to this project, including bash scripts, R scripts, R Markdown scripts, etc., as well as some intermediate checkpoints results, can all be found in the public GitHub repository kevinsunofficial/angsd_project. The contents are also available on the Weill Cornell Aphrodite server following directory `/home/yus4008/cmpb5004/project/angsd_project/`. The data availablity is explained in the Methods section of this report (see Data downloading).

Reference

1. Amirfakhryan, H., & Safari, F. (2021). Outbreak of SARS-CoV2: Pathogenesis of infection and cardiovascular involvement. *Hellenic journal of cardiology : HJC = Hellenike kardiologike epitheorese, 62*(1), 13-23. https://doi.org/10.1016/j.hjc.2020.05.007

2. Zhang, J., Lin, D., Li, K., Ding, X., Li, L., Liu, Y., Liu, D., Lin, J., Teng, X., Li, Y., Liu, M., Shen, J., Wang, X., He, D., Shi, Y., Wang, D., & Xu, J. (2021). Transcriptome analysis of peripheral blood mononuclear cells reveals distinct immune response in asymptomatic and re-detectable positive COVID-19 patients. *Frontiers in Immunology, 12.* https://doi.org/10.3389/fimmu.2021.716075