**R code for analysis (****Running codes in the RStudio Version 1.3)**

Multiple-organ failure for COVID-19

# These R packages need to be installed before running the code

library(“WGCNA”)

library(“Scatterplot3d”)

library(“rgl”)

library(“vegan”)

library(“GOplot”)

library(“ggplot”)

library(“Combat”)

library(“Seurat)

library(“dplyr”)

library(CellChat)

library(ggalluvial)

library(monocle3)

# Set the working directory for reading and saving the data

setwd("/Users/Destination/R analysis")

#=========================================================  
# WGCNA calculation

Data preparation #=========================================================

#read the sample data (GSE162113)

Options(stringsAsFactors = FALSE);

expro=read.table('raw\_counts.txt', sep = '\t', head=T, row.names = 1)

#Select gene expression variance greater than 90% of the whole genome

m.vars=apply(expro,1,var)

expro.upper=expro[which(m.vars>quantile(m.vars, probs = seq(0,1,0.1)[10])),]

# Convert data type to matrix

datExpr0=as.data.frame(expro.upper)

#Evaluate whether the matrix information is qualified

gsg = goodSamplesGenes(datExpr0, verbose = 3)

gsg$allOK

#optional: When GSG is not shown “allOK”

if (!gsg$allOK)  
{ if (sum(!gsg$goodGenes)>0)

printFlush(paste("Removing genes:", paste(names(datExpr0)[!

gsg$goodGenes], collapse = ", ")));

if (sum(!gsg$goodSamples)>0)

printFlush(paste("Removing samples:", paste(rownames(datExpr0)[!

gsg$goodSamples], collapse = ", ")));

# Remove the offending genes and samples from the data:

datExpr0 = datExpr0[gsg$goodSamples, gsg$goodGenes]

}

#=========================================================  
#Sample clustering and detect outliers #=========================================================

dim(datExpr0)

heartdata <- datExpr0[,1:8]

Kidneydata <- datExpr0[,9:15]

Lungdata <- datExpr0[,16:21]

Spleendata <- datExpr0[,22:29]

#cluster heart data

sampleTree = hclust(dist(t(heartdata)), method = "average")

sizeGrWindow(20,15)

#pdf(file = "HeartClustering.pdf", width = 20, height = 15)

par(cex = 1.5);

par(mar = c(0,4,2,0))

plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="",

cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5,lwd=4)

abline(h = 500000, col = "red",lwd = 4)

dev.off()

#cluster Kidney data

sampleTree = hclust(dist(t(Kidneydata)), method = "average")

sizeGrWindow(20,15)

#pdf(file = "HeartClustering.pdf", width = 20, height = 15)

par(cex = 1.5);

par(mar = c(0,4,2,0))

plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="",

cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5,lwd=4)

abline(h = 400000, col = "red",lwd = 4)

#dev.off()

#cluster Lung data

sampleTreeLung = hclust(dist(t(Lungdata)), method = "average")

sizeGrWindow(20,15)

#pdf(file = "HeartClustering.pdf", width = 20, height = 15)

par(cex = 1.5);

par(mar = c(0,4,2,0))

plot(sampleTreeLung, main = "Sample clustering to detect outliers", sub="", xlab="",

cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.5,lwd=4)

abline(h = 200000, col = "red",lwd = 4)

#dev.off()

#cluster Spleen data

sampleTree = hclust(dist(t(Spleendata)), method = "average")

sizeGrWindow(20,15)

#pdf(file = "HeartClustering.pdf", width = 20, height = 15)

par(cex = 1.5);

par(mar = c(0,4,2,0))

plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="",

cex.lab = 1.5, cex.axis = 1.5, cex.main = 2,lwd=4)

abline(h = 250000, col = "red",lwd = 4)

#dev.off()

# All data datExpr0

sampleTree = hclust(dist(t(datExpr0)), method = "average")

# Plot the sample tree: Open a graphic output window of size 12 by 9 inches

# The user should change the dimensions if the window is too large or too small.

sizeGrWindow(20,9)

pdf (file = "AllClustering.pdf", width = 20, height = 15)

par(cex = 1);

par(mar = c(0,4,2,0))

plot(sampleTree, main = "Sample clustering to detect outliers", sub="", xlab="",

cex.lab = 1, cex.axis = 1, cex.main = 1,lwd=1)

#=========================================================

# Choose a set of soft-thresholding powers #=========================================================

RdatExpr0 <- t(expro)

allowWGCNAThreads()

powers1=c(seq(1,11,by=1),seq(12,20,by=2))

RpowerTable=pickSoftThreshold(RdatExpr0, powerVector=powers1)[[2]]

#Plot the result, including Soft-thresholding powers and Mean Connectivity

cex1=1

par(mfrow=c(1,2),cex=1.4)

plot(RpowerTable[,1], -sign(RpowerTable[,3])\*RpowerTable[,2],xlab="

Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed R^2",type="n",ylim=c(-1,1))

text(RpowerTable[,1], -sign(RpowerTable[,3])\*RpowerTable[,2], labels=powers1,cex=cex1,col="red")

abline(h=0.8,col="red")

plot(RpowerTable[,1], RpowerTable[,5],xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n")

text(RpowerTable[,1], RpowerTable[,5], labels=powers1, cex=cex1,col="red")

dev.off()

#=========================================================

# Here we define the adjacency matrix using soft thresholding with beta=9

#=========================================================

beta1=9

Connectivity=softConnectivity(RdatExpr0,power=beta1)

pdf("scalefree= softConnectivity.pdf",15,10)

par(mfrow=c(1,1),cex=1.5,lwd=1)

scaleFreePlot(Connectivity, main=paste("soft threshold, power=",beta1), truncated=T,cex=1, col="red",pch=16,ylim=c(-2.5,0))

dev.off()

#=========================================================

# Convert to adjacency matrix (power)

#=========================================================

Filterdat <- t(datExpr0)

adjacency = adjacency(Filterdat, power = 9)

#=========================================================

# Convert to topological matrix and calculate dissimilarity dissTOM

#=========================================================

TOM = TOMsimilarity(adjacency)

dissTOM = 1-TOM

#=========================================================

# Gene clustering on TOM-based dissimilarity

#=========================================================

geneTree = hclust(as.dist(dissTOM), method = "average")

# Plot the resulting clustering tree (dendrogram)

sizeGrWindow(12,12)

plot(geneTree, xlab="", sub="", main = "Gene clustering on TOM-based dissimilarity",labels = FALSE, hang = 0.04);

#=========================================================

# Gene dendrogram and module colors

#=========================================================

# At least 100 genes in a module

minModuleSize = 100

dynamicMods = cutreeDynamic(dendro = geneTree, distM = dissTOM,deepSplit = 2, pamRespectsDendro =FALSE, minClusterSize = minModuleSize)

table(dynamicMods)

dynamicColors = labels2colors(dynamicMods)

table(dynamicColors)

sizeGrWindow(8,12)

plotDendroAndColors(geneTree, dynamicColors, "Dynamic Tree Cut",

dendroLabels = FALSE, hang = 0.03,

addGuide = TRUE, guideHang = 0.05,

main = "Gene dendrogram and module colors")

#=========================================================

# Calculate the eigengene, perform hierarchical clustering on the modules, and

merge the more similar modules

#=========================================================

# Calculate eigengenes

Filterdat <- t(datExpr0)

MEList = moduleEigengenes(Filterdat, colors = dynamicColors)

MEs = MEList$eigengenes

# Calculate dissimilarity of module eigengenes

MEDiss = 1-cor(MEs)

# Cluster module eigengenes

METree = hclust(as.dist(MEDiss), method = "average")

# Plot the result

sizeGrWindow(7, 6)

par(cex = 1.6, lwd=4)

plot(METree, main = "Clustering of module eigengenes",xlab = "", sub = "")

#=========================================================

# Set a cut-off height = 0.2 to merge similar modules on the cluster tree

#=========================================================

MEDissThres = 0.2

abline(h=MEDissThres, col = "red")

merge = mergeCloseModules(Filterdat, dynamicColors, cutHeight = MEDissThres, verbose = 3)

mergedColors = merge$colors

#=========================================================

# Draw a new cluster tree and module diagram with the cut module

#=========================================================

sizeGrWindow(12, 9)

plotDendroAndColors(geneTree, cbind(dynamicColors, mergedColors),

c("Dynamic Tree Cut", "Merged dynamic"),

dendroLabels = FALSE, hang = 0.03,

addGuide = TRUE, guideHang = 0.05,

cex.colorLabels = 1,cex.dendroLabels =3,cex.rowText =1 )

# Calculate the eigengenes of the merged module

MEList2 = moduleEigengenes(Filterdat, colors = mergedColors)

MEs2 <- MEList2$eigengenes

MEDiss2 = 1-cor(MEs2)

METree2 = hclust(as.dist(MEDiss2), method = "average")

# Plot the cluster tree with merged module

sizeGrWindow(7, 6)

par(cex = 1.6, lwd=4)

plot(METree2, main = "Clustering of module eigengenes",xlab = "", sub = "")

#=========================================================

# Draw a heat map based on topological overlap

#=========================================================

# Calculate the dissTOM among genes

nGenes <- ncol(t(Filterdat))

nSamples = nrow(t(Filterdat))

softPower = 9

adjacency = adjacency(Filterdat, power = 9)

TOM = TOMsimilarity(adjacency)

dissTOM = 1-TOM

adjacency2 = adjacency(Filterdat, power = 9)

TOMsimilarity(adjacency2)

dissTOM = 1-TOMsimilarity(adjacency2)

plotTOM = dissTOM^9

diag(plotTOM) = NA

geneTree = hclust(as.dist(dissTOM), method = "average")

moduleColors = mergedColors

sizeGrWindow(9,9)

TOMplot(plotTOM, geneTree, mergedColors, main = "Network heatmap plot, all genes")

#=========================================================

# Save the result of values of module Eigengenes

#=========================================================

MEList2 = moduleEigengenes(Filterdat, colors = mergedColors)[[1]]

MEs2 <- MEList2$eigengenes

colors2 <-as.character(mergedColors)

datKME<-signedKME(Filterdat, MEList2)

geneInfo0 <-data.frame(geneSymbol=rownames(t(Filterdat)),moduleColors=NEWcolor, datKME)

#=========================================================

# Geometric data shown in 3D scattering

#=========================================================

cmd1=cmdscale(as.dist(dissTOM),3)

pairs(cmd1, col=as.character(moduleColors), main="MDS plot",pch=16)

par(mfrow=c(1,1), mar=c(4,3,2,3)+0.1,cex=0.85,pch=16,lwd=3)

s3d <- scatterplot3d(cmd1,color=moduleColors,angle=210,xlab="Scaling Axis 1", ylab="Scaling Axis 2", zlab="Scaling Axis 3",type = "p")

my.lm <- lm(cmd1[,3] ~ cmd1[,1] + cmd1[,2])

s3d$plane3d(my.lm,col="blue4")

#=========================================================

# Pairwise illustration of gene module correlation

#=========================================================

pairs(datKME[1:500,],

panel= function(x,y){points(x,y,col =

c("black", "blue", "brown","magenta","purple","turquoise"),pch=16,cex=1.4)

abline(lm(y~x), col='black',lwd=2.5)

text(0.5,0.8,labels = paste('R2=',round((cor(x,y))^2,2)),

col='red',cex=1)})

#=========================================================

# Pearson’s R square and P value between modules

#=========================================================

PearsonsR <-signif(cor(MEList2, use="p")

R square­<- PearsonsR^2

nSamples=nrow(datKME)

PearsonsP <-corPvalueStudent(modul,nSamples)

#=========================================================

**# Single-cell analysis**

#=========================================================

# Read10X data from GSE165080

scRNA.data <- Read10X("10X")

dim(scRNA.data)

#=========================================================

#Read COVID-19 Single-cell RNA-seq only

#=========================================================

scRNA.data <-scRNA.data[,1:181562]

#Creat Seurat object

scRNA <- CreateSeuratObject(counts = scRNA.data,

min.cells = 3,

min.features = 200,

project ="scRNA")

#=========================================================

#Quality check

#=========================================================

HB.ref <-c("HBA1","HBA2","HBB","HBD","HBE1","HBG1","HBG2","HBM","HBQ1","HBZ")

rowSums (scRNA.data[HB.ref,])

HB.genes <- intersect(HB.ref, rownames(scRNA))

HB.genes

# Calculate the percentage of each feature set, hemoglobin, mitochondria, ribosomes

scRNA[["percent.HB"]] <- PercentageFeatureSet(scRNA, features = HB.genes)

scRNA[["percent.MT"]] <- PercentageFeatureSet(scRNA, pattern = "^MT-")

scRNA[["percent.Ribosome"]] <- PercentageFeatureSet(scRNA, pattern = "^RP[SL]")

#=========================================================

# Plot the percentage of each feature set in each cell

#=========================================================

pdf('1\_feature\_UMI\_MT\_HB\_Ribosome.pdf', width = 13, height = 6)

VlnPlot(scRNA, features = c("nFeature\_RNA", "nCount\_RNA", "percent.HB", "percent.MT", "percent.Ribosome"), ncol = 5)

dev.off()

#=========================================================

#Fillter cells and data normalization

#=========================================================

scRNA <- subset(scRNA, subset = nFeature\_RNA > 200 & nFeature\_RNA < 2500 & percent.MT < 20 & percent.HB< 5 & percent.Ribosome< 20)

scRNA <- NormalizeData(scRNA, normalization.method = "LogNormalize", scale.factor = 10000)

#=========================================================

# Select genes with high variation

#=========================================================

scRNA <- FindVariableFeatures(scRNA, selection.method = "vst", nfeatures = 5000)

#=========================================================

Scale transformation using all genes

#=========================================================

all.genes <- rownames(scRNA)

scRNA <- ScaleData(scRNA, features = all.genes)

#=========================================================

Identification of differentially expressed genes

#=========================================================

scRNA.markers <- FindAllMarkers(scRNA, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)

#=========================================================

Dotplot, ridgeplot, feature plot, and heatmap

#=========================================================

DotPlot(scRNA,features = c("PTPRC", "JAK2", "TYK2", "LAT", "IKZF1",

"STAT6", "CPSF6", "CBL", "SMC4", “TRAF3IP3”)

RidgePlot(scRNA, features = c("PTPRC", "JAK2", "TYK2", "LAT", "IKZF1", "STAT6", "CPSF6", "CBL", "SMC4", “TRAF3IP3”)

FeaturePlot(scRNA, features = c("PTPRC", "JAK2", "TYK2", "LAT", "IKZF1", "STAT6", "CPSF6", "CBL", "SMC4", “TRAF3IP3”)

DoHeatmap(scRNA, features = c("PTPRC", "JAK2", "TYK2", "LAT", "IKZF1", "STAT6", "CPSF6", "CBL", "SMC4", “TRAF3IP3”)) + NoLegend()

#=========================================================

Name each cell subpopulation according to the annotation results

#=========================================================

ew.cluster.ids <- c("CD14+ monocytes","Natural killer cell","Dendritic cell",

"Macrophage","CD4+ T cell","FCGR3A+ monocytes", "CD8+ T cell", "B cell","Memory T cell","Megakaryocyte","Platelets","Neutrophil cell")

names(new.cluster.ids) <- levels(scRNA)

scRNA <- RenameIdents(scRNA, new.cluster.ids)

Idents(scRNA)

#=========================================================

Umap and dotplot after annotation

#=========================================================

DimPlot(scRNA, split.by="cellType",label.size=3,reduction = "umap", label = T, pt.size = 0.5) + NoLegend()

genesToplot <- c("PTPRC", "JAK2", "TYK2", "LAT", "IKZF1", "STAT6", "CPSF6", "CBL", "SMC4", “TRAF3IP3”)

DotPlot(scRNA, features = genesToplot) + RotatedAxis()

#=========================================================

Cellchat analysis

#=========================================================

cellchat <- createCellChat(scRNA@assays$RNA@data)

scRNA$cellType <- Idents(scRNA)

meta <- data.frame(cellType = scRNA$cellType, row.names = Cells(scRNA))

cellchat <- addMeta(cellchat, meta = meta, meta.name = "cellType")

cellchat <- setIdent(cellchat, ident.use = "cellType")

groupSize <- as.numeric(table(cellchat@idents))

CellChatDB <- CellChatDB.human

str(CellChatDB)

unique(CellChatDB$interaction$annotation)

CellChatDB.use <- subsetDB(CellChatDB, search = "Secreted Signaling")

cellchat@DB <- CellChatDB.use

cellchat <- subsetData(cellchat)

cellchat <- identifyOverExpressedGenes(cellchat)

cellchat <- identifyOverExpressedInteractions(cellchat)

cellchat <- projectData(cellchat, PPI.human)

cellchat <- computeCommunProb(cellchat)

#=========================================================

Show the important communication between signallings by Cellchat function

#=========================================================

cellchat@netP$pathways

levels(cellchat@idents)

netVisual\_aggregate(cellchat, signaling = pathways.show, layout = "hierarchy", vertex.receiver = vertex.receiver, vertex.weight = groupSize,vertex.label.cex = 2)

#=========================================================

Show the Receptor-ligand relationship among signalling by Cellchat function

#=========================================================

netVisual\_bubble(cellchat, sources.use = 1:5, targets.use = c(1:5), remove.isolate = FALSE, vertex.label.cex = 1.2)

#=========================================================

Monocle3 for pseudotime analysis

#=========================================================

seurat\_object <- scRNA

library("org.Hs.eg.db")

gene\_symbol <- as.list(org.Hs.egSYMBOL)

raw\_count\_data <- GetAssayData(seurat\_object, assay = "RNA", slot = "counts")

class(raw\_count\_data)

cells\_info <- seurat\_object@meta.data

gene\_name <- gene\_symbol[rownames(raw\_count\_data)]

gene\_name <- sapply(gene\_name, function(x) x[[1]][1])

gene\_name <- ifelse(is.na(gene\_name), names(gene\_name), gene\_name)

gene\_short\_name <- gene\_name

gene\_id <- rownames(raw\_count\_data)

genes\_info <- cbind(gene\_id, gene\_short\_name)

genes\_info <- as.data.frame(genes\_info)

rownames(genes\_info) <- rownames(raw\_count\_data)

cds <- new\_cell\_data\_set(expression\_data = raw\_count\_data,

cell\_metadata = cells\_info,

gene\_metadata = genes\_info)

monocle3::reduce\_dimension

cds <- reduce\_dimension(cds)

seurat\_object <- RunUMAP(seurat\_object, dims = 1:10)

reducedDims(cds)$UMAP <- [seurat\_object@reductions$umap@cell.embeddings](mailto:seurat_object@reductions$umap@cell.embeddings)

seurat\_object <- FindClusters(seurat\_object, resolution = 0.3)

cds@clusters$UMAP$partitions <- seurat\_object@meta.data$seurat\_clusters

names(cds@clusters$UMAP$partitions) <- rownames(seurat\_object@meta.data)

cds@clusters$UMAP$clusters <- seurat\_object@meta.data$seurat\_clusters

names(cds@clusters$UMAP$clusters) <- rownames([seurat\_object@meta.data](mailto:seurat_object@meta.data))

cds\_3d <- preprocess\_cds(cds)

cds\_3d <- reduce\_dimension(cds\_3d, max\_components = 3,reduction\_method = 'UMAP')

cds\_3d <- cluster\_cells(cds\_3d,resolution = 1e-5,reduction\_method = "UMAP")

cds\_3d <- learn\_graph(cds\_3d,use\_partition = F)

cds\_3d <- order\_cells(cds\_3d,reduction\_method = 'UMAP')

cds\_3d\_plot\_obj <- plot\_cells\_3d(cds\_3d, color\_cells\_by="pseudotime")

cds\_3d\_plot\_obj <- plot\_cells\_3d(cds\_3d, color\_cells\_by="cluster")

pdf(' pseudotime plot.pdf')

plot\_cells\_3d(cds\_3d, color\_cells\_by="pseudotime")

plot\_cells\_3d(cds\_3d, color\_cells\_by="cellType")

dev.off()