* **R codes**

library(Seurat)

library(dplyr)

library(stringr)

**#Read raw data including single-cell features**

path<-"data"

files <- list.files(path = path,pattern = 'csv$')

fullpath<-paste(path,files,sep="/")

names(fullpath) <- lapply(files,function(x) {str\_extract(x, '[\\w\\d]+')})

rawdata <- lapply(fullpath, function(x) {

cells.table <- read.table(x, sep = ",",check.names = F, header = TRUE, row.names = 1)})

lapply(rawdata, dim)

**#Establish Seurat** **object**

obj.list <- lapply(names(rawdata), function(x) {

CreateSeuratObject(counts = rawdata[[x]],

project = x,

min.cells = 3,

min.features = 200)})

names(obj.list) <- names(rawdata)

obj.list

**# Add grouping information**

obj.list$CDD\_pbmc\_data@meta.data$group <- "CDD"

**#Quality control**

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

for (x in names(obj.list)){

obj.list[[x]][["percent.MT"]] <- PercentageFeatureSet(obj.list[[x]], pattern = "^MT-")

obj.list[[x]][["percent.Ribo"]] <- PercentageFeatureSet(obj.list[[x]], pattern = "^RP[SL]")

HB\_genes <- intersect(HB\_all, rownames(obj.list[[x]]))

obj.list[[x]][["percent.HB"]] <- PercentageFeatureSet(obj.list[[x]], features = HB\_genes)

}

qc\_feature <- c("nFeature\_RNA", "nCount\_RNA", "percent.HB", "percent.MT", "percent.Ribo")

for(sample in names(obj.list)){

pdf(file=paste0("1\_",sample,"\_quality\_control.pdf"),width = 15,height=7)

print(VlnPlot(obj.list[[sample]], features = qc\_feature, ncol = 5, pt.size = 0.5))

dev.off()

}

obj.list <- lapply(obj.list, function(x) {

subset(x, subset = nFeature\_RNA > 200 &

nFeature\_RNA < 6000 &

percent.MT < 20)})

**#****Data normalization**

obj.list <- lapply(obj.list,function(x) {

NormalizeData(x)})

**# Searching for High Variability Genes**

obj.list <- lapply(obj.list, function(x) {

FindVariableFeatures(x, selection.method = "vst", nfeatures = 2000)

})

obj.list

**#Integrate into one object by finding anchors**

obj.anchors <- FindIntegrationAnchors(object.list = obj.list, dims = 1:20)

obj.combined <- IntegrateData(anchorset = obj.anchors, dims = 1:20)

DefaultAssay(obj.combined) <- "integrated"

**# Scale transformation of integrated data**

all.genes <- rownames(obj.combined[["RNA"]]@data)

length(all.genes)

obj.combined <- ScaleData(obj.combined, features = all.genes)

**# Reduce dimensionality and cluster the integrated data**

obj.combined <- RunPCA(obj.combined, npcs = 30, verbose = FALSE)

obj.combined <- RunUMAP(obj.combined, reduction = "pca", dims = 1:20)

obj.combined <- FindNeighbors(obj.combined, reduction = "pca", dims = 1:20)

obj.combined <- FindClusters(obj.combined, resolution = 0.4)

**# Identifying the marker genes**

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

top5 <- sample.markers %>% group\_by(cluster) %>% top\_n(n = 5, wt = avg\_log2FC)

obj.CDD <- subset(obj.combined, subset = group == "CDD")

**# Identify all cells in Cluster 1 and 8 of CDD**

cluster\_cells\_CDD\_1 <- WhichCells(obj.CDD, idents = "1\_CDD")

cluster\_cells\_CDD\_8<- WhichCells(obj.CDD, idents = "8\_CDD")

**# Fetch all gene expression data for** **cluster\_cells\_CDD\_1 and CDD\_8**

all\_genes\_data\_CDD\_1 <- obj.CDD@assays$RNA@counts[, cluster\_cells\_CDD\_1]

all\_genes\_data\_CDD\_1\_matrix <- as.matrix(all\_genes\_data\_CDD\_1)

all\_genes\_data\_CDD\_8 <- obj.CDD@assays$RNA@counts[, cluster\_cells\_CDD\_8]

all\_genes\_data\_CDD\_8\_matrix <- as.matrix(all\_genes\_data\_CDD\_8)

cor(all\_genes\_data\_CDD\_1, all\_genes\_data\_CDD\_8)

**#Linear regression**

x <- all\_genes\_data\_CDD\_1

y <- all\_genes\_data\_CDD\_8

library(ggplot2)

df <- data.frame(x, y)

fit <- lm(y ~ x, data = df)

intercept <- coef(fit)[1]

slope <- coef(fit)[2]

p <- ggplot(df, aes(x=x, y=y)) +geom\_point(colour="purple", alpha=0.3) +

geom\_smooth(method=lm, se=FALSE, color="brown", level=0) +

ggtitle(paste("Regression Line: y =", round(slope, 2), "\*x +", round(intercept, 2))) +theme\_minimal()+theme(axis.text.x=element\_text(color="black", size=25),

axis.text.y=element\_text(color="black", size=25),

axis.title=element\_text(size=25),

axis.line = element\_line(size=1, colour = "black"))