R.version

.libPaths()

getwd()

rm(list = ls())

library(tidyverse)

#### 0 GSEs, GSEs\_sm, GSE\_expr, GSE\_pd ----

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

GSEs <- c("GSE137340",

"GSE137341",

"GSE100159",

"GSE33118",

"GSE9960",

"GSE69063",

"GSE69528",

"GSE54514")

GSE\_sm <- paste0(GSEs, "\_sm")

GSE\_expr <- paste0(GSEs, "\_expr")

GSE\_pd <- paste0(GSEs, "\_pd")

#if (!requireNamespace("BiocManager", quietly = TRUE))

# install.packages("BiocManager")

#BiocManager::install("GEOquery")

library(GEOquery)

for (i in 1:8) {

assign(GSE\_sm[i], getGEO(GEO = GSEs[i],

filename = paste0(GSEs[i], "\_series\_matrix.txt.gz"),

getGPL = F))

assign(GSE\_expr[i], exprs(get(GSE\_sm[i])))

assign(GSE\_pd[i], pData(get(GSE\_sm[i])))

print(i)

}

save.image(file='.//alldata.Rdata')

#load("alldata.Rdata")

save(GSE137340\_expr, GSE137340\_pd, GSE137340\_sm, file = "./GSE137340/GSE137340.Rda")

save(GSE100159\_expr, GSE100159\_pd, GSE100159\_sm, file = "./GSE100159/GSE100159.Rda")

save(GSE33118\_expr, GSE33118\_pd, GSE33118\_sm, file = "./GSE33118/GSE33118.Rda")

save(GSE9960\_expr, GSE9960\_pd, GSE9960\_sm, file = "./GSE9960/GSE9960.Rda")

save(GSE69063\_expr, GSE69063\_pd, GSE69063\_sm, file = "./GSE69063/GSE69063.Rda")

save(GSE69528\_expr, GSE69528\_pd, GSE69528\_sm, file = "./GSE69528/GSE69528.Rda")

save(GSE54514\_expr, GSE54514\_pd, GSE54514\_sm, file = "./GSE54514/GSE54514.Rda")

pdf("./figures/Boxplots.PDF", 15, 6)

for (i in 1:8) {

boxplot(get(GSE\_expr[i]),

las=2,

outline=FALSE,

main = paste0("Boxplot for ", GSEs[i]))

}

dev.off()

#### 1 GSE137340 ----

### load data

#view(GSE137340\_pd)

#fix(GSE137340\_pd)

#dir.create("GSE137340")

rm(list = ls())

library(tidyverse)

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE137340")

getwd()

load("GSE137340.Rda")

### organize target data

# load("./GSE137340.Rda")

# write.table(GSE137340\_pd, file = "GSE137340\_pd\_1.txt",sep = "\t",row.names = F,col.names = T,quote = F)

# write.csv(GSE137340\_pd, file = "GSE137340\_pd\_1.csv",sep = "\t",row.names = F,col.names = T,quote = F)

# write.csv(GSE137340\_pd, file = "GSE137340\_pd\_2.csv",quote = F)

## Extract target GSE targets

GSE137340\_targets <- GSE137340\_pd %>%

dplyr::select(geo\_accession, source\_name\_ch1) %>%

separate(source\_name\_ch1, into = paste0("x", 1:3), sep = ",", remove = F) %>%

mutate(sample\_id = geo\_accession,

# sample\_name = str\_c(x1, x2, sep = "\_"),

sample\_name = x1,

patiend\_id = str\_c(x2,str\_extract(x1, "\\d+"),sep = "\_"),

Day = str\_sub(source\_name\_ch1,-5,-1),

# group = if\_else(str\_detect(source\_name\_ch1, 8, 8) %in% c("C", "H"),"Control", "NAFLD"),

group = if\_else(str\_detect(source\_name\_ch1, "Healthy"),"Control", "Sepsis")) %>%

filter(!str\_detect(source\_name\_ch1,"Day 2")) %>%

arrange(group, sample\_id) %>%

dplyr::select(sample\_id:group)

GSE137340\_targets$Day <- gsub("althy","Healthy", GSE137340\_targets$Day)

dim(GSE137340\_expr)

GSE137340\_expr <- GSE137340\_expr[,GSE137340\_targets$sample\_id]

# fix(GSE137340\_targets)

### check qualifty

source("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/Custom\_Functions.R")

boxplot(GSE137340\_expr, las=2, outline=FALSE, main = "GSE137340")

PCA\_new(GSE137340\_expr,

ntop = nrow(GSE137340\_expr),

# ntop = 500,

# show\_name = T,

group = GSE137340\_targets$group)

dev.off()

# RMA normalized: the standardization of the original data

GSE137340\_sm@phenoData@data$data\_processing[1]

# GSE137340\_pd$data\_processing[1] #

library(limma)

GSE137340\_expr\_n = normalizeBetweenArrays(GSE137340\_expr)

dim(GSE137340\_expr\_n)

if(!dir.exists("figures")) dir.create("figures")

pdf('./figures/GSE137340\_normalize\_norm\_boxplot.pdf',width = 8,height = 8)

boxplot(GSE137340\_expr\_n,

outline=FALSE,

#notch=T,

#col=group\_list,

main = "GSE137340",

las=2)

dev.off()

pdf('./figures/GSE137340\_normalize\_norm\_PCA\_new.pdf',width = 8,height = 8)

PCA\_new(GSE137340\_expr\_n,

ntop = 1000,

# show\_name = T,

group = GSE137340\_targets$group)

dev.off()

range(GSE137340\_expr\_n)

# exp <- log2(GSE137340\_expr+1)

# range(GSE137340\_expr)

### annotation

# BiocManager::install("illuminaHumanv4.db",force = TRUE)

library(illuminaHumanv4.db)

ls("package:illuminaHumanv4.db") #query

ls(illuminaHumanv4.db)

keytypes(illuminaHumanv4.db)

# View annotation data

GSE137340\_probe2symbol <- as.data.frame(illuminaHumanv4SYMBOL)

# GSE137340\_probe2symbol = toTable(illuminaHumanv4SYMBOL)

head(GSE137340\_probe2symbol)

GSE137340\_anno <- GSE137340\_expr\_n %>%

as.data.frame() %>%

dplyr::select(GSE137340\_targets$sample\_id) %>%

rownames\_to\_column(var = "probe\_id") %>%

inner\_join(GSE137340\_probe2symbol, ., by = "probe\_id") %>%

dplyr::select(-1) %>%

as.data.frame() %>%

aggregate(.~ symbol, data = ., mean) %>%

column\_to\_rownames(var = "symbol")

# View annotation results

GSE137340\_anno[1:6,1:6]

dim(GSE137340\_expr\_n)

dim(GSE137340\_anno)

## save data

save(GSE137340\_expr\_n, GSE137340\_targets, GSE137340\_anno, file = "GSE137340\_processsed.Rda")

library(limma)

group <- factor(GSE137340\_targets$group, levels = c("Control", "Sepsis"))

expr <- GSE137340\_anno

contrast <- paste0(rev(levels(group)), collapse = "-")

contrast

design <- model.matrix( ~ 0 + group)

colnames(design)

colnames(design) <- levels(group)

contrast.matrix <- makeContrasts(contrast, levels = design)

contrast.matrix

## arrayWeights

aw <- arrayWeights(expr, design = design)

aw

barplot(aw)

fit <- lmFit(expr, design, weights = aw)

fit <- contrasts.fit(fit, contrast.matrix)

fit <- eBayes(fit)

GSE137340\_DEG <- topTable(fit, coef = 1, n = Inf) %>%

rownames\_to\_column(var = "symbol")

#head(GSE137340\_DEG)

GSE137340\_DEG\_p <- GSE137340\_DEG %>%

dplyr::filter(abs(logFC) > 1.2, P.Value < 0.05)

head(GSE137340\_DEG\_p)

# setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE137340")

save(GSE137340\_DEG, GSE137340\_DEG\_p, file = "./GSE137340\_DEG.Rda")

# write.table(GSE137340\_DEG, file = "GSE137340\_DEG.txt",sep = "\t",row.names = F,col.names = T,quote = F)

# write.table(GSE137340\_DEG\_p, file = "GSE137340\_DEG\_p.txt",sep = "\t",row.names = F,col.names = T,quote = F)

##

# plot\_df

plot\_df <- GSE137340\_DEG

# Define volcano plot thresholds

plot\_df$threshold = factor(ifelse(plot\_df$P.Value < 0.05 & abs(plot\_df$logFC) > 1.2,

ifelse(plot\_df$logFC > 1.2 ,'Up','Down'),'NS'),

levels=c('Up','Down','NS'))

#

library(ggplot2)

pdf('./figures/Volcano\_GSE137340.pdf',width = 8,height = 8)

ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +#

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +#

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

dev.off()

p1 <- ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

p1

dev.off()

save(p1, file = "p1.Rda")

#### 2 GSE69063 ----

rm(list = ls())

### load data

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE69063")

getwd()

load\_input <- load("GSE69063.Rda")

load\_input

library(tidyverse)

view(GSE69063\_pd)

#fix(GSE69063\_pd)

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

#dir.create("GSE69063")

if(!dir.exists("GSE69063")) dir.create("GSE69063")

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE69063")

getwd()

### organize target data

GSE69063\_targets <- GSE69063\_pd %>% dplyr::select(2, 1) %>%

separate(title, into = paste0("x", 1:3), sep = ",", remove = F) %>%

filter(!str\_detect(title,"Day 2")) %>%

filter(!str\_detect(title,"Day 4")) %>%

filter(!str\_detect(title,"Day 5")) %>%

mutate(sample\_id = geo\_accession,# add several columns

#file\_name = paste0(sample\_id, ".CEL.gz"),

#sample\_name = title,

sample\_name = str\_c(x1, x2, sep = "\_"),# Join multiple strings into a single string.

patient\_id = x2,

# patient\_id = str\_match(x2, "\\d")[, 2],

# strings match the first character

# and see the computer 360 browser collection folder (matching characters)

# and see (13\_Reading Affymetrix chip data.R) code: P1455

#str\_sub----Extract and replace substrings from a character vector. Extract the substring from the last character

#GSE\_num = "GSE69063",

group = if\_else(str\_detect(title, "Healthy"),"Control", "Sepsis")) %>%

dplyr::select(sample\_id:group)

GSE69063\_expr <- GSE69063\_expr[,GSE69063\_targets$sample\_id] %>% as.data.frame()

dim(GSE69063\_expr)

## View the scope of the data

range(GSE69063\_expr)

# exp <- log2(GSE9960\_expr+1)

# range(GSE9960\_expr)

if(!dir.exists('figures'))dir.create('figures')

### Check qualifty

## boxplot

boxplot(GSE69063\_expr, las=2, outline=FALSE,

# col = as.factor(GSE69063\_targets$group),

main = "GSE69063\_expr")

dev.off()

source("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/Custom\_Functions.R")

PCA\_new(GSE69063\_expr,

ntop = nrow(GSE69063\_expr),

# ntop = 500, #the number of top-ranked genes

# show\_name = T,

group = GSE69063\_targets$group)

dev.off()

#### 3 GSE69528 ----

rm(list = ls())

### load data

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE69528")

getwd()

load\_input <- load("GSE69528.Rda")

load\_input

library(tidyverse)

view(GSE69528\_pd)

#fix(GSE69528\_pd)

### create new folder

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

#dir.create("GSE69528")

if(!dir.exists("GSE69528")) dir.create("GSE69528")

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE69528")

getwd()

### Collate target data

GSE69528\_targets <- GSE69528\_pd %>% dplyr::select(2, 1, 8) %>% dplyr::select(1, 2) %>%

separate(title, into = paste0("x", 1:7), sep = " ", remove = F) %>%

filter(!str\_detect(title,"Day 2")) %>%

mutate(sample\_id = geo\_accession,# add several columns

#file\_name = paste0(sample\_id, ".CEL.gz"),

#sample\_name = title,

sample\_name = str\_c(x1, x2, sep = "\_"),# Join multiple strings into a single string.

patient\_id = x7,

#str\_sub----Extract and replace substrings from a character vector. Extract the substring from the last character

#GSE\_num = "GSE69063",

group = if\_else(str\_detect(title, "Healthy"),"Control", "Sepsis")) %>%

dplyr::select(sample\_id:group) %>%

arrange(group, sample\_id) %>%

dplyr::select(2,3,4,1) %>% arrange(group,sample\_id)

GSE69528\_expr <- GSE69528\_expr[,GSE69528\_targets$sample\_id] %>% as.data.frame()

## View the scope of the data

range(GSE69528\_expr)

# exp <- log2(GSE69528\_expr+1)

# range(GSE69528\_expr)

if(!dir.exists("figures")) dir.create("figures")

## checkout quality of data

pdf('./figures/boxplot\_GSE69528.pdf')

boxplot(GSE69528\_expr,

las=2,

outline=FALSE,

main = "GSE69528\_expr")

dev.off()

source("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/Custom\_Functions.R")

PCA\_new(GSE69528\_expr,

ntop = nrow(GSE69528\_expr),

# ntop = 500, #the number of top-ranked genes

# show\_name = T,

group = GSE69528\_targets$group)

dev.off()

## View the original data processing method

GSE69528\_sm@phenoData@data$data\_processing[1]

# GSE69528\_pd$data\_processing[1] # This code is equivalent to the above code

#### 4 GSE54514 ----

### 4.4 GSE54514 ----

#### 1 GSE54514 data import

### 1.1 load data

###control group NO.1 day VS sepsis group NO.3 day

rm(list=ls())

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE54514")

getwd()

### load data

load\_input <- load("GSE54514.Rda")

load\_input

library(tidyverse)

### Since PCA analysis shows that the sepsis group and the control group cannot be well separated,

### we do not use the expression matrix extracted from the Series Matrix, but read the original data to remove batch effects

## Read the original data expression matrix

# See 1: F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 1/L51\_Codes/NAFLD\_6Datasets\_DEG.R Code: P160~P173

# See 2: F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 1/L51\_Codes/13\_Reading Affymetrix Chip Data.R Code: P12~P111

library(oligo)

# In the oligo package, the read.celfiles function reads the CEL file:

# obtain the original file list

cel\_files <- list.celfiles("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE54514/GSE54514\_RAW/",

full.name = T,

# full.name = F,

listGzipped = T)

cel\_files\_1 <- ifelse(str\_match(cel\_files,"GSM\\d+") %in% GSE54514\_targets$sample\_id,TRUE,FALSE)

cel\_files\_2 <- cel\_files[cel\_files\_1]

cel\_files1 <- list.celfiles("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE54514/GSE54514\_RAW/",

# full.name = T,

full.name = F,

listGzipped = T)

cel\_files\_11 <- ifelse(str\_match(cel\_files1,"GSM\\d+") %in% GSE54514\_targets$sample\_id,TRUE,FALSE)

cel\_files\_21 <- cel\_files1[cel\_files\_11]

identical(cel\_files\_1,cel\_files\_11)

GSE54514\_targets <- GSE54514\_targets %>% mutate(FileName = cel\_files\_2) %>%

relocate(FileName,everything())

## new cel: read the original file to obtain the ExpressionFeatureset

GSE54514\_cel <- read.celfiles(filenames = GSE54514\_targets$FileName,

phenoData = AnnotatedDataFrame(data = GSE54514\_targets),

sampleNames = GSE54514\_targets$sample\_id)

# See F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 1/L51\_Codes/NAFLD\_6Datasets\_DEG.R Code: P166

GSE54514\_cel1 <- read.celfiles(filenames = paste0("GSE54514\_RAW/", cel\_files\_21))

identical(GSE54514\_cel,GSE54514\_cel1)

a <- exprs(GSE54514\_cel) %>% as.data.frame()

b <- exprs(GSE54514\_cel1) %>% as.data.frame()

head(a)[,1:6]

head(b)[,1:6]

identical(a,b)

## Lookup batch effect: The first way to extract batch effect

cel <- GSE54514\_cel

cel\_pd <- pData(cel)

cel@protocolData@data$dates

cel\_dates <- str\_split(cel@protocolData@data$dates, pattern = "\\s+", simplify = T)[,1]

# \s matches any whitespace character, and is equivalent to [ \f\n\r\t\v]

class(cel\_dates)

table(cel\_dates) # Check batches

## Lookup batch effect: The second way to extract batch effect

library(sva)

library(tidyverse)

# Chip scanning date can be considered as an important batch effect factor

GSE54514\_targets\_batch <- GSE54514\_targets

cel\_dates1 <- cel@protocolData@data$dates %>%

str\_sub(1, 8)

identical(cel\_dates,cel\_dates1)

GSE54514\_targets\_batch$batch <- cel@protocolData@data$dates %>%

str\_sub(1, 8)# Extract the first 8 characters

## save data

save(GSE54514\_cel,GSE54514\_targets\_batch,

file = ".\\GSE54514\_cel.Rda")

## Remove batch effect

## PCA analysis shows that the normal group and the sepsis group have batch effects,

## so the original data is read to remove batch effects

## 4.4.1 Do not include grouping factors

mod <- model.matrix(~1, data= GSE54514\_targets\_batch)

GSE54514\_combat <- ComBat(dat = GSE54514\_expr,

batch = GSE54514\_targets\_batch$batch,

mod = mod,

par.prior=TRUE, # Use parameter estimation, false for non-parametric estimation which is more time-consuming

prior.plots=FALSE)

# ## 4.4.2 Include grouping factors, tell the algorithm what grouping factors we are interested in (It turns out that adding grouping factors is useless)

# mod1 <- model.matrix(~ as.factor(group), data= GSE54514\_targets\_batch)

# mod2 <- model.matrix(~1 + group, data= GSE54514\_targets\_batch)

# identical(mod1,mod2)

# GSE54514\_combat1 <- ComBat(dat = GSE54514\_expr,

# batch = GSE54514\_targets\_batch$batch,

# mod = mod1,

# par.prior=TRUE,

# prior.plots=FALSE)

## Check the effect of removing batch effects

# boxplot after batch

boxplot(GSE54514\_expr,

# ylab = expression(log[2](intensity)),

ylab = expression(intensity),

las = 2,

outline = FALSE)

boxplot(GSE54514\_combat,

# ylab = expression(log[2](intensity)),

ylab = expression(intensity),

las = 2,

outline = FALSE)

### PCA after batch

PCA\_new(GSE54514\_combat,

ntop = nrow(GSE54514\_combat),

# show\_name = T,

group = GSE54514\_targets\_batch$group)

dev.off()

# #### After the above analysis, it was found that the data obtained by reading the original data and removing batch effects is the most reasonable,

# so this part of the data is used for subsequent analysis ----

### annotation

GSE54514\_sm@annotation

## GPL570

library(hgu133plus2.db)

library(illuminaHumanv3.db)

ls("package:hgu133plus2.db")

p2s <- toTable(hgu133plus2SYMBOL) #%>%

# set\_names("probe\_id", "symbol")# rename

head(p2s)

## Custom annotation function

annotate\_expr <- function(expr, p2s, fun = mean){# Define three parameters, the column names of p2s are fixed

library(dplyr)

library(tibble)

# expr

expr <- expr %>%

as.data.frame() %>%

tibble::rownames\_to\_column("probe\_id")# Change the row name to a column called "probe\_id"

# annotate

expr\_annotated <- p2s %>%

dplyr::inner\_join(expr, by = "probe\_id") %>% # Take the intersection of expr and p2s

na.omit() %>% # Remove NAs

dplyr::select(-probe\_id) %>% # Remove the "probe\_id" column

stats::aggregate(. ~ symbol, data =., FUN = fun) %>% # For multiple probes corresponding to the same symbol, take the mean of these probes

tibble::column\_to\_rownames("symbol")# Change the "symbol" column to the row name

return(expr\_annotated)# Return this expr\_annotated, and then assign it to annotate\_expr

}

## Annotation

GSE54514\_anno <- annotate\_expr(GSE54514\_expr, p2s)

head(GSE54514\_anno)[, 1:3]

GSE54514\_expr\_n <- GSE54514\_expr # This step is to keep all data sets consistent

## save data

save(GSE54514\_expr\_n, GSE54514\_targets, GSE54514\_anno,

file = ".\\GSE54514\_processed.Rda")

### Differential analysis DEGs

library(limma)

## Load chip data and define groups

group <- factor(GSE54514\_targets$group, levels = c("Control", "Sepsis")) # Use factor to define the grouping group

expr <- GSE54514\_anno

contrast <- paste0(rev(levels(group)), collapse = "-")

contrast

## Create design matrix

design <- model.matrix( ~ 0 + group)

colnames(design)

colnames(design) <- levels(group)

## Create contrast matrix

contrast.matrix <- makeContrasts(contrast, levels = design)

contrast.matrix

## Use limma for differential analysis

fit <- lmFit(expr, design)

fit <- contrasts.fit(fit, contrast.matrix)

fit <- eBayes(fit)

## Check all results

results <- summary(decideTests(fit, lfc = 1.2))

results

## For design

design <- model.matrix(~ group, data = GSE54514\_targets)

colnames(design)

colnames(design) <- c("Control", "SepsisvsControl")

head(design)

## arrayWeights

expr <- GSE54514\_anno

aw <- arrayWeights(expr, design = design)

aw

barplot(aw)

## Use limma for differential analysis

fit <- lmFit(expr, design, weights = aw)

fit <- contrasts.fit(fit, contrast.matrix)

fit <- eBayes(fit)

## DEG results

results <- summary(decideTests(fit, lfc = 1.2))

results

# number = Inf: all differential analysis results; SepsisvsControl: represents the type to be compared

res\_DEG <- topTable(fit, coef = "SepsisvsControl", number = Inf)

# res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1)# Here p is the corrected p

res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1.2)

GSE54514\_DEG1 <- topTable(fit, coef = 2, n = Inf)

identical(res\_DEG, GSE54514\_DEG1)

GSE54514\_DEG <- topTable(fit, coef = 2, n = Inf) %>% # coef = 1 when there is no contrast.matrix()

rownames\_to\_column(var = "symbol")

# Filter for |logFC|>1.2 and p<0.05 results

GSE54514\_DEG\_p <- GSE54514\_DEG %>%

dplyr::filter(abs(logFC) > 1.2 , P.Value < 0.05)

### save data

save(GSE54514\_DEG, GSE54514\_DEG\_p,

file = ".\\GSE54514\_DEG.Rda")

### Draw volcano plot

## plot\_df

plot\_df <- GSE54514\_DEG

# Define volcano plot thresholds

plot\_df$threshold = factor(ifelse(plot\_df$P.Value < 0.05 & abs(plot\_df$logFC) > 1.2,

ifelse(plot\_df$logFC > 1.2 ,'Up','Down'),'NS'),

levels=c('Up','Down','NS'))

# Start plotting

library(ggplot2)

pdf('./figures/Volcano\_GSE54514.pdf',width = 8,height = 8)

ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

dev.off()

p4 <- ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

p4

dev.off()

save(p4, file = "./p4.Rda")

#### 5 GSE57065----

### 1 load packages

# rm(list = ls())

library(GEOquery)

library(readxl)

library(tidyverse)

### 2 Load data

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

getwd()

GSE57065\_sm<- getGEO(filename = "GSE57065\_series\_matrix.txt.gz",

destdir = ".",

getGPL = F )

GSE57065\_expr <- exprs(GSE57065\_sm)

head(GSE57065\_expr)[, 1:6]

GSE57065\_pd <- pData(GSE57065\_sm)

### Create file

# dir.create("GSE57065")

if(!dir.exists("GSE57065"))dir.create("GSE57065")

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE57065")

getwd()

### Save data

save(GSE57065\_sm,GSE57065\_expr,GSE57065\_pd,

file = ".\\GSE57065.Rda")

# load(".\\GSE57065\_data.Rda")

### Organize target data

GSE57065\_targets <- GSE57065\_pd %>% dplyr::select(2, 1, 13) %>% dplyr::select(1, 2) %>%

separate(title, into = paste0("x", 1:3), sep = "\_", remove = F) %>%

mutate(sample\_id = geo\_accession,# add several columns

#file\_name = paste0(sample\_id, ".CEL.gz"),

#sample\_name = title,

sample\_name = str\_c(x1, x2, sep = "\_"),# Join multiple strings into a single string.

patient\_id = x2,

#str\_sub----Extract and replace substrings from a character vector. Extract the substring from the last character

#GSE\_num = "GSE69063",

group = if\_else(str\_sub(x2,1,1) == "P","Sepsis","Control")) %>%

filter(!(x3 %in% c("H24", "H48"))) %>% # "F:/HelixCourses/R Language from Scratch to Mastery\_7 days/Day 3/Section 3 Live Broadcast" Code: P264

dplyr::slice(-c(1:10)) %>%

dplyr::slice(-c(31:42)) %>%

arrange(group, sample\_id) %>%

dplyr::select(sample\_id:group)

GSE57065\_expr <- GSE57065\_expr[,GSE57065\_targets$sample\_id] %>% as.data.frame()

dim(GSE57065\_expr)

## View Data

range(GSE57065\_expr)

# exp <- log2(GSE57065\_expr+1)

# range(GSE57065\_expr)

### Evaluate data

if(!dir.exists("figures")) dir.create("figures")

# boxplot(GSE29450\_rma, las = 2, col = rep(c("blue", "red"), each = 10))

# boxplot(GSE57065\_expr, las=2, outline=FALSE,

# col = rep(c("blue", "red"), c(22,25)),

# main = "GSE57065\_expr")

# Below code see F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 1/L51\_Codes/32\_batch\_effect\_data2.R Code: P29~P32

# boxplot(GSE57065\_expr, las=2, outline=FALSE,

# col = rep(c("blue", "red"), c(22,25)), outline = F)

boxplot(GSE57065\_expr,

# ylab = expression(log[2](intensity)),

ylab = expression(intensity),

las = 2,

outline = FALSE)

dev.off()

source("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Custom\_Functions.R")

targets <- GSE57065\_targets

PCA\_new(GSE57065\_expr, show\_name = F)

dev.off()

# View the original data processing method

GSE57065\_sm@phenoData@data$data\_processing[1]

# GSE57065\_pd$data\_processing[1] # This code is equivalent to the above code

## save data

# save(GSE57065\_expr,GSE57065\_targets,

# file = ".\\GSE57065\_processed.Rda")

### 5 remove batch effect

## PCA analysis shows that the normal group and the sepsis group have batch effects,

## so the original data is read to remove batch effects

## Use oligo package to read CEL files

# See 1: F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 1/L51\_Codes/NAFLD\_6Datasets\_DEG.R Code: P160~P173

# See 2: F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 1/L51\_Codes/13\_Reading Affymetrix Chip Data.R Code: P12~P111

library(oligo)

cel\_files <- list.celfiles("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE57065/GSE57065\_RAW/",

full.name = T,

# full.name = F,

listGzipped = T)

# # str\_match returns multiple column matching results, the first column is the overall matching result,

# # the following columns are the matching results for each group.

# c <- str\_match(cel\_files,"GSM\\d+")[,1] # Extract the vector

# c1 <- str\_match(cel\_files,"GSM\\d+") # Extract the list

# c2 <- str\_match(cel\_files,"GSM\\d+")[,2]

# d <- str\_match(GSE54514\_targets\_2$x4, "\\d+")[, 1]

# d <- str\_match(GSE54514\_targets\_2$x4, "\\d+")[, 2]

cel\_files\_1 <- ifelse(str\_match(cel\_files,"GSM\\d+") %in% GSE57065\_targets$sample\_id,TRUE,FALSE)

cel\_files <- cel\_files[cel\_files\_1]

cel\_files1 <- list.celfiles("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE57065/GSE57065\_RAW/",

# full.name = T,

full.name = F,

listGzipped = T)

cel\_files\_11 <- ifelse(str\_match(cel\_files1,"GSM\\d+") %in% GSE57065\_targets$sample\_id,TRUE,FALSE)

cel\_files1 <- cel\_files1[cel\_files\_11]

identical(str\_match(cel\_files,"GSM\\d+")[,1],GSE57065\_targets$sample\_id)

identical(cel\_files\_1,cel\_files\_11)

GSE57065\_targets\_FileName <- GSE57065\_targets %>% mutate(FileName = cel\_files) %>%

relocate(FileName,everything())

## new cel

GSE57065\_cel <- read.celfiles(filenames = GSE57065\_targets\_FileName$FileName,

phenoData = AnnotatedDataFrame(data = GSE57065\_targets\_FileName),

sampleNames = GSE57065\_targets\_FileName$sample\_id)

# GSE57065\_cel1 <- read.celfiles(cel\_files) # This reading method only obtains the expression matrix, without target and sampleNames data

## Save raw data

save(GSE57065\_cel, file = "./GSE57065\_cel.Rdata")

## Lookup batch effect

# cel <- GSE57065\_cel

# cel\_pd <- pData(cel)

# cel@protocolData@data$dates

# cel\_dates <- str\_split(cel@protocolData@data$dates, pattern = "\\s+", simplify = T)[,1]

# \s matches any whitespace character, and is equivalent to [ \f\n\r\t\v]

class(cel\_dates)

table(cel\_dates) # Check batches

# Chip scanning date can be considered as an important batch effect factor

GSE57065\_targets\_batch <- GSE57065\_targets

GSE57065\_targets\_batch$batch <- GSE57065\_cel@protocolData@data$dates %>%

str\_sub(1, 8)# Extract the first 8 characters

table(GSE57065\_targets\_batch$batch)

## 5.9.1 Do not include grouping factors

library(sva)

mod <- model.matrix(~1, data= GSE57065\_targets\_batch)

batch <- GSE57065\_targets\_batch$batch

GSE57065\_expr <- exprs(GSE57065\_cel) %>% as.data.frame()

range(GSE57065\_expr)

# GSE57065\_expr\_rma <- rownames\_to\_column(GSE57065\_expr, "ID")

# GSE57065\_expr\_rma <- GSE57065\_expr\_rma[!duplicated(GSE57065\_expr\_rma$ID), ]

GSE57065\_combat <- ComBat(dat = GSE57065\_expr,

batch = batch,

mod = mod,

par.prior=TRUE, # Use parameter estimation, false for non-parametric estimation which is more time-consuming

prior.plots=FALSE)

# ## 5.9.2 Include grouping factors, tell the algorithm what grouping factors we are interested in

# mod1 <- model.matrix(~ as.factor(group), data= GSE57065\_targets\_batch)

# mod2 <- model.matrix(~1 + group, data= GSE57065\_targets\_batch)

# GSE57065\_combat1 <- ComBat(dat = GSE57065\_expr,

# batch = batch,

# mod = mod1,

# par.prior=TRUE,

# prior.plots=FALSE)

## Check quality

## boxplot after batch

boxplot(GSE57065\_expr ,

# ylab = expression(log[2](intensity)),

ylab = expression(intensity),

las = 2,

outline = FALSE)

boxplot(GSE57065\_combat,

# ylab = expression(log[2](intensity)),

ylab = expression(intensity),

las = 2,

outline = FALSE)

dev.off()

## plot

GSE57065\_p1 <- PCA\_new(GSE57065\_expr,

ntop = 10000,

# ntop = nrow(GSE57065\_expr), #the number of top-ranked genes

# show\_name = T,

group = GSE57065\_targets\_batch$group )

GSE57065\_p1

dev.off()

GSE57065\_p2 <- PCA\_new(GSE57065\_combat,

ntop = 10000,

group = GSE57065\_targets\_batch$group,

show\_name = F)+

ggtitle("PCA before batch effect correction") +

theme(plot.title = element\_text(hjust = 0.5))

GSE57065\_p2

GSE57065\_p3 <- PCA\_new(GSE57065\_combat1,

ntop = 10000,

group = GSE57065\_targets\_batch$group,

show\_name = T)+

ggtitle("PCA before batch effect correction") +

theme(plot.title = element\_text(hjust = 0.5))

GSE57065\_p3

dev.off()

library(cowplot)

GSE57065\_p123 <- cowplot::plot\_grid(GSE57065\_p1, GSE57065\_p2, GSE57065\_p3, nrow = 2)

# pdf("batch/PCA\_GSE57065\_batch.pdf", height = 7, width = 15)

GSE57065\_p123

dev.off()

## save data

# save(GSE57065\_combat1,GSE57065\_targets\_batch,

# file = ".\\GSE57065\_batched.Rda")

## Annotation

GSE57065\_sm@annotation

## GPL570

library(hgu133plus2.db)

library(illuminaHumanv3.db)

ls("package:hgu133plus2.db")

p2s <- toTable(hgu133plus2SYMBOL) #%>%

# set\_names("probe\_id", "symbol")# rename

head(p2s)

## Custom annotation function

annotate\_expr <- function(expr, p2s, fun = mean){# Define three parameters, the column names of p2s are fixed

library(dplyr)

library(tibble)

# expr

expr <- expr %>%

as.data.frame() %>%

tibble::rownames\_to\_column("probe\_id")# Change the row name to a column called "probe\_id"

# annotate

expr\_annotated <- p2s %>%

dplyr::inner\_join(expr, by = "probe\_id") %>% # Take the intersection of expr and p2s

na.omit() %>% # Remove NAs

dplyr::select(-probe\_id) %>% # Remove the "probe\_id" column

stats::aggregate(. ~ symbol, data =., FUN = fun) %>% # For multiple probes corresponding to the same symbol, take the mean of these probes

tibble::column\_to\_rownames("symbol")# Change the "symbol" column to the row name

return(expr\_annotated)# Return this expr\_annotated, and then assign it to annotate\_expr

}

## Annotation

GSE57065\_expr\_anno <- annotate\_expr(GSE57065\_combat1, p2s)

head(GSE57065\_expr\_anno)[, 1:3]

## save data

GSE57065\_expr\_n <- GSE57065\_combat1

GSE57065\_anno <- GSE57065\_expr\_anno

save(GSE57065\_expr\_n, GSE57065\_targets, GSE57065\_anno,

file = ".\\GSE57065\_processed.Rda")

### Differential analysis DEGs

library(limma)

## For design

design <- model.matrix(~ group, data = GSE57065\_targets\_batch)

colnames(design)

colnames(design) <- c("Control", "SepsisvsControl")

head(design)

## arrayWeights

# if (!requireNamespace("arrayWeights", quietly = TRUE)) {

# BiocManager::install("arrayWeights", ask = F, update = F)

# }

expr <- GSE57065\_anno

aw <- arrayWeights(expr, design = design)

aw

barplot(aw)

dev.off()

# fit <- lmFit(expr\_anno, design)# Linear fitting

fit <- lmFit(expr, design, weights = aw)

fit <- eBayes(fit)# Bayesian

## DEG results

results <- summary(decideTests(fit, lfc = 1.2))

results

# number = Inf: all differential analysis results; SepsisvsControl: represents the type to be compared

res\_DEG <- topTable(fit, coef = "SepsisvsControl", number = Inf)

# res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1)# Here p is the corrected p

res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1.2)

GSE57065\_DEG1 <- topTable(fit, coef = 2, n = Inf)

identical(res\_DEG, GSE57065\_DEG1)

GSE57065\_DEG <- topTable(fit, coef = 2, n = Inf) %>% # coef = 1 when there is no contrast.matrix()

rownames\_to\_column(var = "symbol")

# Filter for |logFC|>1.2 and p<0.05 results

GSE57065\_DEG\_p <- GSE57065\_DEG %>%

dplyr::filter(abs(logFC) > 1.2 , P.Value < 0.05)

### save data

save(GSE57065\_DEG, GSE57065\_DEG\_p,

file = ".\\GSE57065\_DEG.Rda")

### 5.12 Draw volcano plot

## plot\_df

plot\_df <- GSE57065\_DEG

# Define volcano plot thresholds

plot\_df$threshold = factor(ifelse(plot\_df$P.Value < 0.05 & abs(plot\_df$logFC) > 1.2,

ifelse(plot\_df$logFC > 1.2 ,'Up','Down'),'NS'),

levels=c('Up','Down','NS'))

# Start plotting

library(ggplot2)

pdf('./figures/Volcano\_GSE57065.pdf', width = 8, height = 8)

ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

dev.off()

p5 <- ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

p5

dev.off()

save(p5, file = "./p5.Rda")

#### 6 GSE95233 ----

### 6.1 load packages

# rm(list = ls())

library(GEOquery)

library(readxl)

library(tidyverse)

### 2 Load data

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

getwd()

GSE95233\_sm<- getGEO(filename = "GSE95233\_series\_matrix.txt.gz",

destdir = ".",

getGPL = F )

GSE95233\_expr <- exprs(GSE95233\_sm)

head(GSE95233\_expr)[, 1:6]

GSE95233\_pd <- pData(GSE95233\_sm)

### Create file

# dir.create("GSE95233")

if(!dir.exists("GSE95233"))dir.create("GSE95233")

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE95233")

getwd()

### Save data

save(GSE95233\_sm, GSE95233\_expr, GSE95233\_pd,

file = ".\\GSE95233.Rda")

# load(".\\GSE95233\_data.Rda")

### Organize target data

GSE95233\_pd\_1 <- GSE95233\_pd # "gsub" means replace characters # "F:/Dr. Chen/Dr. Chen wants to lie down/Xena.R" Code: P1455

GSE95233\_pd\_1$title <- gsub("-","\_",GSE95233\_pd\_1$title)

GSE95233\_targets <- GSE95233\_pd\_1 %>% dplyr::select(2, 1, 8) %>%

# dplyr::select(1, 2) %>%

separate(title, into = paste0("x", 1:4), sep = "\_", remove = F) %>%

filter(!(x4 %in% c("D02", "D03"))) %>% # "F:/HelixCourses/R Language from Scratch to Mastery\_7 days/Day 3/Section 3 Live Broadcast" Code: P264

mutate(sample\_id = geo\_accession,# add several columns

#file\_name = paste0(sample\_id, ".CEL.gz"),

#sample\_name = title,

sample\_name = str\_c(x1, x2, x3, sep = "\_"),# Join multiple strings into a single string.

patient\_id = x3 ,

# patient\_id = str\_match(GSE54514\_targets\_2$x4, "\\d+")[, 1],

# patient\_id\_1 = str\_match(GSE54514\_targets\_2$x4, "\\d+")[, 1],

# strings match the first character

# and see the computer 360 browser collection folder (matching characters)

# and see (13\_Reading Affymetrix chip data.R) Code: P1455

#str\_sub----Extract and replace substrings from a character vector. Extract the substring from the last character

#GSE\_num = "GSE69063",

# Below this line of code is seen in: "F:/Dr. Chen/Dr. Chen wants to lie down/GSE84402" Code: P240~P241

# group\_list <- ifelse(str\_detect(pdata$source\_name\_ch1, "hepatocellular carcinoma"), "tumor", "normal")

group = ifelse(str\_sub(source\_name\_ch1, 1, 1) == "P","Sepsis","Control")) %>%

dplyr::select(sample\_id:group) #%>%

# dplyr::select(2,3,4,1)

GSE95233\_expr <- GSE95233\_expr[ , GSE95233\_targets$sample\_id] %>% as.data.frame()

### View Data

range(GSE95233\_expr)

# exp <- log2(GSE57065\_expr+1)

# range(GSE57065\_expr)

### Evaluate data

if(!dir.exists("figures")) dir.create("figures")

# boxplot(GSE29450\_rma, las = 2, col = rep(c("blue", "red"), each = 10))

# boxplot(GSE95233\_expr, las=2, outline=FALSE,

# col = rep(c("blue", "red"), c(22,51)),

# main = "GSE95233\_expr")

boxplot(GSE95233\_expr, las=2, outline=FALSE,

# col = rep(c("blue", "red"), c(22,51)),

main = "GSE95233\_expr")

dev.off()

source("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Custom\_Functions.R")

targets <- GSE95233\_targets

pdf('./figures/PCA\_GSE95233.pdf')

PCA\_new(exprs(GSE95233\_rma),

nrow(GSE95233\_rma),

group = GSE95233\_targets$group)

dev.off()

# View the original data processing method

GSE95233\_sm@phenoData@data$data\_processing[1]

# GSE95233\_pd$data\_processing[1] # This code is equivalent to the above code

### 6.5 Remove batch effect

## PCA analysis shows that the normal group and the sepsis group have batch effects,

## so the original data is read to remove batch effects

## Use oligo package to read CEL files

library(oligo)

cel\_files <- list.celfiles("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE95233/GSE95233\_RAW/",

full.name = T,

# full.name = F,

listGzipped = T)

# # str\_match returns multiple column matching results, the first column is the overall matching result,

# # the following columns are the matching results for each group.

# c <- str\_match(cel\_files,"GSM\\d+")[,1] # Extract the vector

# c1 <- str\_match(cel\_files,"GSM\\d+") # Extract the list

# c2 <- str\_match(cel\_files,"GSM\\d+")[,2]

# d <- str\_match(GSE54514\_targets\_2$x4, "\\d+")[, 1]

# d <- str\_match(GSE54514\_targets\_2$x4, "\\d+")[, 2]

cel\_files\_1 <- ifelse(str\_match(cel\_files,"GSM\\d+") %in% GSE95233\_targets$sample\_id,TRUE,FALSE)

cel\_files <- cel\_files[cel\_files\_1]

cel\_files1 <- list.celfiles("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE95233/GSE95233\_RAW/",

# full.name = T,

full.name = F,

listGzipped = T)

cel\_files\_11 <- ifelse(str\_match(cel\_files1,"GSM\\d+") %in% GSE95233\_targets$sample\_id,TRUE,FALSE)

cel\_files1 <- cel\_files1[cel\_files\_11]

identical(str\_match(cel\_files,"GSM\\d+")[,1],GSE95233\_targets$sample\_id)

identical(cel\_files\_1,cel\_files\_11)

GSE95233\_targets\_FileName <- GSE95233\_targets %>% mutate(FileName = cel\_files) %>%

relocate(FileName,everything())

## new cel

GSE95233\_cel <- read.celfiles(filenames = GSE95233\_targets\_FileName$FileName,

phenoData = AnnotatedDataFrame(data = GSE95233\_targets\_FileName),

sampleNames = GSE95233\_targets\_FileName$sample\_id)

# GSE95233\_cel1 <- read.celfiles(cel\_files) # This reading method only obtains the expression matrix, without target and sampleNames data

## Save raw data

save(GSE95233\_cel, file = "./GSE95233\_cel.Rdata")

## Lookup batch effect

# cel <- GSE95233\_cel

# cel\_pd <- pData(cel)

# cel@protocolData@data$dates

# cel\_dates <- str\_split(cel@protocolData@data$dates, pattern = "\\s+", simplify = T)[,1]

# \s matches any whitespace character, and is equivalent to [ \f\n\r\t\v]

class(cel\_dates)

table(cel\_dates) # Check batches

# Chip scanning date can be considered as an important batch effect factor

GSE95233\_targets\_batch <- GSE95233\_targets

GSE95233\_targets\_batch$batch <- GSE95233\_cel@protocolData@data$dates %>%

str\_sub(1, 8)

table(GSE95233\_targets\_batch$batch)

## 6.5.1 Do not include grouping factors

library(sva)

mod <- model.matrix(~1, data= GSE95233\_targets\_batch)

batch <- GSE95233\_targets\_batch$batch

GSE95233\_expr <- exprs(GSE95233\_rma) %>% as.data.frame()

range(GSE95233\_expr)

# GSE95233\_expr\_rma <- rownames\_to\_column(GSE95233\_expr, "ID")

# GSE95233\_expr\_rma <- GSE95233\_expr\_rma[!duplicated(GSE95233\_expr\_rma$ID), ]

GSE95233\_combat <- ComBat(dat = GSE95233\_expr,

batch = batch,

mod = mod,

par.prior=TRUE, # Use parameter estimation, false for non-parametric estimation which is more time-consuming

prior.plots=FALSE)

## 6.5.2 Include grouping factors, tell the algorithm what grouping factors we are interested in

mod1 <- model.matrix(~ as.factor(group), data= GSE95233\_targets\_batch)

mod2 <- model.matrix(~1 + group, data= GSE95233\_targets\_batch)

GSE95233\_combat1 <- ComBat(dat = GSE95233\_expr,

batch = batch,

mod = mod1,

par.prior=TRUE,

prior.plots=FALSE)

affy::boxplot(GSE95233\_expr, las = 2,

col = rep(c("blue", "red"), c(22,51)), outline = F)

affy::boxplot(GSE95233\_combat, las = 2,

col = rep(c("blue", "red"), c(22,51)), outline = F)

affy::boxplot(GSE95233\_combat1, las = 2,

col = rep(c("blue", "red"), c(22,51)), outline = F)

dev.off()

## plot

GSE95233\_p0 <- PCA\_new(GSE95233\_expr,

ntop = 10000,

# ntop = nrow(GSE95233\_expr\_n),

show\_name = F)

GSE95233\_p0

GSE95233\_p1 <- PCA\_new(GSE95233\_expr\_n,

ntop = 10000,

group = GSE95233\_targets\_batch$group,

show\_name = F) +

ggtitle("PCA before batch effect correction") +

theme(plot.title = element\_text(hjust = 0.5))

GSE95233\_p1

GSE95233\_p2 <- PCA\_new(GSE95233\_combat,

ntop = 10000,

group = GSE95233\_targets\_batch$group,

show\_name = T)+

ggtitle("PCA before batch effect correction") +

theme(plot.title = element\_text(hjust = 0.5))

GSE95233\_p2

GSE95233\_p3 <- PCA\_new(GSE95233\_combat1,

ntop = 10000,

group = GSE95233\_targets\_batch$group,

show\_name = T)+

ggtitle("PCA before batch effect correction") +

theme(plot.title = element\_text(hjust = 0.5))

GSE95233\_p3

library(cowplot)

GSE95233\_p0123 <- cowplot::plot\_grid(GSE95233\_p0, GSE95233\_p1, GSE95233\_p2, GSE95233\_p3, nrow = 2)

# pdf("batch/PCA\_GSE95233\_batch.pdf", height = 7, width = 15)

GSE95233\_p0123

dev.off()

# After the above analysis, it was found that the data obtained by reading the original data and removing batch effects is the most reasonable,

# so this part of the data is used for subsequent analysis

### 6.10 Annotation ----

GSE95233\_sm@annotation

## GPL570

library(hgu133plus2.db)

library(illuminaHumanv3.db)

ls("package:hgu133plus2.db")

p2s <- toTable(hgu133plus2SYMBOL) #%>%

# set\_names("probe\_id", "symbol")# rename

head(p2s)

## Custom annotation function

annotate\_expr <- function(expr, p2s, fun = mean){# Define three parameters, the column names of p2s are fixed

library(dplyr)

library(tibble)

# expr

expr <- expr %>%

as.data.frame() %>%

tibble::rownames\_to\_column("probe\_id")# Change the row name to a column called "probe\_id"

# annotate

expr\_annotated <- p2s %>%

dplyr::inner\_join(expr, by = "probe\_id") %>% # Take the intersection of expr and p2s

na.omit() %>% # Remove NAs

dplyr::select(-probe\_id) %>% # Remove the "probe\_id" column

stats::aggregate(. ~ symbol, data =., FUN = fun) %>% # For multiple probes corresponding to the same symbol, take the mean of these probes

tibble::column\_to\_rownames("symbol")# Change the "symbol" column to the row name

return(expr\_annotated)# Return this expr\_annotated, and then assign it to annotate\_expr

}

## Annotation

GSE95233\_expr\_anno <- annotate\_expr(GSE95233\_combat1, p2s)

head(GSE95233\_expr\_anno)[, 1:3]

## save data

GSE95233\_expr\_n <- GSE95233\_combat1

GSE95233\_anno <- GSE95233\_expr\_anno

GSE95233\_targets <- GSE95233\_targets\_batch

save(GSE95233\_expr\_n,GSE95233\_targets, GSE95233\_anno,

file = ".\\GSE95233\_processed.Rda")

### 6.11 Differential analysis DEGs ----

library(limma)

# For design

design <- model.matrix(~ group, data = GSE95233\_targets\_batch)

colnames(design)

colnames(design) <- c("Control", "SepsisvsControl")

head(design)

# arrayWeights

expr <- GSE95233\_anno

aw <- arrayWeights(expr, design = design)

aw

barplot(aw)

# Use limma for differential analysis

fit <- lmFit(expr, design, weights = aw)

fit <- contrasts.fit(fit, contrast.matrix)

fit <- eBayes(fit)

## DEG results

results <- summary(decideTests(fit, lfc = 1.2))

results

# number = Inf: all differential analysis results; SepsisvsControl: represents the type to be compared

res\_DEG <- topTable(fit, coef = "SepsisvsControl", number = Inf)

# res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1)# Here p is the corrected p

res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1.2)

GSE95233\_DEG1 <- topTable(fit, coef = 2, n = Inf)

identical(res\_DEG, GSE95233\_DEG1)

GSE95233\_DEG <- topTable(fit, coef = 2, n = Inf) %>% # coef = 1 when there is no contrast.matrix()

rownames\_to\_column(var = "symbol")

# Filter for |logFC|>1.2 and p<0.05 results

GSE95233\_DEG\_p <- GSE95233\_DEG %>%

dplyr::filter(abs(logFC) > 1.2 , P.Value < 0.05)

### save data

save(GSE95233\_DEG, GSE95233\_DEG\_p,

file = ".\\GSE95233\_DEG.Rda")

### 6.12 Draw volcano plot

## plot\_df

plot\_df <- GSE95233\_DEG

# Define volcano plot thresholds

plot\_df$threshold = factor(ifelse(plot\_df$P.Value < 0.05 & abs(plot\_df$logFC) > 1.2,

ifelse(plot\_df$logFC > 1.2 ,'Up','Down'),'NS'),

levels=c('Up','Down','NS'))

# Start plotting

library(ggplot2)

pdf('./figures/Volcano\_GSE95233.pdf', width = 8, height = 8)

ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

dev.off()

p6 <- ggplot(plot\_df, aes(y = logFC, x = -log10(P.Value), color = threshold)) +

geom\_point(alpha = 0.8) +# point size

scale\_color\_manual(values = c("#DC143C", "#143cdc", "#808080")) +# point color

xlab('-log10 (P-value)') +

ylab('log2 (FoldChange)') +

ggtitle("Volcano Plot") +

geom\_hline(yintercept = c(-1.2, 1.2), lty = 3, col = "black", lwd = 0.5) +

geom\_vline(xintercept = -log10(0.05), lty = 3, col = "black", lwd = 0.5) +

theme\_bw() +

theme(legend.title = element\_blank(),

legend.position = "none",

plot.title = element\_text(hjust = 0.5))

p6

save(p6,file = "p6.Rda")

dev.off()

# Merge volcano plots----

rm(list = ls())

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS")

load("./GSE137340/P1.Rda")

load("./GSE69063/P2.Rda")

load("./GSE69528/P3.Rda")

load("./GSE54514/P4.Rda")

load("./GSE57065/P5.Rda")

load("./GSE95233/P6.Rda")

library(cowplot)

# dev.off()

# Coordinate axis horizontal

p <- cowplot::plot\_grid(p1, p2, p3, p4, p5, p6, nrow = 2)# grid plot

p

dev.off()

if(!dir.exists("figures"))dir.create("figures")

# tiff("batch/gene\_wise.tiff", height = 10, width = 10, unit = "in",res = 300)

pdf('F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/figures/Volcano\_batch\_cross.pdf',width = 10,height = 10)

cowplot::plot\_grid(p1, p2, p3, p4, p5, p6, nrow = 2)

dev.off()

# Coordinate axis vertical

library(ggplot2)

p1\_1 <- p1 + coord\_fixed(ratio = 1)+coord\_flip()+

theme(panel.grid.major =element\_blank(), # Remove major grid lines

panel.grid.minor = element\_blank(),# Remove minor grid lines

# panel.border = element\_blank(),# Remove border

panel.background = element\_blank())# Remove background

pdf('./figures/p1\_1.pdf')

p1\_1

dev.off()

p2\_1 <- p2 + coord\_fixed(ratio = 1)+coord\_flip()+

theme(panel.grid.major =element\_blank(), # Remove major grid lines

panel.grid.minor = element\_blank(),# Remove minor grid lines

# panel.border = element\_blank(),# Remove border

panel.background = element\_blank())# Remove background

pdf('./figures/p2\_1.pdf')

p2\_1

dev.off()

p3\_1 <- p3 + coord\_fixed(ratio = 1)+coord\_flip()+

theme(panel.grid.major =element\_blank(), # Remove major grid lines

panel.grid.minor = element\_blank(),# Remove minor grid lines

# panel.border = element\_blank(),# Remove border

panel.background = element\_blank())# Remove background

pdf('./figures/p3\_1.pdf')

p3\_1

dev.off()

p4\_1 <- p4 + coord\_fixed(ratio = 1)+coord\_flip()+

theme(panel.grid.major =element\_blank(), # Remove major grid lines

panel.grid.minor = element\_blank(),# Remove minor grid lines

# panel.border = element\_blank(),# Remove border

panel.background = element\_blank())# Remove background

pdf('./figures/p4\_1.pdf')

p4\_1

dev.off()

p5\_1 <- p5 + coord\_fixed(ratio = 1)+coord\_flip()+

theme(panel.grid.major =element\_blank(), # Remove major grid lines

panel.grid.minor = element\_blank(),# Remove minor grid lines

# panel.border = element\_blank(),# Remove border

panel.background = element\_blank())# Remove background

pdf('./figures/p5\_1.pdf')

p5\_1

dev.off()

p6\_1 <- p6 + coord\_fixed(ratio = 1)+coord\_flip()+

theme(panel.grid.major =element\_blank(), # Remove major grid lines

panel.grid.minor = element\_blank(),# Remove minor grid lines

# panel.border = element\_blank(),# Remove border

panel.background = element\_blank())# Remove background

pdf('./figures/p6\_1.pdf')

p6\_1

dev.off()

p\_1 <- cowplot::plot\_grid(p1\_1, p2\_1, p3\_1, p4\_1, p5\_1, p6\_1, nrow = 2)# grid plot

p\_1

dev.off()

if(!dir.exists("figures"))dir.create("figures")

# tiff("batch/gene\_wise.tiff", height = 10, width = 10, unit = "in",res = 300)

pdf('F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/figures/Volcano\_batch\_vertical.pdf')

# ,width = 15,height = 10)

cowplot::plot\_grid(p1\_1, p2\_1, p3\_1, p4\_1, p5\_1, p6\_1, nrow = 2)

dev.off()

# Same project.R Code: 1863

# # library(cowplot)

# # plot\_grid(plotlist=pp, nrow = 2)

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

getwd()

###

save(p1, p2, p3, p4, p5, p6, file = ".\\P\_batch.Rdata")

save(p1, p2, p3, p4, p5, p6, file = ".\\P\_batch.Rda")

#### 7 Results ----

#### 7.1 RRA ----

rm(list=ls())

library(RobustRankAggreg)

# The principle is very simple: if a gene is ranked high and appears frequently, it is more likely to be a significant differential gene

### load data

setwd("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS")

load("./GSE137340/GSE137340\_DEG.Rda")

load(".//GSE69063/GSE69063\_DEG.Rda")

load("./GSE69528/GSE69528\_DEG.Rda")

load("./GSE54514/GSE54514\_DEG.Rda")

load("./GSE57065/GSE57065\_DEG.Rda")

load("./GSE95233/GSE95233\_DEG.Rda")

# c(GSE137340\_DEG, GSE69063\_DEG, GSE69528\_DEG, GSE54514\_DEG, GSE57065\_DEG, GSE95233\_DEG)

# c(GSE137340\_DEG\_p, GSE69063\_DEG\_p, GSE69528\_DEG\_p, GSE54514\_DEG\_p, GSE57065\_DEG\_p, GSE95233\_DEG\_p)

GSEs <- c("GSE137340", "GSE69063", "GSE69528", "GSE54514", "GSE57065", "GSE95233")

# Filter for up-regulated and down-regulated genes in each dataset

glist\_up <- list()# Create two empty lists

glist\_down <- list()

library(tidyverse)

### 7.1.1 RRA Conventional ----

# for(i in 1:6){ #get Return the value of a named object

# glist\_up[[i]] <- get(paste0(GSEs[i],"\_DEG\_p")) %>% filter(logFC > 0) %>% .[[1]]

# glist\_down[[i]] <- get(paste0(GSEs[i],"\_DEG\_p")) %>% filter(logFC < 0) %>% .[[1]]

# }

#

# # Perform RRA analysis: # aggregateRanks is the most crucial function

# res\_RRA\_up <- aggregateRanks(glist\_up) # Rank genes and sort them, Score is the p-value after RRA aggregation

# res\_RRA\_down <- aggregateRanks(glist\_down)

#

# # Combine up-regulated and down-regulated results

# all\_signi\_result <- bind\_rows(res\_RRA\_up %>% mutate(direction = "up"),

# res\_RRA\_down %>% mutate(direction = "down")) %>%

# filter(Score < 0.05) %>% # Score is the p-value after RRA aggregation

# rownames\_to\_column(var = "SYMBOL") %>%

# write\_tsv("./RRP/all\_signi\_result.tsv")

#

# # Extract the top 10 up-regulated and down-regulated results separately

# top10\_up <- res\_RRA\_up %>% # Extract the top 10

# mutate(across(.fn = as.character)) %>% # across converts each row to as.character

# dplyr::slice(1:10) %>% # Take the first 10 rows

# .[[1]] # Then take the first column

#

# top10\_down <- res\_RRA\_down %>%

# mutate(across(.fn = as.character)) %>%

# dplyr::slice(1:10) %>%

# .[[1]]

#

# top20 <- c(top10\_up, top10\_down)

#

# ### Construct heatmap data

# # Filter for the top genes' symbols and logFC in each GSE

# heatmap\_list <- list()

# for(i in 1:6) {

# heatmap\_list[[i]] <- get(paste0(GSEs[i], "\_DEG")) %>%

# dplyr::select(1, 2) %>% # Select the first two columns: take symbol and logFC

# dplyr::filter(symbol %in% top20) %>% # Filter for symbols in top20

# mutate(GSE = GSEs[i])# Add a GSE column

# }

#

# # Combine the filtered results from all GSEs into a single plotting data

# heatmap\_df <- bind\_rows(heatmap\_list) %>% # This is long data

# # Since not every GSE contains data for the top20 genes, use bind\_rows to combine data, filling blanks with NA

# pivot\_wider(names\_from = GSE, values\_from = logFC) %>% # Convert long data to wide data

# mutate(symbol = factor(symbol, levels = top20)) %>% # Add a column

# arrange(symbol) %>% # Sort by symbol

# column\_to\_rownames(var = "symbol")# Change the symbol column to row name

#

# # Figure

# # install.packages('pheatmap')

# library(pheatmap)

# pdf("./RRP/heatmap\_Figure.pdf", height = 8, width = 8)

# pheatmap::pheatmap(mat = heatmap\_df,

# color = colorRampPalette(c("navy", "white", "firebrick3"))(100),

# cluster\_rows = F,

# cluster\_cols = F,

# display\_numbers = T)

# dev.off()

#

# save(res\_RRA\_up, res\_RRA\_down, heatmap\_list, heatmap\_df, file = "./RRP/RRP.Rda")

### 7.1.2 RRA This project ----

## Filter for adj. p-value < 0.05 results

GSE137340\_DEG\_S <- GSE137340\_DEG %>% dplyr::filter(adj.P.Val < 0.05)

GSE69063\_DEG\_S <- GSE69063\_DEG %>% dplyr::filter(adj.P.Val < 0.05)

GSE69528\_DEG\_S <- GSE69528\_DEG %>% dplyr::filter(adj.P.Val < 0.05)

GSE54514\_DEG\_S <- GSE54514\_DEG %>% dplyr::filter(adj.P.Val < 0.05)

GSE57065\_DEG\_S <- GSE57065\_DEG %>% dplyr::filter(adj.P.Val < 0.05)

GSE95233\_DEG\_S <- GSE95233\_DEG %>% dplyr::filter(adj.P.Val < 0.05)

## Filter for |logFC|>1.2 and adj. p-value<0.05 results

GSE137340\_DEG\_P\_F <- GSE137340\_DEG %>% dplyr::filter(abs(logFC) > 1.2 , adj.P.Val < 0.05)

GSE69063\_DEG\_P\_F <- GSE69063\_DEG %>% dplyr::filter(abs(logFC) > 1.2 , adj.P.Val < 0.05)

GSE69528\_DEG\_P\_F <- GSE69528\_DEG %>% dplyr::filter(abs(logFC) > 1.2 , adj.P.Val < 0.05)

GSE54514\_DEG\_P\_F <- GSE54514\_DEG %>% dplyr::filter(abs(logFC) > 1.2 , adj.P.Val < 0.05)

GSE57065\_DEG\_P\_F <- GSE57065\_DEG %>% dplyr::filter(abs(logFC) > 1.2 , adj.P.Val < 0.05)

GSE95233\_DEG\_P\_F <- GSE95233\_DEG %>% dplyr::filter(abs(logFC) > 1.2 , adj.P.Val < 0.05)

# Filter for up-regulated and down-regulated genes

glist\_up <- list()# Create two empty lists

glist\_down <- list()

## Use RRA package to extract genes with adj. p-value < 0.05

GSEs

GSEs[6]

for(i in 1:6){ #get Return the value of a named object

glist\_up[[i]] <- get(paste0(GSEs[i],"\_DEG\_S")) %>% filter(logFC > 0) %>% .[[1]]

glist\_down[[i]] <- get(paste0(GSEs[i],"\_DEG\_S")) %>% filter(logFC < 0) %>% .[[1]]

}

# Perform RRA analysis: # aggregateRanks is the most crucial function

library(RobustRankAggreg)

res\_RRA\_up\_P <- aggregateRanks(glist\_up) # Rank genes and sort them, Score is the p-value after RRA aggregation

res\_RRA\_down\_P <- aggregateRanks(glist\_down)

# Combine up-regulated and down-regulated results

all\_signi\_result\_P <- bind\_rows(res\_RRA\_up\_P %>% mutate(direction = "up"),

res\_RRA\_down\_P %>% mutate(direction = "down")) %>%

filter(Score < 0.05) %>% # Score is the p-value after RRA aggregation

rownames\_to\_column(var = "SYMBOL") %>%

write\_tsv("./RRP/all\_signi\_result\_P.tsv")

# Use RRA package to extract genes with |logFC|>1.2 and adj. p-value<0.05

for(i in 1:6){ #get Return the value of a named object

glist\_up[[i]] <- get(paste0(GSEs[i],"\_DEG\_P\_F")) %>% filter(logFC > 0) %>% .[[1]]

glist\_down[[i]] <- get(paste0(GSEs[i],"\_DEG\_P\_F")) %>% filter(logFC < 0) %>% .[[1]]

}

# Perform RRA analysis: # aggregateRanks is the most crucial function

res\_RRA\_up\_P\_F <- aggregateRanks(glist\_up) # Rank genes and sort them, Score is the p-value after RRA aggregation

res\_RRA\_down\_P\_F <- aggregateRanks(glist\_down)

# Combine up-regulated and down-regulated results

all\_signi\_result\_P\_F <- bind\_rows(res\_RRA\_up\_P\_F %>% mutate(direction = "up"),

res\_RRA\_down\_P\_F %>% mutate(direction = "down")) %>%

filter(Score < 0.05) %>% # Score is the p-value after RRA aggregation

rownames\_to\_column(var = "SYMBOL") %>%

write\_tsv("./RRP/all\_signi\_result\_P\_F.tsv")

# Extract the top 10 up-regulated and down-regulated results separately

top10\_up\_P\_F <- res\_RRA\_up\_P\_F %>% # Extract the top 10

mutate(across(.fn = as.character)) %>% # across converts each row to as.character

dplyr::slice(1:10) %>% # Take the first 10 rows

.[[1]] # Then take the first column

top10\_down\_P\_F <- res\_RRA\_down\_P\_F %>%

mutate(across(.fn = as.character)) %>%

dplyr::slice(1:10) %>%

.[[1]]

top20\_P\_F <- c(top10\_up\_P\_F, top10\_down\_P\_F)

### Construct heatmap data

# Filter for the top genes' symbols and logFC in each GSE

heatmap\_list\_P\_F <- list()

for(i in 1:6) {

heatmap\_list\_P\_F[[i]] <- get(paste0(GSEs[i], "\_DEG\_P\_F")) %>%

dplyr::select(1, 2) %>% # Select the first two columns: take symbol and logFC

dplyr::filter(symbol %in% top20\_P\_F) %>% # Filter for symbols in top20

mutate(GSE = GSEs[i])# Add a GSE column

}

# Combine the filtered results from all GSEs into a single plotting data

heatmap\_df\_P\_F <- bind\_rows(heatmap\_list\_P\_F) %>% # This is long data

# Since not every GSE contains data for the top20 genes, use bind\_rows to combine data, filling blanks with NA

pivot\_wider(names\_from = GSE, values\_from = logFC) %>% # Convert long data to wide data

# b <- gather(a,key=CIBERSORT,value = Proportion,-c(group,sample)) # Convert wide data to long data

mutate(symbol = factor(symbol, levels = top20\_P\_F)) %>% # Add a column

arrange(symbol) %>% # Sort by symbol

column\_to\_rownames(var = "symbol")# Change the symbol column to row name

# Figure

# install.packages('pheatmap')

library(pheatmap)

pdf("./RRP/heatmap\_Figure\_P\_F.pdf", height = 8, width = 8)

# pheatmap::pheatmap(mat = heatmap\_df\_P\_F,

# color = colorRampPalette(c("navy", "white", "firebrick3"))(100),

# cluster\_rows = F,

# cluster\_cols = F,

# display\_numbers = T)

# dev.off()

pheatmap::pheatmap(mat = heatmap\_df\_P\_F,

color = colorRampPalette(c("ForestGreen", "white", "firebrick3"))(100),

cluster\_rows = F,

cluster\_cols = F,

display\_numbers = T)

dev.off()

save(res\_RRA\_up\_P\_F, res\_RRA\_down\_P\_F, heatmap\_list\_P\_F, heatmap\_df\_P\_F, file = "./RRP/RRP\_P\_F.Rda")

### 7.1.3 RRA: GSEA analysis----

## Load required R packages

rm(list=ls())

library(clusterProfiler)

library(org.Hs.eg.db)

## Load data

input <- read.csv("F:\\HelixCourses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\RRP\\GSEA\_data.csv",sep=",",header=T,check.names=F)

head(input)

# geneList = input[,1]

# class(geneList)

## Sort the geneList by Score1 value

# 1: Extract the Score1 values and store them in a vector

geneList = input[,4]

# 2: Name the geneList with the gene symbols

names(geneList) = as.character(input[,1])

class(geneList)

head(geneList)

## The gene names: ENSEMBLE

# MCEMP1 S100A12 UPP1 HP IRAK3 ARG1

## The corresponding logFC values of the genes

# 15.54516 15.25414 14.32422 14.19723 13.59176 13.10018

## Sort geneList by logFC values in descending order

geneList = sort(geneList, decreasing = TRUE)

class(geneList)

head(geneList)

# F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 2/L64\_Codes/6-2-GSEA\_GSVA\_Analysis/Step02-GSEA\_Plot.R Code: P15

# fpkm\_sd\_sorted = order(fpkm\_sd, decreasing = T)

setwd("./RRP")

##

# gmt file folder name

msigdb\_GMTs <- "msigdb\_v2022.1.Hs\_GMTs"

# The corresponding gene name for the input gene name, ENTREZID corresponds to Entrez Gene ID

# c2.cp.reactome.v2022.1.Hs.symbols.gmt corresponds to Gene Symbol name

# Specify the GMT file used for GSEA enrichment analysis

msigdb <- "c2.cp.reactome.v2022.1.Hs.symbols.gmt"

# Read the GMT file

all\_msigdb <- read.gmt(file.path(msigdb\_GMTs,msigdb))

gsegmt <- GSEA(geneList, TERM2GENE=all\_msigdb, verbose=FALSE)#TERM2GENE, the relationship between TERM and gene name, is generally a two-column data frame format

head(gsegmt)

# Save the Reactome pathway GSEA analysis results to a file

write.table(gsegmt,file="./GSEA\_reactome pathway\_result.txt",sep="\t",

quote=F,row.names = F)

save(gsegmt, file = "./gsegmt.Rda")

### 7.1.4 RRA: GSEA plot----

### Load required R packages

# See F:/HelixCourses/Full Code Breakthrough in Bioinformatics Data Mining/Part 2 of the Bioinformatics System Course/Level 2/L62\_Codes/6-2-GSEA\_GSVA\_Analysis/Step02-GSEA\_Plot.R

rm(list = ls())

# if (!requireNamespace("DOSE", quietly = TRUE)) {

# BiocManager::install("DOSE", ask = F, update = F)

# }

library(clusterProfiler)

library(ggplot2)

## Ridge plot

# Read the file:

setwd("./RRP")

# gsegmt <- read.table(".\\GSEA\_reactome pathway\_result.txt",sep="\t",header=T,check.names=F)

load("./gsegmt.Rda")

# load(file = "./gsegmt.Rda") # Equivalent to the above

if(!dir.exists("figures")) dir.create("figures")

# options("repos" = c(CRAN = "<url id="" type="url" status="" title="" wc="">https://mirrors.tuna.tsinghua.edu.cn/CRAN/</url> "))

# options("BioC\_mirror" = "<url id="" type="url" status="" title="" wc="">http://mirrors.tuna.tsinghua.edu.cn/bioconductor/</url> ")

# if (!requireNamespace("ggridges", quietly = TRUE)) {

# BiocManager::install("ggridges", ask = F, update = F)

# }

pdf("./figures/ridgeplot.pdf", 20, 8)

ridgeplot(gsegmt, showCategory = 6) # showCategory indicates the number of categories to display

# ??ridgeplot : ??Add a function to query

dev.off()

library(ggupset)

library(enrichplot)

## GSEA plot

# KEGG\_result\_df <- as.data.frame(KEGG) # F:/Dr. Chen/Dr. Chen wants to lie down/xena.R Code: 1300

gsegmt\_result\_df <- as.data.frame(gsegmt) # Convert to data frame

write.table(gsegmt\_result\_df, file = "gsegmt\_result\_df.txt",sep="\t", quote=F, col.names=T,row.names = T)

# pp6 <- lapply(1:6, function(i) {

# anno <- gsegmt[i, c("NES", "pvalue", "p.adjust")]

# lab <- paste0(names(anno), "=", round(anno, 3), collapse="\n")

#

# gseaplot2(gsegmt, i, gsegmt[i, 2],base\_size = 5) + #xlab(NULL) +ylab(NULL) + # xlab(NULL),ylab(NULL): Hide axis labels

# # base\_size = 5: Indicates font size

# annotate("text", x=1000, y=0, gsegmt[i, "enrichmentScore"] \*.9, size=2, label = lab, hjust=0, vjust=0)

# annotate Inside the function, "text" is used as the type of annotation, and then x and y specify the position of the text. The content of the text is specified by the label parameter. See “<url id="" type="url" status="" title="" wc="">https://blog.csdn.net/g\_r\_c/article/details/19673625</url> ”

# # box(): Form a complete border (2023.2.8 added)

# # hjust =0 indicates left alignment, hjust =1 indicates right alignment, hjust =0.5 indicates center alignment

# }) # hjust =0 indicates left alignment, hjust =1 indicates right alignment, hjust =0.5 indicates center alignment

#

# # # pp and pp1 are both empty, the reason is that gseaplot2 draws three plots, and annotate() does not know which plot to add annotations to

# #

# pp2 <- lapply(1:3, function(i) {

# anno <- gsegmt[i, c("NES", "pvalue", "p.adjust")]

# lab <- paste0(names(anno), "=", round(anno, 3), collapse="\n")

#

# gsearank(gsegmt, i, gsegmt[i, 2]) + xlab(NULL) +ylab(NULL) +

# annotate("text", 0, gsegmt[i, "enrichmentScore"] \*.9, label = lab, hjust=0, vjust=0)

# })

# plot\_grid(plotlist=pp2, nrow = 2)

# # pp2 can be plotted, the reason is that gsearank draws one plot, and annotate() does not have to worry about which plot to add annotations to

# #

# # # The following code is to break down the steps above:

# anno <- gsegmt[1, c("NES", "pvalue", "p.adjust")]

# lab <- paste0(names(anno), "=", round(anno, 3), collapse="\n")

# # pvalue\_table = lab

# # P4 <- gseaplot2(gsegmt, 1, gsegmt[1, 2])

# P5\_0 <- gseaplot2(gsegmt, 1, gsegmt[1, 2],base\_size = 6,subplots = 1)+

# annotate("text", x=500 ,y=0.2, size=5, label = lab)

# P5\_0 <- P5\_0 + gseaplot2(gsegmt, 1, gsegmt[1, 2],base\_size = 6,subplots = 2)+plot\_layout(nrow = 2)

# P5\_1 <-gseaplot2(gsegmt, 1, gsegmt[1, 2],base\_size = 6,subplots = 2)

# P5\_2 <-gseaplot2(gsegmt, 1, gsegmt[1, 2],base\_size = 6,subplots = 3)

# P5\_0 + P5\_1 + P5\_2 + plot\_layout(ncol = 1)

# plot\_grid(P5\_0+P5\_1+P5\_2, nrow = 3)

#

# P6 <- enrichplot::gseaplot2(gsegmt, 1, gsegmt[1, 2],base\_size = 6,pvalue\_table = TRUE)+

# annotate("text", x=2000 , y=6,size=5, label = lab, hjust=1, vjust=1) # Draw three plots

#

# # R how to install Git: Website: <url id="" type="url" status="" title="" wc="">https://www.jianshu.com/p/4d1388c82fc0</url>

# remotes::install\_git('<url id="" type="url" status="" title="" wc="">https://gitee.com/swcyo/myenrichplot/</url> ') # Install the improved gseaplot2() plotting package via GitHub

# remotes::install\_git('<url id="" type="url" status="" title="" wc="">https://gitee.com/swcyo/Rtoolbox</url> ') # Install the replotGSEA() plotting package via GitHub

# remotes::install\_github("swcyo/myenrichplot")

# remotes::install\_local("C:/Users/admin/Desktop/Rtoolbox-master\_2.zip",upgrade = F,dependencies = T)

library(myenrichplot)

pp7 <- lapply(1:6, function(i) {

myenrichplot::gseaplot2(gsegmt, i, gsegmt[i, 2], pvalue\_table = T) #xlab(NULL),ylab(NULL): Hide axis labels

# gseaplot2(gsegmt, i, gsegmt[i, 2],base\_size = 5) + #xlab(NULL) +ylab(NULL) + # xlab(NULL),ylab(NULL): Hide axis labels

}) # hjust =0 indicates left alignment, hjust =1 indicates right alignment, hjust =0.5 indicates center alignment

pp8 <- lapply(1:6, function(i) {

enrichplot::gseaplot2(gsegmt, i, gsegmt[i, 2], pvalue\_table = T)

# gseaplot2(gsegmt, i, gsegmt[i, 2],base\_size = 5) + #xlab(NULL) +ylab(NULL) + # xlab(NULL),ylab(NULL): Hide axis labels

}) # hjust =0 indicates left alignment, hjust =1 indicates right alignment, hjust =0.5 indicates center alignment

# library(cowplot)

# pdf("./figures/GSEA\_C2\_pathway.pdf", 20, 8)

# plot\_grid(plotlist=pp7, nrow = 2)

# dev.off()

# cowplot::plot\_grid(p1, p2, p3, p4, p5, p6, nrow = 2) # Same project.R Code: 1576

pdf("./figures/GSEA\_C2\_pathway.pdf",14,4.5)

lapply(1:6, function(i) {

myenrichplot::gseaplot2(gsegmt, i, gsegmt[i, 2], pvalue\_table = T) #xlab(NULL),ylab(NULL): Hide axis labels

})

dev.off()

# pdf("./figures/Boxplots.PDF", 15, 6) # See this project Code: 52~59

# for (i in 1:8) {

# boxplot(get(GSE\_expr[i]),

# las=2,

# outline=FALSE,

# main = paste0("Boxplot for ", GSEs[i]))

# }

# dev.off()

### 8.2 cibersort\_GSE69063 barplot and boxplot----

## 8.2.1 cibersort\_GSE69063 barplot----

rm(list=ls())

library(e1071)

library(parallel)

library(preprocessCore)

library(tidyverse)

# if(!dir.exists("CIBERSORT")) dir.create("CIBERSORT")

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/CIBERSORT")

source("./CIBERSORT.R")

if(!dir.exists("CIBERSORT\_GSE69063"))dir.create("CIBERSORT\_GSE69063")

# Assign the file name "LM22.txt" to sig\_matrix, and the program will automatically read this file in the directory

sig\_matrix <- "LM22.txt" # "LM22.txt" is the annotation file for 22 immune cells

load("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE69063/GSE69063\_processed.Rda")

# write.table(GSE69063\_anno, file = "GSE69063\_anno.txt",sep="\t", quote=F, col.names=T,row.names = T)

mixture\_file = 'GSE69063\_anno.txt' # Expression data for the GSE137340 dataset (note to adjust the first row of the table)

# perm indicates the number of calculations; the more times, the more stable the result

res\_cibersort <- CIBERSORT(sig\_matrix,

mixture\_file,

perm=100, # perm, the number of permutation tests, default is 0; significant p-values can be calculated when greater than 100

QN=TRUE) # QN, whether to normalize the input expression matrix

# Equivalent to calculating 100 times continuously to see if the result is random in 100 times

# Generate a normalization function normalize, create a function

normalize <- function(x){# (x - minimum of x) / (maximum of x - minimum of x)

return((x-min(x))/(max(x)-min(x)))}

# Normalize CIBERSORT scores using normalize

CIBERSORT\_Score <- res\_cibersort

CIBERSORT\_Score\_GSE69063 <- normalize(CIBERSORT\_Score)

save(CIBERSORT\_Score\_GSE69063, CIBERSORT\_Score,

file = "./CIBERSORT\_GSE69063/CIBERSORT\_Score\_GSE69063.Rdata") # Save intermediate file

load("./CIBERSORT\_GSE69063/CIBERSORT\_Score\_GSE69063.Rdata")

CIBERSORT\_Score\_GSE69063 <- CIBERSORT\_Score\_GSE69063[,1:22]

# colSums adds up the columns

ciber.res <- CIBERSORT\_Score\_GSE69063[,colSums(CIBERSORT\_Score\_GSE69063) > 0] # Remove cells with zero abundance

# Visualization (by Teacher Achim)

mycol <- ggplot2::alpha(rainbow(ncol(ciber.res)), 0.7) # Create a rainbow color palette (with 70% transparency)

# alpha: Add transparency, the range of alpha is from 0 to 1, 1 is the original color, the larger the number, the darker the color

par(bty="o", mgp = c(2.5,0.3,0), mar = c(2.1,4.1,2.1,10.1),tcl=-.25,las = 1,xpd = F)

# las: Can only be one of 0, 1, 2, 3, used to indicate the direction of the tick mark values. 0 means always parallel to the axis; 1 means always horizontal direction; 2 means always perpendicular to the axis; 3 means always vertical direction

# tcl: Can also be used to set the length of the tick marks, but the unit is different from tck. Its default value is -0.5

# bty: The parameter value is a string type, used to limit the type of the plot border. If the value of bty is "o" (default value: i.e., borders on all four sides), "l", "7", "c", "u", or "]" among any one, the corresponding border type is similar to the shape of the letter.

# If the value of bty is "n", it means no border.

# mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.

# xpd: Logical parameter or NA, FALSE means only display text within the plot region, TRUE means only display text within the figure region, NA means display text within the entire device region;

barplot(as.matrix(t(ciber.res)),

border = NA, # No border for the bars

names.arg = rep("",nrow(ciber.res)), # No sample names on the x-axis

yaxt = "n", # Do not draw the y-axis for now

ylab = "Relative percentage", # Modify the y-axis label

col = mycol) # Use the rainbow color palette

axis(side = 2, at = c(0,0.2,0.4,0.6,0.8,1), # Add percentages to the y-axis

labels = c("0%","20%","40%","60%","80%","100%"))

legend(par("usr")[2]-2, # Adjust the legend position according to the actual plot (here -20)

par("usr")[4],

legend = colnames(ciber.res),

xpd = T,

fill = mycol,

cex = 0.7,

border = NA,

y.intersp = 1,

x.intersp = 0.2,

bty = "n")

dev.off() # Close the plot

## 8.2.2 cibersort\_GSE69063 boxplot----

library(tidyverse)

a <- CIBERSORT\_Score\_GSE69063 %>% as.data.frame()

b <- GSE69063\_targets %>% dplyr::select(1,4) # Pipeline function (Ctrl + Shift + M)

identical(rownames(a),rownames(b))# Check if the row names of the two tables are in the same order

b$group <- as.factor(b$group) # Convert the group column in data frame b to a factor type

class(b$group)

# a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

# Remove cells with zero abundance

ciber.res <- CIBERSORT\_Score\_GSE69063[,colSums(CIBERSORT\_Score\_GSE69063) > 0] # Remove cells with zero abundance

a <- a [,colnames(ciber.res)] %>% rownames\_to\_column("sample")

a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

library(ggsci)

library(tidyr)

library(ggpubr)

# b <- b[rownames(a),]

# identical(rownames(a),rownames(b))

# # Take the intersection of row names

# comgene <- intersect(rownames(a),rownames(b))

# a <- a[comgene,]

# class(a)# Check data type

# class(comgene)

# b <- b[comgene,]

# identical(rownames(a),rownames(b))# Check if multiple elements are identical

b <- gather(a,key=CIBERSORT,value = Proportion,-c(group,sample)) # Convert wide data to long data (code: 2119)

# pivot\_wider(names\_from = GSE, values\_from = logFC) # Convert long data to wide data (code: 1802)

pdf("./CIBERSORT\_GSE69063/boxplot\_GSE69063.pdf")

ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10), # Font size of all text in the plot

axis.text.x = element\_text(angle=45, hjust=1))

# hjust =0 means left-aligned, hjust =1 means right-aligned, hjust =0.5 means centered

dev.off()

CB2 <- ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10),

axis.text.x = element\_text(angle=45, hjust=1))

CB2

dev.off()

### 8.3 cibersort\_GSE69528 barplot and boxplot----

## 8.3.1 cibersort\_GSE69528 barplot----

rm(list=ls())

library(e1071)

library(parallel)

library(preprocessCore)

library(tidyverse)

# if(!dir.exists("CIBERSORT")) dir.create("CIBERSORT")

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/CIBERSORT")

source("./CIBERSORT.R")

if(!dir.exists("CIBERSORT\_GSE69528"))dir.create("CIBERSORT\_GSE69528")

# Assign the file name "LM22.txt" to sig\_matrix, and the program will automatically read this file in the directory

sig\_matrix <- "LM22.txt" # "LM22.txt" is the annotation file for 22 immune cells

load("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE69528/GSE69528\_processed.Rda")

# write.table(GSE69528\_anno, file = "GSE69528\_anno.txt",sep="\t", quote=F, col.names=T,row.names = T)

mixture\_file = 'GSE69528\_anno.txt' # Expression data for the GSE137340 dataset (note to adjust the first row of the table)

# perm indicates the number of calculations; the more times, the more stable the result

res\_cibersort <- CIBERSORT(sig\_matrix,

mixture\_file,

perm=100, # perm, the number of permutation tests, default is 0; significant p-values can be calculated when greater than 100

QN=TRUE) # QN, whether to normalize the input expression matrix

# Equivalent to calculating 100 times continuously to see if the result is random in 100 times

# Generate a normalization function normalize, create a function

normalize <- function(x){# (x - minimum of x) / (maximum of x - minimum of x)

return((x-min(x))/(max(x)-min(x)))}

# Normalize CIBERSORT scores using normalize

CIBERSORT\_Score <- res\_cibersort

CIBERSORT\_Score\_GSE69528 <- normalize(CIBERSORT\_Score)

save(CIBERSORT\_Score\_GSE69528, CIBERSORT\_Score,

file = "./CIBERSORT\_GSE69528/CIBERSORT\_Score\_GSE69528.Rdata") # Save intermediate file

load("./CIBERSORT\_GSE69528/CIBERSORT\_Score\_GSE69528.Rdata")

CIBERSORT\_Score\_GSE69528<- CIBERSORT\_Score\_GSE69528[,1:22]

# colSums adds up the columns

ciber.res <- CIBERSORT\_Score\_GSE69528[,colSums(CIBERSORT\_Score\_GSE69528) > 0] # Remove cells with zero abundance

# Visualization (by Teacher Achim)

mycol <- ggplot2::alpha(rainbow(ncol(ciber.res)), 0.7) # Create a rainbow color palette (with 70% transparency)

# alpha: Add transparency, the range of alpha is from 0 to 1, 1 is the original color, the larger the number, the darker the color

par(bty="o", mgp = c(2.5,0.3,0), mar = c(2.1,4.1,2.1,10.1),tcl=-.25,las = 1,xpd = F)

# las: Can only be one of 0, 1, 2, 3, used to indicate the direction of the tick mark values. 0 means always parallel to the axis; 1 means always horizontal direction; 2 means always perpendicular to the axis; 3 means always vertical direction

# tcl: Can also be used to set the length of the tick marks, but the unit is different from tck. Its default value is -0.5

# bty: The parameter value is a string type, used to limit the type of the plot border. If the value of bty is "o" (default value: i.e., borders on all four sides), "l", "7", "c", "u", or "]" among any one, the corresponding border type is similar to the shape of the letter.

# If the value of bty is "n", it means no border.

# mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.

# xpd: Logical parameter or NA, FALSE means only display text within the plot region, TRUE means only display text within the figure region, NA means display text within the entire device region;

barplot(as.matrix(t(ciber.res)),

border = NA, # No border for the bars

names.arg = rep("",nrow(ciber.res)), # No sample names on the x-axis

yaxt = "n", # Do not draw the y-axis for now

ylab = "Relative percentage", # Modify the y-axis label

col = mycol) # Use the rainbow color palette

axis(side = 2, at = c(0,0.2,0.4,0.6,0.8,1), # Add percentages to the y-axis

labels = c("0%","20%","40%","60%","80%","100%"))

legend(par("usr")[2]-2, # Adjust the legend position according to the actual plot (here -20)

par("usr")[4],

legend = colnames(ciber.res),

xpd = T,

fill = mycol,

cex = 0.6,

border = NA,

y.intersp = 1,

x.intersp = 0.2,

bty = "n")

dev.off() # Close the plot

## 8.3.2 cibersort\_GSE69528 boxplot----

library(tidyverse)

a <- CIBERSORT\_Score\_GSE69528 %>% as.data.frame()

b <- GSE69528\_targets %>% dplyr::select(1,4) # Pipeline function (Ctrl + Shift + M)

identical(rownames(a),rownames(b))# Check if the row names of the two tables are in the same order

b$group <- as.factor(b$group) # Convert the group column in data frame b to a factor type

class(b$group)

# a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

# Remove cells with zero abundance

ciber.res <- CIBERSORT\_Score\_GSE69528[,colSums(CIBERSORT\_Score\_GSE69528) > 0] # Remove cells with zero abundance

a <- a [,colnames(ciber.res)] %>% rownames\_to\_column("sample")

a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

library(ggsci)

library(tidyr)

library(ggpubr)

# b <- b[rownames(a),]

# identical(rownames(a),rownames(b))

# # Take the intersection of row names

# comgene <- intersect(rownames(a),rownames(b))

# a <- a[comgene,]

# class(a)# Check data type

# class(comgene)

# b <- b[comgene,]

# identical(rownames(a),rownames(b))# Check if multiple elements are identical

b <- gather(a,key=CIBERSORT,value = Proportion,-c(group,sample)) # Convert wide data to long data (code: 2119)

# pivot\_wider(names\_from = GSE, values\_from = logFC) # Convert long data to wide data (code: 1802)

pdf("./CIBERSORT\_GSE69528/boxplot\_GSE69528.pdf")

ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10), # Font size of all text in the plot

axis.text.x = element\_text(angle=45, hjust=1))

# hjust =0 means left-aligned, hjust =1 means right-aligned, hjust =0.5 means centered

dev.off()

CB3 <- ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10),

axis.text.x = element\_text(angle=45, hjust=1))

CB3

dev.off()

### 8.4 cibersort\_GSE54514 barplot and boxplot----

## 8.4.1 cibersort\_GSE54514 barplot----

rm(list=ls())

library(e1071)

library(parallel)

library(preprocessCore)

library(tidyverse)

# if(!dir.exists("CIBERSORT")) dir.create("CIBERSORT")

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/CIBERSORT")

source("./CIBERSORT.R")

if(!dir.exists("CIBERSORT\_GSE54514"))dir.create("CIBERSORT\_GSE54514")

# Assign the file name "LM22.txt" to sig\_matrix, and the program will automatically read this file in the directory

sig\_matrix <- "LM22.txt" # "LM22.txt" is the annotation file for 22 immune cells

load("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE54514/GSE54514\_processed.Rda")

# write.table(GSE54514\_anno, file = "GSE54514\_anno.txt",sep="\t", quote=F, col.names=T,row.names = T)

mixture\_file = 'GSE54514\_anno.txt' # Expression data for the GSE137340 dataset (note to adjust the first row of the table)

# perm indicates the number of calculations; the more times, the more stable the result

res\_cibersort <- CIBERSORT(sig\_matrix,

mixture\_file,

perm=100, # perm, the number of permutation tests, default is 0; significant p-values can be calculated when greater than 100

QN=TRUE) # QN, whether to normalize the input expression matrix

# Equivalent to calculating 100 times continuously to see if the result is random in 100 times

# Generate a normalization function normalize, create a function

normalize <- function(x){# (x - minimum of x) / (maximum of x - minimum of x)

return((x-min(x))/(max(x)-min(x)))}

# Normalize CIBERSORT scores using normalize

CIBERSORT\_Score <- res\_cibersort

CIBERSORT\_Score\_GSE54514 <- normalize(CIBERSORT\_Score)

save(CIBERSORT\_Score\_GSE54514, CIBERSORT\_Score,

file = "./CIBERSORT\_GSE54514/CIBERSORT\_Score\_GSE54514.Rdata") # Save intermediate file

load("./CIBERSORT\_GSE54514/CIBERSORT\_Score\_GSE54514.Rdata")

CIBERSORT\_Score\_GSE54514<- CIBERSORT\_Score\_GSE54514[,1:22]

# colSums adds up the columns

ciber.res <- CIBERSORT\_Score\_GSE54514[,colSums(CIBERSORT\_Score\_GSE54514) > 0] # Remove cells with zero abundance

# Visualization (by Teacher Achim)

mycol <- ggplot2::alpha(rainbow(ncol(ciber.res)), 0.7) # Create a rainbow color palette (with 70% transparency)

# alpha: Add transparency, the range of alpha is from 0 to 1, 1 is the original color, the larger the number, the darker the color

par(bty="o", mgp = c(2.5,0.3,0), mar = c(2.1,4.1,2.1,10.1),tcl=-.25,las = 1,xpd = F)

# las: Can only be one of 0, 1, 2, 3, used to indicate the direction of the tick mark values. 0 means always parallel to the axis; 1 means always horizontal direction; 2 means always perpendicular to the axis; 3 means always vertical direction

# tcl: Can also be used to set the length of the tick marks, but the unit is different from tck. Its default value is -0.5

# bty: The parameter value is a string type, used to limit the type of the plot border. If the value of bty is "o" (default value: i.e., borders on all four sides), "l", "7", "c", "u", or "]" among any one, the corresponding border type is similar to the shape of the letter.

# If the value of bty is "n", it means no border.

# mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.

# xpd: Logical parameter or NA, FALSE means only display text within the plot region, TRUE means only display text within the figure region, NA means display text within the entire device region;

barplot(as.matrix(t(ciber.res)),

border = NA, # No border for the bars

names.arg = rep("",nrow(ciber.res)), # No sample names on the x-axis

yaxt = "n", # Do not draw the y-axis for now

ylab = "Relative percentage", # Modify the y-axis label

col = mycol) # Use the rainbow color palette

axis(side = 2, at = c(0,0.2,0.4,0.6,0.8,1), # Add percentages to the y-axis

labels = c("0%","20%","40%","60%","80%","100%"))

legend(par("usr")[2]-2, # Adjust the legend position according to the actual plot (here -20)

par("usr")[4],

legend = colnames(ciber.res),

xpd = T,

fill = mycol,

cex = 0.7,

border = NA,

y.intersp = 1,

x.intersp = 0.2,

bty = "n")

dev.off() # Close the plot

## 8.4.2 cibersort\_GSE54514 boxplot----

library(tidyverse)

a <- CIBERSORT\_Score\_GSE54514 %>% as.data.frame()

b <- GSE54514\_targets # Pipeline function (Ctrl + Shift + M)

identical(rownames(a),rownames(b))# Check if the row names of the two tables are in the same order

b$group <- as.factor(b$group) # Convert the group column in data frame b to a factor type

class(b$group)

# a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

# Remove cells with zero abundance

ciber.res <- CIBERSORT\_Score\_GSE54514[,colSums(CIBERSORT\_Score\_GSE54514) > 0] # Remove cells with zero abundance

a <- a [,colnames(ciber.res)] %>% rownames\_to\_column("sample")

a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

library(ggsci)

library(tidyr)

library(ggpubr)

# b <- b[rownames(a),]

# identical(rownames(a),rownames(b))

# # Take the intersection of row names

# comgene <- intersect(rownames(a),rownames(b))

# a <- a[comgene,]

# class(a)# Check data type

# class(comgene)

# b <- b[comgene,]

# identical(rownames(a),rownames(b))# Check if multiple elements are identical

b <- gather(a,key=CIBERSORT,value = Proportion,-c(group,sample)) # Convert wide data to long data (code: 2119)

# pivot\_wider(names\_from = GSE, values\_from = logFC) # Convert long data to wide data (code: 1802)

pdf("./CIBERSORT\_GSE54514/boxplot\_GSE54514.pdf")

ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10), # Font size of all text in the plot

axis.text.x = element\_text(angle=45, hjust=1))

# hjust =0 means left-aligned, hjust =1 means right-aligned, hjust =0.5 means centered

dev.off()

CB4 <- ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10),

axis.text.x = element\_text(angle=45, hjust=1))

CB4

dev.off()

### 8.5 cibersort\_GSE57065 barplot and boxplot----

## 8.5.1 cibersort\_GSE57065 barplot----

rm(list=ls())

library(e1071)

library(parallel)

library(preprocessCore)

library(tidyverse)

# if(!dir.exists("CIBERSORT")) dir.create("CIBERSORT")

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/CIBERSORT")

source("./CIBERSORT.R")

if(!dir.exists("CIBERSORT\_GSE57065"))dir.create("CIBERSORT\_GSE57065")

# Assign the file name "LM22.txt" to sig\_matrix, and the program will automatically read this file in the directory

sig\_matrix <- "LM22.txt" # "LM22.txt" is the annotation file for 22 immune cells

load("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE57065/GSE57065\_processed.Rda")

# write.table(GSE57065\_anno, file = "GSE57065\_anno.txt",sep="\t", quote=F, col.names=T,row.names = T)

mixture\_file = 'GSE57065\_anno.txt' # Expression data for the GSE137340 dataset (note to adjust the first row of the table)

# perm indicates the number of calculations; the more times, the more stable the result

res\_cibersort <- CIBERSORT(sig\_matrix,

mixture\_file,

perm=100, # perm, the number of permutation tests, default is 0; significant p-values can be calculated when greater than 100

QN=TRUE) # QN, whether to normalize the input expression matrix

# Equivalent to calculating 100 times continuously to see if the result is random in 100 times

# Generate a normalization function normalize, create a function

normalize <- function(x){# (x - minimum of x) / (maximum of x - minimum of x)

return((x-min(x))/(max(x)-min(x)))}

# Normalize CIBERSORT scores using normalize

CIBERSORT\_Score <- res\_cibersort

CIBERSORT\_Score\_GSE57065 <- normalize(CIBERSORT\_Score)

save(CIBERSORT\_Score\_GSE57065, CIBERSORT\_Score,

file = "./CIBERSORT\_GSE57065/CIBERSORT\_Score\_GSE57065.Rdata") # Save intermediate file

load("./CIBERSORT\_GSE57065/CIBERSORT\_Score\_GSE57065.Rdata")

CIBERSORT\_Score\_GSE57065<- CIBERSORT\_Score\_GSE57065[,1:22]

# colSums adds up the columns

ciber.res <- CIBERSORT\_Score\_GSE57065[,colSums(CIBERSORT\_Score\_GSE57065) > 0] # Remove cells with zero abundance

# Visualization (by Teacher Achim)

mycol <- ggplot2::alpha(rainbow(ncol(ciber.res)), 0.7) # Create a rainbow color palette (with 70% transparency)

# alpha: Add transparency, the range of alpha is from 0 to 1, 1 is the original color, the larger the number, the darker the color

par(bty="o", mgp = c(2.5,0.3,0), mar = c(2.1,4.1,2.1,10.1),tcl=-.25,las = 1,xpd = F)

# las: Can only be one of 0, 1, 2, 3, used to indicate the direction of the tick mark values. 0 means always parallel to the axis; 1 means always horizontal direction; 2 means always perpendicular to the axis; 3 means always vertical direction

# tcl: Can also be used to set the length of the tick marks, but the unit is different from tck. Its default value is -0.5

# bty: The parameter value is a string type, used to limit the type of the plot border. If the value of bty is "o" (default value: i.e., borders on all four sides), "l", "7", "c", "u", or "]" among any one, the corresponding border type is similar to the shape of the letter.

# If the value of bty is "n", it means no border.

# mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.

# xpd: Logical parameter or NA, FALSE means only display text within the plot region, TRUE means only display text within the figure region, NA means display text within the entire device region;

barplot(as.matrix(t(ciber.res)),

border = NA, # No border for the bars

names.arg = rep("",nrow(ciber.res)), # No sample names on the x-axis

yaxt = "n", # Do not draw the y-axis for now

ylab = "Relative percentage", # Modify the y-axis label

col = mycol) # Use the rainbow color palette

axis(side = 2, at = c(0,0.2,0.4,0.6,0.8,1), # Add percentages to the y-axis

labels = c("0%","20%","40%","60%","80%","100%"))

legend(par("usr")[2]-2, # Adjust the legend position according to the actual plot (here -20)

par("usr")[4],

legend = colnames(ciber.res),

xpd = T,

fill = mycol,

cex = 0.7,

border = NA,

y.intersp = 1,

x.intersp = 0.2,

bty = "n")

dev.off() # Close the plot

## 8.5.2 cibersort\_GSE57065 boxplot----

library(tidyverse)

a <- CIBERSORT\_Score\_GSE57065 %>% as.data.frame() # Pipeline function (Ctrl + Shift + M)

b <- GSE57065\_targets

identical(rownames(a),rownames(b)) # Check if the row names of the two tables are in the same order

b$group <- as.factor(b$group) # Convert the group column in data frame b to a factor type

class(b$group)

# a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

# Remove cells with zero abundance

ciber.res <- CIBERSORT\_Score\_GSE57065[,colSums(CIBERSORT\_Score\_GSE57065) > 0] # Remove cells with zero abundance

a <- a [,colnames(ciber.res)] %>% rownames\_to\_column("sample")

a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

library(ggsci)

library(tidyr)

library(ggpubr)

# b <- b[rownames(a),]

# identical(rownames(a),rownames(b))

# # Take the intersection of row names

# comgene <- intersect(rownames(a),rownames(b))

# a <- a[comgene,]

# class(a)# Check data type

# class(comgene)

# b <- b[comgene,]

# identical(rownames(a),rownames(b))# Check if multiple elements are identical

b <- gather(a,key=CIBERSORT,value = Proportion,-c(group,sample)) # Convert wide data to long data (code: 2119)

# pivot\_wider(names\_from = GSE, values\_from = logFC) # Convert long data to wide data (code: 1802)

pdf("./CIBERSORT\_GSE57065/boxplot\_GSE57065.pdf")

ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10), # Font size of all text in the plot

axis.text.x = element\_text(angle=45, hjust=1))

# hjust =0 means left-aligned, hjust =1 means right-aligned, hjust =0.5 means centered

dev.off()

CB5 <- ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10),

axis.text.x = element\_text(angle=45, hjust=1))

CB5

dev.off()

### 8.6 cibersort\_GSE95233 barplot and boxplot----

## 8.6.1 cibersort\_GSE95233 barplot----

rm(list=ls())

# options("BioC\_mirror" = "http://mirrors.tuna.tsinghua.edu.cn/bioconductor/ ")

# if (!requireNamespace("e1071", quietly = TRUE)) {

# BiocManager::install("e1071", ask = F, update = F)

# }

library(e1071)

library(parallel)

library(preprocessCore)

library(tidyverse)

# if(!dir.exists("CIBERSORT")) dir.create("CIBERSORT")

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/CIBERSORT")

source("./CIBERSORT.R")

if(!dir.exists("CIBERSORT\_GSE95233"))dir.create("CIBERSORT\_GSE95233")

# Assign the file name "LM22.txt" to sig\_matrix, and the program will automatically read this file in the directory

sig\_matrix <- "LM22.txt" # "LM22.txt" is the annotation file for 22 immune cells

load("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/GSE95233/GSE95233\_processed.Rda")

# write.table(GSE95233\_anno, file = "GSE95233\_anno.txt",sep="\t", quote=F, col.names=T,row.names = T)

mixture\_file = 'GSE95233\_anno.txt' # Expression data for the GSE137340 dataset (note to adjust the first row of the table)

# perm indicates the number of calculations; the more times, the more stable the result

res\_cibersort <- CIBERSORT(sig\_matrix,

mixture\_file,

perm=100, # perm, the number of permutation tests, default is 0; significant p-values can be calculated when greater than 100

QN=TRUE) # QN, whether to normalize the input expression matrix

# Equivalent to calculating 100 times continuously to see if the result is random in 100 times

# Generate a normalization function normalize, create a function

normalize <- function(x){# (x - minimum of x) / (maximum of x - minimum of x)

return((x-min(x))/(max(x)-min(x)))}

# Normalize CIBERSORT scores using normalize

CIBERSORT\_Score <- res\_cibersort

CIBERSORT\_Score\_GSE95233 <- normalize(CIBERSORT\_Score)

save(CIBERSORT\_Score\_GSE95233, CIBERSORT\_Score, file = "./CIBERSORT\_GSE95233/CIBERSORT\_Score\_GSE95233.Rdata") # Save intermediate file

load("./CIBERSORT\_GSE95233/CIBERSORT\_Score\_GSE95233.Rdata")

CIBERSORT\_Score\_GSE95233<- CIBERSORT\_Score\_GSE95233[,1:22]

# colSums adds up the columns

ciber.res <- CIBERSORT\_Score\_GSE95233[,colSums(CIBERSORT\_Score\_GSE95233) > 0] # Remove cells with zero abundance

# Visualization (by Teacher Achim)

mycol <- ggplot2::alpha(rainbow(ncol(ciber.res)), 0.7) # Create a rainbow color palette (with 70% transparency)

# alpha: Add transparency, the range of alpha is from 0 to 1, 1 is the original color, the larger the number, the darker the color

par(bty="o", mgp = c(2.5,0.3,0), mar = c(2.1,4.1,2.1,10.1),tcl=-.25,las = 1,xpd = F)

# las: Can only be one of 0, 1, 2, 3, used to indicate the direction of the tick mark values. 0 means always parallel to the axis; 1 means always horizontal direction; 2 means always perpendicular to the axis; 3 means always vertical direction

# tcl: Can also be used to set the length of the tick marks, but the unit is different from tck. Its default value is -0.5

# bty: The parameter value is a string type, used to limit the type of the plot border. If the value of bty is "o" (default value: i.e., borders on all four sides), "l", "7", "c", "u", or "]" among any one, the corresponding border type is similar to the shape of the letter.

# If the value of bty is "n", it means no border.

# mar: A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(5, 4, 4, 2) + 0.1.

# xpd: Logical parameter or NA, FALSE means only display text within the plot region, TRUE means only display text within the figure region

# xpd: Logical parameter or NA, FALSE means only display text within the plot region,

# TRUE means only display text within the figure region, NA means display text within the entire device region

barplot(as.matrix(t(ciber.res)),

border = NA, # No border for the bars

names.arg = rep("",nrow(ciber.res)), # No sample names on the x-axis

yaxt = "n", # Do not draw the y-axis for now

ylab = "Relative percentage", # Modify the y-axis label

col = mycol) # Use the rainbow color palette

axis(side = 2, at = c(0,0.2,0.4,0.6,0.8,1), # Add percentages to the y-axis

labels = c("0%","20%","40%","60%","80%","100%"))

legend(par("usr")[2]-2, # Adjust the legend position according to the actual plot (here -20)

par("usr")[4],

legend = colnames(ciber.res),

xpd = TRUE,

fill = mycol,

cex = 0.7,

border = NA,

y.intersp = 1,

x.intersp = 0.2,

bty = "n")

dev.off() # Close the plot

## 8.6.2 cibersort\_GSE95233 boxplot----

library(tidyverse)

a <- CIBERSORT\_Score\_GSE95233 %>% as.data.frame()

b <- GSE95233\_targets

identical(rownames(a),rownames(b)) # Check if the row names of the two tables are in the same order

b$group <- as.factor(b$group) # Convert the group column in data frame b to a factor type

class(b$group)

# a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

# Remove cells with zero abundance

ciber.res <- CIBERSORT\_Score\_GSE95233[,colSums(CIBERSORT\_Score\_GSE95233) > 0] # Remove cells with zero abundance

a <- a [,colnames(ciber.res)] %>% rownames\_to\_column("sample")

a$group <- b$group # Add the group column from table b to table a; note that both a and b must be data frames

library(ggsci)

library(tidyr)

library(ggpubr)

# b <- b[rownames(a),]

# identical(rownames(a),rownames(b))

# # Take the intersection of row names

# comgene <- intersect(rownames(a),rownames(b))

# a <- a[comgene,]

# class(a)# Check data type

# class(comgene)

# b <- b[comgene,]

# identical(rownames(a),rownames(b))# Check if multiple elements are identical

b <- gather(a,key=CIBERSORT,value = Proportion,-c(group,sample)) # Convert wide data to long data (code: 2119)

# pivot\_wider(names\_from = GSE, values\_from = logFC) # Convert long data to wide data (code: 1802)

pdf("./CIBERSORT\_GSE95233/boxplot\_gse95233.pdf")

ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10), # Font size of all text in the plot

axis.text.x = element\_text(angle=45, hjust=1))

# hjust =0 means left-aligned, hjust =1 means right-aligned, hjust =0.5 means centered

dev.off()

CB6 <- ggboxplot(b, x = "CIBERSORT", y = "Proportion",

fill = "group", palette = "lancet")+

stat\_compare\_means(aes(group = group),

method = "wilcox.test",

label = "p.signif",

symnum.args=list(cutpoints = c(0, 0.001, 0.01, 0.05, 1),

symbols = c("\*\*\*", "\*\*", "\*", "ns")))+

theme(text = element\_text(size=10),

axis.text.x = element\_text(angle=45, hjust=1))

CB6

dev.off()

#### 9 GO|KEGG ----

rm(list = ls())

setwd("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS")

# options("repos" = c(CRAN = "https://mirrors.tuna.tsinghua.edu.cn/CRAN/ "))

# options("BioC\_mirror" = "http://mirrors.tuna.tsinghua.edu.cn/bioconductor/ ")

# BiocManager::install("clusterProfiler")

# BiocManager::install("org.Hs.eg.db")

devtools::install\_local("tinyarray-master.zip",upgrade = FALSE,dependencies = TRUE)

library(tinyarray)

library(clusterProfiler)

library(org.Hs.eg.db)

library(tidyverse)

if(!dir.exists("GO\_KEGG"))dir.create("GO\_KEGG")

input <- read\_tsv("F:/HelixCourses/GEOdatasets\_spsis1/DEG\_SPSIS/RRP/all\_signi\_result\_P\_F.tsv")

head(input)

# Convert Gene Symbol to Entrez ID

gene.df <- bitr(input$SYMBOL,

fromType = "SYMBOL",

toType = c("ENSEMBL", "ENTREZID"),

# drop = TRUE, # Remove NA and unmatched gene names

OrgDb = org.Hs.eg.db)

gene.df\_1 <- bitr(input$SYMBOL,

fromType = "SYMBOL",

toType = "ENTREZID",

# drop = TRUE, # Remove NA and unmatched gene names

OrgDb = org.Hs.eg.db)

# rm(gene1)

gene1 <- gene.df$ENTREZID

gene <- gene1[!duplicated(gene1)]

class(gene)

### Start GO and KEGG analysis

# GO enrichment analysis

ego <- enrichGO(gene = gene,

OrgDb = org.Hs.eg.db,

ont = "ALL", # Default is "MF", ALL means all

pvalueCutoff =0.05,

qvalueCutoff =0.05,

readable = TRUE)

head(ego)

table(ego$ONTOLOGY)

ego\_result\_df <- as.data.frame(ego)

library(enrichplot)

??dotplot

enrichplot::dotplot(ego,x = "GeneRatio", color = "p.adjust", size = "Count", # Default parameters

showCategory =5,# Show only the top 5

split="ONTOLOGY") + # Split by ONTOLOGY type

facet\_grid(ONTOLOGY~., scale='free') # Split the plot by ONTOLOGY type

dev.off()

# Write GO enrichment results to a file

# write.table(ego,file="./RRP/GO\_result.txt",sep="\t", quote=FALSE,row.names = FALSE)

# KEGG enrichment analysis

kk <- enrichKEGG(gene = gene,

organism = "hsa",

# keyType = "kegg",

pvalueCutoff =0.9999,

qvalueCutoff =0.9999)

head(kk)

# Write KEGG enrichment results to a file

# write.table(kk,file="./RRP/KEGG\_result.txt",sep="\t",quote=FALSE,row.names = FALSE)

# Save results in Rda format

save(ego, kk, file = "./RRP/clusterProfiler\_result.Rda")

### Start visualization

# Load GO and KEGG results

load(file = "./RRP/clusterProfiler\_result.Rda")

## Bar plot

## Create directories (data and figures)

# if(!dir.exists("data")) dir.create("data")

pdf("./GO\_KEGG/GO\_barplot.pdf", 20, 8)

barplot(ego, showCategory = 10,color = "pvalue")

dev.off()

pdf("./GO\_KEGG/KEGG\_barplot.pdf", 20, 8)

barplot(kk, showCategory = 10,color = "pvalue")

dev.off()

# Dot plot

pdf("./GO\_KEGG/GO\_dotplot.pdf", 20, 8)

dotplot(ego, showCategory = 10,color = "pvalue")

dev.off()

# Faceted Dot Plot

pdf("./GO\_KEGG/GO\_facet\_dotplot.pdf", width = 20, height = 8)

dotplot(ego,

x = "GeneRatio", color = "p.adjust", size = "Count", # Default parameters (x and color can be changed based on ego content)

showCategory = 5, # Display top 5 categories

split = "ONTOLOGY") + # Split by ONTOLOGY type

facet\_grid(ONTOLOGY ~ ., scale = 'free') # Facet by ONTOLOGY type with free scales

dev.off()

pdf("./GO\_KEGG/KEGG\_dotplot.pdf", width = 20, height = 8)

dotplot(kk, showCategory = 10, color = "pvalue")

dev.off()

####

#### 10 WGCNA ----

### 10 WGCNA

#### 10.1 Data Processing

### 10.1.1 Loading Data (GSE69528) ###

rm(list = ls())

input <- load("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE69528\\GSE69528\_processed.Rda")

input

dim(GSE69528\_anno)

## 10.1.1 Selecting Genes by Standard Deviation ###

# # Calculate standard deviation for each gene

# GSE69528\_anno\_sd = apply(GSE69528\_anno, 1, sd) # 1 for rows, 2 for columns

#

# ## Sort genes by standard deviation in descending order

# GSE69528\_anno\_sd\_sorted = order(GSE69528\_anno\_sd, decreasing = TRUE)

# # F:/Decoding Helix Courses/Full Code Breakthrough Bioinformatics Data Mining/Level 2/L62\_Codes/6-2-GSEA\_GSVA\_Analysis/Step01-GSEA\_Analysis.R Code: P34

# # geneList = sort(geneList, decreasing = TRUE)

#

# ## Select top 5000 genes with the highest standard deviation

# GSE69528\_anno\_num = GSE69528\_anno\_sd\_sorted[1:5000]

# ## Extract genes from the expression matrix

# GSE69528\_anno\_filter = GSE69528\_anno[GSE69528\_anno\_num,]

### 10.1.1 Using All Data ###

## Transpose the expression matrix

WGCNA\_matrix = t(GSE69528\_anno) # Rows become samples, columns become genes

dim(WGCNA\_matrix)

## Save the filtered data

if (!dir.exists("WGCNA")) dir.create("WGCNA")

save(WGCNA\_matrix, file = "WGCNA/WGCNA\_matrix.Rda")

### 10.1.2 Removing Samples with Many Missing Values ###

rm(list = ls())

# Load the expression matrix

load(file = "WGCNA/WGCNA\_matrix.Rda")

# Load WGCNA package

library(WGCNA)

# Check if there are too many missing samples or genes

datExpr0 = WGCNA\_matrix

dim(datExpr0)

gsg = goodSamplesGenes(datExpr0, verbose = 3)

# Determine if filtering is needed, TRUE means no filtering, FALSE means filtering

gsg$allOK

# When gsg$allOK is TRUE, the following code does not run; if FALSE, filter samples

if (!gsg$allOK)

{

# Print removed gene and sample names

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 = ", ")));

# Extract retained genes and samples

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

}

### 10.1.3 Identifying and Removing Outlier Samples via Sample Clustering ###

sampleTree = hclust(dist(datExpr0), method = "average") # Perform hierarchical clustering using average linkage

# Plot the sample clustering dendrogram to identify outliers

sizeGrWindow(30, 9)

pdf(file = "WGCNA/figures/sampleClustering.pdf", width = 30, height = 9)

par(cex = 0.6)

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)

# Draw a red line at height h to indicate where to cut

abline(h = 140, col = "red") # Cut at height 140 based on the dendrogram

dev.off()

# Remove outlier samples based on the cut height

clust = cutreeStatic(sampleTree, cutHeight = 140, minSize = 10)

table(clust)

# clust

# 0 1

# 0 is to be removed, 1 is to be kept

# Extract samples belonging to cluster 1

keepSamples = (clust == 1)

datExpr = datExpr0[keepSamples, ]

dim(datExpr)

# Record the number of genes and samples for later visualization

nGenes = ncol(datExpr) # Number of genes

nSamples = nrow(datExpr) # Number of samples

save(datExpr, nGenes, nSamples, file = "WGCNA/WGCNA\_input.Rda")

#### 10.2 Step-by-Step WGCNA

# Clear the environment

rm(list = ls())

# Load necessary packages

library(WGCNA)

# Enable multi-threading

enableWGCNAThreads()

# Load the expression matrix

load("WGCNA/WGCNA\_input.Rda")

### 10.2.1 Selecting Soft Threshold ###

sizeGrWindow(9, 5)

par(mfrow = c(1, 2))

powers = c(c(1:10), seq(from = 12, to = 20, by = 2))

# powers = c(1:20) # Power exponent range from 1 to 20

# RpowerTable\_1 = pickSoftThreshold(datExpr, powerVector = powers) is equivalent to the following

RpowerTable = pickSoftThreshold(datExpr, powerVector = powers)[[2]]

sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)

# If R^2 does not reach 0.85 or above, choose an empirical beta value.

# For nSamples < 20, soft threshold at 9; for 20 <= nSamples < 30, choose 8; for 30 <= nSamples < 40, choose 7

# Visualize results to determine the soft threshold

cex1 = 0.7

pdf(file = "WGCNA/figures/softThresholding.pdf", width = 14)

par(mfrow = c(1, 2))

plot(RpowerTable[, 1], -sign(RpowerTable[, 3]) \* RpowerTable[, 2], xlab = "Soft Threshold (power)", ylab = "Scale Free Topology Model Fit, signed R^2", type = "n", main = paste("Scale independence"))

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

abline(h = 0.85, col = "red")

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

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

dev.off()

## Verify Scale-Free Network: Ensure the soft threshold is reasonable

# softpower = 10

softpower = sft$powerEstimate

ADJ = abs(cor(datExpr, use = "p"))^softpower # cor(): Calculate pairwise correlations

head(ADJ)[, 1:6]

# "p" indicates pairwise deletion; cor can only calculate correlation coefficients, need cor.mtset to calculate p-values

# na.rm = TRUE parameter; remove missing values during calculation for valid results

k = as.vector(apply(ADJ, 2, sum, na.rm = TRUE))

pdf(file = "WGCNA/figures/scaleFree.pdf", width = 14)

par(mfrow = c(1, 2))

hist(k)

scaleFreePlot(k, main = "Check scale free topology")

dev.off()

### 10.2.2 Calculate Adjacency, Topological, and Dissimilarity Matrices ###

## Calculate adjacency matrix

adjacency = adjacency(datExpr, power = softpower)

adjacency[1:6, 1:6]

## Calculate TOM (Topological Overlap Matrix)

TOM = TOMsimilarity(adjacency)

TOM[1:6, 1:6]

## Calculate dissimilarity

dissTOM = 1 - TOM

dissTOM[1:6, 1:6]

# The above code is found on P4642~P4643

### 10.2.3 Construct Gene Hierarchical Clustering Tree and Network Modules ###

## 10.2.3.1 Construct Gene Hierarchical Clustering Tree

# Preliminary module clustering analysis: (Cluster genes based on dissimilarity)

library(flashClust)

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

# The above two lines are equivalent to geneTree = hclust(as.dist(dissTOM), method = "average")

# Plot the hierarchical clustering tree

pdf(file = "WGCNA/figures/GeneClusterTOM.pdf")

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

dev.off()

## 10.2.3.2 Construct Network Modules

# Build preliminary gene modules

# Set minimum module size to 50 genes

minModuleSize = 50

# Dynamically cut the tree to identify network modules

dynamicMods = cutreeDynamic(dendro = geneTree, # hclust result from clustering

distM = dissTOM, # Dissimilarity matrix

deepSplit = 0,

pamRespectsDendro = FALSE,

minClusterSize = minModuleSize) # Set minimum genes per module to 30

table(dynamicMods)

## 10.2.3.3 Plot Hierarchical Clustering Tree and Network Modules

# Convert labels to colors

# standardColors(20)

dynamicColors = labels2colors(dynamicMods) # Map numerical labels to module colors

table(dynamicColors) # View which modules and colors genes are clustered into

pdf(file = "WGCNA/figures/DynamicTreeCut.pdf", width = 9, height = 5)

plotDendroAndColors(dendro = geneTree,

colors = dynamicColors,

groupLabels = "Dynamic Tree Cut",

dendroLabels = FALSE, hang = 0.03,

addGuide = TRUE,

main = "Gene dendrogram and module colors")

dev.off()

# Rename the merged modules

moduleColors = mergedColors

colorOrder = c("grey", standardColors(50))

moduleLabels = match(moduleColors, colorOrder) - 1

MEs = mergedMEs

MEDiss2 = 1 - cor(MEs) # Calculate and convert module dissimilarity

METree2 = flashClust(as.dist(MEDiss2), method = "average") # Cluster the merged modules

# Plot clustering results

pdf(file = "WGCNA/figures/MECombined.pdf", width = 12, height = 5)

par(mfrow = c(1, 2))

plot(METree1, xlab = "", sub = "", main = "Clustering of ME before combined") # METree1 is from dynamic tree cut modules

abline(h = MEDissThres, col = "red") # Dissimilarity threshold at 0.25

plot(METree2, xlab = "", sub = "", main = "Clustering of ME after combined") # METree2 is after merging

dev.off()

pdf(file = "WGCNA/figures/MergedDynamics.pdf", width = 10, height = 4)

plotDendroAndColors(dendro = geneTree, # Cut tree

colors = cbind(dynamicColors, mergedColors), # Combine both methods' module colors

groupLabels = c("Dynamic Tree Cut", "Merged Dynamics"),

dendroLabels = FALSE,

hang = 0.03,

addGuide = TRUE,

guideHang = 0.05,

main = "Gene Dendrogram and module colors")

dev.off()

# Number of genes in each module

write.table(table(moduleColors), "WGCNA/MEgeneCount.txt", quote = FALSE, row.names = FALSE)

# Save network information

moduleColors = mergedColors

# standardColors(100)

colorOrder = c("grey", standardColors(50))

moduleLabels = match(moduleColors, colorOrder) - 1 # match: Match two vectors, return the position of the second vector in the first

MEs = mergedMEs

save(MEs, moduleLabels, softpower, moduleColors, geneTree, file = "WGCNA/Step\_by\_step\_buildnetwork.rda")

## Plot Module Correlations

rm(list = ls())

library(WGCNA)

load("WGCNA/WGCNA\_input.Rda")

load(file = "WGCNA/Step\_by\_step\_buildnetwork.rda")

MEs = orderMEs(MEs)

sizeGrWindow(5, 7.5)

pdf(file = "WGCNA/figures/moduleCor.pdf", width = 5, height = 7.5)

par(cex = 0.9)

plotEigengeneNetworks(MEs, "", marDendro = c(0, 4, 1, 2), marHeatmap = c(3, 4, 1, 2), cex.lab = 0.8, xLabelsAngle = 90)

dev.off()

# Enable multi-threading for WGCNA

enableWGCNAThreads()

## TOMplot

dissTOM = 1 - TOMsimilarityFromExpr(datExpr, power = softpower) # Consistent with earlier dissimilarity calculation

head(dissTOM)[, 1:6]

# The above code is found on P4514~P4519

nSelect = 400

# Randomly select 400 genes for visualization, set seed for reproducibility

set.seed(10)

select = sample(nGenes, size = nSelect)

selectTOM = dissTOM[select, select] # Extract TOM matrix for selected genes

# Re-cluster the selected genes

selectTree = hclust(as.dist(selectTOM), method = "average")

mergedColors = moduleColors

selectColors = mergedColors[select] # Extract module colors for the 400 selected genes

# Open a plotting window

sizeGrWindow(9, 9)

pdf(file = "WGCNA/figures/TOMplot.pdf", width = 9, height = 9)

# Enhance the TOM plot by exponentiating

plotDiss = selectTOM^7

diag(plotDiss) = NA # Set diagonal to NA

TOMplot(plotDiss, selectTree, selectColors,

main = "Network heatmap plot, selected genes")

dev.off()

## Adjust color scheme for better visualization

??colorpanel

library(gplots)

myheatcol = colorpanel(250, 'red', "orange", 'lemonchiffon')

pdf(file = "TOM\_plot0.pdf")

plotDiss = selectTOM^7

diag(plotDiss) = NA

TOMplot(plotDiss, selectTree, col = myheatcol, selectColors, main = "Network heatmap plot, selected genes")

dev.off()

### 10.3 Clinical Phenotype Module Construction ###

# Clear environment variables

rm(list = ls())

# Load necessary packages

library(WGCNA)

# Load expression matrix

load("WGCNA/WGCNA\_input.Rda")

load("GSE69528/GSE69528\_processed.Rda")

## 10.3.1 Load Clinical Information

# clinical = read.table("data/clinical.txt", stringsAsFactors = TRUE, header = TRUE, row.names = 1, na.strings = "", sep = "\t")

# clinical <- GSE69528\_targets[c(1,4)]

library(tidyverse)

clinical <- GSE69528\_targets %>% rownames\_to\_column()

clinical1 <- clinical[c(2, 5)] %>% mutate("sepsis" = group) %>%

column\_to\_rownames("sample\_id") %>%

set\_names("Control", "Sepsis")

datExpr <- as.data.frame(datExpr)

clinical2 <- clinical1[rownames(datExpr), ] %>% as.data.frame()

# write.table(clinical2, file = "WGCNA/clinical2.txt", row.names = TRUE, col.names = TRUE, sep = "\t", quote = FALSE)

clinical = read.table("WGCNA/clinical2.txt", stringsAsFactors = FALSE, header = TRUE, row.names = 1, na.strings = "", sep = "\t")

# View clinical information

head(clinical)

str(clinical)

# Preprocess the expression matrix

# datTraits = as.data.frame(do.call(cbind, lapply(clinical, as.factor)))

# rownames(datTraits) = rownames(clinical)

datTraits <- clinical

dim(datTraits)

head(datTraits)

## 10.3.2 Cluster Samples

head(datExpr)[, 1:5]

sampleTree2 = hclust(dist(datExpr), method = "average")

plot(sampleTree2)

## 10.3.3 Convert Clinical Information to Colors

# Convert clinical information to colors,

## 10.3.3 Convert sample clinical information to colors

# Convert clinical information to colors, white represents low, red represents high, gray represents missing

traitColors = numbers2colors(datTraits, signed = FALSE)

# Step03-Step\_by\_Step\_for\_WGCNA.R code: 78

# dynamicColors = labels2colors(dynamicMods) # Map numeric labels to modules

table(traitColors)

pdf(file = "WGCNA/figures/Sample\_dendrogram\_and\_trait\_heatmap.pdf", width = 24);

## 10.3.4 Sample clustering plot and sample trait heatmap

plotDendroAndColors(sampleTree2,

traitColors,

groupLabels = names(datTraits),

main = "Sample dendrogram and trait heatmap")

dev.off()

#### 10.3.5 Network analysis

###### Relationship between gene modules and clinical information

# Load the constructed network

load(file = "WGCNA/Step\_by\_step\_buildnetwork.rda")

# 10.3.5.1 Sort the module eigengenes matrix

MEs=orderMEs(MEs)

# 10.3.5.2 Calculate the correlation between module eigengenes matrix and sample information matrix.

moduleTraitCor=cor(MEs, datTraits, use="p")

# write.table(file="data/Step04-modPhysiological.cor.xls",moduleTraitCor,sep="\t",quote=F)

moduleTraitPvalue=corPvalueStudent(moduleTraitCor, nSamples)

# write.table(file="data/Step04-modPhysiological.p.xls",moduleTraitPvalue,sep="\t",quote=F)

# 10.3.5.3 Visualize the correlation matrix and p-values using labeledHeatmap().

pdf(file="WGCNA/figures/Module\_trait\_relationships.pdf",width=9,height=7)

textMatrix=paste(signif(moduleTraitCor,2),"\n(",signif(moduleTraitPvalue,1),")",sep="")

head(textMatrix)

dim(textMatrix)=dim(moduleTraitCor)

# Heatmap of module-trait relationships

labeledHeatmap(Matrix=moduleTraitCor,# Core parameter: correlation matrix between modules and phenotypes

xLabels=colnames(datTraits),

yLabels=names(MEs),

ySymbols=names(MEs),

colorLabels=FALSE,

colors=blueWhiteRed(50),

textMatrix=textMatrix,

setStdMargins=FALSE,

cex.text=0.5,

cex.lab=0.5,

zlim=c(-1,1),

main=paste("Module-trait relationships"))

dev.off()

### 10.4 Batch output of trait-module scatter plots and module genes ###

# Detailed analysis of different modules and gene traits

## Matrix 1: Construct correlation coefficients and p-values between each gene and module

modNames = substring(names(MEs), 3) # Extract column names of MEs starting from the third character

geneModuleMembership = as.data.frame(cor(datExpr, MEs, use = "p")) # Calculate correlation coefficients

#### Check which genes are most correlated with a specific trait

a <- geneModuleMembership

head(a)

a <- a %>% rownames\_to\_column()

MMPvalue = as.data.frame(corPvalueStudent(as.matrix(geneModuleMembership), nSamples)) # Calculate p-values of correlations

names(geneModuleMembership) = paste("MM", modNames, sep="") # Rename columns of geneModuleMembership

names(MMPvalue) = paste("p.MM", modNames, sep="") # Rename columns of MMPvalue

## Matrix 2: Construct correlation coefficients and p-values between each gene and clinical traits

traitNames=names(datTraits) # Get column names

geneTraitSignificance = as.data.frame(cor(datExpr, datTraits, use = "p")) # use = "p": pairwise deletion in cor() function

GSPvalue = as.data.frame(corPvalueStudent(as.matrix(geneTraitSignificance), nSamples))

names(geneTraitSignificance) = paste("GS.", traitNames, sep="")

names(GSPvalue) = paste("p.GS.", traitNames, sep="")

## Extract key module core genes: Method 1

# Extract genes with GS.Sepsis > 0.2 and MM.blue > 0.8

FilterGenes= abs(geneTraitSignificance$GS.Sepsis)> .2 & abs(geneModuleMembership$MMblue)>.8

Genes <- colnames(datExpr)

FilterGenes\_name <- Genes[FilterGenes] # Extract gene names where logical condition is TRUE

write.table(FilterGenes\_name,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Module and gene\\FilterGenes.txt",

sep="\t",row.names=TRUE,col.names=TRUE,quote=FALSE)

## Extract key module core genes: Method 2

# Select MMblue module: |MM|>0.8 & |GS|>0.2

hub<- abs(geneModuleMembership$MMblue)>0.8 & abs(geneTraitSignificance$GS.Sepsis)>0.2

table(hub)

# FALSE TRUE

# 10892 616

identical(FilterGenes,hub)

hub\_name <- dimnames(data.frame(datExpr))[[2]][hub]

identical(FilterGenes\_name,hub\_name)

## Upon inspection, FilterGenes\_name and hub\_name differ because:

## hub\_name contains "." in some gene names, while FilterGenes\_name contains "-" in some gene names

## Therefore, use: gsub('\\.', '-', x); gsub('[.]', '-', x); gsub(".", "-", x, fixed = TRUE)

hub\_name <- gsub('\\.', '-', hub\_name)

identical(FilterGenes\_name,hub\_name)

# hub<-as.data.frame(dimnames(data.frame(datExpr))[[2]][hub])

# write.csv(hub, "./WGCNA/Module and gene/hubgene\_GSMM\_blue\_sepsis.csv")

write.table(hub\_name,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Module and gene\\hubgene\_GSMM\_blue\_sepsis.txt",

sep="\t",row.names=TRUE,col.names=TRUE,quote=FALSE)

# Different |MM| thresholds: |MM|>0.85 & |GS|>0.2

hub1<- abs(geneModuleMembership$MMblue)>0.85 & abs(geneTraitSignificance$GS.Sepsis)>0.2

table(hub1)

hub1<-as.data.frame(dimnames(data.frame(datExpr))[[2]][hub1])

# write.csv(hub1, "./WGCNA/Module and gene/hubgene1\_GSMM\_blue\_sepsis.csv")

write.table(hub1,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Module and gene\\hubgene1\_GSMM\_blue\_sepsis.txt",

sep="\t",row.names=FALSE,col.names=TRUE,quote=FALSE)

# It was found that 41 genes do not belong to this module: See "WGCNA（6）：筛选 hub gene" in browser bookmarks

# Handling method as follows:

module = "blue"

column = match(module, modNames)

moduleGenes = moduleColors==module

blue\_module<-as.data.frame(dimnames(data.frame(datExpr))[[2]][moduleGenes])

names(blue\_module)="genename"

MM<-abs(geneModuleMembership[moduleGenes,column])

GS<-abs(geneTraitSignificance[moduleGenes, 2])

blue\_MMGS<-as.data.frame(cbind(MM,GS))

rownames(blue\_MMGS)=blue\_module$genename

hub\_b<-abs(blue\_MMGS$MM)>0.8&abs(blue\_MMGS$GS)>0.2

table(hub\_b)

# FALSE TRUE

# 1755 424

blue\_hub\_b <- subset(blue\_MMGS, abs(blue\_MMGS$MM)>0.8 & abs(blue\_MMGS$GS)>0.2)

write.csv(blue\_hub\_b,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Module and gene\\hubgene\_MMGS\_blue.csv")

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS/WGCNA/figures/Module and clinical character map")

### 10.4.1 Batch output of trait-module scatter plots ###

for (trait in traitNames){

traitColumn=match(trait,traitNames)

for (module in modNames){

column = match(module, modNames)

moduleGenes = moduleColors==module

if (nrow(geneModuleMembership[moduleGenes,]) > 1){

outPdf=paste(trait, "\_", module,".pdf",sep="")

pdf(file=outPdf,width=7,height=5) # Adjust figure dimensions

par(mfrow = c(1,1))

verboseScatterplot(abs(geneModuleMembership[moduleGenes, column]),

abs(geneTraitSignificance[moduleGenes, traitColumn]),

xlab = paste("Module Membership in", module, "module"),

ylab = paste("Gene significance for ",trait),

main = paste("Module membership vs. gene significance\n"),

cex.main = 1.2, cex.lab = 1.2, cex.axis = 1.2, col = module)

abline(v=0.8,h=0.2,col="red") # Highlight the region for extracting core genes

dev.off()

}

}

}

### 10.4.2 Batch output of genes in each module ###

for (mod in 1:nrow(table(moduleColors)))

{

modules = names(table(moduleColors))[mod]

probes = colnames(datExpr)

inModule = (moduleColors == modules)

modGenes = probes[inModule]

write.table(modGenes, file =paste0(modules,".txt"),sep="\t",row.names=FALSE,col.names=FALSE,quote=FALSE)

}

### 10.5 GO/KEGG analysis of genes in the blue module ###

rm(list=ls())

library(tinyarray)

library(clusterProfiler)

library(org.Hs.eg.db)

library(tidyverse)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS/WGCNA")

if(!dir.exists("GO\_KEGG"))dir.create("GO\_KEGG")

input <- read.table(".\\figures\\Module and clinical character map\\blue.txt",

stringsAsFactors = FALSE, header = FALSE,

# row.names = 1,

na.strings = "",

sep = "\t")

head(input)

# Convert Gene Symbols to Entrez ID

gene.df <- bitr(input$V1,

fromType = "SYMBOL",

toType = c("ENSEMBL", "ENTREZID"),

# drop = TRUE, # Remove NA and genes that cannot be matched

OrgDb = org.Hs.eg.db)

# gene.df\_1 <- bitr(input$SYMBOL,

# fromType = "SYMBOL",

# toType = "ENTREZID",

# # drop = TRUE, # Remove NA and genes that cannot be matched

# OrgDb = org.Hs.eg.db)

# rm(gene1)

gene1 <- gene.df$ENTREZID

gene <- gene1[!duplicated(gene1)]

class(gene)

head(gene)

### Start GO and KEGG analysis

# GO enrichment analysis

ego <- enrichGO(gene = gene,

OrgDb = org.Hs.eg.db,

ont = "ALL", # Default is "MF", ALL includes all

pvalueCutoff =0.05,

qvalueCutoff =0.05,

readable = TRUE)

head(ego)

table(ego$ONTOLOGY)

ego\_result\_df <- as.data.frame(ego) # Convert to data frame

dotplot(ego,x = "GeneRatio", color = "p.adjust", size = "Count", # Default parameters

showCategory =5,# Show top 5

split="ONTOLOGY") + # Split by ONTOLOGY type

facet\_grid(ONTOLOGY~., scale='free') # Facet by ONTOLOGY type

dev.off()

# Save GO enrichment results to file

# write.table(ego,file="./RRP/GO\_result.txt",sep="\t", quote=F,row.names = F)

# KEGG enrichment analysis

kk <- enrichKEGG(gene = gene,

organism = "hsa",

# keyType = "kegg",

pvalueCutoff =0.9999,

qvalueCutoff =0.9999)

head(kk)

# Save results as Rda format

save(ego, kk, file = "./GO\_KEGG/clusterProfiler\_result.Rda")

# Paginated dot plot

pdf("./GO\_KEGG/GO\_facet\_dotplot.pdf", 20, 8)

dotplot(ego,

x = "GeneRatio", color = "p.adjust", size = "Count", # Default parameters (x and color can be changed based on ego content)

showCategory =5, # Show top 5

split="ONTOLOGY") + # Split by ONTOLOGY type

facet\_grid(ONTOLOGY~., scale='free') # Facet by ONTOLOGY type

dev.off()

pdf("./GO\_KEGG/KEGG\_dotplot.pdf", 20, 8)

dotplot(kk, showCategory = 10,color = "pvalue")

dev.off()

### Intersection of results ###

rm(list=ls())

input <- read.table('F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\RRP\\all\_signi\_result\_P\_F.tsv')

FilterGenes <- read.table("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Module and gene\\FilterGenes.txt")

hub <- read.csv("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Module and gene\\hubgene\_MMGS\_blue.csv")

identical(hub$x,FilterGenes$x)

Comgene <- as.data.frame(intersect(hub$X,input$V1))

input\_up <- read.table(".\\Venn\\all\_signi\_result\_P\_F\_up.txt")

input\_down <- read.table(".\\Venn\\all\_signi\_result\_P\_F\_down.txt")

Comgene\_up <- as.data.frame(intersect(hub$X,input\_up$V1))

Comgene\_down <- as.data.frame(intersect(hub$X,input\_down$V1))

write.table(Comgene,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Venn\\Comgene.txt",

sep="\t",

row.names=FALSE,

col.names=FALSE,

quote=FALSE)

write.table(Comgene\_up,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Venn\\Comgene\_up.txt",

sep="\t",

row.names=FALSE,

col.names=FALSE,

quote=FALSE)

write.table(Comgene\_down,

file ="F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\WGCNA\\Venn\\Comgene\_down.txt",

sep="\t",

row.names=FALSE,

col.names=FALSE,

quote=FALSE)

#### 11 STRING analysis ----

# Perform STRING analysis using an online tool

rm(list = ls())

library(tidyverse)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS/STRING/ELF1\_miRNA")

miRNA <- read.table(".\\venn\_ELF1\_miRNA\_3com.txt",header = TRUE)

lncRNA\_miRNA <- read.table("lncRNA\_miRNA\_interaction\_tiqu.txt",

header = TRUE,sep="\t",check.names=FALSE)

com\_miRNA <- intersect(miRNA$GENE,lncRNA\_miRNA$miRNAname)

com\_lncRNA\_miRNA <- lncRNA\_miRNA %>% filter(lncRNA\_miRNA$miRNAname %in% com\_miRNA) %>% arrange(miRNAname,geneName)

com\_lncRNA\_miRNA\_1 <- lncRNA\_miRNA[lncRNA\_miRNA$miRNAname %in% com\_miRNA, ] %>% arrange(miRNAname,geneName)

identical(com\_lncRNA\_miRNA,com\_lncRNA\_miRNA\_1)

a <- unique(com\_lncRNA\_miRNA$miRNAname)

# Extract duplicates

com\_lncRNA\_miRNA\_dup <- com\_lncRNA\_miRNA[!duplicated(com\_lncRNA\_miRNA),]

# duplicated() identifies the first occurrence of duplicates in a vector or data frame and returns a logical vector indicating which elements (rows) are duplicates.

com\_lncRNA\_miRNA\_dup\_LINC <- com\_lncRNA\_miRNA\_dup[duplicated(com\_lncRNA\_miRNA\_dup$geneName)==TRUE,]

b <- table(com\_lncRNA\_miRNA\_dup\_LINC$geneName) %>% as.data.frame() %>% arrange(.,desc(Freq))

NEAT1\_miRNA <- com\_lncRNA\_miRNA\_dup\_LINC[com\_lncRNA\_miRNA\_dup\_LINC$geneName=="NEAT1",]

XIST\_miRNA <- com\_lncRNA\_miRNA\_dup\_LINC[com\_lncRNA\_miRNA\_dup\_LINC$geneName=="XIST",]

c <- intersect(lncRNA\_miRNA\_NEAT1$miRNAname,lncRNA\_miRNA\_XIST$miRNAname)

# Save data

write.table(b, file = "./lncRNA\_miRNA\_mRNA.txt",

sep = "\t",row.names = FALSE,col.names = TRUE,quote = FALSE)

write.table(lncRNA\_miRNA\_NEAT1, file = "./NEAT1\_miRNA.txt",

sep = "\t",row.names = FALSE,col.names = TRUE,quote = FALSE)

#### 12 Validation analysis ----

rm(list = ls())

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS")

library(GEOquery)

library(readxl)

library(tidyverse)

# ## Read GSE13015 data

# GSE13015\_sm<- getGEO(filename = "./Validated dataset/GSE13015/GSE13015-GPL6

# destdir = ".",

# getGPL = F )

# GSE13015\_expr <- exprs(GSE13015\_sm)

# head(GSE13015\_expr)[, 1:6]

# GSE13015\_pd <- pData(GSE13015\_sm)

#

# ### create file

# setwd("F:/Decoding Helix Course/GEOdatasets\_spsis1/DEG\_SPSIS/Validated dataset/GSE13015")

# # dir.create("GSE13015")

# if(!dir.exists("GSE13015"))dir.create("GSE13015")

# getwd()

#

# ### save data

# save(GSE13015\_sm, GSE13015\_expr, GSE13015\_pd,

# file = ".\\GSE13015.Rda")

# # load(".\\GSE13015\_data.Rda")

#

# ### organize sample information

# GSE13015\_targets <- GSE13015\_pd %>% dplyr::select(2,1,14) %>%

# # dplyr::select(1, 2) %>%

# # separate(title, into = paste0("x", 1:4), sep = "\_", remove = F) %>%

# # filter(!(x4 %in% c("D02", "D03"))) %>% # "F:/Decoding Helix Course/R language basics\_7 days/Lesson 3/Lesson 3 live broadcast" code: P264

# mutate(sample\_id = geo\_accession,#add several columns

# #file\_name = paste0(sample\_id, ".CEL.gz"),

# sample\_name = title,

# # sample\_name = str\_c(x1, x2, x3, sep = "\_"),# Join multiple strings into a single string.

# # patient\_id = str\_extract(x2,"\\d+"),

# # patient\_id = x3 ,

# patient\_id = str\_sub(GSE13015\_pd$characteristics\_ch1.4, start = 10, end = -1),

# #str\_sub----Extract and replace substrings from a character vector. Extract from the last but one to the last three characters

# #GSE\_num = "GSE69063",

# # The following line of code is shown in: "F:/Dr. Chen's medical records/Dr. Chen wants to be lazy/GSE84402" code: P240~P241

# # group\_list <- ifelse(str\_detect(pdata$source\_name\_ch1, "hepatocellular carcinoma"), "tumor", "normal")

# # group = ifelse(str\_sub(source\_name\_ch1, 1, 1) == "P","Sepsis","Control")) %>%

# group = ifelse(str\_detect(GSE13015\_pd$characteristics\_ch1.4, "Control"), "Control", "Sepsis")) %>%

# arrange(group,sample\_id) %>%

# dplyr::select(sample\_id:group) #%>%

# # dplyr::select(2,3,4,1)

#

# GSE13015\_expr <- GSE13015\_expr[ , GSE13015\_targets$sample\_id] %>% as.data.frame()

#

# ### View Data

# range(GSE13015\_expr)

# exp <- log2(GSE13015\_expr+1)

# range(expr)

# ### Error, indicating this method is not feasible

#

# ### Check how the original data was processed

# GSE13015\_sm@phenoData@data$data\_processing[1]

#

# #### Start from raw data extraction

# # Original data still contains negative values, so this dataset is abandoned

# Read GSE28750 data

GSE28750\_sm<- getGEO(filename = "./Validated dataset/GSE28750/GSE28750\_series\_matrix.txt.gz",

destdir = ".",

getGPL = F )

GSE28750\_expr <- exprs(GSE28750\_sm)

head(GSE28750\_expr)[, 1:6]

GSE28750\_pd <- pData(GSE28750\_sm)

### create file

setwd("F:/Decoding Helix Course/GEOdatasets\_spsis1/DEG\_SPSIS/Validated dataset/GSE28750")

# dir.create("GSE28750")

# if(!dir.exists("GSE28750"))dir.create("GSE28750")

getwd()

### save data

save(GSE28750\_sm, GSE28750\_expr, GSE28750\_pd,

file = ".\\GSE28750.Rda")

# load(".\\GSE28750.Rda")

### organize sample information

GSE28750\_targets <- GSE28750\_pd %>% #dplyr::select(2,1,14) %>%

dplyr::select(2, 1) %>%

separate(title, into = paste0("x", 1:4), sep = "\_", remove = F) %>%

# filter(!(x2 == "Post")) %>%

mutate(sample\_id = geo\_accession,#add several columns

#file\_name = paste0(sample\_id, ".CEL.gz"),

sample\_name = title,

# sample\_name = str\_c(x1, x2, x3, sep = "\_"),# Join multiple strings into a single string.

# patient\_id = str\_extract(x2,"\\d+"),

# patient\_id = x3 ,

# patient\_id = str\_sub(GSE28750\_pd$characteristics\_ch1.4, start = 10, end = -1),

patient\_id = str\_c(x2, x3, sep = "\_"),

#str\_sub----Extract and replace substrings from a character vector. Extract from the last but one to the last three characters

#GSE\_num = "GSE69063",

# The following line of code is shown in: "F:/Dr. Chen's medical records/Dr. Chen wants to be lazy/GSE84402" code: P240~P241

# group\_list <- ifelse(str\_detect(pdata$source\_name\_ch1, "hepatocellular carcinoma"), "tumor", "normal")

# group = ifelse(str\_sub(source\_name\_ch1, 1, 1) == "P","Sepsis","Control")) %>%

# group = ifelse(str\_detect(GSE28750\_pd$characteristics\_ch1.4, "Control"), "Control", "Sepsis")) %>%

group = ifelse(x1 == "Control", "Control", "Sepsis")) %>%

arrange(group,sample\_id) %>%

dplyr::select(sample\_id:group) #%>%

# dplyr::select(2,3,4,1)

GSE28750\_expr <- GSE28750\_expr[ , GSE28750\_targets$sample\_id] %>% as.data.frame()

### View Data

range(GSE28750\_expr)

# exp <- log2(GSE28750\_expr+1)

# range(expr)

### Evaluate data

if(!dir.exists("figures")) dir.create("figures")

pdf('./figures/GSE28750\_boxplot.pdf',width = 8,height = 8)

# boxplot(GSE29450\_rma, las = 2, col = rep(c("blue", "red"), each = 10))

# boxplot(GSE95233\_expr, las=2, outline=FALSE,

# col = rep(c("blue", "red"), c(22,51)),

# main = "GSE95233\_expr")

boxplot(GSE28750\_expr, las=2, outline=FALSE,

# col = rep(c("blue", "red"), c(22,51)),

main = "GSE28750\_expr")

dev.off()

source("F:\\Decoding Helix Course\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Custom\_Functions.R")

# BiocManager::install("genefilter", ask = F, update = F)

if (!requireNamespace("genefilter", quietly = TRUE))

BiocManager::install("genefilter", ask = F, update = F)

targets <- GSE28750\_targets

pdf('./figures/GSE28750\_PCA\_new.pdf',width = 8,height = 8)

# PCA\_new(GSE28750\_expr, show\_name = F) #default take top 500 genes with largest differences: i.e. ntop = 500

PCA\_new(GSE28750\_expr,

ntop = nrow(GSE28750\_expr),

show\_name = F)

# From the above two PCA plots, we can see that different numbers of genes have a big impact on the results

dev.off()

## Check how the original data was processed

GSE28750\_sm@phenoData@data$data\_processing[1]

# GSE28750\_pd$data\_processing[1] #this code is equivalent to the above code

## normalization

# # RMA normalized: original data standardization

# #Series matrix files txt or soft files data standardization

library(limma)

GSE28750\_expr\_n = normalizeBetweenArrays(GSE28750\_expr)

boxplot(GSE28750\_expr\_n, las=2, outline=FALSE, main = "GSE28750\_expr\_n")

dev.off()

PCA\_new(GSE28750\_expr\_n,

ntop = nrow(GSE28750\_expr\_n),

# ntop = 500, #number of top-ranked genes

# show\_name = T,

group = GSE28750\_targets$group)

dev.off()

# # From the above data processing method and boxplot/PCA plots, we know that GSE69528\_expr doesn't need further standardization

# GSE69528\_expr\_n <- GSE69528\_expr #this step is to keep all datasets consistent

### annotation

GSE28750\_sm@annotation

## GPL570

# chooseCRANmirror()

# chooseBioCmirror()

# pkgs <- c("hgu133plus2.db","illuminaHumanv3.db")

# pkgs\_in(pkgs)

library(hgu133plus2.db)

library(illuminaHumanv3.db)

ls("package:hgu133plus2.db")

p2s <- toTable(hgu133plus2SYMBOL) #%>%

# set\_names("probe\_id", "symbol")#rename columns

head(p2s)

## Custom function for chip annotation

annotate\_expr <- function(expr, p2s, fun = mean){#three parameters: p2s has fixed column names

library(dplyr)

library(tibble)

# expr

expr <- expr %>%

as.data.frame() %>%

tibble::rownames\_to\_column("probe\_id")#row names become a column called probe\_id

# annotate

expr\_annotated <- p2s %>%

dplyr::inner\_join(expr, by = "probe\_id") %>% #intersection of expr and p2s

na.omit() %>% #remove NA

dplyr::select(-probe\_id) %>% #delete probe\_id column

stats::aggregate(. ~ symbol, data =., FUN = fun) %>% #for multiple probes corresponding to the same symbol, take mean of these probes

tibble::column\_to\_rownames("symbol")#symbol column becomes row names

return(expr\_annotated)#return this expr\_annotated, then assign to annotate\_expr

}

## Chip annotation

GSE28750\_anno <- annotate\_expr(GSE28750\_expr\_n, p2s)

head(GSE28750\_anno)[, 1:3]

## save data

save(GSE28750\_expr\_n, GSE28750\_targets, GSE28750\_anno,

file = ".\\GSE28750\_processed.Rda")

### Differential analysis DEGs

library(limma)

## for design

design <- model.matrix(~ group, data = GSE28750\_targets)

colnames(design)

colnames(design) <- c("Control", "SepsisvsControl")

head(design)

# fit <- lmFit(expr, design)

# fit <- eBayes(fit)#Bayesian

# ## DEG results

# results <- summary(decideTests(fit, lfc = 0.5))

# results

## arrayWeights

expr <- GSE28750\_anno

aw <- arrayWeights(expr, design = design)

aw

barplot(aw)

# fit <- lmFit(expr\_anno, design)#linear fitting

fit <- lmFit(expr, design, weights = aw)

fit <- eBayes(fit)#Bayesian

## DEG results

results <- summary(decideTests(fit, lfc = 0.5))

results

# number = Inf: all differential analysis results; SepsisvsControl: represents the type to be compared

res\_DEG <- topTable(fit, coef = "SepsisvsControl", number = Inf)

# res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1)#here p is adjusted p

res\_DEG\_sig <- topTable(fit, coef = "SepsisvsControl", n = Inf, p.value = 0.05, lfc = 1.2)

GSE28750\_DEG1 <- topTable(fit, coef = 2, n = Inf)

identical(res\_DEG, GSE28750\_DEG1)

GSE28750\_DEG <- topTable(fit, coef = 2, n = Inf) %>% # coef = 1 when there's no contrast.matrix()

rownames\_to\_column(var = "symbol")

# Filter results with |logFC|>0.5 and p<0.05 significant res

GSE28750\_DEG\_p <- GSE28750\_DEG %>%

dplyr::filter(abs(logFC) > 1.2 , P.Value < 0.05)

### save data

save(GSE28750\_DEG, GSE28750\_DEG\_p,

file = ".\\GSE28750\_DEG.Rda")

### Extract genes for analysis ----

rm(list = ls())

load(".\\Validated dataset\\GSE28750\\GSE28750\_DEG.Rda")

load(".\\Validated dataset\\GSE28750\\GSE28750\_processed.Rda")

### Extract key gene expression

# df <- c("GPR84","HK3","MCEMP1","S100A12",'CLEC5A',"CLEC4D")

df <- c("GPR84","HK3","S100A12",'CLEC5A',"CLEC4D")

library(tidyverse)

GSE28750\_keygene <- GSE28750\_anno %>% rownames\_to\_column(var = "symbol")

GSE28750\_keygene <- GSE28750\_keygene %>%

filter(GSE28750\_keygene$symbol %in% df ) %>%

column\_to\_rownames("symbol")

# GSE28750\_keygene\_1 <- GSE28750\_anno %>% rownames\_to\_column(var = "symbol")

# GSE28750\_keygene\_1 <- GSE28750\_keygene\_1 %>% filter(GSE28750\_keygene\_1$symbol %in% df)

# identical(GSE28750\_keygene,GSE28750\_keygene\_1)

### Extract key gene differential statistics

GSE28750\_DEG\_keygene <- GSE28750\_DEG %>%

filter(GSE28750\_DEG$symbol %in% df) %>% column\_to\_rownames("symbol")

identical(colnames(GSE28750\_keygene),GSE28750\_targets$sample\_id)

### Save data

write.table(GSE28750\_keygene, file = ".\\Validated dataset\\GSE28750\\GSE28750\_keygene.txt",

sep = "\t",row.names = T,col.names = T,quote = F)

write.table(GSE28750\_DEG\_keygene, file = ".\\Validated dataset\\GSE28750\\GSE28750\_DEG\_keygene.txt",

sep = "\t",row.names = T,col.names = T,quote = F)

### Draw grouped violin plots ###

## Data organization

library(tidyverse)

GSE28750\_keygene\_t <- t(GSE28750\_keygene) %>% as.data.frame() %>% rownames\_to\_column("sample\_id")

GSE28750\_targets\_extract <- GSE28750\_targets %>% dplyr::select(1,4)

identical(GSE28750\_keygene\_t$sample\_id,GSE28750\_targets\_extract$sample\_id)

GSE28750\_keygene\_merge <- inner\_join(GSE28750\_keygene\_t,GSE28750\_targets\_extract,by="sample\_id")

GSE28750\_violin\_data <- gather(GSE28750\_keygene\_merge,

category, value,-c(group,sample\_id)) #wide to long format conversion code: 2119

# library(reshape2)

# GSE28750\_keygene\_merge$group <- as.factor(GSE28750\_keygene\_merge$group)

# GSE28750\_violin\_data\_1 <- reshape2::melt(GSE28750\_keygene\_merge, id.vars=c('sample\_id','group'),

# variable.name='gene',

# measure.vars=c("GPR84","HK3","MCEMP1","S100A12",'CLEC5A',"CLEC4D"),

# value.name='value')

# # id.vars: identifier variables (remain on columns, unchanged)

# # measure.vars: variables to be melted into one column

# # variable.name: name for the new column, defaults to "variable" if not specified

# # value.name: name for the column containing the values

fix(GSE28750\_violin\_data)

# See https://www.jianshu.com/p/8c3f74b55154; https://zhuanlan.zhihu.com/p/524095571;

pdf('F:\\Decoding Helix Courses\\GEOdatasets\_sepsis1\\DEG\_Sepsis\\Validated datasets\\GSE28750\\figures\\GSE28750\_vio.pdf',10,5)

ggplot(GSE28750\_violin\_data, aes(x=category, y=value, fill=group))+

geom\_violin(position = position\_dodge(width=0.8),

# trim = FALSE,

scale = 'area')+

geom\_boxplot(position = position\_dodge(width = 0.8),

outlier.size = 0, width = 0.1, show.legend = FALSE)+

scale\_fill\_manual(values = c('Sepsis'='orange','Control'='blue'))+

# geom\_text(inherit.aes = F,data = FDR\_text\_table, aes(x=gene, y=height, label=annotations), size=2.5, fontface='bold')+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsl,xend=xel,y=ysl,yend=yel))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsr,xend=xer,y=ysr,yend=yer))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsh,xend=xeh,y=ysh,yend=yeh))+

theme\_bw()+theme(panel.grid = element\_line(colour = 'white'), plot.title = element\_text(hjust = 0.5))+

ylab('normalized\_value')+xlab('genes')+ylim(3,18)+

ggtitle('Violin plot of multiple genes')

dev.off()

GSE28750\_vio <- ggplot(GSE28750\_violin\_data, aes(x=category, y=value, fill=group))+

geom\_violin(position = position\_dodge(width=0.8),

# trim = FALSE,

scale = 'area')+

geom\_boxplot(position = position\_dodge(width =0.8),

outlier.size = 0, width = 0.1, show.legend = FALSE)+

scale\_fill\_manual(values = c('Sepsis'='orange','Control'='blue'))+

# geom\_text(inherit.aes = F,data = FDR\_text\_table, aes(x=gene, y=height, label=annotations), size=2.5, fontface='bold')+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsl,xend=xel,y=ysl,yend=yel))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsr,xend=xer,y=ysr,yend=yer))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsh,xend=xeh,y=ysh,yend=yeh))+

theme\_bw()+theme(panel.grid = element\_line(colour = 'white'), plot.title = element\_text(hjust = 0.5))+

ylab('normalized\_value')+xlab('genes')+ylim(3,18)+

ggtitle('Violin plot of multiple genes')

GSE28750\_vio

dev.off()

#### 13 ceRNA ----

##########################################

# See F:\\Decoding Helix Courses\\GEOdatasets\_sepsis1\\DEG\_Sepsis\\starbase\_prediction.R for miRNA-lncRNA and lncRNA-miRNA sections

##########################################

# #Read original lncRNA and miRNA interaction data

# mir\_lnc=read.table("starbase/Human/lncRNA\_miRNA\_interaction.txt", sep="\t", stringsAsFactors=F, header=T)

# #Extract only miRNAname and geneName columns and remove duplicate rows

# mir\_lnc=unique(mir\_lnc[,c("miRNAname","geneName")])

# #View first six rows

# head(mir\_lnc)

# #Convert dataframe to list with lncRNA names as list names

# lnc\_mir\_list=unstack(mir\_lnc,miRNAname~geneName)

# #View first six items in the list

# head(lnc\_mir\_list)

# #Convert dataframe to list with miRNA names as list names

# mir\_lnc\_list=unstack(mir\_lnc,geneName~miRNAname)

# #View first six items in the list

# head(mir\_lnc\_list)

# #Set random seed for reproducibility (commented out)

# # set.seed(123)

# # #Randomly select 100 miRNAs to simulate candidate miRNAs

# # mir\_candidate2=sample(names(mir\_lnc\_list),100)

# mir\_candidate2 <- c("hsa-miR-4505",

# "hsa-miR-574-5p",

# "hsa-miR-4710",

# "hsa-miR-5589-5p",

# "hsa-miR-5004-5p",

# "hsa-miR-765",

# "hsa-miR-3189-3p",

# "hsa-miR-4534",

# "hsa-miR-4306",

# "hsa-miR-185-5p",

# "hsa-miR-4729",

# "hsa-miR-5696",

# "hsa-miR-6835-3p",

# "hsa-miR-4644",

# "hsa-miR-548e-5p",

# "hsa-miR-6780a-5p",

# "hsa-miR-8085",

# "hsa-miR-4446-3p")

# #Get subset of the list to get miRNA-lncRNA interactions, then convert to dataframe

# mir\_target2=stack(mir\_lnc\_list[mir\_candidate2])

# #Rename dataframe columns

# names(mir\_target2)=c("lncRNA","miRNA")

# table(mir\_target2$miRNA)

# write.table(mir\_target2, file = ".\\STRING\\mir\_target2.txt",

# sep = "\t",row.names = F,col.names = T,quote = F)

#### Extract lincRNAs corresponding to miRNAs commonly predicted by three databases for E2F1 ----

rm(list = ls())

#Read original lncRNA and miRNA interaction data

mir\_lnc=read.table("starbase/Human/lncRNA\_miRNA\_interaction.txt", sep="\t", stringsAsFactors=F, header=T)

#Extract only miRNAname and geneName columns and remove duplicate rows

mir\_lnc=unique(mir\_lnc[,c("miRNAname","geneName")])

# These two steps are equivalent to the above step

# mir\_lnc\_1 <- mir\_lnc[,c("miRNAname","geneName")]

# mir\_lnc\_2 <- unique(mir\_lnc\_1)

#View first six rows

head(mir\_lnc)

#Convert dataframe to list with lncRNA names as list names

lnc\_mir\_list=unstack(mir\_lnc,miRNAname~geneName)

head(lnc\_mir\_list) #View first six items in the list

#Convert dataframe to list with miRNA names as list names

mir\_lnc\_list=unstack(mir\_lnc,geneName~miRNAname)

head(mir\_lnc\_list) #View first six items in the list

#Get miRNAs commonly predicted by three databases from venn website analysis tool

mir\_candidate1 <- read.table("F:/Decoding Helix Courses/GEOdatasets\_sepsis1/DEG\_Sepsis/STRING/E2F1/venn\_E2F1\_miRNA\_3com.txt",

sep="\t", stringsAsFactors=F, header=F)

mir\_candidate2 <- mir\_candidate1$V1

#Get subset of the list to get miRNA-lncRNA interactions, then convert to dataframe

mir\_target2=stack(mir\_lnc\_list[mir\_candidate2])

#Rename dataframe columns

names(mir\_target2)=c("lncRNA","miRNA")

table(mir\_target2$miRNA)

a <- table(mir\_target2$"lncRNA")

# b <- table(mir\_target2$lncRNA)

# identical(a,b)

library(tidyverse)

lncRNA\_rank <- a %>% as.data.frame() %>% arrange(.,desc(Freq))

# b <- table(com\_lncRNA\_miRNA\_dup\_LINC$geneName) %>% as.data.frame() %>% arrange(.,desc(Freq))

write.table(lncRNA\_rank, file = ".\\STRING\\E2F1\\lncRNA\_rank\_E2F1.txt",

sep = "\t",row.names = F,col.names = T,quote = F)

### Extract top 15 lncRNAs with most miRNA connections

lncRNA\_15 <- read.table("F:/Decoding Helix Courses/GEOdatasets\_sepsis1/DEG\_Sepsis/STRING/E2F1/lncRNA\_rank\_E2F1(15).txt",

sep="\t", stringsAsFactors=F, header=F)

mir\_target3 <- mir\_target2[mir\_target2$lncRNA %in% lncRNA\_15$V1,]

write.table(mir\_target3, file = ".\\STRING\\E2F1\\lncRNA(15)\_mirRNA.txt",

sep = "\t",row.names = F,col.names = T,quote = F)

##### 14 Correlation ----

#### 1. Correlation between core genes ----

keygene <- read.table(".\\Validated datasets\\GSE28750\\GSE28750\_keygene.txt",sep="\t",header=T,check.names=F,row.names = 1) #Read gene expression file

fix(keygene) #View data

keygene=t(keygene) #Transpose the data

keygene <- keygene[ ,-2]#Remove CLEC5A

keygene\_cor <- cor(keygene) #Calculate correlation analysis

library(corrplot)

pdf('./Correlation/keygene\_four/GSE28750\_keygene\_cor.pdf',width = 8,height = 8)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

# See: F:/Decoding Helix Courses/Complete Code Breakthrough Bioinformatics Data Mining/Bioinformatics System Course Lower Section Code/Level 3/L71\_Codes/3.phenotype/cell cycle/cell\_cycle.R code: 74

# col3 <- colorRampPalette(c("blue", "white", "red")) #Custom gradient colors corrgram can also specify See "R Language Practical" First Edition P259

# See: F:/Decoding Helix Courses/Complete Code Breakthrough Bioinformatics Data Mining/Bioinformatics System Course Lower Section Code/Level 3/L71\_Codes/4.cross/cross/cross.R code: 27~34

# color1 <- colorRampPalette(c("blue","red"))

# corrplot(cor,

# method = "circle",

# order = "hclust", #Clustering

# addCoef.col = "black",

# addrect = 2, #Number of rectangles drawn on the graph according to hierarchical clustering, only valid when order = "hclust"

# col = color1(100))#Add colors, 100 indicates uniformity of gradient colors, smaller values make boundaries clearer

# dev.off()

cor.plot <- corrplot(corr = keygene\_cor,

col=col3(10),

type="upper",

tl.pos="d", #tl.pos=”d” specifies text label (variable name) position on the diagonal

#tl.pos=”n” #means no text labels

# col="black",

tl.cex = 0.6) #Plot upper right Method defaults to "circle"

#tl.cex = 0.75 ＃Text label (variable name) size, default is 0.75 times

cor.plot <- corrplot(corr = keygene\_cor,

add=TRUE,

type="lower",

# col=col3(10), #col=col3(10) #Color specified gradient colors 10 gradient levels

# method="color",

method="number",

col="black",

# addCoef.col="black",

diag=FALSE,

tl.pos="n",

cl.pos="n",

number.cex = 0.7) #Plot lower left Method "color" #Size of correlation coefficient text in the graph, default is 0.7 times

dev.off()

#### 2. Correlation between genes and immune cells ----

rm(list=ls())

library(e1071)

library(parallel)

library(preprocessCore)

# if(!dir.exists("CIBERSORT")) dir.create("CIBERSORT")

setwd("F:/Decoding Helix Courses/GEOdatasets\_sepsis1/DEG\_Sepsis/Correlation")

source("./CIBERSORT.R")

if(!dir.exists("CIBERSORT\_GSE28750"))dir.create("CIBERSORT\_GSE28750")

#Assign "LM22.txt" to sig\_matrix, then the program will automatically read this file in this directory

sig\_matrix <- "LM22.txt" # "LM22.txt" is the annotation file for 22 immune cells

load("./GSE28750\_processed.Rda")

library(tidyverse)

GSE28750\_anno <- GSE28750\_anno %>% rownames\_to\_column("Gene")

write.table(GSE28750\_anno, file = "GSE28750\_anno.txt",sep="\t", quote=F, col.names=T,row.names = F)

mixture\_file = 'GSE28750\_anno.txt' # GSE137340 dataset expression data (note to adjust the first row of the table)

# GSE28750\_anno <- GSE28750\_anno %>% rownames\_to\_column("Gene")

# #If the above step is not done, the first row will shift left when saving. So adjust the format after saving.

# write.table(GSE28750\_anno, file = "GSE28750\_anno.txt",sep="\t", quote=F, col.names=T,row.names = T)

# mixture\_file = 'GSE137340\_anno.txt' # GSE137340 dataset expression data (note to adjust the first row of the table)

# perm indicates number of permutations, more times means more stable results

res\_cibersort <- CIBERSORT(sig\_matrix,

mixture\_file,

perm=100, #perm, number of permutation tests, default is 0, >100 can calculate significance p-values

QN=TRUE) #QN, whether to standardize the input expression matrix

#Equivalent to calculating 100 times continuously to see if the results are random

# Create a normalization function

normalize <- function(x){# (x-min(x))/(X's max-x's min)

return((x-min(x))/(max(x)-min(x)))}

# Use normalize to standardize CIBERSORT scores

CIBERSORT\_Score <- res\_cibersort

CIBERSORT\_Score\_GSE28750 <- normalize(CIBERSORT\_Score)

save(CIBERSORT\_Score\_GSE28750, CIBERSORT\_Score,

file = "./CIBERSORT\_GSE28750/CIBERSORT\_Score\_GSE28750.Rdata") #Save intermediate files

load("./CIBERSORT\_GSE28750/CIBERSORT\_Score\_GSE28750.Rdata")

CIBERSORT\_Score\_GSE28750 <- CIBERSORT\_Score\_GSE28750[,1:22]

#colSums sums columns

ciber.res <- CIBERSORT\_Score\_GSE28750[,colSums(CIBERSORT\_Score\_GSE28750) > 0] #Remove cells with all zero abundance

### Core gene expression levels

keygene <- read.table("F:\\Decoding Helix Courses\\GEOdatasets\_sepsis1\\DEG\_Sepsis\\Validated datasets\\GSE28750\\GSE28750\_keygene.txt",

sep="\t",header=T,check.names=F,row.names = 1) #Read gene expression file

# fix(keygene) #View data

keygene=t(keygene) #Transpose the data

keygene <- keygene[ ,-2] #Remove CLEC5A

category, value,-c(group,sample\_id)) # wide to long format conversion code: 2119

# library(reshape2)

# GSE28750\_keygene\_merge$group <- as.factor(GSE28750\_keygene\_merge$group)

# GSE28750\_violin\_data\_1 <- reshape2::melt(GSE28750\_keygene\_merge, id.vars=c('sample\_id','group'),

# variable.name='gene',

# measure.vars=c("GPR84","HK3","MCEMP1","S100A12",'CLEC5A',"CLEC4D"),

# value.name='value')

# # id.vars: identifier variables (remain on columns, unchanged)

# # measure.vars: variables to be melted into one column

# # variable.name: name for the new column, defaults to "variable" if not specified

# # value.name: name for the column containing the values

fix(GSE28750\_violin\_data)

# See https://www.jianshu.com/p/8c3f74b55154; https://zhuanlan.zhihu.com/p/524095571;

pdf('F:\\Decoding Helix Courses\\GEOdatasets\_sepsis1\\DEG\_Sepsis\\Validated datasets\\GSE28750\\figures\\GSE28750\_vio.pdf',10,5)

ggplot(GSE28750\_violin\_data, aes(x=category, y=value, fill=group))+

geom\_violin(position = position\_dodge(width=0.8),

# trim = FALSE,

scale = 'area')+

geom\_boxplot(position = position\_dodge(width = 0.8),

outlier.size = 0, width = 0.1, show.legend = FALSE)+

scale\_fill\_manual(values = c('Sepsis'='orange','Control'='blue'))+

# geom\_text(inherit.aes = F,data = FDR\_text\_table, aes(x=gene, y=height, label=annotations), size=2.5, fontface='bold')+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsl,xend=xel,y=ysl,yend=yel))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsr,xend=xer,y=ysr,yend=yer))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsh,xend=xeh,y=ysh,yend=yeh))+

theme\_bw()+theme(panel.grid = element\_line(colour = 'white'), plot.title = element\_text(hjust = 0.5))+

ylab('normalized\_value')+xlab('genes')+ylim(3,18)+

ggtitle('Violin plot of multiple genes')

dev.off()

GSE28750\_vio <- ggplot(GSE28750\_violin\_data, aes(x=category, y=value, fill=group))+

geom\_violin(position = position\_dodge(width=0.8),

# trim = FALSE,

scale = 'area')+

geom\_boxplot(position = position\_dodge(width =0.8),

outlier.size = 0, width = 0.1, show.legend = FALSE)+

scale\_fill\_manual(values = c('Sepsis'='orange','Control'='blue'))+

# geom\_text(inherit.aes = F,data = FDR\_text\_table, aes(x=gene, y=height, label=annotations), size=2.5, fontface='bold')+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsl,xend=xel,y=ysl,yend=yel))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsr,xend=xer,y=ysr,yend=yer))+

# geom\_segment(inherit.aes = F,data=segment\_table,aes(x=xsh,xend=xeh,y=ysh,yend=yeh))+

theme\_bw()+theme(panel.grid = element\_line(colour = 'white'), plot.title = element\_text(hjust = 0.5))+

ylab('normalized\_value')+xlab('genes')+ylim(3,18)+

ggtitle('Violin plot of multiple genes')

GSE28750\_vio

dev.off()

#### 15 ROC Analysis ----

rm(list = ls())

library(tidyverse)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS")

if(!dir.exists("ROC")) dir.create("ROC")

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS/ROC")

# load("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Validated dataset\\GSE28750\\GSE28750\_DEG.Rda")

load("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\GSE54514\\GSE54514\_processed.Rda")

## Extract key genes for analysis:

GetFactors <- c("GPR84","HK3","MCEMP1","S100A12",'CLEC5A',"CLEC4D")

# GetFactors <- c("GPR84","HK3","S100A12",'CLEC5A',"CLEC4D")

## Merge key gene expression files

GSE54514\_targets\_sur <- GSE54514\_targets %>%

mutate(sur=(ifelse(str\_detect(sample\_name, "survivor"),

ifelse(str\_detect(sample\_name, "non"),"nonsurvivor","survivor"),"normal")))

# GSE54514\_targets\_sur\_factor <- GSE54514\_targets\_sur %>% rownames\_to\_column("sample") %>%

# mutate(fustatus=factor(ifelse(GSE54514\_targets\_sur$sur == "nonsurvivor","1","0"),

# levels=c('1','0')))

# Notes on converting character to factor type:

# Character type needs to be converted to numeric first before converting to factor

# Or convert to character factor type; e.g.: level(normal,tumor)

GSE54514\_targets\_sur\_factor <- GSE54514\_targets\_sur %>% rownames\_to\_column("sample") %>%

mutate(fustatus=ifelse(GSE54514\_targets\_sur$sur == "nonsurvivor","1","0"))

GSE54514\_targets\_sur\_factor$fustatus <- as.numeric(GSE54514\_targets\_sur\_factor$fustatus)

GSE54514\_targets\_sur\_factor$fustatus <- as.factor(GSE54514\_targets\_sur\_factor$fustatus)

GSE54514\_anno\_GetFactors <- GSE54514\_anno[GetFactors,]

identical(GSE54514\_targets\_sur\_factor$sample,colnames(GSE54514\_anno\_GetFactors))

GSE54514\_anno\_GetFactors <- GSE54514\_anno\_GetFactors %>% t() %>% as.data.frame() %>% rownames\_to\_column("sample")

GSE54514\_anno\_GetFactors\_factor <- merge(GSE54514\_anno\_GetFactors,GSE54514\_targets\_sur\_factor,by="sample")

GSE54514\_anno\_GetFactors\_factor <- GSE54514\_anno\_GetFactors\_factor %>% column\_to\_rownames("sample")

# Univariate analysis:

GetFactors1 <- c("GPR84","HK3","S100A12",'CLEC5A',"CLEC4D")

XNames <- colnames(GSE54514\_anno\_GetFactors\_factor)[c(1,2,4,5,6)] ## N\_category is in the seventh row

YNames <- colnames(GSE54514\_anno\_GetFactors\_factor)[13]

df\_train\_all <- GSE54514\_anno\_GetFactors\_factor

df\_train <- df\_train\_all[!df\_train\_all$group=="Control",]

# df\_train <- df\_train\_all

save(df\_train , df\_train\_all,

file = ".\\df\_train.Rda")

source('../Lesson1\_Function/UniLog.R', encoding = "utf-8") ## Load R function

## We created a function below that can obtain univariate analysis results. Note: This uses an object-oriented approach.

UniVar <- lapply(XNames,function(x){

UniLog(df = df\_train,

x = x,

y = YNames)

})

library(plyr)

UniVar <- ldply(UniVar,data.frame) ## ldply is a function that performs the same operation on multiple objects, mainly for convenience.

UniVar

# Multivariate analysis:

# Here we decide to include P < 0.2 based on the univariate results of the simulated data into the multivariate model.

GetFactors <- UniVar$characteristics[which(UniVar$Pval < 0.2)] %>% as.character()

GetFactors

# save(GetFactors,file = './Lesson1\_Results/Tmp\_GetFactors.RData') ## Save results as RData file for future loading.

## Logistic regression for multivariate data ##

# fml: is the formula

fml <- as.formula(paste('fustatus', ## Outcome variable Y

paste(GetFactors, collapse='+'), sep=" ~ ")) ##

# glm: is the logistic regression function

mod.full <- glm(fml, ## Key glm function

data = df\_train, ## Training set name

family = binomial()) ## Logistic regression method

summary(mod.full) # View results

## Stepwise regression for multivariate data ## Stepwise regression considers multicollinearity

library(MASS)

# Create a null model first

mod.none <- glm(fustatus~1, # First create a null model

data = df\_train,

family = binomial())

mod.full <- glm(fml,

data = df\_train,

family = binomial())

stepAIC(mod.full) ## Method I: AIC, the smaller the better

# N\_category ~ T\_category + HPV\_status

#

# Df Deviance AIC

# <none> 213.89 219.89

# - HPV\_status 1 226.14 230.14 Removing HPV increases AIC from 219 to 230, so we shouldn't remove HPV

# - T\_category 1 229.09 233.09

mod\_step <- step(mod.none,scope = formula(mod.full),direction = 'forward',trace = F)

MultiSum <- summary(mod\_step)

MultiSum

## Table

MultiName <- as.character(UniVar$characteristics[UniVar$Pval < 0.2])

tmp\_Multi <- MultiSum$coefficients %>% data.frame()

MR <- tmp\_Multi[MultiName, ]$Estimate %>% round(.,2)

MP <- tmp\_Multi[MultiName, ]$Pr...z.. %>% round(.,4)

MulLog <- data.frame('characteristics' = MultiName,

'Rval' = MR,

'Pval' = MP)

Final\_Results <- merge.data.frame(UniVar,MulLog, by = "characteristics", all = T, sort = T) # Merge univariate and multivariate

Final\_Results

View(Final\_Results)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS/ROC")

write.table(Final\_Results, file = "Final\_Results.txt",sep = "\t",row.names = F,col.names = T,quote = F)

## Build Logistic regression formula ####

## This code (explaining univariate and multivariate logistic regression) is the same as

## (univariate and multivariate analysis where the dependent variable is binary distributed)

rm(list = ls())

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS/ROC")

load("./df\_train.Rda")

df\_train\_all <- df\_train\_all[ ,-3]

df\_train <- df\_train[ ,-3]

# GetFactors <- c("GPR84","HK3","S100A12",'CLEC5A',"CLEC4D","CLEC5A")

# GetFactors <- c("S100A12")

# GetFactors <- c("HK3")

# GetFactors <- c("CLEC4D")

# GetFactors <- c("GPR84")

GetFactors <- c("CLEC5A")

fml <- as.formula(paste('fustatus', ## Outcome variable Y in the formula

'~', ## Tilde symbol

paste(GetFactors,collapse = '+'),collapse = ' ')) ## Predictors X connected by "+" First use + then ~

# fml\_1 <- as.formula(fustatus~CLEC5A)

# identical(fml,fml\_1)

library(rms)

# df\_train <- df\_train\_all

ddist <- datadist(df\_train) ## Environment variable setting, necessary for later nomogram creation

options(datadist = 'ddist')

#### Step I Logistic regression ###

## Method I ##

# lrm(): simple linear regression; glm is more powerful than lm, can do discrete outcome models, etc.;

# glm is the generalized linear regression function, of course you can also use it for linear regression

# lm is the linear regression function, cannot fit generalized linear regression models

mod.bi <- lrm(fml, ## Input logistic regression formula fml lrm——Logistic Regression Model

data = df\_train) ## Define dataset

#summary(mod.bi)

mod.bi# View regression coefficients Coef and p-values

## ROC

# # The following two codes are for multivariate indicators

# fml <- as.formula(paste('fustatus', ## Outcome variable Y in the formula

# '~', ## Tilde symbol

# paste(GetFactors,collapse = '+'),collapse = ' ')) ## Predictors X connected by "+"

# fit2 <- glm(fml,x=T,y=T,data = df\_train, family = "binomial") ## Establish logistic regression model

library("pROC") ## Load package

mycol <- rainbow(10) ## Define some colors, you can set your own

# pdf(paste0('Try\_log','\_Plot5\_ROC.pdf'),height=12,width=12)

auc.out <- c()

x <- plot.roc(df\_train$fustatus , # y-value, i.e. outcome variable

# fit2$fitted.values,# Different from univariate ROC

# df\_train$GPR84,# x-value, i.e. whether drinking

# df\_train$HK3,

df\_train$CLEC5A,

# df\_train$CLEC5A,

# df\_train$S100A12,

ylim=c(0,1), # Y-axis range

xlim=c(1,0), # X-axis range

# smooth=T, # Draw smooth curve

ci=TRUE, # Calculate 95%CI confidence interval

main='Receiver Operating Characteristic', # Define a graph name

# print.thres="best", # Write the threshold on the graph where sensitivity + specificity is maximized

col=mycol[2],# Line color

lwd=2, # Line thickness

legacy.axes=T, ## X-axis is "1-specificity", from 0 to 1

newpage = F) # Don't start a new page (for plotting)

ci.lower <- round(as.numeric(x$ci[1]),3) # Lower bound of confidence interval

ci.upper <- round(as.numeric(x$ci[3]),3) # Upper bound of confidence interval

auc.ci <- c('CLEC5A',round(as.numeric(x$auc),3),paste(ci.lower,ci.upper,sep="-")) # Format for outputting 95%CI

auc.out <- rbind(auc.out,auc.ci) # Combine together

### Output AUC, AUC CI to file

auc.out <- as.data.frame(auc.out)

colnames(auc.out) <- c('Factors',"AUC","AUC CI")

### Draw legend

legend.name <- paste('CLEC5A',"AUC",auc.out$AUC,sep=" ") # Legend we want to output

legend("bottomright", # Bottom right of the graph

legend=legend.name, # Output text

col = mycol[2:length(df)], # Color definition

lwd = 2, # Line thickness

bty="n",

cex = 0.8) # Don't draw border

dev.off()

#### 16 Core Gene Correlation Heatmap ####

#### 16.1 Correlation between core genes and immune/inflammatory genes ----

rm(list = ls())

library(tidyverse)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS")

## Read expression files of immune and inflammatory genes

gene\_immu <- read.csv(".\\Validated dataset\\GSE28750\\GeneCards-immune regulators.csv",

sep=",",header=T,check.names=F)

gene\_imfla <- read.csv(".\\Validated dataset\\GSE28750\\GeneCards-Inflammatory factor.csv",

sep=",",header=T,check.names=F)

gene\_inter <- intersect(gene\_immu$"Gene Symbol",gene\_imfla$"Gene Symbol")

## Sort gene\_immu by Relevance score and extract genes with `Relevance score` > 20

gene\_immu <- gene\_immu %>% arrange("Relevance score")

gene <- gene\_immu$`Gene Symbol`[which(gene\_immu$`Relevance score` > 20)] %>% as.character()

load(".\\Validated dataset\\GSE28750\\GSE28750\_DEG.Rda")

load(".\\Validated dataset\\GSE28750\\GSE28750\_processed.Rda")

gene\_com <- intersect(gene,GSE28750\_DEG\_p$symbol)

## Sort gene\_imfla by Relevance score and extract genes with `Relevance score` > 20

gene\_imfla <- gene\_imfla %>% arrange("Relevance score")

gene1 <- gene\_imfla$`Gene Symbol`[which(gene\_imfla$`Relevance score` > 20)] %>% as.character()

gene1\_com <- intersect(gene1,GSE28750\_DEG\_p$symbol)

com <- intersect(gene\_com,gene1\_com)

## Calculate correlation between core genes and immune/inflammatory genes:

# Extract expression of core genes in GSE28750

keygene <- read.table("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Validated dataset\\GSE28750\\GSE28750\_keygene.txt",

sep="\t",header=T,check.names=F,row.names = 1) # Read gene expression file

keygene <- keygene[-2,]

# Extract expression of immune genes in GSE28750

GSE28750\_anno <- GSE28750\_anno %>% rownames\_to\_column("sample")

GSE28750\_anno\_immu <- GSE28750\_anno %>% filter(GSE28750\_anno$sample %in% gene\_com)

# Extract expression of inflammatory genes in GSE28750

GSE28750\_anno\_imfla <- GSE28750\_anno %>% filter(GSE28750\_anno$sample %in% gene1\_com)

## Data format conversion

keygene=t(keygene) # Transpose the data

GSE28750\_anno\_immu <- GSE28750\_anno\_immu %>% column\_to\_rownames("sample")

GSE28750\_anno\_immu <- t(GSE28750\_anno\_immu)

GSE28750\_anno\_imfla <- GSE28750\_anno\_imfla %>% column\_to\_rownames("sample")

GSE28750\_anno\_imfla <- t(GSE28750\_anno\_imfla)

identical(rownames(GSE28750\_anno\_immu),rownames(GSE28750\_anno\_imfla))

identical(rownames(GSE28750\_anno\_immu),rownames(keygene))

## Calculate correlation using corr.test()

library(psych)

# See webpage: https://www.jianshu.com/p/d86ddf8fd48f

cor\_t <- corr.test(keygene, GSE28750\_anno\_immu, method = "pearson",adjust="none")

## Extract correlation coefficients and p-values

cor\_t\_r <- cor\_t$r

cor\_t\_p <- cor\_t$p

# As above, cor\_t\_r == cor == merge\_cor\_1

# But cor\_t\_p is not equal to cor\_res\_2

cmt <- cor\_t\_r

# cmt <- cmt %>% as.data.frame() %>% arrange(CLEC4D) # arrange: means ascending order

cmt <- t(cmt)

# cmt <- cmt %>% as.data.frame() %>% arrange(desc(Clec4D)) # After adding desc, it means descending order

cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D)

cmt <- t(cmt)

pmt <- cor\_t\_p[ ,colnames(cmt)]

identical(colnames(pmt),colnames(cmt))

pmt\_1 <- cor\_t\_p

# Determine significance

if (!is.null(pmt)){

ssmt <- pmt< 0.01

pmt[ssmt] <-'\*\*'

smt <- pmt >0.01& pmt <0.05

pmt[smt] <- '\*'

pmt[!ssmt&!smt]<- ''

} else {

pmt <- F

}

# Visualization

# mycol <- ggplot2::alpha(rainbow(ncol(cor\_t\_r)), 0.7) # Create color palette (70% transparency)

# col3 <- colorRampPalette(c("blue", "white", "red")) # Custom gradient colors, corrgram can also specify, see "R Language Practical" First Edition P259

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_immu.PDF", 15, 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt,scale = "none",

cluster\_row = F, cluster

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_immu.PDF", width = 15, height = 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt, scale = "none",

cluster\_row = FALSE, cluster\_col = FALSE,

border = NA,

display\_numbers = pmt, fontsize\_number = 12, number\_color = "white",

color = col3(10),

cellwidth = 20, cellheight = 20)

dev.off()

## Calculate correlation using corr.test()

library(psych)

# See webpage: https://www.jianshu.com/p/d86ddf8fd48f

cor\_t <- corr.test(keygene, GSE28750\_anno\_imfla, method = "pearson", adjust = "none")

## Extract correlation coefficients and p-values

cor\_t\_r <- cor\_t$r

cor\_t\_p <- cor\_t$p

# As shown above, cor\_t\_r == cor == merge\_cor\_1

# But cor\_t\_p is not equal to cor\_res\_2

cmt <- cor\_t\_r

# cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D) # arrange: means ascending order

cmt <- t(cmt)

cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D) # After adding desc, it means descending order

cmt <- t(cmt)

pmt <- cor\_t\_p[, colnames(cmt)]

# Determine significance

if (!is.null(pmt)) {

ssmt <- pmt < 0.01

pmt[ssmt] <- '\*\*'

smt <- pmt > 0.01 & pmt < 0.05

pmt[smt] <- '\*'

pmt[!ssmt & !smt] <- ''

} else {

pmt <- FALSE

}

# Visualization

# mycol <- ggplot2::alpha(rainbow(ncol(cor\_t\_r)), 0.7) # Create color palette (70% transparency)

# col3 <- colorRampPalette(c("blue", "white", "red")) # Custom gradient colors, corrgram can also specify, see "R Language Practical" First Edition P259

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_imfla.PDF", width = 15, height = 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt, scale = "none",

cluster\_row = FALSE, cluster\_col = FALSE,

border = NA,

display\_numbers = pmt, fontsize\_number = 12, number\_color = "white",

color = col3(10),

cellwidth = 20, cellheight = 20)

dev.off()

#### 16.2 Correlation between core genes and metabolic genes ----

rm(list = ls())

library(tidyverse)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS")

## Read metabolic gene expression file

gene\_meta <- read.csv(".\\Validated dataset\\GSE28750\\GeneCards-metabolic\_genes.csv",

sep = ",", header = TRUE, check.names = FALSE)

## Sort gene\_immu by Relevance score and extract genes with `Relevance score` > 20

# Note: This seems to be using gene\_meta instead of gene\_immu

gene\_meta\_1 <- gene\_meta %>% arrange("Relevance score")

gene\_meta\_2 <- gene\_meta %>% arrange(desc(gene\_meta$`Relevance score`))

identical(gene\_meta, gene\_meta\_1)

identical(gene\_meta\_1, gene\_meta\_2)

gene <- gene\_meta$`Gene Symbol`[which(gene\_meta$`Relevance score` > 20)] %>% as.character()

load(".\\Validated dataset\\GSE28750\\GSE28750\_DEG.Rda")

load(".\\Validated dataset\\GSE28750\\GSE28750\_processed.Rda")

gene\_com <- intersect(gene, GSE28750\_DEG\_p$symbol)

## Calculate correlation between core genes and metabolic genes:

# Extract expression of core genes in GSE28750

keygene <- read.table("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Validated dataset\\GSE28750\\GSE28750\_keygene.txt",

sep = "\t", header = TRUE, check.names = FALSE, row.names = 1) # Read gene expression file

keygene <- keygene[-2, ]

# Extract expression of metabolic genes in GSE28750

GSE28750\_anno <- GSE28750\_anno %>% rownames\_to\_column("sample")

GSE28750\_anno\_meta <- GSE28750\_anno %>% filter(GSE28750\_anno$sample %in% gene\_com)

## Data format conversion

keygene <- t(keygene) # Transpose the data

GSE28750\_anno\_meta <- GSE28750\_anno\_meta %>% column\_to\_rownames("sample")

GSE28750\_anno\_meta <- t(GSE28750\_anno\_meta)

identical(rownames(GSE28750\_anno\_meta), rownames(keygene))

## Calculate correlation using corr.test()

library(psych)

# See webpage: https://www.jianshu.com/p/d86ddf8fd48f

cor\_t <- corr.test(keygene, GSE28750\_anno\_meta, method = "pearson", adjust = "none")

## Extract correlation coefficients and p-values

cor\_t\_r <- cor\_t$r

cor\_t\_p <- cor\_t$p

# As shown above, cor\_t\_r == cor == merge\_cor\_1

# But cor\_t\_p is not equal to cor\_res\_2

cmt <- cor\_t\_r

# cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D) # arrange: means ascending order

cmt <- t(cmt)

# cmt <- cmt %>% as.data.frame() %>% arrange(desc(Clec4D)) # After adding desc, it means descending order

cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D)

cmt <- t(cmt)

pmt <- cor\_t\_p[, colnames(cmt)]

identical(colnames(pmt), colnames(cmt))

pmt\_1 <- cor\_t\_p

# Determine significance

if (!is.null(pmt)) {

ssmt <- pmt < 0.01

pmt[ssmt] <- '\*\*'

smt <- pmt > 0.01 & pmt < 0.05

pmt[smt] <- '\*'

pmt[!ssmt & !smt] <- ''

} else {

pmt <- FALSE

}

# Visualization

# mycol <- ggplot2::alpha(rainbow(ncol(cor\_t\_r)), 0.7) # Create color palette (70% transparency)

# col3 <- colorRampPalette(c("blue", "white", "red")) # Custom gradient colors, corrgram can also specify, see "R Language Practical" First Edition P259

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_meta.PDF", width = 15, height = 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt, scale = "none",

cluster\_row = FALSE, cluster\_col = FALSE,

border = NA,

display\_numbers = pmt, fontsize\_number = 12, number\_color = "white",

color = col3(10),

cellwidth = 20, cellheight = 20)

dev.off()

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_immu.PDF", width = 15, height = 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt, scale = "none",

cluster\_row = FALSE, cluster\_col = FALSE,

border = NA,

display\_numbers = pmt, fontsize\_number = 12, number\_color = "white",

color = col3(10),

cellwidth = 20, cellheight = 20)

dev.off()

## Calculate correlation using corr.test()

library(psych)

# See webpage: https://www.jianshu.com/p/d86ddf8fd48f

cor\_t <- corr.test(keygene, GSE28750\_anno\_imfla, method = "pearson", adjust = "none")

## Extract correlation coefficients and p-values

cor\_t\_r <- cor\_t$r

cor\_t\_p <- cor\_t$p

# As shown above, cor\_t\_r == cor == merge\_cor\_1

# But cor\_t\_p is not equal to cor\_res\_2

cmt <- cor\_t\_r

# cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D) # arrange: means ascending order

cmt <- t(cmt)

cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D) # After adding desc, it means descending order

cmt <- t(cmt)

pmt <- cor\_t\_p[, colnames(cmt)]

# Determine significance

if (!is.null(pmt)) {

ssmt <- pmt < 0.01

pmt[ssmt] <- '\*\*'

smt <- pmt > 0.01 & pmt < 0.05

pmt[smt] <- '\*'

pmt[!ssmt & !smt] <- ''

} else {

pmt <- FALSE

}

# Visualization

# mycol <- ggplot2::alpha(rainbow(ncol(cor\_t\_r)), 0.7) # Create color palette (70% transparency)

# col3 <- colorRampPalette(c("blue", "white", "red")) # Custom gradient colors, corrgram can also specify, see "R Language Practical" First Edition P259

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_imfla.PDF", width = 15, height = 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt, scale = "none",

cluster\_row = FALSE, cluster\_col = FALSE,

border = NA,

display\_numbers = pmt, fontsize\_number = 12, number\_color = "white",

color = col3(10),

cellwidth = 20, cellheight = 20)

dev.off()

#### 16.2 Correlation between core genes and metabolic genes ----

rm(list = ls())

library(tidyverse)

setwd("F:/Decoding Helix Courses/GEOdatasets\_spsis1/DEG\_SPSIS")

## Read metabolic gene expression file

gene\_meta <- read.csv(".\\Validated dataset\\GSE28750\\GeneCards-metabolic\_genes.csv",

sep = ",", header = TRUE, check.names = FALSE)

## Sort gene\_immu by Relevance score and extract genes with `Relevance score` > 20

# Note: This seems to be using gene\_meta instead of gene\_immu

gene\_meta\_1 <- gene\_meta %>% arrange("Relevance score")

gene\_meta\_2 <- gene\_meta %>% arrange(desc(gene\_meta$`Relevance score`))

identical(gene\_meta, gene\_meta\_1)

identical(gene\_meta\_1, gene\_meta\_2)

gene <- gene\_meta$`Gene Symbol`[which(gene\_meta$`Relevance score` > 20)] %>% as.character()

load(".\\Validated dataset\\GSE28750\\GSE28750\_DEG.Rda")

load(".\\Validated dataset\\GSE28750\\GSE28750\_processed.Rda")

gene\_com <- intersect(gene, GSE28750\_DEG\_p$symbol)

## Calculate correlation between core genes and metabolic genes:

# Extract expression of core genes in GSE28750

keygene <- read.table("F:\\Decoding Helix Courses\\GEOdatasets\_spsis1\\DEG\_SPSIS\\Validated dataset\\GSE28750\\GSE28750\_keygene.txt",

sep = "\t", header = TRUE, check.names = FALSE, row.names = 1) # Read gene expression file

keygene <- keygene[-2, ]

# Extract expression of metabolic genes in GSE28750

GSE28750\_anno <- GSE28750\_anno %>% rownames\_to\_column("sample")

GSE28750\_anno\_meta <- GSE28750\_anno %>% filter(GSE28750\_anno$sample %in% gene\_com)

## Data format conversion

keygene <- t(keygene) # Transpose the data

GSE28750\_anno\_meta <- GSE28750\_anno\_meta %>% column\_to\_rownames("sample")

GSE28750\_anno\_meta <- t(GSE28750\_anno\_meta)

identical(rownames(GSE28750\_anno\_meta), rownames(keygene))

## Calculate correlation using corr.test()

library(psych)

# See webpage: https://www.jianshu.com/p/d86ddf8fd48f

cor\_t <- corr.test(keygene, GSE28750\_anno\_meta, method = "pearson", adjust = "none")

## Extract correlation coefficients and p-values

cor\_t\_r <- cor\_t$r

cor\_t\_p <- cor\_t$p

# As shown above, cor\_t\_r == cor == merge\_cor\_1

# But cor\_t\_p is not equal to cor\_res\_2

cmt <- cor\_t\_r

# cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D) # arrange: means ascending order

cmt <- t(cmt)

# cmt <- cmt %>% as.data.frame() %>% arrange(desc(Clec4D)) # After adding desc, it means descending order

cmt <- cmt %>% as.data.frame() %>% arrange(Clec4D)

cmt <- t(cmt)

pmt <- cor\_t\_p[, colnames(cmt)]

identical(colnames(pmt), colnames(cmt))

pmt\_1 <- cor\_t\_p

# Determine significance

if (!is.null(pmt)) {

ssmt <- pmt < 0.01

pmt[ssmt] <- '\*\*'

smt <- pmt > 0.01 & pmt < 0.05

pmt[smt] <- '\*'

pmt[!ssmt & !smt] <- ''

} else {

pmt <- FALSE

}

# Visualization

# mycol <- ggplot2::alpha(rainbow(ncol(cor\_t\_r)), 0.7) # Create color palette (70% transparency)

# col3 <- colorRampPalette(c("blue", "white", "red")) # Custom gradient colors, corrgram can also specify, see "R Language Practical" First Edition P259

library(pheatmap)

pdf("./Correlation/keygene\_four/keygene\_meta.PDF", width = 15, height = 6)

col3 <- colorRampPalette(c("#6D9EC1", "white", "#E46726"))

pheatmap(cmt, scale = "none",

cluster\_row = FALSE, cluster\_col = FALSE,

border = NA,

display\_numbers = pmt, fontsize\_number = 12, number\_color = "white",

color = col3(10),

cellwidth = 20, cellheight = 20)

dev.off()