# cellFindR 2.0, version for Seurat 3.1

#04.9.2020 Kevin Yu

# get\_matrix make log scaled.

# iterative change to sub

library(Seurat)

library(dplyr)

library(stringr)

library(ggplot2)

library(RColorBrewer)

dev.off()

# load data

# loading tenx data

# input: file\_loc = location of file to load

# res = resolution to run once

# proj\_name = name of project

# cutoff = high end cut off of num genes

# output saves to file\_loc a rdata file.

load\_tenx <- function(file\_loc, res = 0.1, proj\_name = 'tenx\_data', cutoff= 10000, mito = FALSE){

tenx.data <- Read10X(data.dir = file\_loc)

tenx <- CreateSeuratObject(counts = tenx.data, project = proj\_name, min.cells = 3, min.features = 200)

#processing

tenx[["percent.mt"]] <- PercentageFeatureSet(tenx, pattern = "^mt-")

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

VlnPlot(tenx, features = c("nFeature\_RNA", "nCount\_RNA", "percent.mt"), ncol = 3)

###############################

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

tenx <- FindVariableFeatures(tenx, selection.method = "vst", nfeatures = 2000)

top10 <- head(VariableFeatures(tenx), 10)

plot1 <- VariableFeaturePlot(tenx)

plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)

CombinePlots(plots = list(plot1, plot2))

all.genes <- rownames(tenx)

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

#####################################

tenx <- RunPCA(tenx, features = VariableFeatures(object = tenx))

tenx <- FindNeighbors(tenx, dims = 1:20)

tenx <- FindClusters(tenx, resolution = 0.1)

tenx <- RunUMAP(tenx, dims = 1:20, verbose = FALSE)

DimPlot(tenx, reduction = "umap")

saveRDS(tenx, file = paste(file\_loc,'/', proj\_name, ".rds", sep = ''))

}

##############

# asking if the grouping is a cluster

# input: tenx = tenx object

# thresh\_genes = threshold of genes at thresh\_val

# thresh\_val = value of the threshold in log space

# pval = cut off of pval for the signficance

is\_cluster <- function(tenx, thresh\_genes = 10, thresh\_val = log(2), pval = 1e-4){

val = 0 # groups that does not satisfy threshold genes

counter = 0 # groups that satisfy threshold genes

# loop through the identitiy

matrix\_output <- data.frame(row.names = row.names(tenx))

for (j in sort(unique(tenx@active.ident))){

if (sum(tenx@active.ident == j) < 5){

return(FALSE)

}

markers <- FindMarkers(tenx, ident.1 = j, min.pct = 0.25, base = exp(1))

markers <- markers[markers$p\_val\_adj < pval,]

#find if the 10th biggest is less than log2, sum

print(sort(markers$avg\_logFC, decreasing = TRUE)[thresh\_genes])

# if less than 10 significant genes

if (length((markers$avg\_logFC)) < 10){

val <- val + 1

} else if (sort(markers$avg\_logFC, decreasing = TRUE)[thresh\_genes] < thresh\_val){

#print(val)

val <- val + 1

} else{

counter = counter + 1

}

if (val > 1){

return(FALSE)

}

}

if (val > 1){

return(FALSE)

}

else{

return(TRUE)

}

}

# finds resolution that satisfy

# input: tenx object

# initial resolution of starting clustering

# how much to increment up

# threshold of genes

# value of the threshold

find\_res <- function(tenx, initial\_res = 0.1, jump = 0.1, thresh\_genes = 10, thresh\_val = log(2)) {

RES\_POST <- initial\_res # keeping

RES\_IT <- initial\_res # iterative

while(TRUE){

print(paste('Trying',RES\_IT, sep = ' '))

tenx <- FindNeighbors(tenx, dims = 1:20)

tenx <- FindClusters(tenx, resolution = RES\_IT)

# also check if theres only 1 cluster/ then can go up higher es

# Find number of clusters

length\_group <- length(unique(tenx@active.ident))

# if only one group then need to look deeper

if (length\_group == 1){

print(paste('noclusterfound',RES\_IT, sep = ' '))

# still not groups at 0.7 res stop and just step as 1

if (RES\_IT == 0.7){

print(paste('noclusterfound',RES\_IT, sep = ' '))

break

}

} else{

testing <- is\_cluster(tenx)

if (testing == FALSE){ # if not real group

print(paste('broke', RES\_IT, sep = ' '))

RES\_IT <- RES\_IT - jump

RES\_POST <- RES\_IT

print(RES\_POST)

break

} else{ # valid groups

RES\_POST <- RES\_IT

print(paste('ok',RES\_IT, sep = ' '))

}

}

RES\_IT <- RES\_IT + jump

}

# if there is only 1 group, return 0,

return(RES\_POST)

}

# getsubclustering

# input: tenx object

# location of output folder

# project\_name

sub\_clustering <- function(tenx, output\_folder = '.', proj\_name = 'proj\_name',

thresh\_genes = 10, thresh\_val = log(2)){

print('Running subclustering')

#public variables:

##resolution keeper:

res\_keep <- data.frame('cluster'= NA,'res'= NA, 'num\_clusters' =NA)

##cellname cluster:

celltocluster <- data.frame(row.names = colnames(tenx))

celltocluster$cellnames <- colnames(tenx)

#add cellfindr column to metadata as string

tenx@meta.data$cellfindr <- as.character(tenx@active.ident)

#what to iterate across:queue

lib\_c <- as.character(sort(unique(tenx@active.ident)))

while(length(lib\_c != 0)){

# set to first value

j <- lib\_c[1]

sub\_tenx <- subset(tenx, idents = toString(j))

print(paste('clustering ', j, sep = ''))

# need to recenter

sub\_tenx <- FindVariableFeatures(sub\_tenx, selection.method = "vst", nfeatures = 2000)

sub\_tenx <- FindNeighbors(sub\_tenx, dims = 1:20)

sub\_tenx <- RunUMAP(sub\_tenx, dims = 1:20, n.neighbors = 10, verbose = FALSE)

# get subgroups if there is a cluster, if not remove and label cells.

set\_res <- find\_res(sub\_tenx)

sub\_tenx <-FindClusters(sub\_tenx,pc.use = 1:20, resolution = set\_res)

# so subgroups, remove from queue: lib\_c

if (set\_res == 0 || length(levels(sub\_tenx@active.ident)) == 1){

lib\_c <- lib\_c[lib\_c != j]

}

# with subgroups

else {

# create the plots and matrix

gen\_matrix\_plot(sub\_tenx, output\_folder, j)

hold\_group\_names <- c()

# find subgroups from j:

for (k in sort(unique(sub\_tenx@active.ident))){

l <- paste(j,k, sep = '.')

# get column names to create

sub2\_tenx <- subset(sub\_tenx, idents = toString(k))

cellnames <- colnames(sub2\_tenx)

rownames(tenx@meta.data) %in% cellnames

# add to the metadata file

tenx@meta.data[rownames(tenx@meta.data) %in% cellnames,]$cellfindr = l

hold\_group\_names<- c(hold\_group\_names, l)

}

# add the new subgroups

lib\_c <- c(hold\_group\_names, lib\_c)

# remove original column

lib\_c <- lib\_c[lib\_c != j]

}

tenx <- SetIdent(tenx, value = 'cellfindr')

}

#resort order of labels

levels(tenx) <-str\_sort(levels(tenx), numeric = TRUE)

#graph umap

ggsave(paste(output\_folder, '/', proj\_name, '\_CellfindR\_umap.pdf',sep = ""),

DimPlot(tenx, label = TRUE), width = 10, height = 8)

return(tenx)

}

get\_analysis <- function(tenx, output\_folder = '.', proj\_name = 'proj\_name'){

# output DataQ files

# percent.mt plot

ggsave(paste(output\_folder, '/', proj\_name, '\_percent\_mito.pdf',sep = ""),

FeaturePlot(tenx, 'percent.mt'), width = 8, height = 8)

# percent.mt violin plot

ggsave(paste(output\_folder, '/', proj\_name, '\_percent\_mito\_vln.pdf',sep = ""),

VlnPlot(tenx, 'percent.mt'), width = length(levels(tenx@active.ident)), height = 5)

# umi plot

ggsave(paste(output\_folder, '/', proj\_name, '\_nCount\_RNA.pdf',sep = ""),

FeaturePlot(tenx, 'nCount\_RNA'), width = 8, height = 8)

# umi violin plot

ggsave(paste(output\_folder, '/', proj\_name, '\_nCount\_RNA\_vln.pdf',sep = ""),

VlnPlot(tenx, 'nCount\_RNA'), width = length(levels(tenx@active.ident)), height = 5)

#############

#output Matrices

tenx <-SetIdent(tenx, value = 'cellfindr')

levels(tenx) <-str\_sort(levels(tenx), numeric = TRUE)

z <- get\_matrix(tenx)

write.csv(z, file = paste(output\_folder, '/', 'matrix\_cellfindr.csv', sep = ''))

a <- get\_stats(tenx)

write.csv(a, file = paste(output\_folder, '/', 'all\_stats\_cellfindr.csv', sep = ''))

tenx <-SetIdent(tenx, value = 'seurat\_clusters')

y <- get\_matrix(tenx)

write.csv(y, file = paste(output\_folder, '/', 'matrix\_big\_groups.csv', sep = ''))

a <- get\_stats(tenx)

write.csv(a, file = paste(output\_folder, '/', 'all\_stats\_big\_groups.csv', sep = ''))

}

# generate matrix and plots

gen\_matrix\_plot <-function(tenx, output\_folder = '.', proj\_name = 'proj\_name'){

file\_create <-paste(output\_folder,'/', proj\_name,sep = '')

print(file\_create)

### Output data destinations

# create folder

ggsave(paste(file\_create, '\_umap.pdf', sep = ''), DimPlot(tenx, label = TRUE))

markers <- FindAllMarkers(tenx, only.pos = TRUE, min.pct = 0.25, thresh.use = 0.25, base = exp(1))

markers\_filtered <- markers %>% group\_by(cluster) %>% top\_n(n = 20, wt = avg\_logFC)

genes <- unique(markers\_filtered$gene)

matrix\_gen <- get\_matrix(tenx)

write.csv(matrix\_gen,paste(file\_create, '\_matrix.csv', sep = ''), row.names = TRUE)

#create subdirectories

file\_create2 <- paste(output\_folder, '/',proj\_name, sep ='')

dir.create(file\_create2)

dir.create(paste(file\_create2, 'Cluster', sep = '/'))

dir.create(paste(file\_create2, 'Violin', sep = '/'))

for (i in genes) {

# cluster maps

ggsave(paste(paste(file\_create, 'Cluster', '', sep = '/'),i,'.pdf',sep = ''),

FeaturePlot(tenx, features = c(i),pt.size = 2), width = 8, height = 8)

#violin plot

ggsave(paste(paste(file\_create, 'Violin', '', sep = '/'), i, '.pdf', sep = ''),

VlnPlot(tenx, c(i)),width = 6, height = 4)

}

}

# generate matrix

get\_matrix <- function(tenx){

print("getting matrix")

avg\_expression <- AverageExpression(tenx, use.scale = TRUE) #

matrix\_all <- data.frame(row.names = rownames(avg\_expression$RNA))

for (i in levels(tenx@active.ident)){

print(i)

markers <- FindMarkers(tenx, ident.1 = i, logfc.threshold = 0.1, base = exp(1))

avg\_val <- avg\_expression$RNA[i]

avg\_diff <- markers[rownames(avg\_expression$RNA),]$avg\_logFC

avg\_diff[is.na(avg\_diff)] <-0

p\_val <- markers[rownames(avg\_expression$RNA),]$p\_val\_adj

p\_val[is.na(p\_val)] <-1

matrix\_all <- cbind(matrix\_all, avg\_val)

matrix\_all <- cbind(matrix\_all, avg\_diff)

matrix\_all <- cbind(matrix\_all, p\_val)

}

name\_col <- c()

for (k in levels(tenx@active.ident)){

print(k)

name\_col <- c(name\_col,(c(paste(k,'Mean', sep = '\_'),paste(k,'Avg\_diff', sep = '\_') , paste(k,'Pval', sep = '\_'))))

}

colnames(matrix\_all) <- name\_col

matrix

return(matrix\_all)

}

# get stats

get\_stats <- function(tenx, num\_genes = 50){

aoe <- c("Group", "cell\_number", "avg\_read", "avg\_umi")

for (i in 1:num\_genes){

aoe <- c(aoe, paste('top\_',i, sep = ""))

}

df <- data.frame(aoe)

#initialize matrix

for (groups in levels(tenx@active.ident)){

subgroup <-subset(tenx, idents = groups)

# group name

aod <- c(groups)

# cell number

aod <- c(aod, length(subgroup@meta.data$nCount\_RNA))

# avg\_read

aod <- c(aod, mean(subgroup@meta.data$nCount\_RNA))

# avg\_umi

aod <- c(aod, mean(subgroup@meta.data$nFeature\_RNA))

# top 10 diff genes

markers <- FindMarkers(tenx, groups, base = exp(1))

top\_markers <- row.names(markers)[1:num\_genes]

for (topm in top\_markers){

aod <- c(aod, topm)

}

df[groups] <-aod

}

return(df)

}

# getting plots

get\_plots<- function(tenx, output\_folder = '.'){

dir\_creater <- paste(output\_folder, '/plots', sep = '')

dir.create(dir\_creater)

for (groups in levels(tenx@active.ident)){

markers <-FindMarkers(tenx, groups, base = exp(1))

maxer <- min(50, length(rownames(markers)))

for (gene in rownames(markers)[1:maxer]){

ggsave(paste(dir\_creater,'/', gene, '\_cluster.pdf', sep = ''),

FeaturePlot(tenx, features = gene), width = 6, height = 6)

ggsave(paste(dir\_creater,'/', gene, '\_violin.pdf', sep = ''),

VlnPlot(tenx, features = gene, slot = "counts", log = TRUE), width = length(levels(tenx@active.ident)), height = 5)

}

}

}

# output metrics

metrics\_output <- function(tenx, output\_folder = '.', species = 'mouse'){

tenx@active.assay

# mito percent

if (species == 'mouse'){

ggsave(paste(output\_folder, '/', 'percent\_mito.pdf', sep = ''),

VlnPlot(tenx, features = 'percent.mt'), width = length(levels(tenx@active.ident)), height = 5)

}

if (species == 'human'){

ggsave(paste(output\_folder, '/', 'percent\_mito.pdf', sep = ''),

VlnPlot(tenx, features = 'percent.MT'), width = length(levels(tenx@active.ident)), height = 5)

}

# umi

print('test')

ggsave(paste(output\_folder, '/', 'uMI.pdf', sep = ''),

VlnPlot(tenx, features = 'nFeature\_RNA'), width =length(levels(tenx@active.ident)), height = 5)

}

# get the top 100 genes from the values

get\_top100 <- function(a, s){

sorted <- a[order(a[s], decreasing = TRUE), ]

return(row.names(sorted)[1:100])

}

intersection\_top100 <- function(new\_expression\_table, old\_expression\_table){

mat\_cor <- 0

names <- (colnames(new\_expression\_table))

mat\_cor <-data.frame(row.names = names)

for (i in colnames(old\_expression\_table)){

print(paste('Subset', i))

#sort based on column one and extract list

tester <- get\_top100(old\_expression\_table, i)

list\_of\_i <- c()

for (j in colnames(new\_expression\_table)){

# new tables for correlation

o <-get\_top100(new\_expression\_table, j)

list\_of\_i <- c(list\_of\_i, length(intersect(tester,o)))

print(length(intersect(tester,o)))

}

df <- data.frame(j=list\_of\_i)

mat\_cor <- cbind(mat\_cor, df)

}

colnames(mat\_cor) <- colnames(old\_expression\_table)

return(mat\_cor)

}

##############################

#mouse human comparison

# get intersection of mouse to human

# gets the index of the reference in mouse or in human,

ortholog\_index <- function(list\_genes, id = "Mouse"){

table\_ref <- read.csv('/Users/kyu/Desktop/Project\_Cochlea/look\_up\_table/look\_up\_table.csv')

index\_list <- c()

if (id == "Mouse"){

for (gene in list\_genes){

index\_list <- c(index\_list, which(table\_ref$Mouse == gene)[1])

}

}

if (id =="Human"){

for (gene in list\_genes){

index\_list <- c(index\_list, which(table\_ref$Human == gene)[1])

}

}

return(index\_list)

}

# creates intersection based on the

intersection\_h\_m <- function(human\_matrix, mouse\_matrix){

# set up ending coorelation matrix

mat\_cor <- 0

names <- (colnames(mouse\_matrix))

mat\_cor <-data.frame(row.names = names)

# iterate through each subgroup in the human matrix

for (i in colnames(human\_matrix)){

print(paste('Subset', i))

#set up all the human genes and gets the index with respect to the hash table

human\_genes<- get\_top100(human\_matrix, i)

human\_index <- ortholog\_index(human\_genes, id= 'Human')

list\_of\_i <- c()

for (j in colnames(mouse\_matrix)){

#convert to human

mouse\_genes <- get\_top100(mouse\_matrix, j)

mouse\_index <- ortholog\_index(mouse\_genes)

# new tables for correlation

# could do it by homology index (returns might be easier)

#get intersect of the values:

intersect\_hm <- intersect(human\_index, mouse\_index)

val\_hm <- length(intersect\_hm)

# if has a NA remove 1

if (is.element(NA, intersect\_hm)){

val\_hm <- val\_hm -1

}

list\_of\_i <- c(list\_of\_i, val\_hm)

print(paste('Intersect', val\_hm))

}

df <- data.frame(j=list\_of\_i)

mat\_cor <- cbind(mat\_cor, df)

}

colnames(mat\_cor) <- colnames(human\_matrix)

return(mat\_cor)

}

# get deafness gene plots need to reupdate

get\_deafness\_plot <- function(tenx, id = 'Mouse', name = 'deafness'){

if (id == "Mouse"){

df\_deafness <- read.csv("/Users/kyu/Desktop/Project\_Cochlea/look\_up\_table/Deafness\_gene\_list.csv", header = FALSE)

}

else{

df\_deafness <- read.csv("/Users/kyu/Desktop/Project\_Cochlea/look\_up\_table/Deafness\_gene\_list\_human.csv", header = FALSE)

}

Genes <- 0

# loop through all the different subclusters

# initialize data.frame

names <- rownames(tenx@scale.data)

Genes <-data.frame(row.names = names)

for (j in levels(tenx@ident)){

# subset each one and then make each column that way.

test <- SubsetData(tenx, ident.use = j)

mat\_simp <- test@scale.data # scaled data from the values

mat\_simp <-as.matrix(mat\_simp)

row\_means <- rowMeans(mat\_simp, na.rm=TRUE)

df <- data.frame( j=row\_means)

colnames(df) <-j

Genes <- cbind(Genes, df)

print(j)

}

# remove genes column

# set upper limit as 1

Genes[Genes <0 ] <- 0

Genes[Genes >1] <- 1

# write to heatmap (find the file for the deafness genes)

list\_deafness <- c()

#get only the unique ones

genenames <- row.names(tenx@scale.data)

for (gene in df\_deafness[,1]){

if (any(genenames == gene)){

list\_deafness <- c(list\_deafness, gene)

}

}

list\_deafness <- unique(list\_deafness)

###

# get rid of it

heatmap\_test <- Genes[list\_deafness,]

heatmap\_test <- heatmap\_test[1:length(heatmap\_test)]

pdf(paste('./', name,'.pdf', sep = ''), width = 12, height = 20)

pheatmap(heatmap\_test, color = colorRampPalette(rev(brewer.pal(n = 7, name = "RdYlBu")))(100))

dev.off()

return(heatmap\_test)

}

# get timecourse

get\_timecourse <- function(tenx){

aoe <-c('Group', unique(tenx@meta.data$orig.ident))

df <- data.frame(aoe)

for (groups in levels(tenx@active.ident)){

subgroup <-subset(tenx, idents = groups)

# group name

aod <- c(groups)

subgroup <-subset(tenx, idents = groups)

aod <- c(groups)

for (val in unique(tenx@meta.data$orig.ident)){

aod <- c(aod, sum(subgroup@meta.data == val))

}

df[groups] <-aod

}

return(df)

}