## ##CITE-Seq 1 Script####

####Setup####

#Load required packages

library(Seurat)

library(ggplot2)

library(tidyverse)

library(patchwork)

library(Matrix)

library(RColorBrewer)

library(writexl)

library(ggridges)

library(clustree)

library(scRepertoire)

library(future)

library(alakazam)

library(immunarch)

library(airr)

#Alter working capacity

plan()

plan("multiprocess", workers = 4)

options(future.globals.maxSize= 10097152000) # 10Gb

#Initial setup of colour palettes

col = colorRampPalette(brewer.pal(12, 'Set3'))(20)

colbig = colorRampPalette(brewer.pal(12, 'Set3'))(50)

####Load the 10X Cell Ranger output####

#Read the 10x Cell Ranger Output

c1\_ge.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Sample\_GE\_out/C1\_GE/outs/filtered\_feature\_bc\_matrix")

c2\_ge.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Sample\_GE\_out/C2\_GE/outs/filtered\_feature\_bc\_matrix")

d1\_ge.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Sample\_GE\_out/D1\_GE/outs/filtered\_feature\_bc\_matrix")

d2\_ge.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Sample\_GE\_out/D2\_GE/outs/filtered\_feature\_bc\_matrix")

a\_ge.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_2\_files/GE/A\_WT\_GE/outs/filtered\_feature\_bc\_matrix")

b\_ge.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_2\_files/GE/B\_WT\_GE/outs/filtered\_feature\_bc\_matrix")

#Add the sample to the cell names, consistent with antibody data below

colnames(c1\_ge.data)=gsub("-1","\_c1",colnames(c1\_ge.data))

colnames(c2\_ge.data)=gsub("-1","\_c2",colnames(c2\_ge.data))

colnames(d1\_ge.data)=gsub("-1","\_d1",colnames(d1\_ge.data))

colnames(d2\_ge.data)=gsub("-1","\_d2",colnames(d2\_ge.data))

colnames(a\_ge.data)=gsub("-1","\_a",colnames(a\_ge.data))

colnames(b\_ge.data)=gsub("-1","\_b",colnames(b\_ge.data))

#Uppercase the gene names for easier matching later

rownames(c1\_ge.data)=toupper(rownames(c1\_ge.data))

rownames(c2\_ge.data)=toupper(rownames(c2\_ge.data))

rownames(d1\_ge.data)=toupper(rownames(d1\_ge.data))

rownames(d2\_ge.data)=toupper(rownames(d2\_ge.data))

rownames(a\_ge.data)=toupper(rownames(a\_ge.data))

rownames(b\_ge.data)=toupper(rownames(b\_ge.data))

head(b\_ge.data)

####Load 10X Antibody data####

#Read the 10x Antibody output

c1\_ab.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Antibody\_fraction/C1\_SP\_out\_2/umi\_count",gene.column=1)

c2\_ab.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Antibody\_fraction/C2\_SP\_out\_2/umi\_count",gene.column=1)

d1\_ab.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Antibody\_fraction/D1\_SP\_out\_2/umi\_count",gene.column=1)

d2\_ab.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/Antibody\_fraction/D2\_SP\_out\_2/umi\_count",gene.column=1)

a\_ab.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_2\_files/second\_batch\_data\_CP/A\_WT/umi\_count",gene.column=1)

b\_ab.data <- Read10X(data.dir = "~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_2\_files/second\_batch\_data\_CP/B\_WT/umi\_count",gene.column=1)

#Tidy up the rownames from the data

rownames(c1\_ab.data)=gsub("-[^-]+$","",rownames(c1\_ab.data),perl=TRUE)

rownames(c2\_ab.data)=gsub("-[^-]+$","",rownames(c2\_ab.data),perl=TRUE)

rownames(d1\_ab.data)=gsub("-[^-]+$","",rownames(d1\_ab.data),perl=TRUE)

rownames(d2\_ab.data)=gsub("-[^-]+$","",rownames(d2\_ab.data),perl=TRUE)

rownames(a\_ab.data)=gsub("-[^-]+$","",rownames(a\_ab.data),perl=TRUE)

rownames(b\_ab.data)=gsub("-[^-]+$","",rownames(b\_ab.data),perl=TRUE)

#Add the Sample to the cell names in each sample

colnames(c1\_ab.data)=paste(colnames(c1\_ab.data),"\_c1",sep="")

colnames(c2\_ab.data)=paste(colnames(c2\_ab.data),"\_c2",sep="")

colnames(d1\_ab.data)=paste(colnames(d1\_ab.data),"\_d1",sep="")

colnames(d2\_ab.data)=paste(colnames(d2\_ab.data),"\_d2",sep="")

colnames(a\_ab.data)=paste(colnames(a\_ab.data),"\_a",sep="")

colnames(b\_ab.data)=paste(colnames(b\_ab.data),"\_b",sep="")

head(b\_ab.data)

####Combine 10X Cell Ranger and Antibody Data into a Suerat Object####

m <- Matrix(nrow = nrow(c1\_ab.data), ncol = ncol(c1\_ge.data), data = 0, sparse = TRUE)

rownames(m)=rownames(c1\_ab.data)

colnames(m)=colnames(c1\_ge.data)

common=intersect(colnames(c1\_ge.data),colnames(c1\_ab.data))

m[,common]=c1\_ab.data[,common]

c1 = CreateSeuratObject(counts = c1\_ge.data,project="c1", min.cells = 3)

adt\_assay <- CreateAssayObject(counts = m)

c1[["ADT"]] <- adt\_assay

m <- Matrix(nrow = nrow(c2\_ab.data), ncol = ncol(c2\_ge.data), data = 0, sparse = TRUE)

rownames(m)=rownames(c2\_ab.data)

colnames(m)=colnames(c2\_ge.data)

common=intersect(colnames(c2\_ge.data),colnames(c2\_ab.data))

m[,common]=c2\_ab.data[,common]

c2 = CreateSeuratObject(counts = c2\_ge.data,project="c2", min.cells = 3)

adt\_assay <- CreateAssayObject(counts = m)

c2[["ADT"]] <- adt\_assay

m <- Matrix(nrow = nrow(d1\_ab.data), ncol = ncol(d1\_ge.data), data = 0, sparse = TRUE)

rownames(m)=rownames(d1\_ab.data)

colnames(m)=colnames(d1\_ge.data)

common=intersect(colnames(d1\_ge.data),colnames(d1\_ab.data))

m[,common]=d1\_ab.data[,common]

d1 = CreateSeuratObject(counts = d1\_ge.data,project="d1", min.cells = 3)

adt\_assay <- CreateAssayObject(counts = m)

d1[["ADT"]] <- adt\_assay

m <- Matrix(nrow = nrow(d2\_ab.data), ncol = ncol(d2\_ge.data), data = 0, sparse = TRUE)

rownames(m)=rownames(d2\_ab.data)

colnames(m)=colnames(d2\_ge.data)

common=intersect(colnames(d2\_ge.data),colnames(d2\_ab.data))

m[,common]=d2\_ab.data[,common]

d2 = CreateSeuratObject(counts = d2\_ge.data,project="d2", min.cells = 3)

adt\_assay <- CreateAssayObject(counts = m)

d2[["ADT"]] <- adt\_assay

m <- Matrix(nrow = nrow(a\_ab.data), ncol = ncol(a\_ge.data), data = 0, sparse = TRUE)

rownames(m)=rownames(a\_ab.data)

colnames(m)=colnames(a\_ge.data)

common=intersect(colnames(a\_ge.data),colnames(a\_ab.data))

m[,common]=a\_ab.data[,common]

a = CreateSeuratObject(counts = a\_ge.data,project="a", min.cells = 3)

adt\_assay <- CreateAssayObject(counts = m)

a[["ADT"]] <- adt\_assay

m <- Matrix(nrow = nrow(b\_ab.data), ncol = ncol(b\_ge.data), data = 0, sparse = TRUE)

rownames(m)=rownames(b\_ab.data)

colnames(m)=colnames(b\_ge.data)

common=intersect(colnames(b\_ge.data),colnames(b\_ab.data))

m[,common]=b\_ab.data[,common]

b = CreateSeuratObject(counts = b\_ge.data,project="b", min.cells = 3)

adt\_assay <- CreateAssayObject(counts = m)

b[["ADT"]] <- adt\_assay

head(b[[]])

####Incoperate VDJ data####

#Load contig file

c1\_cl.data <- read.csv("~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/VDJ/C1\_VDJ/outs/filtered\_contig\_annotations.csv")

c2\_cl.data <- read.csv("~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/VDJ/C2\_VDJ/outs/filtered\_contig\_annotations.csv")

d1\_cl.data <- read.csv("~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/VDJ/D1\_VDJ/outs/filtered\_contig\_annotations.csv")

d2\_cl.data <- read.csv("~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_1\_files/VDJ/D2\_VDJ/outs/filtered\_contig\_annotations.csv")

a\_cl.data <- read.csv("~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_2\_files/VDJ\_batch2/A\_WT\_VDJ/outs/filtered\_contig\_annotations.csv")

b\_cl.data <- read.csv("~/Desktop/CITE-Sequencing\_Data/CITE\_Seq\_2\_files/VDJ\_batch2/B\_WT\_VDJ/outs/filtered\_contig\_annotations.csv")

#match barcode names with GE and ADT data

c1\_cl.data$barcode=gsub("-1","\_c1",c1\_cl.data$barcode)

c2\_cl.data$barcode=gsub("-1","\_c2",c2\_cl.data$barcode)

d1\_cl.data$barcode=gsub("-1","\_d1",d1\_cl.data$barcode)

d2\_cl.data$barcode=gsub("-1","\_d2",d2\_cl.data$barcode)

a\_cl.data$barcode=gsub("-1","\_a",a\_cl.data$barcode)

b\_cl.data$barcode=gsub("-1","\_b",b\_cl.data$barcode)

contig\_list <- list(c1\_cl.data, c2\_cl.data, d1\_cl.data, d2\_cl.data, a\_cl.data, b\_cl.data)

head(contig\_list[[1]])

#Generate combined object

combined <- combineBCR(contig\_list, samples = c("c1", "c2", "d1", "d2", "a", "b"))

combined[[1]]

str(combined)

head(combined[[1]])

#Make sure barcodes are identical to GE and ADT data

combined$c1$barcode=gsub("c1\_","",combined$c1$barcode)

combined$c2$barcode=gsub("c2\_","",combined$c2$barcode)

combined$d1$barcode=gsub("d1\_","",combined$d1$barcode)

combined$d2$barcode=gsub("d2\_","",combined$d2$barcode)

combined$a$barcode=gsub("a\_","",combined$a$barcode)

combined$b$barcode=gsub("b\_","",combined$b$barcode)

head(combined$c1)

#####Process samples as one####

experiments=c(c1,c2,d1,d2,a,b)

experiment\_names=c("c1","c2","d1","d2","a","b")

experiment<-merge(x= c1, y=c(c2,d1,d2,a,b))

experiment

str(experiment)

head(experiment[[]])

####Merge seurat object with VDJ data####

experiment <- combineExpression(combined,

experiment,

cloneCall="gene", group.by = "sample")

head(experiment[[]])

####Quality control, filtering, normalisation and scaling####

#Mitochondrial QC metrics

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

#Remove where nCount\_ADT = 0

DefaultAssay(experiment) <- "ADT"

experiment <- subset(experiment, nCount\_ADT >0)

DefaultAssay(experiment) <- "RNA"

#Visualize QC metrics as violin plot

RNA\_QC <- VlnPlot(experiment, features = c("nFeature\_RNA", "nCount\_RNA", "percent.mt"))

ADT\_QC <- VlnPlot(experiment, features = "nFeature\_ADT")

ADT\_QC <- VlnPlot(experiment, features = "nCount\_ADT", y.max = 10000)

RNA\_QC

ADT\_QC

#FeatureScatter is typically used to visualize feature-feature relationships, but can be used for anything calculated by the object, i.e. columns in object metadata, PC scores etc.

feature\_count.RNA\_vs\_percent.mt = FeatureScatter(experiment, feature1 = "nCount\_RNA", feature2 = "percent.mt") + NoLegend() +

ylab("% of mitochondrial genes") +

xlab("UMI counts") +

geom\_hline(yintercept = 5)

feature\_count.RNA\_vs\_feature.RNA = FeatureScatter(experiment, feature1 = "nCount\_RNA", feature2 = "nFeature\_RNA") + NoLegend() +

ylab("Number of genes") +

xlab("UMI counts") +

geom\_hline(yintercept = 200)

feature\_count.RNA\_vs\_percent.mt + feature\_count.RNA\_vs\_feature.RNA

#Generally aim to filter out unique feature counts over 2,500 and less than 200; and percent.mt over 5%

filter\_seurat = function(seurat\_object){

message("Performing filter by number of genes and mitochondrial percentage.")

seurat\_object = subset(seurat\_object, subset = nFeature\_RNA > 200 & percent.mt < 5 & nFeature\_RNA < 2500)

message("Now the object has ", dim(seurat\_object)[1], " genes and ", dim(seurat\_object)[2], " cells.")

return(seurat\_object)

}

experiment <- filter\_seurat(experiment)

####Normalise dataset - SCTransform####

experiment = SCTransform(experiment, verbose = TRUE)

experiment[["SCT"]]

top20 = head(VariableFeatures(experiment), 20)

plot1.1 = VariableFeaturePlot(experiment)

top20\_plot = LabelPoints(plot = plot1.1, points = top20, repel = TRUE, xnudge = 0, ynudge = 0)

top20\_plot

#Cell Cycle genes

S.genes = cc.genes.updated.2019$s.genes

G2M.genes = cc.genes.updated.2019$g2m.genes

experiment = CellCycleScoring(experiment, s.features=S.genes, g2m.features=G2M.genes, set.ident = TRUE)

Idents(object = experiment) <- "old.ident"

####Dimensionality reduction - PCA####

#Perform linear dimensional reduction (PCA)

experiment <- RunPCA(experiment, verbose = FALSE, features = VariableFeatures(object = experiment))

##Visualise PCA results

print(experiment[["pca"]], dims = 1:5, nfeatures = 5)

VizDimLoadings(experiment, dims = 1:4, reduction = "pca", nfeatures = 15)

plot5 <- DimPlot(experiment, reduction = "pca", dims = c(1,2))

plot6 <- DimPlot(experiment, reduction = "pca", dims = c(1,3))

plot5 + plot6

DimHeatmap(experiment, dims = 1:6, cells = 500, balanced = TRUE)

####Determine dimensionality of the dataset - How many principal components should be included to capture the majority of variance?####

pca\_variance <- experiment@reductions$pca@stdev^2

plot(pca\_variance/sum(pca\_variance),

ylab="Proportion of variance explained",

xlab="Principal component")

abline(h = 0.01) #23

#Cluster the cells

experiment <- FindNeighbors(experiment, dims = 1:23)

experiment <- FindClusters(experiment, resolution = 1.3, verbose = FALSE) #1.3 or 1.2 for the resolution

#Cluster Tree Analysis

clustree(experiment, prefix = "SCT\_snn\_res.") + theme(legend.position="bottom")

#RNA UMAP

experiment <- RunUMAP(experiment, dims = 1:23)

DimPlot(experiment, label = TRUE, cols=colbig) + ggtitle("RNA Clustering")

####Scale antibody data####

DefaultAssay(experiment) <- "ADT"

VariableFeatures(experiment) <- rownames(experiment[["ADT"]])

experiment <- NormalizeData(experiment, normalization.method = "CLR", margin = 2)

experiment <- ScaleData(experiment)

experiment <- RunPCA(experiment,reduction.name = 'apca')

#Visualise antibody PCA

print(experiment[["apca"]], dims = 1:10, nfeatures = 5)

Plot\_13 <- VizDimLoadings(experiment, dims = 1:4, reduction = "apca", nfeatures = 15)

Plot\_13

plot14 <- DimPlot(experiment, reduction = "apca", dims = c(1,2), group.by = "orig.ident") + ggtitle("ADT PCA")

plot15 <- DimPlot(experiment, reduction = "apca", dims = c(1,3), group.by = "orig.ident") + ggtitle("ADT PCA")

plot14 + plot15

DimHeatmap(experiment, dims = 1:6, cells = 500, balanced = TRUE, reduction = "apca")

#Determine number of PCs for ADT assay

apca\_variance <- experiment@reductions$apca@stdev^2

plot(apca\_variance/sum(apca\_variance),

ylab="Proportion of variance explained",

xlab="Principal component")

abline(h = 0.01) #24

#Number of clusters for UMAP?

####Combine into wnn plot####

experiment <- FindMultiModalNeighbors(

experiment, reduction.list = list("pca", "apca"),

dims.list = list(1:23, 1:24), modality.weight.name = "RNA.weight")

#UMAP plots for RNA, ADT and WNN

experiment <- RunUMAP(experiment, reduction = 'pca', dims = 1:23, assay = 'RNA',

reduction.name = 'rna.umap', reduction.key = 'rnaUMAP\_')

experiment<- RunUMAP(experiment, reduction = 'apca', dims = 1:24, assay = 'ADT',

reduction.name = 'adt.umap', reduction.key = 'adtUMAP\_')

experiment <- RunUMAP(experiment, nn.name = "weighted.nn", reduction.name = "wnn.umap", reduction.key = "wnnUMAP\_")

experiment <- FindClusters(experiment, graph.name = "wsnn", algorithm = 3, resolution = 1.7, verbose = TRUE)

#Cluster Tree Analysis of wsnn graph

clustree(experiment, prefix = "wsnn\_res.") + theme(legend.position="bottom")#1.7

DefaultAssay(experiment) <- "RNA"

p1=DimPlot(experiment, label = TRUE,cols=colbig,reduction = "rna.umap", label.size = 2.5) + NoLegend()

p2=DimPlot(experiment, label = TRUE,cols=colbig,reduction = "adt.umap", label.size = 2.5) + NoLegend()

p3=DimPlot(experiment, label = TRUE,cols=colbig, reduction = "wnn.umap", label.size = 2.5) + NoLegend()

p1

p2

p3

head(experiment[[]])

###Umap-wnn by mouse

plot\_mouse <- DimPlot(experiment, label = TRUE,reduction = "wnn.umap", label.size = 2.5, group.by = "orig.ident") + ggtitle("Coloured by mouse")

plot\_mouse

###Umap-wnn by sample

DimPlot(experiment, label = TRUE,cols=colbig, reduction = "wnn.umap", label.size = 2.5, split.by = "orig.ident", ncol = 2) + NoLegend()

###Umap-wnn by cell cycle stage

DimPlot(experiment, label = TRUE,reduction = "wnn.umap", label.size = 2.5, group.by = "Phase") + ggtitle("Coloured by cell cycle stage")

head(experiment[[]])

###Match the RNA Names to the Antibodies, this should be checked

list1=c(rownames(a\_ab.data))

list2=c("PTPRC","FAS","CD19","IGHM","CR2","FCER2A","CD93","CD83","CD86","IGHD","CD8A","SELL","CD44","CD4","CXCR5","PDCD1","IL2RA","CD274","PDCD1LG2","CTLA4","CD80","CD40","CD69","ICOS","CD38","TNFRSF18")

####Analysis of clusters####

##Different plotting options

DefaultAssay(experiment) <- "RNA"

DefaultAssay(experiment) <- "ADT"

FeaturePlot(experiment, features = c("CD19", "CD4", "CD8A", "PRDM1", "PPBP", "NKG7", "CST3", "FOXP3", "B220"), reduction = "wnn.umap")

RidgePlot(experiment, features = c("CD19", "CYP11A1"), ncol = 2)

FeaturePlot(experiment, features = c("IGHV1-53", "IGKV3-4", "IGHD1-1"), reduction = "wnn.umap")

FeaturePlot(experiment, feature = "IGHG", reduction = "wnn.umap")

FeaturePlot(experiment, features = c("CD8A"), reduction = "wnn.umap")

VlnPlot(experiment, feature = "CYP11A1")

?VlnPlot

p3

Idents(object = experiment) <- "old.ident"

head(experiment[[]])

##Finding all the markers

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

experiment.markers %>%

group\_by(cluster) %>%

top\_n(n = 10, wt = avg\_log2FC) -> top10

DoHeatmap(experiment, features = top10$gene) + NoLegend()

##DE genes of individual clusters

Cluster\_19 <- FindMarkers(experiment, ident.1 = 19, assay = "RNA")

Cluster\_19\_adt <- FindMarkers(experiment, ident.1 = 19, assay = "ADT")

Cluster\_4 <- FindMarkers(experiment, ident.1 = 4, assay = "RNA")

Cluster\_4\_adt <- FindMarkers(experiment, ident.1 = 4, assay = "ADT")

p3

##Subsetting unknown cluster

Unknown\_cells <- subset(experiment, idents = c(2, 12, 27, 33))

Unknown\_cells <- FindClusters(Unknown\_cells, resolution = 0.8, verbose = FALSE, graph.name = "wsnn")

Unknown\_cells <- RunUMAP(Unknown\_cells, dims = 1:30, reduction.name = "unknown.umap")

DimPlot(Unknown\_cells, label = TRUE, cols=colbig, reduction = "unknown.umap", label.size = 2.5) + NoLegend()

FeaturePlot(Unknown\_cells, "KLS", reduction = "unknown.umap")

####Clonotype analysis####

##Data visualization

#Percent/total number of unique clonotypes

quantContig(combined, cloneCall = "gene+nt", scale = T) #percent of unique clonotypes of total size of the size of clonotyeps

quantContig(combined, cloneCall = "gene+nt", scale = F) #number of uniqe clonotypes

quantContig(combined, cloneCall = "gene+nt", scale = T, chain = "IGL")#by chain

#Abundance of clonotypes

Abundance\_clonotypes <- abundanceContig(combined, cloneCall = "gene", scale = F, exportTable = T)

Abundance\_clonotypes <- Abundance\_clonotypes %>%

arrange(desc(Abundance))

Abundance\_clonotypes

abundanceContig(combined, cloneCall = "gene", scale = T)

#Length of clonotypes

lengthContig(combined, cloneCall = "aa")

lengthContig(combined, cloneCall = "nt")

#Compare clonotypes

compareClonotypes(combined, samples = c("a", "b"), cloneCall = "aa", graph = "alluvial") #Computationally intese

#Visualise Gene Usage

vizGenes(combined, gene = "D", chain = "IGH", plot = "bar", order = "variance", scale = TRUE)

vizGenes(combined, gene = "V", chain = "IGL", plot = "bar", order = "variance", scale = TRUE)

vizGenes(combined, gene = "V", chain = "IGL", plot = "heatmap", scale = TRUE, order = "gene")

#Clonal overlap

clonalOverlap(combined, cloneCall = "gene+nt",

method = "morisita")

#Clonotype proportion

clonalProportion(combined, cloneCall = "gene")

clonalProportion(combined, cloneCall = "nt")

#Clonal Homeostasis

clonalHomeostasis(combined, cloneCall = "gene")

clonalHomeostasis(combined, cloneCall = "nt")