**Process single-cell CAT sequencing data**

Timing：xx h

The processing of single-cell CAT sequencing data is based on DNBelab\_C\_Series\_HT\_scCAT-analysis-software (github xx), and the detail procedures are listed below.

1. Process single-cell ATAC sequencing data
   1. Alignment

Mapping the Fastq format data to the reference genome using the Chromap.

## set output dirctory

outdir=/path/ourdir/

mkdir -p ${outdir}

mkdir -p ${outdir}/RNA/01.cDNAAnno

mkdir -p ${outdir}/RNA/02.M280UMI\_stat

mkdir -p ${outdir}/RNA/03.Matrix

mkdir -p ${outdir}/ATAC

mkdir -p ${outdir}/ATAC/02.d2cfile

mkdir -p ${outdir}/ATAC/01.out

mkdir -p ${outdir}/ATAC/01.out/Peak

mkdir -p ${outdir}/ATAC/01.out/Promoter

mkdir -p ${outdir}/Joint

mkdir -p ${outdir}/Joint/plot

mkdir -p ${outdir}/Joint/report

mkdir -p ${outdir}/Joint/report/div

mkdir -p ${outdir}/Joint/report/base64

mkdir -p ${outdir}/Joint/report/table

mkdir -p ${outdir}/Joint/report/ATAC

mkdir -p ${outdir}/Joint/report/RNA

## Alignment for ATAC-seq

export PATH=/path/gcc-9.1.0/bin:$PATH

export LD\_LIBRARY\_PATH="/path/lib/gcc-9.1.0/lib/gcc/x86\_64-pc-linux-gnu/9.1.0/:$LD\_LIBRARY\_PATH"

export LD\_LIBRARY\_PATH="/path/lib/gcc-9.1.0/lib:/path/lib/gcc-9.1.0/lib64:$LD\_LIBRARY\_PATH"

root="/path/software"

ref\_index="/path/database/genome.index"

ref="/path/database/genome.fa"

whitelist="/path/database/whitelist.tsv"

${root}/bin/chromap --preset atac --bc-error-threshold 0 --trim-adapters -x ${ref\_index} -r ${ref} -1 ${fastq1} -2 ${fastq2} -o ${outdir}/ATAC/01.out/aln.sam --barcode ${fastq1} --barcode-whitelist ${whitelist} --read-format bc:0:19,r1:20:-1 -t 10 --SAM 2> ${outdir}/ATAC/01.out/alignment\_report.tsv

${root}/bin/samtools view -S -b ${outdir}/ATAC/01.out/aln.sam > ${outdir}/ATAC/01.out/aln.bam

${root}/bin/samtools index ${outdir}/ATAC/01.out/aln.bam

rm ${outdir}/ATAC/01.out/aln.sam

* 1. Beads calling and beads merge

To get more cells, the beads usually upload in excess, that is why the debris or cell-free DNA can also be encapsulated into the droplets creating unwanted signals and lead to more than one beads captured by the same droplet. To avoid those beads with debris or cell-free misleading the biological conclusion, we use d2c to isolate real cells from debris. To get a single cell chromatin accessibility profile, the subcellular profiles should be merged based on the chromatin accessibility similarity of beads using d2c (https://github.com/STOmics/d2c).

export LD\_LIBRARY\_PATH="/path/gcclib/lib:/path/gcclib/lib64:$LD\_LIBRARY\_PATH" && export PATH="/path/Python-3/bin:$PATH"

root="/path/software"

alnbed="${outdir}/ATAC/01.out/aln.sam"

chrmt="chrM"

ID="sampleName"

${root}/bin/d2c\_v1.4.7/bin/d2c merge -i ${alnbed} --mapq 30 --bf 0 --fb 20000 -o ${outdir}/ATAC/02.d2cfile -c 10 -n ${ID} --mc ${chrmt} -r hg38 --sat --bt1 CB

* 1. Peak calling

Creating pseudo bulk replicate for all the cells.

root="/path/software"

ID="sampleName"

outdir="/path/ourdir/"

${root}/bin/macs2 callpeak -t ${outdir}/02.d2cfile/${ID}\_fragment.tsv.gz -f BED -g hs -n ${ID} -B -q 0.001 --nomodel --outdir ${outdir}/ATAC/01.out

1. Process single-cell RNA sequencing data
   1. Reads processing and alignment
      1. To get high quality reads, those reads would be removed listed below. Then parse the cell barcode to the read name.
2. read1 with N, read2 with N.
3. read2 has an average mass value below 4.
4. barcode failing to match whitelist (greater than two mismatches).
5. adapter removal (length of reads after filtering adapter < 20bp).
   * 1. Alignment

Mapping the processed reads to the reference genome.

|  |
| --- |
| export LD\_LIBRARY\_PATH=/path/software/lib/:$LD\_LIBRARY\_PATH  export PATH=/path/lib/gcc-9.1.0/bin:$PATH  export LD\_LIBRARY\_PATH="/path/lib/gcc-9.1.0/lib:/path/lib/gcc-9.1.0/lib64:$LD\_LIBRARY\_PATH"  echo "${sep='\n' fastq1}" > ${outdir}/RNA/01.cDNAAnno/in1.list  echo "${sep='\n' fastq2}" > ${outdir}/RNA/01.cDNAAnno/in2.list  bcPara=${outdir}/RNA/01.cDNAAnno/bc.Para  echo "in1=${outdir}/RNA/01.cDNAAnno/in1.list" > $bcPara  echo "in2=${outdir}/RNA/01.cDNAAnno/in2.list" >> $bcPara  echo "config=${barcode}" >> $bcPara  echo "cbdis=${outdir}/RNA/01.cDNAAnno/${barcode\_counts\_raw}" >> $bcPara  echo "report=${outdir}/RNA/01.cDNAAnno/${report}" >> $bcPara  echo "adapter=/jdfssz1/ST\_SUPERCELLS/PUB/scRNA/pipeline/v3.1.5/common/file/adapter.txt" >> $bcPara  ${root}/bin/scStar --outSAMattributes singleCell --outSAMtype BAM Unsorted --genomeDir ${refdir} --out FileNamePrefix ${outdir}/RNA/01.cDNAAnno/ --stParaFile $bcPara --outSAMmode NoQS --runThreadN 10 --limitOutSJcollapsed 10000000 --limitIObufferSize 350000000  ${root}/bin/Anno -I ${outdir}/RNA/01.cDNAAnno/Aligned.out.bam -a ${gtf} -L ${outdir}/RNA/01.cDNAAnno/${barcode\_counts\_raw} -o ${outdir}/RNA/01.cDNAAnno -c 10 -m chrM -B ${barcode} –-intron --anno 1 |

* 1. Beads selected and beads merge

Using the merged beads in scATAC-seq data.

1. integrative analysis of multiple modalities

get cells with RNA-seq and ATAC-seq data and integrated analysis with multiple modalities (ATAC and RNA)

|  |
| --- |
| library(harmony)  library(Signac)  library(Seurat)  library(GenomeInfoDb)  library(ggplot2)  library(patchwork)  library(data.table)  library(GenomicRanges)  library(stringr)  library(scMCA)  library(scHCL)  library(rhdf5)  library(svglite)  library(tidyr)  library(SoupX)  library(DropletUtils)  library(dplyr)  library(cowplot)  library(DoubletFinder)  library(RColorBrewer)  #library(hdf5r)  set.seed(1234)  ### Get the parameters  parser = argparse::ArgumentParser(description="merging wnn, QC and BC function")  parser$add\_argument('-R','--rna', help='input rna FilterMatrix file')  parser$add\_argument('-A','--atac', help='input atac Peak file')  parser$add\_argument('-SN','--samplename', help='sample name default:None')  parser$add\_argument('-F','--frag', help='input \*fragments.tsv.gz')  parser$add\_argument('-SP','--species', help='hg38 or mm10')  parser$add\_argument('-RM','--RawMatrix', help='input rna RawMatrix file')  parser$add\_argument('-BT','--BarcodeTran', help='input barcodeTranslate\_16.txt file')  parser$add\_argument('-D','--dim',help='dim usage')  parser$add\_argument('-MP','--mtgenepercentage',help='filter cells with mtgenes percentage')  parser$add\_argument('-MF','--minfeatures',help='filter cells with minimum nfeatures')  parser$add\_argument('-PC','--pc',help='pc usage')  parser$add\_argument('-RES','--res',help='resolution usage')  parser$add\_argument('-K','--knn',help='defines k for the k-nearest neighbor algorithm')  parser$add\_argument('-MD','--maxdim',help='max dimension to keep from UMAP procedure')  parser$add\_argument('-O','--outdir', help='outputdir default:None')  parser$add\_argument('-IS','--ifSoupx', help='whether use SoupxMatrix to analysis')  parser$add\_argument('-MT','--chrmt', help='type of mt')  parser$add\_argument('-GTF',"--gtf", help='gtf file')  args = parser$parse\_args()  gtf <- "/path/genes.gtf"  ###########################  # overlap and save h5file #  ###########################  # rna  rna\_counts <- Read10X(args$rna,gene.column = 1)  # atac  mtx\_path <- paste(args$atac, "/matrix.mtx", sep = '')  feature\_path <- paste(args$atac, "/peak.bed", sep = '')  barcode\_path <- paste(args$atac, "/barcodes.tsv", sep = '')  features <- readr::read\_tsv(feature\_path, col\_names = F) %>% tidyr::unite(feature)  barcodes <- readr::read\_tsv(barcode\_path, col\_names = F) %>% tidyr::unite(barcode)  atac\_counts <- Matrix::readMM(mtx\_path) %>%    magrittr::set\_rownames(features$feature) %>%    magrittr::set\_colnames(barcodes$barcode)  # intersect  cells<- intersect(colnames(atac\_counts),colnames(rna\_counts))  rna\_counts\_cells <- rna\_counts[,which(colnames(rna\_counts) %in% cells)]  atac\_counts\_cells <- atac\_counts[,which(colnames(atac\_counts) %in% cells)]  # save.h5  saveh5file <- function(rna\_counts\_cells,atac\_counts\_cells){    cell\_name <- rna\_counts\_cells@Dimnames[[2]]    multi.data <- rbind(rna\_counts\_cells,atac\_counts\_cells)    h5createFile(paste(args$outdir,"/overlap\_matrix.h5",sep = ''))    # Saving matrix information.    h5createGroup(paste(args$outdir,"/overlap\_matrix.h5",sep = ''),"matrix")    h5write(multi.data@Dimnames[[2]] , paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/barcodes")    h5write(multi.data@x, paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/data")    h5createGroup(paste(args$outdir,"/overlap\_matrix.h5",sep = ''),"matrix/features")    key <- c('genome','interval')    h5write(key, paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/features/\_all\_tag\_keys")    Genes <- rep('Gene Expression', length(rna\_counts\_cells@Dimnames[[1]]))    Peaks <- rep("Peaks", length(atac\_counts\_cells@Dimnames[[1]]))    Features <- c(Genes,Peaks)    h5write(Features,paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/features/feature\_type")    Genome <- rep(args$species, length(multi.data@Dimnames[[1]]))    h5write(Genome,paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/features/genome")    h5write(multi.data@Dimnames[[1]],paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/features/id")    h5write(multi.data@Dimnames[[1]],paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/features/name")    h5write(multi.data@i, paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/indices") # already zero-indexed.    h5write(multi.data@p, paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/indptr")    h5write(dim(multi.data), paste(args$outdir,"/overlap\_matrix.h5",sep = ''), "matrix/shape")    h5closeAll()  }  unlink(paste(args$outdir,"/overlap\_matrix.h5",sep = ''),recursive=T,force=T)  saveh5file(rna\_counts\_cells,atac\_counts\_cells)  print('save.h5 file has already!')  #########  # SoupX #  #########  run\_soupx <- function(toc,tod,rho=NULL) {    tod <- tod[rownames(toc),]    all <- toc    all <- CreateSeuratObject(all)    all <- NormalizeData(all, normalization.method = "LogNormalize", scale.factor = 10000)    all <- FindVariableFeatures(all, selection.method = "vst", nfeatures = 3000)    all.genes <- rownames(all)    all <- ScaleData(all, features = all.genes)    all <- RunPCA(all, features = VariableFeatures(all), npcs = 40, verbose = F)    all <- FindNeighbors(all, dims = 1:30)    all <- FindClusters(all, resolution = 0.5)    all <- RunUMAP(all, dims = 1:30)    matx <- all@meta.data    sc = SoupChannel(tod, toc)    sc = setClusters(sc, setNames(matx$seurat\_clusters, rownames(matx)))    sc = setContaminationFraction(sc, 0.2)    out = adjustCounts(sc)    saveRDS(sc,"sc.rds")    DropletUtils:::write10xCounts(paste(args$outdir,"/../RNA/03.Matrix/SoupxMatrix",sep=""), out,version="3")  }  toc <- Read10X(args$rna,gene.column = 1)  bac <- Read10X(args$RawMatrix,gene.column = 1)  valid <- read.table(args$BarcodeTran,header = F)  bac <- bac[rownames(toc),]  bac <- bac[,which(!(colnames(bac) %in% valid$V1))]  tod <- cbind(bac,toc)  unlink(paste(args$outdir,"/../RNA/03.Matrix/SoupxMatrix",sep=""),recursive=T,force=T)  run\_soupx(toc,tod)  print('soupx has already!')  ########################  # User choosing matrix #  ########################  if(args$ifSoupx %in% c('t','T','true','True')){    object <- Read10X(paste(args$outdir,"/../RNA/03.Matrix/SoupxMatrix",sep=""),gene.column=1)    object <- CreateSeuratObject(object)  }else{    object <- CreateSeuratObject(rna\_counts\_cells)  }  #################  # QC\_analysis.R #  #################  Find\_doublet <- function(data){    sweep.res.list <- paramSweep\_v3(data, PCs = 1:dim.usage, sct = FALSE)    sweep.stats <- summarizeSweep(sweep.res.list, GT = FALSE)    bcmvn <- find.pK(sweep.stats)    nExp\_poi <- round(as.numeric(doublets.percentage)\*ncol(data))    p<-as.numeric(as.vector(bcmvn[bcmvn$MeanBC==max(bcmvn$MeanBC),]$pK))    data <- doubletFinder\_v3(data, PCs = 1:dim.usage, pN = 0.25, pK = p, nExp = nExp\_poi, reuse.pANN = FALSE, sct = FALSE)    colnames(data@meta.data)[ncol(data@meta.data)] = "doublet\_info"    #data<-subset(data,subset=doublet\_info=="Singlet")    data  }  nfeature\_plot <- function(EC){    extra\_nfeature <- data.frame(FetchData(object = EC, vars = 'nFeature\_RNA'))    extra\_nfeature$group <- "nFeature\_RNA"    p <- ggplot(extra\_nfeature, aes(x = group, y = nFeature\_RNA, fill=group)) +      geom\_violin(trim=TRUE,color="white",show.legend = F,outlier.fill = "white",outlier.colour = "white") +      geom\_boxplot(width=0.06,position=position\_dodge(0.9),show.legend = F,fill="white",outlier.size = 0,                   outlier.stroke = 0)+      scale\_fill\_manual(values = "#999999")+      theme\_cowplot()+      theme(axis.text.x=element\_blank(),            axis.title.x=element\_blank(),            axis.ticks.x=element\_blank(),            axis.text.y=element\_text(family="Times",size=14,face="plain"),            axis.title.y=element\_text(family="Times",size = 18,face="plain"))+      guides(fill="none")+      ylab(expression("nFeature\_RNA"))+xlab("")    ylims\_feature <- extra\_nfeature %>%      group\_by(extra\_nfeature$group) %>%      summarise(Q1 = quantile(extra\_nfeature$nFeature\_RNA, 1/4,na.rm=T), Q3 = quantile(extra\_nfeature$nFeature\_RNA, 3/4,na.rm=T)) %>%      ungroup() %>%      #get lowest Q1 and highest Q3      summarise(lowQ1 = 0, highQ3 = max(Q3)\*4)    p <- p + coord\_cartesian(ylim = as.numeric(ylims\_feature))    p  }  ncount\_plot <- function(EC){    extra\_ncount <- data.frame(FetchData(object = EC, vars = 'nCount\_RNA'))    extra\_ncount$group <- "nCount\_RNA"    p <- ggplot(extra\_ncount, aes(x = group, y = nCount\_RNA, fill=group)) +      geom\_violin(trim=TRUE,color="white",show.legend = F,outlier.fill = "white",outlier.colour = "white") +      geom\_boxplot(width=0.06,position=position\_dodge(0.9),show.legend = F,fill="white",outlier.size = 0,                   outlier.stroke = 0)+      scale\_fill\_manual(values = "#E69F00")+      theme\_cowplot()+      theme(axis.text.x=element\_blank(),            axis.title.x=element\_blank(),            axis.ticks.x=element\_blank(),            axis.text.y=element\_text(family="Times",size=14,face="plain"),            axis.title.y=element\_text(family="Times",size = 18,face="plain"))+      guides(fill="none")+      ylab(expression("nCount\_RNA"))+xlab("")    ylims\_feature <- extra\_ncount %>%      group\_by(extra\_ncount$group) %>%      summarise(Q1 = quantile(extra\_ncount$nCount\_RNA, 1/4,na.rm=T), Q3 = quantile(extra\_ncount$nCount\_RNA, 3/4,na.rm=T)) %>%      ungroup() %>%      #get lowest Q1 and highest Q3      summarise(lowQ1 = 0, highQ3 = max(Q3)\*4)    p <- p + coord\_cartesian(ylim = as.numeric(ylims\_feature))    p  }  percentMt\_plot <- function(EC){    extra\_mt <- data.frame(FetchData(object = EC, vars = 'percent.mt'))    extra\_mt$group <- "percent.mt"    p <- ggplot(extra\_mt, aes(x = group, y = percent.mt, fill=group)) +      geom\_violin(trim=TRUE,color="white",show.legend = F,outlier.fill = "white",outlier.colour = "white") +      geom\_boxplot(width=0.06,position=position\_dodge(0.9),show.legend = F,fill="white",outlier.size = 0,                   outlier.stroke = 0)+      scale\_fill\_manual(values = "#56B4E9")+      theme\_cowplot()+      theme(axis.text.x=element\_blank(),            axis.title.x=element\_blank(),            axis.ticks.x=element\_blank(),            axis.text.y=element\_text(family="Times",size=14,face="plain"),            axis.title.y=element\_text(family="Times",size = 18,face="plain"))+      guides(fill="none")+      ylab(expression("percent.mt"))+xlab("")    ylims\_feature <- extra\_mt %>%      group\_by(extra\_mt$group) %>%      summarise(Q1 = quantile(extra\_mt$percent.mt, 1/4,na.rm=T), Q3 = quantile(extra\_mt$percent.mt, 3/4,na.rm=T)) %>%      ungroup() %>%      #get lowest Q1 and highest Q3      summarise(lowQ1 = 0, highQ3 = max(Q3)\*20)    p <- p + coord\_cartesian(ylim = as.numeric(ylims\_feature))    p  }  dim.usage <- as.numeric(if(!is.null(args$dim)) args$dim else 20)  doublets.percentage <- as.numeric(if(!is.null(args$percentage)) args$percentage else 0.05)  mtgene\_path <- if(!is.null(args$mtgenes)) args$mtgenes else "auto"  mtegne\_filter <- as.numeric(if(!is.null(args$mtgenepercentage)) args$mtgenepercentage else 10)  minfeatures <- as.numeric(if(!is.null(args$minfeatures)) args$minfeatures else 200)  ### Creat Seurat object  grange.counts <- StringToGRanges(rownames(atac\_counts\_cells), sep = c(":", "-"))  grange.use <- seqnames(grange.counts) %in% standardChromosomes(grange.counts)  atac\_counts\_cells <-  atac\_counts\_cells[as.vector(grange.use), ]  gtf <- rtracklayer::import(gtf)  gene.coords <- gtf[gtf$type == 'gene']  seqlevelsStyle(gene.coords) <- 'UCSC'  gene.coords <- keepStandardChromosomes(gene.coords, pruning.mode = 'coarse')  if(species.usage %in% c("mouse","Mouse","mm10","human","Human","hg19","hg38","Human\_nucleus","hg19\_premRNA","Mouse\_nucleus","Human\_2020A\_mkgtf","Human\_2020A","Human\_93","Human\_V2.3","Human\_optimizedv1","Mouse\_V2.3")){    gene.coords$gene\_biotype <- gene.coords$gene\_type  }  frag.file <- args$frag  chrom\_assay <- CreateChromatinAssay(    counts = atac\_counts\_cells,    sep = c(":", "-"),    fragments = frag.file,    min.cells = 10,    annotation = gene.coords  )  object[["ATAC"]] <- chrom\_assay  DefaultAssay(object) <- "ATAC"  object <- TSSEnrichment(object)  total\_fragments <- CountFragments(frag.file)  rownames(total\_fragments)<- total\_fragments$CB  object@meta.data$fragments <- total\_fragments[colnames(object), "frequency\_count"]  object <- FRiP(    object = object,    assay = 'ATAC',    total.fragments = 'fragments'  )  object@meta.data$loguniqueFrag <- log10(object@meta.data$fragments)  ######  # QC #  ######  DefaultAssay(object) <- "RNA"  object <- NormalizeData(object)  object <- FindVariableFeatures(object, selection.method = "vst", nfeatures = 2000)  object <- ScaleData(object)  if(dim(object)[2] <=50){    object <- RunPCA(object,npcs = (dim(object)[2]-1))  }else{    object <- RunPCA(object)  }  object <- RunUMAP(object, dims = 1:dim.usage,reduction.name = "umap.rna")  mtgene <- length(grep ("^mt-", rownames(object[["RNA"]]),value = T))  print(mtgene)  MTgene <- length(grep ("^MT-", rownames(object[["RNA"]]),value = T))  print(MTgene)  #### Plot raw and filter QC Vlnplot  png(paste(args$outdir,"/plot/",args$samplename,"\_raw\_QCplot.png",sep=""))  if(mtgene\_path != "auto"){    mt\_gene\_table <- read.table(mtgene\_path,sep="\t")    mtgene <- as.character(mt\_gene\_table[,1])    object[["percent.mt"]] <- PercentageFeatureSet(object, features = mtgene)    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p3 <- percentMt\_plot(object)    p <- p1|p2|p3  }else if(mtgene>0){    object[["percent.mt"]] <- PercentageFeatureSet(object, pattern = "^mt-")    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p3 <- percentMt\_plot(object)    p <- p1|p2|p3  }else if(MTgene>0){    object[["percent.mt"]] <- PercentageFeatureSet(object, pattern = "^MT-")    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p3 <- percentMt\_plot(object)    p <- p1|p2|p3  }else{    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p <- p1|p2  }  print(p)  dev.off()  ggsave(paste(args$outdir,"/plot/",args$samplename,"\_raw\_QCplot.svg",sep=""), p , width = 7, height = 4)  ### Filter cells with nfeatures/percent.mt  png(paste(args$outdir,"/plot/",args$samplename,"\_filter\_QCplot.png",sep=""))  objectmeta <- object@meta.data[order(-object@meta.data$nFeature\_RNA),]  n95 <- as.numeric(as.integer(nrow(objectmeta) \* 0.05))  n95features <- as.numeric(objectmeta[n95,"nFeature\_RNA"])  if(mtgene\_path != "auto"){    object <- subset(object, subset = nFeature\_RNA > minfeatures & nFeature\_RNA < n95features & percent.mt < mtegne\_filter)    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p3 <- percentMt\_plot(object)    p <- p1|p2|p3  }else if(mtgene>0){    object <- subset(object, subset = nFeature\_RNA > minfeatures & nFeature\_RNA < n95features & percent.mt < mtegne\_filter)    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p3 <- percentMt\_plot(object)    p <- p1|p2|p3  }else if(MTgene>0){    object <- subset(object, subset = nFeature\_RNA > minfeatures & nFeature\_RNA < n95features & percent.mt < mtegne\_filter)    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p3 <- percentMt\_plot(object)    p <- p1|p2|p3  }else{    object <- subset(object, subset = nFeature\_RNA > minfeatures & nFeature\_RNA < n95features)    p1 <- nfeature\_plot(object)    p2 <- ncount\_plot(object)    p <- p1|p2  }  print(p)  dev.off()  ggsave(paste(args$outdir,"/plot/",args$samplename,"\_filter\_QCplot.svg",sep=""), p , width = 7, height = 4)  #Find doublets  if(dim(object)[2] >50){    object <- Find\_doublet(object)    write.table(object@meta.data,paste0(args$outdir,"/",args$samplename,"\_doublets\_info.txt"),sep="\t",quote=FALSE)    object <- subset(object,subset=doublet\_info=="Singlet")  }  object@meta.data$split = args$samplename  print('QC plotting has already!')  ############  # analysis #  ############  # ATAC analysis  # We exclude the first dimension as this is typically correlated with sequencing depth  DefaultAssay(object) <- "ATAC"  object <- RunTFIDF(object)  object <- FindTopFeatures(object, min.cutoff = 'q0')  object <- RunSVD(object)  object <- RunUMAP(object, reduction = 'lsi', dims = 2:30, reduction.name = "umap.atac", reduction.key = "atacUMAP\_")  object <- FindMultiModalNeighbors(object, reduction.list = list("pca", "lsi"), dims.list = list(1:30, 2:30))  object <- RunUMAP(object, nn.name = "weighted.nn", reduction.name = "wnn.umap", reduction.key = "wnnUMAP\_")  object <- FindClusters(object, graph.name = "wsnn", algorithm = 3, verbose = FALSE)  p1 <- DimPlot(object, reduction = "umap.rna", label = TRUE,pt.size = 0.8, label.size = 4, repel = TRUE) + ggtitle("RNA")+NoLegend()  p2 <- DimPlot(object, reduction = "umap.atac", label = TRUE,pt.size = 0.8, label.size = 4, repel = TRUE) + ggtitle("ATAC")+NoLegend()  p3 <- DimPlot(object, reduction = "wnn.umap", label = TRUE,pt.size = 0.8, label.size = 4, repel = TRUE) + ggtitle("WNN")  ggsave(paste(args$outdir,"/plot/wnn\_cluster.png",sep = ''),p1+p2+p3,width=26,height=8)  ggsave(paste(args$outdir,"/plot/wnn\_cluster.svg",sep = ''),p1+p2+p3,width=26,height=8)  print('wnn cluster plotting has already!')  saveRDS(object,paste0(args$outdir,"/WNN\_cluster.RDS"))  ## get clutering data for html plotting  cluster\_ID=as.data.frame(Idents(object = object))  cluster\_cor= as.data.frame(Embeddings(object = object,reduction = "umap.atac"))  coor=cbind(cluster\_ID,cluster\_cor,object[['nCount\_ATAC']],object[['nFeature\_ATAC']])  colnames(coor) = c("Cluster","UMAP\_1","UMAP\_2","nCount\_ATAC","nFeature\_ATAC")  coorOrder = coor[order(coor$Cluster),]  temp <- coorOrder  names <- rownames(temp)  rownames(temp) <- NULL  dataTemp <- cbind(names,temp)  rm(temp,names,coorOrder,coor,cluster\_cor,cluster\_ID)  cluster\_stat <- as.data.frame(table(dataTemp$Cluster))  colnames(cluster\_stat) <- c("Cluster","cellNum")  cluster\_cell <- dplyr::left\_join(dataTemp,cluster\_stat,by="Cluster")  length = nrow(cluster\_stat)  write.csv(cluster\_stat, file=paste(args$outdir,"/cluster\_cell\_atac.stat",sep=""),quote=FALSE)  # atac annotation  Annotation(object) <- gene.coords  gene.activities <- GeneActivity(object)  if (species.usage %in% c("mouse","Mouse","mm10","human","Human","hg19","hg38","Human\_nucleus","hg19\_premRNA","Mouse\_nucleus","Human\_2020A\_mkgtf","Human\_2020A","Human\_93","Human\_V2.3","Human\_optimizedv1","Mouse\_V2.3")){    if (species.usage == "mouse"||species.usage == "Mouse"||species.usage == "mm10"||species.usage == "Mouse\_nucleus" || species.usage == "Mouse\_V2.3"){      object.combined\_cell\_type <- scMCA(scdata = gene.activities, numbers\_plot = 3)      out=as.data.frame(unlist(object.combined\_cell\_type$scMCA))      out$`unlist(object.combined\_cell\_type$scMCA)`=as.character(out$`unlist(object.combined\_cell\_type$scMCA)`)    }    else if (species.usage == "human"||species.usage == "Human"||species.usage == "hg19"||species.usage == "hg38" ||species.usage == "Human\_nucleus" || species.usage == "hg19\_premRNA" || species.usage == "Human\_2020A\_mkgtf" || species.usage == "Human\_2020A" ||species.usage == "Human\_93" || species.usage == "Human\_V2.3" ||species.usage == "Human\_optimizedv1"){      object.combined\_cell\_type <- scHCL(scdata = gene.activities, numbers\_plot = 3)      out=as.data.frame(unlist(object.combined\_cell\_type$scHCL))      out$`unlist(object.combined\_cell\_type$scHCL)`=as.character(out$`unlist(object.combined\_cell\_type$scHCL)`)    }    object@meta.data$cell\_type\_atac=out[match(rownames(object@meta.data),rownames(out)),1]    out\_meta=object@meta.data    table\_list=list()    for(i in 0:(length(unique(object$seurat\_clusters))-1)){      a=i+1      table\_list[[a]] <- tryCatch(        {          sub=subset(out\_meta,out\_meta$seurat\_clusters==i)          tab=as.data.frame(table(sub$cell\_type\_atac))          tab\_order=tab[order(tab[,2],decreasing = T),]          tab\_order$Cluster=i          #table\_list[[a]]=tab\_order[1,]          tab\_order[1,]        },error=function(e){table\_list[[a]]='NA'}      )    }    sum=do.call(rbind,table\_list)    use=out\_meta[,c("seurat\_clusters","cell\_type\_atac")]    use$seurat\_clusters=as.character(use$seurat\_clusters)    use$ID=rownames(out\_meta)    colnames(sum)=c("predicated.cell.type","Freq","seurat\_clusters")    res=merge(use,sum,by="seurat\_clusters")    object@meta.data$predicated.cell.type.atac=res[match(rownames(out\_meta),res$ID),4]  }else{    print('This refcode has not auto annotation')  }  ## create cluster.csv  if(species.usage %in% c("mouse","Mouse","mm10","human","Human","hg19","hg38","Human\_nucleus","hg19\_premRNA","Mouse\_nucleus","Human\_2020A\_mkgtf","Human\_2020A","Human\_93","Human\_V2.3","Human\_optimizedv1","Mouse\_V2.3")){    cluster\_anno <- as.data.frame(table(object@meta.data$cell\_type\_atac,object@meta.data$seurat\_clusters))    sorted\_cluster\_anno <- cluster\_anno[order(cluster\_anno$Var2,-cluster\_anno$Freq),]    finl\_cluster\_anno <- sorted\_cluster\_anno[!duplicated(sorted\_cluster\_anno$Var2),]    finl\_cluster\_anno <- finl\_cluster\_anno[,c(1,2)]    colnames(finl\_cluster\_anno) <- c("CellType","Cluster")    finl\_cluster\_anno$CellType <- as.character(finl\_cluster\_anno$CellType)    celltype\_cell <- dplyr::left\_join(cluster\_cell,finl\_cluster\_anno,by="Cluster")    temp <- as.data.frame(table(celltype\_cell$CellType))    colnames(temp) <- c("CellType","CellTypeNum")    celltype\_cell <- dplyr::left\_join(celltype\_cell,temp,by="CellType")    rm(temp)    celltype\_cell\_merge <- unite(celltype\_cell, "Cluster", Cluster, cellNum, sep = " CellsNum: ")    celltype\_cell\_merge <- unite(celltype\_cell\_merge, "Predicted cell type", CellType, CellTypeNum, sep = ": ")    rownames(celltype\_cell\_merge) <- celltype\_cell\_merge[,1]    celltype\_cell\_merge <- celltype\_cell\_merge[,-1]    write.csv(celltype\_cell\_merge, file=paste(args$outdir,"/report/ATAC/6.cluster.csv",sep=""),quote=FALSE)    # plot    p1 <- DimPlot(object, reduction = "umap.rna", group.by = "predicated.cell.type.atac", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("RNA")+NoLegend()    p2 <- DimPlot(object, reduction = "umap.atac", group.by = "predicated.cell.type.atac", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("ATAC")+NoLegend()    p3 <- DimPlot(object, reduction = "wnn.umap", group.by = "predicated.cell.type.atac", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("WNN")    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_atac.png",sep = ''),p1+p2+p3,width=26,height=8)    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_atac.svg",sep = ''),p1+p2+p3,width=26,height=8)  }else{    object@meta.data$cell\_type\_atac <- 'NULL'    finl\_cluster\_anno <- as.data.frame(table(object@meta.data$cell\_type\_atac,object@meta.data$seurat\_clusters))    finl\_cluster\_anno <- finl\_cluster\_anno[,c(1,2)]    colnames(finl\_cluster\_anno) <- c("CellType","Cluster")    finl\_cluster\_anno$CellType <- as.character(finl\_cluster\_anno$CellType)    celltype\_cell <- dplyr::left\_join(cluster\_cell,finl\_cluster\_anno,by="Cluster")    temp <- as.data.frame(table(celltype\_cell$CellType))    colnames(temp) <- c("CellType","CellTypeNum")    celltype\_cell <- dplyr::left\_join(celltype\_cell,temp,by="CellType")    rm(temp)    celltype\_cell\_merge <- unite(celltype\_cell, "Cluster", Cluster, cellNum, sep = " CellsNum: ")    celltype\_cell\_merge <- unite(celltype\_cell\_merge, "Predicted cell type", CellType, CellTypeNum, sep = ": ")    rownames(celltype\_cell\_merge) <- celltype\_cell\_merge[,1]    celltype\_cell\_merge <- celltype\_cell\_merge[,-1]    write.csv(celltype\_cell\_merge, file=paste(args$outdir,"/report/ATAC/6.cluster.csv",sep=""),quote=FALSE)    # plot    p1 <- DimPlot(object, reduction = "umap.rna", group.by = "seurat\_clusters", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("RNA")+NoLegend()    p2 <- DimPlot(object, reduction = "umap.atac", group.by = "seurat\_clusters", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("ATAC")+NoLegend()    p3 <- DimPlot(object, reduction = "wnn.umap", group.by = "seurat\_clusters", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("WNN")    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_atac.png",sep = ''),p1+p2+p3,width=26,height=8)    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_atac.svg",sep = ''),p1+p2+p3,width=26,height=8)  }  print('atac annotation has already!')    # rna annotation  DefaultAssay(object) <- "RNA"  rna\_data <- GetAssayData(object = object, slot = "data")  if (species.usage %in% c("mouse","Mouse","mm10","human","Human","hg19","hg38","Human\_nucleus","hg19\_premRNA","Mouse\_nucleus","Human\_2020A\_mkgtf","Human\_2020A","Human\_93","Human\_V2.3","Human\_optimizedv1","Mouse\_V2.3")){    if (species.usage == "mouse"||species.usage == "Mouse"||species.usage == "mm10"||species.usage == "Mouse\_nucleus" || species.usage == "Mouse\_V2.3"){      object.combined\_cell\_type <- scMCA(scdata = rna\_data, numbers\_plot = 3)      out=as.data.frame(unlist(object.combined\_cell\_type$scMCA))      out$`unlist(object.combined\_cell\_type$scMCA)`=as.character(out$`unlist(object.combined\_cell\_type$scMCA)`)    }    else if (species.usage == "human"||species.usage == "Human"||species.usage == "hg19"||species.usage == "hg38" ||species.usage == "Human\_nucleus" || species.usage == "hg19\_premRNA" || species.usage == "Human\_2020A\_mkgtf" || species.usage == "Human\_2020A" ||species.usage == "Human\_93" || species.usage == "Human\_V2.3" ||species.usage == "Human\_optimizedv1"){      object.combined\_cell\_type <- scHCL(scdata = rna\_data, numbers\_plot = 3)      out=as.data.frame(unlist(object.combined\_cell\_type$scHCL))      out$`unlist(object.combined\_cell\_type$scHCL)`=as.character(out$`unlist(object.combined\_cell\_type$scHCL)`)    }    object@meta.data$cell\_type\_rna=out[match(rownames(object@meta.data),rownames(out)),1]    out\_meta=object@meta.data    table\_list=list()    for(i in 0:(length(unique(object$seurat\_clusters))-1)){    a=i+1    table\_list[[a]] <- tryCatch(      {        sub=subset(out\_meta,out\_meta$seurat\_clusters==i)        tab=as.data.frame(table(sub$cell\_type\_rna))        tab\_order=tab[order(tab[,2],decreasing = T),]        tab\_order$Cluster=i        tab\_order[1,]      },error=function(e){table\_list[[a]]='NA'}    )    sum=do.call(rbind,table\_list)    use=out\_meta[,c("seurat\_clusters","cell\_type\_rna")]    use$seurat\_clusters=as.character(use$seurat\_clusters)    use$ID=rownames(out\_meta)    colnames(sum)=c("predicated.cell.type","Freq","seurat\_clusters")    res=merge(use,sum,by="seurat\_clusters")    object@meta.data$predicated.cell.type.rna=res[match(rownames(out\_meta),res$ID),4]    }    print('rna annotation has already!')    # plot    p1 <- DimPlot(object, reduction = "umap.rna", group.by = "predicated.cell.type.rna", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("RNA")+NoLegend()    p2 <- DimPlot(object, reduction = "umap.atac", group.by = "predicated.cell.type.rna", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("ATAC")+NoLegend()    p3 <- DimPlot(object, reduction = "wnn.umap", group.by = "predicated.cell.type.rna", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("WNN")    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_rna.png",sep = ''),p1+p2+p3,width=26,height=8)    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_rna.svg",sep = ''),p1+p2+p3,width=26,height=8)  }else{    p1 <- DimPlot(object, reduction = "umap.rna", group.by = "seurat\_clusters", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("RNA")+NoLegend()    p2 <- DimPlot(object, reduction = "umap.atac", group.by = "seurat\_clusters", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("ATAC")+NoLegend()    p3 <- DimPlot(object, reduction = "wnn.umap", group.by = "seurat\_clusters", label = TRUE, label.size = 2.5, repel = TRUE) + ggtitle("WNN")    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_rna.png",sep = ''),p1+p2+p3,width=26,height=8)    ggsave(paste(args$outdir,"/plot/wnn\_annotation\_rna.svg",sep = ''),p1+p2+p3,width=26,height=8)    print('This refcode has not auto annotation')  }  ##-----------------------------  ## create cluster\_cell\_rna.stat  cluster\_ID=as.data.frame(Idents(object = object))  cluster\_cor= as.data.frame(Embeddings(object = object,reduction = "umap.rna"))  coor=cbind(cluster\_ID,cluster\_cor,object[['nCount\_RNA']],object[['nFeature\_RNA']])  colnames(coor) = c("Cluster","UMAP\_1","UMAP\_2","nUMI","nGene")  coorOrder = coor[order(coor$Cluster),]  temp <- coorOrder  names <- rownames(temp)  rownames(temp) <- NULL  dataTemp <- cbind(names,temp)  rm(temp)  rm(names)  rm(coorOrder)  rm(coor)  rm(cluster\_cor)  rm(cluster\_ID)  cluster\_stat <- as.data.frame(table(dataTemp$Cluster))  colnames(cluster\_stat) <- c("Cluster","cellNum")  cluster\_cell <- dplyr::left\_join(dataTemp,cluster\_stat,by="Cluster")  write.csv(cluster\_stat, file=paste(args$out,"/cluster\_cell\_rna.stat",sep=""),quote=FALSE)  ##-----------------------  ## create cluster\_rna.csv  if(species.usage %in% c("mouse","Mouse","mm10","human","Human","hg19","hg38","Human\_nucleus","hg19\_premRNA","Mouse\_nucleus","Human\_2020A\_mkgtf","Human\_2020A","Human\_93","Human\_V2.3","Human\_optimizedv1","Mouse\_V2.3")){    cluster\_anno <- as.data.frame(table(object@meta.data$cell\_type\_rna,object@meta.data$seurat\_clusters))    sorted\_cluster\_anno <- cluster\_anno[order(cluster\_anno$Var2,-cluster\_anno$Freq),]    finl\_cluster\_anno <- sorted\_cluster\_anno[!duplicated(sorted\_cluster\_anno$Var2),]    finl\_cluster\_anno <- finl\_cluster\_anno[,c(1,2)]    colnames(finl\_cluster\_anno) <- c("CellType","Cluster")    finl\_cluster\_anno$CellType <- as.character(finl\_cluster\_anno$CellType)    celltype\_cell <- dplyr::left\_join(cluster\_cell,finl\_cluster\_anno,by="Cluster")    temp <- as.data.frame(table(celltype\_cell$CellType))    colnames(temp) <- c("CellType","CellTypeNum")    celltype\_cell <- dplyr::left\_join(celltype\_cell,temp,by="CellType")    rm(temp)    celltype\_cell\_merge <- unite(celltype\_cell, "Cluster", Cluster, cellNum, sep = " CellsNum: ")    celltype\_cell\_merge <- unite(celltype\_cell\_merge, "Predicted cell type", CellType, CellTypeNum, sep = ": ")    rownames(celltype\_cell\_merge) <- celltype\_cell\_merge[,1]    celltype\_cell\_merge <- celltype\_cell\_merge[,-1]    write.csv(celltype\_cell\_merge, file=paste(args$outdir,"/cluster\_rna.csv",sep=""),quote=FALSE)  }else{    object@meta.data$cell\_type\_rna <- 'NULL'    finl\_cluster\_anno <- as.data.frame(table(object@meta.data$cell\_type\_rna,object@meta.data$seurat\_clusters))    finl\_cluster\_anno <- finl\_cluster\_anno[,c(1,2)]    colnames(finl\_cluster\_anno) <- c("CellType","Cluster")    finl\_cluster\_anno$CellType <- as.character(finl\_cluster\_anno$CellType)    celltype\_cell <- dplyr::left\_join(cluster\_cell,finl\_cluster\_anno,by="Cluster")    temp <- as.data.frame(table(celltype\_cell$CellType))    colnames(temp) <- c("CellType","CellTypeNum")    celltype\_cell <- dplyr::left\_join(celltype\_cell,temp,by="CellType")    rm(temp)    celltype\_cell\_merge <- unite(celltype\_cell, "Cluster", Cluster, cellNum, sep = " CellsNum: ")    celltype\_cell\_merge <- unite(celltype\_cell\_merge, "Predicted cell type", CellType, CellTypeNum, sep = ": ")    rownames(celltype\_cell\_merge) <- celltype\_cell\_merge[,1]    celltype\_cell\_merge <- celltype\_cell\_merge[,-1]    write.csv(celltype\_cell\_merge, file=paste(args$outdir,"/cluster\_rna.csv",sep=""),quote=FALSE)  }  saveRDS(object,paste0(args$outdir,"/WNN\_annotation.RDS"))  # FindAllMarker  allmarkers.rna <- FindAllMarkers(object, assay = 'RNA')  write.csv(as.data.frame(allmarkers.rna[,c(6,5,1,2,3,4)]),file= paste0(args$outdir,"/",args$samplename,"\_rna\_marker.csv"),quote=FALSE)  cat(paste("Number of cells used for clustering,", length(colnames(object)), "\n",sep=""),      file=paste(args$outdir,"/cell\_report\_2.csv",sep=""))  # statistic of the mitochondria mapping rate  fragment <- as.data.frame(fread(frag.file,header=F))  MT <- fragment[grep(paste0("^",args$chrmt),fragment$V1),]  m1=data.frame(qc="Mitochondria reads ratio",num=paste0(100\*as.numeric(sprintf("%0.4f",sum(MT$V5)/sum(fragment$V5))),"%"),stringsAsFactors = FALSE)  write.table(m1,paste0(args$outdir,"/report/ATAC/3.mapping.csv"),sep=":",quote = FALSE,row.names = FALSE,col.names = FALSE)  print('WNN.R part has already!') |