Organoid scRNA-seq

2023-12-23

## Step 0. Setting Up the Environment

Begin by setting working directory and output directory.

### Note: This is taken from Deepti Murthy and adapted by Andrew Gjelsteen,  
### most of the comments are hers ###  
###\* Andrew's comments are denoted by ###\* \*###  
#TCW Single-Cell RNA Seq Data  
#Dataset Source: TCW\_14311\_run1  
  
#Date: 11/10/22 - 11/13/22  
#Date: 1/1/23 (Altering Cluster Resolution)  
  
#Note, prior to this analysis:  
#The seurat\_object.rds" object was created using RNA and HTO information, including  
#only those cells with at least 10 nUMI for any given HTO.  
#HTODemux was run using default parameters.  
#Across-sample Doublets and dropouts have been removed from the dataset.  
  
#In this analysis, I remove cells with high mitochondrial percentage and   
## remove outliers, scale, normalize, and cluster.  
###  
#R version 4.3.1  
sessionInfo()  
  
#load libraries   
library(dplyr)  
library(patchwork)  
library(ggplot2)  
library(gridExtra)  
library(Matrix)  
library(scales)  
library(cowplot)  
library(Seurat) #Seurat v.3.0  
library(RColorBrewer)  
library(BiocManager) #v 3.17  
library(tibble)  
library(patchwork)  
library(flexmix)  
library(miQC)  
library(singleCellTK)  
library(fgsea)  
library(singleCellTK)  
library(DropletUtils)  
library(SeuratData)  
library(tidyr)  
.libPaths()  
source("~/.Rprofile")  
  
  
setwd("/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/")  
  
dir <- "/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs"  
out<-'outputs/simpleafSeurat'

## Step 1: Read in Seurat Files, Combining them into one object, adding meta.data, and performing first QC steps:

### Subsection 1: Reading in RDS files, adding meta.data, and merging them into one.

This object will be used to be a point of comparison for before/after quality control steps.

###\* These are my individual samples' Seurat files. Produced by previous script. \*###  
S1 <- readRDS(file = "./outputs/simpleafSeurat/sample1.rds")  
S2 <- readRDS(file = "./outputs/simpleafSeurat/sample2.rds")  
S3 <- readRDS(file = "./outputs/simpleafSeurat/sample3.rds")  
S4 <- readRDS(file = "./outputs/simpleafSeurat/sample4.rds")  
###\* No Sample 5 (deemed poor quality by researchers) \*###  
S6 <- readRDS(file = "./outputs/simpleafSeurat/sample6.rds")  
###\* No Sample 7 (deemed poor quality by researchers) \*###  
S8 <- readRDS(file = "./outputs/simpleafSeurat/sample8.rds")  
S9 <- readRDS(file = "./outputs/simpleafSeurat/sample9.rds")  
S10 <- readRDS(file = "./outputs/simpleafSeurat/sample10.rds")  
S11 <- readRDS(file = "./outputs/simpleafSeurat/sample11.rds")  
S12 <- readRDS(file = "./outputs/simpleafSeurat/sample12.rds")  
S13 <- readRDS(file = "./outputs/simpleafSeurat/sample13.rds")  
S14 <- readRDS(file = "./outputs/simpleafSeurat/sample14.rds")  
S15 <- readRDS(file = "./outputs/simpleafSeurat/sample15.rds")  
S16 <- readRDS(file = "./outputs/simpleafSeurat/sample16.rds")  
S17 <- readRDS(file = "./outputs/simpleafSeurat/sample17.rds")  
S18 <- readRDS(file = "./outputs/simpleafSeurat/sample18.rds")  
S19 <- readRDS(file = "./outputs/simpleafSeurat/sample19.rds")  
S20 <- readRDS(file = "./outputs/simpleafSeurat/sample20.rds")  
S21 <- readRDS(file = "./outputs/simpleafSeurat/sample21.rds")  
S22 <- readRDS(file = "./outputs/simpleafSeurat/sample22.rds")  
S23 <- readRDS(file = "./outputs/simpleafSeurat/sample23.rds")  
S24 <- readRDS(file = "./outputs/simpleafSeurat/sample24.rds")  
S25 <- readRDS(file = "./outputs/simpleafSeurat/sample25.rds")  
S26 <- readRDS(file = "./outputs/simpleafSeurat/sample26.rds")  
  
###\* Proceed by defining APOE genotype groups. \*###  
APOE22\_samples\_group <- c("S3", "S9", "S12", "S15", "S21", "S25")  
APOE33\_samples\_group <- c("S1", "S8", "S14", "S16", "S19", "S23")  
APOE33Ch\_samples\_group <- c("S2", "S6", "S13", "S18", "S20", "S24")  
APOE44\_samples\_group <- c("S4", "S10", "S11", "S17", "S22", "S26")  
  
sample\_names\_group22 <- c("sample3", "sample9", "sample12", "sample15", "sample21", "sample25")  
sample\_names\_group33 <- c("sample1", "sample8", "sample14", "sample16", "sample19", "sample23")  
sample\_names\_group33Ch <- c("sample2", "sample6", "sample13", "sample18", "sample20", "sample24")  
sample\_names\_group44 <- c("sample4", "sample10", "sample11", "sample17", "sample22", "sample26")  
  
# APOE22: S3, S9, S12, S15, S21, S25  
# APOE33: S1, S8, S14, S16, S19, S23  
# APOE33Ch: S2, S6, S13, S18, S20, S24  
# APOE44: S4, S10, S11, S17, S22, S26 \*##  
  
out <- "/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/simpleafSeurat/"  
  
###\* Loop through each sample and add a metadata column for sample labels,  
###\* then save each individual Seurat object again.  
###\* This step is important, so that when we read in these files in Step 2,  
###\* we will have proper meta.data assigned to them.  
  
# for APOE22  
for (i in seq\_along(APOE22\_samples\_group)) {  
 object\_name <- APOE22\_samples\_group[i]  
 sample\_name <- sample\_names\_group22[i]  
   
 eval(parse(text = paste0(object\_name, "[['sample']] <- '", sample\_name, "'")))  
 eval(parse(text = paste0(object\_name, "[['genotype']] <- 'APOE22'")))  
  
 filename <- paste0(out, "APOE22\_sample", i, ".rds")  
 eval(parse(text = paste0("saveRDS(", object\_name, ", file = '", filename, "')")))  
}  
  
# for APOE33  
for (i in seq\_along(APOE33\_samples\_group)) {  
 object\_name <- APOE33\_samples\_group[i]  
 sample\_name <- sample\_names\_group33[i]  
   
 eval(parse(text = paste0(object\_name, "[['sample']] <- '", sample\_name, "'")))  
 eval(parse(text = paste0(object\_name, "[['genotype']] <- 'APOE33'")))  
  
 filename <- paste0(out, "APOE33\_sample", i, ".rds")  
 eval(parse(text = paste0("saveRDS(", object\_name, ", file = '", filename, "')")))  
}  
  
# for APOE33Ch  
for (i in seq\_along(APOE33Ch\_samples\_group)) {  
 object\_name <- APOE33Ch\_samples\_group[i]  
 sample\_name <- sample\_names\_group33Ch[i]  
   
 eval(parse(text = paste0(object\_name, "[['sample']] <- '", sample\_name, "'")))  
 eval(parse(text = paste0(object\_name, "[['genotype']] <- 'APOE33Ch'")))  
  
 filename <- paste0(out, "APOE33Ch\_sample", i, ".rds")  
 eval(parse(text = paste0("saveRDS(", object\_name, ", file = '", filename, "')")))  
}  
  
# for APOE44  
for (i in seq\_along(APOE44\_samples\_group)) {  
 object\_name <- APOE44\_samples\_group[i]  
 sample\_name <- sample\_names\_group44[i]  
   
 eval(parse(text = paste0(object\_name, "[['sample']] <- '", sample\_name, "'")))  
 eval(parse(text = paste0(object\_name, "[['genotype']] <- 'APOE44'")))  
  
 filename <- paste0(out, "APOE44\_sample", i, ".rds")  
 eval(parse(text = paste0("saveRDS(", object\_name, ", file = '", filename, "')")))  
}  
  
  
##\* Merge all Seurat objects into one \*##  
##\* Can easily process it as one object \*##  
##\*   
organoid0 <- merge(S1, y = c(S2, S3, S4, S6, S8, S9, S10,  
 S11, S12, S13, S14, S15, S16, S17, S18, S19,  
 S20, S21, S22, S23, S24, S25, S26),   
 add.cell.ids = c("S1", "S2", "S3", "S4", "S6", "S8", "S9", "S10",  
 "S11", "S12", "S13", "S14", "S15", "S16", "S17", "S18", "S19",  
 "S20", "S21", "S22", "S23", "S24", "S25", "S26"),  
 project = "APOE\_Jorganoid"  
)  
  
###\* save the merged objects' Seurat file as organoid0  
out <- "/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/simpleafSeurat/"  
filename = 'organoid0.rds'  
saveRDS(organoid0, file.path(out, filename))

Now that we have combined our Seurat objects into one composite Seurat object, with genotype and sample identifiers added, and saved it as organoid0.rds, we can restart our R session and load the saved Seurat file using the readRDS() function in the Seurat library.

### Subsection 2: Labeling Mitochondrial Percentage

In my experience working with this particular dataset, I had to convert ENSEMBL identifiers to gene symbol identifiers. Strangely enough, the genes which correspond to mitochondrial DNA genes, which are normally denoted by the ‘MT-’ prefix, were not denoted by this prefix. In order to handle this unusual situation, I created a list of all MT DNA ENSEMBL IDs and gene symbols, then scanned through the Seurat object and used these to identify mtDNA percentages. Below is my method of doing this, with Deepti’s original method, which is what you will most likely be using.

Deepti’s method for doing this (what you will most likely be using):

###\* Deepti's original method for labeling mitochondrial DNA percentages: \*###  
#label MT percent in each cell  
organoid0[["percent.mt"]] <- PercentageFeatureSet(organoid0, pattern = "^MT-")  
###\* Save RDS file again to save it with this new metadata added.  
saveRDS(organoid0, file.path(out, filename))

### Subsection 3: Making Violin Plots for Seurat data before undergoing QC:

When working with our Seurat scRNA-seq data, we will want to save violin plots of the object before we perform Quality Control (QC) steps. This is essential to quantify how many cells were recovered following the QC process.

# Loop through each genotype  
library(Seurat)  
library(ggplot2)  
library(reshape2)  
  
# Output directory  
out <- "/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/images/" # Replace with your actual   
  
# Define the features to plot  
features <- c("nFeature\_RNA", "nCount\_RNA", "percent.mt")  
  
# Loop through each genotype and make violin plots for each APOE genotype  
for (genotype in unique\_genotypes) {  
 # Subset Seurat object by genotype  
 organoid\_subset <- subset(organoid0, subset = genotype == genotype)  
   
 # Create violin plots  
 vps <- VlnPlot(object = organoid\_subset, features = c("nFeature\_RNA", "nCount\_RNA", "percent.mt"), ncol = 3, slot = "counts", pt.size = 0)  
   
 # Modify the plots to spread them out evenly  
 vps <- vps + plot\_layout(guides = 'collect') & theme(legend.position = 'none')  
   
 # Add the genotype as the overall figure title  
 combined\_plot <- vps + plot\_annotation(title = genotype, theme = theme(plot.title = element\_text(hjust = 0.5)))  
   
 # Construct the full path for the output file  
 output\_file\_path <- file.path(out, paste0("before\_QC\_violin\_plots\_", genotype, ".png"))  
   
 # Save the combined plot to the specified directory  
 ggsave(output\_file\_path, plot = combined\_plot, width = 12, height = 6, units = "in")  
}

Now that we have merged our object, assigned metadata labels, saved the before-QC violin plots, and saved our RDS object, we can proceed with performing the actual QC.

## Step 2: Perform QC Across All Samples then visualize results

Let’s begin by restarting our R session. This way we can clear the cache memory, since this file is very large (~15GB in my case). We will begin by loading in the .rds object, creating a dataframe which will store the before/after QC metrics. Then we will write a loop, which will iterate through the samples and perform QC while saving the before / after numbers of cells.

### Subsection 1:

library(Seurat)  
library(dplyr)  
  
organoid0 <- readRDS('outputs/simpleafSeurat/organoid0.rds')  
# List of sample identifiers  
samples <- c("1", "2", "3", "4", "6", "8", "9", "10",  
 "11", "12", "13", "14", "15", "16", "17", "18", "19",  
 "20", "21", "22", "23", "24", "25", "26")  
  
# Initialize a data frame to store cell counts  
cell\_counts <- data.frame(sample = character(),  
 cells\_before\_QC = integer(),  
 cells\_after\_QC = integer(),  
 stringsAsFactors = FALSE)  
  
# Initialize variables for combined statistics  
stats <- list(  
 APOE22 = list(cells\_before = 0, cells\_after = 0, genes\_before = vector(), genes\_after = vector()),  
 APOE33 = list(cells\_before = 0, cells\_after = 0, genes\_before = vector(), genes\_after = vector()),  
 APOE33Ch = list(cells\_before = 0, cells\_after = 0, genes\_before = vector(), genes\_after = vector()),  
 APOE44 = list(cells\_before = 0, cells\_after = 0, genes\_before = vector(), genes\_after = vector())  
)

Note, we defined our samples group vectors before, but I’ll include it again here in case you restarted your R session and need to run it again. This is a crucial step because we need to keep these data to

###\* Define our sample groups again  
APOE22\_samples\_group <- c("S3", "S9", "S12", "S15", "S21", "S25")  
APOE33\_samples\_group <- c("S1", "S8", "S14", "S16", "S19", "S23")  
APOE33Ch\_samples\_group <- c("S2", "S6", "S13", "S18", "S20", "S24")  
APOE44\_samples\_group <- c("S4", "S10", "S11", "S17", "S22", "S26")  
  
sample\_names\_group22 <- c("sample3", "sample9", "sample12", "sample15", "sample21", "sample25")  
sample\_names\_group33 <- c("sample1", "sample8", "sample14", "sample16", "sample19", "sample23")  
sample\_names\_group33Ch <- c("sample2", "sample6", "sample13", "sample18", "sample20", "sample24")  
sample\_names\_group44 <- c("sample4", "sample10", "sample11", "sample17", "sample22", "sample26")

Now, this step is fairly complex. The general breakdown of what is occurring is this: \* Each individual rds file is read in (individual meaning the individual sample files). \* Number of cells before QC is recorded for each sample. \* Genes which are only expressed in >3 cells are filtered out. \* Percentage of mtDNA genes is recalculated (in case it was lost before – This step is likely unnecessarily). \* Cells with greater than 5% mtDNA are filtered out. \* Upper limit of nCount\_RNA (corresponds to number of RNA molecules detected per cell) is set to 99.5th percentile. Lower limit is defined by the CalculateBarcodeInflections() function (a function within the Seurat library). \* Cells are then filtered based on having nCount\_RNA between these two cutoffs. \* Append the cell count information to the dataframe we defined in the previous subsection (just above here).

out <- '/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/simpleafSeurat'  
  
# Loop over each sample  
for (sample\_id in samples) {  
 # Load the Seurat object  
 T\_QC <- readRDS(file = paste0("./outputs/simpleafSeurat/sample", sample\_id, ".rds"))  
   
 # Record the number of cells before QC  
 cells\_before <- ncol(T\_QC)  
   
 # Filter genes expressed in less than 3 cells  
 # Using sparse matrix directly to avoid memory issues  
 raw\_counts\_mat <- T\_QC@assays$RNA@counts  
 genes\_to\_keep <- Matrix::colSums(raw\_counts\_mat != 0) >= 3  
 T\_QC <- subset(T\_QC, features = rownames(T\_QC)[genes\_to\_keep])  
   
 # Set Idents, calculate MT and Ribosomal percentages  
 Idents(T\_QC) <- T\_QC$orig.ident  
 # Subset mt\_gene\_names to include only genes present in the Seurat object  
 valid\_mt\_gene\_names <- mt\_gene\_names[mt\_gene\_names %in% rownames(T\_QC[["RNA"]]@counts)]  
   
 # Recalculate the percentage of mitochondrial genes  
 T\_QC[["percent.mt"]] <- PercentageFeatureSet(T\_QC, features = valid\_mt\_gene\_names)  
   
 # Filter out cells with greater than 5% mtDNA  
 T\_QC <- subset(T\_QC, subset = percent.mt <= 5)  
   
 # Assign percent.ribo to cells  
 T\_QC[["percent.ribo"]] <- PercentageFeatureSet(T\_QC, pattern = "^RP[SL]")  
   
 # Calculate upper limit, set at 99.5 percentile  
 upper\_limit <- quantile(T\_QC@meta.data$nCount\_RNA, probs = 0.995)  
   
 # Calculate Barcode Inflections for lower limit  
 T\_QC <- CalculateBarcodeInflections(  
 T\_QC,  
 barcode.column = "nCount\_RNA",  
 group.column = "orig.ident",  
 threshold.low = 1000,  
 threshold.high = NULL  
 )  
 T\_QC <- SubsetByBarcodeInflections(object = T\_QC)  
   
 # Filter cells based on RNA count limits  
 T\_QC <- subset(T\_QC, subset = nCount\_RNA < upper\_limit)  
   
 # Record the number of cells after QC  
 cells\_after <- ncol(T\_QC)  
   
 #save modified object  
 filename <- paste0("QC\_sample", sample\_id, ".rds")  
 saveRDS(T\_QC, file.path(out, filename))  
   
 # Append the cell count information to the data frame  
 cell\_counts <- rbind(cell\_counts, data.frame(sample = sample\_id,  
 cells\_before\_QC = cells\_before,  
 cells\_after\_QC = cells\_after))  
}  
  
# Save the cell counts data frame  
output\_dir <- "/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/"  
write.csv(cell\_counts, file = paste0(output\_dir, "cell\_counts.csv"), row.names = FALSE)  
# Convert stats to a data frame  
data\_frame\_stats <- data.frame(APOE\_group = names(stats),  
 cells\_before = sapply(stats, function(x) x$cells\_before),  
 cells\_after = sapply(stats, function(x) x$cells\_after),  
 genes\_before = sapply(stats, function(x) length(x$genes\_before)),  
 genes\_after = sapply(stats, function(x) length(x$genes\_after)))  
  
# Save the data frame  
write.csv(data\_frame\_stats, file = paste0(output\_dir, "APOE\_group\_stats.csv"), row.names = FALSE)

### Subsection 2: Read the new QC’d files, then merge into new object, “organoid1.rds”

Now that we have performed QC across all these samples, we likely have a lot of memory being used in our R cache. Let’s clear it and begin by reading in these new “QC\_sampleX”.rds objects, then proceed by merging them into a new composite rds object, named organoid1.

S1 <- readRDS('outputs/simpleafSeurat/QC\_sample1.rds')  
S2 <- readRDS('outputs/simpleafSeurat/QC\_sample2.rds')  
S3 <- readRDS('outputs/simpleafSeurat/QC\_sample3.rds')  
S4 <- readRDS('outputs/simpleafSeurat/QC\_sample4.rds')  
S6 <- readRDS('outputs/simpleafSeurat/QC\_sample6.rds')  
S8 <- readRDS('outputs/simpleafSeurat/QC\_sample8.rds')  
S9 <- readRDS('outputs/simpleafSeurat/QC\_sample9.rds')  
S10 <- readRDS('outputs/simpleafSeurat/QC\_sample10.rds')  
S11 <- readRDS('outputs/simpleafSeurat/QC\_sample11.rds')  
S12 <- readRDS('outputs/simpleafSeurat/QC\_sample12.rds')  
S13 <- readRDS('outputs/simpleafSeurat/QC\_sample13.rds')  
S14 <- readRDS('outputs/simpleafSeurat/QC\_sample14.rds')  
S15 <- readRDS('outputs/simpleafSeurat/QC\_sample15.rds')  
S16 <- readRDS('outputs/simpleafSeurat/QC\_sample16.rds')  
S17 <- readRDS('outputs/simpleafSeurat/QC\_sample17.rds')  
S18 <- readRDS('outputs/simpleafSeurat/QC\_sample18.rds')  
S19 <- readRDS('outputs/simpleafSeurat/QC\_sample19.rds')  
S20 <- readRDS('outputs/simpleafSeurat/QC\_sample20.rds')  
S21 <- readRDS('outputs/simpleafSeurat/QC\_sample21.rds')  
S22 <- readRDS('outputs/simpleafSeurat/QC\_sample22.rds')  
S23 <- readRDS('outputs/simpleafSeurat/QC\_sample23.rds')  
S24 <- readRDS('outputs/simpleafSeurat/QC\_sample24.rds')  
S25 <- readRDS('outputs/simpleafSeurat/QC\_sample25.rds')  
S26 <- readRDS('outputs/simpleafSeurat/QC\_sample26.rds')  
  
###\* At this point, let's just combine our QC'd samples into an .rds called \*###  
###\* organoid1.rds and save it \*###  
  
organoid1 <- merge(S1, y = c(S2, S3, S4, S6, S8, S9, S10,  
 S11, S12, S13, S14, S15, S16, S17, S18, S19,  
 S20, S21, S22, S23, S24, S25, S26),   
 add.cell.ids = c("S1", "S2", "S3", "S4", "S6", "S8", "S9", "S10",  
 "S11", "S12", "S13", "S14", "S15", "S16", "S17", "S18", "S19",  
 "S20", "S21", "S22", "S23", "S24", "S25", "S26"),  
 project = "APOE\_Jorganoid"  
)  
  
  
out <- '/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/'  
saveRDS(organoid1,file.path(out,'organoid1.rds'))

### Subsection 3: Visualizing Results of QC:

Let’s create the same Violin plots we did before, but now with the QC’d objects. Begin by checking our organoid1 object to just confirm that we do indeed have the genotype meta.data stored:

# Assuming organoid1 is your Seurat object and 'genotype' is a column in meta.data  
unique\_genotypes <- unique(organoid1@meta.data$genotype)  
  
# Output directory  
out <- "/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/images/" # Replace with your actual directory path  
  
# Define the features to plot  
features <- c("nFeature\_RNA", "nCount\_RNA", "percent.mt")  
  
# Loop through each genotype  
for (genotype in unique\_genotypes) {  
 # Subset Seurat object by genotype  
 organoid\_subset <- subset(organoid1, subset = genotype == genotype)  
   
 # Create violin plots  
 vps <- VlnPlot(object = organoid\_subset, features = c("nFeature\_RNA", "nCount\_RNA", "percent.mt"), ncol = 3, slot = "counts", pt.size = 0)  
   
 # Modify the plots to spread them out evenly  
 vps <- vps + plot\_layout(guides = 'collect') & theme(legend.position = 'none')  
   
 # Add the genotype as the overall figure title  
 combined\_plot <- vps + plot\_annotation(title = genotype, theme = theme(plot.title = element\_text(hjust = 0.5)))  
   
 # Construct the full path for the output file  
 output\_file\_path <- file.path(out, paste0("after\_QC\_violin\_plots\_", genotype, ".png"))  
   
 # Save the combined plot to the specified directory  
 ggsave(output\_file\_path, plot = combined\_plot, width = 12, height = 6, units = "in")  
}

## Step 3: Dimensionality Reduction and Further Analysis

Begin Step 3 by reading in the organoid1.rds quality-controlled object. Again, this is a good time to restart your R session.

organoid1 <- readRDS('/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/organoid1.rds')  
library(Seurat)  
library(miQC)  
  
###\* don't over-saturate your cache. Easier to rename all of these \*###  
# Assuming organoid1 is your Seurat object  
T\_QC <- organoid1  
  
# Preprocessing steps  
T\_QC <- NormalizeData(T\_QC, verbose = FALSE)  
T\_QC <- FindVariableFeatures(T\_QC, selection.method = "vst", nfeatures = 2000)  
T\_QC <- ScaleData(T\_QC, features = rownames(T\_QC), verbose = FALSE)  
  
# Dimensionality reduction  
T\_QC <- RunPCA(T\_QC, features = VariableFeatures(object = T\_QC), verbose = FALSE)  
T\_QC <- RunUMAP(T\_QC, dims = 1:50)  
  
# Clustering  
T\_QC <- FindNeighbors(T\_QC, dims = 1:50, k.param = 30, verbose = FALSE)  
T\_QC <- FindClusters(T\_QC, resolution = 0.6, verbose = FALSE)  
###\* Show in each cluster the genotype (extract from organoid2@meta.data$geno...\*###  
###\* Plot what is each genotype distribution in each cluster \*###  
###\* She was using findCluster, did not do actual cellType annotation here. \*###  
###\* When you do UMAP, you can color after by Seurat cluster (must run her other code first) \*###  
###\*   
###\* Bring questions for Deepti for the meeting tomorrow  
# Visualization  
DimPlot(T\_QC, reduction = "umap", group.by = "genotype")  
  
###\* this is a good point to save our .rds object, this time as organoid2 \*###  
saveRDS(T\_QC,file.path(out,'organoid2.rds'))  
  
# ribo\_counts <- as.data.frame(APOE22$percent.ribo)  
# colnames(ribo\_counts) <- "percent.ribo" # Rename the column without using dplyr  
organoid2 <- readRDS('/projectnb/tcwlab/LabMember/akg/APOE\_Jorganoid\_project/outputs/simpleafSeurat/organoid2.rds')  
organoid2 <- T\_QC  
  
#DimPlot(organoid2, reduction = "umap", group.by = "genotype")  
###\* Let's take a look at the percent.RNA of the cells in the data \*###  
RNA\_counts <- as.data.frame(organoid2$percent.RNA)  
colnames(RNA\_counts) <- "percent.RNA"  
# Now you can plot without renaming  
ggplot(RNA\_counts, aes(x = percent.RNA)) +   
 geom\_density() +  
 theme\_bw() +  
 ggtitle("Percent RNA genes per cell")  
  
# PCA Visualization  
DimPlot(organoid2, reduction = "pca")  
  
# Adjust UMAP dimensions  
organoid2 <- RunUMAP(organoid2, dims = 1:15)  
DimPlot(organoid2, reduction = "umap")  
  
# Clustering  
organoid2 <- FindNeighbors(organoid2, dims = 1:15)  
organoid2 <- FindClusters(organoid2)  
  
# Quality Control Feature Plots  
plot1 <- FeatureScatter(organoid2, feature1 = "nCount\_RNA", feature2 = "percent.mt")  
plot2 <- FeatureScatter(organoid2, feature1 = "nCount\_RNA", feature2 = "nFeature\_RNA")  
pdf(file = "qcfeatureplots\_filtered\_object.pdf", width = 15, height = 10)  
plot1 + plot2  
dev.off()