# BST282_HW5

Caitlin Cheung

2024-04-21

# BST282 Homework 5

Due April 21st 11:59pm on Canvas.

*Important*: you will save yourself many headaches by following the code in `/shared/courseSharedFolders/133853/HW5/env_setup.sh` to set up a custom mamba environment for this homework. The packages used in this homework frequently update, and new versions lead to conflicts between packages. If you follow the environment set up instructions, you shouldn't have these conflicts.

Download a copy of this `.Rmd` file, and keep it open in RStudio on your local computer. After you've run code in the mamba environment on the server, copy and paste the code into this `.Rmd` under the relevant question. Most of the code blocks are set with `eval=FALSE`. We will note a few exceptions– when you need to make plots, you can run that code locally in RStudio.

This homework consists of two parts: -In Part I, we will analyze a single cell RNA sequencing data from pancreatic tumor and adjacent normal samples. You can find details about the original study here: https://www.nature.com/articles/s43018-020-00121-4#Sec32 (https://www.nature.com/articles/s43018-020-00121-4#Sec32). -In Part II, we will analyze a single cell multiome (ATAC + gene expression) dataset to gain some intuition for working with multimodal data.

## Part I

### Question 1

Our first task for the pancreatic scRNAseq dataset is to create the gene expression matrix of raw UMI counts (genes are rows, cells are columns). The data have been deposited as separate count matrices ( `.mtx.gz` ) for each sample. Provide your code below to iterate through all samples in `/shared/courseSharedFolders/133853/HW5/data/GSE155698_data`, merging the per-sample count matrices into a single count matrix containing all 19 samples. -Column names should be cell barcodes. You may need to modify cell barcodes to ensure cell barcodes are unique across samples. -Row names should be genes (hugo symbol). You may need to collapse rows with identical gene names.

```
#!/bin/bash

# Set the directory containing the sample folders
sample_dir="/shared/home/cac8967/HW5/GSE155698_data"

# Iterate over each sample directory
for sample_path in "$sample_dir"/*/; do
    echo "Processing sample: $(basename "$sample_path")"

    sample_path="${sample_path%/}"

    # Unzip the barcode, features, and matrix files
    gunzip -f "$sample_path/filtered_feature_bc_matrix/barcodes.tsv.gz"
    gunzip -f "$sample_path/filtered_feature_bc_matrix/features.tsv.gz"
    gunzip -f "$sample_path/filtered_feature_bc_matrix/matrix.mtx.gz"
done
```

```
library(Matrix)
library(data.table)
library(purrr)

# Set the directory containing the sample folders
sample_dir <- "/shared/home/cac8967/HW5/GSE155698_data"
sample_dirs <- list.dirs(sample_dir, full.names = TRUE, recursive = FALSE)

count_matrices <- list()
for(sample_path in sample_dirs) {
  barcodes <- read.table(file.path(sample_path, "filtered_feature_bc_matrix", "barcodes.tsv"))
  features <- read.table(file.path(sample_path, "filtered_feature_bc_matrix", "features.tsv"))
  count_matrix <- readMM(file.path(sample_path, "filtered_feature_bc_matrix", "matrix.mtx"))
  sample_name <- basename(sample_path)
  modified_barcodes <- paste0(sample_name, ".", barcodes$V1)
  colnames(count_matrix) <- modified_barcodes
  rownames(count_matrix) <- features$V2
  count_matrix <- Matrix::Matrix(rowsum(as.matrix(count_matrix), group = row.names(count_matri
x)))
  count_matrices[[sample_name]] <- count_matrix
}


merged_matrix <- do.call(cbind, count_matrices)

dim(merged_matrix)

# saveRDS(merged_matrix, "merged.rds")
```

What are the dimensions of your final matrix?

```
32643 x 59473
```

## Question 2

Our next task is to create a cell *metadata* dataframe, which will have one row per cell. *Without using* `Seurat` , add these columns to your metadata dataframe: 1) cell barcode 2) sample name (e.g. “AdjNorm_TISSUE_1”, or “AdjNorm_TISSUE_2”) 3) total number of UMI counts 4) number of unique genes expressed 5) percent of UMIs originating from mitochondrial genes (6) doublet score, computed by Scrublet (7) whether or not Scrublet called the cell a doublet For (6) and (7), you can use our results from running Scrublet on these data: `/shared/courseSharedFolders/133853/HW5/GSE155698_scrublet_results.csv` .

```
library(Matrix)
library(dplyr)

scrublet <- read.csv("/shared/home/cac8967/HW5/GSE155698_scrublet_results.csv", sep = ",")
scrublet = scrublet[,-1]

metadata = data.frame(cell_barcode = colnames(merged_matrix))
metadata$sample = sub("\\..*", "", metadata$cell_barcode)
metadata = metadata |> mutate(cell_barcode = gsub("\\.", "_", cell_barcode))
metadata$nUMI = colSums(merged_matrix)
metadata$nGene = colSums(merged_matrix>0)
mt_genes = rownames(merged_matrix)[grepl("^MT-", rownames(merged_matrix))]
metadata$pct_MT = 100* colSums(merged_matrix[rownames(merged_matrix) %in% mt_genes,])/colSums(me
rged_matrix)
metadata <- merge(metadata, scrublet, by = "cell_barcode")

saveRDS(metadata, file = "metadata.rds")
```

## Question 3

*For this question, download your metadata file from the server, and run code on your local computer in RStudio (keep* `eval = TRUE` *)*

Next, we need to visualize the quality control metrics for these data and determine appropriate thresholds.

Produce two scatterplots. There should be one point per cell, total UMI count on the x axis, and unique gene count on the y axis. For the first scatterplot, color each cell by the percent of UMIs that are mitochondrial. For the second scatterplot, color each cell by its doublet score.

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(ggrastr)
```

```
## Warning: package 'ggrastr' was built under R version 4.3.3
```

```
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.3.2
```

```
library(viridis)
```

```
## Warning: package 'viridis' was built under R version 4.3.3
```

```
## Loading required package: viridisLite
```

```
library(tidyverse)
```

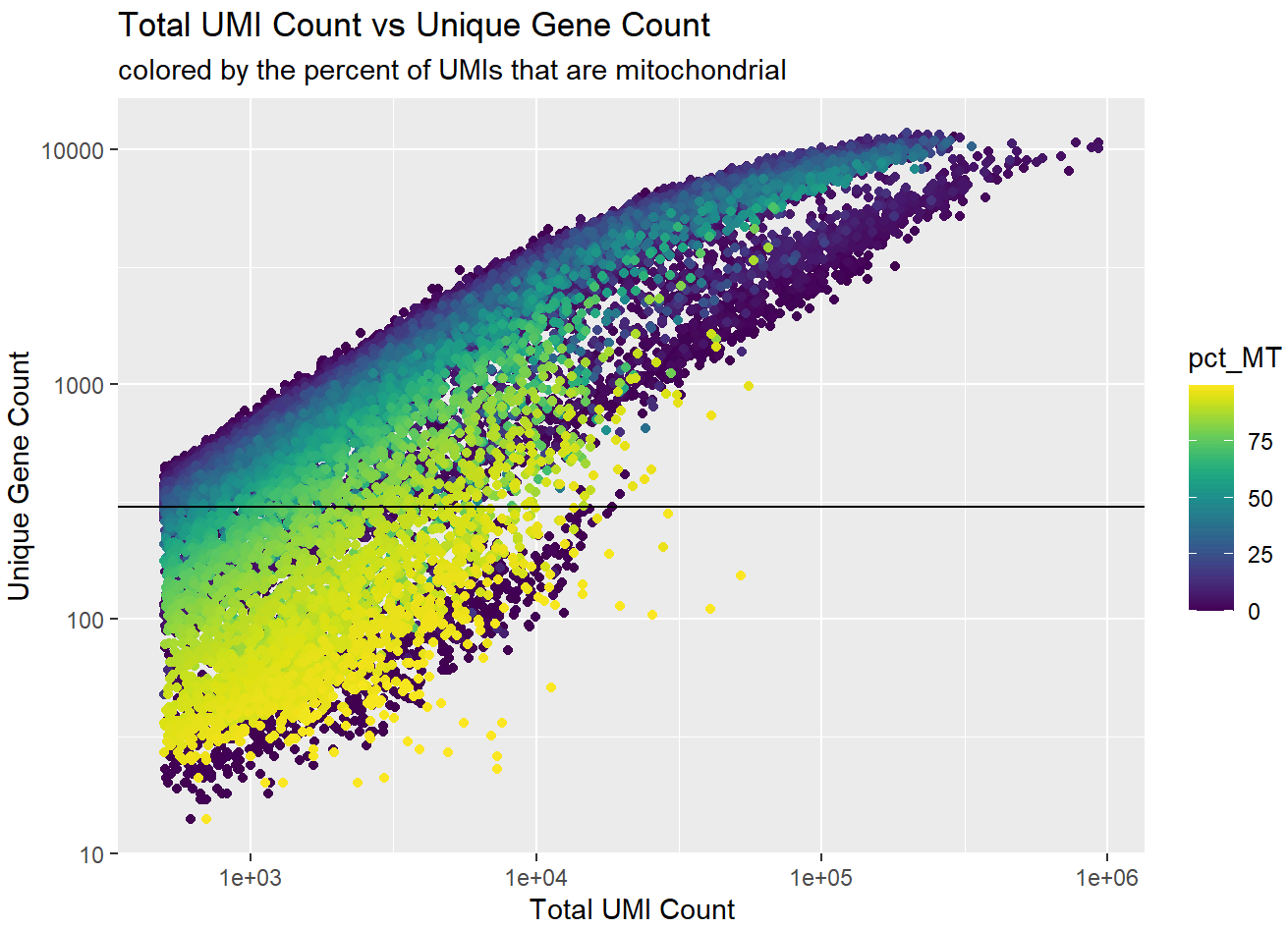
```
## Warning: package 'tidyverse' was built under R version 4.3.2
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

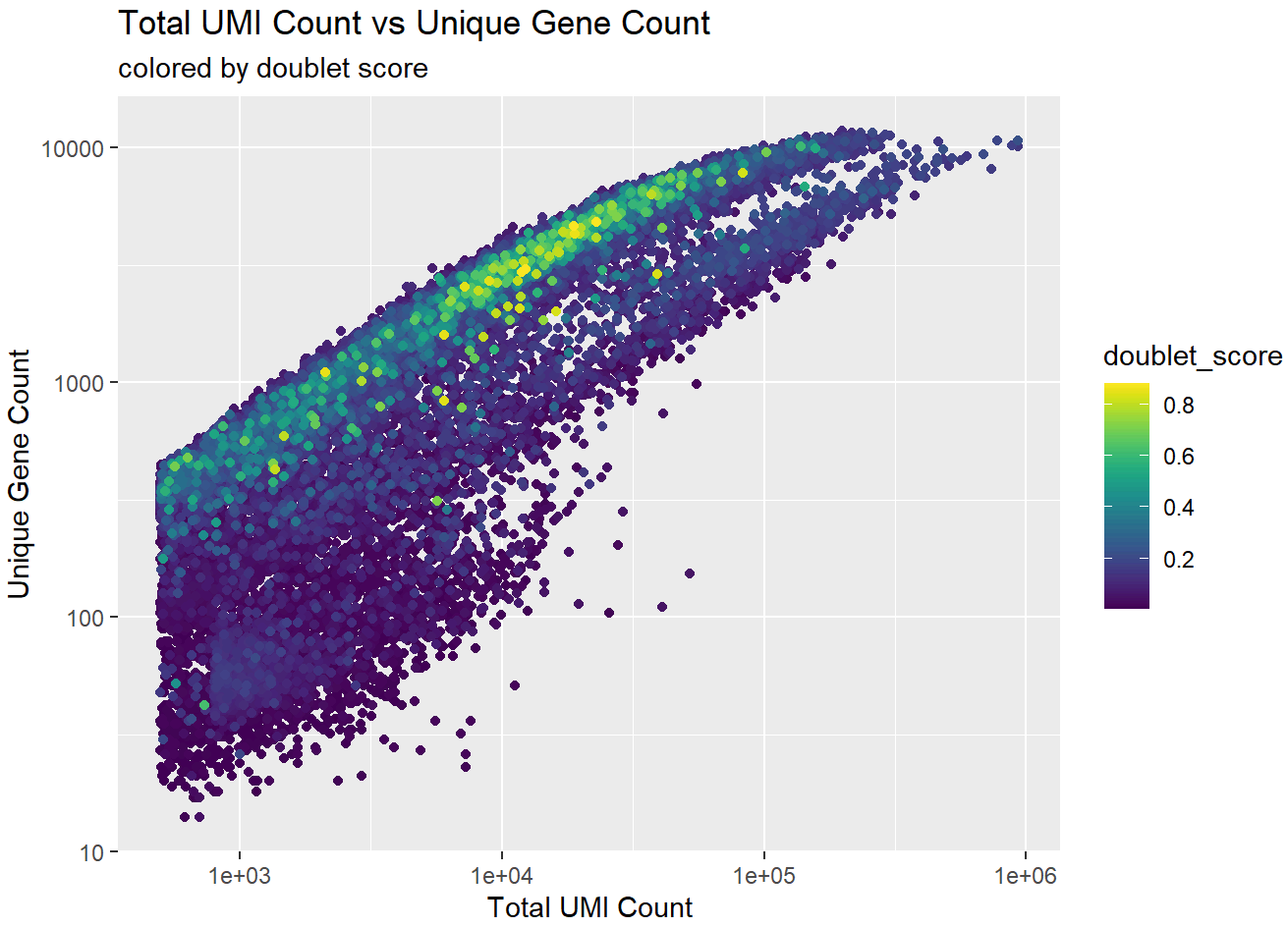
```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ lubridate 1.9.2     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.0
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
e errors
```

```
library(dplyr)

metadata = readRDS("metadata.rds")

# scatterplot 1
metadata[order(metadata$pct_MT),] |> ggplot(aes(x = nUMI, y = nGene, color = pct_MT)) +
  geom_point() +
  xlab("Total UMI Count") +
  ylab("Unique Gene Count") +
  labs(title = "Total UMI Count vs Unique Gene Count",
       subtitle = "colored by the percent of UMIs that are mitochondrial") +
  scale_color_viridis() +
  scale_y_log10() +
  scale_x_log10() +
  geom_hline(yintercept = 300)
```




```
# scatterplot 2
metadata[order(metadata$doublet_score),] |> ggplot(aes(x = nUMI, y = nGene, color = doublet_scor
e)) +
  geom_point() +
  xlab("Total UMI Count") +
  ylab("Unique Gene Count") +
  labs(title = "Total UMI Count vs Unique Gene Count",
       subtitle = "colored by doublet score") +
  scale_color_viridis() +
  scale_y_log10() +
  scale_x_log10()
```

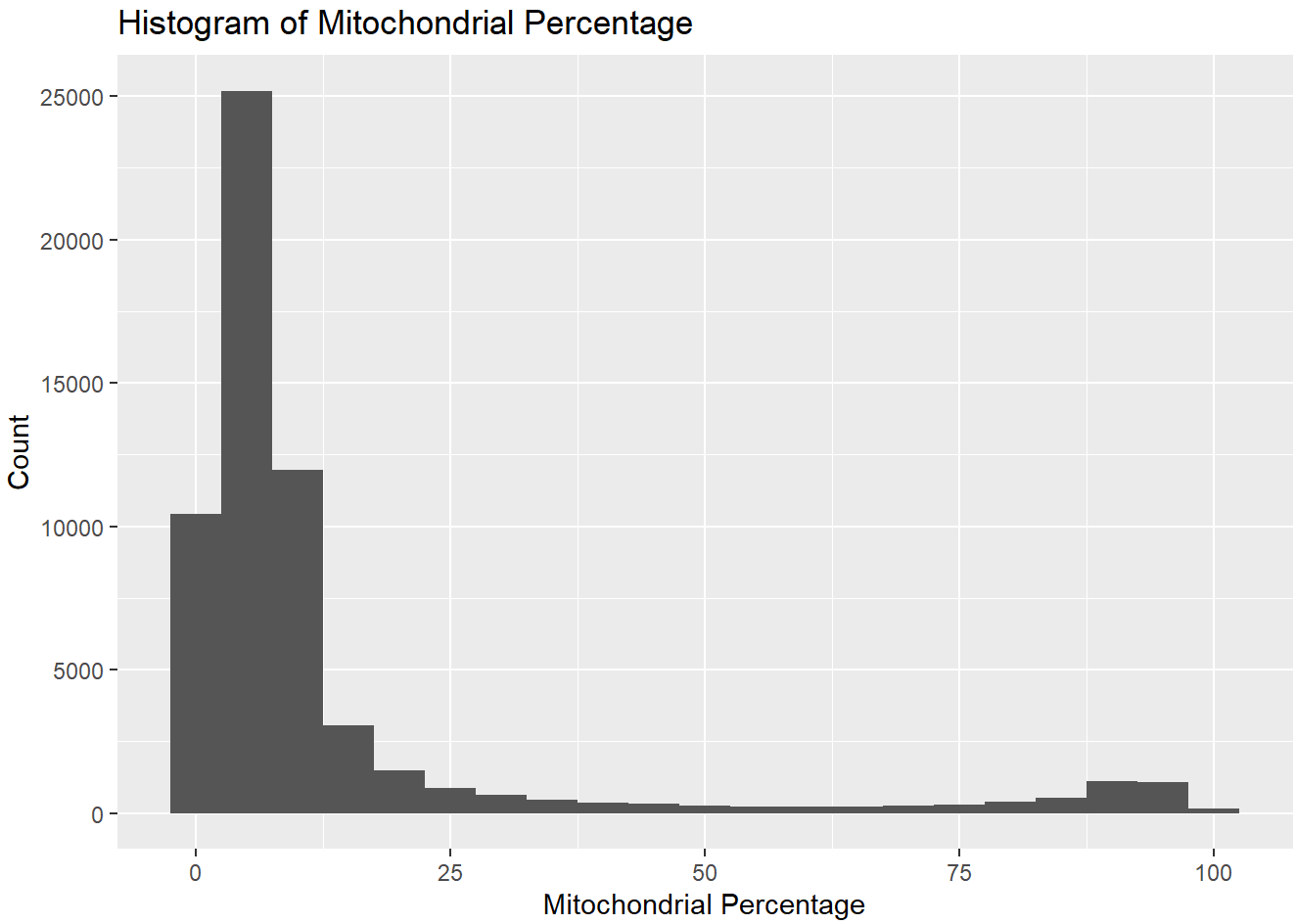



Provide a one-sentence explanation for the observed relationship between doublet score, nUMI, and nGene.

```
The higher relative number of genes and mRNA molecules detected in identified doublets may stem
from their formation through the accidental encapsulation of two cells, thereby doubling the gen
etic material present.
```

Visualize the distribution of mitochondrial percentage with a histogram.

```
metadata |> ggplot(aes(x = pct_MT)) +
  geom_histogram(binwidth = 5) +
  xlab("Mitochondrial Percentage") +
  ylab("Count") +
  labs(title = "Histogram of Mitochondrial Percentage")
```




Based on your plots, choose an appropriate threshold for mitochondrial percentage and for number of unique genes expressed. Apply these filters, as well as a filter to remove cells called doublets by Scrublet. Make sure that the column names of your expression matrix continue to match the row names of your metadata dataframe.

Thresholds: pct_MT < 20 and nGene > 300

```
QCd_cells = metadata$cell_barcode[metadata$pct_MT<20 & metadata$nGene>=300 & metadata$called_dou
blet==0]

merged_matrix = t(merged_matrix)
rownames(merged_matrix) = gsub("\\.", "_", rownames(merged_matrix))
rownames(metadata) = metadata$cell_barcode
table(rownames(merged_matrix)==metadata$cell_barcode)

merged_matrix = merged_matrix[QCd_cells,]
metadata = metadata[QCd_cells,]
```

## Question 4

Now, we need to identify variable genes using the variance stabilizing transformation (VST). Use the `vargenes_vst()` from R package symphony to accomplish this. Collect the *union* of the top 500 variable genes per sample.

```
library(symphony)

vargenes <- vargenes_vst(t(merged_matrix), group = metadata$sample, topn = 500)
```

Why is it important to use the variance stabilizing transformation here? Provide a brief (3 sentences or less) explanation that touches on the statistical distribution of gene counts. You may make a plot to accompany your explanation, but this is not necessary.

```
Using the variance stabilizing transformation (VST) is crucial because while count distributions
in single-cell RNA sequencing data are almost Poisson, where the mean is often approximately equ
al to the variance, this leads to a correlation between variance and mean. Without VST, selectin
g genes based solely on high variance would inadvertently bias towards genes with higher means,
skewing downstream analysis. VST resolves this issue by stabilizing the variance relative to the
mean, allowing for the accurate identification of 'variable' genes based on their true biologica
l variability.
```

## Question 5

For dimensionality reduction, we need to start thinking about genes as features and cells as observations. In other words, we need to transpose the gene expression matrix. Transpose the gene expression matrix, and then apply log(CP10K + 1) normalization to these data, *without using Seurat.* Hint: you will likely run into memory issues if you try to transpose a dense matrix of this size. We recommend keeping the matrix in sparse format, and using the `t_shallow()` function from R package MatrixExtra. (Why does this use so much less memory? Worth thinking about. No need to answer.)

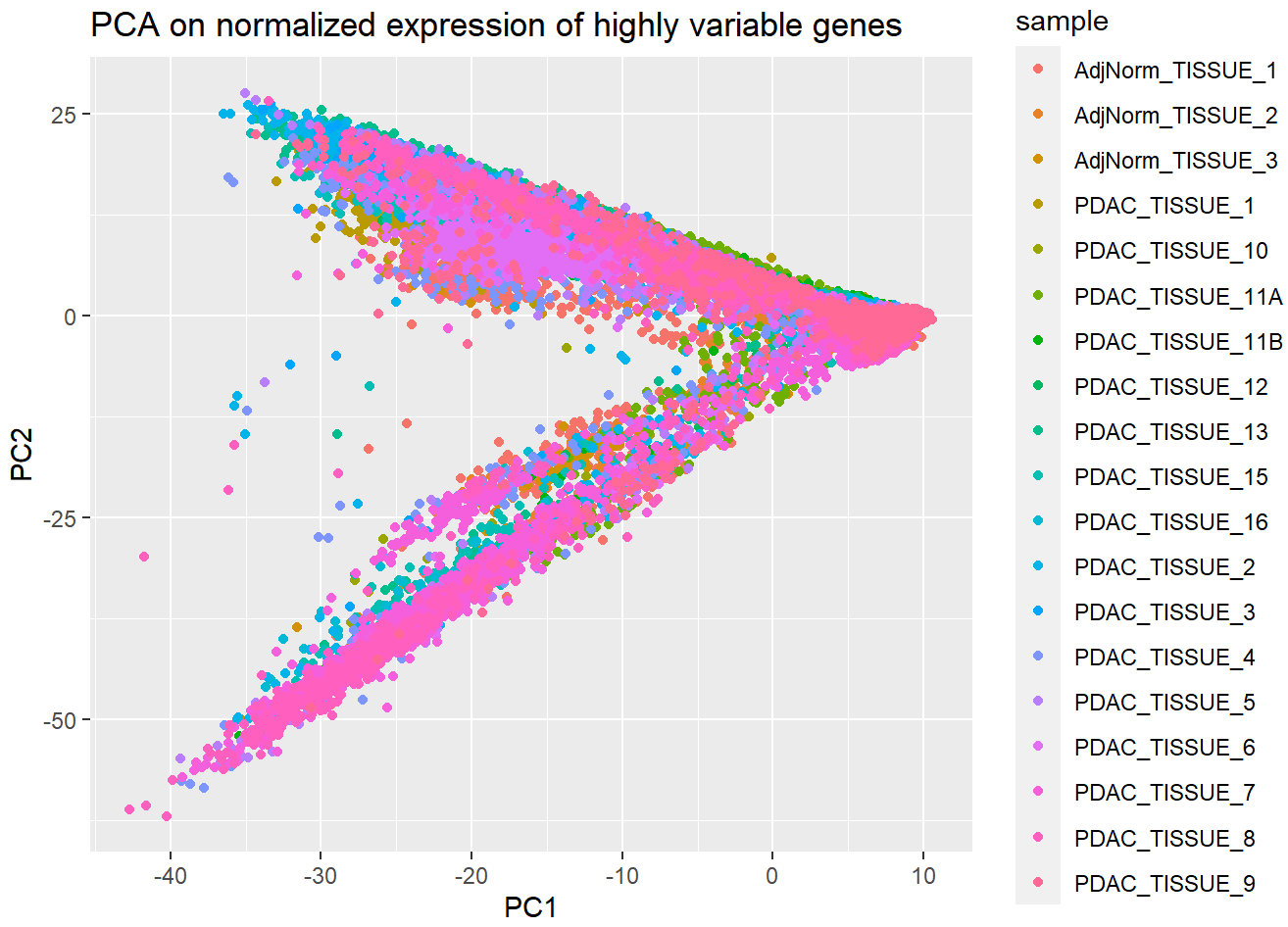
```
scaling_factor = 1e4 ##CP10K
Mnorm = log(1 + scaling_factor*merged_matrix/rowSums(merged_matrix))
Mscale = scale(Mnorm)
```

## Question 6

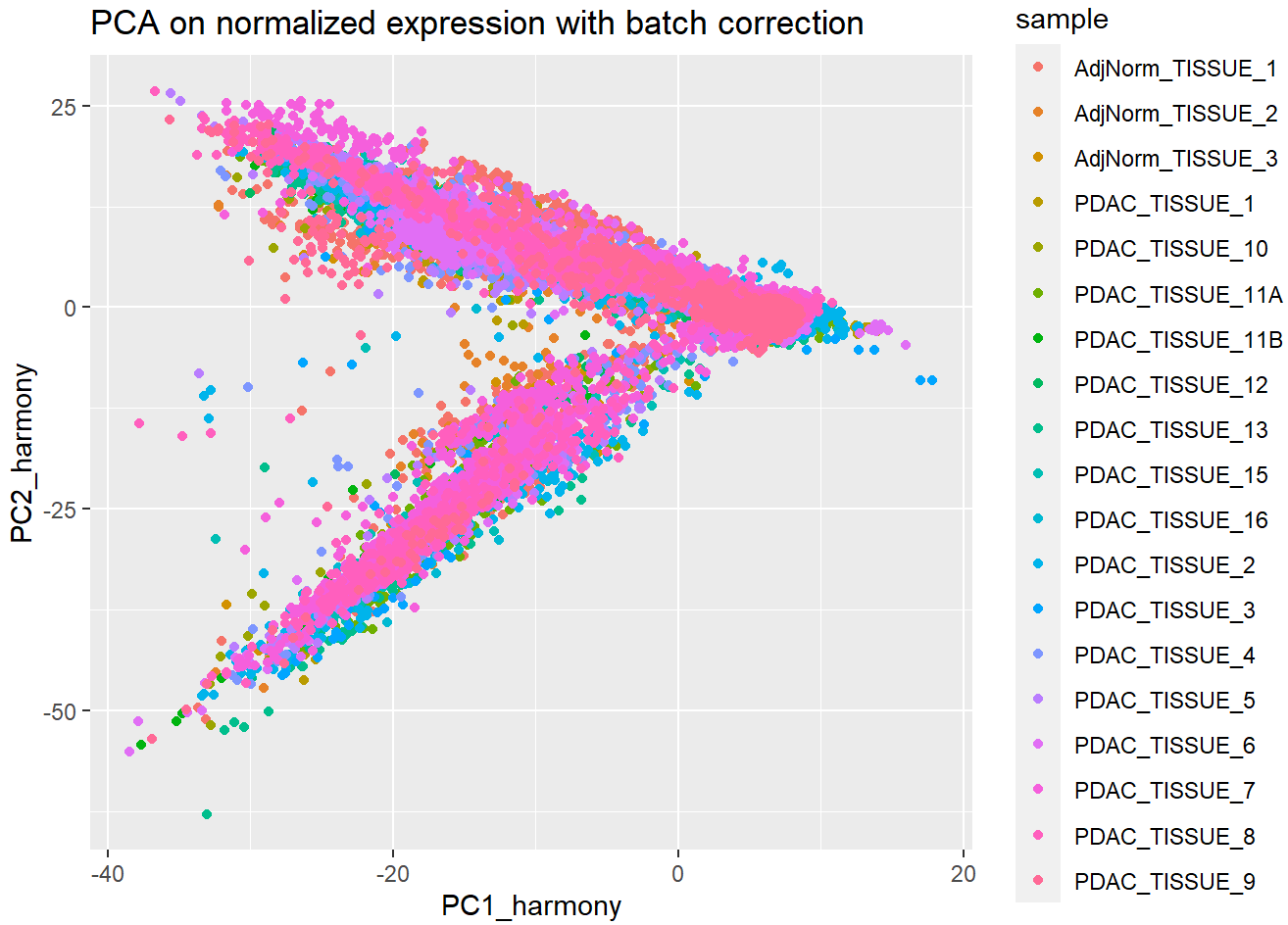
Now, run PCA on the normalized expression of your highly variable genes identified in Question 4. Then, use the function `harmony::RunHarmony()` to apply batch correction to the PC scores. *Hint: *DON'T* call `library(MatrixExtra)` in the same `.R` file that calls `RunHarmony()`. These packages don't work together smoothly.

```
library(Matrix)
library(harmony)

metadata_projections <- metadata

pca.res = prcomp(Mscale[,colnames(Mscale) %in% vargenes])
metadata_projections$PC1 = pca.res$x[,1]
metadata_projections$PC2 = pca.res$x[,2]

PC_scores <- pca.res$x


harmony_res <- harmony::RunHarmony(data = PC_scores, meta_data = metadata, batch = 'sample', var
s_use = c('sample'))

metadata_projections$PC1_harmony <- harmony_res[,"PC1"]
metadata_projections$PC2_harmony <- harmony_res[,"PC2"]

saveRDS(metadata_projections, "metadata_projections.rds")
```

```
metadata_projections = readRDS("metadata_projections.rds")

metadata_projections |> ggplot(aes(PC1, PC2, color=sample)) +
  geom_point() +
  labs(title = "PCA on normalized expression of highly variable genes")
```




```
metadata_projections |> ggplot(aes(PC1_harmony, PC2_harmony, color = sample)) +
  geom_point() +
  labs(title = "PCA on normalized expression with batch correction")
```




From the `harmony::RunHarmony()` function, we can see that Harmony carries out batch correction in PCA space. What is the main limitation of this strategy and what are the advantages?

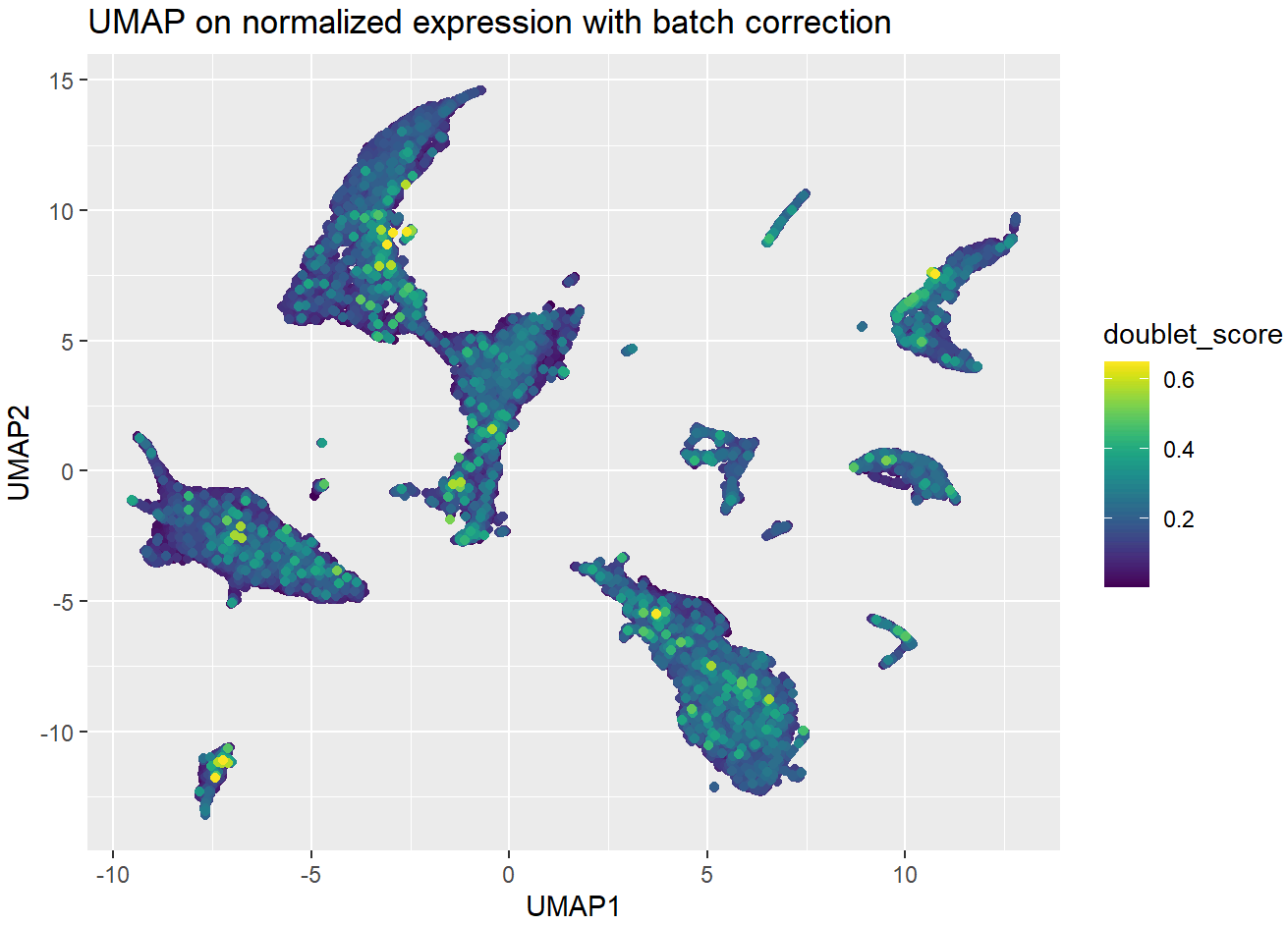
```
The main limitation of batch correction in PCA space by Harmony is that it assumes that the sour
ces of batch effects are linear and captured by the top PCs. This assumption may not hold true i
f batch effects are non-linear or if they affect dimensions beyond those captured by PCA, potent
ially leading to incomplete correction or even overcorrection. However, the advantages of batch
correction in PCA space include computational efficiency and simplicity. PCA reduces the dimensi
onality of the data while retaining most of the variance, making it computationally tractable fo
r large datasets. Additionally, PCA-based batch correction can effectively mitigate batch effect
s when they are linear and well-captured by the top principal components, providing a straightfo
rward approach for harmonizing datasets from multiple sources or batches.
```

# Question 7

Now, use `uwot::umap()` to project your first 10 batch-corrected principal components into two dimensions. Visualize the results in a scatterplot, coloring each cell by its doublet score.

```
library(uwot)
umap_result <- uwot::umap(harmony_res[,1:10])

metadata_projections$UMAP1 <- umap_result[,1]
metadata_projections$UMAP2 <- umap_result[,2]
```

```
metadata_projections[order(metadata_projections$doublet_score),] |> ggplot(aes(UMAP1, UMAP2, col
or = doublet_score)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction") +
  scale_color_viridis()
```

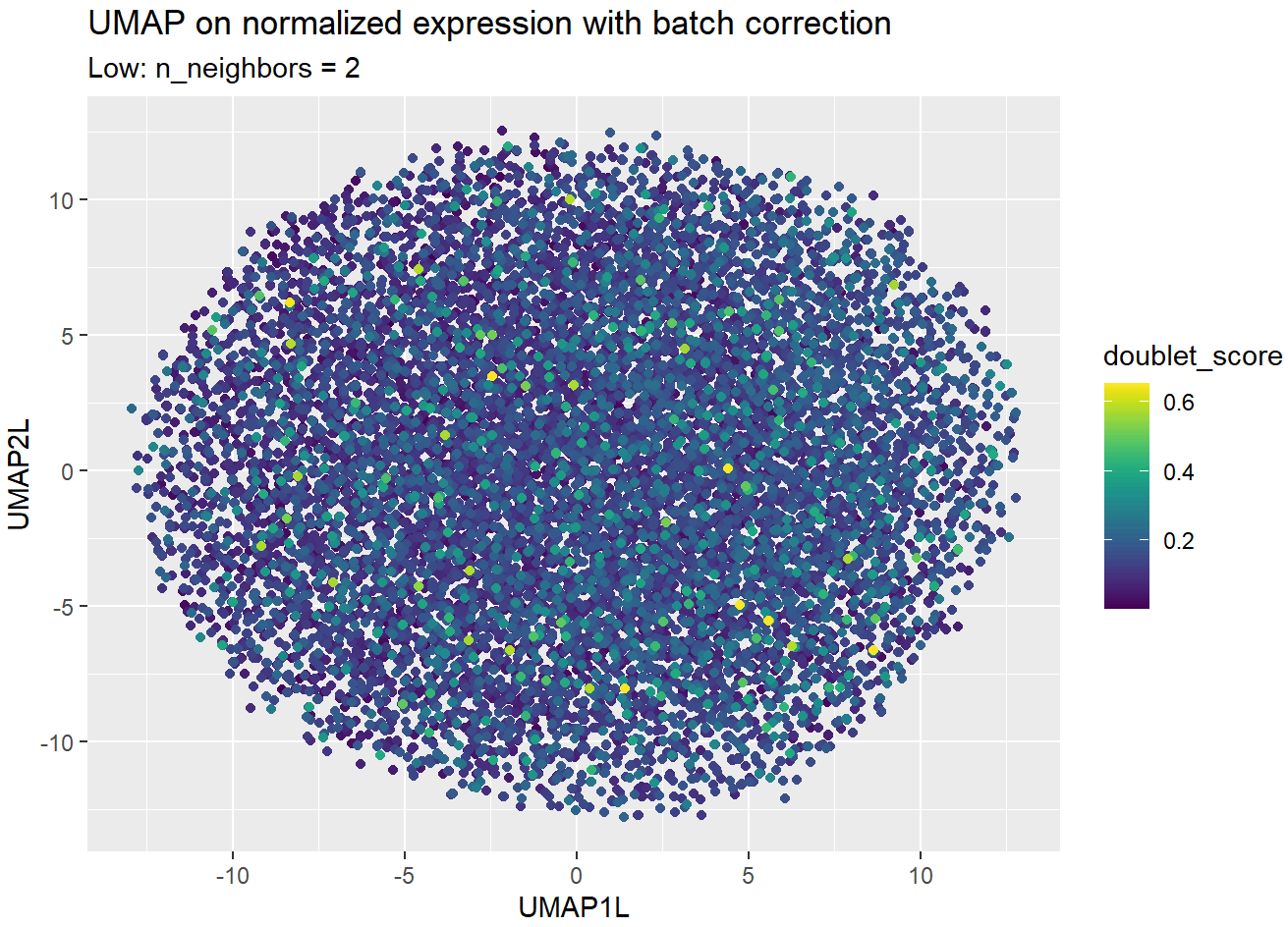



## Question 8

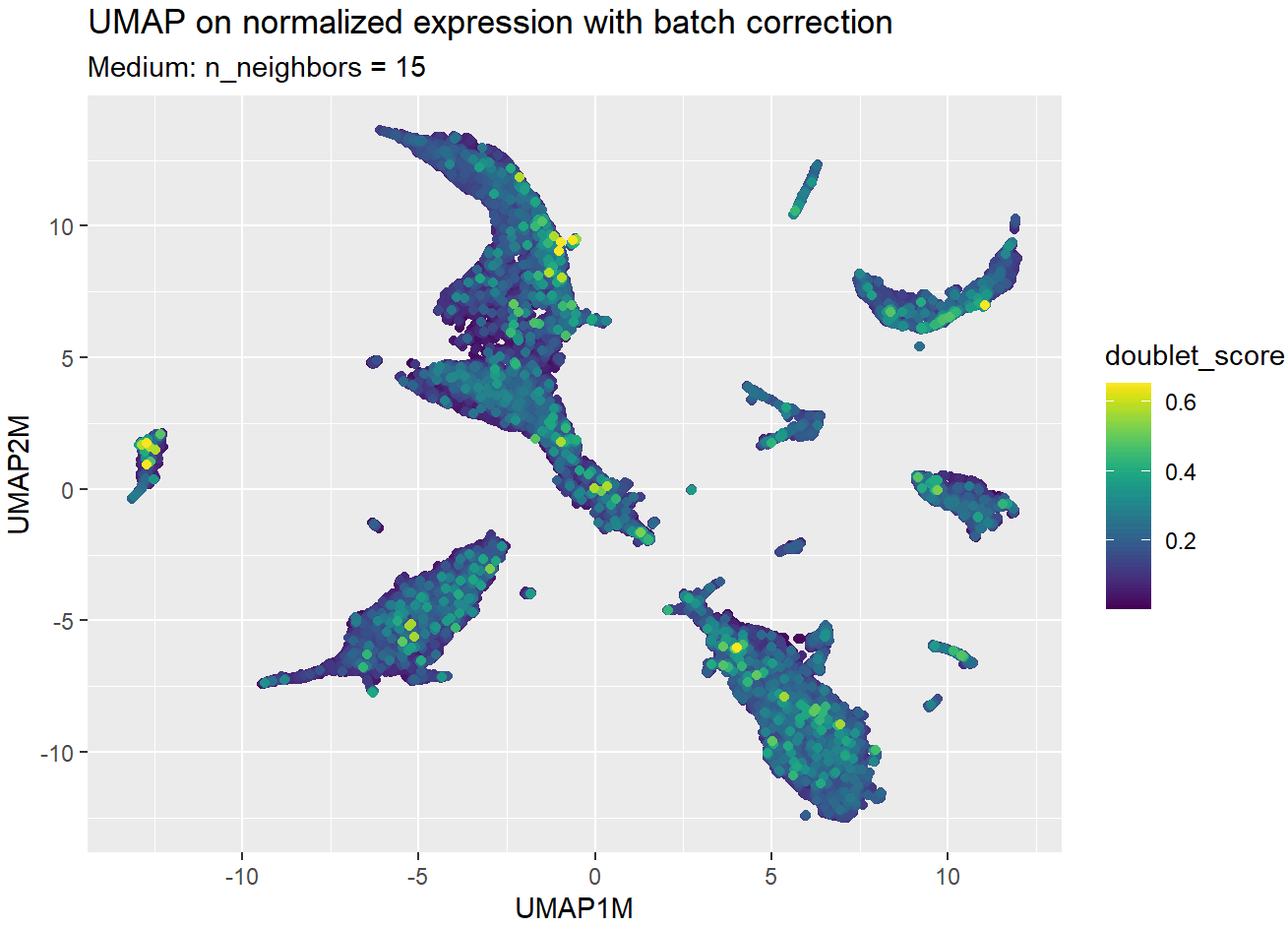
Check out the documentation (https://rdrr.io/cran/uwot/man/umap.html (https://rdrr.io/cran/uwot/man/umap.html)) for `uwot::umap()`, and pick a parameter to explore. Recreate your UMAP from above several times– with low, medium, and high values of this parameter. (Run `umap()` on the server, but plot your results locally)

```
#code to run UMAP with new parameters

umap_low <- uwot::umap(harmony_res[,1:10], n_neighbors = 2)
umap_med <- uwot::umap(harmony_res[,1:10], n_neighbors = 15)
umap_high <- uwot::umap(harmony_res[,1:10], n_neighbors = 100)

metadata_projections$UMAP1L <- umap_low[,1]
metadata_projections$UMAP2L <- umap_low[,2]
metadata_projections$UMAP1M <- umap_med[,1]
metadata_projections$UMAP2M <- umap_med[,2]
metadata_projections$UMAP1H <- umap_high[,1]
metadata_projections$UMAP2H <- umap_high[,2]

saveRDS(metadata_projections, file = "metadata_projections.rds")
```

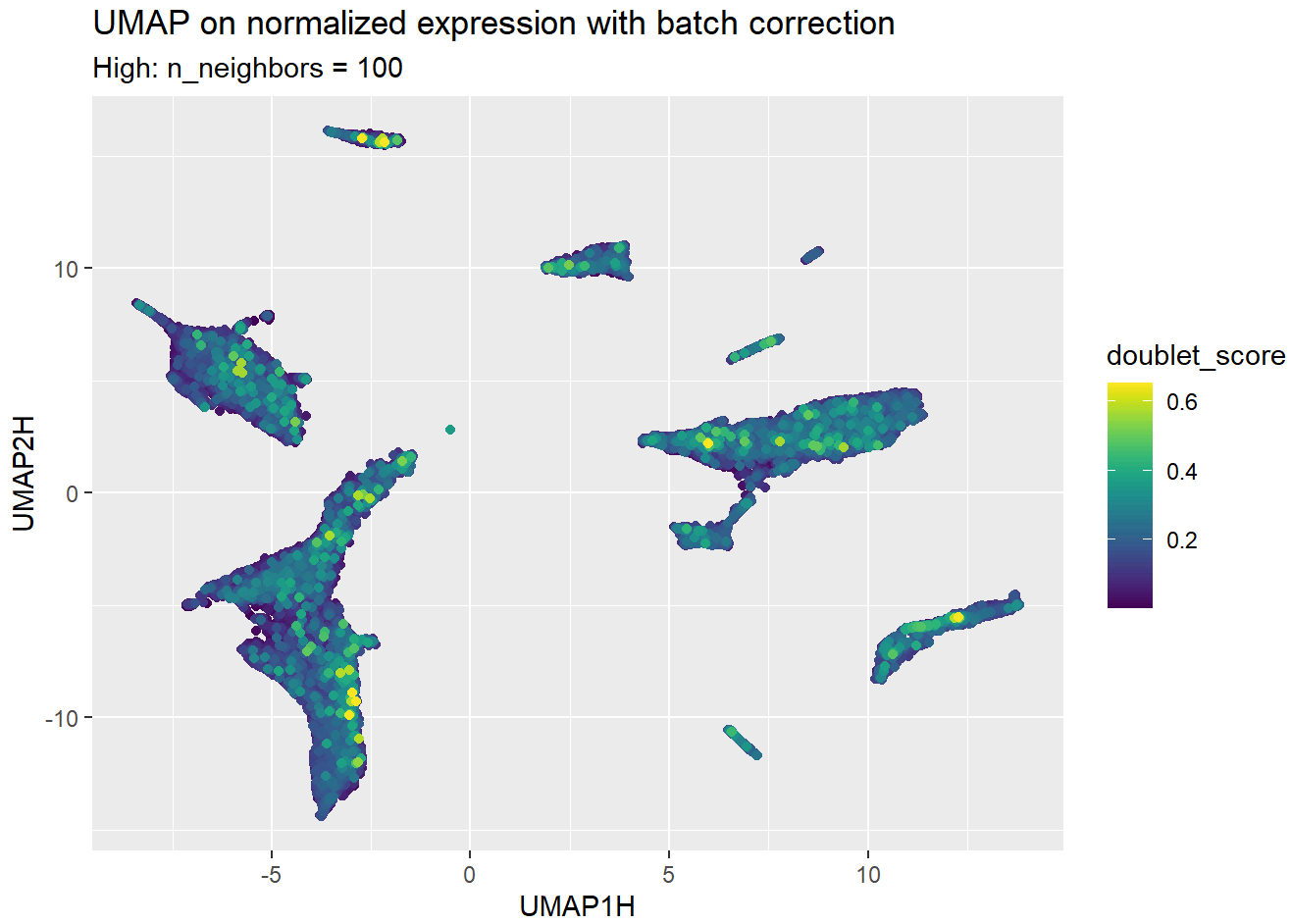
```
metadata_projections[order(metadata_projections$doublet_score),] |>
  ggplot(aes(UMAP1L, UMAP2L, color = doublet_score)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "Low: n_neighbors = 2") +
  scale_color_viridis()
```



```
metadata_projections[order(metadata_projections$doublet_score),] |>
  ggplot(aes(UMAP1M, UMAP2M, color = doublet_score)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "Medium: n_neighbors = 15") +
  scale_color_viridis()
```




```
metadata_projections[order(metadata_projections$doublet_score),] |>
  ggplot(aes(UMAP1H, UMAP2H, color = doublet_score)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "High: n_neighbors = 100") +
  scale_color_viridis()
```




How sensitive are cell embeddings to this parameter? Can you speculate as to why these changes are happening? Would you consider the flexibility of UMAP to be a weakness or a strength? Answer all of the above in 5 sentences or less.

```
Cell embeddings are rather sensitive to the n_neighbors parameter in UMAP, as it influences the
local structure captured in the resulting embeddings. Lower values seem to lead to more emphasis
on local relationships while higher values seem to emphasize global structure. These changes lik
ely occur because varying n_neighbors alters the balance between local and global information us
ed to construct the UMAP graph. UMAP's flexibility to adjust the number of neighbors is both a s
trength and a weakness as it allows for exploration of different scales of structure in the dat
a, but it also requires careful parameter selection to avoid biasing the interpretation of the r
esults and making the results more reproducible.
```

## Question 9

Fill in the lines of code below to: 1) create a Seurat object out of your results 2) apply Louvain clustering to these data

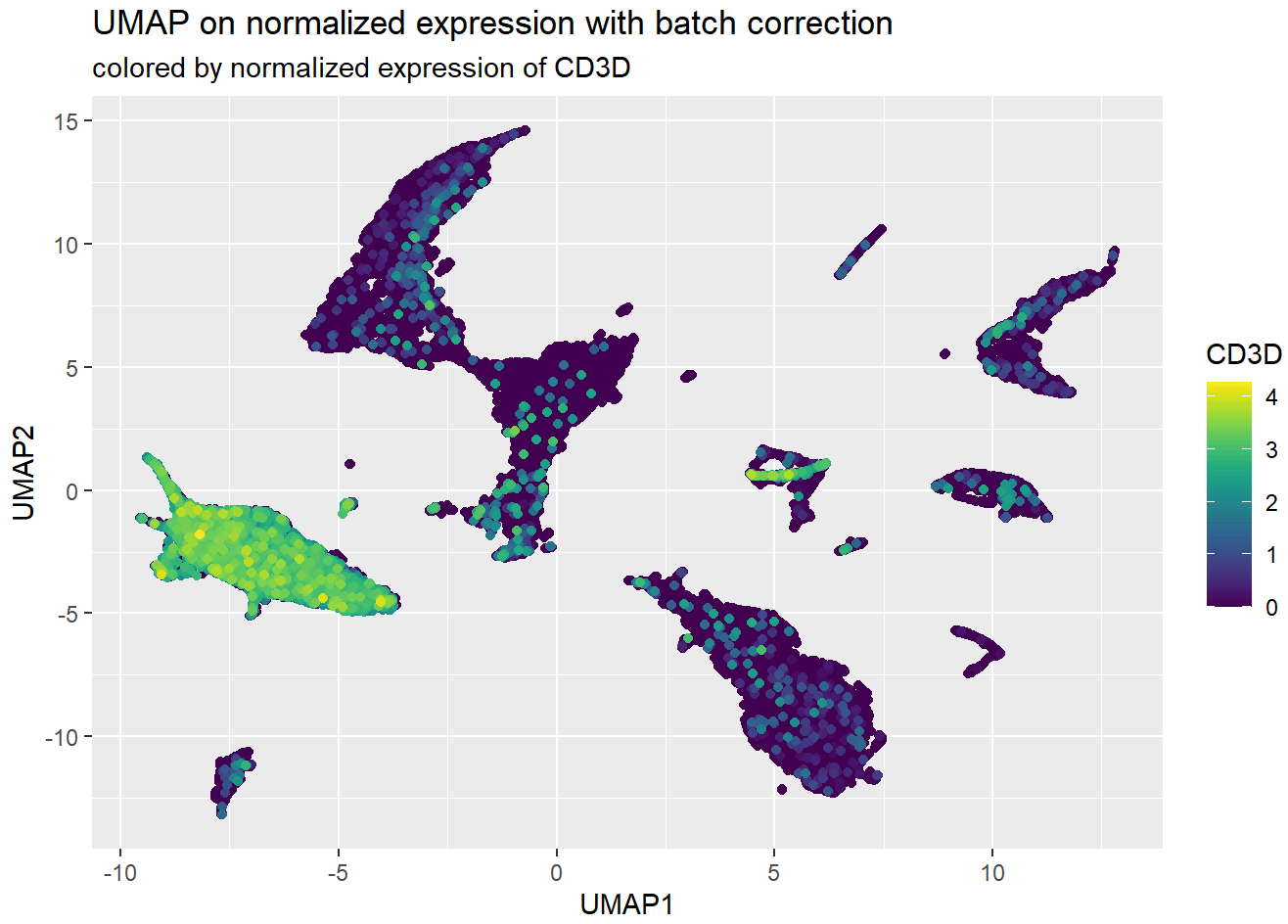
```
library(Seurat)

merged_matrix <- t(merged_matrix)

sobj = CreateSeuratObject(counts = merged_matrix, meta.data = metadata)
dr <- CreateDimReducObject(key = "pca", embeddings = pca.res$x, loadings = pca.res$rotation, ass
ay = "RNA")
sobj[['pca']] <- dr
sobj <- FindNeighbors(object = sobj, dims = 1:20, reduction = "pca")
sobj <- FindClusters(object = sobj, resolution = 2, cluster.name = "clusters")
```

## Question 10

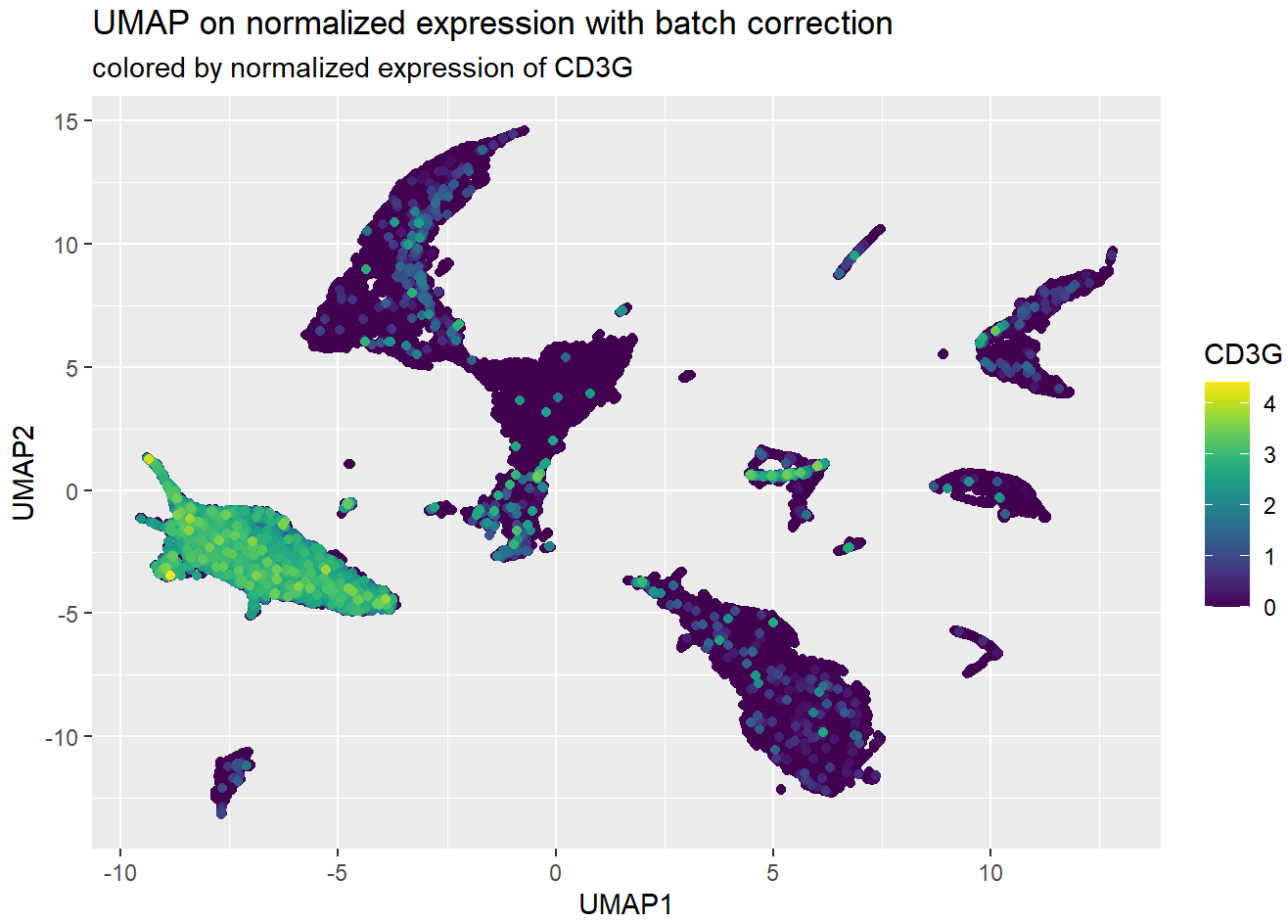
*For this question, download your metadata file from the server, and run code on your local computer in RStudio (keep `eval = TRUE`)* We'd like to identify which cells are T cells. T cells can be identified with marker genes *CD3D* and *CD3G*. In a UMAP scatterplot, color each cell by its [log(CP10K + 1)] normalized expression of *CD3D*. Repeat this for *CD3G*. Compare these to a UMAP scatterplot with each cell colored by its cluster assignment from Question 9. Can you identify which cluster(s) correspond to T cells?

```
CD3D = Mnorm[,"CD3D"]
CD3G = Mnorm[,"CD3G"]

metadata_projections$CD3D = CD3D
metadata_projections$CD3G = CD3G

# Get cluster assignments
#cluster_assignments <- sobj@meta.data$seurat_clusters
cluster_assignments <- Idents(object = sobj)
metadata_projections$cluster <- cluster_assignments

saveRDS(metadata_projections, "metadata_projections.rds")
```

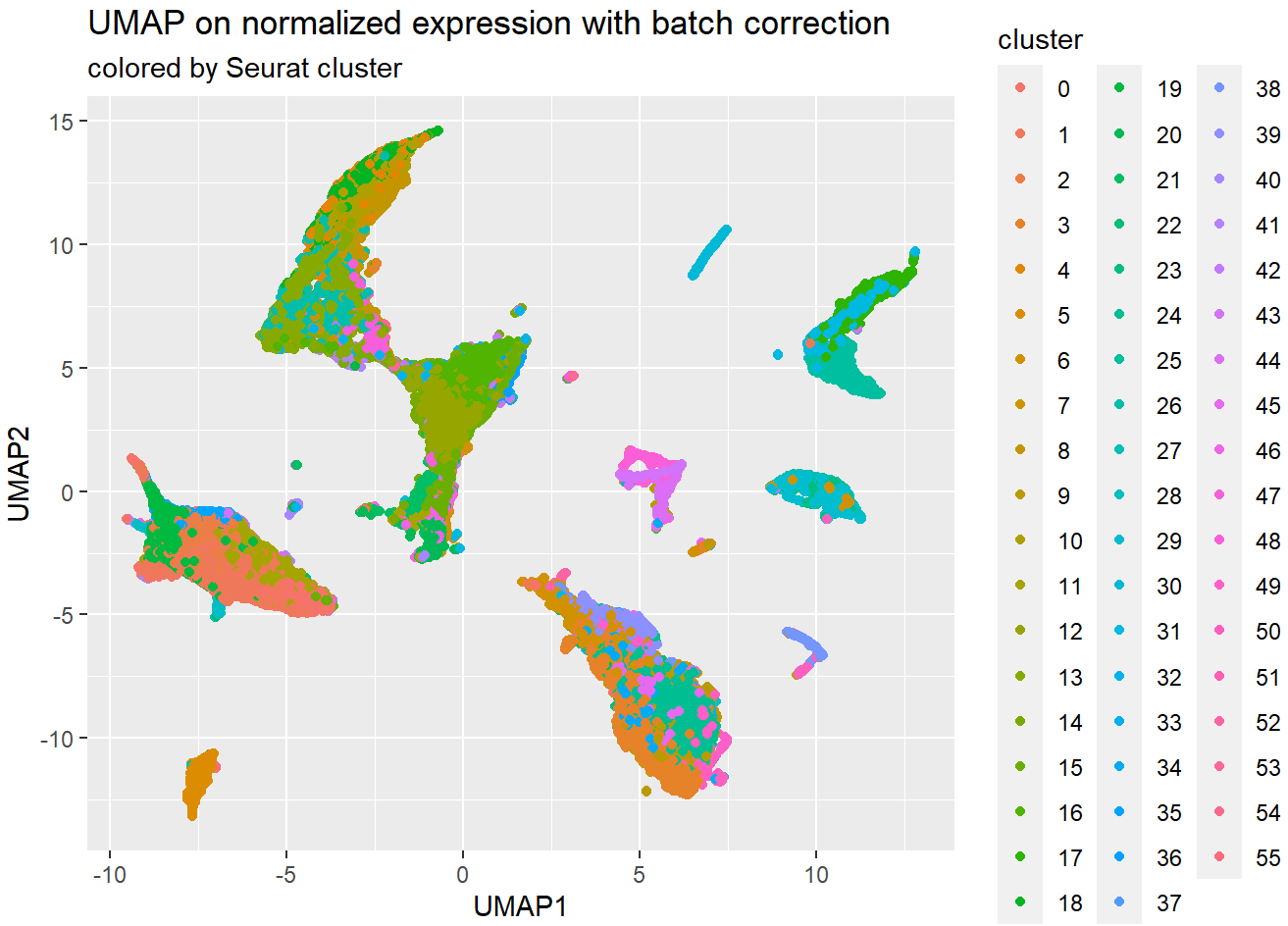
```
metadata_projections = readRDS("metadata_projections.rds")

#metadata_projections <- metadata_projections |>
#  filter(cluster %in% c("0", "1", "2", "11", "19", "28", "35", "40"))

metadata_projections[order(metadata_projections$CD3D),] |>
  ggplot(aes(x = UMAP1, y = UMAP2, color = CD3D)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "colored by normalized expression of CD3D") +
  scale_color_viridis()
```




```
metadata_projections[order(metadata_projections$CD3G),] |>
  ggplot(aes(x = UMAP1, y = UMAP2, color = CD3G)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "colored by normalized expression of CD3G") +
  scale_color_viridis()
```




```
metadata_projections |>
  ggplot(aes(x = UMAP1, y = UMAP2, color = cluster)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "colored by Seurat cluster")
```




```
The UMAP cluster made up of clusters 0, 1, 2, 11, 19, 28, 35, and 40 have the largest normalized expression values of CD3D and CD3G, and therefore are likely to correspond to T-cells.
```

## Question 11

Now that we've identified which cells are T cells, let's find subclusters of T cells. Subset the data to T cells, and repeat variable gene selection, PCA, batch correction, UMAP, and clustering *within* the T cells. Copy and paste your code below.

```
library(Seurat)
library(harmony)
library(symphony)
library(uwot)

clusters <- c("0", "1", "2", "11", "19", "28", "35", "40")
metadata_projections = readRDS("metadata_projections.rds")
metadata_Tcells <- metadata_projections[metadata_projections$cluster %in% clusters,1:7]

merged_matrix <- readRDS("merged.rds")
colnames(merged_matrix) = gsub("\\.", "_", colnames(merged_matrix))
Tcells_matrix <- merged_matrix[,rownames(metadata_Tcells)]

# variable gene selection
vargenes <- vargenes_vst(Tcells_matrix, group = metadata_Tcells$sample, topn = 500)



Tcells_matrix <- t(Tcells_matrix)
scaling_factor = 1e4 ##CP10K
Mnorm = log(1 + scaling_factor*Tcells_matrix/rowSums(Tcells_matrix))
Mscale = scale(Mnorm)

metadata_projections_Tcells <- metadata_Tcells
# PCA
pca.res = prcomp(Mscale[,colnames(Mscale) %in% vargenes])
metadata_projections_Tcells$PC1 = pca.res$x[,1]
metadata_projections_Tcells$PC2 = pca.res$x[,2]

# batch correction
PC_scores <- pca.res$x
harmony_res <- harmony::RunHarmony(data = PC_scores, meta_data = metadata_Tcells, batch = 'sampl
e', vars_use = c('sample'))
metadata_projections_Tcells$PC1_harmony <- harmony_res[,"PC1"]
metadata_projections_Tcells$PC2_harmony <- harmony_res[,"PC2"]

# UMAP
umap_result <- uwot::umap(harmony_res[,1:10])
metadata_projections_Tcells$UMAP1 <- umap_result[,1]
metadata_projections_Tcells$UMAP2 <- umap_result[,2]

# Clustering
Tcells_matrix <- t(Tcells_matrix)
sobj = CreateSeuratObject(counts = Tcells_matrix, meta.data = metadata_Tcells)
dr <- CreateDimReducObject(key = "pca", embeddings = pca.res$x, loadings = pca.res$rotation, ass
ay = "RNA")
sobj[['pca']] <- dr
sobj <- FindNeighbors(object = sobj, dims = 1:20, reduction = "pca")
sobj <- FindClusters(object = sobj, resolution = 2, cluster.name = "clusters")
cluster_assignments <- Idents(object = sobj)
metadata_projections_Tcells$cluster <- cluster_assignments

# TIGIT
```

```
TIGIT = Mnorm[,"TIGIT"]
metadata_projections_Tcells$TIGIT = TIGIT

saveRDS(metadata_projections_Tcells, "metadata_projections_Tcells.rds")
```

Your collaborator is skeptical about re-running these steps. To identify subclusters of T cells, couldn't you just cluster the full dataset at higher resolution?
In three sentences or less, what is the advantage of re-selecting variable genes and re-doing dimensionality reduction within T cells?

```
Re-selecting variable genes and re-doing dimensionality reduction within T cells allows for the
identification of gene expression patterns specific to T cells, potentially revealing finer-scal
e heterogeneity within this cell type. This approach can lead to more accurate characterization
of T cell subclusters by focusing on features that are most relevant and informative within the
T cell population, avoiding potential confounding effects from other cell types present in the d
ataset.
```

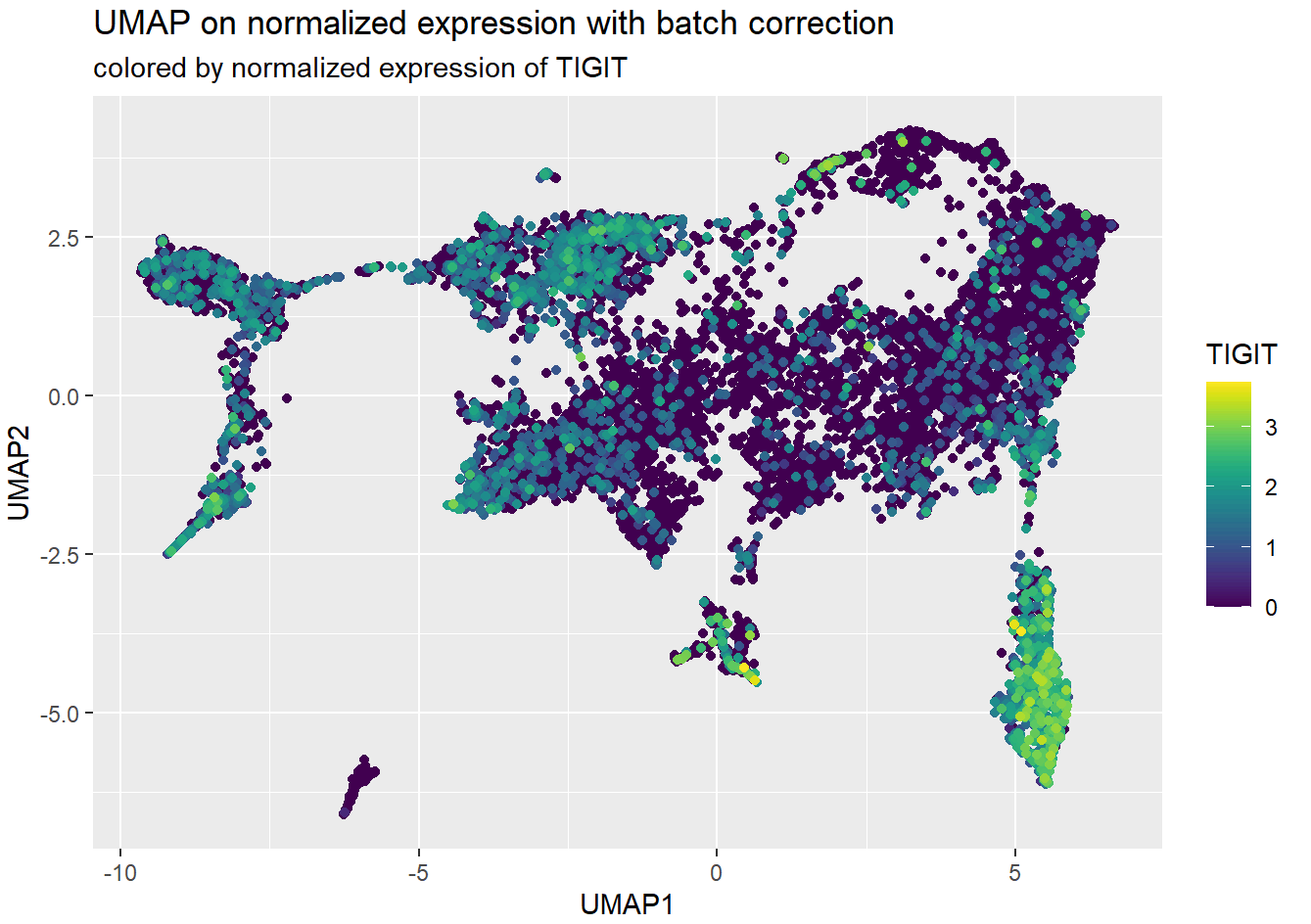
## Question 12

*For this question, download your metadata file from the server, and run code on your local computer in RStudio (keep `eval = TRUE` )*

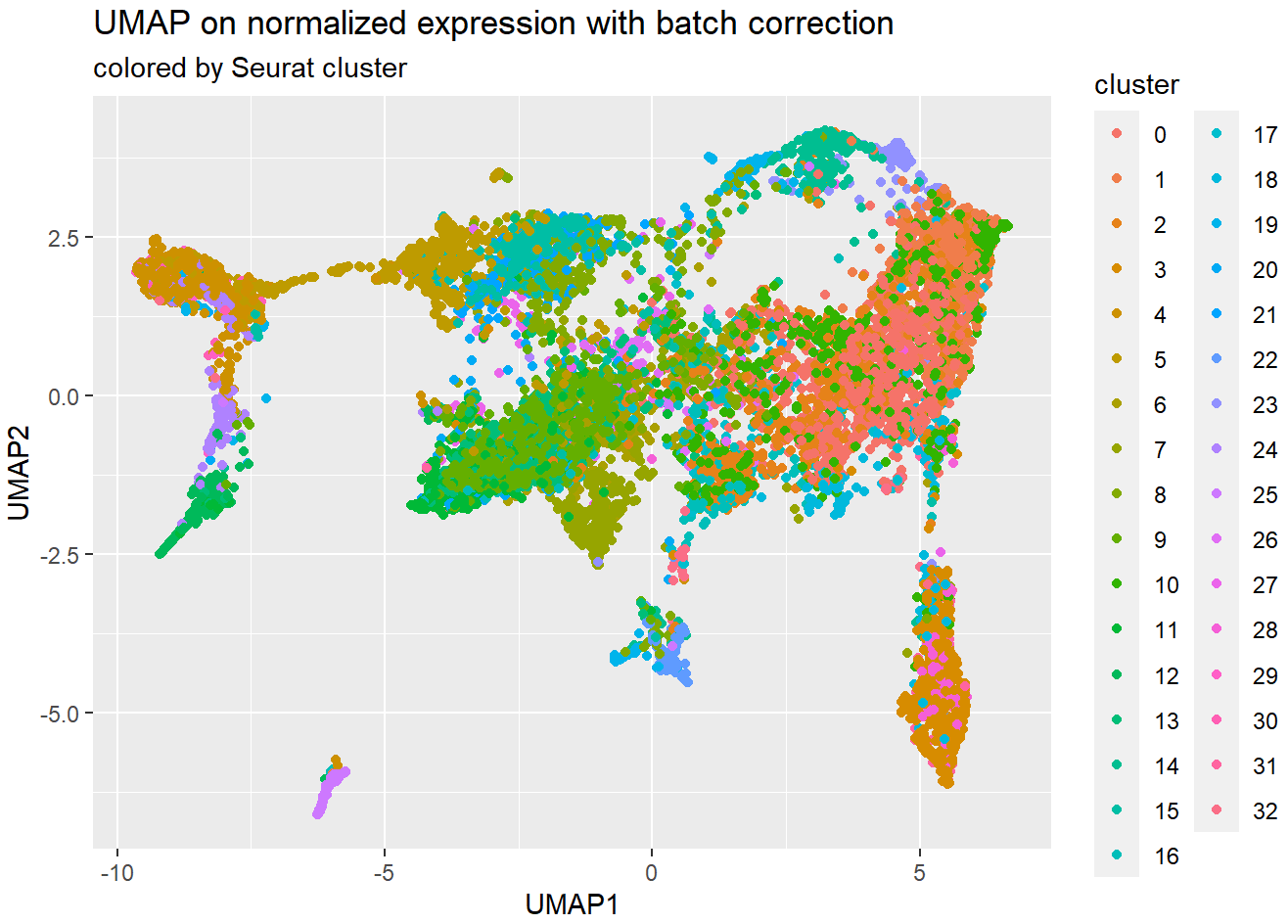
The authors of the original study reported an increased frequency of exhausted T cells in tumor samples compared to adjacent normal tissue. The authors used the gene marker *TIGIT* to identify exhausted T cells. Examine the expression of TIGIT in order to identify T cell subcluster(s) that appear to be exhausted. Then, evaluate the authors claim: do we see an increased frequency of exhausted T cells in tumor samples compared to adjacent normal tissue?

Provide all code and plots you use to answer this question.

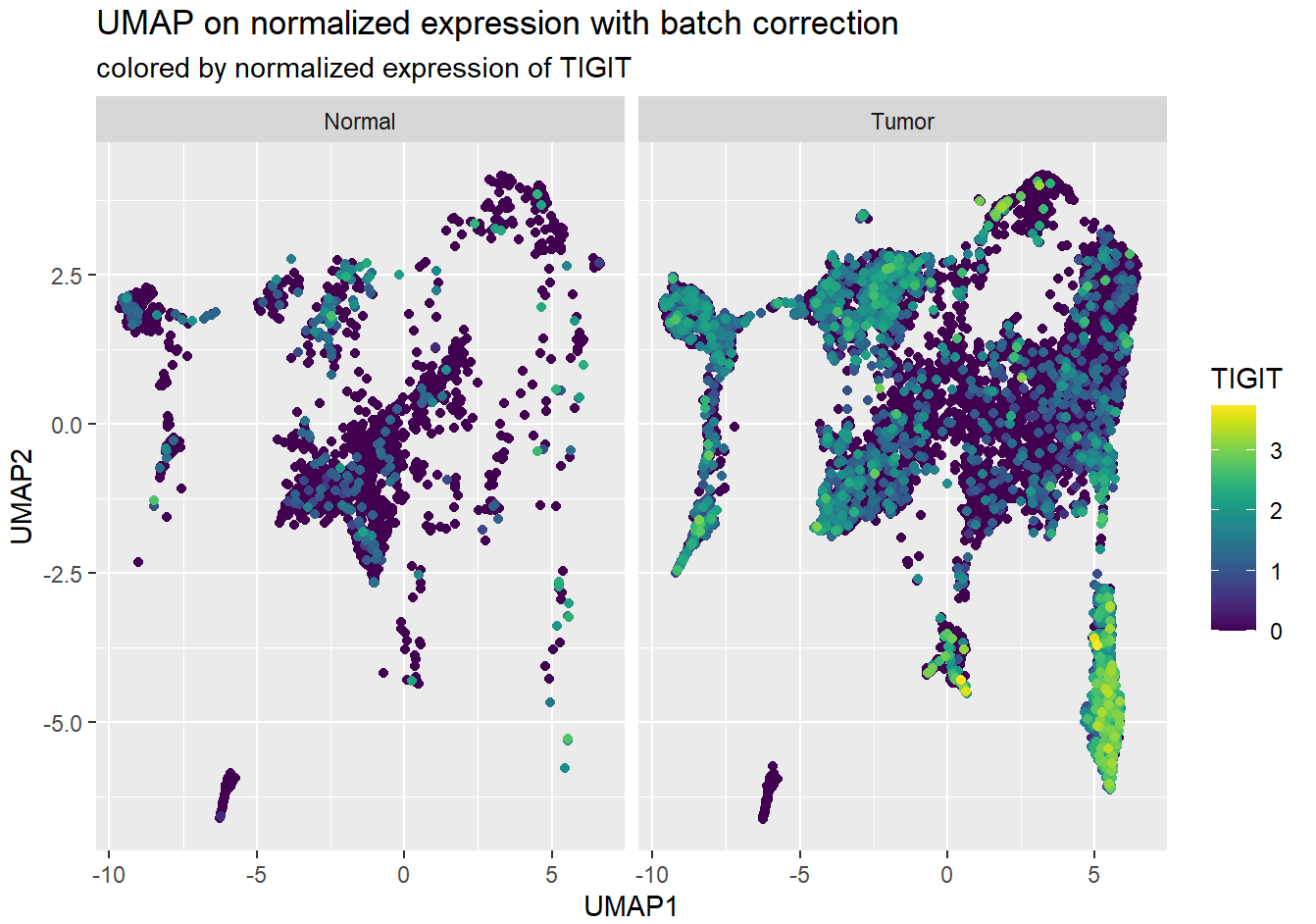
```
metadata_projections_Tcells <- readRDS("metadata_projections_Tcells.rds")
metadata_projections_Tcells <- metadata_projections_Tcells |> mutate(tissue = ifelse(grepl("PDA
C", sample), "Tumor", "Normal"))

#metadata_projections_Tcells <- metadata_projections_Tcells |>
#  filter(cluster %in% c("3", "28"))

metadata_projections_Tcells[order(metadata_projections_Tcells$TIGIT),] |>
  ggplot(aes(x = UMAP1, y = UMAP2, color = TIGIT)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "colored by normalized expression of TIGIT") +
  scale_color_viridis()
```




```
metadata_projections_Tcells |>
  ggplot(aes(x = UMAP1, y = UMAP2, color = cluster)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "colored by Seurat cluster")
```




```
metadata_projections_Tcells[order(metadata_projections_Tcells$TIGIT),] |>
  ggplot(aes(x = UMAP1, y = UMAP2, color = TIGIT)) +
  geom_point() +
  labs(title = "UMAP on normalized expression with batch correction",
       subtitle = "colored by normalized expression of TIGIT") +
  scale_color_viridis() +
  facet_wrap(~tissue)
```




```
metadata_projections_Tcells |> filter(cluster == "3" | cluster == "28") |>
  count(tissue)
```

```
##   tissue   n
## 1 Normal   1
## 2  Tumor 749
```

The UMAP cluster made up of Subclusters 3 and 28 appears to be "exhausted" given that it has the highest expression of TIGIT.

I agree with the author's claim that we see an increased frequency of exhausted T cells in tumor samples compared to adjacent normal tissue. Looking at the Normal vs Tumor UMAP plots, we see th at the Tumor plot had far more expression of TIGIT than the Normal plot. Moreover, when we look at the composition of subcluster 3 and 28 (an exhausted cluster), roughly 99.9% of the cells in come from tumor tissue, supporting the claim that exhausted cells are more likely to be in tumor tissue than normal tissue.

# Part II

Imagine a collaborator has sent you quality-controlled data in the form of a Seurat object: `/shared/courseSharedFolders/133853/HW5/data/multiome_sobj_qcd.rds` . The data is single cell multiome (ATAC + gene expression) from a peripheral blood sample.

For the rest of this homework, we will explore Seurat tools to visualize multimodal data.

## Question 13

Because we have collected two modes of data (RNA and ATAC) for each cell, there are actually two domains in which we can think about cell type and cell similarity.

Reduce the dimensionality of each modality *separately*, arriving at two sets of x,y UMAP coordinates. One set of UMAP coordinates should reflect RNA information alone. The other set of UMAP coordinates should reflect ATAC-seq information alone.

There are plenty of ways to accomplish this task. Feel free to use Seurat functions and to make any parameter choices you feel are reasonable.

```
# Load required libraries
library(Seurat)
multiome_data <- readRDS("/shared/courseSharedFolders/133853/HW5/data/multiome_sobj_qcd.rds")

multiome_data <- NormalizeData(multiome_data, assay = "RNA")
multiome_data  <- ScaleData(multiome_data, assay = "RNA")
multiome_data <- FindVariableFeatures(multiome_data, assay = "RNA")
multiome_data  <- RunPCA(multiome_data, reduction.name = "pca.rna", verbose = FALSE, assay = "RN
A")
multiome_data  <- FindNeighbors(multiome_data, dims = 1:20, assay = "RNA", reduction = "pca.rn
a")
multiome_data  <- FindClusters(multiome_data, resolution = 0.8, assay = "RNA", verbose = FALSE,
reduction = "pca.rna")
multiome_data  <- RunUMAP(multiome_data, reduction = "pca.rna", dims = 1:20, assay = "RNA", redu
ction.name = "umap.rna")
umap_coordinates_RNA <- as.data.frame(Embeddings(object = multiome_data, reduction = "umap.rn
a"))
saveRDS(umap_coordinates_RNA, file = "umap_coordinates_RNA.rds")

multiome_data <- NormalizeData(multiome_data, assay = "ATAC")
multiome_data  <- ScaleData(multiome_data, assay = "ATAC")
multiome_data <- FindVariableFeatures(multiome_data, assay = "ATAC")
multiome_data  <- RunPCA(multiome_data, reduction.name = "pca.atac", verbose = FALSE, assay = "A
TAC")
multiome_data  <- FindNeighbors(multiome_data, dims = 1:20, assay = "ATAC", reduction = "pca.ata
c")
multiome_data  <- FindClusters(multiome_data, resolution = 0.8, assay = "ATAC", verbose = FALSE,
reduction = "pca.atac")
multiome_data  <- RunUMAP(multiome_data, reduction = "pca.atac", dims = 1:20, assay = "ATAC", re
duction.name = "umap.atac")
umap_coordinates_ATAC <- as.data.frame(Embeddings(object = multiome_data, reduction = "umap.ata
c"))
saveRDS(umap_coordinates_ATAC, file = "umap_coordinates_ATAC.rds")
```

# Question 14

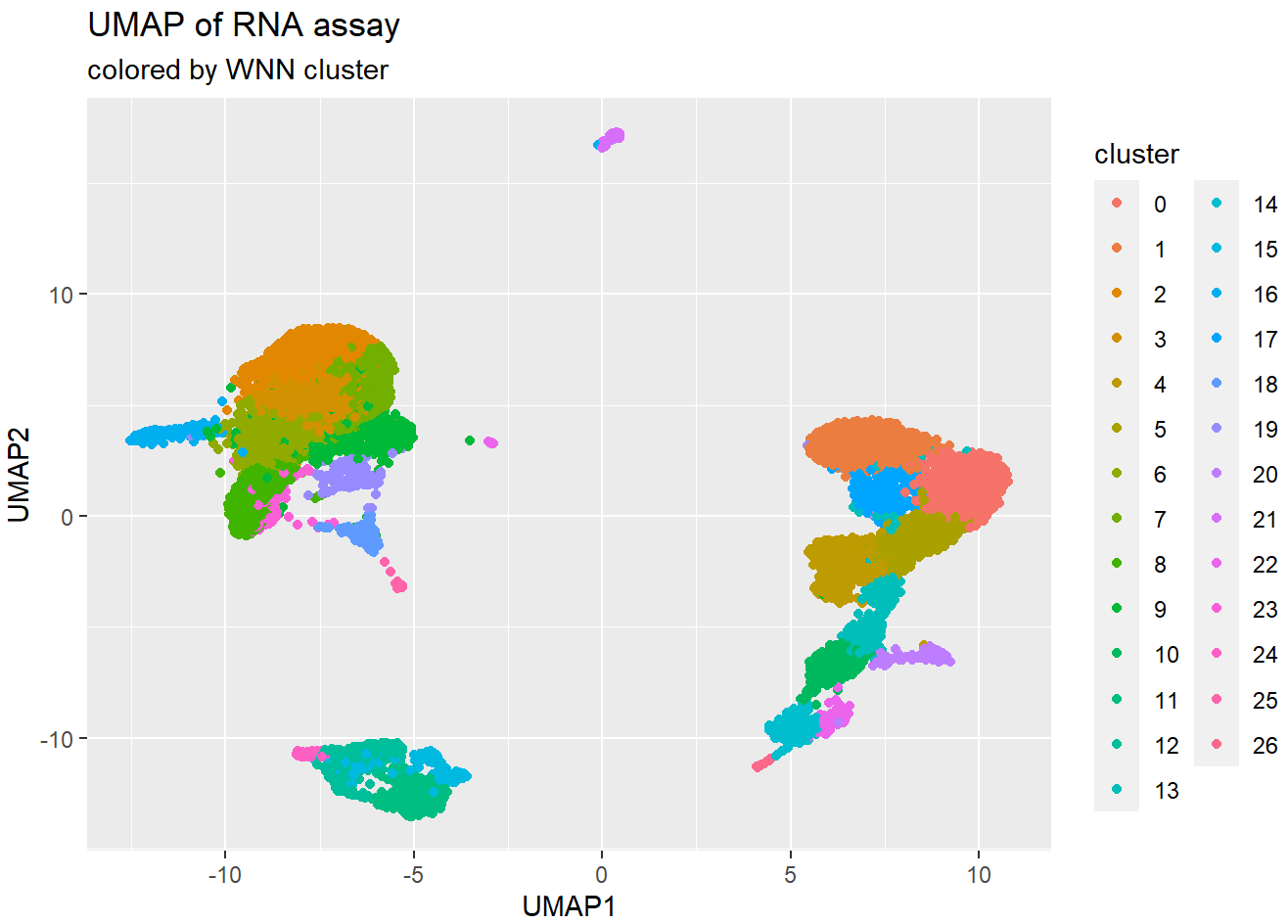
A prominent strategy to integrate two modalities in single cell data is weighted-nearest-neighbor (WNN, https://www.sciencedirect.com/science/article/pii/S0092867421005833 (https://www.sciencedirect.com/science/article/pii/S0092867421005833)) analysis. Use Seurat's `FindMultiModalNeighbors()` function to build a WNN graph, and then use this information to make a third set of UMAP coordinates which reflect both the RNA and the ATAC information. Apply Louvain clustering to the WNN graph.

```
# Find multimodal neighbors using WNN
multiome_data <- FindMultiModalNeighbors(
  object = multiome_data,
  reduction.list = list("pca.rna", "pca.atac"),
  dims.list = list(1:20, 1:20),
  modality.weight.name = "modality.weight",
  verbose = TRUE
)

multiome_data <- RunUMAP(multiome_data, nn.name = "weighted.nn", reduction.name = "umap.wnn")
multiome_data <- FindClusters(multiome_data, algorithm = 3, resolution = 2, verbose = FALSE)

umap_coordinates_WNN <- as.data.frame(Embeddings(object = multiome_data, reduction = "umap.wn
n"))

umap_coordinates_WNN$cluster <- Idents(multiome_data)


saveRDS(umap_coordinates_WNN, file = "umap_coordinates_WNN.rds")
```

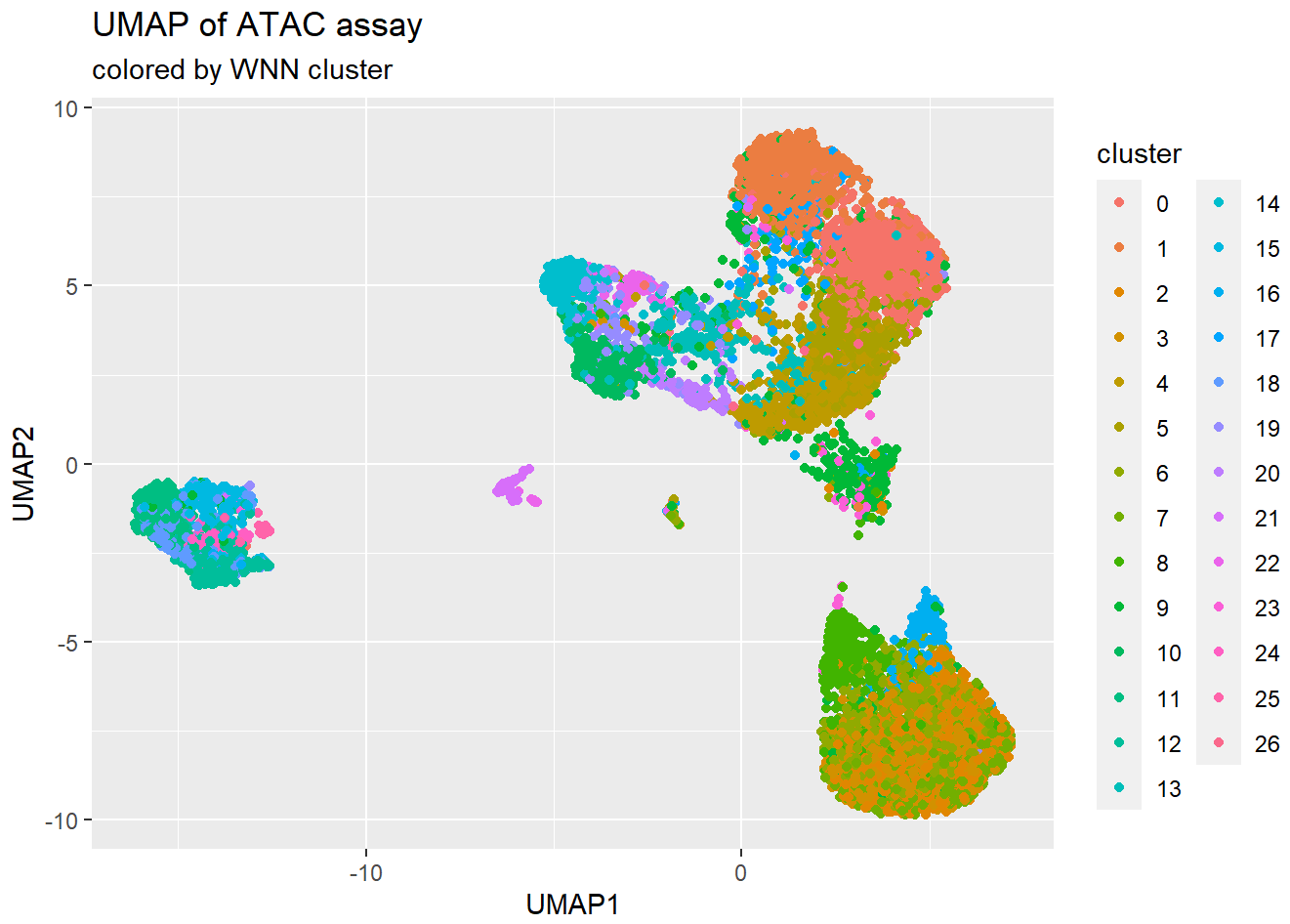
## Question 15

Make three scatterplots to compare the three sets of UMAP coordinates (RNA, ATAC, and WNN). To facilitate visual comparison, color the cells by their WNN cluster assignment in *every plot*.

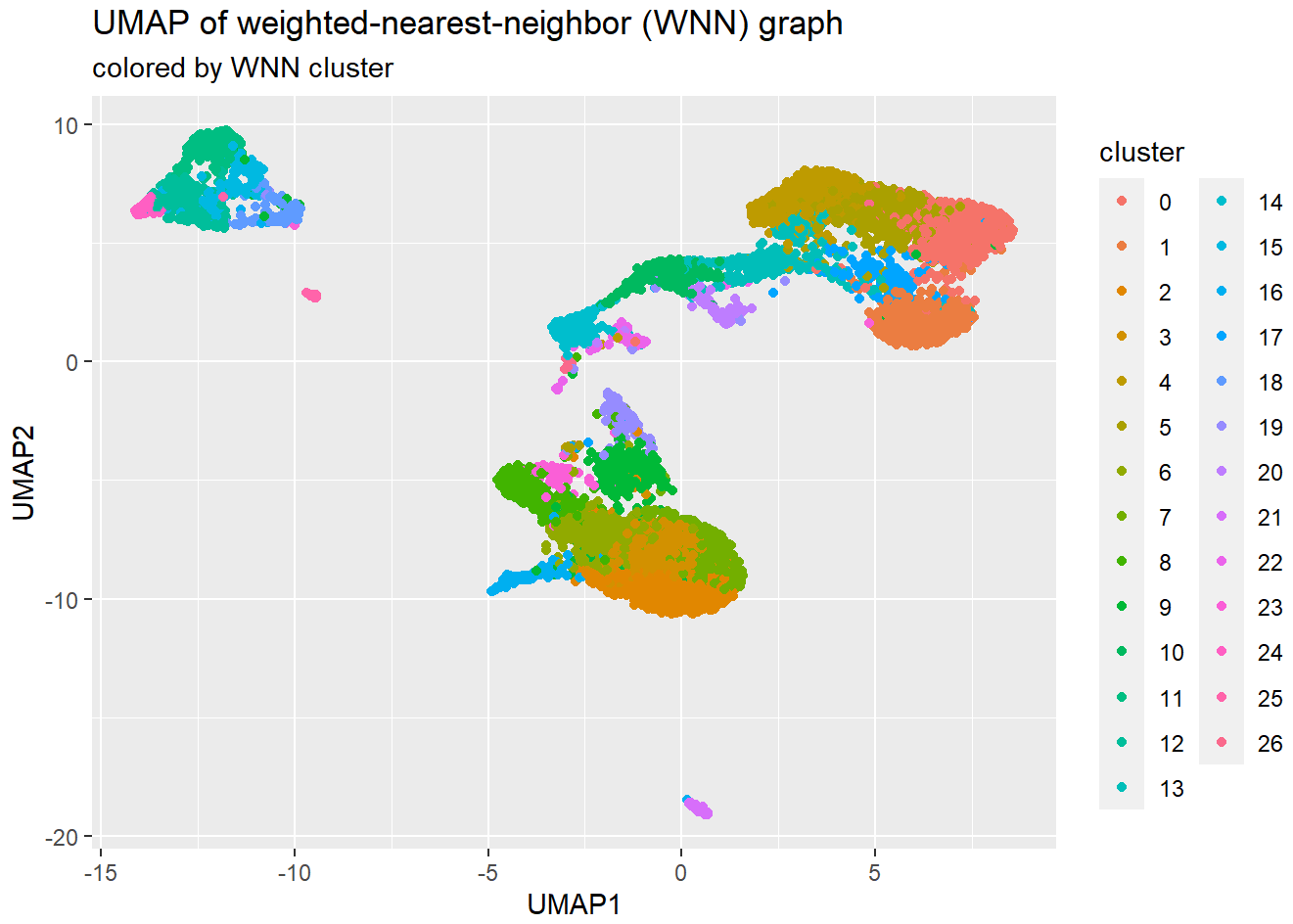
```
rna <- readRDS("umap_coordinates_RNA.rds")
atac <- readRDS("umap_coordinates_ATAC.rds")
wnn <- readRDS("umap_coordinates_WNN.rds")

umap_coordinates <- cbind(rna, atac, wnn)

#umap_coordinates <- umap_coordinates |>
# filter(cluster %in% c("21"))

umap_coordinates |> ggplot(aes(x = umaprna_1, y = umaprna_2, color = cluster)) +
  geom_point() +
  labs(title = "UMAP of RNA assay",
       subtitle = "colored by WNN cluster",
       x = "UMAP1", y = "UMAP2")
```




```
umap_coordinates |> ggplot(aes(x = umapatac_1, y = umapatac_2, color = cluster)) +
  geom_point() +
  labs(title = "UMAP of ATAC assay",
       subtitle = "colored by WNN cluster",
       x = "UMAP1", y = "UMAP2")
```




```
umap_coordinates |> ggplot(aes(x = umapwnn_1, y = umapwnn_2, color = cluster)) +
  geom_point() +
  labs(title = "UMAP of weighted-nearest-neighbor (WNN) graph",
       subtitle = "colored by WNN cluster",
       x = "UMAP1", y = "UMAP2")
```




If you were the lead analyst for this dataset, would you prefer to work with the single WNN representation, or the separate RNA and ATAC representations? How big of a difference would this make? Justify your choice in 3 sentences or less.

```
I would prefer to work with the single WNN representation because it integrates information from
both RNA and ATAC modalities, capturing potential interactions between them. This approach allow
s for a more comprehensive understanding of cellular heterogeneity and potentially reveals novel
biological insights that might be missed when analyzing RNA and ATAC data separately. Additional
ly, it simplifies downstream analyses by providing a unified framework for interpretation, which
is shown by the clear clusters in the WNN graph.
```

## Question 16

chromVAR (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5623146/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5623146/)) provides an efficient algorithm to scan the genome for ATAC fragment pileup surrounding motifs of interest.

Fill in the lines of code below to calculate chromVAR chromatin accessibility scores for the collection of human transcription factor binding motifs in JASPAR 2020 (https://jaspar2020.genereg.net/ (https://jaspar2020.genereg.net/)).

```
library(Matrix)
library(Seurat)
library(Signac)
library(chromVAR)
library(JASPAR2020)
library(TFBSTools)
library(motifmatchr)
library(BSgenome.Hsapiens.UCSC.hg38)

DefaultAssay(multiome_data) <- "ATAC"

pwm_set <- getMatrixSet(x = JASPAR2020, opts = list(species = 9606, all_versions = FALSE))

motif.matrix <- CreateMotifMatrix( features = granges(multiome_data), pwm = pwm_set, genome = 'h
g38', use.counts = FALSE)

motif.object <- CreateMotifObject(data = motif.matrix, pwm = pwm_set)

sobj <- SetAssayData(multiome_data, assay = "ATAC", slot = 'motifs', new.data = motif.object)

sobj <- RunChromVAR(sobj, genome = BSgenome.Hsapiens.UCSC.hg38)
##after filling this in, run it on the server!
```

## Question 17

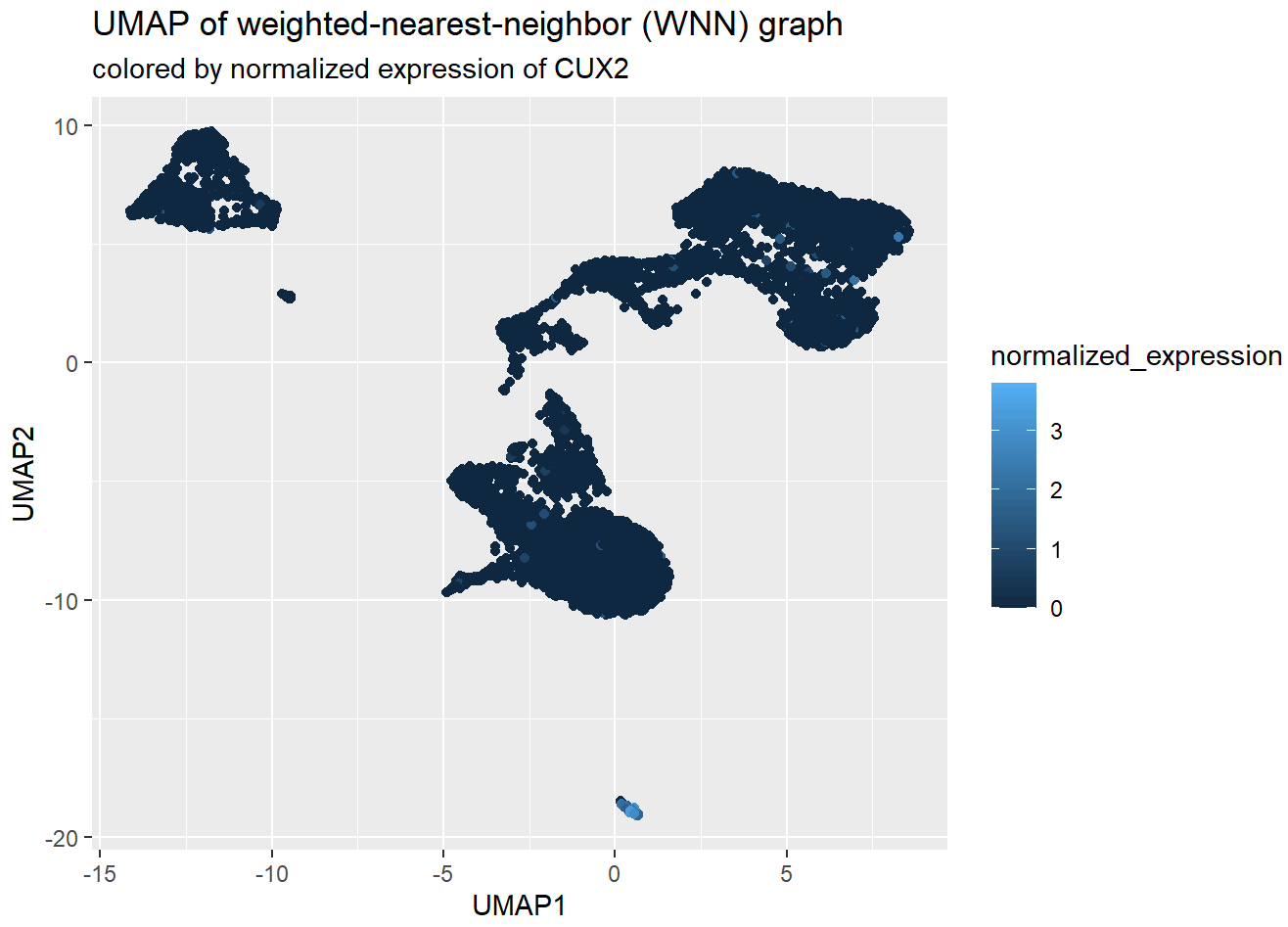
Now, let's identify transcription factor binding motifs that may help to explain the transcriptional heterogeneity in these data. Choose one cluster from your WNN graph to examine more closely. Fill in the lines of code below to identify 1) RNA transcripts that differentially expressed in your favorite cluster compared to clusters 2) transcription factor binding motifs that are differentially accessibile in your favorite cluster compared to other clusters.

```
library(Seurat)

rna_markers = FindMarkers(object = sobj, ident.1 = 21, assay = "RNA")
chromvar_markers = FindMarkers(object = sobj, ident.1 = 21, assay = "chromvar")
chromvar_markers$gene <- ConvertMotifID(sobj, id=rownames(chromvar_markers))
##after filling this in, run it on the server!



rna_markers$gene <- rownames(rna_markers)
overlapped_genes <- intersect(rna_markers$gene, chromvar_markers$gene)

gene <- overlapped_genes[1]

row <- rownames(chromvar_markers)[which(chromvar_markers$gene == gene)]

a_score <- sobj@assays$chromvar@data[row,]
normalized_expressions <- GetAssayData(sobj, assay = "RNA")
norm_expression <- normalized_expressions[gene,]

saveRDS(a_score, "a_score.rds")
saveRDS(norm_expression, "norm_expression.rds")
```

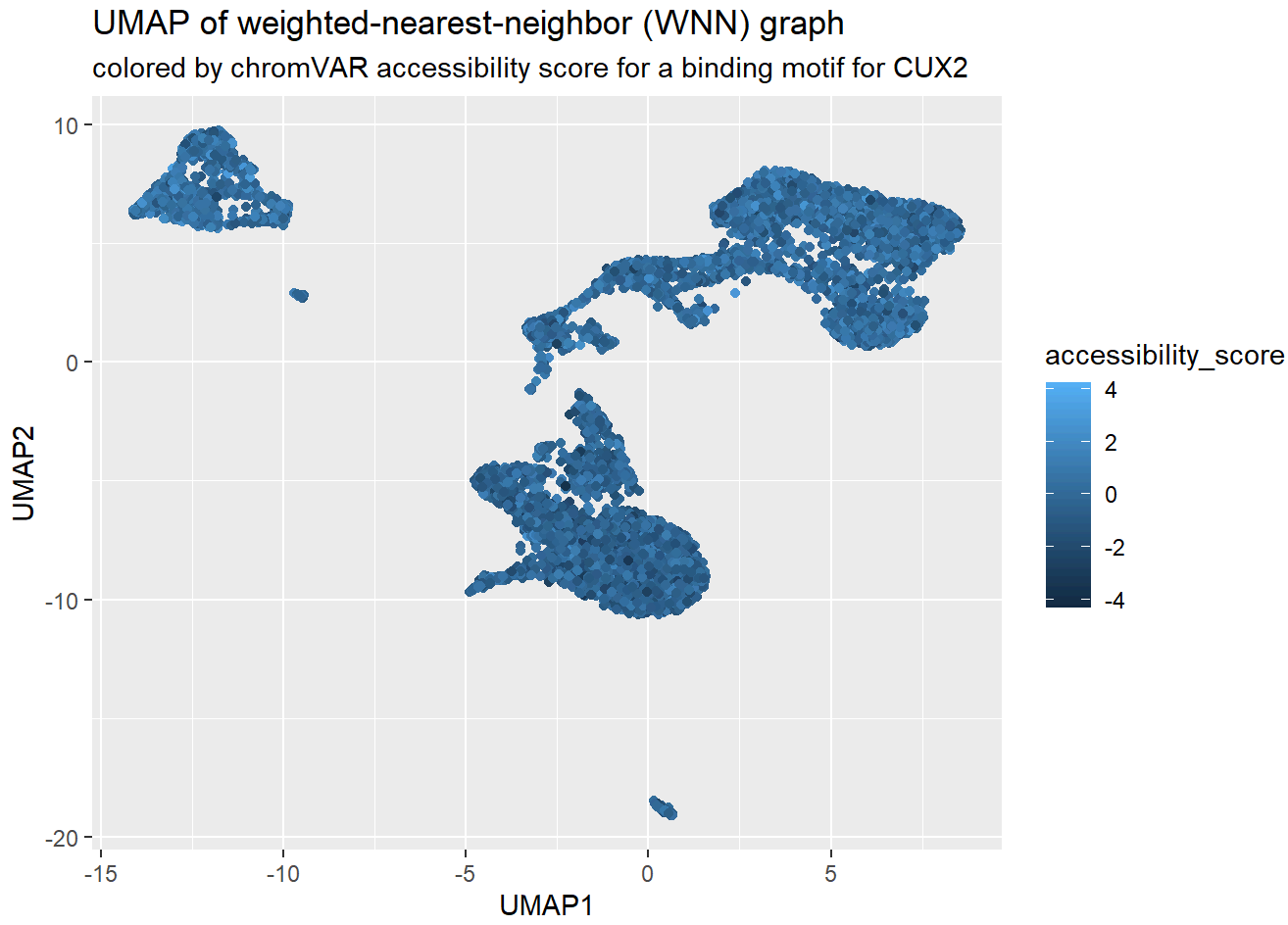
From your results, identify a transcription factor (gene name) that is differentially expressed in your favorite cluster *AND* whose transcription factor binding site is significantly more accessible in your favorite cluster.

To show your result, make two WNN UMAP scatterplots. In the first UMAP, color each cell by the normalized expression [log(CP10K + 1)] of the gene you identified. In the second UMAP, color each cell by the chromVAR accessiblity score for a binding motif for that gene.

```
a_score <- readRDS("a_score.rds")
norm_expression <- readRDS("norm_expression.rds")

umap_coordinates$accessibility_score <- a_score
umap_coordinates$normalized_expression <- norm_expression

umap_coordinates |> ggplot(aes(x = umapwnn_1, y = umapwnn_2, color = normalized_expression)) +
  geom_point() +
  labs(title = "UMAP of weighted-nearest-neighbor (WNN) graph",
       subtitle = "colored by normalized expression of CUX2",
       x = "UMAP1", y = "UMAP2")
```




```
umap_coordinates |> ggplot(aes(x = umapwnn_1, y = umapwnn_2, color = accessibility_score)) +
  geom_point() +
  labs(title = "UMAP of weighted-nearest-neighbor (WNN) graph",
       subtitle = "colored by chromVAR accessibility score for a binding motif for CUX2",
       x = "UMAP1", y = "UMAP2")
```




```
# genes_with_commas <- paste(overlapped_genes, collapse = ",")
# write.table(genes_with_commas, file = "gene_list.txt", row.names = FALSE, col.names = FALSE)
```

## Question 18

Based on your results from Question 17, how would you annotate the cell type of your favorite cluster? Be as specific as you think is appropriate given the data. Provide a brief explanation (2 sentences or less), citing online sources.

```
Based on my results and using the Annotation of Cell Types website, I would annotate the cell ty
pe of cluster 21 as a dendritic cell, more specifically a plasmacytoid dendritic cell, which is
a rare type of immune cell that are known to secrete large quantities of type 1 interferon (IFN
s). It makes sense that the smallest cluster (cluster 21) corresponds to a rare immune cell and
that CUX2 would have the highest expression within this cluster (since Cux2 induces cell autonom
ous development of dendritic branches).
```

```
knitr::include_graphics("q18.png")
```

ACT

Home Search BatchACT Download Help

Heatmap shows intersection of input genes with canonical marker genes ( ), DEGs ( ) or both ( ). Click the colored cell in heatmap to see more details of a marker.

| | Cell Ontology | Cell Type | Padj | Overlap Markers | CUX2 | SPIB | TCF4 | MYBL2 | IRF7 | RUNX2 | IRF8 | SOX4 | POU4F1 | MEIS1 | NR5A1 | MEF2C | E2F7 | POU4F3 | INSM1 | RREB1 | CUX1 | FOXN3 | MEF2D | TBX19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Dendritic cell | Plasmacytoid den... | 1.26e-21 | 5/93 | | | | | | | | | | | | | | | | | | | | |

Showing 1 to 1 of 1 entries

Dendritic cell (1)