Homework5 Single-cell Multiome Data Analysis

Introduction

The technology of gathering data from multi-modality within the same cell offers opportunities for gaining holistic views of cells individually. 10X Chromium Single-cell Multiome ATAC + Gene Expression simultaneously profiling transcriptome and epigenome at single-cell level, enabled deeper characterization of cell types and states.

In addition to their power in addressing biological questions, single-cell multiome also provides good testing data for evaluating label transferring accuracy. In a single omic scATAC-seq analysis, people typically transfer cell type labels from existing scRNA-seq data to obtain the cell type annotation. Leveraging multiome ATAC + Gene Expression, we can link transcriptome and epigenome modalities cell by cell. Thereby, we know the "ground truth" cell type label for each cell in scATAC-seq data and compare the results from Seurat's label transferring methods.

In this homework, we will start from the 10X fastq raw reads, go through pre-processing pipeline and downstream analysis to get an overall idea of single-cell data analysis. For the data pre-processing, we will use the MAESTRO package installed on Cannon. For downstream analysis, we will work on your local computer's Rstudio using Seurat and Signac R packages.

Homework Questions

Part I. Running MAESTRO (Graduate level Students Only)

- Understand MAESTRO workflow
- Set up MAESTRO analysis pipelines

Step0. Get 10X pbmc_granulocyte_sorted_3k multiome data Single-cell multiome data were downloaded from the 10X website. You can find raw data here. Data were downloaded to the Cannon server. There are two data folders under `/n/stat115/2021/HW5/pbmc_granulocyte_sorted_3k/` directory. One is `gex/` for scRNA-seq data and the other one is `atac/` for scATAC-seq data. You will need to activate the conda environment to access all the required packages: `$ source /n/stat115/2021/HW5/miniconda3/bin/activate MAESTRO`. If you want to learn more about conda environment management, you can read through the link conda.

```
source /n/stat115/2021/HW5/miniconda3/bin/activate MAESTRO
```

Step1. Configure MAESTRO working directory. MAESTRO is a snakemake pipeline developed for streamlined pre-processing of single-cell data and downstream analysis. For more details, you can read through the documentation of MAESTRO. MAESTRO is written in Snakemake. Please learn more in the snakemake documentation.

This step will configure two working directories for scRNA-seq and scATAC-seq, respectively. A `Snakefile` and a `config.yaml` file will be configured within each directory. `Snakefile` includes all the snakemake rules that will be later executed. `config.yaml` file contains the detailed parameter settings you need to specify when initiating the directory. You can also manually change the contents of the config.yaml file to customize your run. Let's call the directory for processing scRNA-seq data as `multiome_scrna/` and scATAC-seq data as `multiome_scatac/`.

Please read the detailed tutorials in MAESTRO documentations to have a better idea of how to set each parameter:

scRNA-seq

scATAC-seq

Before running, Make sure you already have the below files on your server.

For scRNA-seq:

1. STAR mapping index file: This is the STAR genome reference file for mapping human single-cell RNA-seq data.

- Path: `/n/stat115/2021/HW5/references/Refdata_scRNA_MAESTRO_GRCh38_1.2.2`

2. barcode whitelist: This is the complete list of multiome scRNA-seq cell barcodes from the 10X Cell Ranger ARC workflow. We use the barcode whitelist to correct the cell barcodes we get from reads.

- Path: `/n/stat115/2021/HW5/references/whitelist/rna/737K-arc-v1.txt`

3. lisa TF annotation file: This file contains data used for running LISA2.

- Path: `/n/stat115/2021/HW5/references/lisa_data/hg38_1000_2.0.h5`

For scATAC-seq:

1. giggleannotation file: giggle annotation file is required for regulator identification.

- Path: `/n/stat115/2021/HW5/references/giggle.all`

2. minimap2 reference file: This is the STAR genome reference file for mapping human single-cell ATAC-seq data.

- Path: `/n/stat115/2021/HW5/references/Refdata_scATAC_MAESTRO_GRCh38_1.1.0`

3. barcode whitelist: This is the complete list of multiome scATAC-seq cell barcodes from the 10X Cell Ranger ARC workflow. We use the barcode whitelist to correct the cell barcodes we get from reads.

- Path: `/n/stat115/2021/HW5/references/whitelist/atac/737K-arc-v1.txt`

hints:

1. MAESTRO has two sub-commands to initiate the working directories.
2. You will need to feed each parameter settings to each sub-commands for initialization.

Step2. Run the snakemake pipeline. You should submit `slurm` jobs to the `Cannon` cluster for scRNA-seq and scATAC-seq, respectively.

hint: Estimated running time and memory usage:

3k multiome scrna-seq: 3-6 hrs; 60G; 12 cores

3k multiome scatac-seq: 10 hrs; 60G; 12 cores

```
source /n/stat115/2021/HW5/miniconda3/bin/activate MAESTRO

# scRNA-seq initiation:
MAESTRO scrna-init --platform 10x-genomics --species GRCh38 \
--fastq-dir /n/stat115/2021/HW5/pbmc_granulocyte_sorted_3k/gex --fastq-prefix pbmc_granulocyte_sorted_3
--cores 16 --rseqc --directory multiome_scrna --outprefix scrna_pbmc_granulocyte_sorted_3k \
--mapindex /n/stat115/2021/HW5/references/Refdata_scRNA_MAESTRO_GRCh38_1.2.2/GRCh38_STAR_2.7.6a \
--whitelist /n/stat115/2021/HW5/references/whitelist/rna/737K-arc-v1.txt \
--umi-length 12 --lisadir /n/stat115/2021/HW5/references/lisa_data/hg38_1000_2.0.h5 --signature human.i
```

```
cd multiome_scrna/

#First, test with a dry run to see if the pipeline work
##Remember to add -np for a DRY run:
snakemake -np --rerun-incomplete -j 1
```

This code is inside of `multiome_scrna/maestro-scrna-seq.sbatch`:

```
#If a dry run is 100% complete and no red error messages reported, we will create a sbatch job script w
#-j 12 is the total number of cores you will need to specify when creating the job.

#!/bin/bash

source /n/stat115/2021/HW5/miniconda3/bin/activate MAESTRO
snakemake -j 12 --latency-wait 200
```

This code is run in `multiome_scrna`:

```
sbatch -n 1 -N 1 --mem=64G --cpus-per-task=16 --time=06:00:00 maestro-scrna-seq.sbatch
```

```
cd ../

# scATAC-seq initiation:
MAESTRO scatac-init --platform 10x-genomics --format fastq --species GRCh38 \
--fastq-dir /n/stat115/2021/HW5/pbmc_granulocyte_sorted_3k/atac --fastq-prefix pbmc_granulocyte_sorted_3
--cores 16 --directory multiome_scatac --outprefix scatac_pbmc_granulocyte_sorted_3k \
--peak-cutoff 100 --count-cutoff 1000 --frip-cutoff 0.2 --cell-cutoff 50 \
--giggleannotation /n/stat115/2021/HW5/references/giggle.all \
--fasta /n/stat115/2021/HW5/references/Refdata_scATAC_MAESTRO_GRCh38_1.1.0/GRCh38_genome.fa \
--whitelist /n/stat115/2021/HW5/references/whitelist/atac/737K-arc-v1.txt \
--rpmodel Enhanced \
--annotation --method RP-based --signature human.immune.CIBERSORT

cd multiome_scatac/

#First, test with a dry run to see if the pipeline work
source /n/stat115/2021/HW5/miniconda3/bin/activate MAESTRO
##Remember to add -np for a DRY run:
snakemake -np --rerun-incomplete -j 1

#If a dry run is 100% completed and no red error messages reported, we will create a sbatch job script
#-j 16 is the total number of cores you need to specify when creating the job.
```

This code is inside of `multiome_scatac/maestro-scatac-seq.sbatch`:

```
#!/bin/bash

source /n/stat115/2021/HW5/miniconda3/bin/activate MAESTRO
snakemake -j 16 --latency-wait 200
```

This code is run in `multiome_scatac`:

```
sbatch -n 1 -N 1 --mem=64G --cpus-per-task=16 --time=06:00:00 -o output.out -e error.out maestro-scatac
```

I used `squeue` to keep track of the progress of my jobs.

1 Reads mapping stats: After getting the `Result/` output folder, for scRNA-seq run, please copy

the content of `multiome_scrna/Result/QC/scrna_pbmc_granulocyte_sorted_3k_bam_stat.txt` below (1 pts;). For the scATAC-seq run, please copy the content of `multiome_scatac/Result/QC/flagstat.txt` below (1 pts;).

content of `multiome_scrna/Result/QC/scrna_pbmc_granulocyte_sorted_3k_bam_stat.txt`

```
#================================================
#All numbers are READ count
#================================================

Total records:                          211864897

QC failed:                              0
Optical/PCR duplicate:                  0
Non primary hits                        55340229
Unmapped reads:                         0
mapq < mapq_cut (non-unique):           15703353

mapq >= mapq_cut (unique):              140821315
Read-1:                                 0
Read-2:                                 0
Reads map to '+':                       82338792
Reads map to '-':                       58482523
Non-splice reads:                       122222735
Splice reads:                           18598580
Reads mapped in proper pairs:           0
Proper-paired reads map to different chrom:0
```

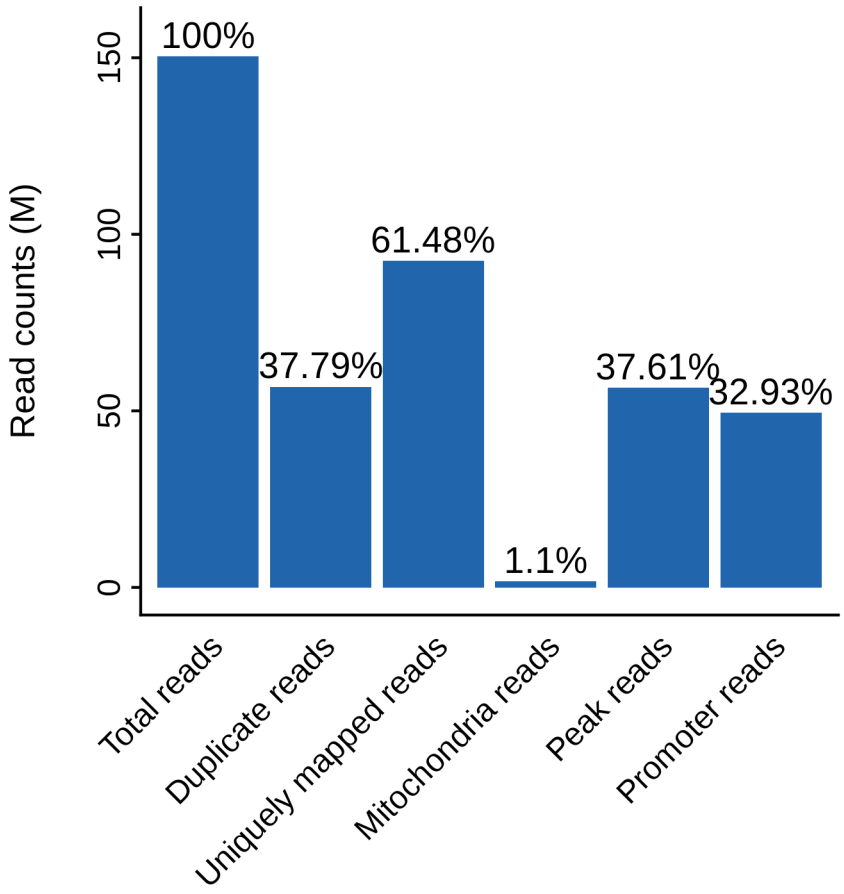
content of `multiome_scatac/Result/QC/flagstat.txt`

```
150362779 + 0 in total (QC-passed reads + QC-failed reads)
0 + 0 secondary
15795 + 0 supplementary
56816207 + 0 duplicates
142195627 + 0 mapped (94.57% : N/A)
150346984 + 0 paired in sequencing
75173492 + 0 read1
75173492 + 0 read2
132427806 + 0 properly paired (88.08% : N/A)
137450532 + 0 with itself and mate mapped
4729300 + 0 singletons (3.15% : N/A)
512146 + 0 with mate mapped to a different chr
197349 + 0 with mate mapped to a different chr (mapQ>=5)
92444360
1658444
49512627
56549714
```

2. Cell Filtering QC plot: Please attach the cell filtering QC plot for the scATAC-seq run, which is saved as `multiome_scatac/Result/QC/scatac_pbmc_granulocyte_sorted_3k_scATAC_read_distr.png` (1 pts;). Please briefly describe what you observed from this figure and how the filtering was affected by the cutoff value you set in the `MAESTRO scatac-init` subcommand (1 pts;).

`multiome_scatac/Result/QC/scatac_pbmc_granulocyte_sorted_3k_scATAC_read_distr.png` (1 pts;)

```
library(knitr)
knitr::include_graphics('scatac_pbmc_granulocyte_sorted_3k_scATAC_read_distr.png')
```



describe what you observed from this figure and how the filtering was affected by the cutoff value you set in the `MAESTRO scatac-init` subcommand (1 pts;)

```
In `MAESTRO scatac-init`, cutoffs were set so that the minimum number of peaks included in each cell was

This means that cells with fewer than 1000 unique fragments, fewer than 100 peaks, or 20% fraction of pe

The final amount of reads in peak regions was 37.61%, and the final amount of reads in promoter regions
```

3. Bonus Question: Suppose you have a list of 500 cell barcodes, can you sub-sample the scATAC-seq data by cell barcodes to get a raw fastq file with only 500 cells? Please provide the path to downsampled fastq file on Cannon (3 pts;).

hints:

1. After running MAESTRO, cells passed the QC filter will be stored in `multiome_scatac/Result/QC/scatac_pbmc_granul` You can sub-sample 500 cell barcodes using the command `shuf -n 500 scatac_pbmc_granulocyte_sorted_3k_scATAC_` `> 500_vallidcells.txt`.
2. Sinto has a function called filterbarcods that can sub-sample bam file according to the cell barcodes.
3. There are several tools that can convert ban files back to fastq. 10X has a script called bamtofastq. Another tool bam2fastq have similar functions.
4. You don't have to start from bam files. Any methods are appreciated.

Part II. Single-cell RNA-seq (Undergraduate level Starts from here)

- Create a scRNA-seq Seurat Object
- QC and Pre-processing
- Normalization and Dimension Reduction
- KNN and Clustering
- Finding Markers
- Cell Type Annotation

Now, we are done with all the work on the server. In this part, we'll continue to work on single-cell Seurat objects generated by MAESTRO. After running MAESTRO, all the output files will be organized under a data folder named `Result/`. A Seurat object containing the count matrix and the metadata will be stored in a .rds data list. You can always find them in the MAESTRO analysis results: `multiome_scrna/Result/Analysis` and `multiome_scatac/Result/Analysis`. In a real data analysis workflow, you will continue to work on the .rds file you got from MAESTRO. But in this homework, let's use the prepared Seurat objects to keep downstream analysis consistent.

Please go to your local computer and download two .rds file from `/n/stat115/2021/HW5/data`. The single-cell RNA-seq object is `/n/stat115/2021/HW5/data/scrna_pbmc_granulocyte_sorted_3k_scRNA_Object.rds` and the single-cell ATAC-seq object is stored as `/n/stat115/2021/HW5/data/scatac_pbmc_granulocyte_sorted_3k_scATAC`

Please install and load the following packages:

```
library(dplyr)
library(Seurat)
library(patchwork)
library(tidyverse)
```

1. In Rstudio, you can use `readRDS()` to load the .rds data. You will find that the .rds is a data list containing an `RNA` Seurat object and a differentially expressed gene list. We will only need to extract the Seurat object for further analysis. Please Describe the raw dataset's composition: what are the number of genes and number of cells in your cell feature matrix (1 pts;) ?

```
##Read the .rds file for scRNA-seq data:
rna <- readRDS("scrna_pbmc_granulocyte_sorted_3k_scRNA_Object.rds")

#MAESTRO also attached differentially expressed genes as a table under the .rds file. We need to extrac
rna <- rna$RNA
rna
```

```
## An object of class Seurat
## 14251 features across 2633 samples within 1 assay
## Active assay: RNA (14251 features, 2000 variable features)
##  2 dimensional reductions calculated: pca, umap
```

Answer:

There are 14251 genes (features) across 2633 cells (samples) in the cell feature matrix.

2. Filtering cells with a high proportion of mitochondrial reads (potential dead cells) or outlier number of genes (possible low reactions or multiplets) are essential steps in single-cell analysis. Outlier cells with too high or low gene coverage should be removed. The cutoff depends on the scRNA-seq technology and the distribution of each dataset. MAESTRO has already filtered the cells based on the number of counts and genes. In this question, please calculate the percentage of UMIs mapped to the mitochondrial genes, save the results under `percent.mt` column in the Seurat `@metadata` slot (1 pts;). Please visualize the distribution of `nFeature_RNA`, `nCount_RNA` and `percent.mt` in a single violin plot (1 pts;). hints: You may want

to use `Idents(rna) <- rna$orig.ident` before running violin plot for better visualization. Use a table to show how many of the cells have mitochondrial rate > 20% (1 pts;).

```
Idents(rna) <- rna$orig.ident
```

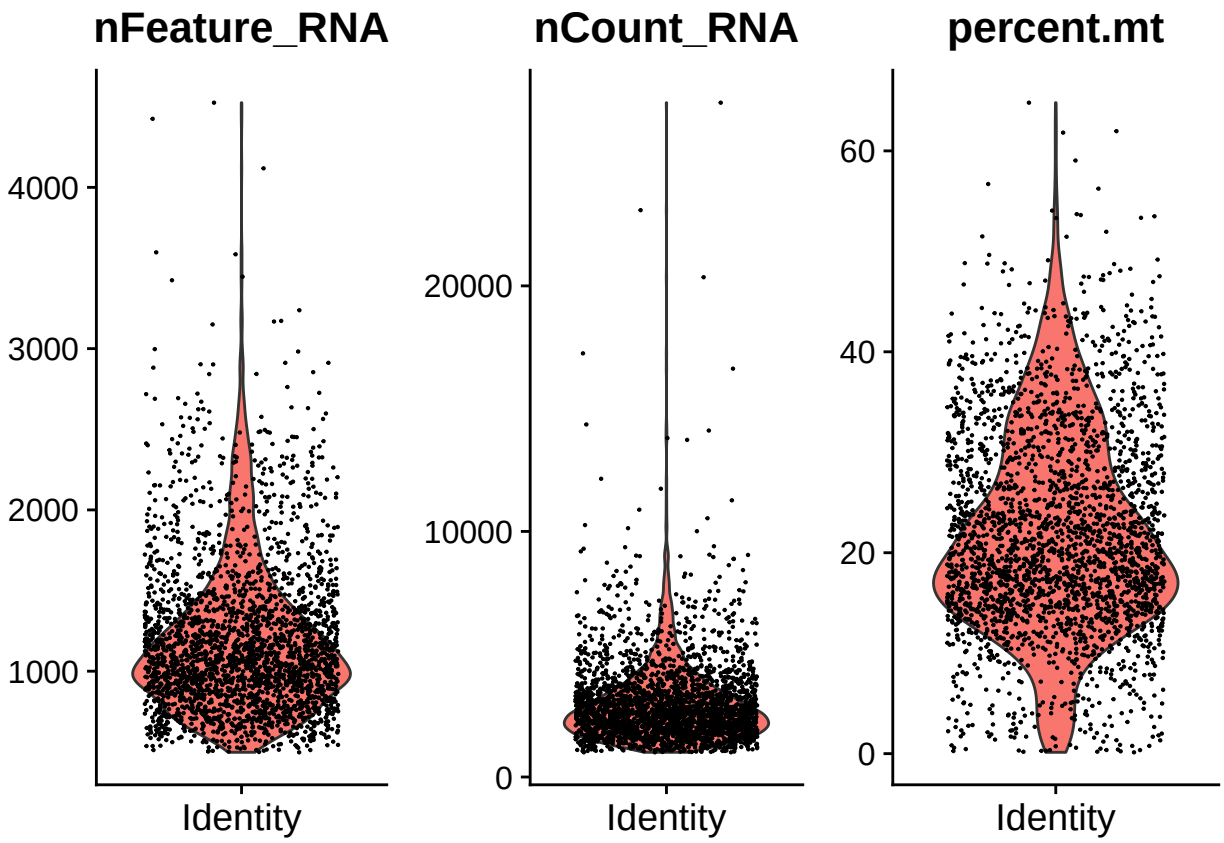
Calculate the percentage of UMIs mapped to the mitochondrial genes, save the results under `percent.mt` column in the Seurat `@metadata` slot (1 pts;)

```
rna[['percent.mt']] <- PercentageFeatureSet(rna, pattern = "^MT-")
head(rna@meta.data)
```

```
##                                     orig.ident nCount_RNA nFeature_RNA
## AAACAGCCAAATATCC scrna_pbmc_granulocyte_sorted_3k       3380         1586
## AAACAGCCAGGAACTG scrna_pbmc_granulocyte_sorted_3k       5048         2084
## AAACAGCCAGGCTTCG scrna_pbmc_granulocyte_sorted_3k       1990          742
## AAACCAACACCTGCTC scrna_pbmc_granulocyte_sorted_3k       1472          667
## AAACCAACAGATTCAT scrna_pbmc_granulocyte_sorted_3k       2106          935
## AAACCAACAGTTGCGT scrna_pbmc_granulocyte_sorted_3k       1810          811
##                  RNA_snn_res.0.6 seurat_clusters assign.ident assign.score
## AAACAGCCAAATATCC               8               8           NK     7.566970
## AAACAGCCAGGAACTG               3               3   Mono/Macro     4.158901
## AAACAGCCAGGCTTCG               1               1   Mono/Macro     3.892433
## AAACCAACACCTGCTC               4               4            B     7.929692
## AAACCAACAGATTCAT               5               5       CD8Tex     5.373344
## AAACCAACAGTTGCGT               0               0     CD4Tconv     2.388740
##                  percent.mt
## AAACAGCCAAATATCC   13.16568
## AAACAGCCAGGAACTG   21.51347
## AAACAGCCAGGCTTCG   42.66332
## AAACCAACACCTGCTC   26.42663
## AAACCAACAGATTCAT   18.32858
## AAACCAACAGTTGCGT   15.58011
```

Visualize the distribution of `nFeature_RNA`, `nCount_RNA` and `percent.mt` in a single violin plot (1 pts;)

```
library(cowplot)
plots <- VlnPlot(
  rna,
  features = c("nFeature_RNA", "nCount_RNA", "percent.mt"),
  combine = FALSE
)

themed_plots <- list()
for (i in seq_along(plots)){
    #Change x and y tick label font size.
    themed_plots[[i]] = plots[[i]] +
      scale_x_discrete(labels = c()) +
      theme(legend.position = "none")

}

cowplot::plot_grid(plotlist = themed_plots, ncol = 3) +
  ggtitle("Distribution of QC Values")
```



Low-quality cells or empty droplets will have few genes, or low values in nCount. Cell doublets or multi

Low quality or dying cells have high mitochondrial contamination, or high values of percent.mt. Mitochon

Use a table to show how many of the cells have mitochondrial rate > 20% (1 pts;)

```
table(rna@meta.data$percent.mt > 20)
```

```
##
## FALSE  TRUE
##  1296  1337
```

```
table(rna@meta.data$percent.mt > 20)/length(rna@meta.data$percent.mt)
```

```
##
##     FALSE      TRUE
## 0.4922142 0.5077858
```

Answer:

1337 of the 2633 cells have mitocondrial rates over 20%. This is 50.78%, or about half, of the cells.

3. After removing unwanted cells from the dataset (already done by MAESTRO. No need to filter in this homework), the next step is to normalize the data. Please use the default settings in Seurat to do the normalization: `NormalizeData()` (1 pts;). We next calculate a subset of features that exhibit high cell-to-cell variation in the data set. These features will be used for downstream analysis to reduce computing time. Please use `FindVariableFeatures(..., selection.method = "vst", nfeatures = 2000)` to return the top 2,000 variable features (1 pts;). Next, we will apply a linear transformation (also known as scaling), a standard pre-processing step prior to dimensional reduction techniques. Please perform scaling on the top 2,000 variable genes (default) using `ScaleData()` (1 pts;).

Use the default settings in Seurat to do the normalization: `NormalizeData()` (1 pts;)

```
rna_norm <- NormalizeData(rna)
```

Use `FindVariableFeatures(..., selection.method = "vst", nfeatures = 2000)` to return the top 2,000 variable features (1 pts;)

```
rna_feature_select <- FindVariableFeatures(rna_norm, selection.method = 'vst', nfeatues = 2000)
```

Perform scaling on the top 2,000 variable genes (default) using `ScaleData()` (1 pts;).

```
rna_scaled <- ScaleData(rna_feature_select)

# results stored in rna_scaled$RNA@scale.data
```

4. Perform linear dimensional reduction. Next, we'll perform PCA on the scaled data. Please use `RunPCA()` to do the dimension reduction on the 2,000 variable genes we detected in the previous question (1 pts;). You can take a look at PCA cell embeddings at `rna[['pca']]@cell.embeddings`.

```
# Default npcs = 50
rna_pca <- RunPCA(rna_scaled)

# results stored in rna_pca$pca@cell.embeddings
```

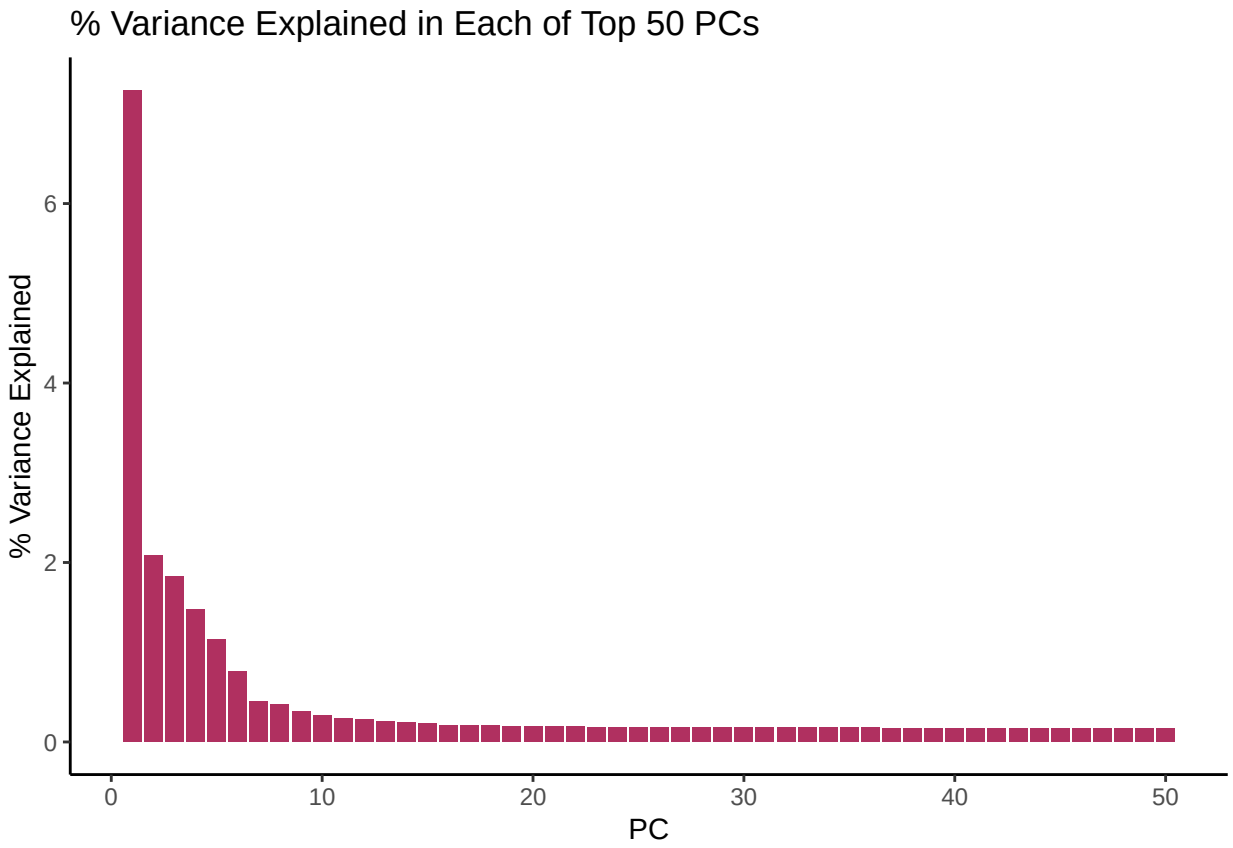
5. Since not all the PCs we calculated will be used in downstream analysis. In this question, We will determine how many PCs we should use in downstream analysis:

- **5.1 How much variability is explained in each of the first 50PCs? Please show a scree plot with each PCs on the x-axis and variation explained by each PC on the y-axis (1 pts;).**
- **hint** : https://github.com/satijalab/seurat/issues/982

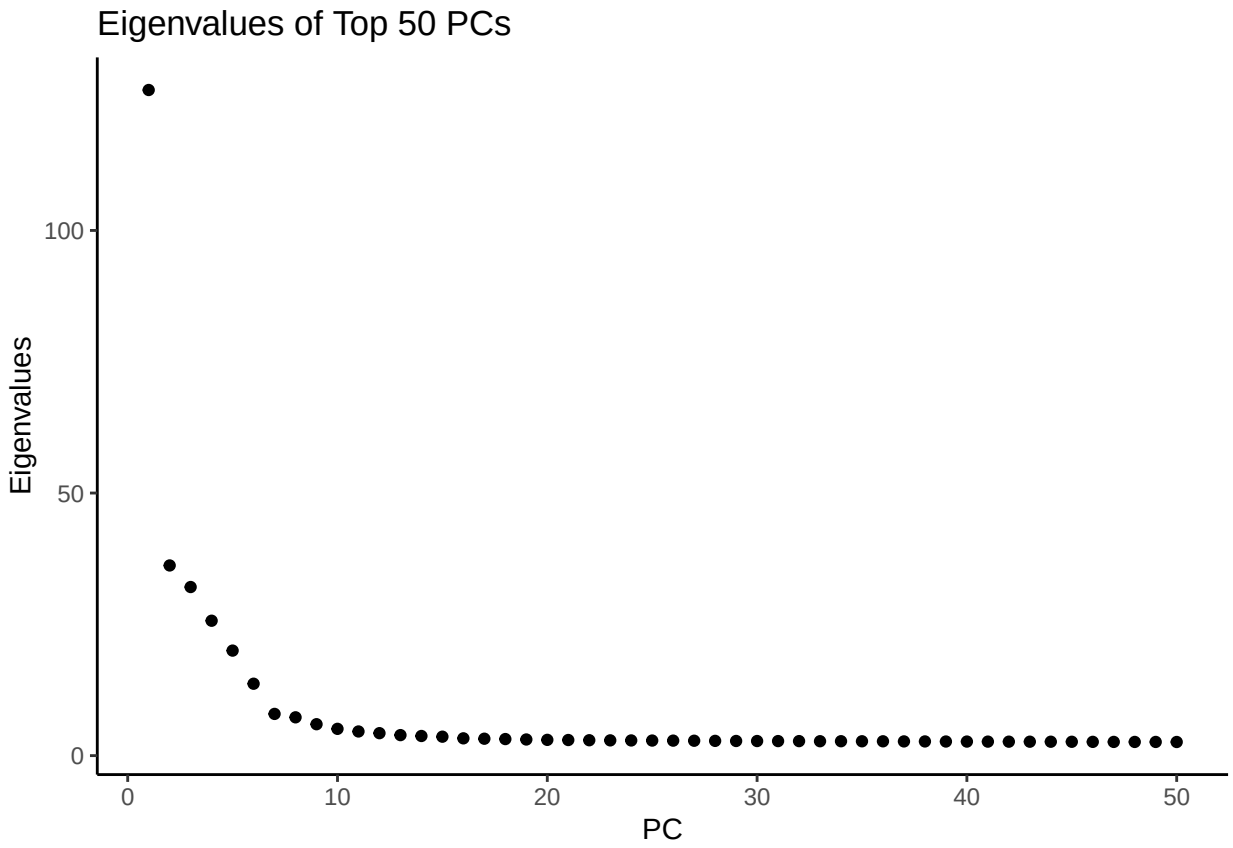
```
pca <- rna_pca$pca
eigValues = (pca@stdev)^2

# Because default npcs = 50, sum(eigValues) is NOT total variance,
# just variance accounted for by first 50 PCs.
mat <- Seurat::GetAssayData(rna_pca, assay = "RNA", slot = "scale.data")
total_variance <- sum(matrixStats::rowVars(mat))

varExplained = eigValues / total_variance

sum(varExplained)
```

```
## [1] 0.2286528
```

```
data.frame(varExplained = varExplained*100, PC = 1:50) %>%
  ggplot(aes(x = PC, y = varExplained)) +
  geom_bar(stat = 'identity', fill = 'maroon') +
  ylab("% Variance Explained") +
  theme_classic() +
  ggtitle("% Variance Explained in Each of Top 50 PCs")
```




```
data.frame(Eigenvalues = eigValues, PC = 1:50) %>%
  ggplot(aes(x = PC, y = Eigenvalues)) +
  geom_point(fill = 'maroon') +
  theme_classic() +
  ggtitle("Eigenvalues of Top 50 PCs")
```

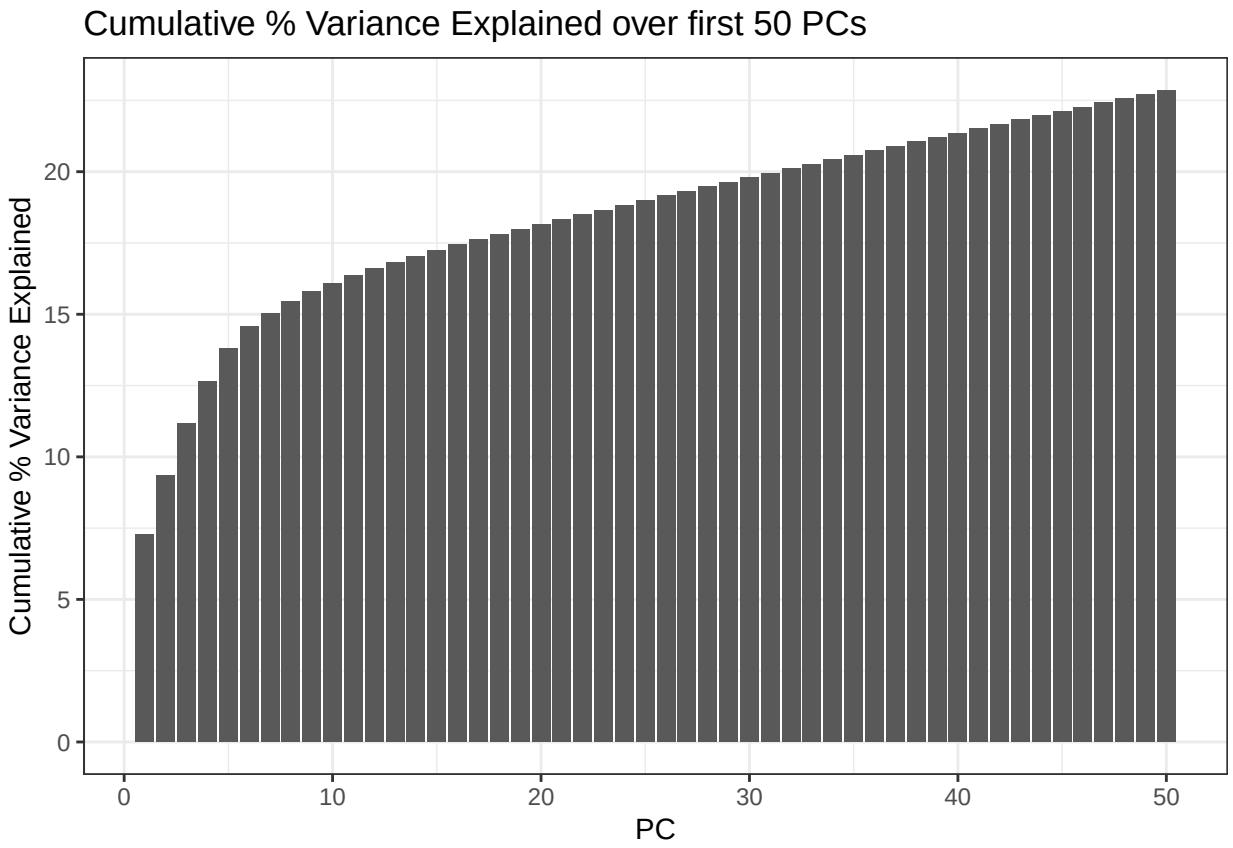


Answer:

```
22.87% of variance is explained in the first 50 PCs.
```

- **5.2 How many PCs do you need to cover 20% of the total variance? Show a bar plot with PCs on the x-axis and the cumulative sum of variance explained by each PC on the y-axis (1 pts;). In bulk RNA-seq experiments, can you recall how many PCs you need to explain 20% of the variability? What do you think is the main reason that causes such difference between single-cell and bulk RNA-seq (Graduate level 2 pts;)?**

```
data.frame(cumsum = 100*cumsum(varExplained), PC = 1:50) %>%
  ggplot(aes(x = PC, y = cumsum)) +
  geom_bar(stat = 'identity') +
  ylab("Cumulative % Variance Explained") +
  ggtitle("Cumulative % Variance Explained over first 50 PCs") +
  theme_bw()
```




```
howmany <- which(cumsum(varExplained) > 0.2)[1]

print(howmany)
```

```
## [1] 32
```

```
sum(varExplained[1:howmany])
```

```
## [1] 0.2011086
```

Answer:

```
32 principal components are needed to explain 20% of the variability. The top 32 PCs explain 20.1% of t

In earlier homeworks, we found that many fewer PCs would explain the same amount of variance. The first

This is likely because bulk RNA-seq is basically the average expression over a group of cells, instead
```

- **5.3 What are The top 5 genes with the most positive and negative coefficients in each of the first 10 PCs? Use a table to show the results (1 pts;).**

```
library(kableExtra)

top_5 <- function(PC_name) {
  neg <- rna_pca$pca@feature.loadings %>%
    data.frame() %>%
    select(PC_name) %>%
    arrange(!!rlang::sym(PC_name)) %>%
    rownames() %>%
    head(5) %>%
```

Table 1: Top 5 genes with the most positive and negative coefficients in each of the first 10 PCs

PC	negative	positive
PC_1	RPS27, RPS27A, RPS29, RPS3, IL32	TYMP, FCN1, LYZ, SAT1, TNFAIP2
PC_2	HLA-DPA1, HLA-DPB1, HLA-DQA1, HLA-DRB5, PLD4	AC020916.1, CSF3R, VCAN, MT-RNR2, FOSB
PC_3	S100A4, NKG7, GZMA, GNLY, PRF1	MS4A1, IGHM, CD79A, BANK1, RALGPS2
PC_4	GZMB, GNLY, PRF1, NKG7, CST7	EEF1A1, IL7R, LTB, RPS6, NAP1L1
PC_5	LILRA4, CLEC4C, LINC00996, SERPINF1, DNASE1L3	MS4A1, CD79B, CD79A, BANK1, PAX5
PC_6	CDKN1C, FCGR3A, HES4, SMIM25, TCF7L2	CD1C, GAPDH, FCER1A, VCAN, S100A8
PC_7	FCER1A, CLEC10A, CD1C, NDRG2, FLT3	ITGB1, S100A4, CD82, MAF, CRIP1
PC_8	S100A12, S100A8, CD14, FTL, S100A9	CD1C, FCER1A, CLEC10A, ITGB1, NDRG2
PC_9	GBP1, GBP5, SERPING1, IFI44L, HERC5	CDKN1C, PADI4, SLC2A3, RHOC, CES1
PC_10	SH2D1B, IGFBP7, TNFRSF18, KLRF1, CLIC3	CD8A, TRGC2, LAG3, GZMK, CCL5

```
    paste(collapse = ', ')

  pos <- rna_pca$pca@feature.loadings %>%
    data.frame() %>%
    select(PC_name) %>%
    arrange(-!!rlang::sym(PC_name)) %>%
    rownames() %>%
    head(5) %>%
    paste(collapse = ', ')

  list(
    "pc" = PC_name,
    "neg" = neg,
    "pos" = pos
  )
}

out <- data.frame(
  "PC" = rep("", 10),
  "negative" = rep("", 10),
  "positive" = rep("", 10)
)
for (i in 1:10) {
  PC_name <- paste("PC_", i, sep = "")
  out[i,] <- top_5(PC_name)
}

kbl(out, caption = "Top 5 genes with the most positive and negative coefficients in each of the first 10
  kable_classic(full_width = F, html_font = "Cambria")
```

6. Umap Visualization: Dimensionality reduction is a powerful tool for machine learning practitioners to visualize and understand large, high-dimensional datasets. Seurat offers several non-linear dimension reduction techniques, such as tSNE and UMAP (as opposed to PCA which is a linear dimensional reduction technique). UMAP is a new technique by McInnes et al. that offers many advantages over t-SNE, most notably increased speed and better preservation of the data's global structure.

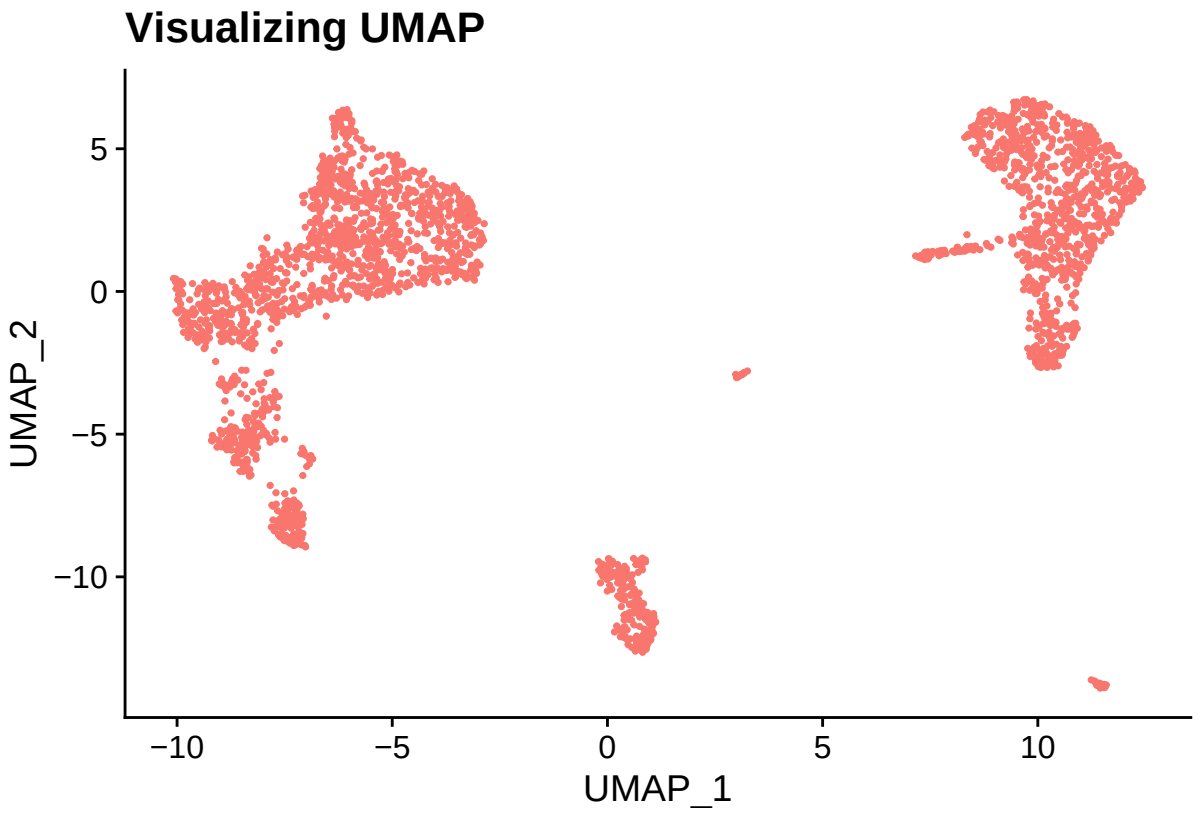
- **6.1 Use the first 15 PCs, apply `RunUMAP()` to perform non-linear dimension reduction for visualization. The results will be stored in `rna[['umap]]` (1 pts;).**

```
rna_umap <- RunUMAP(rna_pca, dims = 1:15)
rna_umap[['umap']]
```

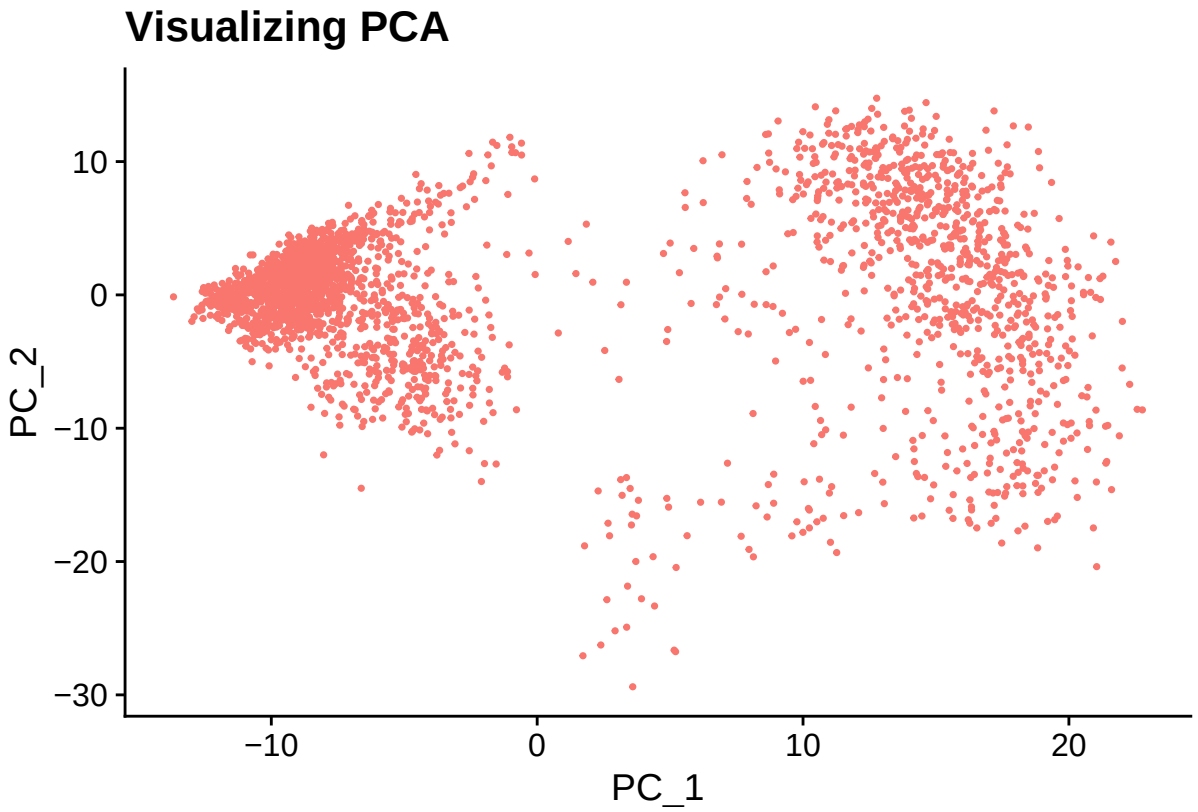
```
## A dimensional reduction object with key UMAP_
##  Number of dimensions: 2
##  Projected dimensional reduction calculated:  FALSE
##  Jackstraw run: FALSE
##  Computed using assay: RNA
```

- **6.2 Please Visualize the cells on the PCA and UMAP embeddings individually (2 pts;) and comment on the number of cell clusters that appear in each plot (1 pts;). hint: Use `DimPlot()`. Describe the difference between PCA and UMAP on 2D plots (2 pts;).**

```
DimPlot(rna_umap, reduction = "umap") +
  theme(legend.position = 'none') +
  ggtitle("Visualizing UMAP")
```




```
DimPlot(rna_pca, reduction = "pca") +
  theme(legend.position = 'none') +
  ggtitle("Visualizing PCA")
```



Answer:

The visualization for UMAP seems to have three discrete, large clusters. However, the cluster on the le

The visualization for PCA has two distinct clusters, and many points in between that are ambigous as to

7. Clustering: Seurat v3 applies a graph-based clustering approach, building upon initial strategies from Macosko et al.. To cluster the cells, we first construct a KNN graph based on the euclidean distance in PCA space and refine the edge weights between any two cells based on the shared overlap in their local neighborhoods (Jaccard similarity).

- **7.1 Use the `FindNeighbors()` function and take first 15 PCs as input to perform KNN (1 pts;).**

```
rna_neighbors <- FindNeighbors(rna_umap, dims = 1:15)
rna_neighbors
```

```
## An object of class Seurat
## 14251 features across 2633 samples within 1 assay
## Active assay: RNA (14251 features, 2000 variable features)
##  2 dimensional reductions calculated: pca, umap
```

- **7.2 We next apply modularity optimization techniques to iteratively group cells together. Use `FindClusters()` function with different `resolution` to perform clustering and draw the resulting clusters in different colors on UMAP (`resolution` = 0.4, 0.6, 0.8) (3 pts;).**

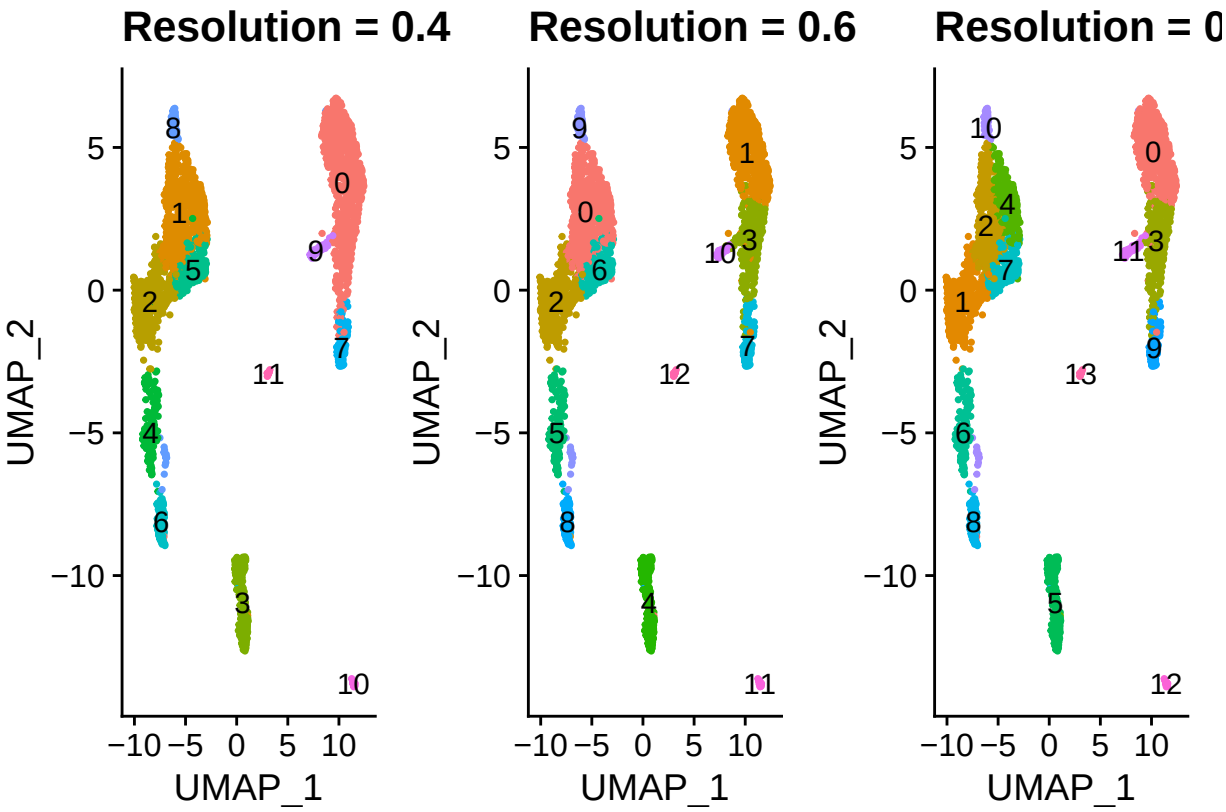
```
rna_clusters <- FindClusters(rna_neighbors, resolution = c(0.4, 0.6, 0.8))
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2633
## Number of edges: 90925
```

```
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8982
## Number of communities: 12
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2633
## Number of edges: 90925
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8700
## Number of communities: 13
## Elapsed time: 0 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2633
## Number of edges: 90925
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8477
## Number of communities: 14
## Elapsed time: 0 seconds
```

Draw the resulting clusters in different colors on UMAP

```
Idents(object = rna_clusters) <- "RNA_snn_res.0.4"
p1 = DimPlot(rna_clusters, reduction = "umap", label = TRUE) +
  theme(legend.position = 'none') +
  ggtitle("Resolution = 0.4")

Idents(object = rna_clusters) <- "RNA_snn_res.0.6"
p2 = DimPlot(rna_clusters, reduction = "umap", label = TRUE) +
  theme(legend.position = 'none') +
  ggtitle("Resolution = 0.6")

Idents(object = rna_clusters) <- "RNA_snn_res.0.8"
p3 = DimPlot(rna_clusters, reduction = "umap", label = TRUE) +
  theme(legend.position = 'none') +
  ggtitle("Resolution = 0.8")

library(patchwork)
p1 + p2 + p3
```



Description:

Moving from 12 clusters to 13 clusters, the top right cluster cluster 0 from the left plot splits into

- **7.3 How does resolution influence the number of clusters and the number of cells assigned to each cluster? Please provide a table to show the number of cells in each cluster (Graduate 1 pts;). Is there a correct number of clusters in a particular data set? why or why not (Graduate level 1pts;)?**

```
dat <- data.frame(
  'Resolution' = c(0.4, 0.6, 0.8),
  'Number of Clusters' = c(12,13,14),
  '0' = rep(0, 3), '1' = rep(0, 3), '2' = rep(0, 3), '3' = rep(0, 3),
  '4' = rep(0, 3), '5' = rep(0, 3), '6' = rep(0, 3), '7' = rep(0, 3),
  '8' = rep(0, 3), '9' = rep(0, 3), '10' = rep(0, 3), '11' = rep(0, 3),
  '12' = rep(0, 3), '13' = rep(0, 3)
)

dat[1,] <- c(0.4, 12, table(rna_clusters@meta.data$RNA_snn_res.0.4), '-', '-')
dat[2,] <- c(0.6, 13, table(rna_clusters@meta.data$RNA_snn_res.0.6), '-')
dat[3,] <- c(0.8, 14, table(rna_clusters@meta.data$RNA_snn_res.0.8))

colnames(dat) <- c('Resolution', 'nClusters', 0:13)

kbl(dat, caption = "Number of Cells Assigned to Each Cluster") %>%
  kable_classic(full_width = F, html_font = "Cambria")
```

Answer:

With increased resolution, there are more clusters. Resolution of 0.4 gave us 12 clusters, 0.6 gave us

Table 2: Number of Cells Assigned to Each Cluster

Resolution	nClusters	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0.4	12	748	596	380	193	179	163	107	104	60	59	27	17	-	-
0.6	13	593	475	379	276	193	180	166	112	107	60	48	27	17	-
0.8	14	472	377	377	276	227	193	180	157	107	104	60	59	27	17

There is no correct number of clusters in a particular data set because we do not know the underlying s

8. Find Cluster Markers: For further analysis, please keep using resolution = 0.6 to cluster the cells. Seurat has a function called `FindMarkers()` that can perform several tests to identify differentially expressed genes for a single cluster.

- **8.1 Use Wilcox Rank Sum test (default) in `FindMarkers(..., min.pct = 0.25)` to identify all markers of cluster 5 (1 pts;). Print the top5 markers in cluster 5. hints: You need to rerun `FindClusters()` with resolution = 0.6 to get correct number of cell clusters.**

```
rna_clusters_res6 <- FindClusters(rna_neighbors, resolution = 0.6)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2633
## Number of edges: 90925
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8700
## Number of communities: 13
## Elapsed time: 0 seconds
```

```
Idents(object = rna_clusters_res6) <- "RNA_snn_res.0.6"
rna_markers <- FindMarkers(rna_clusters_res6, min.pct = 0.25, ident.1 = 5)
rna_markers %>%
  arrange(-avg_log2FC) %>%
  head(5)
```

```
##              p_val avg_log2FC pct.1 pct.2     p_val_adj
## CCL5 5.225108e-219   3.770360 0.978 0.118 7.446301e-215
## NKG7 5.037707e-173   3.103419 0.950 0.145 7.179237e-169
## GZMH 4.429950e-186   3.017419 0.617 0.035 6.313122e-182
## GZMA 5.284179e-185   2.528293 0.850 0.086 7.530484e-181
## GNLY 9.258060e-102   2.470035 0.711 0.124  1.319366e-97
```

Answer:

The top 5 markers by avgerage log2FC in cluster 5 are, in order, CCL5, NKG7, GZMH, GZMA, and GNLY.

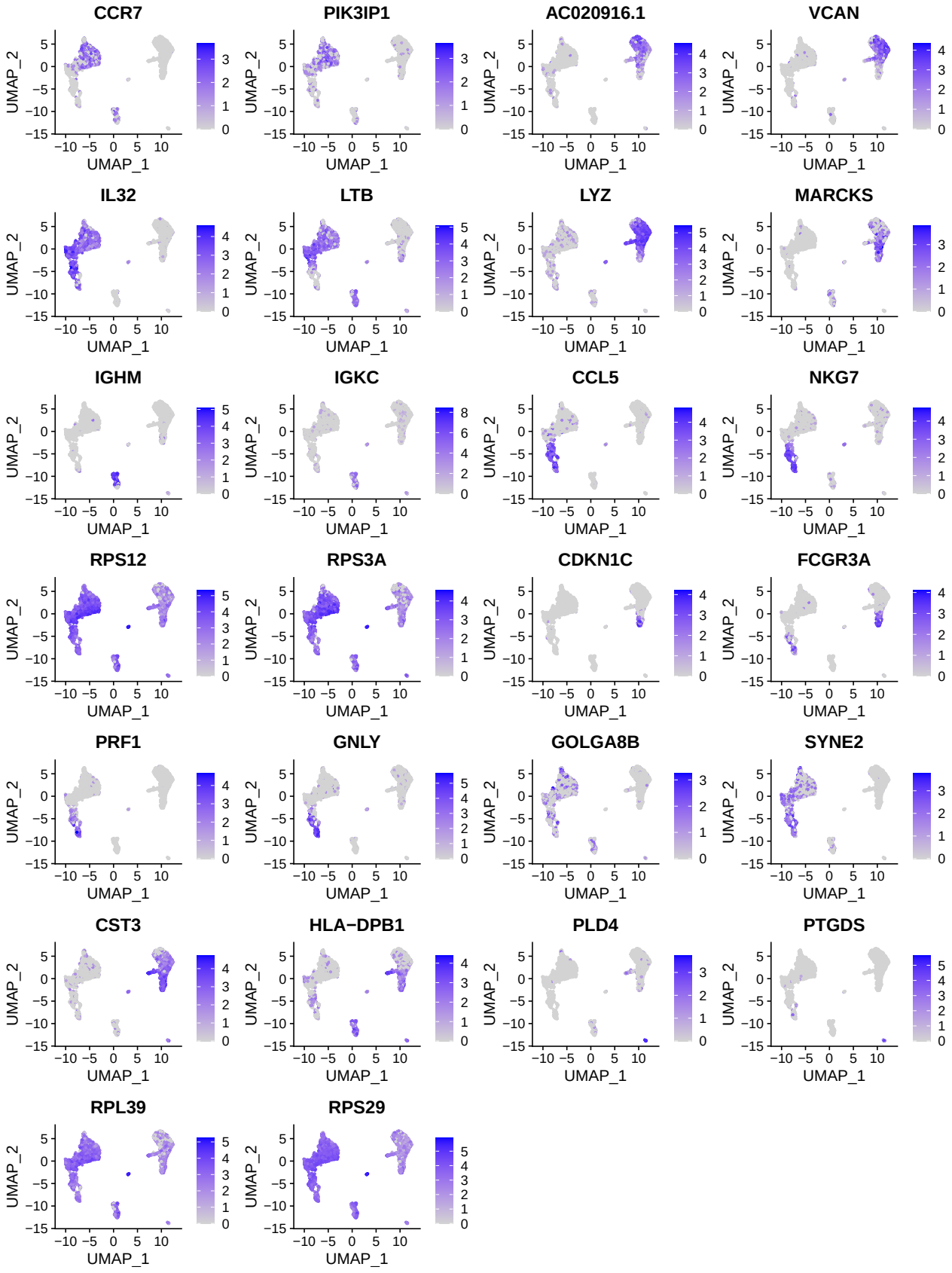
- **8.2 `FindAllMarkers()` function can find markers for every cluster compared to all remaining cells (one vs. the rest). Apply `FindAllMarkers()` function with `min.pct = 0.25` and `logfc.threshold = 0.25` to report only positive genes in each cluster (1 pts;). Please print top 2 markers in each cluster weighted by log2FC (1pts;).**

```
rna_all_markers <- FindAllMarkers(rna_clusters_res6, only.pos = TRUE, min.pct = 0.25, logfc.threshold =
top2 <- rna_all_markers %>%
  group_by(cluster) %>%
  top_n(n = 2, wt = avg_log2FC)
top2
```

```
## # A tibble: 26 x 7
## # Groups:   cluster [13]
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 6.50e-132       1.77 0.659 0.18  9.26e-128 0       CCR7
##  2 2.58e-102       1.54 0.632 0.23  3.67e- 98 0       PIK3IP1
##  3 9.77e-259       3.07 0.901 0.189 1.39e-254 1       AC020916.1
##  4 2.36e-241       2.78 0.872 0.168 3.37e-237 1       VCAN
##  5 4.51e-124       1.68 0.987 0.474 6.42e-120 2       IL32
##  6 2.41e-113       1.79 0.974 0.527 3.44e-109 2       LTB
##  7 1.97e-133       2.42 0.996 0.473 2.80e-129 3       LYZ
##  8 4.87e-119       2.06 0.793 0.19  6.94e-115 3       MARCKS
##  9 1.20e-307       5.37 0.86  0.043 1.70e-303 4       IGHM
## 10 2.95e-102       5.43 0.736 0.181 4.20e- 98 4       IGKC
## # ... with 16 more rows
```

- **8.3 Visualize the gene expression values of these potential markers (top2) on your UMAP plots using `FeaturePlot()` (2 pts;).**

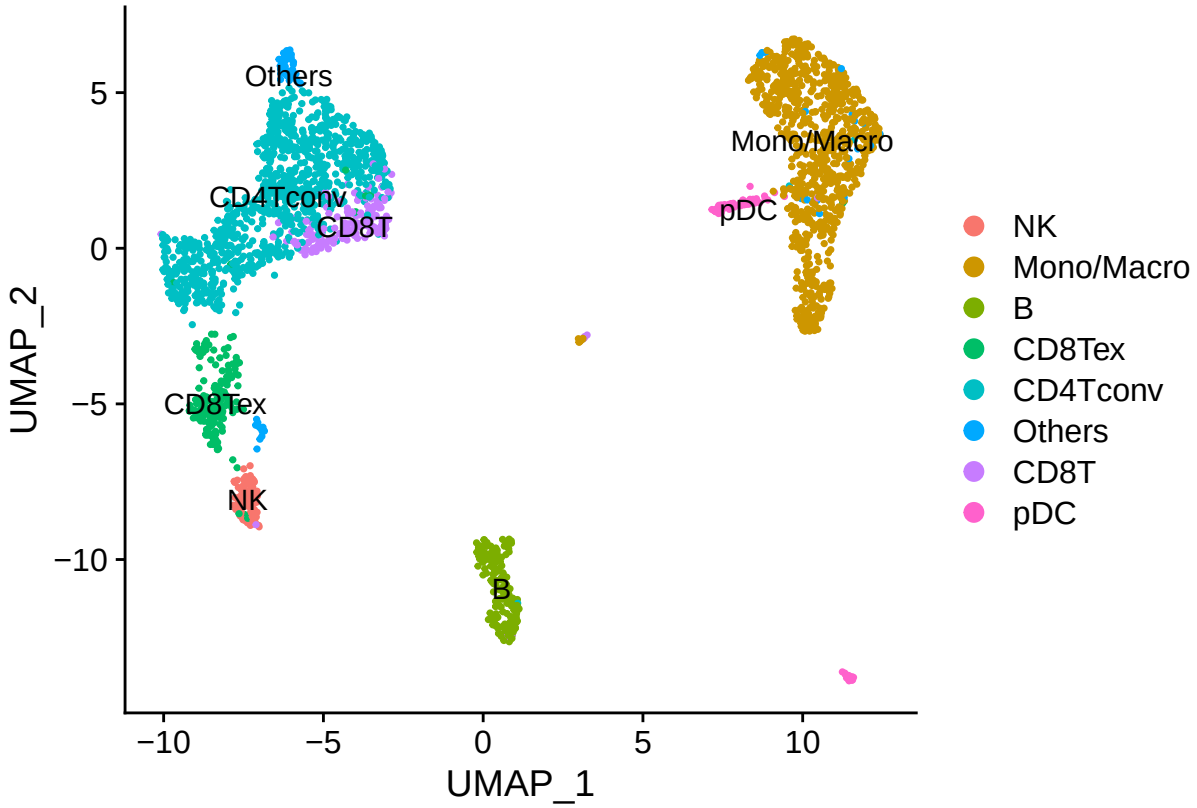
```
FeaturePlot(rna_clusters_res6, features = unique(top2$gene))
```



9. Annotation: Based on markers in each cluster, MAESTRO referenced human.immune.CIBERSORT

data to do the cell type annotations. The annotated cell types are stored in the `assign.ident` column in Seurat metadata. Please set `Idents(rna) <- rna$assign.ident` and plot the cell type annotation results in UMAP (1 pts;)

```
Idents(rna_clusters_res6) <- rna_clusters_res6@meta.data$assign.ident
DimPlot(rna_clusters_res6, reduction = 'umap', label = TRUE)
```



Part III. Single-cell ATAC-seq

- Create a scATAC-seq Seurat Object
- Data Normalization and Dimension Reduction
- Clustering
- Cell Type Annotation
- Finding Differentially Accessible Peaks

Please install and load following the packages:

```
library(data.table)
library(dplyr)
library(Seurat)
library(Signac)
library(ggplot2)
```

1. Read the multiome scATAC-seq Seurat object: Same as scRNA-seq .rds file, we need to extract Seurat object from .rds data list. How many assays are stored in this Seurat object and what is the current active assay (1 pts;)? Now switch the default assay to 'ATAC' using `DefaultAssay(atac) <- 'ATAC'`. How many cells and peaks are retained in the peak count matrix (1 pts;)? Important note: Please use `Idents(atac) <- atac$orig.ident` to update the cell ID before moving to Question 2.

Extract Seurat object from .rds data list

```
atac<- readRDS("scatac_pbmc_granulocyte_sorted_3k_scATAC_Object.rds")
atac<-atac$ATAC

Idents(atac) <- atac$orig.ident
atac
```

```
## An object of class Seurat
## 72599 features across 2664 samples within 2 assays
## Active assay: ACTIVITY (28307 features, 2000 variable features)
##  1 other assay present: ATAC
##  2 dimensional reductions calculated: lsi, umap
```

How many assays are stored in this Seurat object and what is the current active assay (1 pts;)

Answer:

```
The current active assay is ACTIVITY. There is another assay present, ATAC, so there are two total assay
```

Switch the default assay to 'ATAC' using `DefaultAssay(atac) <- 'ATAC'`. How many cells and peaks are retained in the peak count matrix (1 pts;)

```
DefaultAssay(atac) <- 'ATAC'

atac
```

```
## An object of class Seurat
## 72599 features across 2664 samples within 2 assays
## Active assay: ATAC (44292 features, 44292 variable features)
##  1 other assay present: ACTIVITY
##  2 dimensional reductions calculated: lsi, umap
```

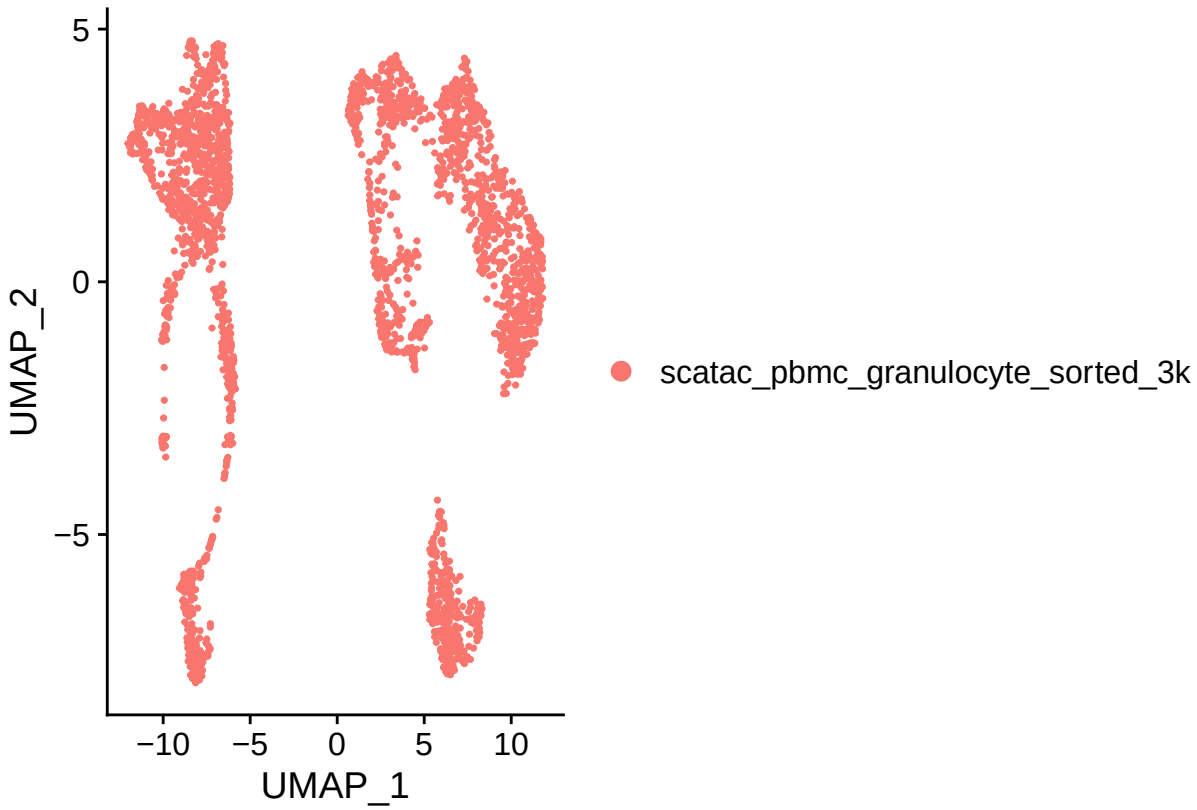
Answer:

```
There are 44292 peaks (44292 features) across 2664 cells (2664 samples).
```

2. Normalization and Linear Dimensional Reduction: scATACseq data are very sparse. It is sparser than scRNAseq. Thus, It is necessary to do pre-processing steps before clustering scATACseq data. Signac performs a two-step normalization method called TF-IDF (term frequency-inverse document frequency). To get rid of the low dynamic range of scATAC-seq data, we further use `FindTopFeatures()` to only choose the top n% of features (peaks) for dimension reduction. Next, we run SVD (singular value decomposition) on the normalized TD-IDF matrix for linear dimensional reduction. The combined steps of TF-IDF followed by SVD are also known as LSI (latent semantic indexing).

- **2.1: Before performing normalization, let's plot a UMAP on the first 15 PCs to see how the plot looks(1 pts;). hints: Same as we have done in scRNA-seq analysis, do `ScaleData()` and `RunPCA()` (Running PCA will take some time), then `RunUMAP(..., reduction = 'pca', dims =1:15)`. Plot the UMAP. Does this UMAP look good? Leave a brief comment on what you have observed (1 pts;). We will perform normalization later and you can compare their differences.**

```
atac_scale <- ScaleData(atac)
atac_pca <- RunPCA(atac_scale)
atac_umap <- RunUMAP(atac_pca, reduction = 'pca', dims =1:15)
DimPlot(atac_umap, reduction = "umap")
```



Does this UMAP look good? Leave a brief comment on what you have observed (1 pts;).

Answer:

The clustering using UMAP is less defined than it was in part II of this homework using scRNA-seq. There

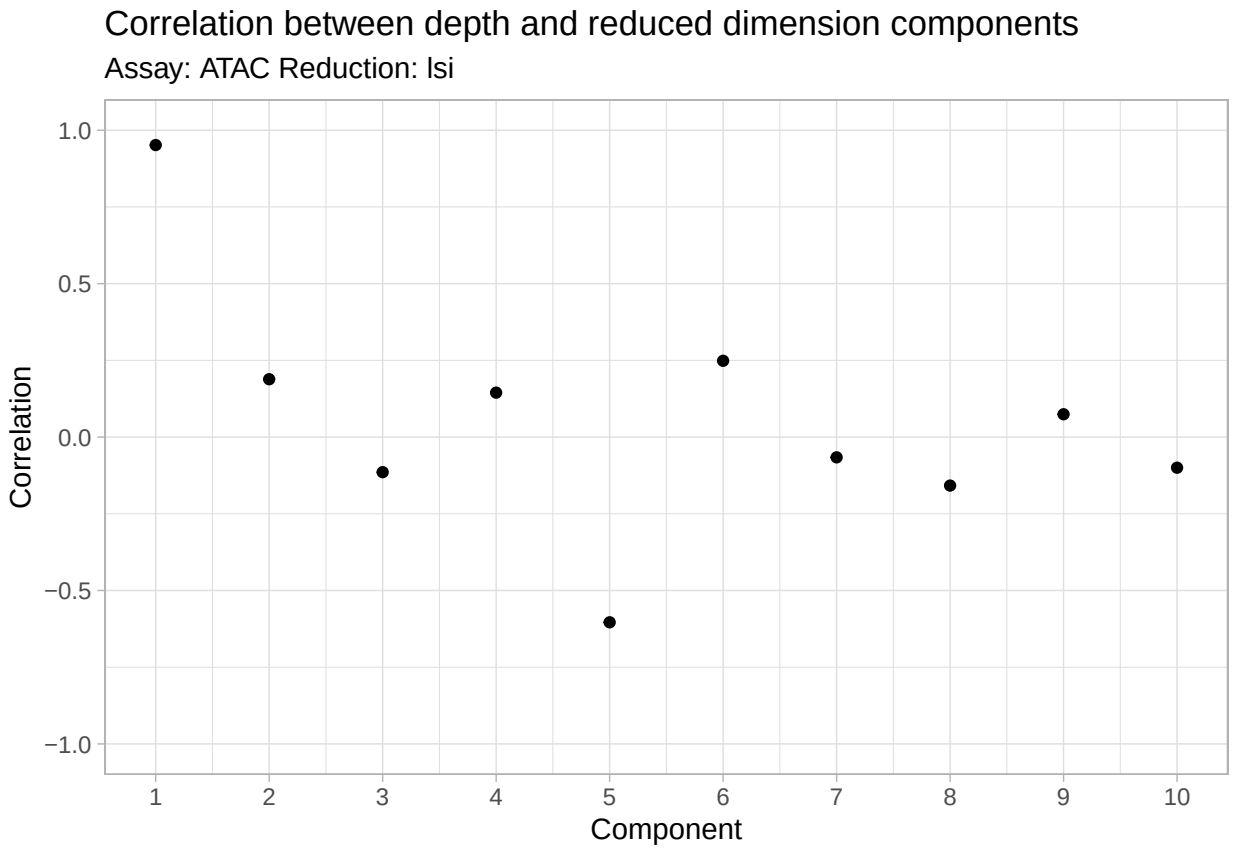
- **2.2: Let's apply `RunTFIDF()`, `FindTopFeatures(..., min.cutoff = 'q0')`, and `RunSVD()` sequentially to do the normalization and linear dimension reduction (1 pts;). After running SVD, you will find the cell imbedding information under `atac[['lsi']]`**

```
atac_tfidf <- RunTFIDF(atac_umap)
atac_top_features <- FindTopFeatures(atac_tfidf, min.cutoff = 'q0')
atac_svd <- RunSVD(atac_top_features)

atac_svd[['lsi']]
```

```
## A dimensional reduction object with key LSI_
##  Number of dimensions: 50
##  Projected dimensional reduction calculated:  FALSE
##  Jackstraw run: FALSE
##  Computed using assay: ATAC
```

- **2.3: Let's use `DepthCor()` to measure the correlation of sequencing depth with each LSI component (1 pts;). Did you observe any strong correlation in this plot (1 pts;)?**

```
DepthCor(atac_svd)
```



Did you observe any strong correlation in this plot (1 pts;)?

Answer:

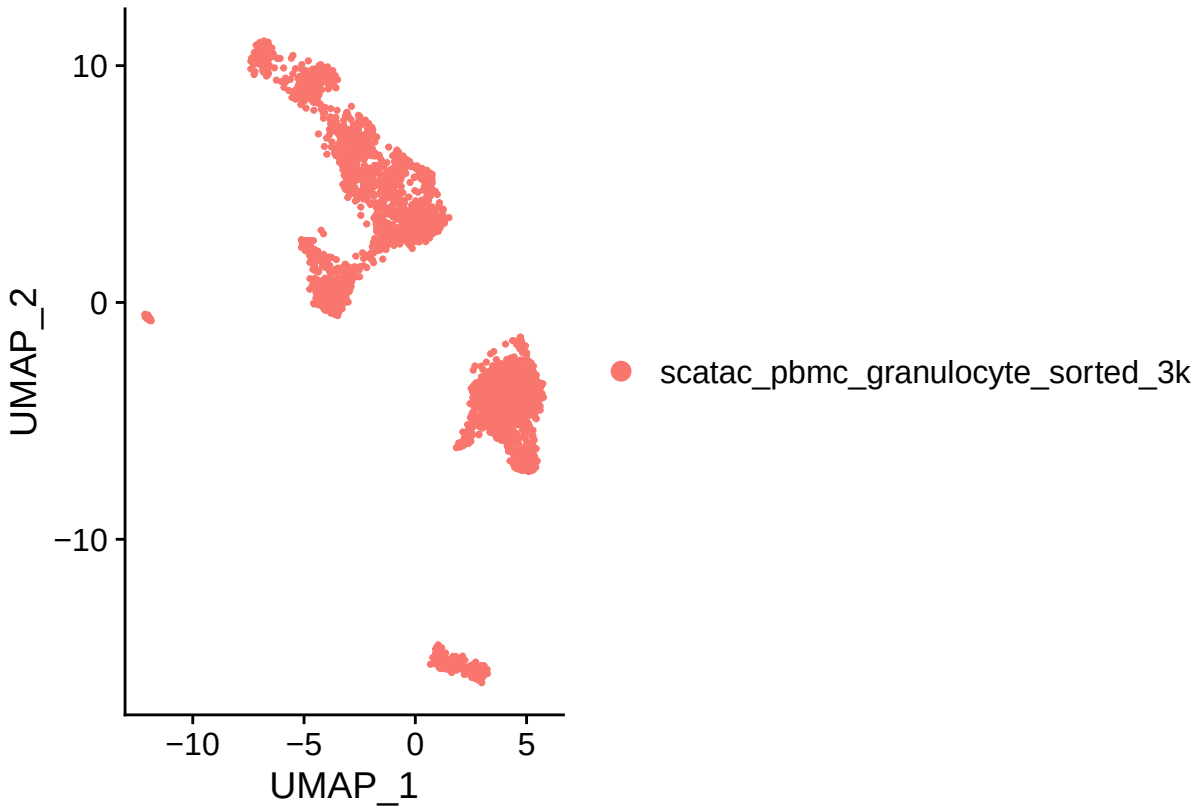
There is a very high positive correlation between total counts and dimension component 1 (correlation c

3. Non-Linear Dimension Reduction and Clustering: Like scRNA-seq analysis, we will project the cell on a 2D plot using UMAP and do the clustering.

- **3.1 Run UMAP dimension reduction on 2:30 LSI components (1 pts;) and plot the UMAP on a 2D graph (1 pts;). Compared to the UMAP you got from question 2.1 (without normalization), does this UMAP looks better? Leave your comment below (1 pts;).**

Run UMAP dimension reduction on 2:30 LSI components (1 pts;) and plot the UMAP on a 2D graph (1 pts;)

```
atac_umap_lsi <- RunUMAP(atac_svd, reduction = 'lsi', dims =2:30)
DimPlot(atac_umap_lsi, reduction = "umap")
```



does this UMAP looks better (1 pts;)

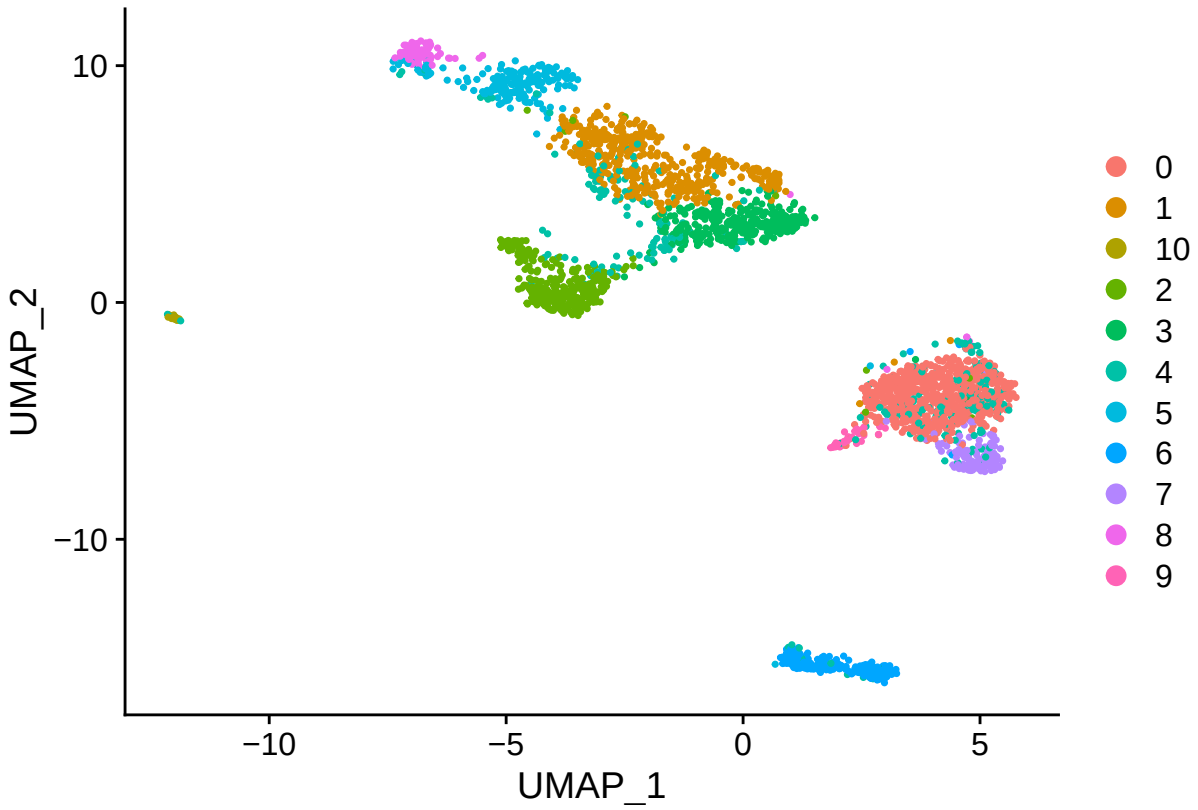
Answer: `This UMAP looks better. The cluster boundaries are much more clearly defined.`

- **3.2 Use the same dimension (reduction = 'lsi', dims = 2:30) to perform KNN `FindNeighbors()`, cluster the cells with `resolution = 0.6` and `algorithm = 3` (1 pts;), and visualize the clustering results on a UMAP embedding (1 pts;). How many clusters did you get (1 pts;)?**

```
atac_knn <- FindNeighbors(atac_umap_lsi, reduction = 'lsi', dims = 2:30)
atac_clusters <- FindClusters(atac_knn, resolution = 0.6, algorithm = 3)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 2664
## Number of edges: 115342
##
## Running smart local moving algorithm...
## Maximum modularity in 10 random starts: 0.8361
## Number of communities: 12
## Elapsed time: 0 seconds
```

```
Idents(object = atac_clusters) <- "ATAC_snn_res.0.6"
DimPlot(atac_clusters, reduction = "umap")
```

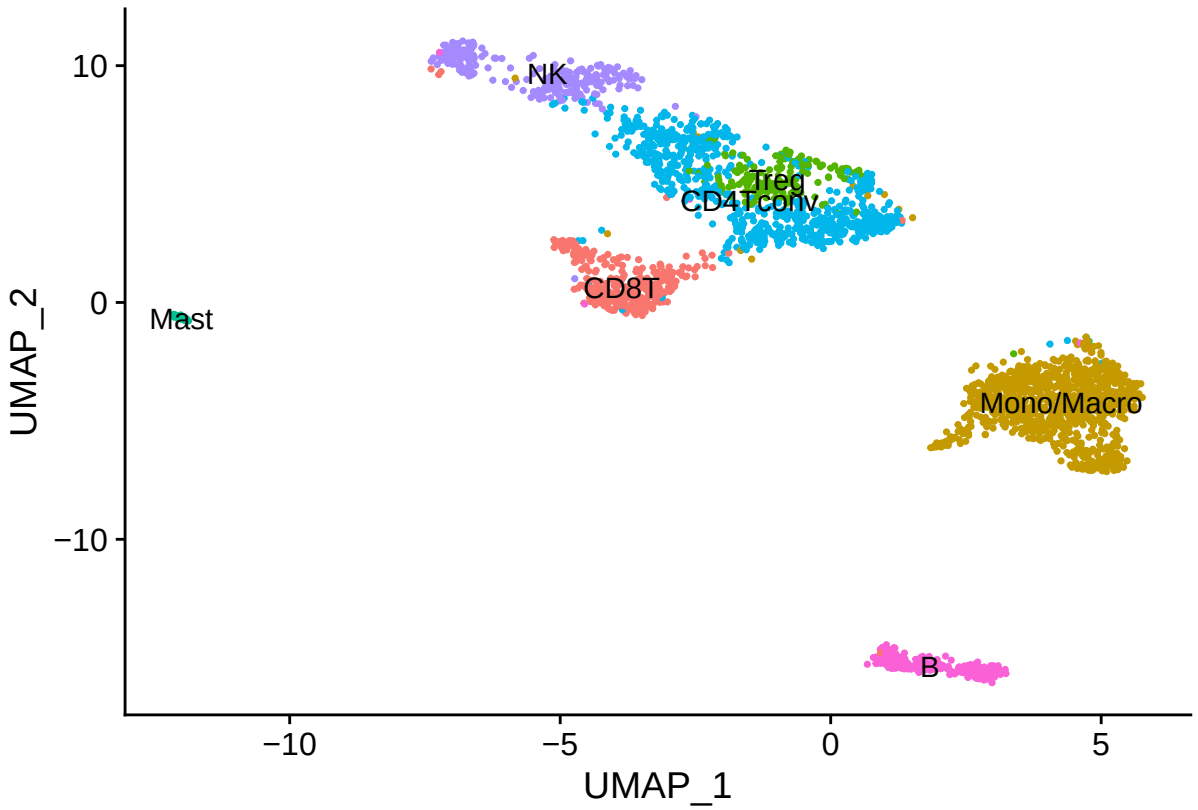


Answer:

I got 11 clusters.

- **3.3 MAESTRO performed cell type annotation and stored the information as `assign.ident` in metadata. Please use `Idents()` to change the cell id as cell type labels and plot the cell annotation in UMAP (1 pts;). How many cell types did you get (1 pts)?**

```
Idents(object = atac_clusters) <- "assign.ident"
DimPlot(atac_clusters, reduction = "umap", label = T) +
  theme(legend.position = 'none')
```



Answer:

There are 7 cell types. They seem to be clustered very well by cell type.

4. Differentially accessible peaks: To find differentially accessible regions between clusters of cells, we can perform a differential accessibility (DA) test.

- **4.1 Let's compare CD8T cells and B cells to find differentially accessible peaks (1 pts;). hint: use `FindMarkers(..., min.pct = 0.2, test.use = 'LR')`. Print the top5 regions differentially expressed between CD8T and B cells (1 pts;).**

```
atac_markers <- FindMarkers(atac_clusters, ident.1 = "CD8T",
                            ident.2 = "B", min.pct = 0.2, test.use = 'LR')

top5 <- atac_markers %>%
  arrange(-avg_log2FC) %>%
  head(5)

top5
```

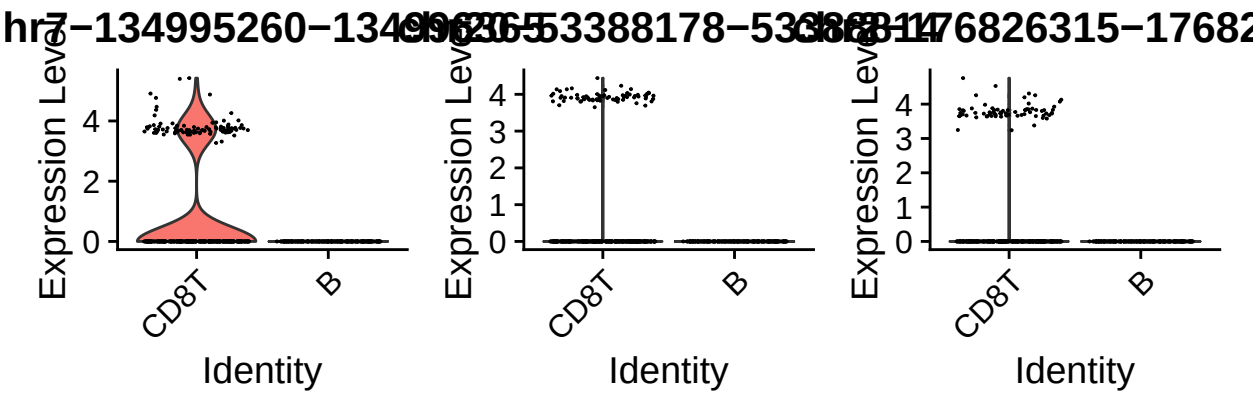
```
##                                 p_val avg_log2FC pct.1 pct.2    p_val_adj
## chr7-134995260-134996365 3.279299e-22   3.799730 0.266 0.000 1.452467e-17
## chr20-53388178-53388814  5.498680e-17   3.502535 0.205 0.000 2.435475e-12
## chr2-176826315-176826938 1.800792e-17   3.383605 0.211 0.000 7.976069e-13
## chr12-10554186-10555392  2.542991e-28   3.262715 0.357 0.005 1.126341e-23
## chr1-8926729-8927922     1.984952e-16   3.230592 0.234 0.010 8.791751e-12
```

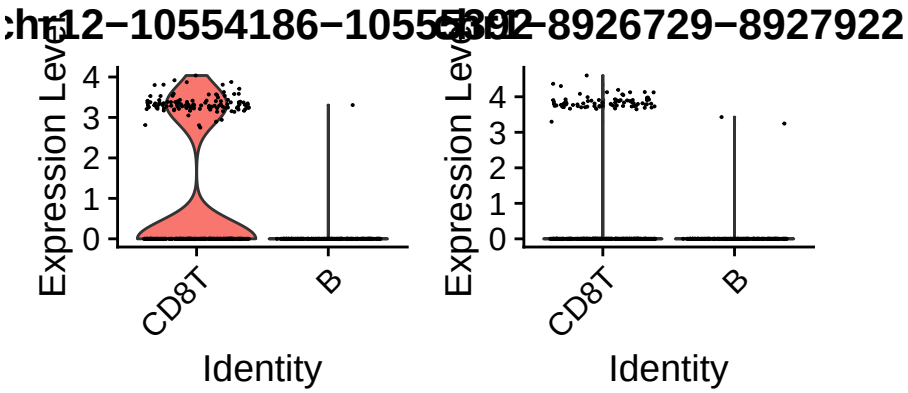
Answer:

The top five regions differentially expressed, sorted by average log fold change, are shown in the table

- **4.2 Use Violin plot to show the first DA peaks in the last question. Pleas show a violin plot with CD8T and B cell types on the x-axis, and expression level on the y-axis (1 pts;).**

```
VlnPlot(
  atac_clusters,
  features = c("chr7-134995260-134996365","chr20-53388178-53388814","chr2-176826315-176826938","chr12-1
  idents = c('CD8T','B')
)
```





5. Based on the peak count matrix (assay 'ATAC'), MAESTRO created a gene activity matrix according to regulatory potential model. This matrix is stored as 'ACTIVITY' assay under Seurat object. Please switch the default assay to 'ACTIVITY' using `DefaultAssay(atac) <- 'ACTIVITY'`, Normalize and scale the gene activity matrix by using `NormalizeData()` (1 pts;). This gene activity matrix will be used for label transferring later.

```
DefaultAssay(atac) <- 'ACTIVITY'
atac_normalized <- NormalizeData(atac)
atac_scaled <- ScaleData(atac_normalized)
```

Part IV. Integrating scRNA-seq and scATAC-seq data

- Integrative analysis of multiome scRNA-seq and scATAC-seq
- Barcodes matching in multiome experiments data
- Label Transferring evaluation

Please install and load following the packages:

```
library(circlize)
library(tidyverse)
```

1. When we pre-processed scRNA and scATAC separately by MAESTRO, some low-

quality cells were filtered out during the analysis (Recall Part I. Question 2). In this Part, let's work on the cells common in single-cell gene expression and ATAC profiles. How are we going to pair the cells if they have different cell barcodes in scRNA and scATAC data (use `colnames()` to explore)? 10X multiome data provides two separate lists of barcodes (barcode whitelist we used for barcode correction in MAESTRO): one for gene expression and another for ATAC. Please download two barcodes lists from `/n/stat115/2021/HW5/references/whitelist/atac/737K-arc-v1.txt` and `/n/stat115/2021/HW5/references/whitelist/rna/737K-arc-v1.txt`to your local computer.

- **1.1 Though listed in two separate files, The positional encoding of the barcodes in two files indicates the pairing, which means they have the same length, and you can match the cells from RNA to ATAC side by side. Now, Please match the scRNA-seq and scATAC-seq barcodes to find the list of common cells (2 pts;). How many cells are in common (1 pts;)?**

```
#Read multiome scatac-seq cell barcodes
atac_whitelist<- read_tsv("atac.737K-arc-v1.txt", col_names = FALSE)

#Read multiome scrna-seq cell barcodes
rna_whitelist<- read_tsv("rna.737K-arc-v1.txt", col_names = FALSE)


#Match scatac barcodes whitelists with scrna barcodes whitelist
barcode_map <- set_names(atac_whitelist$X1, nm= rna_whitelist$X1)
atac_barcode <- as.character(barcode_map[Cells(rna_clusters_res6)])
# Attach atac cell barcodes to rna
renamed_rna.obj <- RenameCells(object = rna_clusters_res6, new.names = atac_barcode)

#Find Overlapped cells between rna and atac
```

Please match the scRNA-seq and scATAC-seq barcodes to find the list of common cells (2 pts;)

```
in_common <- base::intersect(Cells(renamed_rna.obj), Cells(atac_scaled))
length(in_common)
```

```
## [1] 2534
```

How many cells are in common (1 pts;)

`2534 cells are in common between the scRNA-seq and the scATAC-seq.`

- **1.2 Please use the list of common cells to subset scRNA and scATAC Seurat object (2 pts;). We will use these sub-sampled Seurat objects for the rest of the homework. hint: use `subset()`**

```
#Taking common subset of scRNA-seq and scATAC-seq
renamed_rna.obj@meta.data$cells <- rownames(renamed_rna.obj@meta.data)
rna_subset <- subset(renamed_rna.obj, subset = cells%in%in_common)

atac_scaled@meta.data$cells <- rownames(atac_scaled@meta.data)
atac_subset <- subset(atac_scaled, subset = cells%in%in_common)

# check that this is 2534
# length(base::intersect(Cells(rna_subset), Cells(atac_subset)))
```

2. Label Transferring: In Part III Question 3.3, we displayed a UMAP with cell type annotated by MAESTRO. You probably have noticed that the annotation result is not promising. In this section, let's use the label transferring methods from Seurat to redo the cell annotation.

You can find the detailed tutorial from Seurat.

- **2.1 First, let's identify anchors between scATAC-seq data set and scRNA-seq data set (1 pts;). Set `DefaultAssay()` as 'ACTIVITY' for scATAC object. In `FindTransferAnchors()`, set 'RNA' as reference assay and 'ACTIVITY' as query assay.**

```
transfer.anchors <- FindTransferAnchors(reference=rna_subset, query=atac_subset, reduction = "cca")
transfer.anchors
```

```
## An AnchorSet object containing 2499 anchors between the reference and query Seurat objects.
##  This can be used as input to TransferData.
```

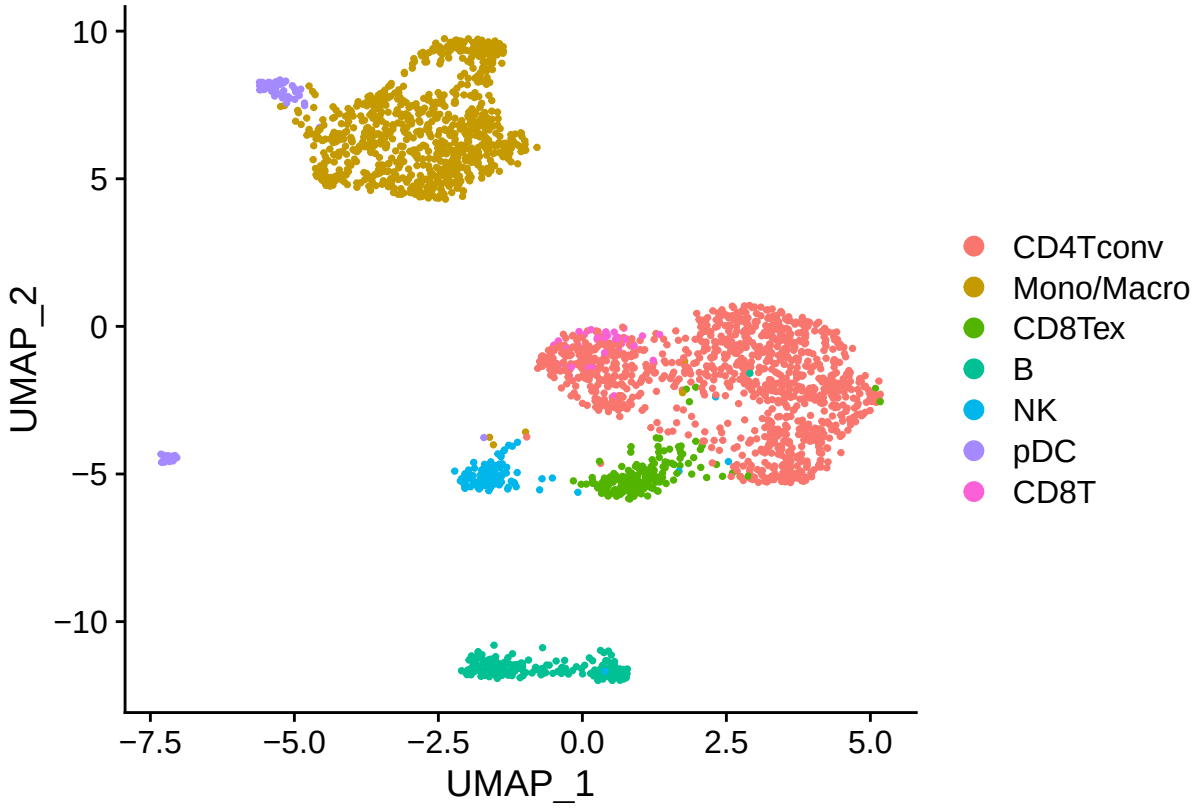
- **2.2 Based on anchors between two modalities, please use 2:30 LSI as `weight.reduction` to transfer scRNA-seq cell types stored in `assign.ident` to scATAC-seq data (1 pts;). Attach predicted cell types to scATAC-seq metadata and Visualize the predicted cell types on UMAP (1 pts;).**

use 2:30 LSI as `weight.reduction` to transfer scRNA-seq cell types stored in `assign.ident` to scATAC-seq data

```
predictions <- TransferData(anchorset = transfer.anchors,
                            refdata = rna_subset$assign.ident,
                            weight.reduction = atac_subset[['lsi']],
                            dims=2:30)
```

Attach predicted cell types to scATAC-seq metadata and Visualize the predicted cell types on UMAP

```
atac_subset <- AddMetaData(atac_subset, metadata = predictions)
atac_subset_umap <- RunUMAP(atac_subset, reduction = 'lsi', dims=2:30)
Idents(atac_subset_umap) <- 'predicted.id'
DimPlot(atac_subset_umap, reduction = 'umap')
```

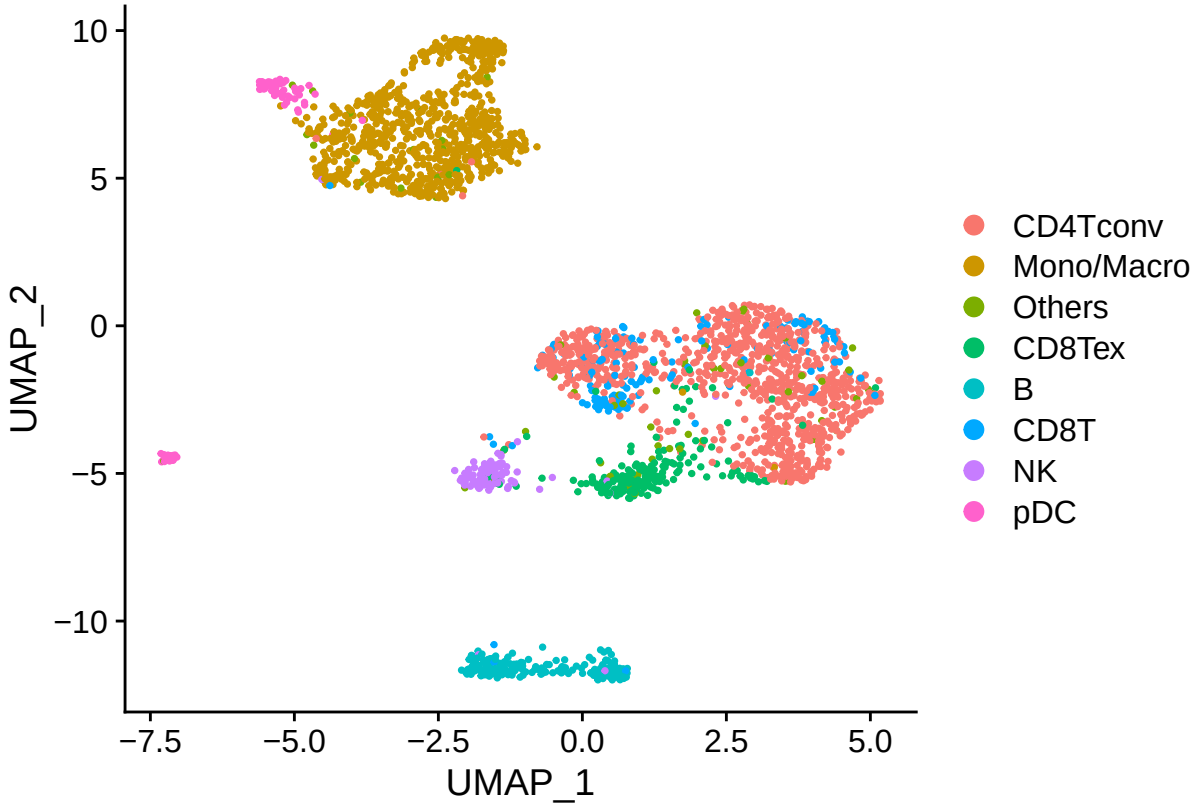


3. Though we can do label transferring to annotate scATAC-seq cells, back to the time without multiome data, it is hard to say if this label transferring method is accurate. The good thing in multiome data is that, by matching cell barcodes between scRNA-seq and scATAC-seq data, we actually know the "true label" for cells in scATAC-seq. We have a ground-truth annotation that can be used for evaluating the accuracy of label transferring.

- **3.1 Since we subset the scRNA-seq and scATAC-seq objects with the list of common cells, the `assign.ident` column stored in scRNA-seq metadata is exactly the true cell type label for cells in scATAC-seq. So we can easily add `assign.ident` in scATAC-seq metadata and plot a UMAP. Please visualize the ground-truth cell type labels of scATAC-seq data on UMAP (1 pts;). Compare the results with predicted cell type UMAP from the last question. Virtually, does label transferring seem like an accurate method (1 pts;)?**

Visualize the ground-truth cell type labels of scATAC-seq data on UMAP (1 pts;)

```
ground_truth <- rna_subset@meta.data %>%
  select(assign.ident) %>%
  rename('ground_truth' = 'assign.ident')

atac_subset_umap <- AddMetaData(atac_subset_umap, metadata = ground_truth)

Idents(atac_subset_umap) <- 'ground_truth'
DimPlot(atac_subset_umap, reduction = 'umap')
```



Does label transferring seem like an accurate method (1 pts;)

Answer:

`The transferred clusters reflect ground truth very well. This is epsecially clear in the distinct Mono/`

- **3.2 What is the accuracy of the label transferring method based on the RP-enhanced**

model we used to calculate gene activity score? In Part IV Question 2.1, We used the 'ACTIVITY' slot to find anchors. This gene activity matrix is derived from the peak count matrix based on the RP-enhanced model we set in MAESTRO. Please show a table of what percent of the cells are correctly labeled (predicted.id == True_label) (1 pts;) ?

```
sum(atac_subset_umap$predicted.id == atac_subset_umap$ground_truth) / nrow(atac_subset_umap@meta.data)
```

```
## [1] 0.8772691
```

```
atac_subset_umap@meta.data %>%
  mutate(same = predicted.id == ground_truth) %>%
  group_by(ground_truth) %>%
  summarize(mean(same))
```

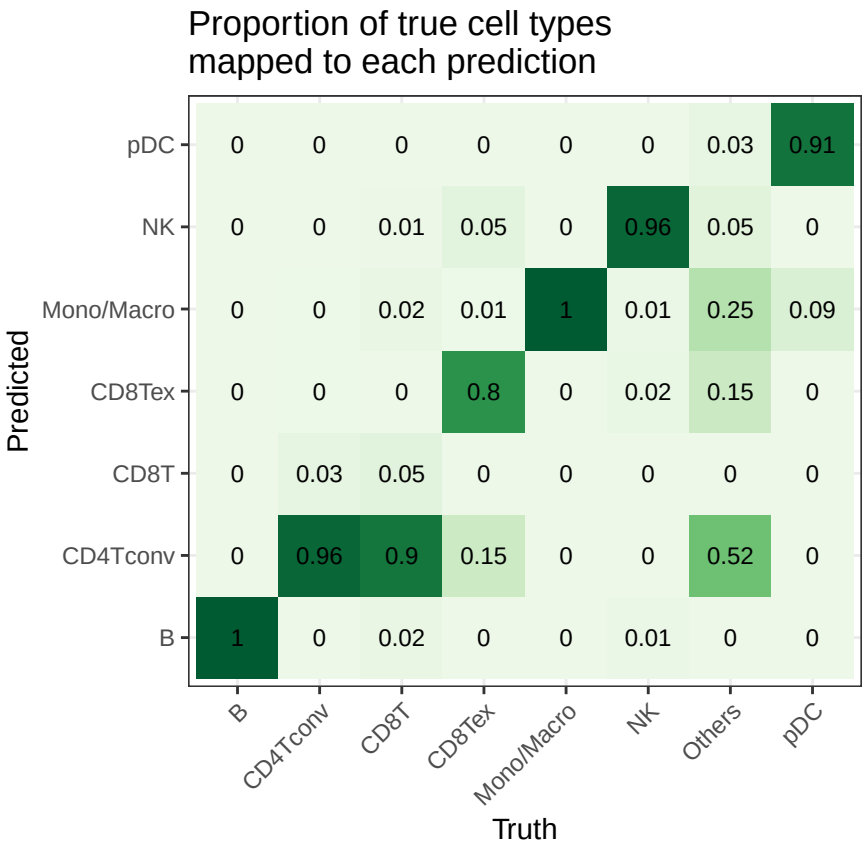
```
## # A tibble: 8 x 2
##   ground_truth `mean(same)`
##   <chr>               <dbl>
## 1 B                   1
## 2 CD4Tconv            0.959
## 3 CD8T                0.0510
## 4 CD8Tex              0.799
## 5 Mono/Macro          0.998
## 6 NK                  0.960
## 7 Others              0
## 8 pDC                 0.915
```

Answer:

```
The overall accuracy of the label transferring was 87.83%.

Separated by true label, cell types B, CD4Tconv, Mono/Macro, NK, and pDC were identified very well, wit
```

- **3.3 For scATAC-seq data, create a heatmap with true cell type labels on x-axis and predicted cell type labels on y-axis. Fill with gradient colors to indicate the fraction of cells matched between corresponding cell types (Graduate level 2 pts;). Describe what clusters appear to map 1 to 1 between the two modalities and which clusters appear to split (Graduate level 2 pts;)?**

```
match_counts <- atac_subset_umap@meta.data %>%
  select(ground_truth, predicted.id) %>%
  table()

match_props <- match_counts/rowSums(match_counts)

t(match_props) %>% data.frame() %>%
  ggplot(aes(x = ground_truth, y = predicted.id, fill = Freq)) +
  geom_tile()  + theme_bw() + coord_equal() +
  scale_fill_distiller(palette="Greens", direction=1) +
  geom_text(aes(label=round(Freq, 2)), color="black", size = 3) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = 'none') +
  labs(
    x = "Truth",
    y = "Predicted",
    title = "Proportion of true cell types\nmapped to each prediction"
  )
```



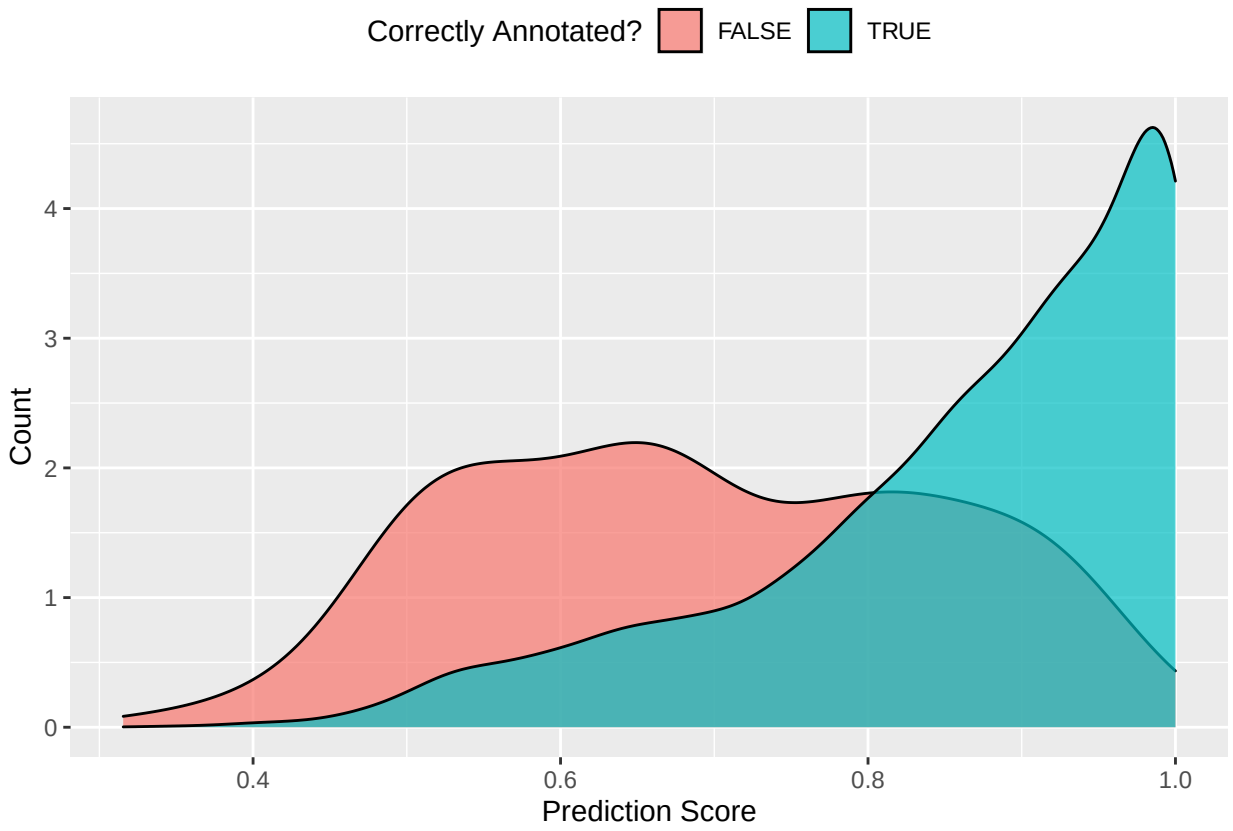
Describe what clusters appear to map 1 to 1 between the two modalities and which clusters appear to split (Graduate level 2 pts;)

Answer:

```
This plot shows for each true label (x axis) the proportion of cells mapped to each predicted label (y a

Cell types B, CD4Tconv, Mono/Macro, NK, and pDC seem to map 1 to 1 between the two modalities. That is,

CD8T is almost always mapped to CD4Tconv (90% of the time).

"Other" cells are most often mapped to CD4Tconv, but frequently mapped to Mono/Macro and CD8Tex as well
```

- **3.4 Generate a density plot for prediction score generated in label transferring steps. Use two different colors for correctly annotated cells and incorrectly annotated cells (Graduate level 2 pts;).**

```
data.frame(
  pred_score = atac_subset_umap@meta.data$prediction.score.max,
  correct = atac_subset_umap@meta.data$ground_truth == atac_subset_umap@meta.data$predicted.id
) %>%
  ggplot(aes(x = pred_score, fill = correct)) +
  geom_density(bins = 30, position = 'identity', alpha = 0.7) +
  labs(
    fill = "Correctly Annotated?",
    x = "Prediction Score",
    y = "Count"
  ) +
  theme(legend.position = 'top')
```



It makes sense that correctly annotated cells tend to have higher predcition scores--the cells that the