Master's thesis in Bioinformatics and Computational Biology

Alba Méndez Alejandre

22/05/2025

Table of contents

1 Mouse and Human SComatic analysis

Preface

This document presents a comprehensive analysis of single-cell transcriptomics somatic mutations in mouse and human datasets, covering:

- Preprocessing of single-cell RNA-seq data
- Clustering and annotation of cell types
- RNA velocity inference to understand cellular dynamics
- Somatic variant calling using SComatic
- Functional annotation and interpretation of variants

2 Introduction

This repository aims to be a log of the overall work i did for my master's thesis.

2.1 Computers

I used mainly two computers for all the calculations, though the HCA dataset was in a third one, so i had to use it sporadically.

- **matterhorn:** main computer. Mainly used for storage and explorative analysis.
- **nuptse:** used for storage and explorative analysis of HCA dataset.
- **folia:** small computing server. Used for clusterization, alignment, etc.

All code was executed in computers running Ubuntu 22.04.4 LTS.

2.2 Conda environments

Most of the software was installed using mamba/conda environments when possible.

```
mamba config --add channels bioconda
mamba config --add channels conda-forge
```

2.2.1 d_rstudio

State: active **Computer:** nuptse, folia, matterhorn **Purpose:** to have a functional RStudio/VSCode installation along with the packages for the kallisto-bustools velocity workflow. **Creation:** run the following commands to install RStudio along with the packages needed for data analysis.

```
mamba create -n d_rstudio -c conda-forge rstudio-desktop jupyter r-seurat

conda activate d_rstudio

mamba install r-devtools r-tidyverse r-zeallot r-ggally bioconductor-bsgenome.mmusculus.ucsc

# Install packages from source
R

# Hard-code the commit for reproducibility
devtools::install_github("satijalab/seurat-wrappers@73466e361ee759c6b1add58faa3bc4e7a2ee5753

q()

# Posterior installations
mamba install r-velocyto.r
mamba install -c bioconda bioconductor-slingshot
mamba install leidenalg # for clustering
mamba install numpy pandas
mamba install -c conda-forge r-clustree
mamba install -c conda-forge r-svglite

# Installing packages to convert to H5AD data
R

# Hard-code commit for future reproducibility. Skip updates when asked
devtools::install_github("mojaveazure/seurat-disk@877d4e18ab38c686f5db54f8cd290274ccdbe295")


mamba install -c conda-forge plotly python-kaleido
mamba install -c plotly plotly-orca
mamba install -c conda-forge r-processx
mamba install -c conda-forge r-pals
mamba install -c conda-forge r-ggvenn r-ggvenndiagram r-venn r-venndiagram
```

2.2.2 SComatic

State: active **Computer:** nuptse, folia, matterhorn **Purpose:** to have an isolated environment with SComatic for scRNA-seq mutation calling

```
mamba create -n d_scomatic -c bioconda python=3.7 r-base=3.6.1 samtools datamash bedtools

# You can download a zip file with the repository "Code" button in the web
# Or you can do the same thing in the linux terminal
# For future reproducibility
wget -P /home/dario/bin/ https://github.com/cortes-ciriano-lab/SComatic/archive/f515f4ee3e7c
# To grab the latest branch
wget -P /home/dario/bin/ https://github.com/cortes-ciriano-lab/SComatic/archive/main.zip

unzip *zip
mv SComatic-main SComatic

# You could also clone the repository to keep the files up to date if needed
git clone --single-branch https://github.com/cortes-ciriano-lab/SComatic.git /path/to/dir/

# I install the remaining dependencies as instructed, using the "requirements.txt" file in t
mamba activate d_scomatic

pip install -r requirements.txt
```

Part I

Mouse analysis

3 1. Data Pre-processing

This notebook contains the bash scripts used to download the fastq, quality control with fastQC and multiQC and alignment of the sequences.

3.1 1.1. Download fastQ files from ENA

https://www.ebi.ac.uk/biostudies/arrayexpress/studies/E-MTAB-8662

```
#!/bin/bash
# [matterhorn]
# AUTHOR: Alba Mendez Alejandre
# DESCRIPTION: Download the raw fastQ via ftp from ENA. The metadata is stored in the downloa
# DATE: 10/06/2024

seq_ids=("run1" "run2")

for seq_number in "${seq_ids[@]}"; do

    # Path to our CSV file
    CSV_FILE="/mnt/D/mcGinn_2021/data_info/${seq_number}_download.csv"

    # Directory where we are going to download the data (bam and fastq)
    DOWNLOAD_DIR="./${seq_number}_run1/${seq_number}_adult_P70"

    # Create the directory if it doesn't exist
    mkdir -p "$DOWNLOAD_DIR"

    # Iterate through each line in the CSV file
    {
        read # skip header
        while IFS=, read -r name source_name bam_uri R1_fastq_uri R2_fastq_uri I1_fastq_uri #
        do
            # Skip header
            if [[ $name == "name" ]]; then
```

```
            continue
            fi

            # Ensure the URI is not empty
            if [[ -z $R1_fastq_uri ]]; then
            echo "Empty URI for name: $name"
            continue
            fi

            # Download R1_fastq_uri
            curl -L -o "${DOWNLOAD_DIR}/${name}_$(basename "$R1_fastq_uri")" "$R1_fastq_uri"

            if [[ -z $R2_fastq_uri ]]; then
            echo "Empty URI for name: $name"
            continue
            fi

            # Download R2_fastq_uri
            curl -L -o "${DOWNLOAD_DIR}/${name}_$(basename "$R2_fastq_uri")" "$R2_fastq_uri"

            if [[ -z $I1_fastq_uri ]]; then
            echo "Empty URI for name: $name"
            continue
            fi

            # Download I1_fastq_uri
            curl -L -o "${DOWNLOAD_DIR}/${name}_$(basename "$I1_fastq_uri")" "$I1_fastq_uri"

            if [[ -z $bam_uri ]]; then
            echo "Empty URI for name: $name"
            continue
            fi

        done < "$CSV_FILE"
    }
done
```

```
./
```

After the download of fastQ files, they were concatenated in order to obtain

3.2 1.2 Quality control with MultiQC

View MultiQC report for sequencing run 2

View MultiQC report for sequencing run 1

3.3 1.3. STAR aligment

Separate scripts were used for each sequencing session (names as run1 and run2 for simplification).

```
#!/bin/bash
# [folia]
# This script must be executed in the server where we want to run STAR

# activate env where we have installed STAR
source /home/albax/miniforge3/bin/activate STAR

FOLIA_BASE="/home/albax/mcGinn_2021"
MATTERHORN_BASE="/media/storage/mcGinn_2021"
CONCATENATED_RUNS_DIR="${MATTERHORN_BASE}/concatenated_runs/sample_concatenated"


SAMPLE_NAMES=("lib1" "lib2" "lib3" "lib4" "lib5" "lib6" "lib7" "lib8") # list of samples to p

i=0
for SAMPLE in "${SAMPLE_NAMES[@]}";
do
    i=$((i + 1))

    SAMPLE_DIR="${CONCATENATED_RUNS_DIR}/${SAMPLE}"
    LOCAL_SAMPLE_DIR="${FOLIA_BASE}${SAMPLE_DIR}"
    echo "Processing sample: ${SAMPLE}"

    R2_FILE="${SAMPLE_DIR}/${SAMPLE}*R2*run1.fastq.gz"
    R1_FILE="${SAMPLE_DIR}/${SAMPLE}*R1*run1.fastq.gz"

    # Files' pull from matterhorn to folia
    rsync -avR --progress -hh "alba@matterhorn:${R2_FILE}" "${FOLIA_BASE}"
    rsync -avR --progress -hh "alba@matterhorn:${R1_FILE}" "${FOLIA_BASE}"
```

```
    LOCAL_R2_FILE="${FOLIA_BASE}${R2_FILE}"
    LOCAL_R1_FILE="${FOLIA_BASE}${R1_FILE}"

    echo "R2 file (folia): ${LOCAL_R2_FILE}"
    echo "R1 file (folia): ${LOCAL_R1_FILE}"

    cd ${FOLIA_BASE}
    mkdir -p "${FOLIA_BASE}/STAR_out/${SAMPLE}"

  # Process files
#   ~/miniforge3/envs/STAR/bin/STAR \
    STAR --runMode alignReads --genomeDir /home/albax/reference_genomes/STAR_indexes/Mus_musc

    if [[ $? -ne 0 ]]; then
        echo "STAR alignment failed for sample ${SAMPLE}."
        continue
    fi

        # Transfer back to matterhorn
    ssh alba@matterhorn "mkdir -p /media/storage/mcGinn_2021/STARalignment/${SAMPLE}/"

    rsync -av --progress -hh ${FOLIA_BASE}/STAR_out/*${SAMPLE}* alba@matterhorn:${MATTERHORN_

    # Remove to save space
    echo -e "\n[ $(date +'%Y/%m/%d %T.%3N') ] Finished processing ${SAMPLE}"
    rm ${LOCAL_R1_FILE} ${LOCAL_R2_FILE}
    rm -r ${FOLIA_BASE}/STAR_out/*${SAMPLE}*/

done

# clean up base directory
rm -r ${FOLIA_BASE}/STAR_out

conda deactivate
```

Generate the reference genome indexed file

```
#!/bin/bash

cd ~/reference_genomes/
mkdir STAR_indexes
mkdir logs
```

```
# path to star
~/miniforge3/envs/STAR/bin/STAR \
            --runThreadN 4 \
        --runMode genomeGenerate \
        --genomeDir ./STAR_indexes/Mus_musculus/mm10 \
        --genomeFastaFiles ./Mus_musculus_C57BL-6J/Mus_musculus.GRCm38.dna.primary_assembly.
        --sjdbGTFfile ./Mus_musculus_C57BL-6J/Mus_musculus.GRCm38.97.gtf \
                                |& tee logs/star_index.log
```

3.4 1.4. Create Seurat object

The next step is to create the seurat object with the count matrices obtained from STARsolo, with spliced, unspliced and gene matrices.

> [The file is named "1-4_merge_seurat_fixedrank.R"]

4 2. Clustering and Cell annotation

We are going to perform the downstream analysis of single cell transcriptomics mainly with R package Seurat (https://satijalab.org/seurat/).

The version 4 was used instead of the 5 due to version compatibility problems with other methods needed downstream.

4.1 Import libraries

```
.libPaths("/home/albax/miniforge3/envs/seurat_v4/lib/R/library")

if(.Platform$OS.type == "linux") Sys.setenv(PATH= paste("/home/albax/miniforge3/envs/seurat_v

library(reticulate)

use_condaenv("/home/albax/miniforge3/envs/seurat_v4", required = TRUE)
py_config()

import("numpy")
import("leidenalg")
import("pandas")

suppressMessages(library(Seurat))
suppressMessages(library(dplyr))
suppressMessages(library(DropletUtils)) #QC filtering
suppressMessages(library(ggplot2))
suppressMessages(library(plotly))
suppressMessages(library(SingleCellExperiment))
suppressMessages(library(clustree))
suppressMessages(library(httpgd))
suppressMessages(library(patchwork))
# suppressMessages(library(BPCells)) # for on-disk memory
```

```r
suppressMessages(library(future))
suppressMessages(library(future.apply))
suppressMessages(library(BiocParallel))


# For data management
suppressMessages(library(tidyverse))
suppressMessages(library(Matrix))
suppressMessages(library(gtools))
suppressMessages(library(R.utils))

# For plotting
suppressMessages(library(RColorBrewer))
suppressMessages(library(viridis))
suppressMessages(library(gplots))
suppressMessages(library(gridExtra))
suppressMessages(library(ggrepel))
suppressMessages(library(ggridges))

# for matrix
library(Matrix)
library(Matrix.utils)

color.list <- RColorBrewer::brewer.pal(12,"Paired")
color.list <- c(color.list,RColorBrewer::brewer.pal(12,"Set3"))

# Palette from orange to violet
palette <- scale_color_viridis_c(option = "plasma", direction = -1) # continue colors palette
palette_d <- scale_color_viridis_d(option = "turbo", direction = 1) # discrete colors palette

name_order <- c("lib1", "lib2", "lib3", "lib4", "lib5", "lib6", "lib6", "lib7") # fixed libra

setwd("/home/albax/mcGinn_2021")
```

4.2 Load data

First, we load our rds object, and we get only the data from the assay "gene". Note that we have different rds objects, each of them contains different step versions (non-filtered, filtered, annotated), in order to maintain consistency and avoid errors and accidental deletes.

```
# Esoph <- readRDS(file = "./results/esoph_star_filtfixed_mm10.rds") # this is with mm10 alig
Esoph # 57186 genes genes accross 46321 cells
# An object of class Seurat
# 171558 features across 46321 samples within 3 assays
# Active assay: gene (57186 features, 0 variable features)
#  2 layers present: counts, data
#  2 other assays present: spliced, unspliced

# Esoph_clusts <- readRDS(file = "./output/esoph_star_clusts_mm10.rds") # after clustering, u

# Esoph_filt <- readRDS(file = "./output/esoph_star_filtered_mm10_vep_annot.rds") # also load
Esoph_filt # 57186 genes genes accross 39763 cells
# An object of class Seurat
# 250024 features across 39763 samples within 5 assays
# Active assay: RNA (57186 features, 0 variable features)
#  2 layers present: counts, data
#  4 other assays present: gene, spliced, unspliced, SCT
#  3 dimensional reductions calculated: pca, tsne, umap

Esoph_filt <- readRDS(file = "./output/esoph_star_filtered_mm10_vep_annot_rep.rds") # seurat
Esoph_filt
# An object of class Seurat
# 250024 features across 39763 samples within 5 assays
# Active assay: RNA (57186 features, 0 variable features)
#  2 layers present: counts, data
#  4 other assays present: gene, spliced, unspliced, SCT
#  3 dimensional reductions calculated: pca, tsne, umap
```

The features(genes) are automatically collapsed -> there are no duplicated rows in any of the assays.

Look for duplicated genes:

```
gene_features <- rownames(Esoph[["unspliced"]])
duplicated_genes <- gene_features[duplicated(gene_features)]
print("Duplicated genes in the 'gene' assay:")
print(duplicated_genes)
```

4.3 QC stats

We are going to perform everything only in the “spliced” assay.

```
DefaultAssay(Esoph) <- "spliced" # change active assay to spliced
head(Esoph@meta.data, 5) #it contains nCount and nFeature for each assay
```

Typically, we want to filter: - **Blood cells (erythrocites)** - The percentage of reads that map to the mitochondrial genome - Low-quality / dying cells often exhibit extensive mitochondrial contamination - We calculate mitochondrial QC metrics with the PercentageFeatureSet() function which calculates the percentage of counts originating from a set of features - We use the set of all genes starting with **MT-(^MT)** as a set of mitochondrial genes low quality barcodes (a treshold)

- Empty barcodes (using a treshold)
- The number of unique genes detected in each cell.
  - **Low-quality cells or empty droplets** will often have very few genes
  - Cell **doublets** or multiplets may exhibit an aberrantly high gene count
- Similarly, the total number of molecules detected within a cell (correlates strongly with unique genes)

We have already filtered our data with barcodes ranks, and now we are going to inspect the seurat object in order to decide how we filter it spoiler: we end up filtering using the clusterization (7 clusters) shown in umap with resolution 0.3 (which we decide viewing clustree), where we are going to delete clusters 4 and 7, as they belong to fibroblasts (Vim marker) or have really high mt percent.

4.3.1 Detect MT genes

```
head(row.names(Esoph)) # first genes
Esoph[["percent.mt"]] <- PercentageFeatureSet(Esoph, assay = "spliced", pattern = "^mt-") # c
head(Esoph@meta.data, 5)

# we have NaN values in percent.mt because percentage is calculated by percentage is (x$nCoun

# Delete barcodes
# Esoph <- subset(Esoph, subset = !is.na("percent.mt")) # exclude NaN values
```

Visualize QC metrics

```r
VlnPlot(Esoph, features = c("nFeature_spliced", "nCount_spliced", "nFeature_unspliced", "nCou
# VlnPlot(Esoph, features =  "percent.mt", ncol = 1) +
       scale_y_continuous(breaks = seq(0, 100, by = 5))
# VlnPlot(Esoph, features =  "nFeature_spliced", ncol = 1) +
       scale_y_continuous(breaks = seq(0, 10000, by = 500))
# VlnPlot(Esoph, features =  "nCount_spliced", ncol = 1) +
       scale_y_continuous(breaks = seq(0, 1000000, by = 10000))
VlnPlot(Esoph, features =  "nFeature_spliced", split.by = "Sequencing_ID", ncol = 1) +
       scale_y_continuous(breaks = seq(0, 10000, by = 500)) # frecuencia genes en cada sequer

# plot(density(Esoph$nFeature_spliced)) # kernel density plot (alternative to hist())
# plot(density(Esoph$percent.mt)) # kernel density plot (alternative to hist())

VlnPlot(Esoph_filt, features="nFeature_spliced", split.by="Sequencing_ID")
```

```r
plot1 <- FeatureScatter(Esoph, feature1 = "nCount_spliced", feature2 = "percent.mt")
plot2 <- FeatureScatter(Esoph, feature1 = "nCount_spliced", feature2 = "nFeature_spliced")
plot3 <- FeatureScatter(Esoph, feature1 = "nCount_spliced", feature2 = "nCount_unspliced")
plot1 + plot2 + plot3
```

According to the plots we have seen, we are going to filter the Esoph object (by eye).

The original paper filtered the cells according to: - Cells that have >15% mitochondrial counts - Cells that have unique feature counts less than 1200 - Genes expressed in fewer than 3 cells But we are not going to do this.

```r
## General statistics post-second filtering:
# Proportion of UMIs that are from unspliced transcripts:
# (kallisto | bus counts reads that are partially intronic and partially exonic as unspliced
sum(Esoph$nCount_unspliced) / (sum(Esoph$nCount_spliced) + sum(Esoph$nCount_unspliced)) # 0.1
# [1] 0.1281474 mm10

# Most barcodes now have >> 0 or 1 UMIs detected:
filt2_count <- Matrix::colSums(Esoph)
summary(filt2_count) # median genes = 13632 mm39
# median genes = 11900 mm10

# Make a copy of spliced assay (for purpose of unambiguous exportation and interpretation in
Esoph[["RNA"]] <- Esoph[["spliced"]]
Esoph # spliced assay keeps as default assay used as input for downstream normalization etc.
```

4.4 Normalization, feature selection, scaling

```
Esoph <- SCTransform(Esoph, assay = "spliced", new.assay.name = "SCT")
# creates SCT assay (that becomes default)

# Highly-variable features (between cells):  # 3000 by default
# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(Esoph), 10)
# Linear scaling is restricted to highly-variable features by default: Esoph[["SCT"]]@scale.
```

```
# plot variable features with labels
LabelPoints(plot = VariableFeaturePlot(object = Esoph[["SCT"]], selection.method = "sct"), po
```

4.4.1 Dimensional reduction (PCA)

Observation: Makes most sense to plot RNA velocity over cell embeddings from the SCT matrix (built from spliced, not unspliced counts; i.e. we seek arrows as predictions from CURRENT state)

```
DefaultAssay(Esoph) <- "SCT" # already default, but just in case
Esoph <- RunPCA(Esoph, verbose = FALSE) # by default, based on variable features
```

```
# Summary of genes defining most variability: (higher PCA score)
print(Esoph[["pca"]], dims = 1:5, nfeatures = 5)
VizDimLoadings(Esoph, dims = 1:2, reduction = "pca")

# PCA plot:
DimPlot(Esoph, dims = c(1,2), reduction = "pca", pt.size = 0.5) # change the dimensions as yo
```

Heatmap of genes and cells with highest PCA score (a quick supervised analysis of sources of variation)

```
DimHeatmap(Esoph, dims = 1, cells = 500, balanced = TRUE) # picks 500 most extreme cells on e
DimHeatmap(Esoph, dims = 1:15, cells = 500, balanced = TRUE)
```

```
# Set the threshold of significant dimensions based on conjunction of JackStraw, ElbowPlot an

## Significant dimensions: determine most relevant sources of variability
# Extensive technical noise is reduced when eliminating minor components, so top principal co
```

```
# ElbowPlot (heuristic; based on % variance explained by each PC component)
ElbowPlot(Esoph) # use 17
```

4.4.2 Non-linear dimensional representation

Simplifies the complex manifold of the data in a super-reduced dimensional space for visualization. We should run this after PCA, in order to further reduce the dimensionality or our dataset. Later, we can run FindClusters() with whatever dimension we prefer.

```
## tSNE
Esoph <- RunTSNE(Esoph, dims = 1:17, verbose = FALSE)

# saveRDS(Esoph, file = "./output/esoph_star_tsne_mm10.rds")

tsne_plot_15 <- DimPlot(Esoph, reduction = "tsne", pt.size = 0.5) # _ clusters
tsne_plot_15
```

Be careful when running this chunk, as umap can consume a lot of cpu:

```
## UMAP
Esoph <- RunUMAP(Esoph, reduction = "pca", dims = 1:17)
# saveRDS(Esoph, file = "./output/esoph_star_clusts_mm10.rds") # contains also FindNeighbours
```

```
umap_plot_15 <- DimPlot(Esoph, reduction = "umap", pt.size = 0.5, label=TRUE)
umap_plot_15
```

```
unique_names <- sapply(strsplit(colnames(Esoph), "-"), function(x) paste(x[2], x[3], sep = "-
# Add this information as metadata
Esoph$unique_name <- unique_names

# Create a DimPlot colored by the unique names
DimPlot(Esoph, reduction = "umap", pt.size = 0.5, label=FALSE, group.by = 'unique_name') # gr
```

4.4.3 Clustering (graph-based)

1. Cellular distance metric: euclidean distance (on PCAs)
2. Embedding cells in a graph structure - for example a K-nearest neighbor (KNN) graph, with edges drawn between cells with similar feature expression patterns

4.4.3.1 Find Neighbours (KNN)

```
Esoph <- FindNeighbors(Esoph, reduction = "pca", dims = 1:17) # dims based on first N compone
```

4.5 Find Clusters

Algorithm for modularity optimization (1 = original Louvain algorithm; 2 = Louvain algorithm with multilevel refinement; 3 = SLM algorithm; 4 = Leiden algorithm). Leiden requires the leidenalg python. - Louvain algorithm: Partitioning into highly interconnected 'quasi-cliques' or 'communities', optimizing a modularity target function. It may yield arbitrarily badly connected communities. In the worst case, communities may even be disconnected, especially when running the algorithm iteratively. - Leiden algorithm: when applied iteratively, it converges to a partition in which all subsets of all communities are locally optimally assigned. Furthermore, by relying on a fast local move approach, the Leiden algorithm runs faster than the Louvain algorithm.

We will use Leiden algorith (algorithm = 4, method = "igraph") https://www.nature.com/articles/s41598-019-41695-z

```
# Esoph <- FindClusters(Esoph, resolution = seq(0.2, 1.2, by = 0.2), algorithm = 4) # this co

Esoph <- FindClusters(Esoph, resolution = 0.1, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 0.2, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 0.3, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 0.4, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 0.6, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 0.8, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 1.0, algorithm = 4, method = "igraph")
Esoph <- FindClusters(Esoph, resolution = 1.2, algorithm = 4, method = "igraph")
```

```
# To decide how many clusters we should annotate, with the signatures we decide
clustree(Esoph, prefix = "SCT_snn_res.") # we are going to use resolution 0.2 for Esoph in mm
```

```
# Idents() contains cluster info
head(Idents(Esoph), 10) # looks at cluster IDs of the first 10 cells
```

```
DimPlot(Esoph, group.by = "SCT_snn_res.0.2") & palette_d
DimPlot(Esoph, group.by = "SCT_snn_res.0.3") & palette_d
DimPlot(Esoph, group.by = "SCT_snn_res.0.4") & palette_d
```

```
DimPlot(Esoph, group.by = "SCT_snn_res.0.2", split.by = "condition") & palette_d
```

Finding different cell types that may not be epithelial, due to contamination, experiment design, whatever…

Cell markers (they may vary between mouse/human): - Hematopoetic cells -> Cd34 - Fibroblasts -> Vim

```
FeaturePlot(Esoph, features = "Cd34") & palette # marker for hematopoietic stem cells (immune
FeaturePlot(Esoph, features = "Vim") & palette # marker for fibroblasts
```

Thanks to the plots above, we can clearly see that we have fibroblast cells that belong to cluster number 7 (when using resolution 0.2).

Quality control plots with umap:

```
# number of genes per cell
FeaturePlot(Esoph, features = "nFeature_spliced") & palette # no huge heterogeneities
FeaturePlot(Esoph, features = "percent.mt") & palette
```

We an see that the top "leaf" of our umap has low-quality cells, with few genes, and very high MT%. These cells belong to cluster 4 (when using resolution 0.2).

```
# Select resolution for Seurat Clusters:
Idents(Esoph) = Esoph$SCT_snn_res.0.2
Esoph$seurat_clusters = Esoph$SCT_snn_res.0.2
```

```
## Remaining annotation using top markers for every cluster compared to all remaining ones,
# Esoph.markers <- FindAllMarkers(Esoph, only.pos = TRUE, min.pct = 0.25, logfc.threshold =

# write.table(Esoph.markers, file = './output/Esoph_markers.txt', col.names = TRUE, row.names
```

RESULTS:

When we plot the UMAP, we can see that there are: - Region with high MT proportion (belongs to cluster 4) - Two regions that seem to be different type of cells. - The region from the upper-left seems to be fibroblasts, bc if we run DimPlot(Esoph, features = features = c("Vim")), it colours. Vim(vimentin) is a typical fibroblast marker. - The other region must be further investigated to detect what is it. - Also, the cluster 4 looks like a "batch" that represents the whole data - Cluster 6 is specific of sample Old_DEN, which could be biological evidences!!

After annotating and saving the preliminar results, we proceed by deleting the cluster number 4, as it is very heterogenous and somehow represents all the clusters in the data.

We also delete cluster number 7, as they are fibroblasts.

Delete cluster 4 and 7:

```
Esoph_filt <- subset(x = Esoph, idents = c("4", "7"), invert = TRUE)
# mm10
# An object of class Seurat
# 250504 features across 39763 samples within 5 assays
# Active assay: SCT (21760 features, 3000 variable features)
#  3 layers present: counts, data, scale.data
#  4 other assays present: gene, spliced, unspliced, RNA
#  3 dimensional reductions calculated: pca, tsne, umap
```

4.6 Cell annotation

```
Esoph.markers <- FindAllMarkers(Esoph_filt, only.pos = TRUE, min.pct = 0.25, logfc.threshold

write.table(Esoph.markers, file = './output/Esoph_markers_clusts.txt', col.names = TRUE, row

TopMarkers <- Esoph.markers %>%
  group_by(cluster) %>% top_n(n = 5, wt = avg_log2FC) # show just top 5 per cluster


TopMarkers %>% write.csv("./output/Esoph_TopMarkers.csv")
cluster3_markers <- TopMarkers[TopMarkers$cluster == 3, ]  # Suprabasal
cluster4_markers <- TopMarkers[TopMarkers$cluster == 4, ]  # Mito-rich
cluster6_markers <- TopMarkers[TopMarkers$cluster == 6, ]  # Epi_DEN
cluster7_markers <- TopMarkers[TopMarkers$cluster == 7, ]  # Fibroblasts
```

4.7 Re-normalization and clustering of filtered object

Repeat normalization, PCA, tSNE and umap:

```
Esoph_filt <- SCTransform(Esoph_filt, assay = "spliced", new.assay.name = "SCT")
DefaultAssay(Esoph_filt) <- "SCT" # already default, but just in case
```

```
Esoph_filt <- RunPCA(Esoph_filt, verbose = FALSE)

ElbowPlot(Esoph_filt) # 17 dimensions

Esoph_filt <- RunTSNE(Esoph_filt, dims = 1:17, verbose = FALSE, reduction = "pca", reduction

Esoph_filt <- RunUMAP(Esoph_filt, reduction = "pca", dims = 1:17, reduction.name = "umap")
```

```
Esoph_filt <- FindNeighbors(Esoph_filt, reduction = "pca", dims = 1:17)

Esoph_filt <- FindClusters(Esoph_filt, resolution = 0.1, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 0.2, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 0.3, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 0.4, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 0.6, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 0.8, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 1.0, algorithm = 4, method = "igraph")
Esoph_filt <- FindClusters(Esoph_filt, resolution = 1.2, algorithm = 4, method = "igraph")
```

```
clustree(Esoph_filt, prefix = "SCT_snn_res.") # we are going to use resolution 0.2 for Esoph
```

```
DimPlot(Esoph_filt, group.by = "SCT_snn_res.0.2") & palette_d
DimPlot(Esoph_filt, group.by = "SCT_snn_res.0.3") & palette_d
DimPlot(Esoph_filt, group.by = "SCT_snn_res.0.4") & palette_d

DimPlot(Esoph_filt, group.by = "SCT_snn_res.0.2", split.by = "condition") & palette_d
```

```
# number of genes per cell
FeaturePlot(Esoph_filt, features = "nFeature_spliced") & palette # no huge heterogeneities
FeaturePlot(Esoph_filt, features = "percent.mt") & palette
```

Seeing again the plots after rerunning the analysis without the cells from clusters 4 and 7, we choose resolution 0.2 with 5 clusters.

Also, very important: cluster 5 is specific to condition Old_DEN

Now, we have 5 clusters in our data (resolution 0.2), which, after seeing their genes, could be annotated like: - Cluster 1: Basal (B) - Cluster 2 and 4: Basal prolifetaring (BP) - Cluster 3: differentiated (DIF) - Cluster 5: basal (B)

This is a premature proposal, we will decide this later.

```
# We use resolution 0.2
# Select resolution for Seurat Clusters:
Idents(Esoph_filt) = Esoph_filt$SCT_snn_res.0.2
Esoph_filt$seurat_clusters = Esoph_filt$SCT_snn_res.0.2
```

4.8 Cell cycle scoring

```
# Perform cell cycle scoring on Esoph_filt
setwd("/home/albax/mcGinn_2021")
exp.mat <- read.table(file = "./cell_cycle_vignette_files/nestorawa_forcellcycle_expressionMa

# A list of cell cycle markers, from Tirosh et al, 2015, is loaded with Seurat.  We can
# segregate this list into markers of G2/M phase and markers of S phase
s.genes <- cc.genes$s.genes
g2m.genes <- cc.genes$g2m.genes

Esoph_filt <- CellCycleScoring(Esoph_filt, s.features = s.genes, g2m.features = g2m.genes, s
```

```
DimPlot(Esoph_filt, group.by = "Phase")
```

4.9 Cluster annotation based on cellular composition, DGE, FeaturePlots...

The mouse esophageal mucosa consists of three layers: stratified epithelium, lamina propia with connective tissue, and muscularis mucosa with smooth muscle. In our data, the authors peeled the muscle (the muscularis mucosa) from the esophagi. So, we should only have the stratified epithelium and maybe lamina propia. The epithelium, in the lumen, is keratinized in mice.

We are going to create different signatures depending on: 1. Cell state (resting basal, cycling basal, differentiated) -> this is what the authors did in McGinn, 2021; or basal, differentiated, cell cycle - Basal: Cdh3, Itgb1, Krt15, Krt14, Krt5, Col17a1, Sox2, Trp63, Itga6 - Cell cycle: Gmnn, Mcm6, Mcm2, Cdt1, Pcna, Ccne1, E2f1, Ccne1, Cdc6, Aurkb, Top2a, Ccnb2, Bub1, Ube2c, Aurka, Kif23, Ccnb1, Mki67, Mad2l1, Birc5 - Differentiating: Krt13, Klf4, Tgm3, Sbsn, Grhl3, Krt4, Notch3, Krtdap

2. Structure of the epithelium: Stratified epithelium: Lumen

- Keratinized cells (final stage of granular): ; Lor, Ivl, Envoplakin, Periplakin, Sprr1a, Sprr1b, Sprr2a1, Sprr2a2, Sprr2a3, Sprr2b, Sprr2d, Sprr2e, Sprr2f, Sprr3; Lor, Flg, Tchp, Ivl, Capza1, S100A1, Sprr1a, Sprr1b, Sprr2a1, Sprr2a2, Sprr2a3, Sprr2b, Sprr2d, Sprr2e, Sprr2f, Sprr3
- Suprabasal cells -> differentiation
  - Granular: Lor, Flg, Ivl; Tgm3, Krt1, Krt2e, Krt9, Krt10, Dsg1, Dsc1
  - Spinous: Krt10, Krt1, Tgm1, Tgm5; Tgm1, Tgm5, Dsg2, Dsg3, Dsg4
- Basal cells (can be differentiating or be progenitor cells) -> proliferation, p63+,krt5+,krt7-; Bmi1 progenitor cells, Krt5, Krt14, Krt15; Krt5, Krt14, Tgm2, Bpag1; Itgb1, Trp63, Krt5, Krt14, Krt15; Krt5, Krt14, Bpag1, Tgm2
- Basal lamina: Lama5, Itga6, Itgb4, Bpag2 Basal

Disclaimer: keratinized cells and granular cells could be grouped in one category, the cornified envelope.

Summarised by taking the genes in common from the literature: Lumen

- Keratinized cells (final stage of granular): Lor, Ivl, Sprr1a, Sprr1b, Sprr2a1, Sprr2a2, Sprr2a3, Sprr2b, Sprr2d, Sprr2e, Sprr2f, Sprr3, Tchp, Capza1, S100A1, Evpl, Ppl
- Suprabasal cells -> differentiation
  - Granular: Tgm3, Krt1, Krt2e, Krt9, Krt10, Dsg1, Dsc1
  - Spinous: Krt10, Krt1, Tgm1, Tgm5, Dsg2, Dsg3, Dsg4
- Basal cells (can be differentiating or be progenitor cells) -> proliferation: Krt5, Krt14, Krt15, Tgm2, Bpag1, Tp63, Itgb1
- Basal lamina: Lama5, Itga6, Itgb4 Basal

```
# Taking into account the different sub-levels of the oesophagus epithelium; from Figure S1 I

# Lumen
# Keratinized cells (final stage of granular) :
# FeaturePlot(Esoph_filt, features = c("Sprr1a", "Sprr1b", "Sprr3", "Evpl", "Ppl")) & palette

# Differentiation markers:
FeaturePlot(Esoph_filt, features = c("Krt13",  "Tgm3", "Grhl3", "Krt4", "Notch3", "Klf4", "Sl

# Suprabasal cell markers
# FeaturePlot(Esoph_filt, features = c("Krt15", "Trp63", "Krt5", "Krt14")) & palette & plot_a

# Basal cell markers (can be proliferating or be progenitor cells):
FeaturePlot(Esoph_filt, features = c("Krt5", "Krt14", "Krt15", "Trp63", "Itgb1", "Itgb4", "Co
  theme(plot.title = element_text(size = 15, face = "bold", hjust = 0.5)) & palette #Mki67 ar

FeaturePlot(Esoph_filt, features = c("Mki67", "Cenpf", "Cenpa")) & plot_annotation(title = "I
```