Master's thesis in Bioinformatics and Computational Biology

Alba Méndez Alejandre

22/05/2025

Table of contents

1 Introduction

Brief description

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

- 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

This work applies variant calling from **scRNA-seq** to link genetic mutations to cellular phenotypes, using a **customized pipeline** on mouse and human esophageal data. It highlights the importance of experimental design and filtering for reliable mutation detection.

2 Objectives

The main objective of this work is **to assess whether single-cell transcriptomics is suitable for reliable somatic variant detection in mouse esophageal epithelium, and to associate genotype-to-phenotype relations using single cell RNA-seq.**

- Apply and customize variant caller for scRNA-seq in mouse and human esophagus.
- Characterize the diversity of transcriptomic states in the mouse esophageal epithelium.
- Map variants onto UMAP embeddings.

3 Tools & Technologies used

- (variant caller)
- `Seurat`, `scVelo`, `Slingshot` (transcriptomic analysis & trajectory inference)
- `VEP` (variant annotation)

4 Data

We obtained the HCA data from previous work (González-Menéndez, 2024), originally taken from: > Madissoon, E., Wilbrey-Clark, A., Miragaia, R.J. et al. scRNA-seq assessment of the human lung, spleen, and esophagus tissue stability after cold preservation. Genome Biol 21, 1 (2020). https://doi.org/10.1186/s13059-019-1906-x

The mouse data was obtained from a publicly available dataset.

5 Repository structure

The repository holds independent scripts for each dataset:

```
docs/ # quarto book

Introduction.qmd # installations, brief explanation of configurations used

human/ #  Human analysis
  1_Inspection.qmd # preliminary inspection of the dataset
  2_GeneExpression.qmd # calculate average gene expression for sets of genes
  3_VennDiagrams.qmd # obtain venn diagrams for sets of genes or mutated genes of interest
  4_UMAP_mapping.qmd # map mutated cells in the umap
  mut_clones_analysis_hca.qmd # modifying seurat_obj@meta.data to add clones

mouse/ #   Mouse analysis
  fastQC/
      run1/ # multiqc report for the first sequencing run
      run2/ # multiqc report for the second sequencing run
  1-4_merge_seurat_fixedrank.R # script to filter out droplets and doublets from the matri
  1_DataProcessing.qmd # notebook which contains the bash scripts used to download the fil
  2_ClusteringCellAnnotation.qmd # seurat pipeline to cluster the cells and annotate them
  3_Velocity_inference.qmd # infer velocity and pseudotime trajectory
  4_VariantCalling.qmd  # steps to perform variant calling and pre-filtering processing, a
  5_AnnotationVariants.qmd # exploration of the VEP output
  6_GOAnalysis.qmd # GO analysis of the mutated genes

 scripts/ # other general scripts
   annotate_vep.sh # automated annotation with vep
```

6 Set-up and installations

6.1 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.

6.2 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
```

6.2.1 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
```

6.2.2 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 tl
mamba activate d_scomatic

pip install -r requirements.txt
```

Part I

Mouse analysis

7 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.

7.1 1.1. Download fastQ files from ENA

```
#!/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
```

```
./download_fastq.sh
```

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

7.2 1.2 Quality control with MultiQC

View MultiQC report for sequencing run 2

View MultiQC report for sequencing run 1

7.3 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
```

```
./STAR_alignment.sh
```

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
```

7.4 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"]

8 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.

8.1 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")
```

8.2 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)
```

8.3 2.1. 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.

8.3.1 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$nCou

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

Visualize QC metrics

```
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")
```

```
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.

```
## 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.
```

8.4 2.2. 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
```

8.4.1 i. 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
```

8.4.2 ii. 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
```

8.4.3 iii. 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

8.4.3.1 Find Neighbours (KNN)

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

8.5 2.3. 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
```

8.6 2.4. 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
```

8.7 2.5. 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
```

8.8 2.6. 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")
```

8.9 2.7. 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

> **Tip**
>
> 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", "Sb

# 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", "Cd
  theme(plot.title = element_text(size = 15, face = "bold", hjust = 0.5)) & palette #Mki67 an

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

# Basal lamina cell markers
# FeaturePlot(Esoph_filt, features = c("Lama5", "Itga6", "Itgb4")) + plot_annotation(title =
#  theme(plot.title = element_text(size = 15, face = "bold", hjust = 0.5)) & palette
```

```
# Taking into account the three clusters from McGinn et al 2021 (CB, RB and DIF):

# Cycling basal markers(CB), cells that have high expression of cell cycle genes:
FeaturePlot(Esoph, features = c("Gmnn", "Mcm6", "Mcm2", "Cdt1", "Pcna", "Ccne1", "E2f1", "Cdc

# Resting basal markers (RB):
FeaturePlot(Esoph_filt, features = c("Col17a1", "Trp63", "Krt14", "Itga6", "Itgb1")) & plot_a

# Differentiation markers (DIF):
FeaturePlot(Esoph_filt, features = c("Tgm3", "Krt13", "Grhl3", "Krt4", "Notch3", "Klf4", "Sbs
```

```
## Dot plot of some characteristic markers:
DotPlot(Esoph, features = c("Mki67","Krt14","Col17a1","Krt5","Sbsn","Krtdap"), cols = c("grey
```

8.10 2.8. SComatic preparation

SComatic needs a TSV metadata file that relates each cell barcode with each "Cell type" annotation. The annotation will depend on how much granularity we need or is adequate to cal lsomatic variants from single cell transcriptomics. Here, the library design plays a very important role.

8.10.1 Save cell barcodes with cell types in TSV format (SComatic)

For SComatic, save cell barcodes with cell types: (first column is barcode and second column is cell type)

```
# colnames <- c("Index","Cell_type")

#df <- data.frame(Index = Cells(Esoph_filt), Cell_type = Esoph_filt@meta.data$cellType)

# delete suffix??
# df$"Index" <- sapply(strsplit(df$"Index", "-"), `[`, 1)

# write.table(df, file = './output/esoph_celltype.tsv', col.names = TRUE, row.names = FALSE,
```

8.11 2.9. Annotation of clusters (based on known canonical markers of each cluster)

Annotate each mouse .tsv "Cell_type" as: Epi; so that in Step 3 of SComatic we obtain a .tsv with 1 "cluster"

```
 ## Rearrange cluster order to follow differentiation axis (for visualization purposes) # 9
levels(Esoph_filt) <- c(2,4,1,3,5) # from less differentiated to most (basal to luminal)

# we have 5 clusters in Esoph_filt
# now, we want to just annotate Epi (all the cells in our Esoph_filt are Epithelial)
new.cluster.id <- c("Epi", "Epi", "Epi", "Epi", "Epi")

names(new.cluster.id) <- levels(Esoph_filt)
Esoph_filt <- RenameIdents(Esoph_filt, new.cluster.id)

Esoph_filt[["cellType"]] <- Idents(Esoph_filt) # include it in the metadata

# Create a new meta.data column named cellType_B that adds _lib1 if the library_ID is llib1,
Esoph_filt$cellType_B <- paste0(Esoph_filt$Library_ID, "_", Esoph_filt$cellType)
Esoph_filt$cellType_B <- as.factor(Esoph_filt$cellType_B)
```

```
# Get the unique sample IDs
sample_names <- unique(Esoph_filt@meta.data$Sample_name)

# Create the .tsv files for each sample (we will have 16 different tsv files, one for each .
tsv_file <- function(seurat_obj, sample_names) {
  for (sample in sample_names) {
    # Filter the data for the current sample
    sample_data <- seurat_obj@meta.data %>%
      filter(Sample_name == sample)
```

```
    # Add the Index column with the correct rownames and apply the transformation
    sample_data$Index <- sapply(strsplit(rownames(sample_data), "-"), `[`, 1)

    # Select the required columns
    sample_data <- sample_data %>%
      select(Index, Cell_type = cellType_B) # Cell_type is a meta.data feature like "cellType

    # Define the file name
    sample_parts <- strsplit(sample, "_")[[1]]
    file_name <- paste0("./output/esoph_markers_scomatic_", sample_parts[2], "_", sample_part

    # Write the data to a .tsv file
    write.table(sample_data, file = file_name, sep = "\t", row.names = FALSE, col.names = TRU
  }
}

tsv_file(Esoph_filt, sample_names)
```

```
# Check if we have taken all the rows:
output_dir <- "./output/"

# Get a list of all .tsv files in the directory
tsv_files <- list.files(path = output_dir, pattern = "^esoph_markers_scomatic_.*\\.tsv$", ful

# Initialize a data frame to store the file names and row counts
row_counts <- data.frame(File = character(), Rows = integer(), stringsAsFactors = FALSE)

# Loop through each file, read the data, and count the rows
for (file in tsv_files) {
  # Read the data from the .tsv file
  data <- read.table(file, sep = "\t", header = TRUE)

  # Get the number of rows
  num_rows <- nrow(data)

  # Add the file name and row count to the data frame
  row_counts <- row_counts %>%
    add_row(File = basename(file), Rows = num_rows)
}

total_rows <- sum(row_counts$Rows)
```

```
total_rows

dim(Esoph_filt)
```

8.12 2.10. scVelo preparation

8.12.1 Cluster annotation for scvelo

Looking at our umap, with resolution 0.2, we will take 3 clusters (Basal, Suprabasal, Epi_DEN), but annotated differently, in 5 names: - Cluster 1: “Basal 1” - Cluster 2: “Basal 2” - Cluster 3: “Suprabasal” - Cluster 4: “Basal 3” - Cluster 5: “Epithelial_DEN”

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

```
# make sure we are using the idents we want right now:
Idents(Esoph_filt) <- Esoph_filt$SCT_snn_res.0.2
Esoph_filt$seurat_clusters <- Esoph_filt$SCT_snn_res.0.2

## Rearrange cluster order to follow differentiation axis (for visualization purposes)
levels(Esoph_filt) <- c(1, 2, 4, 3, 5) # from less differentiated to most (basal to luminal)
# clusters 1 2 and 4 are basal (red, orange and blue)
# cluster 3 (green) is suprabasal
# cluster 5 is specific of condition sample_DEN

new.cluster.ids <- c("Basal_1", "Basal_2", "Basal_3", "Suprabasal", "Epithelial_DEN") # 1, 2
names(new.cluster.ids) <- levels(Esoph_filt)
Esoph_filt <- RenameIdents(Esoph_filt, new.cluster.ids)
```

```
Idents(Esoph_filt) <- Esoph_filt$annot_scvelo

# Annotated PCA and UMAP plots:
DimPlot(Esoph_filt, reduction = "pca", label = TRUE, pt.size = 0.5) + NoLegend() & palette_d

DimPlot(Esoph_filt, reduction = "umap", label = TRUE, pt.size = 0.5) & palette_d
```

```
DimPlot(Esoph_filt, reduction = "umap", label = TRUE, pt.size = 0.5, split.by = "condition")
```

```
## Remaining annotation using top markers for every cluster compared to all remaining ones,
Esoph_filt.markers <- FindAllMarkers(Esoph_filt, only.pos = TRUE, min.pct = 0.25, logfc.thres
# write.table(Esoph_filt.markers, file = './output/Esoph_markers_filt.txt', col.names = TRUE
Esoph_filt.markers <- read.csv("./output/Esoph_markers_filt.txt", sep = "\t")

TopMarkers <- Esoph_filt.markers %>% group_by(cluster) %>% top_n(n = 25, wt = avg_log2FC) #

TopMarkers %>% write.csv("/home/albax/mcGinn_2021/output/Esoph_TopMarkers.csv")
clusterB1_markers <- TopMarkers[TopMarkers$cluster == "Basal_1", ]  # Basal_1
clusterB2_markers <- TopMarkers[TopMarkers$cluster == "Basal_2", ]  # Basal_2
clusterB3_markers <- TopMarkers[TopMarkers$cluster == "Basal_3", ]  # Basal_3
clusterSB_markers <- TopMarkers[TopMarkers$cluster == "Suprabasal", ]  # Suprabasal
clusterEpDEN_markers <- TopMarkers[TopMarkers$cluster == "Epithelial_DEN", ]  # Epithelial_DE
```

```
Esoph_filt.markers <- read.csv("./output/Esoph_markers_clusts.txt", sep = "\t")

TopMarkers <- Esoph_filt.markers %>% group_by(cluster) %>% top_n(n = 25, wt = avg_log2FC) #

B1_markers <- TopMarkers[TopMarkers$cluster == 'Basal_1', ]
B2_markers <- TopMarkers[TopMarkers$cluster == 'Basal_2', ]
B3_markers <- TopMarkers[TopMarkers$cluster == 'Basal_3', ]
DEN_markers <- TopMarkers[TopMarkers$cluster == 'Epithelial_DEN', ]
suprabasal_markers <- TopMarkers[TopMarkers$cluster == 'Suprabasal', ]
```

8.12.2 Conversion into h5ad object (export to scvelo)

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

# Esoph_filt@meta.data$annot_scvelo <- as.factor(Esoph_filt@meta.data$annot_scvelo)

library(SeuratDisk) # facilitates conversion between h5Seurat and AnnData objects, i.e. inte

# Make RNA assay (raw counts, which is a copy of spliced assay) default:
DefaultAssay(Esoph_filt) <- "RNA"

remove_scaledata <- function(assay) {
    assay@scale.data <- matrix(nrow = 0, ncol = 0)
    return(assay)
}
```

```
counts_to_integer <- function(assay) {
    assay@counts@x <- as.integer(assay@counts@x)
    return(assay)
}

remove_normalization <- function(assay) {
    assay@data <- assay@counts
    return(assay)
}

Esoph_filt@assays <- lapply(Esoph_filt@assays, remove_scaledata)
Esoph_filt@assays <- lapply(Esoph_filt@assays, counts_to_integer)
Esoph_filt@assays <- lapply(Esoph_filt@assays, remove_normalization)

# Add a new metadata column so that cell types are stored as strings, and not as numbers in t
Esoph_filt@meta.data$annot_scvelo_names <- as.character(Esoph_filt@meta.data$annot_scvelo)

# File conversion:
SaveH5Seurat(Esoph_filt, filename = "Esoph.h5Seurat", overwrite=TRUE)
Convert("Esoph.h5Seurat", dest = "h5ad", overwrite=TRUE)
```

Save seurat object:

```
saveRDS(Esoph_filt, file = "./output/esoph_star_filtered_mm10.rds")
```

8.13 2.11. Normalization of counts

We are going to normalize the data by total counts (in each library-Sequencing ID). For that, we have to sum all the columns.

```
levels(Esoph_filt@meta.data$Sample_name)
summary(Esoph_filt[,Esoph_filt@meta.data$Sample_name == 'lib2_run1']@meta.data$Sample_name)

results_df <- data.frame(Sample_name = character(), Sum_nCount_gene = numeric(), Sum_nFeature
cell_counts_per_sample <- table(Esoph_filt@meta.data$Sample_name) # number of cells per sampl

# suma de los nCount_gene para cada Sample_name
for (i in unique(Esoph_filt$Sample_name)) {
  # Subset the data based on the current sample name and calculate the sum of nCount_gene
  sum_count <- sum(Esoph_filt[, Esoph_filt$Sample_name == i]$nCount_gene)
```

```
  sum_feature <- sum(Esoph_filt[, Esoph_filt$Sample_name == i]$nFeature_gene)
  num_cells <- cell_counts_per_sample[i]

  # Append the results to the results_df
  results_df <- rbind(results_df, data.frame(Sample_name = i, Sum_nCount_gene = sum_count, Nu
}
numeric_medians <- apply(results_df[, -1], 2, median)
median_row <- c(Sample_name = "Median", numeric_medians)
results_df <- rbind(results_df, median_row)

write.csv(results_df, "nCounts_per_Sample_name_Esophfilt.csv", row.names = FALSE)
```

Supplementary table from the original paper nCount total is going to ve similar to column 4 * column 6

Supplementary Table 1 - QC Statistics of scRNAseq data (https://www.nature.com/articles/s41556-021-00679-w#Sec31)

Stage	Replicate no.	Batch no.	Number of cells	Median genes/cell	Median UMIs/cell	Total genes detected	Median % mito counts/cell
Adult	1	1	2796	5153	30737	17419	3.66549437744719
Adult	2	1	3344	5232	32745.5	17574	3.76114454638533
Adult	3	1	4059	5070	29744	17843	4.14769410907473
Adult	1	2	1230	5215	33187.5	16440	3.63571622273852
Adult	2	2	2614	5219	33970	17191	3.25793271016975
Adult	3	2	*676*	5438	*36246.5*	15707	3.5389684657119

Warning:

SCTransform corrects the counts from your equivalent RNA assay and creates a new assay (typically SCT) where the counts slot is a corrected counts, data is a log transformation of corrected counts+1 and the scale.data are pearson residuals. Typically, the scale.data slot is only generated for the features listed in VariableFeatures(your_object) which is why it's usually smaller than your SCT data slot. You can tell SCTransform to scale all genes, but whether that's something you need or not is up to you.

8.14 2.12. Do we find barcodes from Epi_DEN shared in other clusters?

- Is each clone contained completely inside the Epithelial_DEN cluster?("Epi_DEN_cells")

- How are these clones shared with other clusters? How many cells of the clone fall inside Epi_DEN_cells cluster?
-

```
epi_den_CBs <- WhichCells(Esoph_filt, idents = "Epithelial_DEN")
write.csv(epi_den_CBs, "Epithelial_DEN_CBS.txt", row.names = FALSE)

metadata <- data.frame(Esoph_filt@meta.data)
metadata_cols <- metadata[,c("clones", "annot_scvelo"), drop = FALSE]
write.csv(metadata_cols, "clones_clusters.csv", row.names = TRUE)

clones_data <- metadata[epi_den_CBs, c("clones", "annot_scvelo"), drop = FALSE]

clones_in_Epi_DEN <- as.character(clones_data[["clones"]])
clones_in_Epi_DEN <- unique(unlist(strsplit(clones_in_Epi_DEN, ",")))
```

9 3. Inference of velocity

9.1 3.1. Slingshot

9.1.1 Import libraries

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

library(Seurat)
library(slingshot)

library(grDevices)
library(RColorBrewer)
library(ggplot2)
library(dplyr)
library(viridis)
```

9.1.2 Load filtered seurat object

```
Esoph <- readRDS(file = "mcGinn_2021/output/esoph_star_filtered_mm10.rds")

# Convert SeuratObject to SingleCellExperiment with Seurat function 'as.SingleCellExperiment
Esoph_sce <- as.SingleCellExperiment(Esoph)

# view idents
Idents(Esoph) # Basal_1, Basal_2, Basal_3, Suprabasal, Epithelial_DEN

# view metadata columns from sce experiment object, in order to know which one to use
Esoph_sce@colData

# we will use annot_scvelo which has 5 levels: Basal_1, Basal_2, Basal_3, Suprabasal, Epithel
Esoph_sce@colData$annot_scvelo
```

```
# add umap data to sce object
umap_data <- Embeddings(Esoph[["umap"]])  # Extract UMAP data
reducedDims(Esoph_sce)$UMAP <- umap_data

factor(Esoph_sce$annot_scvelo)
```

```
# Run trajectory inference with slingshot
set.seed(1)
Esoph_sce <- slingshot(Esoph_sce, clusterLabels = 'annot_scvelo', reducedDim = 'UMAP')
Esoph_sce$slingParams["star.clus"]
Esoph_sce$slingParams["end.clus"]

slingshot_obj <- SlingshotDataSet(Esoph_sce)
slingshot_obj@slingParams$start.clus
slingshot_obj@slingParams$end.clus
```

9.1.3 Plot results

```
# Plot trajectory (curves how they are called by Slingshot)
png("slingshot_results/plots/Esoph_slingshot_mm10.png", width=1000, height=1000, units="px")

breaks <- seq(min(slingPseudotime(Esoph_sce, na=FALSE), na.rm = TRUE),
              max(slingPseudotime(Esoph_sce, na=FALSE), na.rm = TRUE),
              length.out = 100)
viridis_colors <- magma(100)
plotcol <- viridis_colors[cut(slingPseudotime(Esoph_sce, na=FALSE), breaks = breaks)]

layout(matrix(c(1, 2), ncol = 2), widths = c(2, 0.25), height = c(1,0.2))
par(mar=c(5, 4, 4, 1), xpd=TRUE)  # Adjust margins for the main plot
plot(reducedDims(Esoph_sce)$UMAP, col = plotcol, pch=16, asp = 1)

lines(SlingshotDataSet(Esoph_sce), lwd=2, col='black')
lines(SlingshotDataSet(Esoph_sce), type = 'lineages', lwd=2, col='black')

par(mar = c(5, 1, 4, 1))
plot.new()
plot.window(xlim = c(0, 1), ylim = c(0, 1))

legend_image <- as.raster(viridis_colors, ncol=1)
legend_width <- 0.7
```

```
legend_height <- 0.25
x_left <- (1 - legend_width) / 2
x_right <- x_left + legend_width
y_bottom <- (1 - legend_height) / 2
y_top <- y_bottom + legend_height
rasterImage(legend_image, x_left, y_bottom, x_right, y_top)
text(x = 0.5, y = 0.65, labels = "Legend", cex = 1.2, font = 2, pos = 3)

dev.off()
```

```
# Plot trajectory (curves how they are called by Slingshot)
png("slingshot_results/plots/Esoph_slingshot_curves_mm10.png", width=1000, height=1000, units

breaks <- seq(min(slingPseudotime(Esoph_sce, na=FALSE), na.rm = TRUE),
              max(slingPseudotime(Esoph_sce, na=FALSE), na.rm = TRUE),
              length.out = 100)
viridis_colors <- magma(100)
plotcol <- viridis_colors[cut(slingPseudotime(Esoph_sce, na=FALSE), breaks = breaks)]

layout(matrix(c(1, 2), ncol = 2), widths = c(2, 0.25), height = c(1,0.2))
par(mar=c(5, 4, 4, 1), xpd=TRUE)  # Adjust margins for the main plot
plot(reducedDims(Esoph_sce)$UMAP, col = plotcol, pch=16, asp = 1)

lines(SlingshotDataSet(Esoph_sce), lwd=2, col='black')
# lines(SlingshotDataSet(Esoph_sce), type = 'lineages', lwd=2, col='black')

par(mar = c(5, 1, 4, 1))
plot.new()
plot.window(xlim = c(0, 1), ylim = c(0, 1))

legend_image <- as.raster(viridis_colors, ncol=1)
legend_width <- 0.7
legend_height <- 0.25
x_left <- (1 - legend_width) / 2
x_right <- x_left + legend_width
y_bottom <- (1 - legend_height) / 2
y_top <- y_bottom + legend_height
rasterImage(legend_image, x_left, y_bottom, x_right, y_top)
text(x = 0.5, y = 0.65, labels = "Legend", cex = 1.2, font = 2, pos = 3)

dev.off()
```

9.1.4 Plot lineage structure

```
palette_d <- c("#7A0403FF", "#FB8022FF", "#A2FC3CFF", "#28BBECFF", "#30123BFF")

png("./velocity/slingshot_results/plots/Esoph_slingshot_trajectories_clusters_mm10.png", widt

plot(reducedDims(Esoph_sce)$UMAP, col = palette_d[as.numeric(droplevels(Esoph_sce$annot_scvel

lines(SlingshotDataSet(Esoph_sce), lwd=2, type = 'lineages', col = 'black')

legend(x="topright", legend=c("Basal_1", "Basal_2", "Basal_3", "Suprabasal", "Epithelial_DEN

dev.off()
```

9.2 3.2. scVelo

9.2.1 Load of .h5ad file

```
#!/home/albax/miniforge3/envs/ame python3
# -*- coding: utf-8 -*-
"""
Created on Jul 26 2024

@author: albax
"""
import sys
import os

# Add the site-packages directory to sys.path
specific_path = "/home/albax/miniforge3/envs/ame/lib/python3.10/site-packages"
if specific_path not in sys.path:
    sys.path.append(specific_path)

sys.path.insert(0, '/home/albax/scvelo/scvelo')

import scvelo as scv
import scanpy as sc
import scipy.sparse as sp
import matplotlib.pyplot as plt
```

```
import seaborn as sns
import numpy as np
from matplotlib.colors import ListedColormap

adata = scv.read("../Esoph.h5ad", cache=True)

adata
adata.layers.keys() # should include spliced and unspliced
adata.layers['spliced'] # check presence
adata.layers['unspliced'] # check presence
adata.var_names = adata.var['_index']

scv.pp.filter_and_normalize(adata, min_shared_counts=20, n_top_genes=2000)

######## deprecated:
# scv.pp.moments(adata, n_pcs=30, n_neighbors=30)
#################

# 2. Compute PCA with Scanpy
sc.pp.pca(adata, n_comps=30)

# 2. Compute neighbors with Scanpy
sc.pp.neighbors(adata, n_pcs=30, n_neighbors=30, use_rep='X_pca')

scv.pp.moments(adata, n_pcs=30, n_neighbors=30) # uses connectivites from step before

# rename levels from annot_scvelo slot
# rename_dict = {
#     0: 'Basal_1', # 1
#     1: 'Basal_2', # 2
#     2: 'Basal_3', # 3
#     3: 'Suprabasal', # 4
#     4: 'Epithelial_DEN', # 5
# }

# adata.obs['annot_scvelo'] = adata.obs['annot_scvelo'].replace(rename_dict)
```

Define functions and colors for plotting:

```
def save_stream(adata, file, format=["svg", "png"], **kwargs):
    fig, ax = plt.subplots(figsize=(9, 7))
    scv.pl.velocity_embedding_stream(adata, ax=ax, arrow_size=1, cutoff_perc=10, **kwargs)
```

```
    for extension in format:
        file_name = file + "." + extension
        fig.savefig(file_name, format=extension, bbox_inches='tight')

    return fig, ax


def save_grid(adata, file, format=["svg", "png"], **kwargs):
    fig, ax = plt.subplots(figsize=(9, 7))
    scv.pl.velocity_embedding_grid(adata, ax=ax, arrow_length=5, arrow_size=3, **kwargs)

    for extension in format:
        file_name = file + "." + extension
        fig.savefig(file_name, format=extension, bbox_inches='tight')

    return fig, ax


def save_embedding(adata, file, format=["svg", "png"], **kwargs):
    fig, ax = plt.subplots(figsize=(9, 7))
    scv.pl.velocity_embedding(adata, ax=ax, arrow_length=5, arrow_size=3, **kwargs)

    for extension in format:
        file_name = file + "." + extension
        fig.savefig(file_name, format=extension, bbox_inches='tight')

    return fig, ax


palette = sns.color_palette("plasma_r")
colors_d = ["#7A0403FF", "#FB8022FF", "#A2FC3CFF", "#30123BFF", "#28BBECFF"]
palette_d = sns.color_palette(colors_d)
```

9.2.2 i. Stochastic model

```
scv.tl.velocity(adata, mode ="stochastic") # not good model
scv.tl.velocity_graph(adata, n_jobs=6) # number of cores to use

# scv.pl.pca(adata, color="annot_scvelo")
# scv.pl.umap(adata, color="annot_scvelo")
```

9.2.2.1 Save plots for stochastic model

```
fig_stream_stoc, ax_stream_stoc = save_stream(adata, file="scvelo_results/plots/Esoph_stream_

fig_grid_stoc, ax_grid_stoc = save_grid(adata, file="scvelo_results/plots/Esoph_grid_stoch",

fig_embedding_stoc, ax_embedding_stoc = save_embedding(adata, file="scvelo_results/plots/Esop
```

```
# grid layout for stoc (run everything as chunk)
fig_combined, axs = plt.subplots(1, 3, figsize=(27, 7))

axs[0].imshow(fig_stream_stoc.canvas.renderer.buffer_rgba())
axs[0].axis('off')
axs[1].imshow(fig_grid_stoc.canvas.renderer.buffer_rgba())
axs[1].axis('off')
axs[2].imshow(fig_embedding_stoc.canvas.renderer.buffer_rgba())
axs[2].axis('off')

combined_file_path = "scvelo_results/plots/Esoph_combined_plots_stoc"
fig_combined.tight_layout()
fig_combined.savefig(combined_file_path + '_mm10.svg', format='svg', bbox_inches='tight')
fig_combined.savefig(combined_file_path + '_mm10.png', format='png', dpi=300, bbox_inches='ti
plt.show()
```

Save the .h5ad file with stochastic model

```
del adata.raw
adata.var.rename(columns={'_index': 'index'}, inplace=True)
adata.obs.rename(columns={'_index': 'index'}, inplace=True)

adata.write(filename = "scvelo_results/Esoph_stochastic_mm10.h5ad", compression='gzip')
```

9.2.3 ii. Deterministic model

```
scv.tl.velocity(adata, mode = "deterministic")
scv.tl.velocity_graph(adata, n_jobs=6)
# scv.pl.velocity_embedding_stream(adata, basis="umap", color="annot_scvelo")

# scv.pl.pca(adata, color="annot_scvelo")
# scv.pl.umap(adata, color="annot_scvelo")
```

9.2.3.1 Save plots of deterministic model

```
fig_stream_det, ax_stream_det = save_stream(adata, file="scvelo_results/plots/Esoph_stream_de

fig_grid_det, ax_grid_det = save_grid(adata, file="scvelo_results/plots/Esoph_grid_det", form

fig_embedding_det, ax_embedding_det = save_embedding(adata, file="scvelo_results/plots/Esoph_
```

```
# grid layout for deterministic (run everything as chunk)
fig_combined, axs = plt.subplots(1, 3, figsize=(27, 7))

axs[0].imshow(fig_stream_det.canvas.renderer.buffer_rgba())
axs[0].axis('off')
axs[1].imshow(fig_grid_det.canvas.renderer.buffer_rgba())
axs[1].axis('off')
axs[2].imshow(fig_embedding_det.canvas.renderer.buffer_rgba())
axs[2].axis('off')

combined_file_path = "scvelo_results/plots/Esoph_combined_plots_det"
fig_combined.tight_layout()
fig_combined.savefig(combined_file_path + '_mm10.svg', format='svg', bbox_inches='tight')
fig_combined.savefig(combined_file_path + '_mm10.png', format='png', dpi=300, bbox_inches='ti
plt.show()
```

Save the .h5ad file with dererministic model

```
del adata.raw
adata.var.rename(columns={'_index': 'index'}, inplace=True)
adata.obs.rename(columns={'_index': 'index'}, inplace=True)

adata.write(filename = "scvelo_results/Esoph_deterministic_mm10.h5ad", compression='gzip')
```

9.2.4 iii. Dynamical model

```
adata = adata[:, adata.var['highly_variable']]  # Only use highly variable genes
scv.tl.recover_dynamics(adata, n_jobs=6)
scv.tl.velocity(adata, mode = "dynamical")
scv.tl.velocity_graph(adata, n_jobs=6)
```

```
scv.pl.velocity_embedding_stream(adata, basis="umap", color="annot_scvelo_names")

# scv.pl.pca(adata, color="annot_scvelo")
# scv.pl.umap(adata, color="annot_scvelo")
```

9.2.4.1 Save plots for dynamical model

```
fig_stream_dyn, ax_stream_dyn = save_stream(adata, file="scvelo_results/plots/Esoph_stream_dy

fig_grid_dyn, ax_grid_dyn = save_grid(adata, file="scvelo_results/plots/Esoph_grid_det", forn

fig_embedding_dyn, ax_embedding_dyn = save_embedding(adata, file="scvelo_results/plots/Esoph_
```

```
# grid layout for dynamical model (run everything as chunk)
fig_combined, axs = plt.subplots(1, 3, figsize=(27, 7))

axs[0].imshow(fig_stream_dyn.canvas.renderer.buffer_rgba())
axs[0].axis('off')
axs[1].imshow(fig_grid_dyn.canvas.renderer.buffer_rgba())
axs[1].axis('off')
axs[2].imshow(fig_embedding_dyn.canvas.renderer.buffer_rgba())
axs[2].axis('off')

combined_file_path = "scvelo_results/plots/Esoph_combined_plots_dyn"
fig_combined.tight_layout()
fig_combined.savefig(combined_file_path + '_mm10.svg', format='svg', bbox_inches='tight')
fig_combined.savefig(combined_file_path + '_mm10.png', format='png', dpi=300, bbox_inches='ti
plt.show()
```

Save the .h5ad file with dynamical model

```
del adata.raw
adata.var.rename(columns={'_index': 'index'}, inplace=True)
adata.obs.rename(columns={'_index': 'index'}, inplace=True)

adata.write(filename = "scvelo_results/Esoph_dynamical_mm10.h5ad", compression='gzip')
```

9.2.4.2 Coherence/Confidence

```
adata_re = scv.read("scvelo_results/Esoph_dynamical_mm10.h5ad", cache=True)
adata_re.var_names = adata_re.var['index']
adata_re
```

```
scv.tl.velocity_confidence(adata)
scv.pl.scatter(adata, basis='umap', color=['velocity_length', 'velocity_confidence'], color_r

adata.obs['annot_scvelo_names'] = adata.obs['annot_scvelo_names'].astype('category')
scv.tl.rank_velocity_genes(adata, groupby='annot_scvelo_names', min_corr=0.3)
df = scv.get_df(adata.uns['rank_velocity_genes']['names'])
df.to_csv("scvelo_results/Esoph_dyn_genes_rank.csv")
```

9.2.5 Full plots

```
import matplotlib.pyplot as plt
import io

def get_image_data(fig):
    """ Convert a Matplotlib figure to an image array. """
    buf = io.BytesIO()
    fig.savefig(buf, format='png', bbox_inches='tight', pad_inches=0)
    buf.seek(0)
    img = plt.imread(buf)
    buf.close()
    return img

for fig in [fig_stream_dyn, fig_grid_dyn, fig_embedding_dyn,
            fig_stream_det, fig_grid_det, fig_embedding_det,
            fig_stream_stoc, fig_grid_stoc, fig_embedding_stoc]:
    fig.canvas.draw()

fig_combined, axs = plt.subplots(3, 3, figsize=(27, 21))

image_data = [
    get_image_data(fig_stream_dyn),
    get_image_data(fig_grid_dyn),
    get_image_data(fig_embedding_dyn),
```

```
    get_image_data(fig_stream_det),
    get_image_data(fig_grid_det),
    get_image_data(fig_embedding_det),
    get_image_data(fig_stream_stoc),
    get_image_data(fig_grid_stoc),
    get_image_data(fig_embedding_stoc)
]

for ax, img in zip(axs.flat, image_data):
    ax.imshow(img)
    ax.axis('off')  # Hide the axes


fig_combined.tight_layout()
fig_combined.savefig("scvelo_results/plots/Esoph_combined_plots_mm10.png", format='png', dpi=
fig_combined.savefig("scvelo_results/plots/Esoph_combined_plots_mm10.svg", format='svg', dpi=
fig_combined.savefig("scvelo_results/plots/Esoph_combined_plots_mm10.pdf", format='pdf', dpi=
plt.show()
```

9.2.6 iv. Cycling progenitors

```
scv.tl.score_genes_cell_cycle(adata_re)

# Plot cycling progenitors
scv.pl.scatter(adata_re, color_gradients=['S_score', 'G2M_score'], edgecolor='gainsboro', lin
scv.pl.scatter(adata_re, color_gradients=['S_score', 'G2M_score'], edgecolor='gainsboro', lin

scv.pl.scatter(adata_re, basis='umap', color='phase', palette=['silver', 'coral', 'royalblue
scv.pl.scatter(adata_re, basis='umap', color='phase', palette=['silver', 'coral', 'royalblue
```

10 4. SComatic calling

We are going to perform somatic variant calling with the novel algorithm from Muyas et.al (2024) doi: https://doi.org/10.1038/s41587-023-01863-z

10.1 4.1. Script to call somatic variants

- min_ac_cells = 2 (value by default); in HCA was used 1
- min_cell_types = 2
- max_cell_types = 1

```
#!/bin/bash
# Script with bams separated by run (run1 vs run2). Each run bams are analyzed in a scomatic
# There are 6 sample which are control adult, + one sample is old CTL, and + one sample is ol
# Each of the sample has the key like "sample1_Epi", "samp2_Epi", etc.
# DISCLAIMER: ALWAYS USE THE SAME REFERENCE GENOME!!! If you have aligned you rreads with mm
# SComatic nowadays has only panel of normales (PON) and editing sites (A-to-I events) for: C
# The PON for human has been made with GATK data 1000 genomes project. No info of how they di
# The PON for mouse mm10 was done in July 20, using mm10 Tabula muris data. No info of how ex

# Run this script in a environment with scomatic installed.

SCOMATIC=~/bin/SComatic-main

run_names=("run1" "run2") # list of runs

echo "Starting Scomatic analysis."


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

  # iterate though each bam in the directory where they are located
```

```
bam_dir="/media/storage/mcGinn_2021/STARalignment/bam/${run}"
output_dir="/media/storage/mcGinn_2021/scomatic/output/${run}"

mkdir -p $output_dir

logfile="$output_dir/scomatic_log.txt"
mkdir -p "$(dirname "$logfile")"

log_message() {
    local log_time
    log_time=$(date +"%Y-%m-%d %H:%M:%S")
    local message="[$log_time] $1"
    echo "$message" >> "$logfile"
    echo "$message"
}

# Initialize log file
echo "### Scomatic Pipeline Log ###" > "$logfile"
log_message "Running scomatic pipeline for run: ${run}"


# Step 1: Splitting alignment file in cell type specific bams
log_message "Step 1: Splitting alignment file in cell type specific bams..."

output_dir1=$output_dir/Step1_BamCellTypes
mkdir -p $output_dir1
meta_dir="/media/storage/mcGinn_2021/scomatic/markers"

# Iterate through each BAM file in the bam_dir
for bam_file in "$bam_dir"/*.bam
do
    # Extract the sample name from the file name
    base_name=$(basename "$bam_file" Aligned.sortedByCoord.out.bam) # run1_lib1
    sample=${base_name#*_} # remove the prefix up to '_', so that we get only lib1

    echo "Started sample: $sample"

    meta_file="$meta_dir/${run}/esoph_markers_scomatic_${base_name}.tsv"

    # Run the Python script with the appropriate arguments
    python $SCOMATIC/scripts/SplitBam/SplitBamCellTypes.py --bam "$bam_file" \
        --meta "$meta_file" \
```

```
        --id "$sample" \
        --n_trim 5 \
        --max_nM 5 \
        --max_NH 1 \
        --outdir "$output_dir1"

    log_message "Finished sample: $sample"

    sample_list+=("$sample")
done

# Now, we have each .bam file for each sample located in =$output_dir/Step1_BamCellTypes. I
# The bam file will be like: run1_lib1.lib1_Epi.bam


########################################
# Step 2: Collecting base count information

REF=/media/storage/reference_genomes/Mus_musculus.GRCm38.dna.primary_assembly.fa

output_dir2=$output_dir/Step2_BaseCellCounts
mkdir -p $output_dir2

for bam in "$output_dir1"/*.bam
do

  # Cell type, which will be lib1 in this case
  cell_type=$(basename $bam | awk -F'.' '{print $(NF-1)}') # this takes the cell_type from

  # Temp folder
  temp=$output_dir2/temp_${cell_type}
  mkdir -p $temp

  log_message "Processing base counts for cell type: $cell_type"

  # Command line to submit to cluster
  python $SCOMATIC/scripts/BaseCellCounter/BaseCellCounter.py --bam $bam \
    --ref $REF \
    --chrom all \
    --out_folder $output_dir2 \
    --min_bq 30 \
    --tmp_dir $temp \
```

```
    --nprocs 10

  rm -rf $temp

  log_message "Finished base count processing for cell type: $cell_type"
done

log_message "Listing files in $output_dir2 before renaming:"
ls "$output_dir2"
# Now, we have each .tsv for each sample in output_dir2=$output_dir/Step2_BaseCellCounts
# We are going to move both .tsv from the same sample to a new folder inside output_dir2, r

# echo "Organizing .tsv files into sample-specific folders..."
log_message "Renaming .tsv files..."


for tsv in "$output_dir2"/*.tsv
do
    # Create a new directory for the sample
    # Move the .tsv files corresponding to the sample into the sample directory

    base_name=$(basename "$tsv")

    # lib1.lib1_Epi.tsv ; we want lib1.tsv -> this will be our unique "cell types"

    # Extract run name, and spec cell
    sample_run=$(echo "$base_name" | awk -F'.' '{print $1}') # extracts lib1
    spec_cell=$(echo "$base_name" | awk -F'_' '{print $2}') # extracts Epi.tsv

    new_filename="${output_dir2}/${sample_run}_${spec_cell}" # lib1_Epi.tsv

    # Change each name of the .tsv files by adding its run
    # mv "$output_dir2/${sample_name}"*.tsv "$sample_dir/${run_name}${sample_name}"*.tsv
    mv "$tsv" "$new_filename"

    # echo "Moved .tsv files for sample: $sample to $sample_dir"
    log_message "Renamed $tsv to $new_filename"
done

log_message "Listing files in $output_dir2 after renaming:"
ls "$output_dir2"
```

```
# echo "Moved the files successfully!"
log_message "Renamed the files successfully!"

########################################
# Step 3: Merging base count matrices
log_message "Step 3: Merging base count matrices..."

output_dir3=$output_dir/Step3_BaseCellCountsMerged
mkdir -p $output_dir3

log_message "Merging base count matrices into a single .tsv file..."

python $SCOMATIC/scripts/MergeCounts/MergeBaseCellCounts.py \
  --tsv_folder ${output_dir2} \
  --outfile ${output_dir3}/${run}.BaseCellCounts.AllCellTypes.tsv


########################################
# Step 4: Detection of somatic mutations
log_message "Step 4: Detection of somatic mutations..."

# Step 4.1
log_message "Step 4.1: Variant calling..."
output_dir4=$output_dir/Step4_VariantCalling
mkdir -p $output_dir4

python $SCOMATIC/scripts/BaseCellCalling/BaseCellCalling.step1.py \
          --infile ${output_dir3}/${run}.BaseCellCounts.AllCellTypes.tsv \
          --outfile ${output_dir4}/${run} \
      --max_cell_types 1 \
      --min_cell_types 2 \
          --ref $REF


# Step 4.2
log_message "Step 4.2: Somatic mutation detection..."
editing=$SCOMATIC/RNAediting/AllEditingSites.mm10.txt
PON=$SCOMATIC/PoNs/PoN.scRNAseq.mm10.tsv

python $SCOMATIC/scripts/BaseCellCalling/BaseCellCalling.step2.py \
          --infile ${output_dir4}/${run}.calling.step1.tsv \
          --outfile ${output_dir4}/${run}.calling.step2.tsv \
```

```
            --editing $editing \
            --pon $PON

  # extra step: Intersection with bed file
  log_message "Extra step: Intersection with bed file..."
  bedtools intersect -header -a ${output_dir4}/${run}.calling.step2.tsv -b $SCOMATIC/bed_file

  log_message "Finished scomatic pipeline for ${run} successfully!"

done

log_message "Pipeline completed successfully for both runs!"

# Display log file
echo "Log file saved to: $logfile"
cat "$logfile"
```

Execute:

```
# [matterhorn]
./SComatic_mouse.sh
```

10.2 4.2. SingleCellGenotype calling to obtain mutated cell barcodes

```
#!/bin/bash
# SComatic extra functionality: Computing the genotype for each cell at the variant sites
# 01/10/2024
# Author: Alba Méndez Alejandre
# This is going to allow us to map each variant in the UMAP

SCOMATIC=~/bin/SComatic-main

run_names=("run1" "run1") # list of runs

REF=/media/storage/reference_genomes/Mus_musculus.GRCm38.dna.primary_assembly.fa
meta_dir=/media/storage/mcGinn_2021/scomatic/markers
```

```
i=0
for run in "${run_names[@]}";
do
    i=$((i + 1))

    output_dir="/media/storage/mcGinn_2021/scomatic/output/${run}"
    output_dir1=$output_dir/Step1_BamCellTypes
    output_dir4=$output_dir/Step4_VariantCalling

    STEP4_2_pass=${output_dir4}/${run}_modif.calling.step2.pass.tsv # modified chr1 to 1 tsv

    output_dir7=$output_dir/SingleCellAlleles
    mkdir -p $output_dir7


    for bam in $(ls -d $output_dir1/*bam);do
            cell_type=$(basename $bam | awk -F'.' '{print $(NF-1)}')

        temp=$output_dir7/temp_${cell_type}
            mkdir -p $temp

        meta_file="$meta_dir/${run}/esoph_markers_scomatic_${run}_${cell_type}.tsv"

            python $SCOMATIC/scripts/SingleCellGenotype/SingleCellGenotype.py --bam $bam  \
                --infile ${STEP4_2_pass}   \
                --nprocs 10   \
                --meta $meta_file   \
                --outfile ${output_dir7}/${run}_${cell_type}.single_cell_genotype.tsv \
                --tmp_dir $temp  \
                --ref $REF \
            --alt_flag All

        echo "bam file is: $bam, Infile is: ${STEP4_2_pass}, meta file is $meta_file and outf

            rm -rf $temp
        done
done
```

```
# [matterhorn]
cd /home/alba/scripts/scomatic

./variants_mapping.sh
```

10.3 4.3. Discard variants falling in repetitive regions

Creation of bed file with regions overlapping with repetitive ones, in order to remove variants falling in repetitive regions.

```
bedtools intersect -v -a mm10_genes.bed -b mm10_RepeatMasker.bed > UCSC.m10.without.repeatmas
```

10.3.1 Filter SComatic output to discard variants falling in repetitive regions

```
conda activate SComatic
SCOMATIC=~/bin/SComatic-main
output_dir4=/media/storage/mcGinn_2021/scomatic/output/run1/Step4_VariantCalling

bedtools intersect -header -a ${output_dir4}/run1.calling.step2.tsv -b $SCOMATIC/bed_files_o

output_dir4=/media/storage/mcGinn_2021/scomatic/output/run2/Step4_VariantCalling

bedtools intersect -header -a ${output_dir4}/run2.calling.step2.tsv -b $SCOMATIC/bed_files_o
```

> **Note:**
>
> *calling.step2.pass.tsv files contain only FILTER = PASS variants that don't fall in repetitive regions

10.4 4.4. Filtering of TSVs

10.4.1 Filter the TSVs with only PASS in FILTER column

```
seq_ids=("run1" "run2")
wd="/media/storage/mcGinn_2021/scomatic/output/"

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

    awk 'BEGIN {FS="\t"; OFS="\t"} NR <= 29 || ($6 == "PASS" && !/^#/) {print}' "${wd}/${seq

done
```

> **Note:**
>
> *_filtered.calling.step2.tsv files contain only FILTER = PASS variants, including variants that fall in repetitive regions.

10.4.2 Filter the TSVs with PASS or Multiple_cell_types in FILTER column

In order to analyze the variants that are arked solely as "Multiple_cell_types", we obtain them via an awk command.

```
seq_ids=("run1" "run2")
wd="/media/storage/mcGinn_2021/scomatic/output/"

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

    awk 'BEGIN {FS="\t"; OFS="\t"} NR <= 29 || ($6 == "PASS" || $6 == "Multiple_cell_types")
done
```

> **Note:**
>
> *_mult.calling.step2.tsv files contain only FILTER = PASS|Multiple_cell_types variants, including variants that fall in repetitive regions.

```
# [matterhorn]
conda activate SComatic
SCOMATIC=~/bin/SComatic-main
output_dir4=/media/storage/mcGinn_2021/scomatic/output/run1/Step4_VariantCalling

# for multiple_cell type
bedtools intersect -header -a ${output_dir4}/run1_mult.calling.step2.tsv -b $SCOMATIC/bed_fi

output_dir4=/media/storage/mcGinn_2021/scomatic/output/run2/Step4_VariantCalling

# for multiple_cell type
bedtools intersect -header -a ${output_dir4}/run2_mult.calling.step2.tsv -b $SCOMATIC/bed_fi
```

> **Note:**
>
> *_mult.calling.step2.pass.tsv files contain only FILTER = PASS|Multiple_cell_types variants that don't fall in repetitive regions