Myra spns snRNASeq May 2023

10/31/23

Table of contents

1 Clear memory and set working directory

```
rm(list = ls()) # clear the workspace
gc() # clear the memory
setwd("~/Myra_spns_snRNAseq_2023/") # set the working directory
getwd() # check the working directory
```

```
options(future.globals.maxSize = 40 * 1024 ^ 3) # for 20 Gb RAM # set the maximum size of the global environment for
options(future.seed=TRUE) # set seed for reproducibility
```

2 Load libraries

```
# load the libraries
library(tidyverse)
library(Seurat)
library(scDblFinder)
library(patchwork)
library(SingleCellExperiment)
library(harmony)
library(BiocParallel)
library(repr)
library(Libra)
library(clusterProfiler)
library(dittoSeq)
library(ggrepel)
```

```
library(dittoSeq)
library(paletteer)
library(ggpattern)
library(future)
library(liana)
library(viridis)
library(enrichR)
library(ggsankey)
library(paletteer)
options(repr.plot.width=16, repr.plot.height=10) # change the size of the plots
RNGkind("L'Ecuyer-CMRG") # set the random number generator
set.seed(1) # set the seed for reproducibility
bp <- MulticoreParam( RNGseed=1234) # set the seed for reproducibility for BiocParallel

plan("multicore", workers = 4) # set the number of cores for parallel computing

options(ggrepel.max.overlaps = 20) # set the maximum number of overlaps for ggrepel
```

3 Load data

3.1 sib_1

```
# load the data for the first sample
sib_1_seurat<- Read10X('Cellranger_outs/sib_1_cellranger/outs/filtered_feature_bc_matrix/')
sib_1_seurat <- CreateSeuratObject(sib_1_seurat,project = "sib_1")
sib_1_seurat
```

```
# Calculate mitochondrial percentage
sib_1_seurat[["percent.mt"]] <- PercentageFeatureSet(sib_1_seurat, pattern = "^mt-")
```

```
sib_1_seurat@meta.data %>% head
```

```
median(sib_1_seurat@meta.data$percent.mt)
median(sib_1_seurat@meta.data$nCount_RNA)
median(sib_1_seurat@meta.data$nFeature_RNA)
dim(sib_1_seurat@meta.data)
```

```
# plot the relation of the percentage mitochondria and the number of UMIs for the first sample
plot1 <- FeatureScatter(sib_1_seurat, feature1 = "nCount_RNA", feature2 = "percent.mt")
# plot the percentage of relation of the number of genes and the number of UMIs for the first sample
plot2 <- FeatureScatter(sib_1_seurat, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

options(repr.plot.width=12, repr.plot.height=12)
plot1 + plot2
# plot the percentage of mitochondrial genes, number of genes and the number of UMIs for the first sample
VlnPlot(sib_1_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```

```
# filter the cells for the first sample based on the number of genes, the number of UMIs, the percentage of mitochond
sib_1_seurat <- subset(sib_1_seurat, subset = nFeature_RNA > 200 & nFeature_RNA < 5000 &
                    percent.mt < 20 & nCount_RNA <10000)
sib_1_seurat
```

```
# remove the mitochondrial genes
counts <- GetAssayData(sib_1_seurat, assay = "RNA")
non_mt_genes <- rownames(counts)[!grepl("^mt-", rownames(counts))]
sib_1_seurat <- subset(sib_1_seurat, features = non_mt_genes)
sib_1_seurat
sib_1_seurat[["percent.mt_after_removal"]] <- PercentageFeatureSet(sib_1_seurat, pattern = "^mt-") # calculate the pe
VlnPlot(sib_1_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt", "percent.mt_after_removal"), ncol = 3)
```

3.1.1 Calculate the doublet cells using scDblFinder

```
# normalize the data for the first sample using the SCTransform method
sib_1_seurat <- SCTransform(sib_1_seurat, verbose = FALSE,vars.to.regress = "percent.mt") %>%
    RunPCA(npcs = 50, verbose = FALSE)
```

```
# convert the Seurat object to SingleCellExperiment object to be used in the scDblFinder package
sce_sib_1 <- as.SingleCellExperiment(sib_1_seurat, slot="counts", reducedDims = "pca")
sce_sib_1
```

```
# convert the Seurat object to SingleCellExperiment object to be used in the scDblFinder package
sce_sib_1 <- scDblFinder(sce_sib_1, BPPARAM = bp)
```

```
# sce_sib_1 %>% str
sce_sib_1$scDblFinder.score %>% head
```

```
# port the resulting scores back to the Seurat object:
sib_1_seurat$scDblFinder.score <- sce_sib_1$scDblFinder.score
sib_1_seurat$scDblFinder.class <- sce_sib_1$scDblFinder.class
sib_1_seurat@meta.data %>% head
```

```
sib_1_seurat <- subset(sib_1_seurat, subset = scDblFinder.class == "singlet")
sib_1_seurat
```

3.2 Sib_2

```
# load the data for the first sample
sib_2_seurat<- Read10X('Cellranger_outs/sib_2_cellranger/outs/filtered_feature_bc_matrix/')
sib_2_seurat <- CreateSeuratObject(sib_2_seurat,project = "sib_2")
sib_2_seurat
sib_2_seurat[["percent.mt"]] <- PercentageFeatureSet(sib_2_seurat, pattern = "^mt-")
```

```
median(sib_2_seurat@meta.data$percent.mt)
median(sib_2_seurat@meta.data$nCount_RNA)
median(sib_2_seurat@meta.data$nFeature_RNA)
dim(sib_2_seurat@meta.data)
```

```
# plot the relation of the percentage mitochondria and the number of UMIs for the first sample
plot1 <- FeatureScatter(sib_2_seurat, feature1 = "nCount_RNA", feature2 = "percent.mt")
# plot the percentage of relation of the number of genes and the number of UMIs for the first sample
plot2 <- FeatureScatter(sib_2_seurat, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

options(repr.plot.width=12, repr.plot.height=12)
plot1 + plot2
# plot the percentage of mitochondrial genes, number of genes and the number of UMIs for the first sample
VlnPlot(sib_2_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```

```
# filter the cells for the first sample based on the number of genes, the number of UMIs, the percentage of mitochond
sib_2_seurat <- subset(sib_2_seurat, subset = nFeature_RNA > 200 & nFeature_RNA < 5000 &
                    percent.mt < 20 & nCount_RNA <10000)
sib_2_seurat
```

```
counts <- GetAssayData(sib_2_seurat, assay = "RNA")
non_mt_genes <- rownames(counts)[!grepl("^mt-", rownames(counts))]
```

```
sib_2_seurat <- subset(sib_2_seurat, features = non_mt_genes)
sib_2_seurat
sib_2_seurat[["percent.mt_after_removal"]] <- PercentageFeatureSet(sib_2_seurat, pattern = "^mt-") # calculate the pe
VlnPlot(sib_2_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt", "percent.mt_after_removal"), ncol = 3)
```

```
sib_2_seurat <- SCTransform(sib_2_seurat, verbose = FALSE,vars.to.regress = "percent.mt") %>%
    RunPCA(npcs = 50, verbose = FALSE)
```

3.2.1 Calculate the doublet cells using scDblFinder

```
sce_sib_2 <- as.SingleCellExperiment(sib_2_seurat, slot="counts", reducedDims = "pca")
sce_sib_2
```

```
sce_sib_2 <- scDblFinder(sce_sib_2, BPPARAM = bp)
# port the resulting scores back to the Seurat object:
sib_2_seurat$scDblFinder.score <- sce_sib_2$scDblFinder.score
sib_2_seurat$scDblFinder.class <- sce_sib_2$scDblFinder.class
sib_2_seurat@meta.data %>% head
```

```
sib_2_seurat <- subset(sib_2_seurat, subset = scDblFinder.class == "singlet")
sib_2_seurat
```

3.3 mut_1

```
# load the data for the first sample
mut_1_seurat<- Read10X('Cellranger_outs/mut_1_cellranger/outs/filtered_feature_bc_matrix/')
mut_1_seurat <- CreateSeuratObject(mut_1_seurat,project = "mut_1")
mut_1_seurat
```

```r
mut_1_seurat[["percent.mt"]] <- PercentageFeatureSet(mut_1_seurat, pattern = "^mt-")
```

```r
median(mut_1_seurat@meta.data$percent.mt)
median(mut_1_seurat@meta.data$nCount_RNA)
median(mut_1_seurat@meta.data$nFeature_RNA)
dim(mut_1_seurat@meta.data)
```

```r
# plot the relation of the percentage mitochondria and the number of UMIs for the first sample
plot1 <- FeatureScatter(mut_1_seurat, feature1 = "nCount_RNA", feature2 = "percent.mt")
# plot the percentage of relation of the number of genes and the number of UMIs for the first sample
plot2 <- FeatureScatter(mut_1_seurat, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

options(repr.plot.width=12, repr.plot.height=12)
plot1 + plot2
# plot the percentage of mitochondrial genes, number of genes and the number of UMIs for the first sample
VlnPlot(mut_1_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```

```r
# filter the cells for the first sample based on the number of genes, the number of UMIs, the percentage of mitochond
mut_1_seurat <- subset(mut_1_seurat, subset = nFeature_RNA > 200 & nFeature_RNA < 5000 &
                    percent.mt < 20 & nCount_RNA <10000)
mut_1_seurat
```

```r
counts <- GetAssayData(mut_1_seurat, assay = "RNA")
non_mt_genes <- rownames(counts)[!grepl("^mt-", rownames(counts))]
mut_1_seurat <- subset(mut_1_seurat, features = non_mt_genes)
mut_1_seurat
mut_1_seurat[["percent.mt_after_removal"]] <- PercentageFeatureSet(mut_1_seurat, pattern = "^mt-") # calculate the pe
VlnPlot(mut_1_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt", "percent.mt_after_removal"), ncol = 3)
```

```
mut_1_seurat <- SCTransform(mut_1_seurat, verbose = FALSE,vars.to.regress = "percent.mt") %>%
    RunPCA(npcs = 50, verbose = FALSE)
```

3.3.1 Calculate the doublet cells using scDblFinder

```
sce_mut_1 <- as.SingleCellExperiment(mut_1_seurat, slot="counts", reducedDims = "pca")
sce_mut_1
```

```
sce_mut_1 <- scDblFinder(sce_mut_1, BPPARAM = bp)
```

```
# port the resulting scores back to the Seurat object:
mut_1_seurat$scDblFinder.score <- sce_mut_1$scDblFinder.score
mut_1_seurat$scDblFinder.class <- sce_mut_1$scDblFinder.class
mut_1_seurat@meta.data %>% head
```

```
mut_1_seurat <- subset(mut_1_seurat, subset = scDblFinder.class == "singlet")
mut_1_seurat
```

3.4 mut_2

```
# load the data for the first sample
mut_2_seurat<- Read10X('Cellranger_outs/mut_2_cellranger/outs/filtered_feature_bc_matrix/')
mut_2_seurat <- CreateSeuratObject(mut_2_seurat,project = "mut_2")
mut_2_seurat
mut_2_seurat[["percent.mt"]] <- PercentageFeatureSet(mut_2_seurat, pattern = "^mt-")
```

```r
median(mut_2_seurat@meta.data$percent.mt)
median(mut_2_seurat@meta.data$nCount_RNA)
median(mut_2_seurat@meta.data$nFeature_RNA)
dim(mut_2_seurat@meta.data)
```

```r
# plot the relation of the percentage mitochondria and the number of UMIs for the first sample
plot1 <- FeatureScatter(mut_2_seurat, feature1 = "nCount_RNA", feature2 = "percent.mt")
# plot the percentage of relation of the number of genes and the number of UMIs for the first sample
plot2 <- FeatureScatter(mut_2_seurat, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

options(repr.plot.width=12, repr.plot.height=12)
plot1 + plot2
# plot the percentage of mitochondrial genes, number of genes and the number of UMIs for the first sample
VlnPlot(mut_2_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```

```r
# filter the cells for the first sample based on the number of genes, the number of UMIs, the percentage of mitochond
mut_2_seurat <- subset(mut_2_seurat, subset = nFeature_RNA > 200 & nFeature_RNA < 5000 &
                    percent.mt < 20 & nCount_RNA <10000)
mut_2_seurat
```

```r
counts <- GetAssayData(mut_2_seurat, assay = "RNA")
non_mt_genes <- rownames(counts)[!grepl("^mt-", rownames(counts))]
mut_2_seurat <- subset(mut_2_seurat, features = non_mt_genes)
mut_2_seurat
mut_2_seurat[["percent.mt_after_removal"]] <- PercentageFeatureSet(mut_2_seurat, pattern = "^mt-") # calculate the pe
VlnPlot(mut_2_seurat, features = c("nFeature_RNA", "nCount_RNA", "percent.mt", "percent.mt_after_removal"), ncol = 3)
```

```r
mut_2_seurat <- SCTransform(mut_2_seurat, verbose = FALSE,vars.to.regress = "percent.mt") %>%
    RunPCA(npcs = 50, verbose = FALSE)
```

3.4.1 Calculate the doublet cells using scDblFinder

```
sce_mut_2 <- as.SingleCellExperiment(mut_2_seurat, slot="counts", reducedDims = "pca")
sce_mut_2
```

```
sce_mut_2 <- scDblFinder(sce_mut_2, BPPARAM = bp)
```

```
# port the resulting scores back to the Seurat object:
mut_2_seurat$scDblFinder.score <- sce_mut_2$scDblFinder.score
mut_2_seurat$scDblFinder.class <- sce_mut_2$scDblFinder.class
mut_2_seurat@meta.data %>% head
```

```
mut_2_seurat <- subset(mut_2_seurat, subset = scDblFinder.class == "singlet")
mut_2_seurat
```

4 Batch correction using Seurat

4.1 combine all datasets

According to benchmarking- LIGER, HARMONY and Seurat are all good. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1850-9

```
all_samples_list <- list(sib_1_seurat, sib_2_seurat, mut_1_seurat, mut_2_seurat) # create a list of all samples
all_samples_list
```

```
names(all_samples_list) <- c("sib_1", "sib_2", "mut_1", "mut_2") # name the samples
all_samples_list
```

```r
all_samples_features <- SelectIntegrationFeatures(object.list = all_samples_list, nfeatures = 3000) # select the feat
```

```r
all_samples_list <- PrepSCTIntegration(object.list = all_samples_list, anchor.features = all_samples_features) # prep
```

```r
# Integrate the samples according to seurat pipeline
cell_anchors <- FindIntegrationAnchors(object.list = all_samples_list, normalization.method = "SCT",
    anchor.features = all_samples_features)
all_samples_list_sct <- IntegrateData(anchorset = cell_anchors, normalization.method = "SCT")
```

```r
combined_sct <- RunPCA(all_samples_list_sct, verbose = FALSE, npcs = 50) # calculate the PCA for the integrated sampl
```

```r
print(combined_sct[["pca"]], dims = 1:5, nfeatures = 5) # print the first 5 PCs
```

```r
DimPlot(combined_sct, reduction = "pca") # print the PCA plot
```

```r
ElbowPlot(combined_sct,ndims = 50) # print the elbow plot
```

4.2 Clustering

```r
combined_sct <- RunUMAP(combined_sct, reduction = "pca", dims = 1:30, verbose = FALSE) # calculate the UMAP for the i
combined_sct <- FindNeighbors(combined_sct, reduction = "pca", dims = 1:30) # find the neighbors for the integrated s
combined_sct <- FindClusters(combined_sct, resolution = c(0.1,0.15,0.2, 0.4, 0.6, 0.8, 1, 1.2), verbose = TRUE) # fin
```

4.3 UMAP with no annotation and showing normalizaton across datasets

```
DimPlot(combined_sct, reduction = "umap", group.by = "orig.ident", label = TRUE, repel = TRUE) # print the UMAP plot
```

```
# check all resolutions for the integrated samples
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.0.15", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.0.1", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.0.2", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.0.4", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.0.6", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.0.8", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.1", label = TRUE, repel = TRUE)
DimPlot(combined_sct, reduction = "umap", group.by = "integrated_snn_res.1.2", label = TRUE, repel = TRUE)
```

```
Idents(combined_sct) <- combined_sct@meta.data$integrated_snn_res.0.4 # set the resolution to 0.4
```

```
DimPlot(combined_sct, reduction = "umap", label = TRUE, repel = TRUE) # print the UMAP plot for the resolution 0.4
```

4.4 Find Cluster Markers

```
all_markers <- FindAllMarkers(combined_sct, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.2) # find the marker
```

```
# check the markers for all clusters
all_markers %>% group_by(cluster) %>% slice_max(n = 5, order_by = avg_log2FC) %>% ungroup() %>% arrange(cluster, avg_

# make a directory called results_seurat
dir.create("results_seurat")
# write the results to a csv file
all_markers %>% group_by(cluster) %>% slice_max(n = Inf, order_by = avg_log2FC) %>% ungroup() %>% arrange(cluster, de
```

4.5 Convert to Mouse Gene Symbols

```
# Basic function to convert zebrafish to mouse gene names

zgGenes <- sib_1_seurat@assays$RNA %>% rownames() %>% unique() # get the zebrafish gene names
# This function uses biomaRt to convert zebrafish gene names to mouse gene names.
# This is done by using the zebrafish and mouse ensembl mart datasets.
# The function takes a vector of zebrafish gene names as input and returns
# a data frame of zebrafish and mouse gene names.
convertDanioGeneList_Mouse <- function(x){
  require("biomaRt") # load biomaRt package
  mouse = useMart("ensembl", dataset = "mmusculus_gene_ensembl", host = "https://dec2021.archive.ensembl.org/") # use
  danio = useMart("ensembl", dataset = "drerio_gene_ensembl",host = "https://dec2021.archive.ensembl.org/") # use zeb

  genesV2 = getLDS(attributes = c("ensembl_gene_id", "zfin_id_symbol"),
                   filters = "zfin_id_symbol", # get zebrafish gene names
                   values = x , # use the zebrafish gene names
                   mart = danio,  # use the zebrafish mart
                   attributesL = c("mgi_symbol", "ensembl_gene_id", "description"), # get mouse gene names
                   martL = mouse, uniqueRows=T) # use the mouse mart

  colnames(genesV2)[colnames(genesV2)== "Gene.stable.ID"] <- "EnsmblID_Zebrafish" # rename columns
  colnames(genesV2)[colnames(genesV2)== "Gene.stable.ID.1"] <- "EnsmblID_Mouse" # rename columns

  # Check if the gene is not found
  if (length(genesV2) == 0) {
    print("No gene found for this input")
  } else {
    return(genesV2) # return the genes
  }
}
```

```r
# Run the function
Mouse_Genes <- convertDanioGeneList_Mouse(zgGenes)
# print the first 6 genes
head(Mouse_Genes)
```

```r
all_markers_mouse <- merge(all_markers, Mouse_Genes, by.x = "gene", by.y = "ZFIN.symbol", all.x = TRUE) # merge the m
dim(all_markers_mouse)
dim(all_markers)
head(all_markers_mouse)
```

```r
# arrange all_markers_mouse by cluster and log2FC
all_markers_mouse <- all_markers_mouse %>% group_by(cluster) %>% slice_max(n = Inf, order_by = avg_log2FC) %>% ungrou
head(all_markers_mouse)
all_markers_mouse %>% write.csv("results_seurat/all_markers_mouse_mt_removed.csv")
```

4.6 Assign cluster labels

```r
DimPlot(combined_sct, reduction = "umap", label = TRUE, repel = TRUE, label.size= 8)
```

```r
## markers from Junker's Fibroblast paper: https://www.nature.com/articles/s41588-022-01129-5
FB <- c("col1a2", "mdka", "col1a1a", "col1a1b", "col5a1", "dpt", "fn1b", "ccl25b", "sparc", "clu", "dcn", "mfap5", "p
## markers for Valve Fibroblasts
VF <- c("zgc:153704", "abi3bpb", "krt4", "angptl7", "igfbp5b", "cyp26b1", "tnfaip6", "mgp", "rspo1", "fibinb", "aif1l
## markers for Endothelial cells (apnln)
EC_apln <- c("apnln", "hbegfb", "admb", "apln", "si:ch211-195b11.3", "sele", "itga2b", "thbs1b", "fgl2a", "FO681357.1
## markers for Endothelial cells (lyve1)
EC_lyve1 <- c("cxcl12a", "CU929150.1", "thy1", "lyve1a", "selenop", "cdh6", "si:dkey-203a12.9", "lyve1b", "id1", "si:
## markers for Endothelial cells (plvapb)
EC_plvapb <- c("wu:fj16a03", "cxcl12b", "rbp2a", "plvapb", "cldn5b", "rgcc", "fabp11a", "id1", "tmsb1", "ldb2a", "tci
```

```
## markers for Cardiomyocytes (dediff.)
CM_dediff <- c("nppa", "nppb", "hsp90aa1.1", "ttn.2", "ttn.1", "myh6", "atp2a2a", "xirp1", "si:ch211-131k2.3", "hspb1
## markers for Cardiomyocytes (Atrium)
CM_atrium <- c("tnnc1b", "myh6", "si:ch211-270g19.5", "tcap", "tnni1b", "aldoab", "tnnt2a", "smtnl1", "cmlc1", "gapdh
## markers for Cardiomyocytes (Ventricle)
CM_ventricle <- c("tnni4a", "CR926459.1", "fabp3", "myl7", "ak1", "cmlc1", "actc1a", "acta1b", "myh7l", "ndufa4", "ga
## markers for Endocardium (Atrium)
encar_endo_atrium <- c("zgc:158343", "spock3", "im:7152348", "ptgs2a", "ptgs2b", "id2b", "vcam1b", "aqp8a.1", "mycb",
## markers for Endocardium (frzb)
encar_frzb <- c("vwf", "si:ch211-153b23.5", "c2cd4a", "edn2", "cfd", "frzb", "ecscr", "glulb", "efemp2b", "calm1a", "
## markers for Endocardium (Ventricle)
encar_endo_ventricle <- c("si:ch73-86n18.1", "aqp8a.1", "spock3", "mb", "fabp11a", "epas1b", "ramp2", "si:ch211-145b1
## markers for Epicardium (Atrium)
EPIC_atrium <- c("gstm.3", "s100a10a", "fn1b", "krt15", "mmp2", "si:ch211-105c13.3", "krt4", "frzb", "mmel1", "anxa2a
## markers for Epicardium (Ventricle)
EPIC_ventricle <- c("si:ch211-106h4.12", "gstm.3", "krt15", "postnb", "s100a10a", "fn1b", "endouc", "mmp2", "tfa", "k
## markers for Macrophages
MACROPHAGES <- c("grn1", "grn2", "ccl35.1", "cd74a", "c1qb", "ctsd", "c1qc", "lygl1", "lgals3bpb", "si:ch211-147m6.2"
## markers for Monocytes
MONOCYTES <- c("ccl35.1", "epdl1", "si:ch211-214p16.1", "si:busm1-266f07.2", "si:ch211-214p16.2", "ccl35.2", "CU45909
## markers for Myelin cells
MYELIN_CELLS <- c("scn4ab", "cd59", "mbpa", "mpz", "mbpb", "plp1b", "BX936284.1", "anxa13l", "apoeb", "si:rp71-19m20.
## markers for Neuronal cells
NEURONAL_CELLS <- c("vipb", "elavl4", "stmn1b", "syt1a", "tuba1c", "gap43", "snap25a", "stmn2a", "sncga", "rtn1b", "m
## markers for Neutrophils
NEUTROPHILS <- c("lyz", "lect2l", "BX908782.2", "npsn", "si:ch211-117m20.5", "mmp13a.1", "si:ch211-9d9.1", "scpp8", "
## markers for Perivascular cells
PERIVASCULAR_CELLS <- c("TCIM", "cxcl12b", "pdgfrb", "rasl12", "BX901920.1", "kcne4", "rgs5b", "rgs4", "agtr2", "TPM1
## markers for Proliferating cells
PROLIFERATING_CELLS <- c("pcna", "stmn1a", "hmgb2b", "DUT", "sumo3b", "rrm2.1", "si:ch211-288g17.3", "rpa3", "banf1",
## markers for Smooth muscle cells
SMOOTH_MUSCLE_CELLS <- c("rgs5a", "acta2", "TPM1", "C11orf96", "tagln", "myh11a", "krt91", "itih1", "myl6", "myl9a",
```

```
## markers for T-cells
T_CELLS <- c("si:ch211-214p16.1", "ccl34b.4", "ccl36.1", "ccl38.6", "DNAJA4", "ccr9a", "CR936442.1", "cxcr4b", "hspbp
```

```
# render the plot to be bigger
options(repr.plot.width=18, repr.plot.height=32)
# make a heatmap usingd DoHeatmap using the markers mentioned above
DoHeatmap(object = combined_sct, features = c(FB, VF, EC_apln, EC_lyve1, EC_plvapb, CM_dediff, CM_atrium, CM_ventricl
options(repr.plot.width=12, repr.plot.height=12)
```

```
options(repr.plot.width=12, repr.plot.height=12)
# Valve genes accodring to: https://www.nature.com/articles/s42003-021-02571-7--  by myra

valve_markers_nat <- unique(c("fgfr2","tgfb2", "fn1b", "wt1a", "fgfr4", "smad6b", "fgfr3", "gata5", "bmpr2b", "piezo2
# Epicardial markers according to : https://www.sciencedirect.com/science/article/pii/S1534580720300551
epicardial_markers <- unique(c("fstl1a", "fstl1b", "cav1", "aldh1a2", "tcf21","tgm2b", "sema3fb","tbx18", "adma", "ja
VlnPlot(object = combined_sct, features = valve_markers_nat, stack = TRUE, group.by = "integrated_snn_res.0.4") + NoL
VlnPlot(object = combined_sct, features = epicardial_markers, stack = TRUE, group.by = "integrated_snn_res.0.4") + No
```

```
genes_to_annotate <- c("myh6", "cdh5", "pecam1", "pdgfra", "mslnb", "myh11a", "kcnj8", "adgre1", "itgal", "naaa", "s1
VlnPlot(object = combined_sct, features = genes_to_annotate, stack = TRUE, group.by = "integrated_snn_res.0.4") + NoL
```

```
# check the number of cells
table(combined_sct@meta.data$integrated_snn_res.0.4, combined_sct@meta.data$orig.ident)
```

```
# cluster   name
# 0 CM_1
# 1 Neurons
# 2 CM_2
# 3 EC_Endocardium
# 4 Erythrocytes
```

```
# 5 CM_3
# 6 FB_1
# 7 FB_2
# 8 epicardium
# 9 Immune cells
# 10 Neurons?
# 11 Skeletal muscle
# 12 unknown_2
# 13 epithelial cells
```

```
# allocate cell types to the clusters
combined_sct@meta.data$cell_type <- NULL
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 0] <- "CM_1"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 1] <- "Neurons"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 2] <- "CM_2"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 3] <- "EC_Endocardium"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 4] <- "Erythrocytes"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 5] <- "CM_3"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 6] <- "FB_1"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 7] <- "FB_2"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 8] <- "epicardium"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 9] <- "Immune cells"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 10] <- "Neurons?"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 11] <- "Skeletal muscle"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 12] <- "unknown_2"
combined_sct@meta.data$cell_type[combined_sct@meta.data$integrated_snn_res.0.4 == 13] <- "epithelial cells"
```

```
Idents(combined_sct) <- combined_sct@meta.data$cell_type
```

```r
# change the plot size
options(repr.plot.width=12, repr.plot.height=12)
DimPlot(combined_sct, reduction = "umap", label = TRUE, label.size = 6, repel = TRUE, pt.size = 2)
```

5 check gene markers

```r
# read marker genes from Burkhard et al. 2018-- https://elifesciences.org/articles/31515
# read sheet Ventricle
Burkhard_ventricle <- openxlsx::read.xlsx("Burkhard_et_al_fig4.xlsx", sheet="Ventricle")
# make first row as column names
colnames(Burkhard_ventricle) <- Burkhard_ventricle[1,]
# remove first row
Burkhard_ventricle <- Burkhard_ventricle[-1,]

# convert to numeric
Burkhard_ventricle$`PValue` <- as.numeric(Burkhard_ventricle$`PValue`)
Burkhard_ventricle$`logFC` <- as.numeric(Burkhard_ventricle$`logFC`)

Burkhard_ventricle %>% head

# count how many genes are significant
sum(Burkhard_ventricle$`PValue` < 0.05)
```

```r
# read marker genes from Burkhard et al. 2018
# read sheet Atrium
Burkhard_atrium <- openxlsx::read.xlsx("Burkhard_et_al_fig4.xlsx", sheet="Atrium")
# make first row as column names
colnames(Burkhard_atrium) <- Burkhard_atrium[1,]
# remove first row
```

```r
Burkhard_atrium <- Burkhard_atrium[-1,]

# convert to numeric
Burkhard_atrium$`PValue` <- as.numeric(Burkhard_atrium$`PValue`)
Burkhard_atrium$`logFC` <- as.numeric(Burkhard_atrium$`logFC`)

Burkhard_atrium %>% head

# count how many genes are significant
sum(Burkhard_atrium$`PValue` < 0.05)
```

```r
# read marker genes from Burkhard et al. 2018
# read sheet AV canal
Burkhard_AV_canal <- openxlsx::read.xlsx("Burkhard_et_al_fig4.xlsx", sheet="AV canal")
# make first row as column names
colnames(Burkhard_AV_canal) <- Burkhard_AV_canal[1,]
# remove first row
Burkhard_AV_canal <- Burkhard_AV_canal[-1,]

# convert to numeric
Burkhard_AV_canal$`PValue` <- as.numeric(Burkhard_AV_canal$`PValue`)
Burkhard_AV_canal$`logFC` <- as.numeric(Burkhard_AV_canal$`logFC`)

Burkhard_AV_canal %>% head

# count how many genes are significant
sum(Burkhard_AV_canal$`PValue` < 0.05)
```

```r
# read marker genes from Burkhard et al. 2018
# read sheet SA region
Burkhard_SA_region <- openxlsx::read.xlsx("Burkhard_et_al_fig4.xlsx", sheet="SA region")
# make first row as column names
```

```r
colnames(Burkhard_SA_region) <- Burkhard_SA_region[1,]
# remove first row
Burkhard_SA_region <- Burkhard_SA_region[-1,]

# convert to numeric
Burkhard_SA_region$`PValue` <- as.numeric(Burkhard_SA_region$`PValue`)
Burkhard_SA_region$`logFC` <- as.numeric(Burkhard_SA_region$`logFC`)

Burkhard_SA_region %>% head

# count how many genes are significant
sum(Burkhard_SA_region$`PValue` < 0.05)
```

```r
#get genes which are significant in all 4 regions and convert them to vector
Burkhard_ventricle %>%
  filter(`PValue` < 0.05) %>%
  dplyr::pull(`Gene ID`) %>%
  as.character() -> ventricle_genes
Burkhard_atrium %>% filter(`PValue` < 0.05) %>% dplyr::pull(`Gene ID`)  %>% as.character() -> atrium_genes
Burkhard_AV_canal %>% filter(`PValue` < 0.05) %>% dplyr::pull(`Gene ID`)  %>% as.character() -> AV_canal_genes
Burkhard_SA_region %>% filter(`PValue` < 0.05) %>% dplyr::pull(`Gene ID`)  %>% as.character() -> SA_region_genes
ventricle_genes %>% head
```

```r
# render the plot to be bigger
options(repr.plot.width=18, repr.plot.height=32)
# make a heatmap usingd DoHeatmap using the markers mentioned above
DoHeatmap(object = combined_sct, features = c(ventricle_genes, atrium_genes,AV_canal_genes, SA_region_genes ), group.
DoHeatmap(object = combined_sct, features = c(ventricle_genes, atrium_genes,AV_canal_genes, SA_region_genes ))
options(repr.plot.width=12, repr.plot.height=12)
```

```
# CM_1 Ventricle
# CM_2 Atrium
```

```
DotPlot(object = combined_sct, features = c("has2", "bmp4", "tbx2b", "nppa", "vmhc", "myh6", "myh7", "tbx5", "vcana")
DoHeatmap(object = combined_sct, features = c("has2", "bmp4", "tbx2b", "nppa", "vmhc", "myh6", "myh7", "tbx5", "vcana
VlnPlot(object = combined_sct, features = c("has2", "bmp4", "tbx2b", "nppa", "vmhc", "myh6", "myh7", "tbx5", "vcana")
VlnPlot(object = combined_sct, features = c("has2", "bmp4", "tbx2b", "nppa", "vmhc", "myh6", "myh7", "tbx5", "vcana")
```

```
DotPlot(object = combined_sct, features = c("has2", "bmp4", "tbx2b", "nppa", "vmhc", "myh6", "myh7", "elnb", "isl1",
VlnPlot(object = combined_sct, features = c("has2", "bmp4", "tbx2b", "nppa", "vmhc", "myh6", "myh7","elnb", "isl1","f
```

6 Proliferative cells

```
s.genes <- cc.genes$s.genes
s.genes
g2m.genes <- cc.genes$g2m.genes
g2m.genes
```

```
# Basic function to convert human to zebrafish gene names
convertHumanGeneList <- function(x){
require("biomaRt")
human = useMart("ensembl", dataset = "hsapiens_gene_ensembl", host = "https://dec2021.archive.ensembl.org") # use the
zebrafish = useMart("ensembl", dataset = "drerio_gene_ensembl", host = "https://dec2021.archive.ensembl.org") # use t
genesV2 = getLDS(attributes = c("hgnc_symbol"), # use the hgnc_symbol as the attribute to search for
                 filters = "hgnc_symbol", # use the hgnc_symbol as the filter
                 values = x , # x is the input vector of human genes
                 mart = human, # human is the biomart object created above
                 attributesL = c("zfin_id_symbol"), # zfin_id_symbol is the zebrafish gene symbol
```

```
                martL = zebrafish, # zebrafish is the biomart object created above
                uniqueRows=T) # only return unique rows

# Print the first 6 genes found to the screen
print(head(genesV2))
return(genesV2)
}
```

```
s.genes_mouse <- convertHumanGeneList(s.genes)
g2m_mouse  <- convertHumanGeneList(g2m.genes)
```

```
# calculate cell cycle scores using the mouse genes
combined_sct <- CellCycleScoring(combined_sct, s.features = s.genes_mouse$ZFIN.symbol, g2m.features = g2m_mouse$ZFIN.
```

```
# calculate the percentage of _cycle in each cell type according to condition
cell_type_percentages_cell_cycle <- prop.table(table(combined_sct@meta.data$cell_type, combined_sct@meta.data$Phase,
cell_type_percentages_cell_cycle <- as.data.frame(cell_type_percentages_cell_cycle)
cell_type_percentages_cell_cycle %>% head
cell_type_percentages_cell_cycle <- cell_type_percentages_cell_cycle %>% dplyr::rename(cell_type = Var1, cell_cycle =
cell_type_percentages_cell_cycle %>% head
# plot these percentages using ggplot2
ggplot(cell_type_percentages_cell_cycle, aes(x = cell_type, y = Freq, fill = cell_cycle)) +
  geom_bar(stat = "identity", position = "fill") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
  labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")+facet_grid(~condition)
```

```
ggplot(cell_type_percentages_cell_cycle, aes(x = cell_type, y = Freq, fill = cell_cycle)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
  labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")+facet_grid(~condition)
```

```r
write.csv(cell_type_percentages_cell_cycle, file = "results_seurat/cell_type_percentages_cell_cycle.csv") # save the
```

7 Sub cluster cell types

```r
CM_cells  <- subset(combined_sct, subset = cell_type == "CM_1" | cell_type == "CM_2"| cell_type == "CM_3")
CM_cells
```

```r
CM_cells@meta.data %>% head
```

A data.frame: 6 × 24

	orig.ident	nCount_RNA	nFeature_RNA	percent.mt	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	cell_type	[illegible]	replicate	S.Score	G2M.Score	Phase
	<chr>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<chr>	<fct>	<fct>	<fct>	<fct>	<fct>	<chr>	<chr>	<chr>	<dbl>	<dbl>	<chr>
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	484	400	[illegible]	singlet		3	2	2	2	CM_2	sib	1	[illegible]	[illegible]	G1
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	582	492	[illegible]	singlet		9	6	4	6	CM_3	sib	1	[illegible]	-0.010335917	S
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	510	332	[illegible]	singlet		2	1	1	1	CM_1	sib	1	-0.003298153	[illegible]	G2M
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	538	390	[illegible]	singlet		9	6	4	6	CM_3	sib	1	-0.011213720	[illegible]	G2M
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	472	357	[illegible]	singlet		12	6	4	6	CM_3	sib	1	-0.006596306	[illegible]	G2M
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	422	310	[illegible]	singlet		8	1	1	1	CM_1	sib	1	[illegible]	-0.007235142	S

```r
CM_cells <- SCTransform(CM_cells, verbose = FALSE)
CM_cells <- RunPCA(CM_cells, verbose = FALSE, npcs = 50)
ElbowPlot(CM_cells,ndims = 50)
```

```r
CM_cells <- RunUMAP(CM_cells, dims = 1:50)
```

```r
CM_cells <- FindNeighbors(object = CM_cells)
CM_cells <- FindClusters(CM_cells, resolution = c(0.2, 0.4, 0.6, 0.8, 1.0, 1.2))
```

```r
DimPlot(CM_cells, reduction = "umap", group.by = "orig.ident", label = TRUE, repel = TRUE)
```

```r
# CM_cells@meta.data %>% head
```

```r
DimPlot(CM_cells, reduction = "umap", group.by = "SCT_snn_res.0.2", label = TRUE, repel = TRUE)
DimPlot(CM_cells, reduction = "umap", group.by = "SCT_snn_res.0.4", label = TRUE, repel = TRUE)
DimPlot(CM_cells, reduction = "umap", group.by = "SCT_snn_res.0.6", label = TRUE, repel = TRUE)
DimPlot(CM_cells, reduction = "umap", group.by = "SCT_snn_res.0.8", label = TRUE, repel = TRUE)
DimPlot(CM_cells, reduction = "umap", group.by = "SCT_snn_res.1", label = TRUE, repel = TRUE)
DimPlot(CM_cells, reduction = "umap", group.by = "SCT_snn_res.1.2", label = TRUE, repel = TRUE)
```

```r
AV_canal_genes_GO <-c("anxa5b", "apcdd1l", "aplnra", "aplnrb", "axin2", "bmp4", "cldn5a", "cldn5b", "cthrc1a", "cthrc
```

```r
DoHeatmap(object = CM_3_cells, features = c(AV_canal_genes_GO))
```

7.1 sub cluster EC_Endocardium

```r
EC_Endo_cells <- subset(combined_sct, subset = cell_type == "EC_Endocardium" )
EC_Endo_cells
```

```
EC_Endo_cells <- SCTransform(EC_Endo_cells, verbose = FALSE)
EC_Endo_cells <- RunPCA(EC_Endo_cells, verbose = FALSE, npcs = 50)
ElbowPlot(EC_Endo_cells,ndims = 50)
```

```
EC_Endo_cells <- RunUMAP(EC_Endo_cells, dims = 1:50)
```

```
EC_Endo_cells <- FindNeighbors(object = EC_Endo_cells)
EC_Endo_cells <- FindClusters(EC_Endo_cells, resolution = c(0.2, 0.4, 0.6, 0.8, 1.0, 1.2))
```

```
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "orig.ident", label = TRUE, repel = TRUE)
```

```
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.0.2", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.0.4", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.0.6", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.0.8", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.1", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.1.2", label = TRUE, repel = TRUE, label.size = 6)
```

```
Idents(EC_Endo_cells) <- "SCT_snn_res.0.6"
DimPlot(EC_Endo_cells, reduction = "umap", group.by = "SCT_snn_res.0.6", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(EC_Endo_cells, reduction = "umap", label = TRUE, repel = TRUE, label.size = 6)
```

```
EC_Endo_cells_markers <- FindAllMarkers(EC_Endo_cells, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
EC_Endo_cells_markers %>% head
EC_Endo_cells_markers %>% group_by(cluster) %>% top_n(n = 5, wt = avg_log2FC) %>% ungroup() %>% arrange(cluster, avg_
EC_Endo_cells_markers %>% group_by(cluster) %>% arrange(cluster, desc(avg_log2FC)) %>% write.csv(file = "results_seur
```

```
FeaturePlot(EC_Endo_cells, features = c("notch1b"), label=TRUE, reduction = "umap", label.size = 6)
VlnPlot(EC_Endo_cells, features = c("notch1b"), split.by = "condition")
```

```r
# render the plot to be bigger
# options(repr.plot.width=18, repr.plot.height=32)
# make a heatmap usingd DoHeatmap using the markers mentioned above
DoHeatmap(object = EC_Endo_cells, features = c(FB, VF, EC_apln, EC_lyve1, EC_plvapb, CM_dediff, CM_atrium, CM_ventric
# options(repr.plot.width=12, repr.plot.height=12)
DoHeatmap(object = EC_Endo_cells, features = c(VF, valve_markers_nat))
options(repr.plot.width=12, repr.plot.height=12)
```

```r
DoHeatmap(object = EC_Endo_cells, features = c(VF, valve_markers_nat, "spns1", "spns2"), group.by = "condition")
```

```r
DotPlot(object = EC_Endo_cells, features = unique(c(VF, valve_markers_nat, "spns1", "spns2")), group.by = "SCT_snn_re
# VlnPlot(object = combined_sct, features = genes_to_annotate, stack = TRUE, group.by = "integrated_snn_res.0.4") + N
```

```r
EC_Endo_cells@meta.data$EC_identity <- NULL
EC_Endo_cells@meta.data$EC_identity[EC_Endo_cells@meta.data$`SCT_snn_res.0.6`==0] <- "EC_1"
EC_Endo_cells@meta.data$EC_identity[EC_Endo_cells@meta.data$`SCT_snn_res.0.6`==1] <- "EC_2"
EC_Endo_cells@meta.data$EC_identity[EC_Endo_cells@meta.data$`SCT_snn_res.0.6`==2] <- "Valve_cells"
EC_Endo_cells@meta.data$EC_identity[EC_Endo_cells@meta.data$`SCT_snn_res.0.6`==3] <- "EC_3"
EC_Endo_cells@meta.data$EC_identity[EC_Endo_cells@meta.data$`SCT_snn_res.0.6`==4] <- "EC_4"

head(EC_Endo_cells@meta.data)

DimPlot(EC_Endo_cells, group.by = "EC_identity", label = TRUE, repel = TRUE, label.size = 6)
```

```r
table(EC_Endo_cells@meta.data$condition)
table(EC_Endo_cells@meta.data$condition, EC_Endo_cells@meta.data$`EC_identity`)
```

```r
Idents(EC_Endo_cells) <- "EC_identity"
```

```
DefaultAssay(EC_Endo_cells) = "RNA"
DEG_Libra_edger_lrt_broad_EC_endo <- run_de(input = EC_Endo_cells, meta=meta, replicate_col = "replicate", cell_type_
       label_col = "condition",  n_threads = 8)
DefaultAssay(EC_Endo_cells)  <- "integrated"
```

```
DEG_Libra_edger_lrt_broad_EC_endo <- DEG_Libra_edger_lrt_broad_EC_endo %>% group_by(cell_type) %>% arrange(desc(abs(a
DEG_Libra_edger_lrt_broad_EC_endo%>% head
write.csv(DEG_Libra_edger_lrt_broad_EC_endo, "results_seurat/mut_vs_sib_libra_EdgeR_Lrt_broad_mt_removed_EC_endo.csv"
```

7.2 sub cluster FBs

```
FB_cells <- subset(combined_sct, subset = cell_type == "FB_1" | cell_type == "FB_2" )
FB_cells
FB_cells <- SCTransform(FB_cells, verbose = FALSE)
FB_cells <- RunPCA(FB_cells, verbose = FALSE, npcs = 50)
ElbowPlot(FB_cells,ndims = 50)
```

```
FB_cells <- RunUMAP(FB_cells, dims = 1:50)
FB_cells <- FindNeighbors(object = FB_cells)
FB_cells <- FindClusters(FB_cells, resolution = c(0.2, 0.4, 0.6, 0.8, 1.0, 1.2))
```

```
DimPlot(FB_cells, reduction = "umap", group.by = "orig.ident", label = TRUE, repel = TRUE)
```

```
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.0.2", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.0.4", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.0.6", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.0.8", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.1", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.1.2", label = TRUE, repel = TRUE, label.size = 6)
```

```
Idents(FB_cells) <- "SCT_snn_res.0.4"
DimPlot(FB_cells, reduction = "umap", group.by = "SCT_snn_res.0.4", label = TRUE, repel = TRUE, label.size = 6)
DimPlot(FB_cells, reduction = "umap", label = TRUE, repel = TRUE, label.size = 6)
```

```
FB_cells_markers <- FindAllMarkers(FB_cells, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
FB_cells_markers %>% head
FB_cells_markers %>% group_by(cluster) %>% top_n(n = 5, wt = avg_log2FC) %>% ungroup() %>% arrange(cluster, avg_log2F
FB_cells_markers %>% group_by(cluster) %>% arrange(cluster, desc(avg_log2FC)) %>% write.csv(file = "results_seurat/FB
```

```
FeaturePlot(FB_cells, features = c("notch1b"), label=TRUE, reduction = "umap", label.size = 6)
VlnPlot(FB_cells, features = c("notch1b"), split.by = "condition")
```

```
table(FB_cells@meta.data$condition)
table(FB_cells@meta.data$condition, FB_cells@meta.data$`SCT_snn_res.0.4`)
```

```
DefaultAssay(FB_cells) = "RNA"
DEG_Libra_edger_lrt_broad_FB <- run_de(input = FB_cells, meta=meta, replicate_col = "replicate", cell_type_col = "SCT
       label_col = "condition",  n_threads = 8)
DefaultAssay(FB_cells)  <- "integrated"
```

7.3 Transfer labels from different subclusters to the main clusters

```
combined_sct@meta.data$cell_name <- rownames(combined_sct@meta.data)
combined_sct@meta.data %>% head()
```

```
EC_Endo_cells@meta.data$cell_name  <- rownames(EC_Endo_cells@meta.data)
EC_Endo_cells %>% head()
```

```r
EC_Endo_cells@meta.data[,c("cell_name", "EC_identity")] %>% head
```

```r
DimPlot(EC_Endo_cells, label = TRUE)
```

```r
combined_sct@meta.data <- combined_sct@meta.data[,!colnames(combined_sct@meta.data) %in% c('EC_identity.x','EC_identi
combined_sct@meta.data %>% names()
```

```r
combined_sct@meta.data <- merge(combined_sct@meta.data, EC_Endo_cells@meta.data[,c("cell_name", "EC_identity")], by =
rownames(combined_sct@meta.data) <- combined_sct@meta.data$cell_name
combined_sct@meta.data %>% head
```

```r
# replace cell_type in combined_sct@meta.data dataframe with entried of EC identity wherever EC_identiy is not NA
combined_sct@meta.data$cell_type_EC <- combined_sct@meta.data$cell_type
combined_sct@meta.data$cell_type_EC[!is.na(combined_sct@meta.data$EC_identity)] <- combined_sct@meta.data$EC_identity
combined_sct@meta.data %>% head
combined_sct@meta.data$cell_type %>% unique()
combined_sct@meta.data[combined_sct@meta.data$cell_type=="EC_Endocardium",] %>% head(10)
combined_sct@meta.data$cell_type_EC %>% unique()
```

```r
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "CM_1"] <- "Ventricle"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "CM_2"] <- "Atrium"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "CM_3"] <- "AV_canal"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "FB_2"] <- "SMC_BA"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Neurons"] <- "unknown_1"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Neurons?"] <- "Neurons"
```

```r
combined_sct@meta.data$broad_class <- combined_sct@meta.data$cell_type_EC
```

```
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Atrium"] <- "Myocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "AV_canal"] <- "Myocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "EC_1"] <- "Endocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "EC_2"] <- "Endocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "EC_3"] <- "Endocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "EC_4"] <- "Endocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "epicardium"] <- "Epithelial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "epithelial cells"] <- "Epithelial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Erythrocytes"] <- "Erythrocytes"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "FB_1"] <- "Fibroblast_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Immune cells"] <- "Immune_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Neurons"] <- "Neuronal_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Skeletal muscle"] <- "Muscle_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "SMC_BA"] <- "Muscle_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "unknown_1"] <- "Unknown"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "unknown_2"] <- "Unknown"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Valve_cells"] <- "Endocardial_cells"
combined_sct@meta.data$broad_class[(combined_sct@meta.data$cell_type_EC)== "Ventricle"] <- "Myocardial_cells"
```

```
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Atrium"] <- "CM_Atrium"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "AV_canal"] <- "CM_AV_canal"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "EC_1"] <- "EnC_1"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "EC_2"] <- "EnC_2"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "EC_3"] <- "EnC_3"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "EC_4"] <- "EnC_4"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "epicardium"] <- "Ep_epicardium"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "epithelial cells"] <- "Ep_epithelial cel
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Erythrocytes"] <- "Erythrocytes"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "FB_1"] <- "FB"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Immune cells"] <- "Immune_cells"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Neurons"] <- "Neurons"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Skeletal muscle"] <- "Mus_Skeletal_muscl
```

```r
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "SMC_BA"] <- "Mus_SMC_BA"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "unknown_1"] <- "unknown_1"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "unknown_2"] <- "unknown_2"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Valve_cells"] <- "EnC_Valve_cells"
combined_sct@meta.data$cell_type_EC[(combined_sct@meta.data$cell_type_EC)== "Ventricle"] <- "CM_Ventricle"
```

```r
DimPlot(combined_sct, reduction = "umap", group.by = "cell_type_EC", label = TRUE, repel = TRUE, label.size = 6)
```

```r
# change order of combined_sct@meta.data to keep sib first then mutant
combined_sct@meta.data$condition <- factor(combined_sct@meta.data$condition,levels = c("sib", "mut"))
combined_sct@meta.data$orig.ident <- factor(combined_sct@meta.data$orig.ident,levels = c("sib_1","sib_2", "mut_1", "m
```

```r
unique(combined_sct@meta.data$orig.ident)
unique(combined_sct@meta.data$condition)
```

```r
combined_sct@meta.data<- combined_sct@meta.data[order(match(combined_sct@meta.data$cell_name,names(combined_sct@activ
```

```r
identical(combined_sct@meta.data$cell_name,names(combined_sct@active.ident))
```

```r
Idents(combined_sct) <- "cell_type_EC"
```

```r
DimPlot( combined_sct, label = TRUE, repel = TRUE, label.size = 6)
```

```r
DimPlot( combined_sct, group.by = "broad_class",label = TRUE, repel = TRUE, label.size = 6)
```

```r
# save seurat object for shiny app
saveRDS(combined_sct, "results_seurat/combined_sct.rds")
```

8 Markers for each cluster and subclusters after cell assignment

```
# Find markers with all the labels
all_markers_cell_assigned <- FindAllMarkers(combined_sct, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
all_markers_cell_assigned %>% head
# arrange by descending log2fc and cell name and write to csv file
all_markers_cell_assigned %>% arrange(cluster, desc(avg_log2FC)) %>% write.csv("results_seurat/cell_assigned/all_mark
```

```
all_markers_cell_assigned_mouse <- merge(all_markers_cell_assigned, Mouse_Genes[!duplicated(Mouse_Genes$ZFIN.symbol),
head(all_markers_cell_assigned_mouse)
```

```
all_markers_cell_assigned_mouse %>% arrange(cluster, desc(abs(avg_log2FC))) %>% write.csv("results_seurat/cell_assign
```

```
# make plot bigger
options(repr.plot.width=24, repr.plot.height=36)

all_markers_cell_assigned %>%
    group_by(cluster) %>%
    top_n(n = 20, wt = avg_log2FC) -> top10
dev.copy(pdf, "results_seurat/cell_assigned/all_markers_cell_assigned.pdf", width = 24, height = 36)
DoHeatmap(combined_sct, features = top10$gene) + NoLegend()
dev.off()
# make plots small again
options(repr.plot.width=12, repr.plot.height=12)
```

9 Differential Expression Analysis using Libra

```
DefaultAssay(combined_sct) = "RNA"
DEG_Libra_edger_lrt_cell_assigned<- run_de(input = combined_sct, meta=meta, replicate_col = "replicate", cell_type_co
       label_col = "condition",  n_threads = 8)
DefaultAssay(combined_sct)  <- "integrated"
```

```
DEG_Libra_edger_lrt_cell_assigned <- DEG_Libra_edger_lrt_cell_assigned %>% group_by(cell_type) %>% arrange(desc(abs(a
DEG_Libra_edger_lrt_cell_assigned%>% head
write.csv(DEG_Libra_edger_lrt_cell_assigned, "results_seurat/cell_assigned/mut_vs_sib_libra_EdgeR_Lrt_cell_assigned.c
```

```
# plot heatmaps for significant genes for each cell type with padj < 0.05
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type

for (i in 1:length(cell_type)){

genes_to_plot <- DEG_Libra_edger_lrt_cell_assigned$gene[DEG_Libra_edger_lrt_cell_assigned$p_val_adj<=0.05 & DEG_Libra
print(paste0("genes significant in ",cell_type[i], " are:"))
genes_to_plot %>% print

if(length(genes_to_plot)==0){
    print(paste0("no significant genes in cluster ", cell_type[i]))
    next
}

heatmaps <- DoHeatmap(subset(combined_sct, idents = cell_type[i]),features = genes_to_plot,group.by= "condition", ass
print(heatmaps)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/heatmaps_significant_genes_mut_vs_sib/",cell_type[i],"_heatmap_signif
```

```
    width = 10,
    height = 10
    )
    dev.off ()
}
```

```
# plot heatmaps for significant genes for each cell type with pval<=0.05
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type

for (i in 1:length(cell_type)){

genes_to_plot <- DEG_Libra_edger_lrt_cell_assigned$gene[DEG_Libra_edger_lrt_cell_assigned$p_val<=0.05 & DEG_Libra_edg
print(paste0("genes significant (", length(genes_to_plot),") in ",cell_type[i], " are (printing top 20):" ))
genes_to_plot %>%  head(20) %>% print

if(length(genes_to_plot)==0){
    print(paste0("no significant genes in cluster ", cell_type[i]))
    next
}

heatmaps <- DoHeatmap(subset(combined_sct, idents = cell_type[i]),features = genes_to_plot,group.by= "condition", ass
print(heatmaps)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/heatmaps_significant_genes_mut_vs_sib_pva05/",cell_type[i],"_heatmap_
    width = 10,
    height = 10
    )
    dev.off ()
}
```

```
# merge with Mouse genes
DEG_Libra_edger_lrt_cell_assigned_mouse <- merge(DEG_Libra_edger_lrt_cell_assigned, Mouse_Genes, by.x = "gene", by.y
DEG_Libra_edger_lrt_cell_assigned_mouse  <- DEG_Libra_edger_lrt_cell_assigned_mouse %>% arrange(cell_type, p_val_adj,
DEG_Libra_edger_lrt_cell_assigned_mouse %>% write.csv("results_seurat/cell_assigned/mut_vs_sib_libra_EdgeR_Lrt_cell_a

DEG_Libra_edger_lrt_cell_assigned_mouse <- merge(DEG_Libra_edger_lrt_cell_assigned, Mouse_Genes[!duplicated(Mouse_Gen
DEG_Libra_edger_lrt_cell_assigned_mouse  <- DEG_Libra_edger_lrt_cell_assigned_mouse %>% arrange(cell_type, p_val_adj,
DEG_Libra_edger_lrt_cell_assigned_mouse %>% head
```

10 Number of cells in each cluster

```
combined_sct@meta.data$condition <- NULL
combined_sct@meta.data$condition[grepl("sib",combined_sct@meta.data$orig.ident)] <- "sib"
combined_sct@meta.data$condition[grepl("mut",combined_sct@meta.data$orig.ident)] <- "mut"
combined_sct@meta.data$condition %>% unique()
```

```
table(combined_sct@meta.data$cell_type, combined_sct@meta.data$orig.ident)
table(combined_sct@meta.data$cell_type, combined_sct@meta.data$condition)
```

```
cell_type_numbers <- table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$orig.ident)
cell_type_numbers <- as.data.frame(cell_type_numbers)
cell_type_numbers %>% head
cell_type_numbers <- cell_type_numbers %>% dplyr::rename(cell_type = Var1, sample_name = Var2)
cell_type_numbers$sample_name <- factor(cell_type_numbers$sample_name, levels = c("sib_1","sib_2", "mut_1", "mut_2"))
cell_type_numbers %>% write.csv("results_seurat/cell_type_numbers.csv")
cell_type_numbers %>% head
```

```r
# plot these percentages using ggplot2
# save the graph as a pdf file
# dev.copy(pdf, "results_seurat/cell_type_percentages_normalized.pdf", width = 10, height = 10)
ggplot(cell_type_percentages, aes(x = cell_type, y = Freq, fill = sample_name)) +
  geom_bar(stat = "identity", position = "fill") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  labs(x = "Cell_type", y = "Percentage of cells", fill = "Sample name")
# dev.off()
```

```r
# calculate the percentage of cell in each cell type according to condition
cell_type_percentages <- prop.table(table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$orig.ident))*10
cell_type_percentages <- as.data.frame(cell_type_percentages)
cell_type_percentages %>% head
cell_type_percentages <- cell_type_percentages %>% dplyr::rename(cell_type = Var1, sample_name = Var2)
cell_type_percentages$sample_name <- factor(cell_type_percentages$sample_name, levels = c("sib_1","sib_2", "mut_1", "
cell_type_percentages %>% write.csv("results_seurat/cell_type_percentages.csv")
cell_type_percentages %>% head

# plot these percentages using ggplot2
# save the graph as a pdf file
dev.copy(pdf, "results_seurat/cell_type_percentages_normalized.pdf", width = 10, height = 10)
ggplot(cell_type_percentages, aes(x = cell_type, y = Freq, fill = sample_name)) +
  geom_bar(stat = "identity", position = "fill") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  labs(x = "Cell_type", y = "Percentage of cells", fill = "Sample name")
dev.off()
```

```r
# calculate the percentage of cell in each cell type according to condition
# save the graph as a pdf file
dev.copy(pdf, "results_seurat/cell_type_percentages.pdf", width = 10, height = 10)
ggplot(cell_type_percentages, aes(x = cell_type, y = Freq, fill = sample_name)) +
  geom_bar(stat = "identity", position = "dodge") +
```

```
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  labs(x = "Cell_type", y = "Percentage of cells", fill = "Sample name")
dev.off()
```

```
cell_type_cond_numbers <- table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$condition)
cell_type_cond_numbers <- as.data.frame(cell_type_cond_numbers)
cell_type_cond_numbers <- cell_type_cond_numbers %>% dplyr::rename(cell_type = Var1, sample_name = Var2)
cell_type_cond_numbers$sample_name <- factor(cell_type_cond_numbers$sample_name, levels = c("sib","mut"))
cell_type_cond_numbers %>% write.csv("results_seurat/cell_type_cond_numbers.csv")
cell_type_cond_numbers
```

```
# calculate the percentage of cell in each cell type according to condition
cell_type_percentages_cond <- prop.table(table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$condition)
cell_type_percentages_cond <- as.data.frame(cell_type_percentages_cond)
cell_type_percentages_cond %>% head
cell_type_percentages_cond <- cell_type_percentages_cond %>% dplyr::rename(cell_type = Var1, condition = Var2)
cell_type_percentages_cond$condition <- factor(cell_type_percentages_cond$condition, levels = c("sib","mut"))
cell_type_percentages_cond %>% write.csv("results_seurat/cell_type_percentages_cond.csv")
cell_type_percentages_cond %>% head
# plot these percentages using ggplot2
# save the graph as a pdf file
dev.copy(pdf, "results_seurat/cell_type_percentages_cond_normalized.pdf", width = 10, height = 10)
ggplot(cell_type_percentages_cond, aes(x = cell_type, y = Freq, fill = condition)) +
  geom_bar(stat = "identity", position = "fill") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
  labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")
  dev.off()
```

```
# calculate the percentage of cell in each cell type according to condition
# plot these percentages using ggplot2
# save the graph as a pdf file
```

```
dev.copy(pdf, "results_seurat/cell_type_percentages_cond.pdf", width = 10, height = 10)
ggplot(cell_type_percentages_cond, aes(x = cell_type, y = Freq, fill = condition)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
  labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")
  dev.off()
```

11 Pathway analysis for Differential Genes in each cluster

```
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type
```

```
DEG_Libra_edger_lrt_cell_assigned_mouse <- merge(DEG_Libra_edger_lrt_cell_assigned, Mouse_Genes[!duplicated(Mouse_Gen
DEG_Libra_edger_lrt_cell_assigned_mouse  <- DEG_Libra_edger_lrt_cell_assigned_mouse %>% arrange(cell_type, p_val_adj,
DEG_Libra_edger_lrt_cell_assigned_mouse %>% head
```

```
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type
```

11.1 Over representation Analysis

```
pval_enrich <- 0.2
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type
# Loop through each cell type
for(i in cell_type){
    # Print message indicating which cell type is being analyzed
    message(paste0("Molecular function Enrichments for Cell type: ", i))
```

```r
# Perform GO enrichment analysis for DEG for the current cell type
compGO_MF_diff <- enrichGO(gene = DEG_Libra_edger_lrt_cell_assigned_mouse$MGI.symbol[DEG_Libra_edger_lrt_cell_ass
                           pAdjustMethod = "BH",OrgDb = "org.Mm.eg.db", ont = "MF")

# If no enriched pathways are found, print message and move on to next cell type
if(is.null(compGO_MF_diff)){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*******************************************************************************"))
    message(paste0("\n"))
    next
}
if(sum(compGO_MF_diff@result$p.adjust<pval_enrich)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*******************************************************************************"))
    message(paste0("\n"))
    next
}

# Generate dotplot of enriched pathways for the current cell type
print(dotplot(compGO_MF_diff, showCategory = 15, title = paste0("GO Pathway Enrichment Analysis for DEG \n Molecu
        font.size = 12)+ scale_colour_viridis(option = "plasma", direction = 1))

# Save dotplot as pdf file
dev.copy(
pdf,
file = paste0("results_seurat/cell_assigned/pathways_DEG_pval05/Molecular_function/",i,"_GO_MF_pathways_pval05_pl
width = 10,
height = 8
)
dev.off ()

# Convert enriched pathways data to data frame and calculate decimal ratios
```

```
    compGO_MF_diff_df <- as.data.frame(compGO_MF_diff)
    compGO_MF_diff_df$GeneRatio_decimal <- compGO_MF_diff_df$GeneRatio
    compGO_MF_diff_df$GeneRatio_decimal <- sapply(compGO_MF_diff_df$GeneRatio_decimal,
                                                  function(x) (eval(parse(text = as.character(x)))))
    compGO_MF_diff_df$BgRatio_decimal <- compGO_MF_diff_df$BgRatio
    compGO_MF_diff_df$BgRatio_decimal <- sapply(compGO_MF_diff_df$BgRatio_decimal,
                                                function(x) (eval(parse(text = as.character(x)))))
    compGO_MF_diff_df <- compGO_MF_diff_df %>% tidyr::separate_rows(geneID, sep = "/", convert = FALSE) %>%
      arrange(desc(GeneRatio_decimal))

    # Print the first few rows of the enriched pathways data frame
    compGO_MF_diff_df %>% head

    # Save enriched pathways data frame as CSV file
    write.csv(compGO_MF_diff_df, paste0("results_seurat/cell_assigned/pathways_DEG_pval05/Molecular_function/",i,"_GO

    # Print message indicating that analysis for the current cell type is complete
    message(paste0("Cell type: ", i, " done"))
    message(paste0("################################################################################"))
    message(paste0("\n"))
}

# Loop through each cell type
for(i in cell_type){
    # Print message indicating which cell type is being processed
    message(paste0("Cellular component Enrichments for Cell type: ", i))

    # Perform GO enrichment analysis for DEG for cellular components
    compGO_CC_diff <- enrichGO(gene = DEG_Libra_edger_lrt_cell_assigned_mouse$MGI.symbol[DEG_Libra_edger_lrt_cell_ass
                               pvalueCutoff  = pval_enrich,
                               keyType = "SYMBOL",
                               pAdjustMethod = "BH",
```

```
                                OrgDb = "org.Mm.eg.db",
                                ont = "CC")

# If there are no enriched pathways, skip to the next cell type
if(is.null(compGO_CC_diff)){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*********************************************************************************"))
    message(paste0("\n"))
    next
}
if(sum(compGO_CC_diff@result$p.adjust<pval_enrich)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*********************************************************************************"))
    message(paste0("\n"))
    next
}
# Generate a dotplot of enriched pathways for cellular components
print(dotplot(compGO_CC_diff,
              showCategory = 15,
              title = paste0("GO Pathway Enrichment Analysis for DEG \n Cellular components for ",i, " Cells"),
              font.size = 12)+ scale_colour_viridis(option = "plasma", direction = 1))

# Save the dotplot as an pdf file
# dev.new()
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/pathways_DEG_pval05/Cellular_component/",i,"_GO_CC_pathways_pval0
    width = 10,
    height = 8
)
dev.off ()
```

```
    # Convert the enriched pathways data to a data frame
    compGO_CC_diff_df <- as.data.frame(compGO_CC_diff)

    # Convert GeneRatio and BgRatio to decimal format
    compGO_CC_diff_df$GeneRatio_decimal <- compGO_CC_diff_df$GeneRatio
    compGO_CC_diff_df$GeneRatio_decimal <- sapply(compGO_CC_diff_df$GeneRatio_decimal,
                                                  function(x) (eval(parse(text = as.character(x)))))
    compGO_CC_diff_df$BgRatio_decimal <- compGO_CC_diff_df$BgRatio
    compGO_CC_diff_df$BgRatio_decimal <- sapply(compGO_CC_diff_df$BgRatio_decimal,
                                                function(x) (eval(parse(text = as.character(x)))))

    # Separate gene IDs and arrange by GeneRatio_decimal in descending order
    compGO_CC_diff_df <- compGO_CC_diff_df %>%
        tidyr::separate_rows(geneID, sep = "/", convert = FALSE) %>%
        arrange(desc(GeneRatio_decimal))

    # Print the first few rows of the enriched pathways data frame
    compGO_CC_diff_df %>% head

    # Save the enriched pathways data frame as a CSV file
    write.csv(compGO_CC_diff_df, paste0("results_seurat/cell_assigned/pathways_DEG_pval05/Cellular_component/",i,"_GO

    # Print message indicating that the cell type is done processing
    message(paste0("Cell type: ", i, " done"))
    message(paste0("********************************************************************************"))
    message(paste0("\n"))
}


# Loop through each cell type
for(i in cell_type){
    # Print message indicating which cell type is being processed
```

```r
message(paste0("Biological pathways Enrichments for Cell type: ", i))

# Perform GO enrichment analysis for DEG for biological pathways
compGO_BP_diff <- enrichGO(gene = DEG_Libra_edger_lrt_cell_assigned_mouse$MGI.symbol[DEG_Libra_edger_lrt_cell_ass
                           pAdjustMethod = "BH",OrgDb = "org.Mm.eg.db", ont = "BP")

# If there are no enriched pathways, skip to the next cell type
if(is.null(compGO_BP_diff)){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("**********************************************************************************"))
    message(paste0("\n"))
    next
}
if(sum(compGO_BP_diff@result$p.adjust<pval_enrich)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("**********************************************************************************"))
    message(paste0("\n"))
    next
}
# Create a dotplot of the enriched pathways
print(dotplot(compGO_BP_diff, showCategory = 15, title = paste0("GO Pathway Enrichment Analysis for DEG \n Biolog
        font.size = 12)+ scale_colour_viridis(option = "plasma", direction = 1))

# Save the dotplot as an pdf file
dev.copy(
pdf,
file = paste0("results_seurat/cell_assigned/pathways_DEG_pval05/Biological_pathways/",i,"_GO_BP_pathways_pval05_p
width = 10,
height = 8
)
dev.off ()
```

```
    # Convert the enriched pathways data to a dataframe and calculate decimal ratios
    compGO_BP_diff_df <- as.data.frame(compGO_BP_diff)
    compGO_BP_diff_df$GeneRatio_decimal <- compGO_BP_diff_df$GeneRatio
    compGO_BP_diff_df$GeneRatio_decimal <- sapply(compGO_BP_diff_df$GeneRatio_decimal,
                                                  function(x) (eval(parse(text = as.character(x)))))
    compGO_BP_diff_df$BgRatio_decimal <- compGO_BP_diff_df$BgRatio
    compGO_BP_diff_df$BgRatio_decimal <- sapply(compGO_BP_diff_df$BgRatio_decimal,
                                                function(x) (eval(parse(text = as.character(x)))))

    # Separate the gene IDs into separate rows and sort by descending GeneRatio_decimal
    compGO_BP_diff_df <- compGO_BP_diff_df %>% tidyr::separate_rows(geneID, sep = "/", convert = FALSE) %>%
      arrange(desc(GeneRatio_decimal))

    # Print the first few rows of the enriched pathways dataframe
    compGO_BP_diff_df %>% head

    # Save the enriched pathways dataframe as a CSV file
    write.csv(compGO_BP_diff_df, paste0("results_seurat/cell_assigned/pathways_DEG_pval05/Biological_pathways/",i,"_G

    # Print message indicating that the cell type is done processing
    message(paste0("Cell type: ", i, " done"))
    message(paste0("********************************************************************************"))
    message(paste0("\n"))
}
```

11.2 Geneset Enrichment Analysis

```
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type
```

```r
pval_enrich <- 0.2
cell_type <- DEG_Libra_edger_lrt_cell_assigned$cell_type %>% unique %>% as.character %>% sort
cell_type
# Loop through each cell type
for(i in cell_type){
    # Print message indicating which cell type is being analyzed
    message(paste0("Molecular function Enrichments for Cell type: ", i))

    # create gene list for the current cell type
    gene_list_df <- DEG_Libra_edger_lrt_cell_assigned_mouse[DEG_Libra_edger_lrt_cell_assigned_mouse$cell_type==i & DE
    gene_list_df <- gene_list_df %>% arrange(desc(avg_logFC))
    gene_list_df <- gene_list_df[!is.na(gene_list_df$MGI.symbol),]
    gene_list_df <- gene_list_df[!duplicated(gene_list_df$MGI.symbol),]
    gene_list <- gene_list_df %>% pull(avg_logFC)
    names(gene_list) <- gene_list_df %>% pull(MGI.symbol)

    gene_list <- gene_list[!duplicated(gene_list)]
     if(is.null(gene_list)|length(gene_list)==0){

        message(paste0("Cell type: ", i, " done"))
        message(paste0("********************************************************************************"))
        message(paste0("\n"))
        next
    }
    # Perform GO enrichment analysis for DEG for the current cell type
    compGO_MF_diff <- gseGO(gene = gene_list, pvalueCutoff  = pval_enrich,keyType = "SYMBOL",
                            pAdjustMethod = "BH",OrgDb = "org.Mm.eg.db", ont = "MF")
    # If no enriched pathways are found, print message and move on to next cell type
    if(is.null(compGO_MF_diff)|nrow(compGO_MF_diff@result)==0){
        message(paste0("Cell type: ", i, " done"))
        message(paste0("********************************************************************************"))
        message(paste0("\n"))
        next
```

```
 }
 if(sum(compGO_MF_diff@result$p.adjust<pval_enrich)==0){
     message(paste0("Cell type: ", i, " done"))
     message(paste0("*********************************************************************************************"))
     message(paste0("\n"))
     next
 }
 # Generate dotplot of enriched pathways for the current cell type
print(dotplot(compGO_MF_diff, showCategory = 15, title = paste0("GO Pathway Geneset Enrichment Analysis for DEG \n
         font.size = 12) + facet_grid(.~.sign)+ scale_colour_viridis(option = "plasma", direction = 1))
 # Save dotplot as pdf file
 dev.copy(
 pdf,
 file = paste0("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Molecular_function/",i,"_gse_GO_MF_pathways_p
 width = 10,
 height = 8
 )
 dev.off ()

 # Convert enriched pathways data to data frame and calculate decimal ratios
 # Generate dotplot of enriched pathways for the current cell type


     compGO_MF_diff_df <- as.data.frame(compGO_MF_diff)
     compGO_MF_diff_df

 compGO_MF_diff_df <- compGO_MF_diff_df %>% tidyr::separate_rows(core_enrichment, sep = "/", convert = FALSE) %>%
   arrange((p.adjust))

 # Save enriched pathways data frame as CSV file
 write.csv(compGO_MF_diff_df, paste0("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Molecular_function/",i,
```

```
    # Print message indicating that analysis for the current cell type is complete
    message(paste0("Cell type: ", i, " done"))
    message(paste0("********************************************************************************"))
    message(paste0("\n"))
}

# Loop through each cell type
for(i in cell_type){
    # Print message indicating which cell type is being processed
    message(paste0("Cellular component Enrichments for Cell type: ", i))

     # create gene list for the current cell type
    gene_list_df <- DEG_Libra_edger_lrt_cell_assigned_mouse[DEG_Libra_edger_lrt_cell_assigned_mouse$cell_type==i & DE
    gene_list_df <- gene_list_df %>% arrange(desc(avg_logFC))
    gene_list_df <- gene_list_df[!is.na(gene_list_df$MGI.symbol),]
    gene_list_df <- gene_list_df[!duplicated(gene_list_df$MGI.symbol),]
    gene_list <- gene_list_df %>% pull(avg_logFC)
    names(gene_list) <- gene_list_df %>% pull(MGI.symbol)

    gene_list <- gene_list[!duplicated(gene_list)]


    gene_list <- gene_list[!duplicated(gene_list)]
     if(is.null(gene_list)|length(gene_list)==0){
        # print("trace_gene_if")
        message(paste0("Cell type: ", i, " done"))
        message(paste0("********************************************************************************"))
        message(paste0("\n"))
        next
    }
    # Perform GO enrichment analysis for DEG for cellular components
    compGO_CC_diff <- gseGO(gene = gene_list,
```

```
                                pvalueCutoff  = pval_enrich,
                                keyType = "SYMBOL",
                                pAdjustMethod = "BH",
                                OrgDb = "org.Mm.eg.db",
                                ont = "CC")

# If there are no enriched pathways, skip to the next cell type
if(is.null(compGO_CC_diff)|nrow(compGO_CC_diff@result)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*********************************************************************************"))
    message(paste0("\n"))
    next
}
if(sum(compGO_CC_diff@result$p.adjust<pval_enrich)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*********************************************************************************"))
    message(paste0("\n"))
    next
}
# Generate a dotplot of enriched pathways for cellular components
print(dotplot(compGO_CC_diff,
              showCategory = 15,
              title = paste0("GO Pathway Geneset Enrichment Analysis for DEG \n Cellular components for ",i, " Ce
              font.size = 12)+ facet_grid(.~.sign)+ scale_colour_viridis(option = "plasma", direction = 1))

# Save the dotplot as an pdf file

dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Cellular_component/",i,"_gse_GO_CC_pathwa
    width = 10,
    height = 8
```

```
    )
    dev.off ()

    # Convert the enriched pathways data to a data frame
    compGO_CC_diff_df <- as.data.frame(compGO_CC_diff)
    compGO_CC_diff_df <- compGO_CC_diff_df %>%
        tidyr::separate_rows(core_enrichment, sep = "/", convert = FALSE) %>%
        arrange(p.adjust)

    # Print the first few rows of the enriched pathways data frame
    compGO_CC_diff_df %>% head

    # Save the enriched pathways data frame as a CSV file
    write.csv(compGO_CC_diff_df, paste0("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Cellular_component/",i,

    # Print message indicating that the cell type is done processing
    message(paste0("Cell type: ", i, " done"))
    message(paste0("**********************************************************************************************"))
    message(paste0("\n"))
}


# Loop through each cell type
for(i in cell_type){
    # Print message indicating which cell type is being processed
    message(paste0("Biological pathways Enrichments for Cell type: ", i))

    gene_list_df <- DEG_Libra_edger_lrt_cell_assigned_mouse[DEG_Libra_edger_lrt_cell_assigned_mouse$cell_type==i & DE
    gene_list_df <- gene_list_df %>% arrange(desc(avg_logFC))
    gene_list_df <- gene_list_df[!is.na(gene_list_df$MGI.symbol),]
    gene_list_df <- gene_list_df[!duplicated(gene_list_df$MGI.symbol),]
    gene_list <- gene_list_df %>% pull(avg_logFC)
```

```
names(gene_list) <- gene_list_df %>% pull(MGI.symbol)

gene_list <- gene_list[!duplicated(gene_list)]
gene_list %>% head
gene_list %>% length
gene_list_df %>% head()


gene_list <- gene_list[!duplicated(gene_list)]
 if(is.null(gene_list)|length(gene_list)==0){
    # print("trace_gene_if")
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*************************************************************************************"))
    message(paste0("\n"))
    next
}
# Perform GO enrichment analysis for DEG for biological pathways
compGO_BP_diff <- gseGO(gene = gene_list, pvalueCutoff  = pval_enrich,keyType = "SYMBOL",
                        pAdjustMethod = "BH",OrgDb = "org.Mm.eg.db", ont = "BP")

# If there are no enriched pathways, skip to the next cell type
if(is.null(compGO_BP_diff)|nrow(compGO_BP_diff@result)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*************************************************************************************"))
    message(paste0("\n"))
    next
}
if(sum(compGO_BP_diff@result$p.adjust<pval_enrich)==0){
    message(paste0("Cell type: ", i, " done"))
    message(paste0("*************************************************************************************"))
    message(paste0("\n"))
    next
```

```
    }
    # Create a dotplot of the enriched pathways
    print(dotplot(compGO_BP_diff, showCategory = 15, title = paste0("GO Pathway Geneset Enrichment Analysis for DEG \
            font.size = 12)+ facet_grid(.~.sign)+ scale_colour_viridis(option = "plasma", direction = 1))

    # Save the dotplot as an pdf file
    dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Biological_pathways/",i,"_gse_GO_BP_pathways_
    width = 10,
    height = 8
    )
    dev.off ()

    # Convert the enriched pathways data to a dataframe and calculate decimal ratios
    compGO_BP_diff_df <- as.data.frame(compGO_BP_diff)

    # Separate the gene IDs into separate rows and sort by descending GeneRatio_decimal
    compGO_BP_diff_df <- compGO_BP_diff_df %>% tidyr::separate_rows(core_enrichment, sep = "/", convert = FALSE) %>%
      arrange(p.adjust)

    # Print /the first few rows of the enriched pathways dataframe
    compGO_BP_diff_df %>% head

    # Save the enriched pathways dataframe as a CSV file
    write.csv(compGO_BP_diff_df, paste0("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Biological_pathways/",i

    # Print message indicating that the cell type is done processing
    message(paste0("Cell type: ", i, " done"))
    message(paste0("################################################################################"))
    message(paste0("\n"))
}
```

12 Graph for Number of DEGs

```
broad_sub_cluster_names <- combined_sct@meta.data %>% dplyr::select(cell_type_EC, broad_class) %>% distinct() %>% arr
rownames(broad_sub_cluster_names) <- NULL
broad_sub_cluster_names
```

```
# filter the data for signicant values with padj <= 0.05
DEG_Libra_edger_lrt_cell_assigned_sig_padj05 <- DEG_Libra_edger_lrt_cell_assigned %>% filter(p_val_adj<=0.05)
# count how many genes are significant for each cell type
num_of_DEG_padj05 <- DEG_Libra_edger_lrt_cell_assigned_sig_padj05 %>% group_by(cell_type) %>% summarise(n=n()) %>% ar
num_of_DEG_padj05
```

```
# filter the data for signicant values with padj <= 0.05
DEG_Libra_edger_lrt_cell_assigned_sig_pval05 <- DEG_Libra_edger_lrt_cell_assigned %>% filter(p_val<=0.05)
# count how many genes are significant for each cell type
num_of_DEG_pval05 <- DEG_Libra_edger_lrt_cell_assigned_sig_pval05 %>% group_by(cell_type) %>% summarise(n=n()) %>% ar
num_of_DEG_pval05
```

```
num_of_DEG_pval05 <-merge(num_of_DEG_pval05, broad_sub_cluster_names, by.x = "cell_type", by.y = "cell_type_EC", all.
head(num_of_DEG_pval05)
```

```
num_of_DEG_pval05 <- num_of_DEG_pval05 %>% group_by(broad_class) %>% arrange(desc(n),cell_type,broad_class, .by_group
num_of_DEG_pval05
```

```
# plot a bar graph coloring the bars by broad_class arrange descending in each group
num_of_DEG_pval05 %>% group_by(broad_class) %>%  arrange(desc(n),cell_type,broad_class, .by_group = TRUE)  %>% ggplot
```

13 Heatmaps for all OFT and AV genes

```
OFT_related_genes_all <- c("axin2", "acvr1l", "atf2", "bmp4", "bmp7a", "bmp7b", "bmpr1aa", "bmpr1ab", "bmpr2a", "bmpr
```

```
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'EnC_1', 'EnC_2', 'EnC_3', 'EnC_4', 'EnC_Valve_cells
    annot.by = c("cell_type_EC", "condition"), complex = TRUE,
    heatmap.colors = PurpleAndYellow(50),
    # scaled.to.max=TRUE,
    # scale='none',
    heatmap.colors.max.scaled = PurpleAndYellow(50))
```

```
AV_related_genes_all <- c("acvr1l", "adamts5", "anxa5b", "apcdd1l", "aplnra", "aplnrb", "axin2", "bgna", "bmp2a", "bm
```

```
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
DoHeatmap(combined_sct, features= AV_related_genes_all) + NoLegend() + ggtitle("AV related genes")
DoHeatmap(combined_sct, features= AV_related_genes_all,assay = "RNA",slot = "counts" ) + ggtitle("AV related genes")
```

```
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/OFT_related_genes_in_Endocardial_cells_normalized_data.pdf"),
    width = 10,
    height = 18
    )
dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'EnC_1', 'EnC_2', 'EnC_3', 'EnC_4', 'EnC_Valve_cells
    annot.by = c("cell_type_EC", "condition"), complex = TRUE,
```

```
    heatmap.colors = PurpleAndYellow(50),
    # scaled.to.max=TRUE,
    # scale='none',
    use_raster=TRUE,
    heatmap.colors.max.scaled = PurpleAndYellow(50),
    main= "OFT_related_genes_all (normalized data)")
dev.off()
```

13.0.1 AV in CM

```
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/AV_related_genes_in_CM_cells_normalized_data.pdf"),
    width = 10,
    height = 18
    )
dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'CM_Ventricle', 'CM_Atrium', 'CM_AV_canal' ))), isGe
    annot.by = c("cell_type_EC", "condition"), complex = TRUE,
    heatmap.colors = PurpleAndYellow(50),
    # scaled.to.max=TRUE,
    # scale='none',
    use_raster=TRUE,
    heatmap.colors.max.scaled = PurpleAndYellow(50),
    main= "AV_related_genes_all in CM cells (normalized data)")



dev.off()
```

```
options(repr.plot.width=24, repr.plot.height=18)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/AV_related_genes_in_CM_cells_raw_counts.pdf"),
    width = 10,
    height = 18
    )

dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'CM_Ventricle', 'CM_Atrium', 'CM_AV_canal' ))),
             isGene(OFT_related_genes_all,combined_sct, assay = "RNA",return.values = TRUE),
             assay = "RNA",
             slot = "counts",
             annot.by = c("cell_type_EC","condition" ), complex = TRUE,
             scaled.to.max = TRUE,
             scale = 'row',
             use_raster=TRUE,
             heatmap.colors = PurpleAndYellow(50),
             heatmap.colors.max.scaled = PurpleAndYellow(25),
             main = "AV_related_genes_all in CM cells (raw counts)",
    ) %>% print
dev.off ()
```

```
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/OFT_related_genes_in_BA_FB_cells_normalized_data.pdf"),
    width = 10,
    height = 18
    )
```

```
dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'Mus_SMC_BA', 'FB' ))), isGene(OFT_related_genes_all
    annot.by = c("cell_type_EC", "condition"), complex = TRUE,
    heatmap.colors = PurpleAndYellow(50),
    # scaled.to.max=TRUE,
    # scale='none',
    use_raster=TRUE,
    heatmap.colors.max.scaled = PurpleAndYellow(50),
    main= "OFT_related_genes_all in BA_FB cells (normalized data)")
dev.off()
```

```
options(repr.plot.width=24, repr.plot.height=18)
dev.copy(
    svg,
    file = paste0("results_seurat/cell_assigned/OFT_related_genes_in_BA_FB_cells_raw_counts.svg"),
    width = 10,
    height = 18
    )

dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'Mus_SMC_BA', 'FB' ))),
             isGene(OFT_related_genes_all,combined_sct, assay = "RNA",return.values = TRUE),
             assay = "RNA",
             slot = "counts",
             annot.by = c("cell_type_EC","condition" ), complex = TRUE,
             scaled.to.max = TRUE,
             scale = 'row',
             use_raster=TRUE,
             heatmap.colors = PurpleAndYellow(50),
             heatmap.colors.max.scaled = PurpleAndYellow(25),
             main = "OFT related genes in Mus_SMC_BA and FB cells (raw counts)",
    ) %>% print
dev.off ()
```

```
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/OFT_related_genes_in_BA_Skeletal_muscle_cells_normalized_data.pdf"),
    width = 10,
    height = 18
    )
dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'Mus_SMC_BA', 'Mus_Skeletal_muscle' ))), isGene(OFT_
    annot.by = c("cell_type_EC", "condition"), complex = TRUE,
    heatmap.colors = PurpleAndYellow(50),
    # scaled.to.max=TRUE,
    # scale='none',
    use_raster=TRUE,
    heatmap.colors.max.scaled = PurpleAndYellow(50),
    main= "OFT_related_genes_all in BA_Skeletal_muscle cells (normalized data)")
dev.off()
```

```
options(repr.plot.width=24, repr.plot.height=18)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/OFT_related_genes_in_BA_Skeletal_muscle_cells_raw_counts.pdf"),
    width = 10,
    height = 18
    )

dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'Mus_SMC_BA', 'Mus_Skeletal_muscle' ))),
             isGene(OFT_related_genes_all,combined_sct, assay = "RNA",return.values = TRUE),
             assay = "RNA",
             slot = "counts",
             annot.by = c("cell_type_EC","condition" ), complex = TRUE,
             scaled.to.max = TRUE,
```

```
            scale = 'row',
            use_raster=TRUE,
            heatmap.colors = PurpleAndYellow(50),
            heatmap.colors.max.scaled = PurpleAndYellow(25),
            main = "OFT related genes in BA and Skeletal muscle cells (raw counts)",
    ) %>% print
dev.off ()
```

14 Transfer FB subclusters to combined_sct

```
FB_cells@meta.data$FB_identity <- paste0("FB_",(FB_cells@meta.data$SCT_snn_res.0.4)) %>% factor
FB_cells@meta.data %>% head
```

```
combined_sct@meta.data$cell_name <- rownames(combined_sct@meta.data)
combined_sct@meta.data %>% head()
FB_cells@meta.data$cell_name  <- rownames(FB_cells@meta.data)
FB_cells %>% head()
```

```
combined_sct@meta.data <- merge(combined_sct@meta.data, FB_cells@meta.data[,c("cell_name", "FB_identity")], by = "cel
rownames(combined_sct@meta.data) <- combined_sct@meta.data$cell_name
combined_sct@meta.data %>% head
```

```
combined_sct@meta.data$cell_type_FB <- combined_sct@meta.data$cell_type_EC
combined_sct@meta.data$FB_identity <- as.character(combined_sct@meta.data$FB_identity)
combined_sct@meta.data$cell_type_FB[!is.na(combined_sct@meta.data$FB_identity)] <- combined_sct@meta.data$FB_identity
combined_sct@meta.data$cell_type_FB <- factor(combined_sct@meta.data$cell_type_FB)
combined_sct@meta.data %>% head
```

```r
combined_sct@meta.data$cell_type_FB  <- as.character(combined_sct@meta.data$cell_type_FB)
```

```r
# change graph size
options(repr.plot.width=24, repr.plot.height=24)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/AV_related_genes_in_FBs_and_unknown_2_normalized_data.pdf"),
    width = 10,
    height = 18
    )
dittoHeatmap(subset(combined_sct,subset =  (cell_type_FB %in% c( 'FB_0','FB_1','FB_2','FB_3','unknown_2' ))), isGene(
    annot.by = c("cell_type_FB", "condition"), complex = TRUE,
    heatmap.colors = PurpleAndYellow(50),
    # scaled.to.max=TRUE,
    # scale='none',
    use_raster=TRUE,
    heatmap.colors.max.scaled = PurpleAndYellow(50),
    main= "AV_related_genes_all in FBs and unknown_2 cells (normalized data)")
dev.off()
```

svg: 3

png: 2

```r
options(repr.plot.width=24, repr.plot.height=18)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/AV_related_genes_in_FBs_and_unknown_2_raw_counts.pdf"),
    width = 10,
    height = 18
    )

dittoHeatmap(subset(combined_sct,subset =  (cell_type_EC %in% c( 'FB_0','FB_1','FB_2','FB_3','unknown_2' ))),
```

```
            isGene(AV_related_genes_all,combined_sct, assay = "RNA",return.values = TRUE),
            assay = "RNA",
            slot = "counts",
            annot.by = c("cell_type_EC","condition" ), complex = TRUE,
            scaled.to.max = TRUE,
            scale = 'row',
            use_raster=TRUE,
            heatmap.colors = PurpleAndYellow(50),
            heatmap.colors.max.scaled = PurpleAndYellow(25),
            main = "AV_related_genes_all in FBs and unknown_2 cells (raw counts)",
    ) %>% print
dev.off ()
```

15 Ligand recepetor analysis using Liana

```
DefaultAssay(combined_sct) <- "SCT"
DefaultAssay(combined_sct)
```

```
show_homologene()
```

```
op_resource <- select_resource("Consensus")[[1]]

# Generate orthologous resource
ortholog_resource <- generate_homologs(op_resource = op_resource,
                                       target_organism = 7955)
```

```
liana_mut <- liana_wrap(subset(combined_sct,subset =  (condition %in% c( 'mut' ))), resource = 'custom', # resource h
                        external_resource = ortholog_resource)
liana_mut %>% dplyr::glimpse()
```

```r
liana_res_mut <- liana_mut %>%
  liana_aggregate()

liana_res_mut %>%
  liana_dotplot(source_groups = c("CM_Ventricle"),
                target_groups = c("EnC_Valve_cells", "CM_AV_canal"),
                ntop = 20)

liana_res_mut %>%
  liana_dotplot(source_groups = c("CM_Atrium"),
                target_groups = c("EnC_Valve_cells", "CM_AV_canal"),
                ntop = 20)
```

```r
liana_sib <- liana_wrap(subset(combined_sct,subset =  (condition %in% c( 'sib' ))), resource = 'custom', # resource h
                        external_resource = ortholog_resource)
liana_sib %>% dplyr::glimpse()
```

```r
liana_res_mut_sig <- liana_res_mut %>% filter(aggregate_rank < 0.05)
liana_res_sib_sig <- liana_res_sib %>% filter(aggregate_rank < 0.05)
liana_res_mut_sig %>% head
liana_res_sib_sig  %>% head
```

```r
# add a row to liana_res_mut_sig with source =  CM_AV_canal and target = EnC_3 so that columns are not missing in gra
liana_res_mut_sig_1 <- liana_res_mut_sig %>% add_row(source = "CM_AV_canal", target = "EnC_3",ligand.complex = "angpt
```

```r
source <- c( "CM_Ventricle", "CM_Atrium", "CM_AV_canal")
target <- c("EnC_Valve_cells", "EnC_1", "EnC_2", "EnC_3", "EnC_4")
cond <- "mut"
dev.copy(pdf, paste0("results_seurat/cell_assigned/LR_analysis/liana_", cond,"_",  paste0(source, collapse="_"), "__T
```

```
liana_res_mut_sig_1 %>%
  liana_dotplot(source_groups = source,
                target_groups = target,
                specificity = "natmi.edge_specificity",
                magnitude = "aggregate_rank",
                colour.label = "Aggregate Rank \nPval",
                ntop = 20) +
                # rotate x axis labels
                 theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))+ scale_colour_viridis(option = "pl
dev.off()

cond <- "sib"

dev.copy(pdf, paste0("results_seurat/cell_assigned/LR_analysis/liana_", cond,"_",  paste0(source, collapse="_"), "__T
liana_res_sib_sig %>%
  liana_dotplot(source_groups = source,
                target_groups = target,
                specificity = "natmi.edge_specificity",
                magnitude = "aggregate_rank",
                colour.label = "Aggregate Rank \nPval",
                ntop = 20) +
                # rotate x axis labels
                 theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))+ scale_colour_viridis(option = "pl
dev.off()
```

```
source <- c("EnC_Valve_cells", "EnC_1", "EnC_2", "EnC_3", "EnC_4")
target <- c( "CM_Ventricle", "CM_Atrium", "CM_AV_canal")
cond <- "mut"
dev.copy(pdf, paste0("results_seurat/cell_assigned/LR_analysis/liana_", cond,"_",  paste0(source, collapse="_"), "__T
liana_res_mut_sig %>%
  liana_dotplot(source_groups = source,
                target_groups = target,
```

```
                specificity = "natmi.edge_specificity",
                magnitude = "aggregate_rank",
                colour.label = "Aggregate Rank /npval",
                ntop = 20) +
                # rotate x axis labels
                 theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
dev.off()

cond <- "sib"

dev.copy(pdf, paste0("results_seurat/cell_assigned/LR_analysis/liana_", cond,"_",  paste0(source, collapse="_"), "__T
liana_res_sib_sig %>%
  liana_dotplot(source_groups = source,
                target_groups = target,
                specificity = "natmi.edge_specificity",
                magnitude = "aggregate_rank",
                colour.label = "Aggregate Rank /npval",
                ntop = 20) +
                # rotate x axis labels
                 theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust=1))
dev.off()
```

```
write.csv(liana_res_mut, file = "results_seurat/cell_assigned/LR_analysis/liana_mut.csv")
write.csv(liana_res_sib, file = "results_seurat/cell_assigned/LR_analysis/liana_sib.csv")

write.csv(liana_res_mut_sig, file = "results_seurat/cell_assigned/LR_analysis/liana_mut_sig_agg05.csv")
write.csv(liana_res_sib_sig, file = "results_seurat/cell_assigned/LR_analysis/liana_sib_sig_agg05.csv")
```

16 Other Figures for Paper

```
# in cell_type_EC change unknown_1 to "neuronal progenitors" and unkown 2 to "chondrocytes"
combined_sct@meta.data$cell_type_EC[combined_sct@meta.data$cell_type_EC== "unknown_1"] <- "Neuronal_Progenitors"
combined_sct@meta.data$cell_type_EC[combined_sct@meta.data$cell_type_EC== "unknown_2"] <- "Chondrocytes"
```

```
factor(combined_sct@meta.data$cell_type_EC, levels = sort(unique(combined_sct@meta.data$cell_type_EC))) %>% head
```

```
combined_sct@meta.data$cell_type_EC <- factor(combined_sct@meta.data$cell_type_EC, levels = sort(unique(combined_sct@
```

```
combined_sct@meta.data <- combined_sct@meta.data[match(names(Idents(combined_sct)),rownames(combined_sct@meta.data)),
combined_sct@meta.data %>% head
Idents(combined_sct)  %>% head
```

```
Idents(combined_sct)  <- "cell_type_EC"
```

```
dir.create("results_seurat/cell_assigned/figures/", showWarnings = FALSE)
dev.copy(pdf, "results_seurat/cell_assigned/figures/umap_subclusters_25072023_nolabel.pdf",
width = 20, height = 12)


# DimPlot(combined_sct, reduction = "umap",label = TRUE, repel = TRUE, label.size=6, pt.size=1
# ) + geom_text_repel( max.overlaps = 20)+NoLegend()

dittoDimPlot(object = combined_sct, var= "ident", reduction = "umap", do.label= FALSE, labels.repel= TRUE, labels.siz
dev.off ()
```

```
Idents(combined_sct) <- as.character(combined_sct@meta.data$cell_type_FB)
```

```
combined_sct@meta.data$cell_type_FB <- factor(combined_sct@meta.data$cell_type_FB, levels = c(unique(sort(combined_sc
```

16.1 Markers for ECs

```
EnC_cells <- subset(combined_sct, subset= (cell_type_EC%in% c("EnC_1","EnC_2", "EnC_3", "EnC_4", "EnC_Valve_cells")))
EnC_cells
EnC_cells@meta.data$cell_type_EC <- factor(EnC_cells@meta.data$cell_type_EC, levels = unique(sort(EnC_cells@meta.data
EnC_cells@meta.data$cell_type_EC %>% head

EnC_cells@meta.data$condition <- factor(EnC_cells@meta.data$condition, levels = c("sib", "mut"))
EnC_cells@meta.data$condition %>% head

Idents(EnC_cells)  <-  factor(EnC_cells@meta.data$cell_type_EC, levels= sort(unique(EnC_cells@meta.data$cell_type_EC)
Idents(EnC_cells) %>% head
# Idents(EnC_cells) %>% unique
```

```
Idents(EnC_cells)  <-  factor(EnC_cells@meta.data$cell_type_EC, levels= sort(unique(EnC_cells@meta.data$cell_type_FB)
Idents(EnC_cells) %>% head
```

```
options(repr.plot.width=10, repr.plot.height=8)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/EnC_cells_markers_dot_plot_magma_reordered_25072023_alphabeti
    width = 12,
    height = 8
    )
DotPlot(object = EnC_cells, features = c("id2b", "egfl7", "esama", "doc2b", "hapln1b", "ednraa", "zfpm1", "slit3", "s
#cols=c(  "blue", "red"),
dev.off()
```

16.2 Violin plot for notch1b

```
options(repr.plot.width=8, repr.plot.height=10)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/EnC_cells_Notch1b_dittovln_no_box_mut_sib_11102023.pdf"),
    width = 9,
    height = 6
    )
dittoPlot(object = EnC_cells, var = c("notch1b"), group.by = "condition", split.by = "cell_type_EC",
plots=c("vlnplot", "jitter"), split.nrow =1 )
dev.off()
```

16.3 CM markers

```
options(repr.plot.width=10, repr.plot.height=8)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/CM_cells_markers_dot_plot_new_24072023.pdf"),
    width = 12,
    height = 8
    )
DotPlot(object = EnC_CM_BAcells, features = c("alcama", "bmpr2b", "cd9a", "col1a1a", "col1a2", "desma", "doc2b", "edn
dev.off()
```

```
options(repr.plot.width=10, repr.plot.height=8)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/CM_EnC_cells_markers_dot_plot_new_24072023.pdf"),
    width = 12,
```

```
    height = 8
    )
DotPlot(object = EnC_CM_BAcells, features = c("elnb", "mylka", "nrp1a", "tgfb2", "fn1a", "notch1b", "ednraa", "smad6b
dev.off()
```

```
options(repr.plot.width=12, repr.plot.height=18)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/CM_EnC_cells_markers_violin_plot_new_24072023.pdf"),
    width = 12,
    height = 18
    )
VlnPlot(object = EnC_CM_BAcells, features = c("elnb", "mylka", "nrp1a", "tgfb2", "fn1a", "notch1b", "ednraa", "smad6b
dev.off()
```

16.4 All cell makers

```
Idents(combined_sct)  <-  factor(combined_sct@meta.data$cell_type_EC, levels= sort(unique(combined_sct@meta.data$cell
Idents(combined_sct) %>% head
```

```
options(repr.plot.width=18, repr.plot.height=12)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/All_cells_markers_dot_plot_magma_25072023_alphabetical_revers
    width = 18,
    height = 12
    )
DotPlot(object = combined_sct, features = c("col11a2", "col2a1a", "col9a1a","myh6", "smtnl1", "rgs6", "myh7", "slc25a
dev.off()
```

```
CM_cells <- subset(combined_sct, subset= (cell_type_EC%in% c( "CM_Atrium","CM_AV_canal","CM_Ventricle")))
# CM_cells
CM_cells@meta.data$cell_type_EC %>% unique
Idents(CM_cells) %>% unique
Idents(CM_cells)  <-  factor(CM_cells@meta.data$cell_type_EC, levels= sort(unique(CM_cells@meta.data$cell_type_EC, de
Idents(CM_cells)  <-  factor(CM_cells@meta.data$cell_type_EC, levels= sort(unique(CM_cells@meta.data$cell_type_EC), d
Idents(CM_cells) %>% head
```

```
options(repr.plot.width=10, repr.plot.height=8)
dev.copy(
    pdf,
    file = paste0("results_seurat/cell_assigned/figures/CM_cells_markers_dot_plot_new_magma_25072023_alphabetical_rev
    width = 10,
    height = 8
    )
DotPlot(object = CM_cells, features = c("ptpn13", "mybpc3", "zfpm1", "alcama", "nrp2b", "zfpm2b", "cd9a", "piezo2a.2"
dev.off()
```

16.5 Cell Number Graph

```
cell_type_cond_numbers <- table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$condition)
cell_type_cond_numbers <- as.data.frame(cell_type_cond_numbers)
cell_type_cond_numbers <- cell_type_cond_numbers %>% dplyr::rename(cell_type = Var1, sample_name = Var2)
cell_type_cond_numbers$sample_name <- factor(cell_type_cond_numbers$sample_name, levels = c("sib","mut"))
# cell_type_cond_numbers %>% write.csv("results_seurat/cell_type_cond_numbers_neuronal_prog.csv")
cell_type_cond_numbers
```

```
# calculate the percentage of cell in each cell type according to condition
cell_type_percentages_cond <- prop.table(table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$condition)
cell_type_percentages_cond <- as.data.frame(cell_type_percentages_cond)
```

```r
cell_type_percentages_cond %>% head
cell_type_percentages_cond <- cell_type_percentages_cond %>% dplyr::rename(cell_type = Var1, condition = Var2)
cell_type_percentages_cond$condition <- factor(cell_type_percentages_cond$condition, levels = c("sib","mut"))
```

```r
# calculate the percentage of cell in each cell type according to condition
cell_type_percentages_cond <- prop.table(table(combined_sct@meta.data$cell_type_EC, combined_sct@meta.data$condition)
cell_type_percentages_cond <- as.data.frame(cell_type_percentages_cond)
cell_type_percentages_cond %>% head
cell_type_percentages_cond <- cell_type_percentages_cond %>% dplyr::rename(cell_type = Var1, condition = Var2)
cell_type_percentages_cond$condition <- factor(cell_type_percentages_cond$condition, levels = c("sib","mut"))
cell_type_percentages_cond %>% write.csv("results_seurat/cell_type_percentages_cond_neuronal_prog.csv")
cell_type_percentages_cond %>% head
# plot these percentages using ggplot2
# save the graph as a pdf file
dev.copy(pdf, "results_seurat/cell_type_percentages_cond_normalized_neuronal_prog_pattern_flipped.pdf", width = 10, h
# ggplot(cell_type_percentages_cond, aes(x = cell_type, y = Freq, fill = condition)) +
#   geom_bar(stat = "identity", position = "fill") +
#   theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
#   labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")
ggplot(cell_type_percentages_cond, aes(x = cell_type, y = Freq, fill = cell_type, pattern=condition)) +
  scale_colour_paletteer_d("ggthemes::Tableau_20")+
  scale_fill_paletteer_d("ggthemes::Tableau_20")+
  geom_bar_pattern(
  # aes(pattern_colour= cell_type),
  colour= '#5e5d5d',
  pattern_fill= '#4d4d4d',
  pattern_alpha=0.2,
  pattern_angle= 0,
  stat = "identity", position = "fill",   pattern_spacing= 0.01) +
  scale_pattern_manual(values=c('none', 'crosshatch')) +
  theme_bw()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
```

```
  labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")+coord_flip()
  dev.off()
```

```
# calculate the percentage of cell in each cell type according to condition
# plot these percentages using ggplot2
# save the graph as a pdf file
dev.copy(pdf, "results_seurat/cell_type_percentages_cond_neuronal_prog_pattern_flipped.pdf", width = 10, height = 10)

ggplot(cell_type_percentages_cond, aes(x = cell_type, y = Freq, fill = cell_type, pattern=condition)) +
  scale_colour_paletteer_d("ggthemes::Tableau_20")+
  scale_fill_paletteer_d("ggthemes::Tableau_20")+
  geom_bar_pattern(
  # aes(pattern_colour= cell_type),
  colour= '#5e5d5d',
  pattern_fill= '#4d4d4d',
  pattern_alpha=0.2,
  pattern_angle= 0,
  stat = "identity", position = "dodge",   pattern_spacing= 0.01) +
  scale_pattern_manual(values=c('none', 'crosshatch')) +
  theme_bw()+
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0.5)) +
  labs(x = "Condition", y = "Percentage of cells", fill = "Cell type")+coord_flip()
  dev.off()
```

16.6 Sankey diagram for pathways

```
GSE_pathways_files <- list.files(path = "results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Cellular_component", pa
GSE_pathways_files
```

```
GSE_names <- gsub("results_seurat/cell_assigned/pathways_DEG_pval05_GSE/Cellular_component/", "", GSE_pathways_files)
GSE_names <- gsub("_gse_GO_CC_pathways_pval05.csv", "", GSE_names)
GSE_names
names(GSE_pathways_files) <- GSE_names
GSE_pathways_files
```

```
pathway_rbind <- purrr::map_df(GSE_pathways_files,
                               read.csv, .id = 'id')
pathway_rbind$id <- gsub('unknown_2', 'Neuronal_progentiors', pathway_rbind$id)
pathway_rbind$id <- gsub('unknown_1', 'Neurons', pathway_rbind$id)
pathway_rbind %>% head()
pathway_rbind %>% dim()
pathway_rbind %>% distinct(id)
pathway_rbind %>% distinct(Description)
pathway_rbind %>% distinct(Description) %>% as.character()
```

```
cell_type <- combined_sct@meta.data$cell_type_EC %>% unique
cell_type
```

```
lysosome_related_pathways <- c("endosome", "lysosome","lytic vacuole","vacuole","vacuolar membrane", "late endosome",
```

```
pathway_numbers_selected <-pathway_rbind %>% filter(Description %in% lysosome_related_pathways)
pathway_numbers_sankey <- pathway_numbers_selected %>%
add_row(id=cell_type[!(cell_type %in% .$id)]) %>%
make_long(id, Description)

pathway_numbers_sankey <- pathway_numbers_sankey %>% dplyr::arrange(next_node,node)

pathway_numbers_sankey$node <- factor(pathway_numbers_sankey$node, levels = c(as.character(cell_type[order(cell_type,
```

```
pathway_numbers_sankey$x <- gsub("id", "Cell Type", pathway_numbers_sankey$x)
pathway_numbers_sankey$x <- gsub("Description", "Pathway", pathway_numbers_sankey$x)

pathway_numbers_sankey <- pathway_numbers_sankey %>% filter(!is.na(node))

p <- ggplot(pathway_numbers_sankey, aes(x = x,
                next_x = next_x,
                node = node,
                next_node = next_node,
                fill=node,
                label = node
            )
                ) +
  geom_sankey(width=0.05, flow.alpha = .6, type= "sankey") +

  theme_sankey(base_size = 32) +
  theme(legend.position = "none")+
NULL
dev.copy(pdf, "results_seurat/cell_assigned/figures/pathway_sankey_graph_labels_gse_resordered_nolablels.pdf", width
print(p)
dev.off()
```

17 Save Rdata and write session info

```
save.image(file = "Myra_spns_snRNAseq_26052023_mt_removed_seurat_integration_25072023.RData")
```

```
load("Myra_spns_snRNAseq_26052023_mt_removed_seurat_integration_25072023.RData")
```

```
sessionInfo()
```

```
R version 4.3.1 (2023-06-16)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.2 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

Random number generation:
 RNG:     L'Ecuyer-CMRG
 Normal:  Inversion
 Sample:  Rejection

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=de_CH.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=de_CH.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=de_CH.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=de_CH.UTF-8 LC_IDENTIFICATION=C

time zone: Europe/Zurich
tzcode source: system (glibc)

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] viridis_0.6.3               viridisLite_0.4.2
 [3] liana_0.1.12                future_1.33.0
 [5] ggpattern_1.1.0-0           paletteer_1.5.0
 [7] ggrepel_0.9.3               dittoSeq_1.12.0
 [9] clusterProfiler_4.8.1       Libra_1.0.0
```

```
[11] repr_1.1.6                  BiocParallel_1.34.2
[13] harmony_0.1.1               Rcpp_1.0.11
[15] patchwork_1.1.2             scDblFinder_1.14.0
[17] SingleCellExperiment_1.22.0 SummarizedExperiment_1.30.2
[19] Biobase_2.60.0              GenomicRanges_1.52.0
[21] GenomeInfoDb_1.36.1         IRanges_2.34.1
[23] S4Vectors_0.38.1            BiocGenerics_0.46.0
[25] MatrixGenerics_1.12.2       matrixStats_1.0.0
[27] SeuratObject_4.1.3          Seurat_4.3.0.1
[29] lubridate_1.9.2             forcats_1.0.0
[31] stringr_1.5.0               dplyr_1.1.2
[33] purrr_1.0.1                 readr_2.1.4
[35] tidyr_1.3.0                 tibble_3.2.1
[37] ggplot2_3.4.2               tidyverse_2.0.0

loaded via a namespace (and not attached):
  [1] progress_1.2.2            goftest_1.2-3
  [3] Biostrings_2.68.1         vctrs_0.6.3
  [5] spatstat.random_3.1-5     shape_1.4.6
  [7] digest_0.6.32             png_0.1-8
  [9] OmnipathR_3.9.6           IRdisplay_1.1
 [11] deldir_1.0-9              parallelly_1.36.0
 [13] MASS_7.3-60               reshape2_1.4.4
 [15] httpuv_1.6.11             foreach_1.5.2
 [17] qvalue_2.32.0             withr_2.5.0
 [19] xfun_0.39                 ggfun_0.0.9
 [21] ellipsis_0.3.2            survival_3.5-5
 [23] memoise_2.0.1             ggbeeswarm_0.7.2
 [25] gson_0.1.0                tidytree_0.4.2
 [27] zoo_1.8-12                GlobalOptions_0.1.2
 [29] pbapply_1.7-2             IRkernel_1.3.2
 [31] prettyunits_1.1.1         rematch2_2.1.2
 [33] KEGGREST_1.40.0           promises_1.2.0.1
 [35] httr_1.4.6                downloader_0.4
```

```
 [37] restfulr_0.0.15          globals_0.16.2
 [39] fitdistrplus_1.1-11      miniUI_0.1.1.1
 [41] generics_0.1.3           DOSE_3.26.1
 [43] base64enc_0.1-3          dir.expiry_1.8.0
 [45] curl_5.0.1               zlibbioc_1.46.0
 [47] ScaledMatrix_1.8.1       ggraph_2.1.0
 [49] polyclip_1.10-4          GenomeInfoDbData_1.2.10
 [51] xtable_1.8-4             doParallel_1.0.17
 [53] evaluate_0.21            S4Arrays_1.0.4
 [55] hms_1.1.3                irlba_2.3.5.1
 [57] colorspace_2.1-0         filelock_1.0.2
 [59] ROCR_1.0-11              readxl_1.4.2
 [61] reticulate_1.30          spatstat.data_3.0-1
 [63] magrittr_2.0.3           lmtest_0.9-40
 [65] later_1.3.1              ggtree_3.8.0
 [67] lattice_0.21-8           spatstat.geom_3.2-2
 [69] future.apply_1.11.0      scattermore_1.2
 [71] XML_3.99-0.14            scuttle_1.10.1
 [73] shadowtext_0.1.2         cowplot_1.1.1
 [75] RcppAnnoy_0.0.21         pillar_1.9.0
 [77] nlme_3.1-162             iterators_1.0.14
 [79] compiler_4.3.1           beachmat_2.16.0
 [81] stringi_1.7.12           tensor_1.5
 [83] minqa_1.2.5              GenomicAlignments_1.36.0
 [85] plyr_1.8.8               crayon_1.5.2
 [87] abind_1.4-5              BiocIO_1.10.0
 [89] scater_1.28.0            blme_1.0-5
 [91] gridGraphics_0.5-1       locfit_1.5-9.8
 [93] sp_2.0-0                 graphlayouts_1.0.0
 [95] bit_4.0.5                fastmatch_1.1-3
 [97] codetools_0.2-19         BiocSingular_1.16.0
 [99] GetoptLong_1.0.5         plotly_4.10.2
[101] mime_0.12                splines_4.3.1
[103] circlize_0.4.15          basilisk_1.13.2
```

```
[105] sparseMatrixStats_1.12.2  HDO.db_0.99.1
[107] cellranger_1.1.0          knitr_1.43
[109] blob_1.2.4                utf8_1.2.3
[111] clue_0.3-64               pbdZMQ_0.3-9
[113] lme4_1.1-34               checkmate_2.2.0
[115] listenv_0.9.0             DelayedMatrixStats_1.22.1
[117] logger_0.2.2              ggplotify_0.1.0
[119] Matrix_1.5-4.1            statmod_1.5.0
[121] tzdb_0.4.0                tweenr_2.0.2
[123] pkgconfig_2.0.3           pheatmap_1.0.12
[125] tools_4.3.1               cachem_1.0.8
[127] RSQLite_2.3.1             rvest_1.0.3
[129] DBI_1.1.3                 numDeriv_2016.8-1.1
[131] rmarkdown_2.23            fastmap_1.1.1
[133] scales_1.2.1              grid_4.3.1
[135] pbmcapply_1.5.1           ica_1.0-3
[137] Rsamtools_2.16.0          RANN_2.6.1
[139] farver_2.1.1              tidygraph_1.2.3
[141] scatterpie_0.2.1          yaml_2.3.7
[143] rtracklayer_1.60.0        cli_3.6.1
[145] tester_0.1.7              leiden_0.4.3
[147] lifecycle_1.0.3           uwot_0.1.16
[149] glmmTMB_1.1.7             backports_1.4.1
[151] bluster_1.10.0            timechange_0.2.0
[153] gtable_0.3.3              rjson_0.2.21
[155] ggridges_0.5.4            progressr_0.13.0
[157] parallel_4.3.1            ape_5.7-1
[159] limma_3.56.2              jsonlite_1.8.7
[161] edgeR_3.42.4              bitops_1.0-7
[163] bit64_4.0.5               xgboost_1.7.5.1
[165] Rtsne_0.16                yulab.utils_0.0.6
[167] spatstat.utils_3.0-3      BiocNeighbors_1.18.0
[169] metapod_1.8.0             GOSemSim_2.26.0
[171] dqrng_0.3.0               lazyeval_0.2.2
```

```
[173] shiny_1.7.4               htmltools_0.5.5
[175] enrichplot_1.20.0         GO.db_3.17.0
[177] sctransform_0.3.5         rappdirs_0.3.3
[179] basilisk.utils_1.13.2     glue_1.6.2
[181] XVector_0.40.0            RCurl_1.98-1.12
[183] treeio_1.24.1             scran_1.28.1
[185] gridExtra_2.3             boot_1.3-28
[187] igraph_1.5.0              TMB_1.9.4
[189] R6_2.5.1                  DESeq2_1.40.1
[191] labeling_0.4.2            cluster_2.1.4
[193] aplot_0.1.10              nloptr_2.0.3
[195] DelayedArray_0.26.6       tidyselect_1.2.0
[197] vipor_0.4.5               xml2_1.3.4
[199] ggforce_0.4.1             AnnotationDbi_1.62.1
[201] rsvd_1.0.5                munsell_0.5.0
[203] KernSmooth_2.23-21        data.table_1.14.8
[205] htmlwidgets_1.6.2         fgsea_1.26.0
[207] ComplexHeatmap_2.16.0     RColorBrewer_1.1-3
[209] rlang_1.1.1               spatstat.sparse_3.0-2
[211] spatstat.explore_3.2-1    lmerTest_3.1-3
[213] uuid_1.1-0                fansi_1.0.4
[215] beeswarm_0.4.0
```