#these libraries are ported here in case you remove the code and place elsewhere.  
# install.packages('Signac','Seurat') # These packages often appear to be outdated/broken, so this line ensures up to date libraries before code running.   
  
  
#Load Libraries  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(Seurat)

## Attaching SeuratObject

library(patchwork)  
library(data.table)

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

library(stringr)  
library(ggplot2)  
library(ggpubr)  
  
# set working directory  
setwd("~/OneDrive - Queen Mary, University of London/QMUL/Lab/Coding/data/R/Seurat/SandLKerato")  
  
# load .rds file with pre-filtered data.   
kerato <- readRDS(file = "smalllargekera.rds")

# From https://scrnaseq-course.cog.sanger.ac.uk/website/seurat-chapter.html  
  
# find all markers of cluster 1  
#Seurat can help you find markers that define clusters via differential expression. By default, it identifes positive and negative markers of a single cluster (specified in ident.1), compared to all other cells. FindAllMarkers automates this process for all clusters, but you can also test groups of clusters vs. each other, or against all cells.  
  
#The min.pct argument requires a feature to be detected at a minimum percentage in either of the two groups of cells, and the thresh.test argument requires a feature to be differentially expressed (on average) by some amount between the two groups. You can set both of these to 0, but with a dramatic increase in time - since this will test a large number of features that are unlikely to be highly discriminatory. As another option to speed up these computations, max.cells.per.ident can be set. This will downsample each identity class to have no more cells than whatever this is set to. While there is generally going to be a loss in power, the speed increases can be significiant and the most highly differentially expressed features will likely still rise to the top.  
  
#percent\_cutoff modulates the cutoff for the FindMarkers function  
percent\_cutoff = 0.25  
  
#all cluster markers  
# cluster0.markers <- FindMarkers(kerato, ident.1 = 0, min.pct = percent\_cutoff)  
# cluster1.markers <- FindMarkers(kerato, ident.1 = 1, min.pct = percent\_cutoff)  
# cluster2.markers <- FindMarkers(kerato, ident.1 = 2, min.pct = percent\_cutoff)  
# cluster3.markers <- FindMarkers(kerato, ident.1 = 3, min.pct = percent\_cutoff)  
# head(cluster1.markers, n = 5)  
#cluster0.markers <- FindMarkers(kerato, ident.1 = 1, min.pct = 0.25)  
#cluster2.markers <- FindMarkers(kerato, ident.1 = 1, min.pct = 0.25)  
#need to check ident.1 identity - Ryan   
#write.xlsx(cluster0.markers,"cluster0\_markers.xlsx")  
#write.xlsx(cluster1.markers,"cluster1\_markers.xlsx")  
#write.xlsx(cluster2.markers,"cluster2\_markers.xlsx")  
#write.xlsx(cluster3.markers,"cluster3\_markers.xlsx")

# find markers for every cluster compared to all remaining cells, report only the positive ones  
kerato.markers <- FindAllMarkers(kerato, only.pos = FALSE, min.pct = 0.25, logfc.threshold = 0.25)

## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

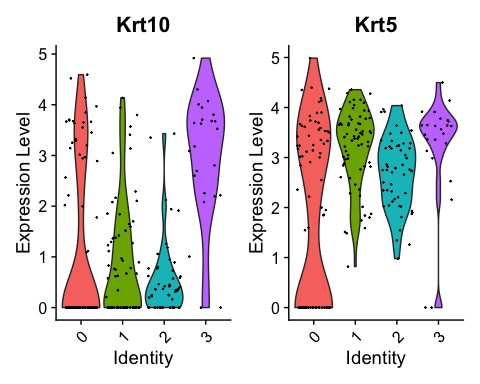
kerato.markers %>% group\_by(cluster) %>% top\_n(n = 2, wt = avg\_log2FC)

## Registered S3 method overwritten by 'cli':  
## method from   
## print.boxx spatstat.geom

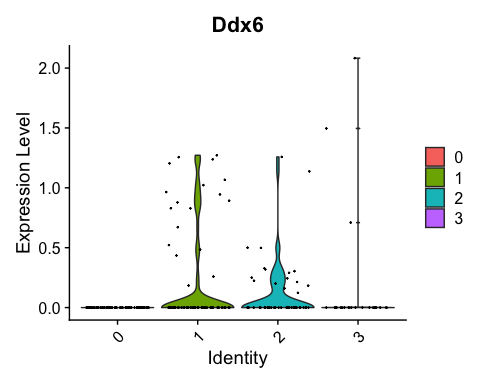
## # A tibble: 8 × 7  
## # Groups: cluster [4]  
## p\_val avg\_log2FC pct.1 pct.2 p\_val\_adj cluster gene   
## <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <chr>   
## 1 2.03e- 4 1.44 0.134 0.441 1 e+ 0 0 Sri   
## 2 2.99e- 4 1.28 0.085 0.331 1 e+ 0 0 LOC100910200  
## 3 2.61e-28 2.66 1 0.761 2.62e-24 1 Mt-co3   
## 4 5.37e-26 2.63 0.985 0.44 5.39e-22 1 Snhg11   
## 5 6.54e-20 2.15 1 0.552 6.55e-16 2 Ftl1   
## 6 1.72e-17 1.49 1 0.948 1.73e-13 2 LOC100364435  
## 7 7.90e- 9 2.07 0.917 0.517 7.92e- 5 3 Krt10   
## 8 6.03e- 8 2.04 0.833 0.448 6.04e- 4 3 Sprr1a

cluster\_markers <- kerato.markers %>% group\_by(cluster) %>% filter(!between(avg\_log2FC, -0.6, 0.6) & p\_val\_adj < 0.05)  
  
  
top\_markers <- kerato.markers %>% group\_by(cluster) %>% filter(!between(avg\_log2FC, -1, 1) & p\_val\_adj < 0.05)  
  
#write out markers to excel file. Cosnider using CSV as these are easier to manipulate.  
library(openxlsx)  
#write.xlsx(top\_markers,"allcluster\_markers.xlsx")  
# https://scrnaseq-course.cog.sanger.ac.uk/website/seurat-chapter.html  
# has a tutorial explaining these things

# We include several tools for visualizing marker expression. VlnPlot (shows expression probability distributions across clusters), and FeaturePlot (visualizes feature expression on a tSNE or PCA plot) are our most commonly used visualizations. We also suggest exploring RidgePlot, CellScatter, and DotPlot as additional methods to view your dataset. Uncomment below with genes in keratinocytes to use VlnPlot  
VlnPlot(kerato, features = c("Krt10", "Krt5"))



VlnPlot(kerato, features = c("Ddx6"))



# you can plot raw counts as well  
# VlnPlot(pbmc, features = c("NKG7", "PF4"), slot = "counts", log = TRUE)

#build finding gene in: VariableFeatures(kerato)  
# genelist <- kerato.data   
# genelist <- as.data.frame(as.matrix(genelist))  
# setDT(genelist, keep.rownames = TRUE)  
# genelist <- genelist[ ,1]  
#   
# in\_genes <- c()  
#in\_genes makes a list of genes and

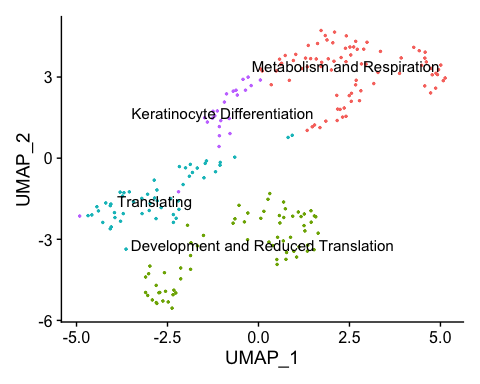
#searching for gene in gene list  
# at current this only saves the list from the R session. Must note down all genes somewhere else for now. Get it to save a file.   
  
  
# pat <- 'Sprr1a'  
#   
# if (str\_detect(genelist, pat) == TRUE) {  
# in\_genes <- c(in\_genes, pat)  
# }  
#   
# in\_genes

#we can use dot plot to show the amount of expression, and average expression of the genes within the clusters.  
# https://rpubs.com/kshekhar/349874  
# DotPlot(object = kerato, features = "percent.mt" , cols = c("purple", "yellow"))

#reading in the correct dimensions for umap pngs.   
#setwd("~/OneDrive - Queen Mary, University of London/QMUL/Lab/Coding/data/R/Seurat/SandLKerato/images")  
# library(png)  
# img <- readPNG("idealsize.png")  
# idealimg\_dims <- dim(img)

# #attempting to automate printing our heatmap pngs. More ggplot2 knowledge will be required to sort formatting.   
# setwd("~/OneDrive - Queen Mary, University of London/QMUL/Lab/Coding/data/R/Seurat/SandLKerato/images")  
# list\_of\_features <- c("Krt10", "Krt5", "Evpl", "Dsp")  
#   
# for (gene in list\_of\_features){  
#   
# p <- (FeaturePlot(kerato, features = (gene)))  
# p <- lapply(X = p, FUN = function(x) x + theme(plot.title = element\_text(size = 5)))  
# png(filename = sprintf("%s\_featureplot.png", gene), width = idealimg\_dims[2], height = idealimg\_dims[1])  
# print(p)  
#   
# dev.off()  
# }

#set working directory  
setwd("~/OneDrive - Queen Mary, University of London/QMUL/Lab/Coding/data/R/Seurat/SandLKerato")  
  
#assign cluster IDs - two separate statements depending on what you want, numerics or character labels  
new.cluster.ids <- c("Metabolism and Respiration", "Development and Reduced Translation", "Translating", "Keratinocyte Differentiation")  
  
#new.cluster.ids <- c(0,1,2,3)  
names(new.cluster.ids) <- levels(kerato)  
kerato <- RenameIdents(kerato, new.cluster.ids)  
DimPlot(kerato, reduction = "umap", label = TRUE, pt.size = 0.5) + NoLegend()



# number and proportion of cells in each cluster  
# from https://satijalab.org/seurat/v3.0/interaction\_vignette.html  
table(Idents(kerato))

##   
## Metabolism and Respiration Development and Reduced Translation   
## 82 68   
## Translating Keratinocyte Differentiation   
## 53 24

prop.table(table(Idents(kerato)))

##   
## Metabolism and Respiration Development and Reduced Translation   
## 0.3612335 0.2995595   
## Translating Keratinocyte Differentiation   
## 0.2334802 0.1057269

#if we want to subset this analysis based off of this data, we can do so with this tutorial: https://satijalab.org/seurat/v3.0/interaction\_vignette.html  
  
# WhichCells(pbmc, idents = "NK")  
  
# How can I extract expression matrix for all NK cells (perhaps, to load into another package)  
# nk.raw.data <- as.matrix(GetAssayData(pbmc, slot = "counts")[, WhichCells(pbmc, ident = "NK")])  
#   
# # Can I create a Seurat object based on expression of a feature or value in object metadata?  
# subset(pbmc, subset = MS4A1 > 1)  
  
# Can I create a Seurat object of all cells except the NK cells and B cells?  
# subset(pbmc, idents = c("NK", "B"), invert = TRUE)

# Monocle Analysis   
# sajita lab have a tutorial on using monocle from seurat objects  
# https://satijalab.org/signac/articles/monocle.html  
  
setwd("~/OneDrive - Queen Mary, University of London/QMUL/Lab/Coding/data/R/Seurat/SandLKerato")  
  
#loading libraries  
library(Signac)  
library(Seurat)  
library(SeuratWrappers)  
library(monocle3)

## Loading required package: Biobase

## Loading required package: BiocGenerics

## Loading required package: parallel

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':  
##   
## clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,  
## clusterExport, clusterMap, parApply, parCapply, parLapply,  
## parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:dplyr':  
##   
## combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':  
##   
## IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':  
##   
## anyDuplicated, append, as.data.frame, basename, cbind, colnames,  
## dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,  
## grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,  
## order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,  
## rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,  
## union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor  
##   
## Vignettes contain introductory material; view with  
## 'browseVignettes()'. To cite Bioconductor, see  
## 'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: SingleCellExperiment

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##   
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':  
##   
## anyMissing, rowMedians

## The following object is masked from 'package:dplyr':  
##   
## count

##   
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':  
##   
## colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,  
## colCounts, colCummaxs, colCummins, colCumprods, colCumsums,  
## colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,  
## colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,  
## colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,  
## colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,  
## colWeightedMeans, colWeightedMedians, colWeightedSds,  
## colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,  
## rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,  
## rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,  
## rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,  
## rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,  
## rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,  
## rowWeightedMads, rowWeightedMeans, rowWeightedMedians,  
## rowWeightedSds, rowWeightedVars

## The following object is masked from 'package:Biobase':  
##   
## rowMedians

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: S4Vectors

##   
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:data.table':  
##   
## first, second

## The following objects are masked from 'package:dplyr':  
##   
## first, rename

## The following object is masked from 'package:base':  
##   
## expand.grid

## Loading required package: IRanges

##   
## Attaching package: 'IRanges'

## The following object is masked from 'package:data.table':  
##   
## shift

## The following objects are masked from 'package:dplyr':  
##   
## collapse, desc, slice

## Loading required package: GenomeInfoDb

##   
## Attaching package: 'SummarizedExperiment'

## The following object is masked from 'package:SeuratObject':  
##   
## Assays

## The following object is masked from 'package:Seurat':  
##   
## Assays

##   
## Attaching package: 'monocle3'

## The following objects are masked from 'package:Biobase':  
##   
## exprs, fData, fData<-, pData, pData<-

library(Matrix)

##   
## Attaching package: 'Matrix'

## The following object is masked from 'package:S4Vectors':  
##   
## expand

library(ggplot2)  
library(patchwork)  
library(tidyseurat)

## ========================================  
## tidyseurat version 0.2.7  
## To restore the Seurat default display use options("restore\_Seurat\_show" = TRUE)   
## ========================================

##   
## Attaching package: 'tidyseurat'

## The following object is masked from 'package:IRanges':  
##   
## slice

## The following object is masked from 'package:S4Vectors':  
##   
## rename

## The following object is masked from 'package:matrixStats':  
##   
## count

## The following objects are masked from 'package:dplyr':  
##   
## add\_count, bind\_cols, bind\_rows, count

## The following object is masked from 'package:stats':  
##   
## filter

set.seed(1234)

# this piece of code may let us subset the data, lifted from tutorial  
# erythroid <- bone[, bone$assigned\_celltype %in% c("HSC", "MEP", "CMP-BMP")]  
# kerato cluster IDs: c("Metabolism and Respiration", "Development and Reduced Translation", "Translating", "Keratinocyte Differentiation")  
  
# this will select the subset of the clusters using their ID number  
# kerato[, kerato@meta.data[["seurat\_clusters"]] %in% c("1", "2","3")]  
# just make a new variable and run it through the analysis. Alternatively, subset the seurat object, recluster and go from there.   
kerato <- readRDS(file = "smalllargekera.rds")  
kerato.cds <- as.cell\_data\_set(kerato)

## Warning: Monocle 3 trajectories require cluster partitions, which Seurat does  
## not calculate. Please run 'cluster\_cells' on your cell\_data\_set object

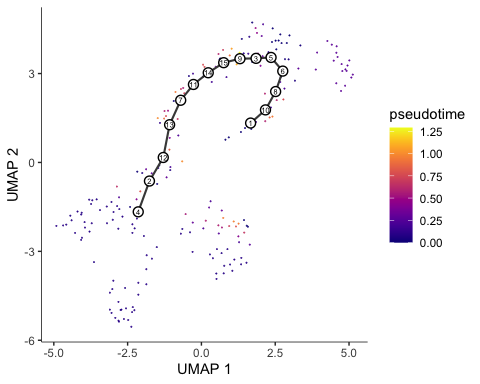
kerato.cds <- cluster\_cells(cds = kerato.cds, reduction\_method = "UMAP")  
kerato.cds <- learn\_graph(kerato.cds, use\_partition = TRUE)

## | | | 0% | |======================================================================| 100%

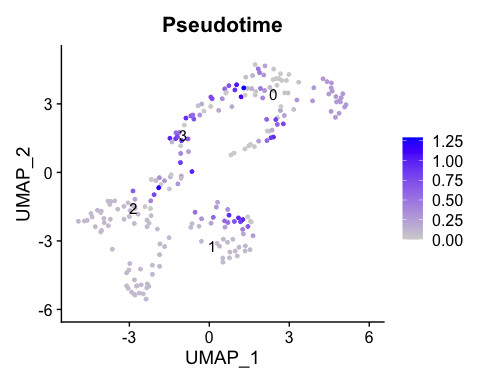
# order cells  
root\_group = colnames(kerato.cds)[clusters(kerato.cds) == 1]  
  
kerato.cds <- order\_cells(kerato.cds, reduction\_method = "UMAP", root\_cells = root\_group)  
  
# plot trajectories colored by pseudotime  
plot\_cells(  
 cds = kerato.cds,  
 color\_cells\_by = "pseudotime",  
 show\_trajectory\_graph = TRUE  
)

## Warning: `select\_()` was deprecated in dplyr 0.7.0.  
## Please use `select()` instead.

## Cells aren't colored in a way that allows them to be grouped.



# add monocle pseudotime data to seurat metadata  
  
kerato <- AddMetaData(  
 object = kerato,  
 metadata = kerato.cds@principal\_graph\_aux@listData$UMAP$pseudotime,  
 col.name = "Pseudotime"  
)  
  
saveRDS(kerato, file = "pseudoSLkera.rds")  
  
FeaturePlot(kerato, features = "Pseudotime", label = TRUE)



#FeaturePlot(kerato, features = "Pseudotime", cols = c("purple", "yellow"))   
#FeaturePlot(kerato, features = "Pseudotime") & scale\_color\_viridis\_c()

#internal monocle function will print citations used.   
get\_citations(kerato.cds)

## Your analysis used methods from the following recent work. Please cite them wherever you are presenting your analyses.

## method  
## 1 Monocle  
## 2 Monocle  
## 3 Monocle  
## 4 leiden  
## 5 clustering  
## 6 paritioning  
## citations  
## 1 Trapnell C. et. al. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. Nat. Biotechnol. 32, 381-386 (2014). https://doi.org/10.1038/nbt.2859  
## 2 Qiu, X. et. al. Reversed graph embedding resolves complex single-cell trajectories. Nat. Methods 14, 979-982 (2017). https://doi.org/10.1038/nmeth.4402  
## 3 Cao, J. et. al. The single-cell transcriptional landscape of mammalian organogenesis. Nature 566, 496-502 (2019). https://doi.org/10.1038/s41586-019-0969-x  
## 4 Traag, V.A., Waltman, L. & van Eck, N.J. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reportsvolume 9, Article number: 5233 (2019). https://doi.org/10.1038/s41598-019-41695-z  
## 5 Levine, J. H. et. al. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. Cell 162, 184-197 (2015). https://doi.org/10.1016/j.cell.2015.05.047  
## 6 Levine, J. H., et. al. Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. Cell 162, 184-197 (2015). https://doi.org/10.1016/j.cell.2015.05.047