R Notebook

# Functions and information taken from these two sources.   
## https://scrnaseq-course.cog.sanger.ac.uk/website/biological-analysis.html#pseudotime-analysis  
# https://stemangiola.github.io/tidyseurat/  
  
  
#Load in .rds object containing filtered and clustered data.  
kerato <- readRDS(file = "pseudoSLkera.rds")  
  
#load all 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(Signac)  
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(tidyseurat)

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

## ========================================  
## 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

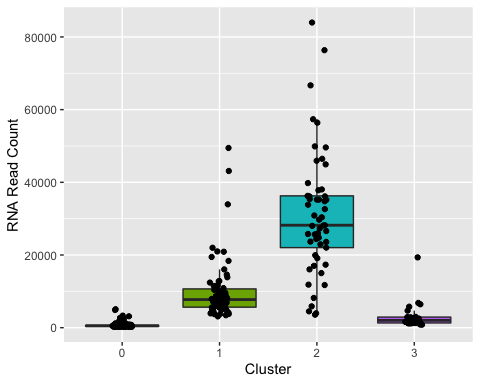
## The following object is masked from 'package:stats':  
##   
## filter

library(DEGreport)  
set.seed(1234)  
  
#working directory, change if using on system other than authors to folder with.rds file in.   
setwd("~/OneDrive - Queen Mary, University of London/QMUL/Lab/Coding/data/R/Seurat/SandLKerato")

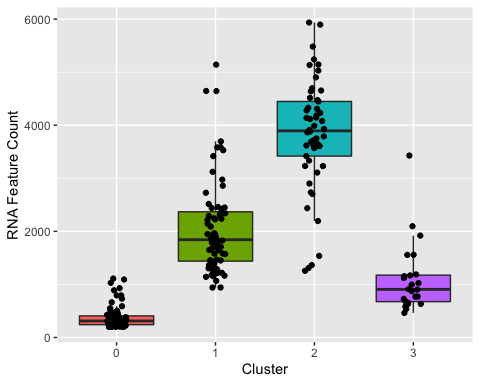
# this section builds lists containing gene IDs. These IDs are sorted into lists related to their function and/or their research area associated with them.   
  
# kerato\_marker\_genes contain general keratinocyte differentiation markers.  
kerato\_marker\_genes <- c("Krt10", "Krt5", "Krt14", "Sprr1a", "Sprr1b", "Evpl", "Dsp", "Ppl", "Klk11", "Spink5", "Klf4", "Klk6", "Klk7", "Klk8", "Klk10")  
  
# cell\_state\_markers contains genes used to mark the cell cycle state of cells, in this case currently proliferative markers.   
cell\_state\_markers <- c("Mki67","Pcna")  
  
# actin\_markers contains all genes associated with the actin cytoskeleton.   
actin\_markers <- c("Arhgdia","Camsap2", "Anxa1", "Rflnb", "Tmsb4x", "Arpc1b", "Sbsn","Dsg3", "Ahnak", "Cdh1", "Col3a1", "Tpt1", "Lgals7", "Rack1", "S100a11", "Eef1b2", "Ran", "Sprr1a", "Klf8", "Ptprf")  
  
# plmn\_markers has putative lamin bodies markers investigated.   
plmn\_markers <- c("Ipo7", "Ran", "Snupn", "Matr3", "Nup153")  
  
# SGPB\_markers contains genes associated with condensates, primarly stress granules and P bodies.  
SGPB\_markers <- c("Ddx6", "Eif4e", "Nxf1", "Lsm14a", "Caprin1", "Csde1", "Pum1", "Zfp36")  
  
# other\_markers contains miscellaneous markers, notes below variable show reasoning.   
other\_markers <- c("Snhg11", "Fabp5", "Slc25a4", "Lgals7", "Tacstd2", "Fosb", "Cd44", "Trpv4", "Trpm4" )  
  
#Tr genes are calcium channels found specifically in UGL Matsui 2021  
#kerato market genes not found  
#("Klk5"))("Klk2"))("Klk3"))("Klk4"))("Klk9"))  
# ("Crnn")) Cornulin ("Sprr2a")) not found ("Sprr2b")) ("Sprr2c"))  
# ("Sprr2d")) ("Sprr2e")) ("Sprr2f")) ("Sprr2g")) ("Sprr3")) ("Sprr4"))  
# ("AABR07012329.1")) ("Lor")) ("Flg")) ("Rgpd1")) ("Syne4")) Dcp2,   
  
#dsg1, dsp - spinous,   
# spink5, granular  
  
#Dsp, Ppl, Evpl  
  
#combines all above lists into one superlist 'list\_of\_features', which is then parsed to later image functions.   
list\_of\_features <- c(kerato\_marker\_genes, cell\_state\_markers, actin\_markers, plmn\_markers, SGPB\_markers, other\_markers)

# will make variable 'tidykerato', a form of object that can be easily parsed to data analysis, similar to tibble.   
# tidykerato <- tidyseurat::kerato

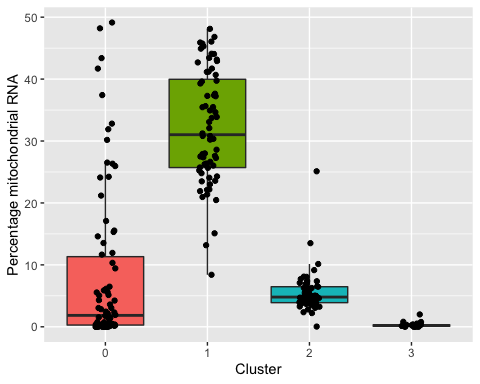
# using pipes to push kerato QC data into boxplots. The three major QC covariates are plotted against frequency over the clusters.   
  
# Number of counts of RNA per barcode.   
kerato %>%  
 tidyseurat::ggplot(aes(seurat\_clusters, nCount\_RNA, fill = seurat\_clusters)) + # using ggplot2 aesthetics to change labels and add jitter.   
 geom\_boxplot(outlier.shape = NA) +  
 geom\_jitter(width = 0.1) +  
 xlab('Cluster') +  
 ylab('RNA Read Count') +  
 labs(fill = "Cluster") + theme(legend.position="none")



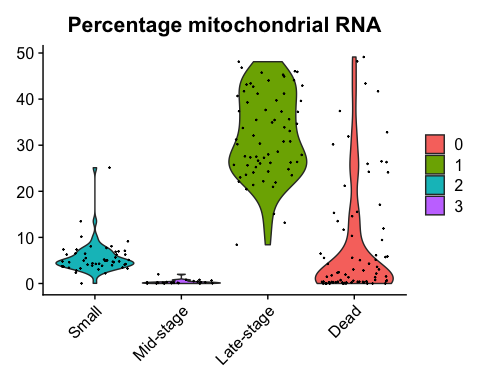
# Number of features per barcode.   
kerato %>%  
 tidyseurat::ggplot(aes(seurat\_clusters, nFeature\_RNA, fill = seurat\_clusters)) +  
 geom\_boxplot(outlier.shape = NA) + # using ggplot2 aesthetics to change labels and add jitter.   
 geom\_jitter(width = 0.1) +  
 xlab('Cluster') +  
 ylab('RNA Feature Count') +  
 labs(fill = "Cluster") + theme(legend.position="none")



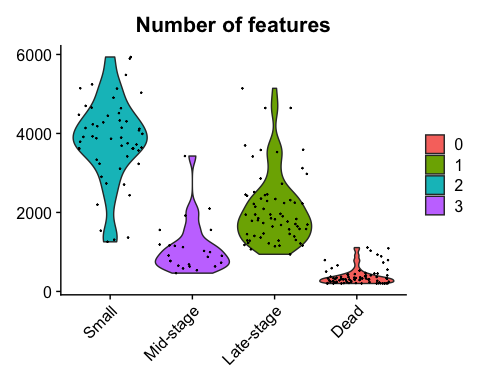
# Percentage mitochondrial RNA content per barcode.   
kerato %>%  
 tidyseurat::ggplot(aes(seurat\_clusters, percent.mt, fill = seurat\_clusters)) +  
 geom\_boxplot(outlier.shape = NA) + # using ggplot2 aesthetics to change labels and add jitter.   
 geom\_jitter(width = 0.1) +  
 xlab('Cluster') +  
 ylab('Percentage mitochondrial RNA') +  
 labs(fill = "Cluster") + theme(legend.position="none")



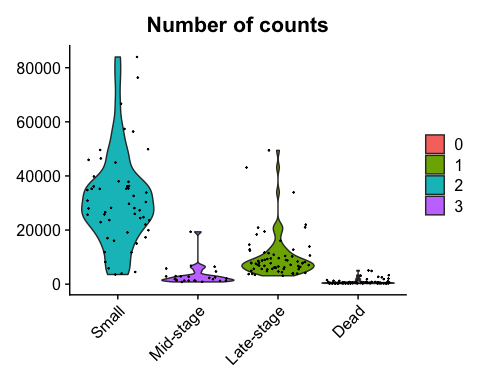
VlnPlot(kerato, features = c("percent.mt")) + scale\_x\_discrete(name ="Cluster", limits=c("2","3","1", "0"), labels=c("Small", "Mid-stage","Late-stage", "Dead")) + ggtitle("Percentage mitochondrial RNA") + theme(axis.title.x = element\_blank())



VlnPlot(kerato, features = c("nFeature\_RNA")) + scale\_x\_discrete(name ="Cluster", limits=c("2","3","1", "0"), labels=c("Small", "Mid-stage","Late-stage", "Dead")) + ggtitle("Number of features") + theme(axis.title.x = element\_blank())



VlnPlot(kerato, features = c("nCount\_RNA")) + scale\_x\_discrete(name ="Cluster", limits=c("2","3","1", "0"), labels=c("Small", "Mid-stage","Late-stage", "Dead")) + ggtitle("Number of counts") + theme(axis.title.x = element\_blank())



# Chunk printing QC metrics to console, allows extraction and analysis of statistics.   
  
cat('counts')

## counts

summary(kerato$nCount\_RNA)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 245 608 4710 10898 14239 83967

cat('mito')

## mito

summary(kerato$percent.mt)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0000 0.7492 5.5753 13.9546 26.2964 49.1361

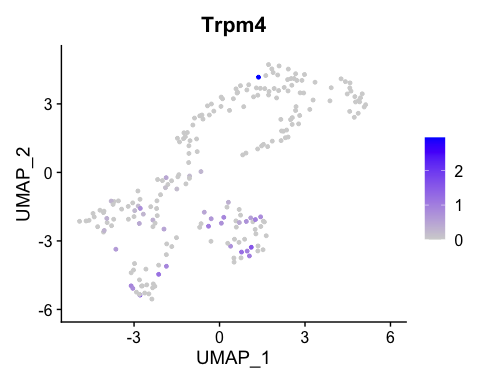
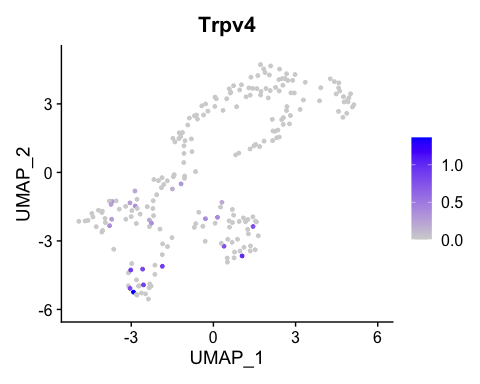
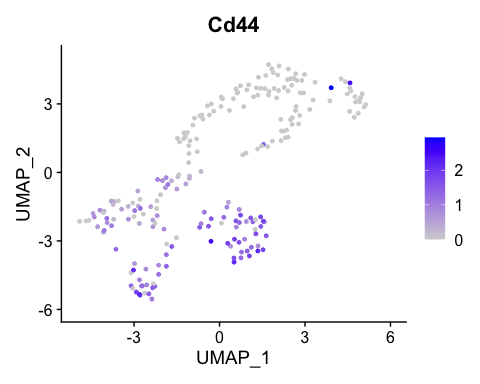
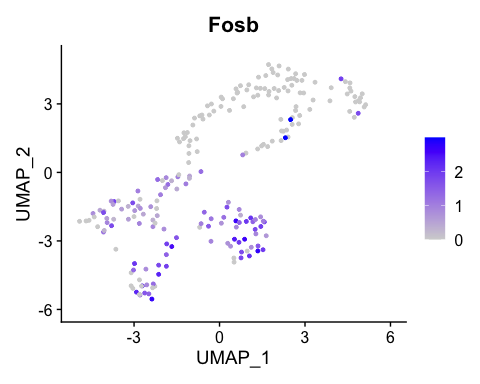
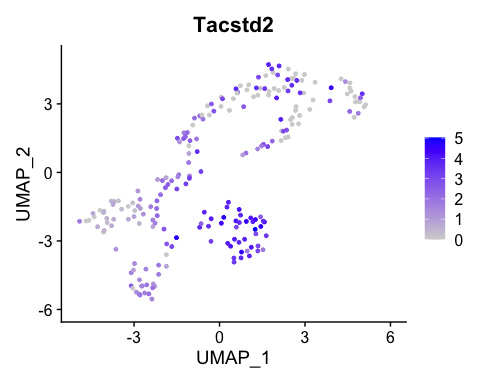
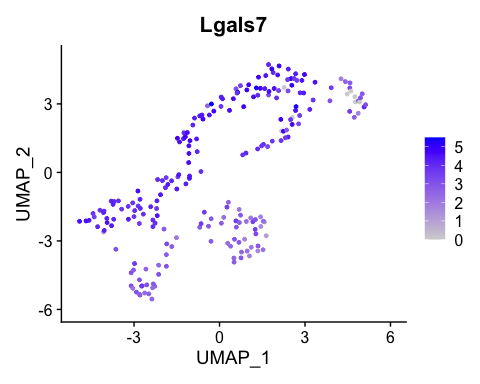
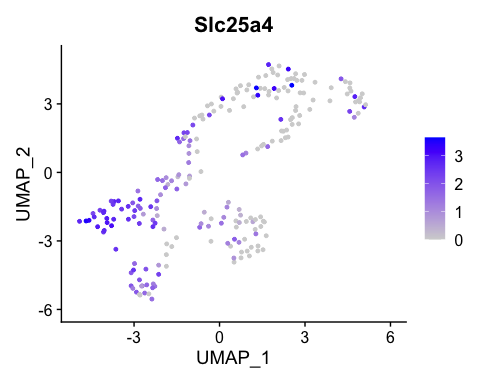
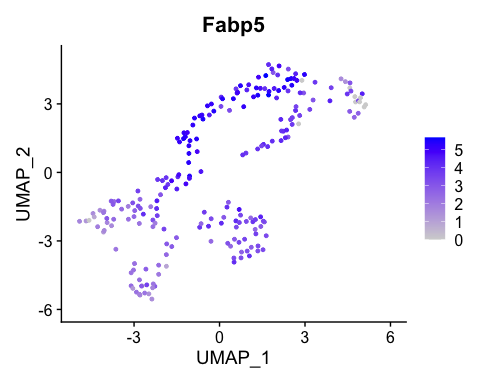
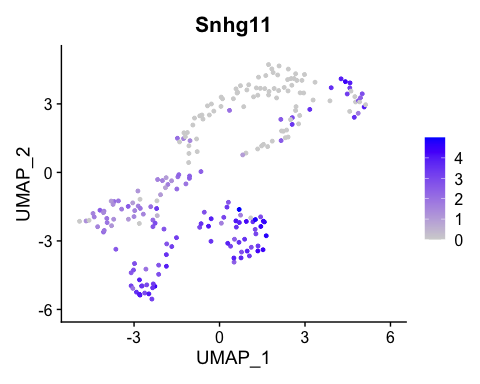
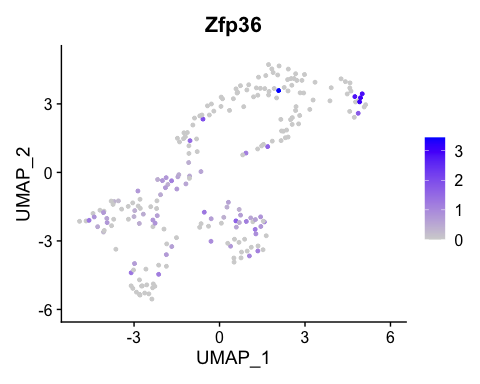
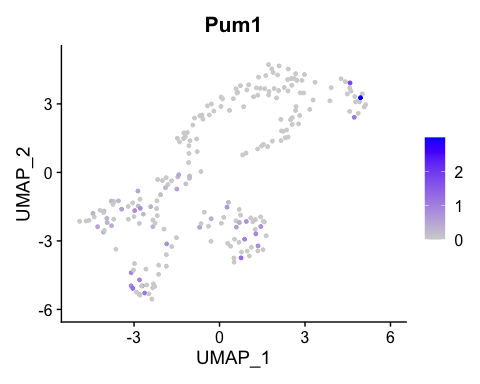
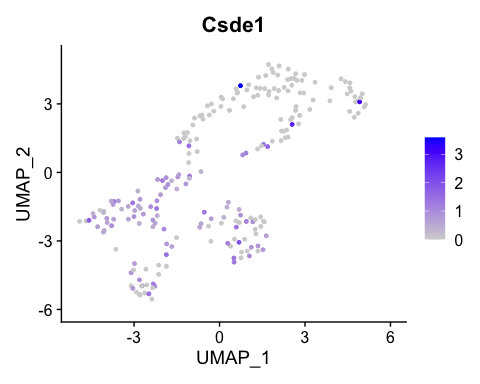
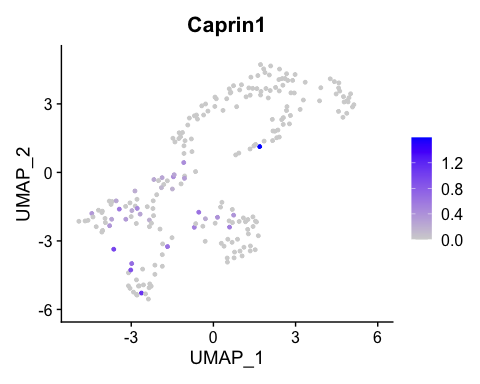
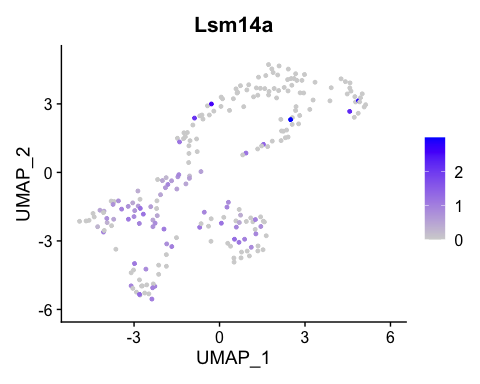
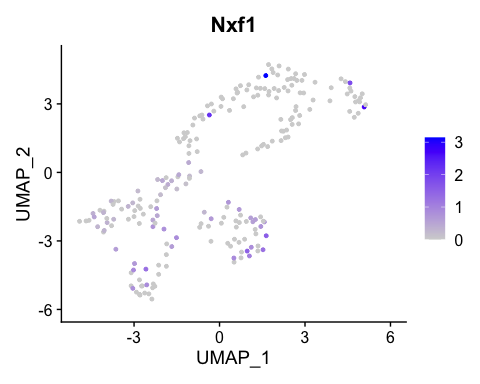
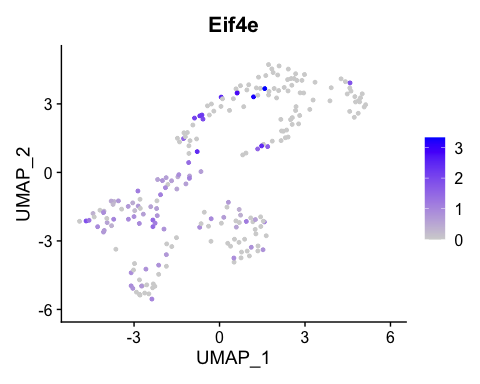
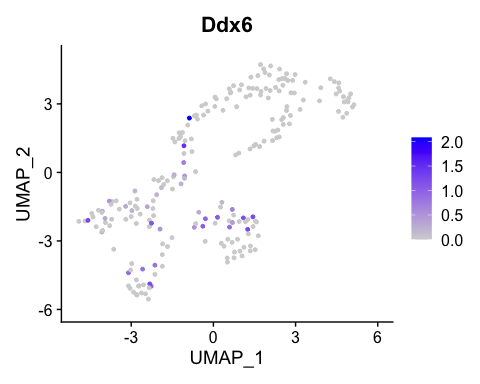
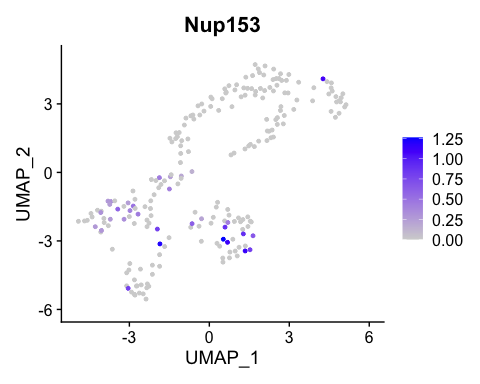
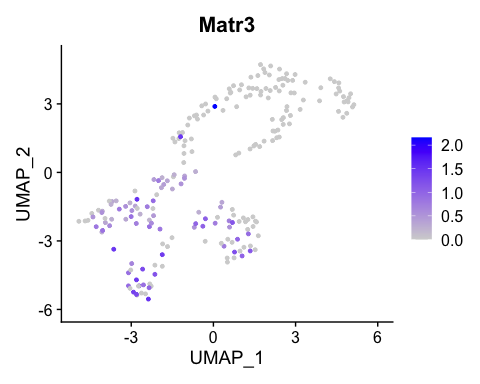
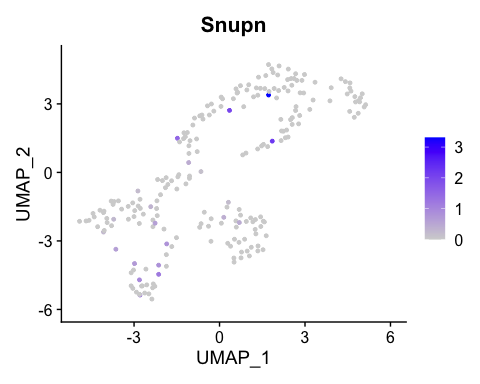
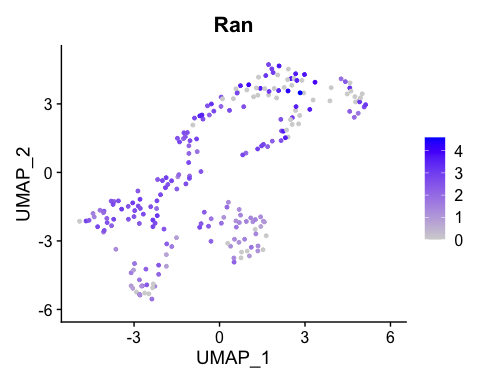
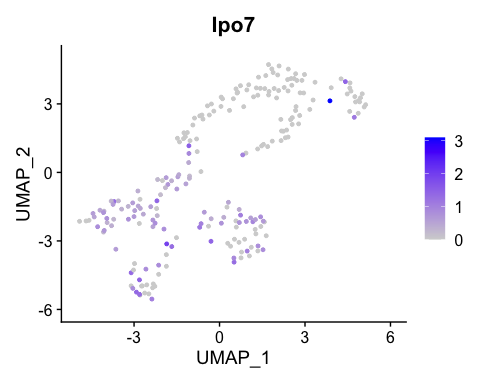
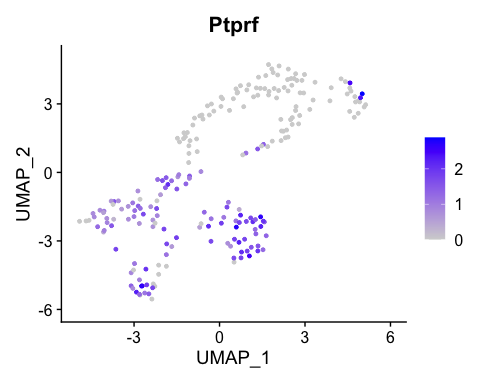
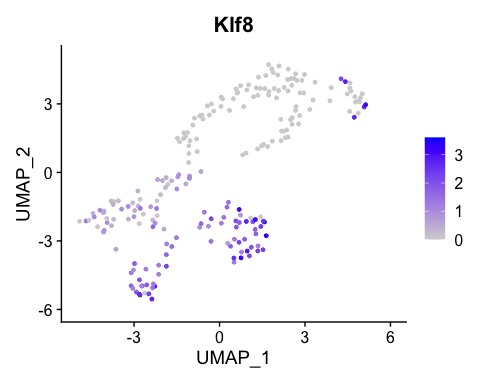
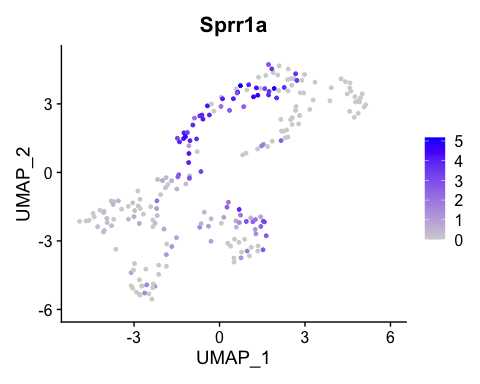
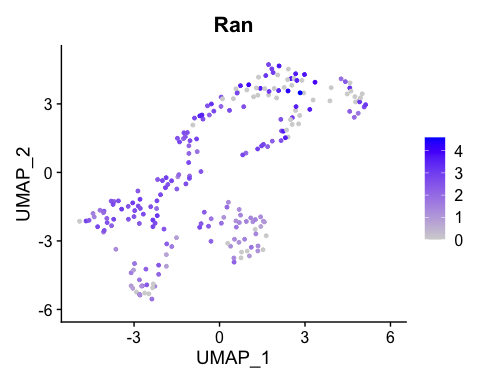
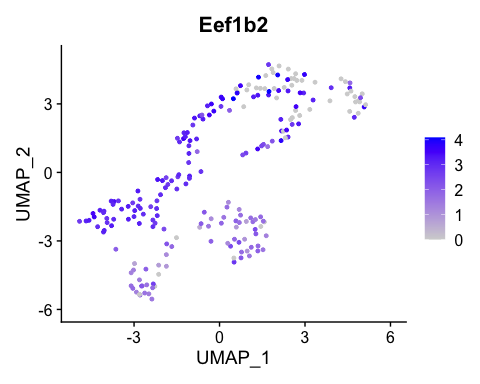
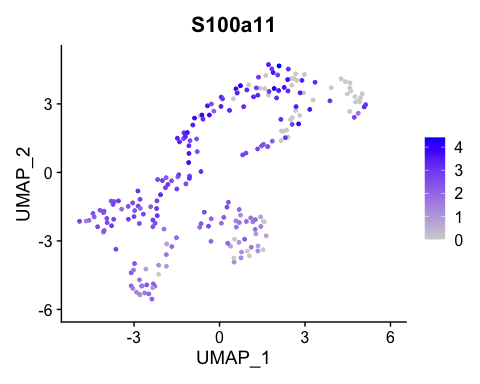
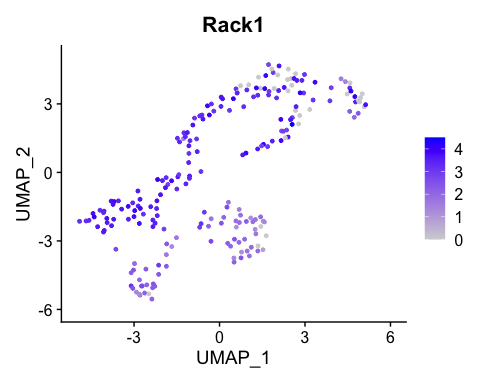
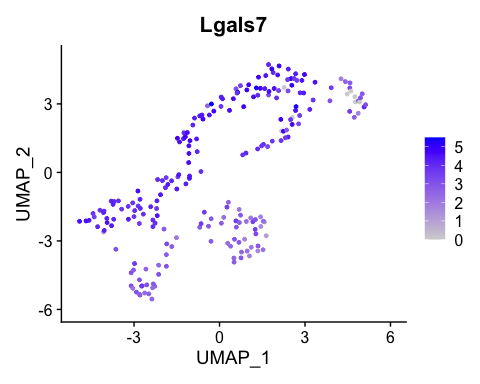
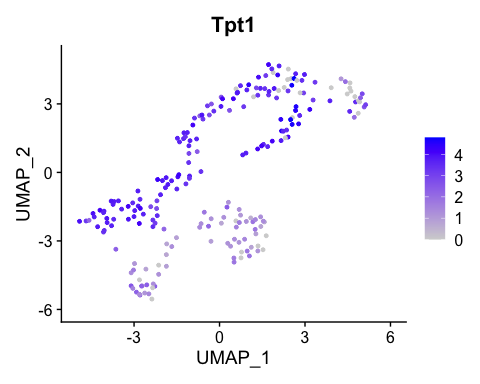
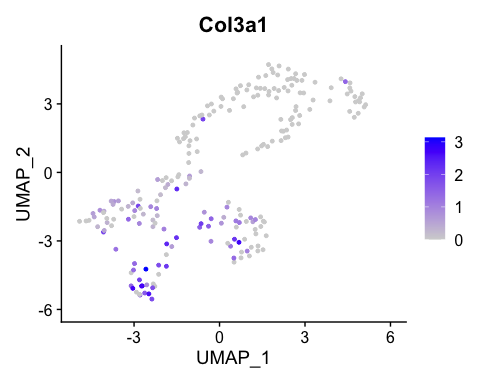
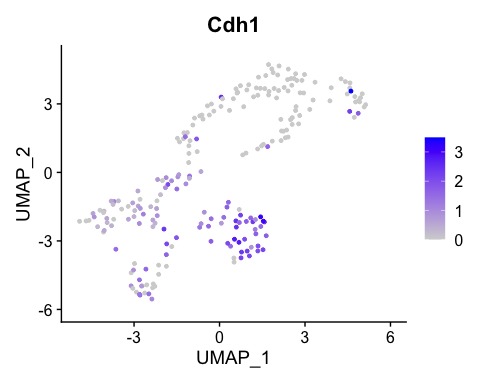
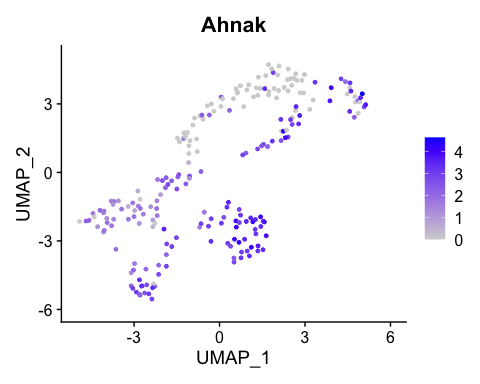
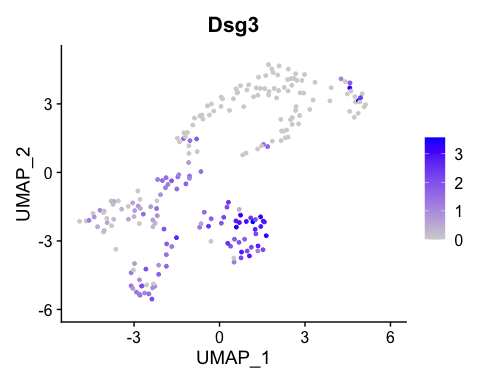
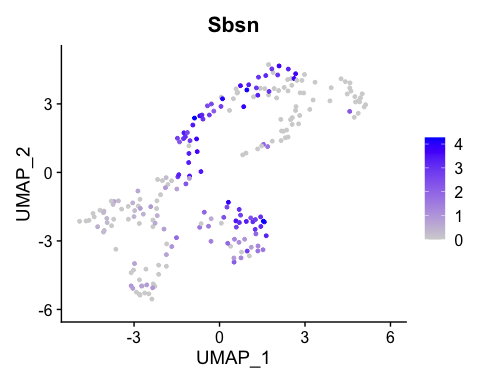
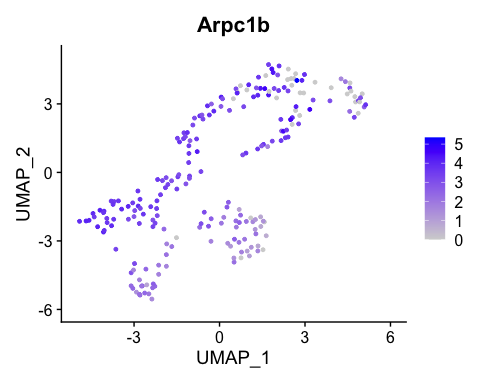
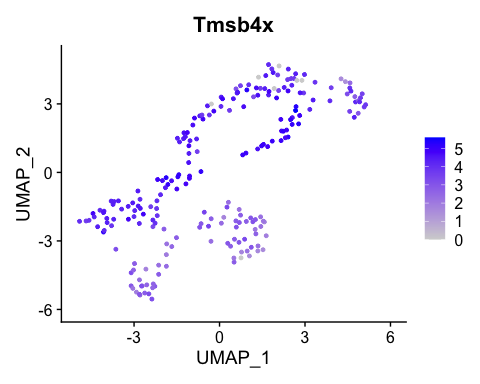
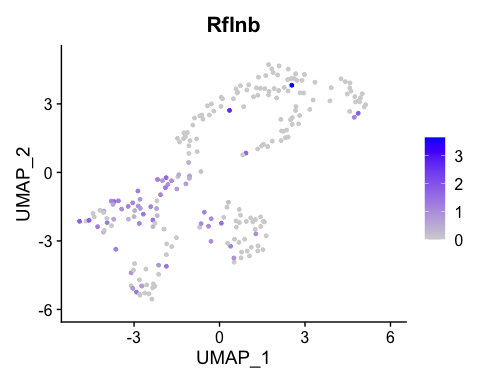
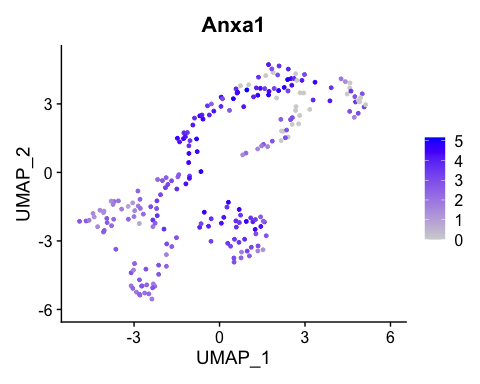
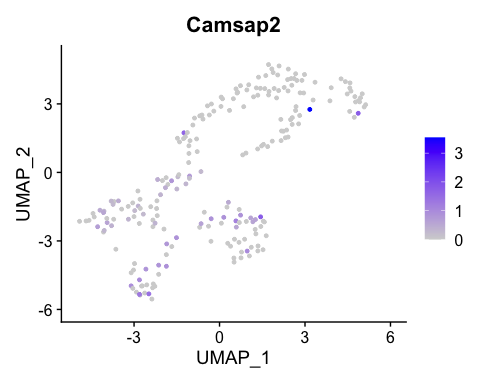
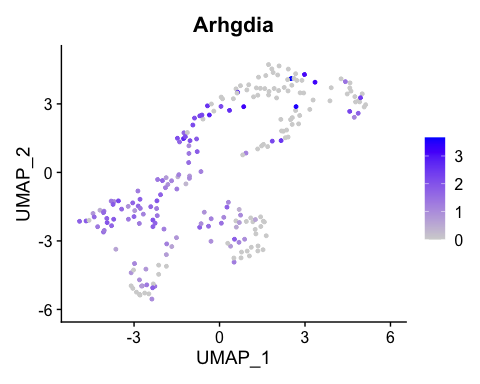
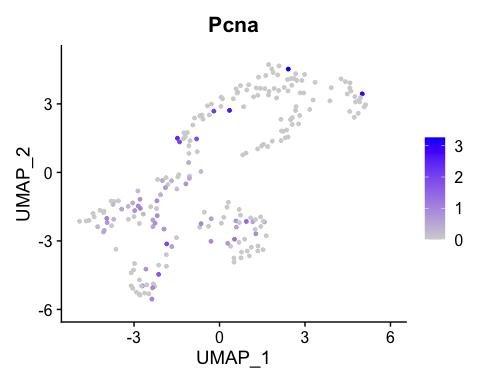
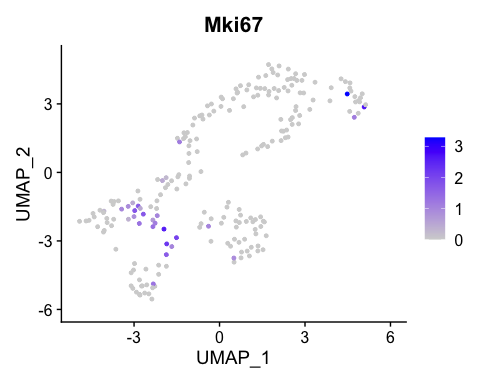
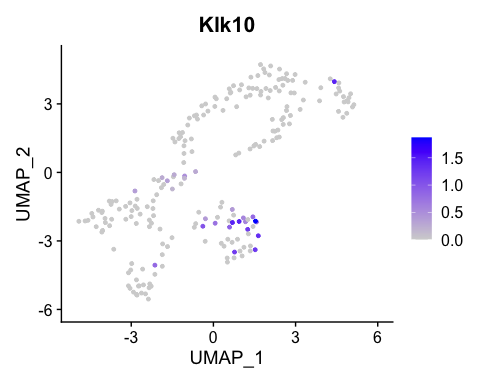
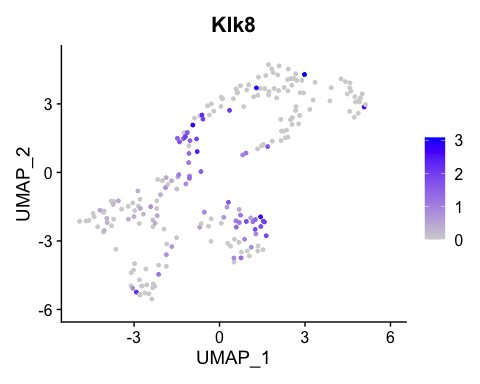
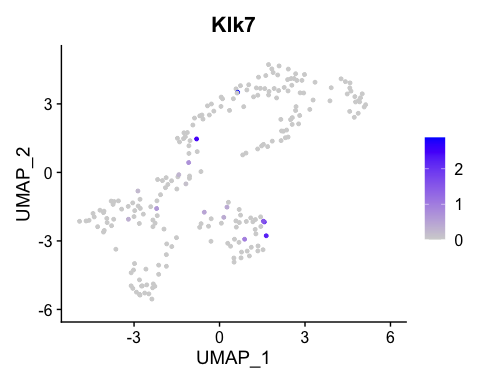
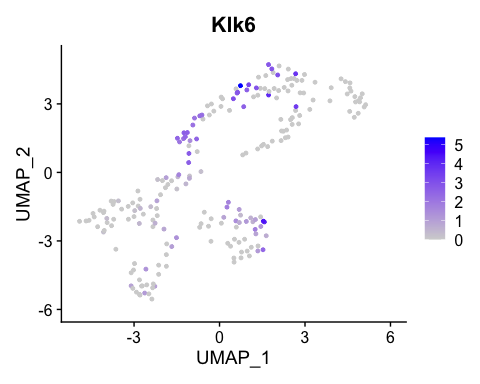
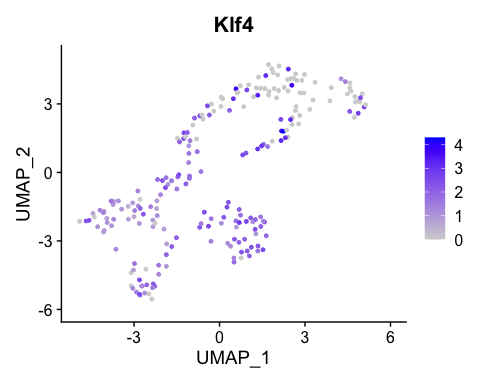
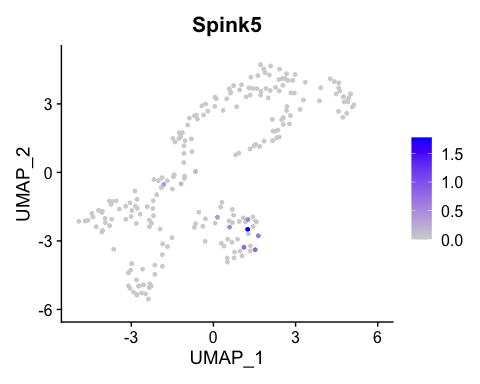
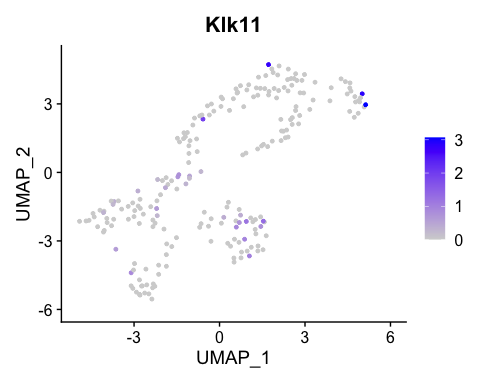
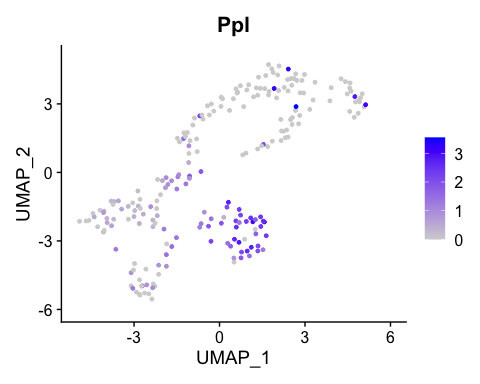
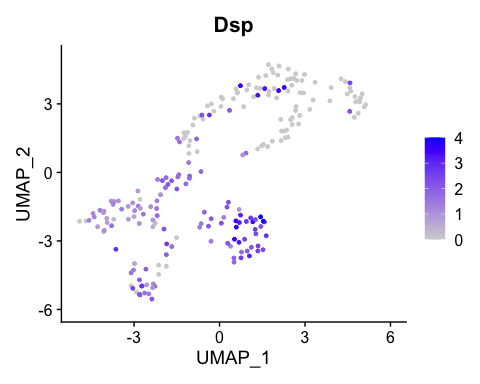
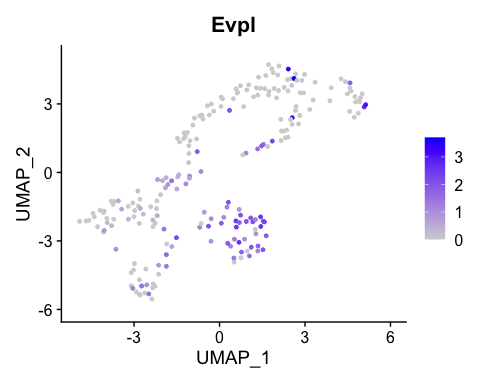
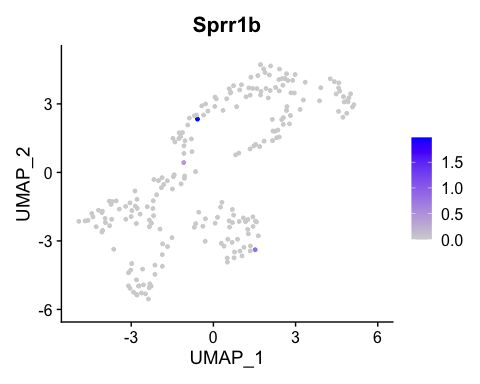
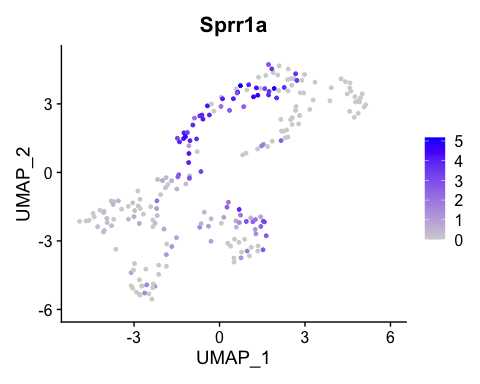
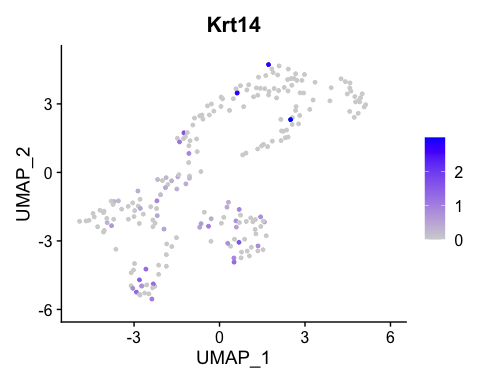
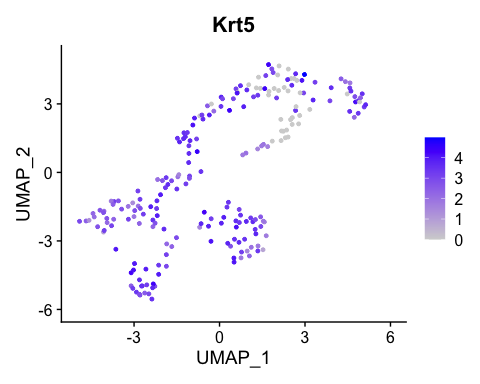
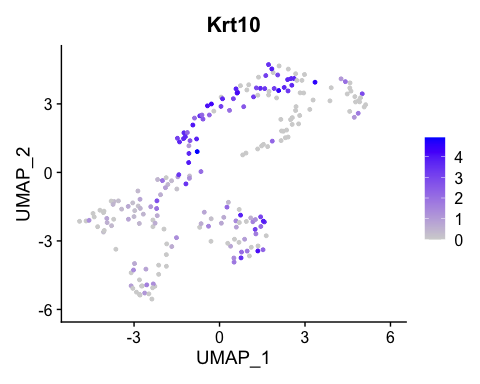
cat('features')

## features

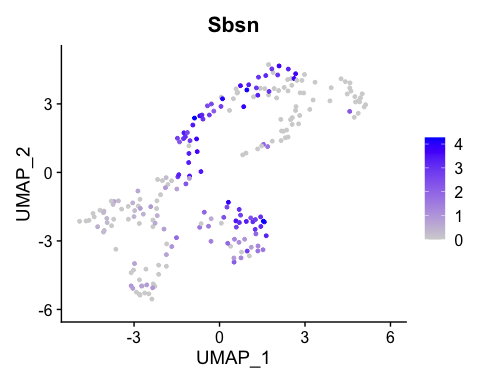
summary(kerato$nFeature\_RNA)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 201.0 372.5 1286.0 1765.8 2797.0 5936.0

# Parse list\_of\_features to FeaturePlot, which gives heatmaps of the clusters showing a features expression.   
  
for (gene in list\_of\_features){  
  
 p <- FeaturePlot(kerato, features = (gene))  
 print(p)  
}

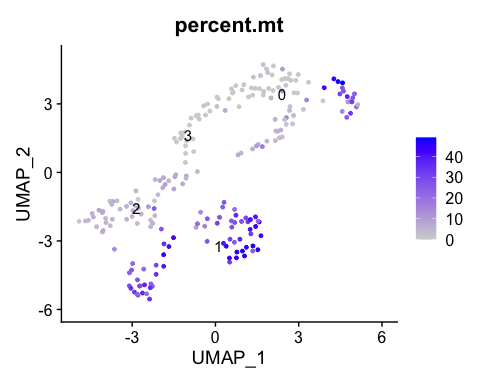


# Use this line to see single feature plot, for ease.   
FeaturePlot(kerato, features = "Sbsn")

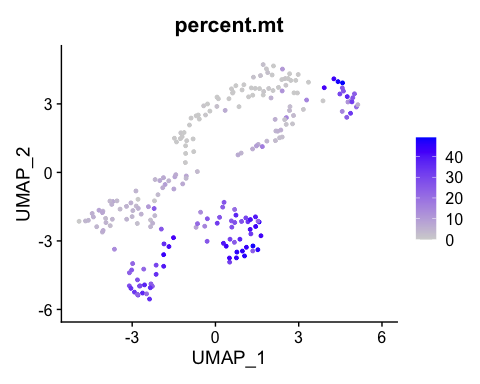


#this will print all genes within a character vector's cluster graphs  
  
# for (gene in actin\_markers){  
#   
# p <- FeaturePlot(kerato, features = (gene))  
# print(p)  
# }

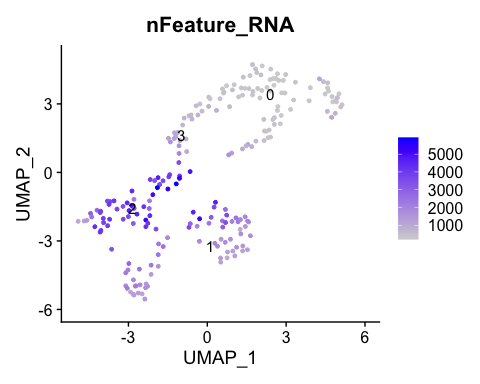
# Parse QC metrics to FeaturePlot, which gives heatmaps of the clusters showing a features expression.   
  
FeaturePlot(kerato, features = "percent.mt", label = TRUE)



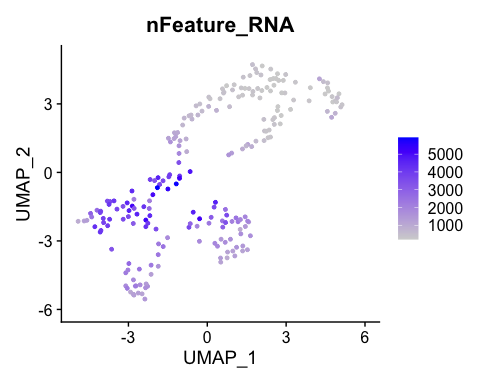
FeaturePlot(kerato, features = "percent.mt")



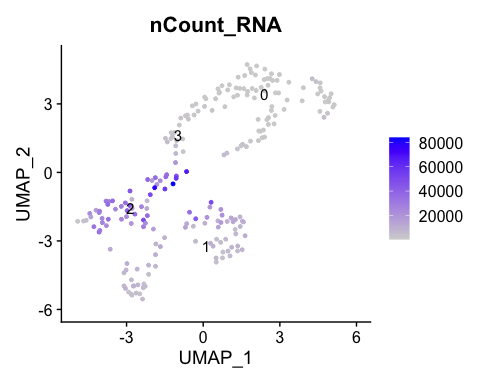
FeaturePlot(kerato, features = "nFeature\_RNA", label = TRUE)



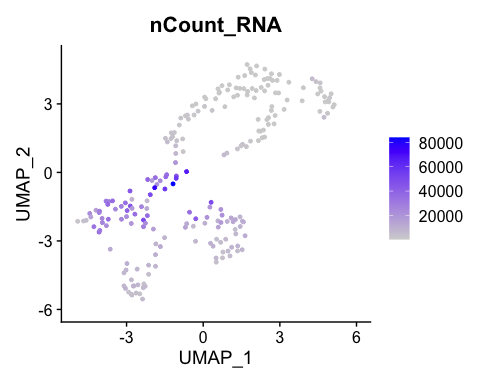
FeaturePlot(kerato, features = "nFeature\_RNA")



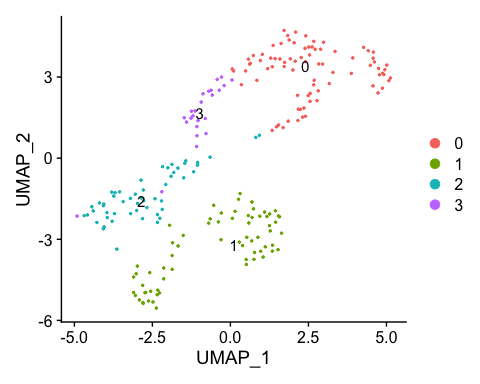
FeaturePlot(kerato, features = "nCount\_RNA", label = TRUE)



FeaturePlot(kerato, features = "nCount\_RNA")



#cluster graph showing labeled clusters  
DimPlot(kerato, reduction = "umap", label = TRUE, pt.size = 0.5)



# Create list of marker genes for the clusters using filters. min.pct is the minimum percentage present. only.pos selects only positive markers for this.  
  
markers\_streamlined <-  
 kerato %>%  
 FindAllMarkers(only.pos = TRUE, min.pct = 0.25, thresh.use = 0.25) %>%  
 group\_by(cluster) %>%   
 top\_n(10, avg\_log2FC) # only take top ten genes.

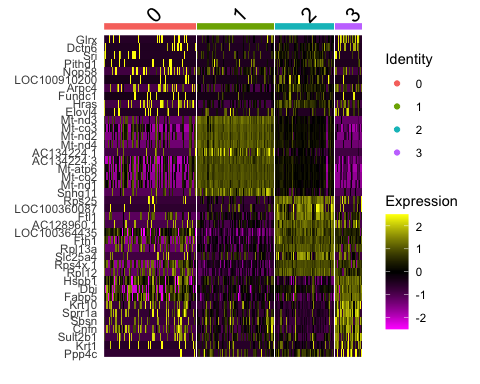
## Calculating cluster 0

## Calculating cluster 1

## Calculating cluster 2

## Calculating cluster 3

# Plot heatmap  
kerato %>%  
 DoHeatmap(  
 features = markers\_streamlined$gene  
   
 )

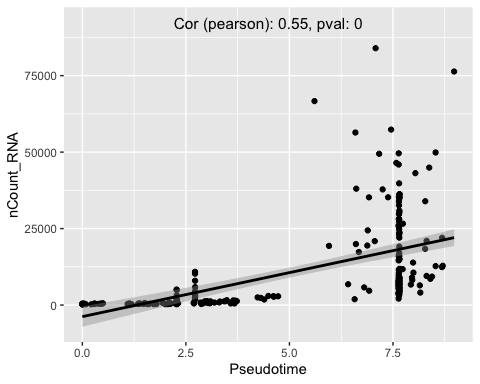


# changing text size with + theme(axis.text.y = element\_text(size = 0.1)) doesn't seem to work, needs some graphical adjustments.

# Parse QC metrics to scatter plots using geom\_point (ggplot2), adding smoothened line of best fit with stat\_smooth, and pearson correlation coefficient with geom\_cor.   
  
kerato %>%  
 tidyseurat::ggplot(aes(x=Pseudotime, y=nCount\_RNA)) +   
 geom\_point() +   
 stat\_smooth(method="lm", se=TRUE, colour = 'black', plot.cor = TRUE) +   
 geom\_cor(method = "pearson")

## Warning: Ignoring unknown parameters: plot.cor

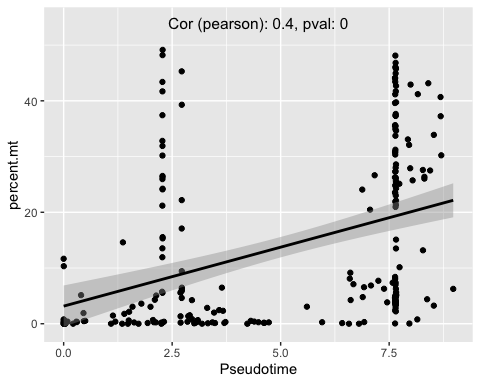
## `geom\_smooth()` using formula 'y ~ x'



kerato %>%  
 tidyseurat::ggplot(aes(x=Pseudotime, y=percent.mt)) + geom\_point() +   
 stat\_smooth(method="lm", se=TRUE, colour = 'black', plot.cor = TRUE) +   
 geom\_cor(method = "pearson")

## Warning: Ignoring unknown parameters: plot.cor

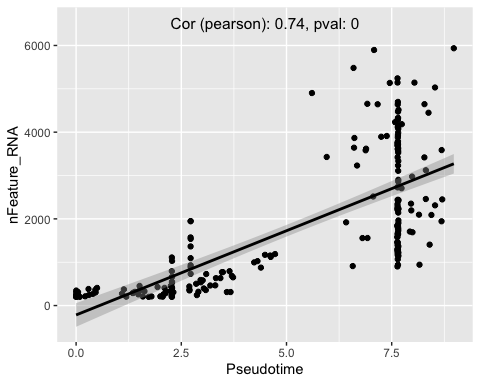
## `geom\_smooth()` using formula 'y ~ x'



kerato %>%  
 tidyseurat::ggplot(aes(x=Pseudotime, y=nFeature\_RNA)) +   
 geom\_point() +   
 stat\_smooth(method="lm", se=TRUE, colour = 'black', plot.cor = TRUE) +   
 geom\_cor(method = "pearson")

## Warning: Ignoring unknown parameters: plot.cor

## `geom\_smooth()` using formula 'y ~ x'



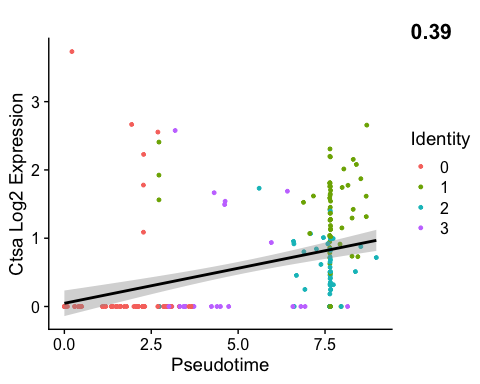
# This chunk parses all feature genes to FeatureScatter, and adds a line of best fit with standard error region present.   
  
# for (gene in list\_of\_features){  
# #zero\_scrub <- subset(x = kerato, subset = (as.factor('Krt10') > 0))  
# # solution may be here: https://github.com/satijalab/seurat/issues/2619  
#   
# p <- FeatureScatter(kerato, feature1 = "Pseudotime", feature2 = (gene))  
#   
# print(p +  
#   
# theme(plot.title = element\_text(hjust = 1.2, vjust = -2)) + # adjust pearsons coefficient to right hand side  
# stat\_smooth(method="lm", se=TRUE, colour = 'black') + # regression line  
#   
# #labels  
# xlab('Pseudotime') +   
# ylab(sprintf('%s Log2 Expression', gene)) +  
# labs(fill = "Cluster"))   
# }

#zero\_scrub <- subset(x = kerato, subset = (as.factor('Krt10') > 0))  
 # solution may be here: https://github.com/satijalab/seurat/issues/2619  
# obj\_subset <- kerato[, GetAssayData(kerato[[assay]])[gene, ] > 1]  
#actin\_markers <- c("Arhgdia","Camsap2", "Anxa1", "Rflnb", "Tmsb4x", "Arpc1b", "Sbsn","Dsg3", "Ahnak", "Cdh1", "Col3a1", "Tpt1", "Lgals7", "Rack1", "S100a11", "Eef1b2", "Ran", "Sprr1a", "Klf8", "Ptprf") Cfl1, Ipo9, Xpo6, Tgm1  
  
# set gene for line plot|  
# |  
# |  
  
#experiment removing zero values from plot  
# obj\_subset <- subset(x = kerato, subset = Xpo6 > 0)  
#   
#   
# p <- FeatureScatter(obj\_subset, feature1 = "Pseudotime", feature2 = (gene))  
#   
# print(p +  
#   
# theme(plot.title = element\_text(hjust = 1.2, vjust = -2)) + # adjust pearsons coefficient to right hand side  
# stat\_smooth(method="lm", se=TRUE, colour = 'black') + # regression line  
#   
# #labels  
# xlab('Pseudotime') +   
# ylab(sprintf('%s Log2 Expression', gene)) +  
# labs(fill = "Cluster"))

#chunk plotting all pseudotime line graphs for genes in a character vector  
  
# for (gene in actin\_markers){  
#   
# p <- FeatureScatter(kerato, feature1 = "Pseudotime", feature2 = (gene))  
#   
# print(p +  
#   
# theme(plot.title = element\_text(hjust = 1.2, vjust = -2)) + # adjust pearsons coefficient to right hand side  
# stat\_smooth(method="lm", se=TRUE, colour = 'black') + # regression line  
#   
# #labels  
# xlab('Pseudotime') +   
# ylab(sprintf('%s Log2 Expression', gene)) +  
# labs(fill = "Cluster"))   
# }

# Similar to chunk above, investigating single genes for ease of use.   
  
pseudotime\_scatter\_gene <- 'Ctsa'  
#"Ddx6", "Eif4e"  
  
#b, l, d, s, h, c, k, e, g, z, j, f, r, m, 7, a, q  
  
FeatureScatter(kerato, feature1 = "Pseudotime", feature2 = (pseudotime\_scatter\_gene)) +  
 theme(plot.title = element\_text(hjust = 1.2, vjust = -2)) + # adjust pearsons coefficient to right hand side  
 stat\_smooth(method="lm", se=TRUE, colour = 'black') + # regression line  
   
 #labels  
 xlab('Pseudotime') +   
 ylab(sprintf('%s Log2 Expression', pseudotime\_scatter\_gene)) +  
 labs(fill = "Cluster")

## `geom\_smooth()` using formula 'y ~ x'



FeaturePlot(kerato, features = (pseudotime\_scatter\_gene))

