Graphical Analyses of Expanded Acne Dataset

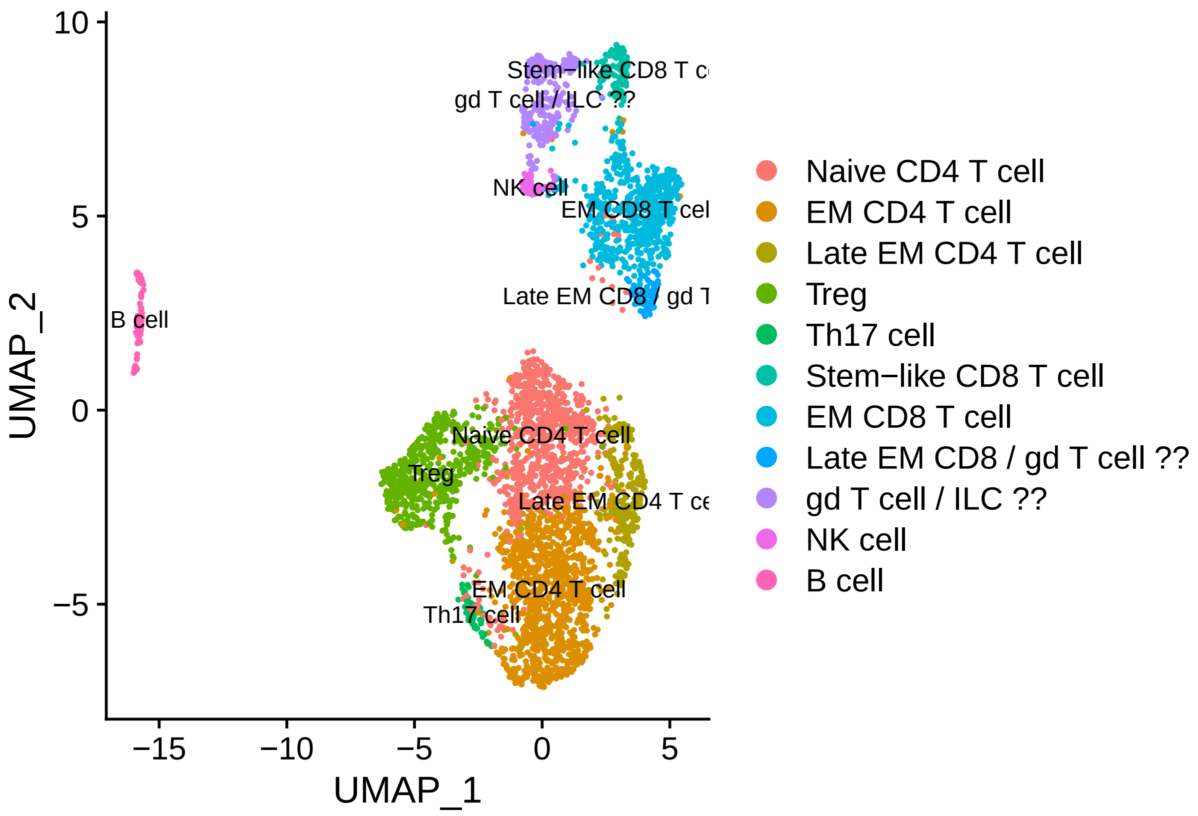
TamTo

2023-10-07

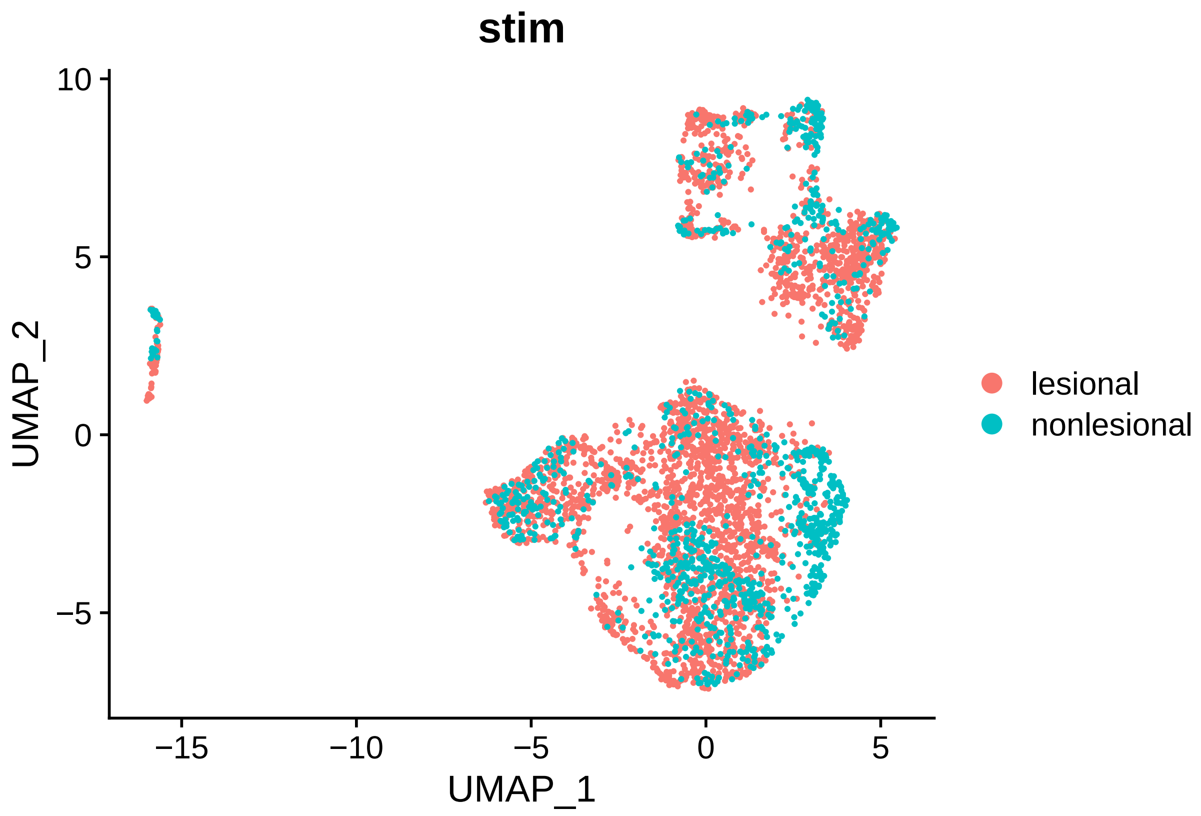
I'm working with the acne dataset I expanded (NewLymphocyteClusters Seurat object). Here we will analyze the data in various graphical forms to let us generate hypotheses and confirm results we may see in wet lab experiments.

To easily find genes of interest and their expression level, you can make feature maps, dot plots, etc.

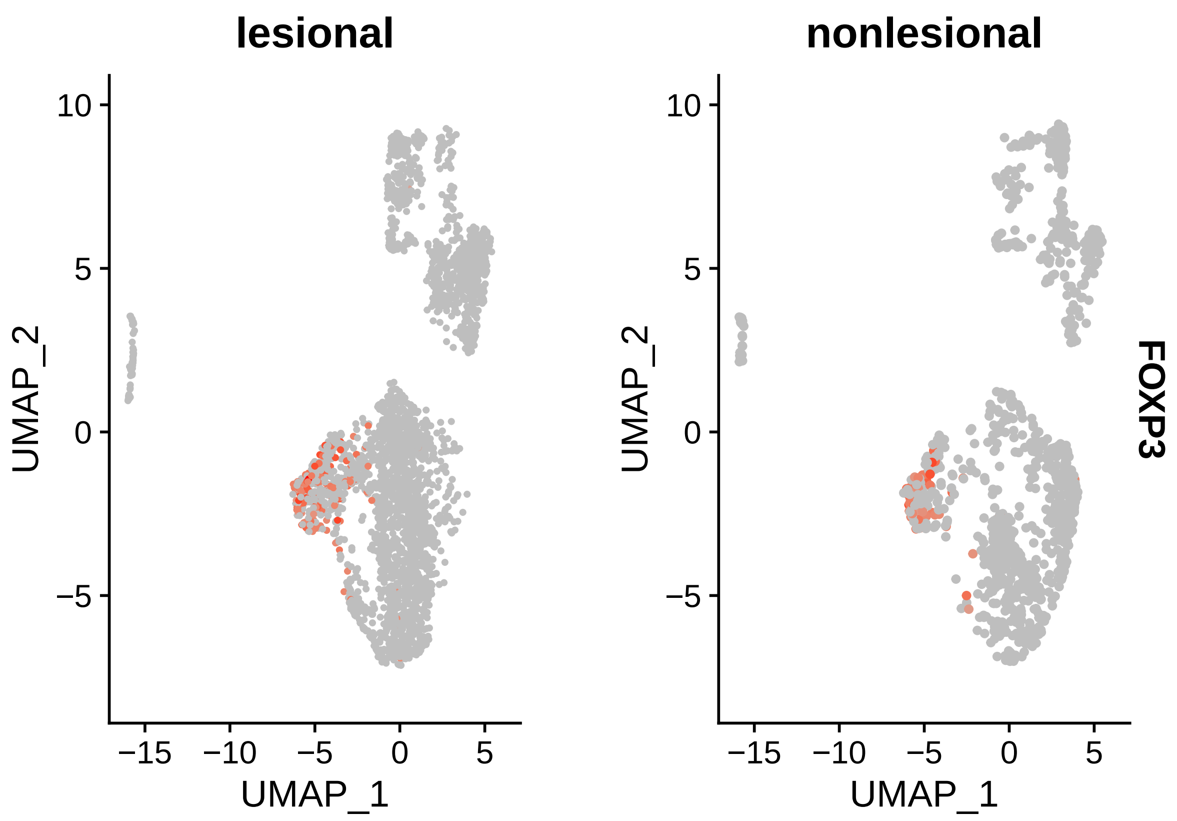
```
# View the UMAP of our Seurat object to look at our cell types again.
DimPlot(object = NewLymphocyteClusters, reduction = "umap", label = TRUE, label.size = 3)
```



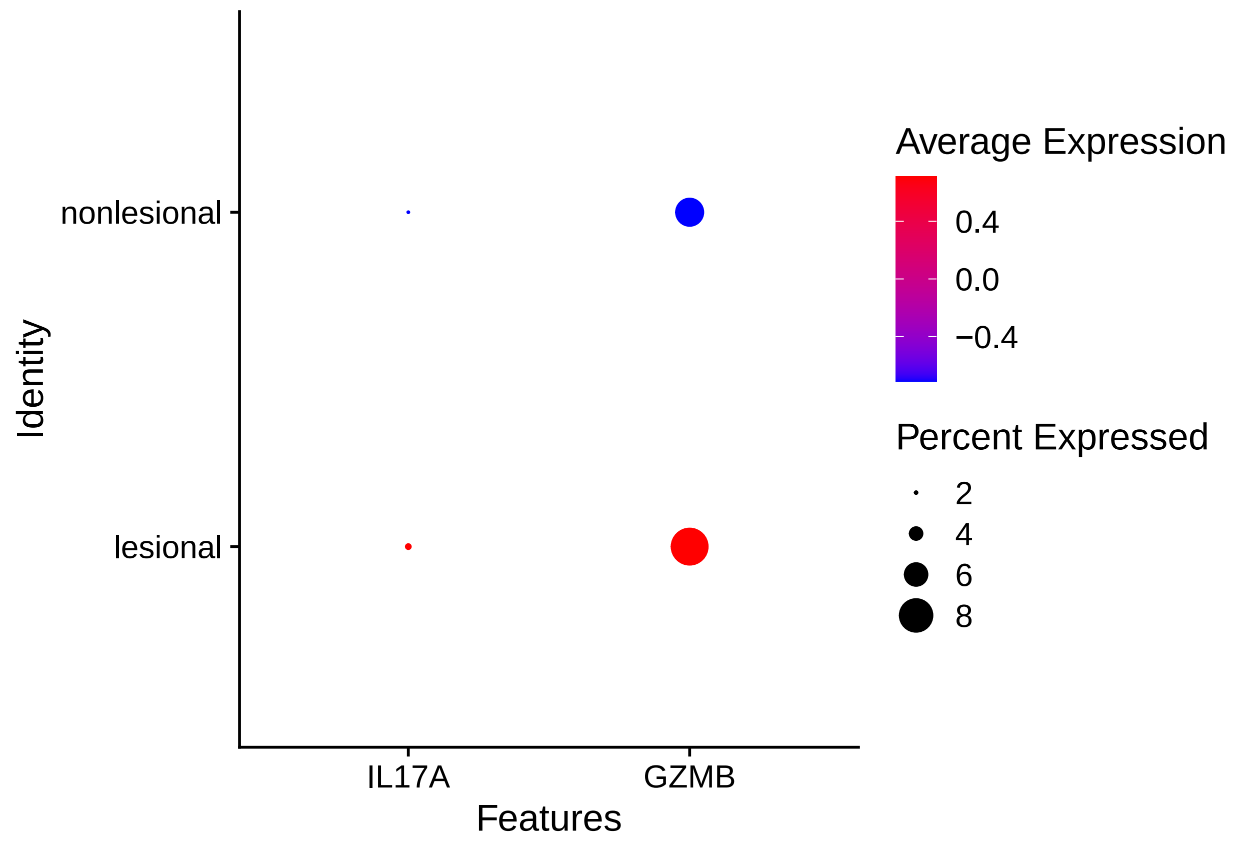
```
# We can also view it based on nonlesional skin or by the acne lesional skin or we can use the split.by
DimPlot(object = NewLymphocyteClusters, reduction = "umap", group.by = "stim", label = FALSE)
```



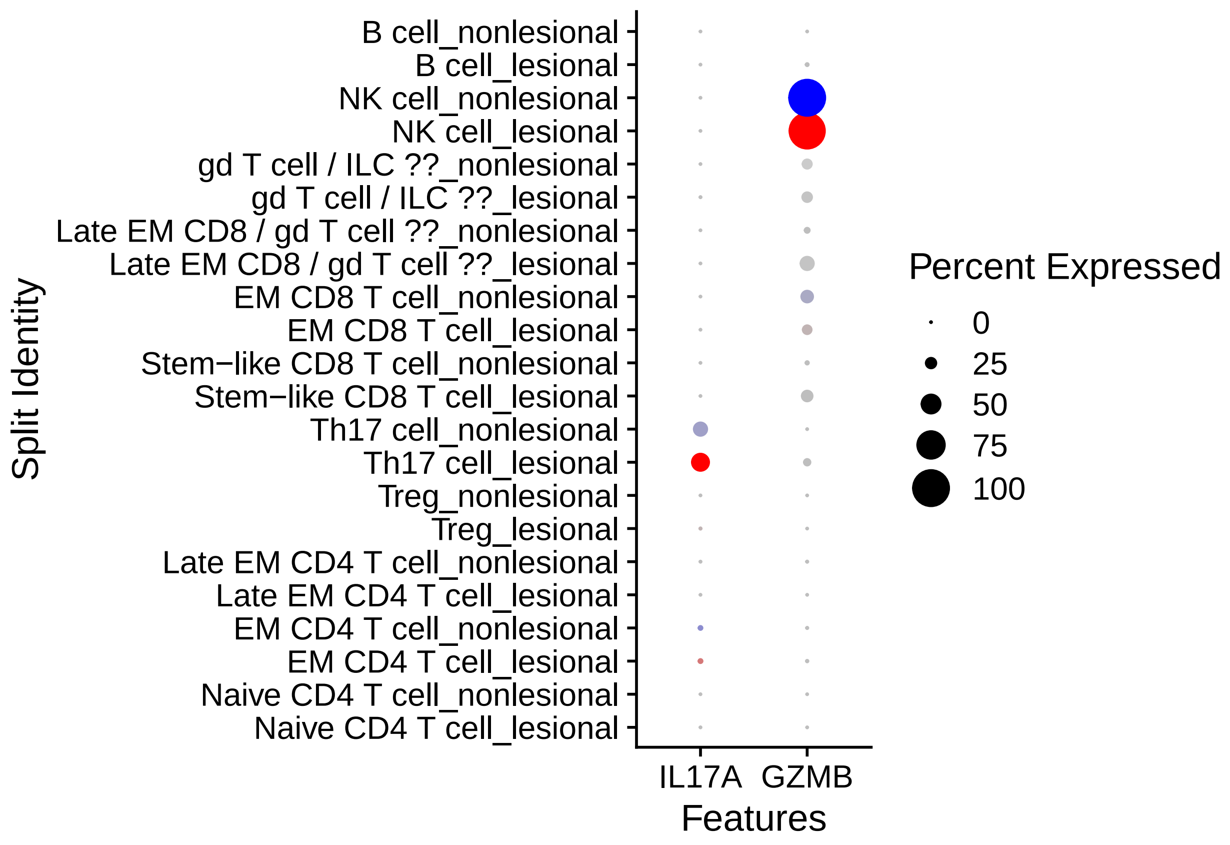
```
# View gene expression on the UMAP. Here, we see that FOXP3 is expressed mainly only in the Treg cluste
FeaturePlot(object = NewLymphocyteClusters, features = c("FOXP3"), split.by = "stim", cols = c("gray",
```



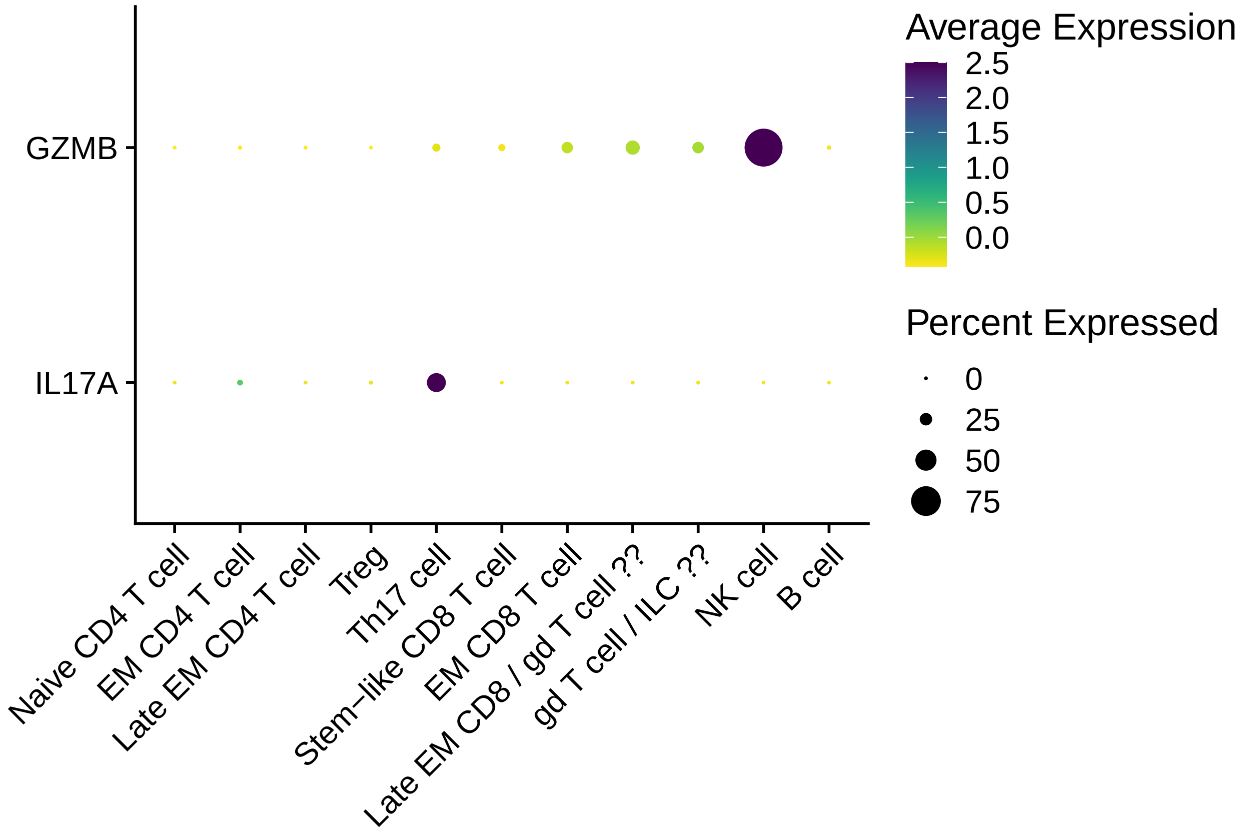
```
# View gene expression by dot plot. You can use group.by to view what genes may be more highly expresse
DotPlot(object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), group.by = "stim", cols = c("blu
```



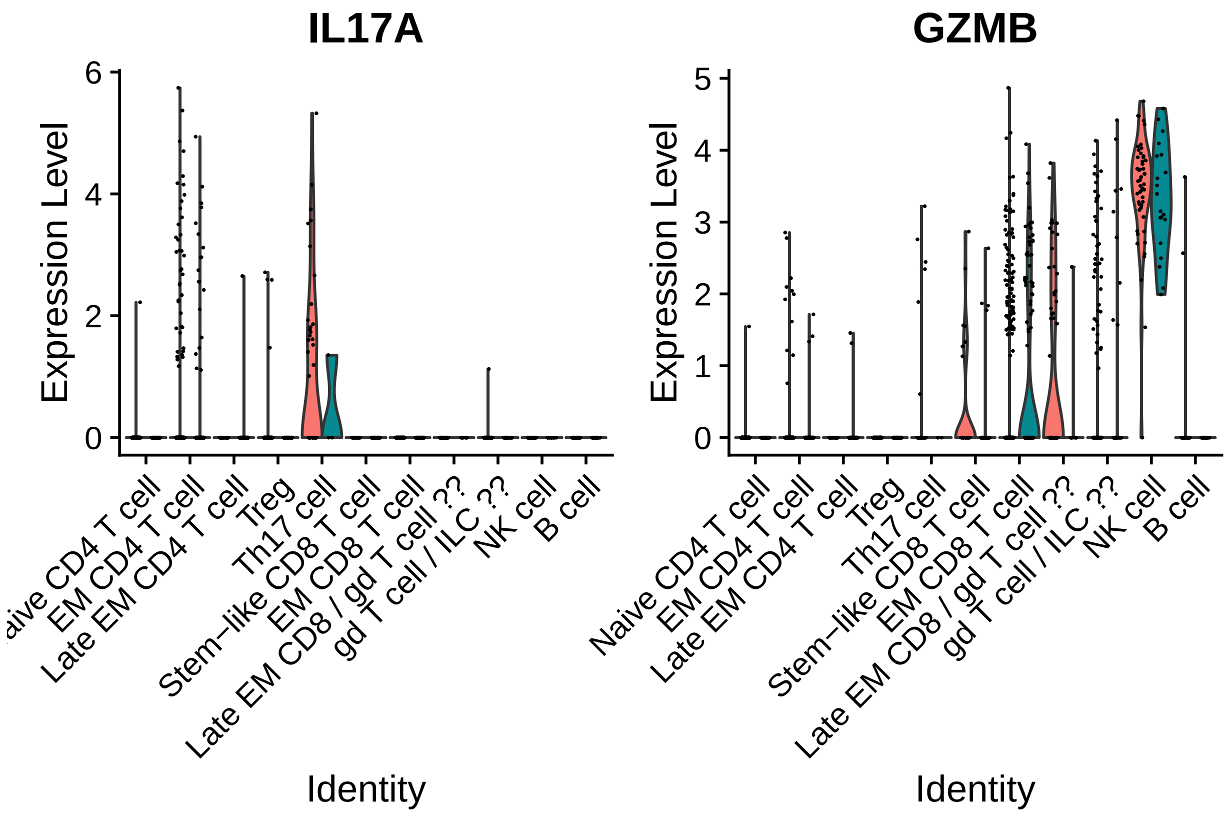
```
# You can see which cell types express the gene the most as well as in which lesion.
DotPlot(object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), group.by = "celltype", split.by =
```



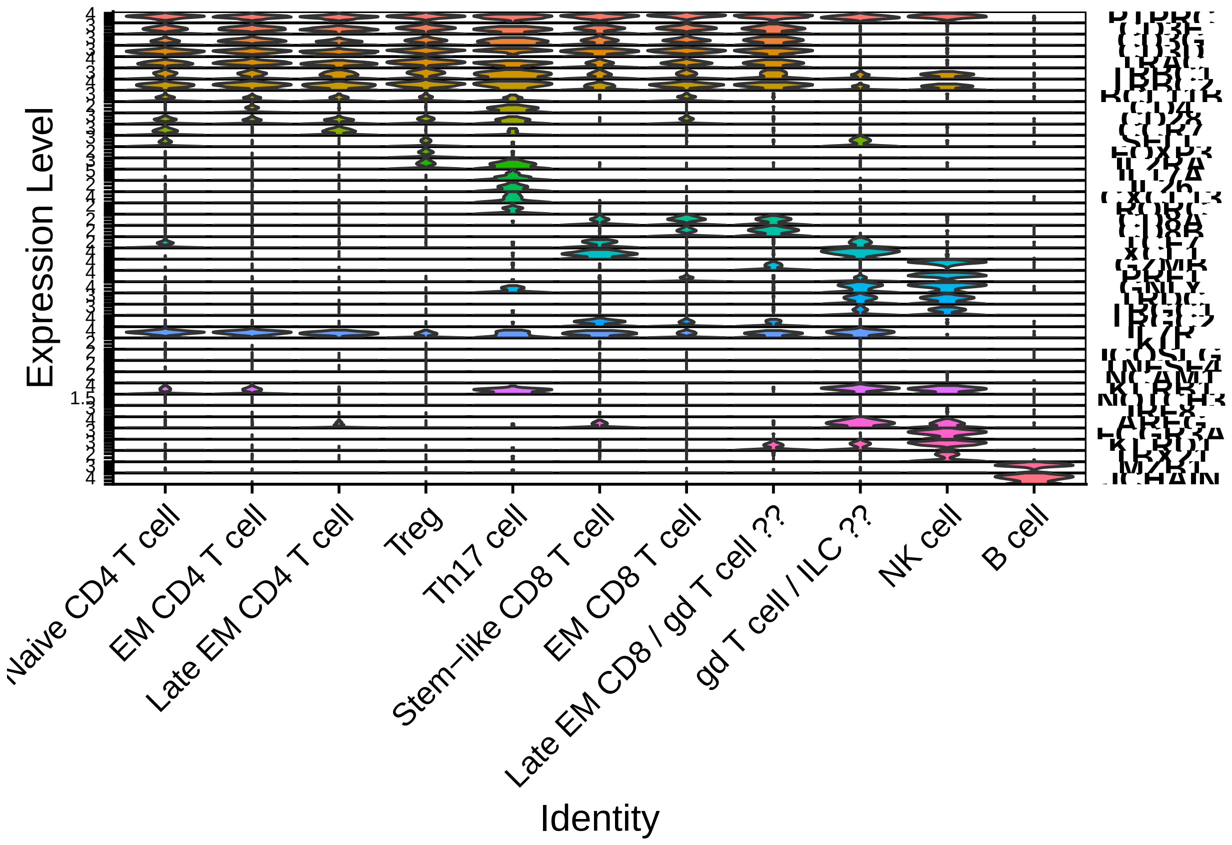
```
# To more aesthetically visualize which cell type expresses the gene, I like to use the scCustomize fun
DotPlot_scCustom(seurat_object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), flip_axes = TRUE
```



```
# We can visualize these genes by violin plots as well. This confirms what we saw in our dot plots.
VlnPlot(object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), split.by = "stim", pt.size = .00
```

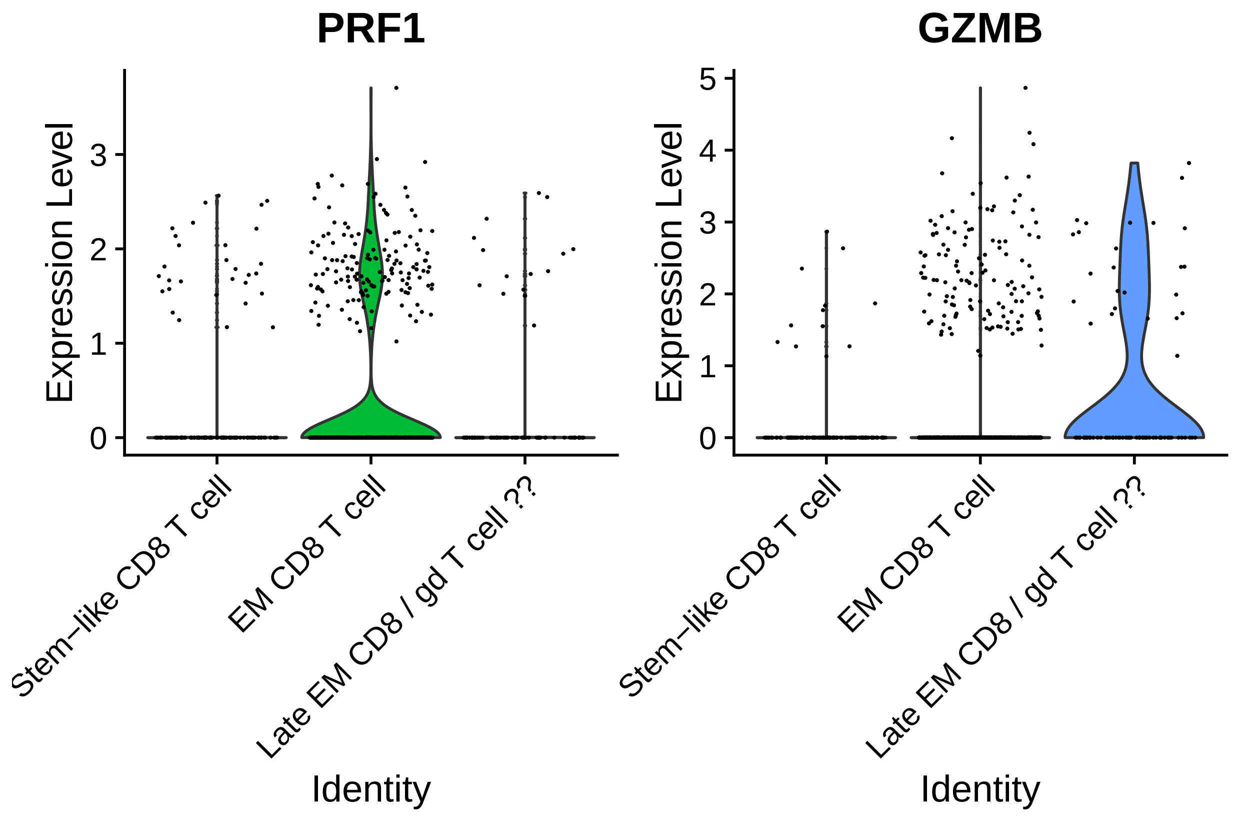


```
# If we want to visualize lots of genes in one graph, I recommend creating a stacked violin plot. Likew

features <- c("PTPRC", "CD3E", "CD3G", "CD3D", "TRAC", "TRBC1", "TRBC2", "BCL11B", "CD4", "CD28", "CCR7

VlnPlot(NewLymphocyteClusters, features, stack = TRUE, sort = FALSE, flip = TRUE) +
        theme(legend.position = "none")
```



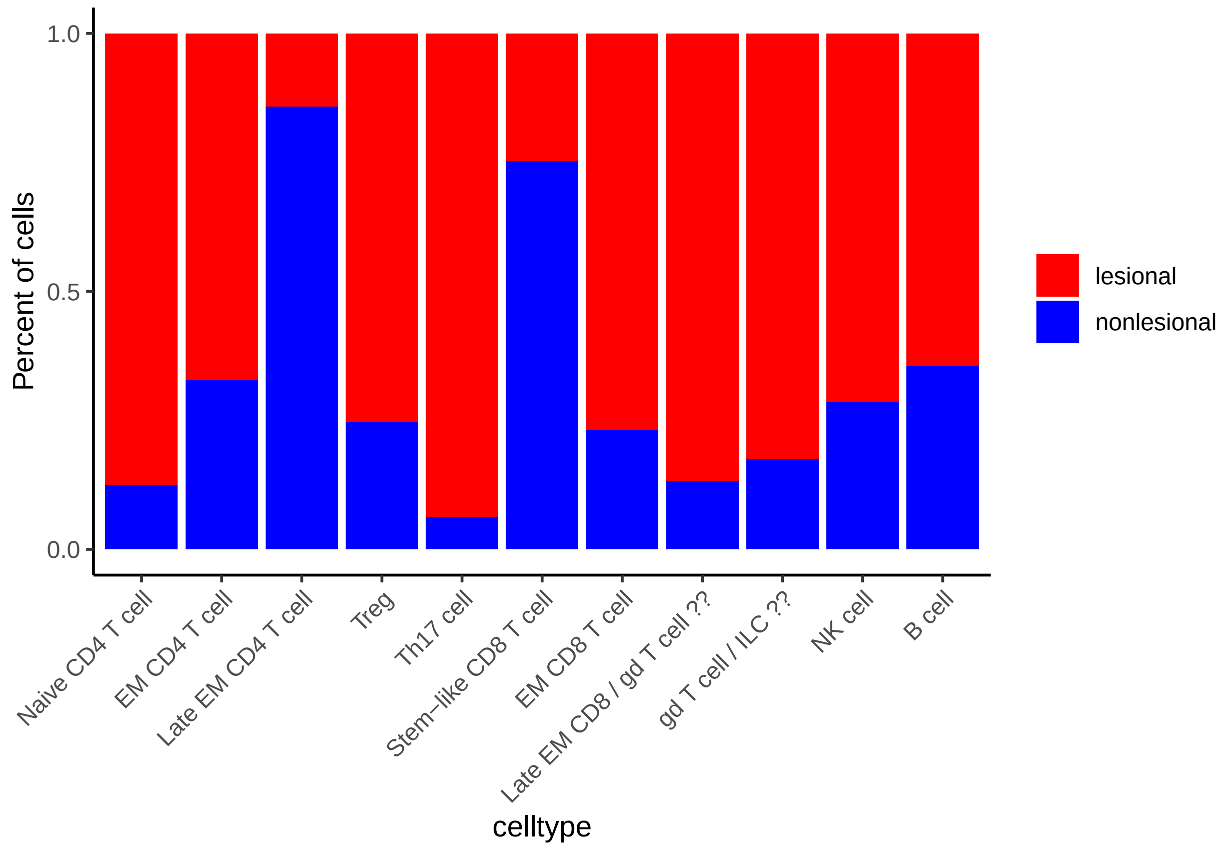
Sometimes, we only want to look at certain cell types (ex: only CD4s, or only CD8s, etc.) and analyze gene expression in only those clusters. In this case, we have to subset them from the rest of the clusters. Let's say we're only interested in the CD8 clusters.

```
# Look at what the active cell identities are and check the order of the levels
# Idents(object = NewLymphocyteClusters)
# levels(x = NewLymphocyteClusters)

# First stash the cell identity classes.
NewLymphocyteClusters[["old.ident"]] <- Idents(object = NewLymphocyteClusters)

# Now we can subset seurat object based on identity class.
CD8subset <- subset(x = NewLymphocyteClusters, idents = c("Naive CD4 T cell", "EM CD4 T cell", "Late EM

# To confirm that we have subsetted our clusters of interest, we can make plots! In this case, PRF1 is
VlnPlot(object = CD8subset, features = c("PRF1", "GZMB"))
```

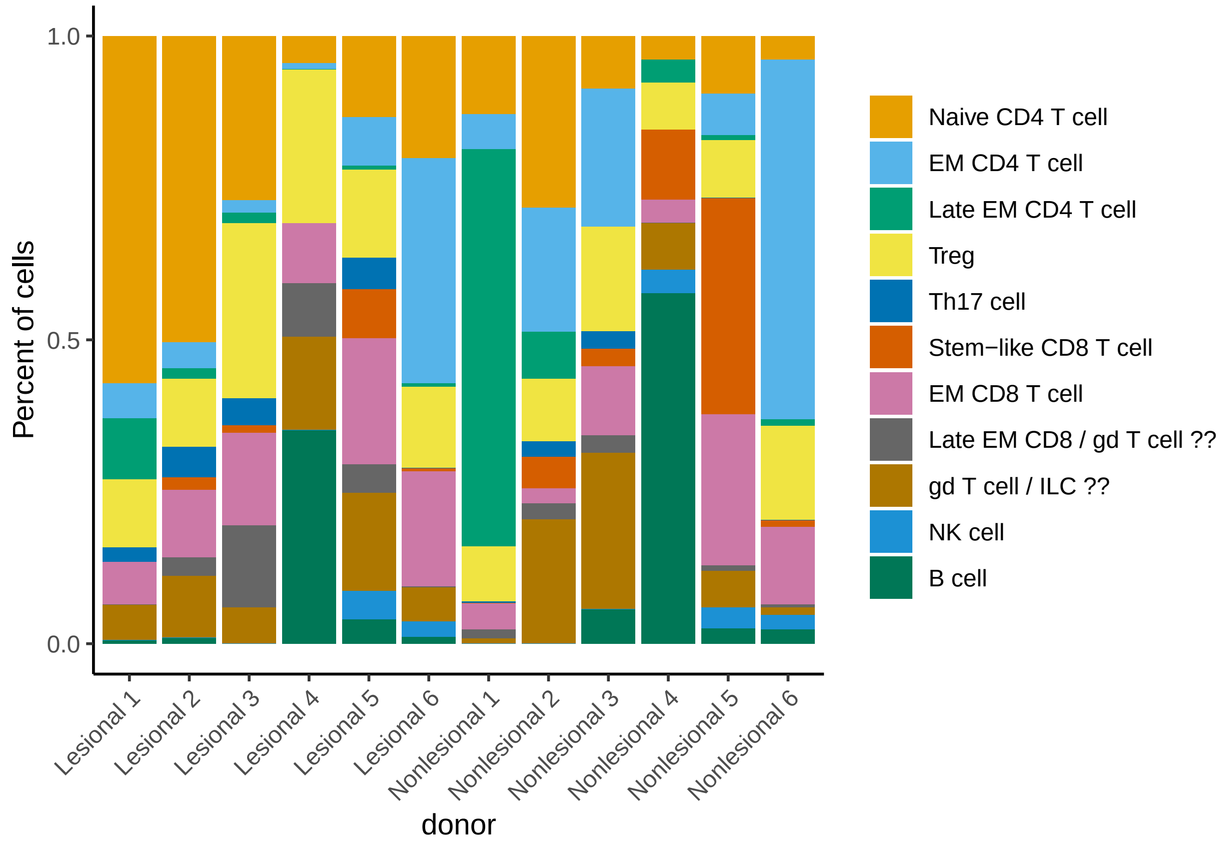


We can also visualize the data and compare cell proportions in each lesion (i.e. whether cell types may be differentially expressed in frequency). To do so, we can use the dittoSeq package to generate stacked bar plots. You can install the package through installing the BiocManager package (ensures the appropriate Bioconductor installation is compatible with the right version of R).

```
# if (!require("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")

# BiocManager::install("dittoSeq")

library(BiocManager)
library(dittoSeq)

# Create a bar plot that shows the cell type proportion split by nonlesional/lesional. Here, we see tha

dittoBarPlot(
    object = NewLymphocyteClusters,
    var = "stim",
    group.by = "celltype",
    main = NULL,
    color.panel = c("red", "blue"),
    x.reorder = c(7, 2, 5, 11, 10, 9, 3, 6, 4, 8, 1), # make sure to reorder the clusters for visual co
    )
```



```
# We can also create a bar plot that shows the proportion by individual donor too. As we can see here,

dittoBarPlot(
    object = NewLymphocyteClusters,
    var = "celltype",
    group.by = "donor",
    var.labels.reorder = c(7, 2, 5, 11, 10, 9, 3, 6, 4, 8, 1),
    main = NULL
    )
```



Let's say we want find the most differentially expressed genes in each lesion by cell type (i.e. which genes are most upregulated in the lesion). We can visually represent this by creating a volcano plot and plot the log2 fold change against negative log of the p-value (to also see which genes are the most statistically significantly upregulated). Let's take the Th17 cell cluster, for example.

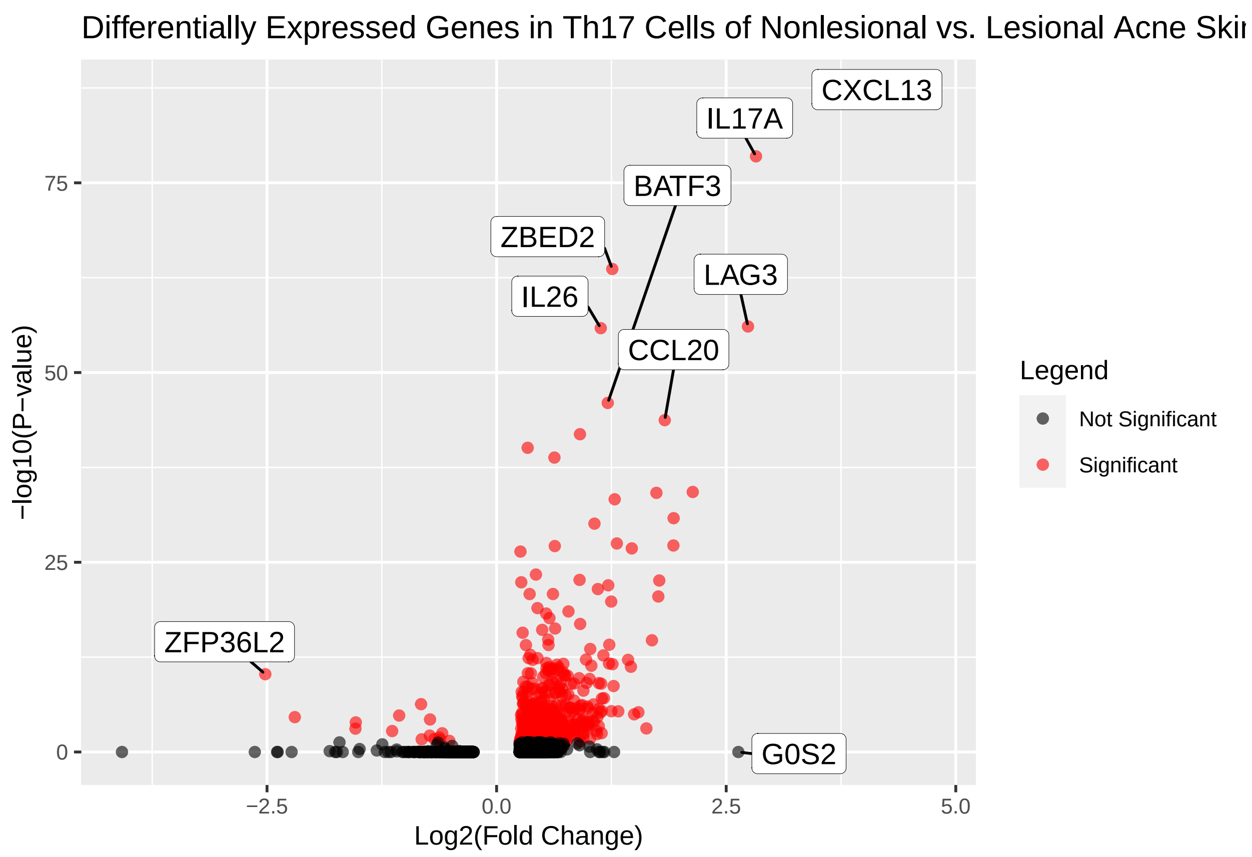
```
# Find all the markers in the gd T cell/ ILC subset.
Th17cell <- FindMarkers(NewLymphocyteClusters, ident.1 = 'Th17 cell')

# Currently it's a dataframe! We need to change it into a data table to be able to graph the gene names
Th17table <- data.table(Th17cell, keep.rownames=TRUE)

# Check column names.
colnames(Th17table)
```

```
## [1] "rn"          "p_val"      "avg_log2FC" "pct.1"      "pct.2"
## [6] "p_val_adj"
```

```
# I want to rename the first column as "gene".
colnames(Th17table)[1] ="gene"

# Check column names to make sure it renamed correctly.
colnames(Th17table)
```

```
## [1] "gene"        "p_val"      "avg_log2FC" "pct.1"      "pct.2"
## [6] "p_val_adj"
```

```
# I want to plot the statistically significant differentially expressed genes with the p_val_adj < 0.05

Legend <- ifelse(Th17table$p_val_adj < 0.05, "Significant", "Not Significant")

ggplot(Th17table, aes(x = avg_log2FC, y = -log10(p_val_adj))) +
  geom_point(aes(color = Legend), alpha = 0.6) +
  scale_color_manual(values = c("Significant" = "red", "Not Significant" = "black")) +
  theme(text = element_text(size = 10)) +
  labs(x = "Log2(Fold Change)", y = "-log10(P-value)", title = "Differentially Expressed Genes in Th17 (
 geom_label_repel(aes(label = gene), min.segment.length = unit(.1, 'lines'), force = 10, label.size = 0
```



```
# Here, we now see that IL17, CCL20, CXCL13, IL16, etc. are some genes that are more highly expressed i
```

Gene Ontology (GO) analysis is a way for use to analyze how genes from our samples translate to biological processes (i.e. we can see which biological processes may be upregulated in acne lesions compared to non-lesions.) We need to first install a few packages such as topGO (tests GO terms), clusterProfiler (visualize profiles of genomic coordinates), AnnotationDbi (queries SQLite-based annotation data), and org.Hs.eg.db (human genome wide annotation).

```
# BiocManager::install("topGO")
# BiocManager::install("clusterProfiler")
# BiocManager::install("AnnotationDbi")
# BiocManager::install("org.Hs.eg.db")

library(topGO)
library(clusterProfiler)
library(AnnotationDbi)
library(org.Hs.eg.db)

# I'm going to be creating a GO enrichment plot that describes biological processes for the genes highl

Th17GOgenes <- FindMarkers(NewLymphocyteClusters, ident.1 = "Th17 cell", min.pct = 0.10, logfc.threshol

geneList <- Th17GOgenes$p_val_adj
geneList <- na.omit(geneList)
names(geneList) <- rownames(Th17GOgenes)

# Use the topGO package to compare it to Gene Ontology http://www.geneontology.org/. Here we are taking

GOdata <- new("topGOdata",
        ontology = "BP",
        allGenes = geneList,
        geneSelectionFun = function(x)x == 1,
              annot = annFUN.org, mapping = "org.Hs.eg.db", ID = "symbol")

GOdata
```

```
##
## ------------------------- topGOdata object -------------------------
##
##  Description:
##    -
##
##  Ontology:
##    -  BP
##
##  1964 available genes (all genes from the array):
##    - symbol:  CXCL13 IL17A ZBED2 LAG3 IL26  ...
##    - score :  1.383e-87 3.25e-79 2.27e-64 8.5e-57 1.4e-56  ...
##    - 1199  significant genes.
##
##  1774 feasible genes (genes that can be used in the analysis):
##    - symbol:  CXCL13 IL17A ZBED2 LAG3 IL26  ...
##    - score :  1.383e-87 3.25e-79 2.27e-64 8.5e-57 1.4e-56  ...
##    - 1094  significant genes.
##
##  GO graph (nodes with at least  1  genes):
##    - a graph with directed edges
##    - number of nodes = 9003
##    - number of edges = 20062
##
## ------------------------- topGOdata object -------------------------
```

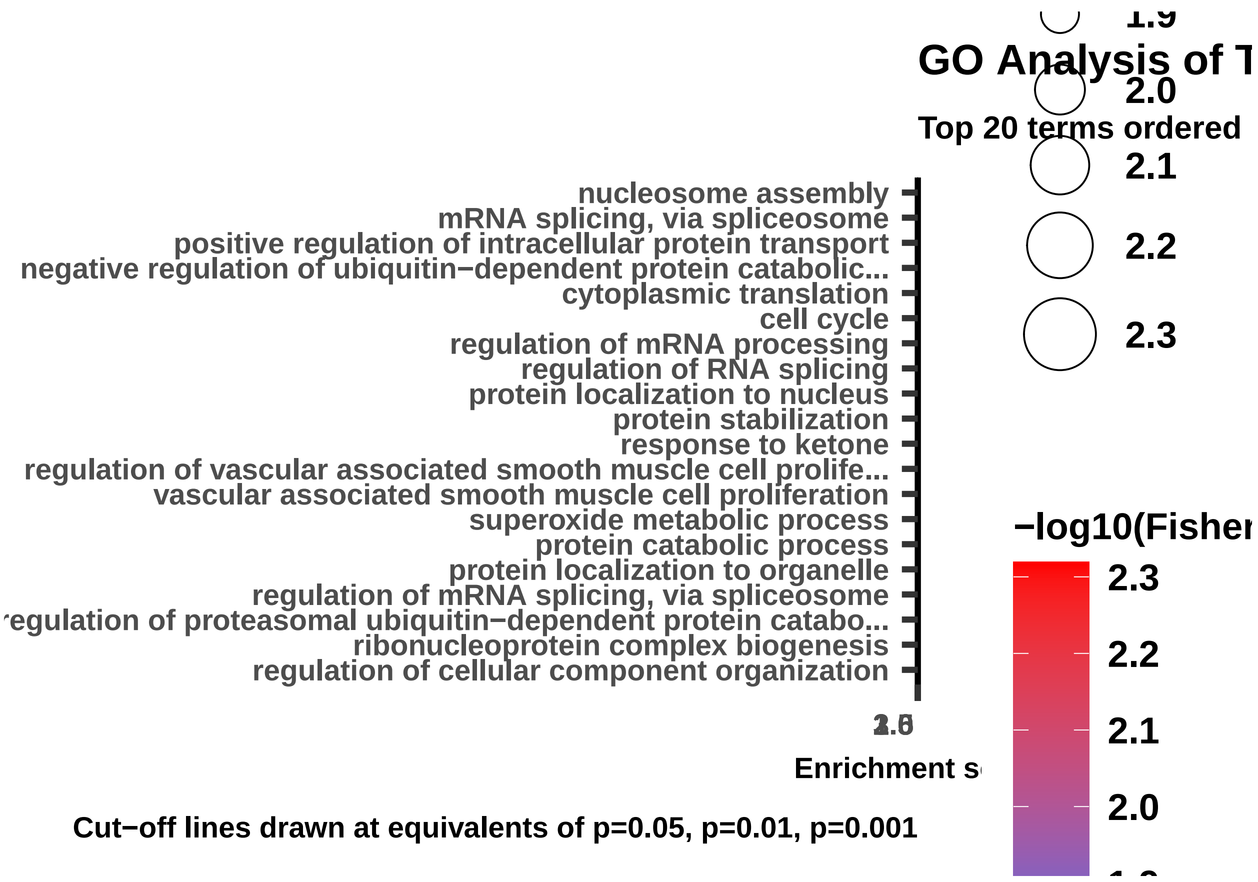
```
# We use Fisher's exact testing here because it determines whether there is a statistically significant

resultFisher <- runTest(GOdata, algorithm = "elim", statistic = "fisher")

GenTable(GOdata, Fisher = resultFisher, topNodes = 20, numChar = 60)
```

```
##         GO.ID                                                         Term
## 1  GO:0006334                                          nucleosome assembly
## 2  GO:0000398                                mRNA splicing, via spliceosome
## 3  GO:0090316          positive regulation of intracellular protein transport
## 4  GO:2000059 negative regulation of ubiquitin-dependent protein catabolic...
## 5  GO:0002181                                      cytoplasmic translation
## 6  GO:0007049                                                   cell cycle
## 7  GO:0050684                               regulation of mRNA processing
## 8  GO:0043484                                  regulation of RNA splicing
## 9  GO:0034504                               protein localization to nucleus
## 10 GO:0050821                                        protein stabilization
## 11 GO:1901654                                          response to ketone
## 12 GO:1904705 regulation of vascular associated smooth muscle cell prolife...
## 13 GO:1990874          vascular associated smooth muscle cell proliferation
## 14 GO:0006801                               superoxide metabolic process
## 15 GO:0030163                                  protein catabolic process
## 16 GO:0033365                             protein localization to organelle
## 17 GO:0048024                  regulation of mRNA splicing, via spliceosome
## 18 GO:0032434 regulation of proteasomal ubiquitin-dependent protein catabo...
## 19 GO:0022613                           ribonucleoprotein complex biogenesis
## 20 GO:0051128                  regulation of cellular component organization
##    Annotated Significant Expected Fisher
## 1         11          11     6.78 0.0048
## 2         69          53    42.55 0.0049
## 3         34          28    20.97 0.0077
## 4         10          10     6.17 0.0078
## 5         67          51    41.32 0.0079
## 6        232         160   143.07 0.0081
## 7         30          25    18.50 0.0088
## 8         43          34    26.52 0.0111
## 9         56          43    34.53 0.0112
## 10        46          36    28.37 0.0120
## 11        29          24    17.88 0.0122
## 12         9           9     5.55 0.0127
## 13         9           9     5.55 0.0127
## 14         9           9     5.55 0.0127
## 15       175         125   107.92 0.0130
## 16       154         108    94.97 0.0139
## 17        25          21    15.42 0.0140
## 18        25          21    15.42 0.0140
## 19        83          61    51.18 0.0141
## 20       303         204   186.86 0.0149
```

```r
# Now we can generate a table that shows the top biological pathways that are statistically significant

goEnrichment <- GenTable(
  GOdata,
  Fisher = resultFisher,
  orderBy = "Fisher",
  topNodes = 20,
  numChar = 60)

head(goEnrichment)
```

```
##        GO.ID                                                        Term
## 1 GO:0006334                                        nucleosome assembly
## 2 GO:0000398                                 mRNA splicing, via spliceosome
## 3 GO:0090316          positive regulation of intracellular protein transport
## 4 GO:2000059 negative regulation of ubiquitin-dependent protein catabolic...
## 5 GO:0002181                                    cytoplasmic translation
## 6 GO:0007049                                                 cell cycle
##   Annotated Significant Expected Fisher
## 1        11          11     6.78 0.0048
## 2        69          53    42.55 0.0049
## 3        34          28    20.97 0.0077
## 4        10          10     6.17 0.0078
## 5        67          51    41.32 0.0079
## 6       232         160   143.07 0.0081
```

```
# Now we can prepare the data for plotting. Let's filter the terms from our goEnrichment table that has

goEnrichment <- goEnrichment[goEnrichment$Fisher < 0.05,]
goEnrichment <- goEnrichment[,c("GO.ID","Term","Fisher")]
goEnrichment$Fisher <- as.numeric(goEnrichment$Fisher)

# I'm going to create a stored numeric value in "ntop" so we can adjust how many pathways we want to se

ntop <- 20
ggdata <- goEnrichment[1:ntop,]
ggdata$Term <- factor(ggdata$Term, levels = rev(ggdata$Term)) # fixes order

# Now we can plot the pathways against enrichment score (we are plotting the adjusted Fisher p-values a

ggplot(ggdata,
  aes(x = Term, y = -log10(Fisher), size = -log10(Fisher), fill = -log10(Fisher))) +
  expand_limits(y = 1) +
  geom_point(shape = 21) +
  scale_size(range = c(2.5,12.5)) +
  scale_fill_continuous(low = 'royalblue', high = 'red') +
  xlab('') + ylab('Enrichment score') +
  labs(
    title = 'GO Analysis of Th17 cells',
    subtitle = 'Top 20 terms ordered by Fisher Exact p-value',
    caption = 'Cut-off lines drawn at equivalents of p=0.05, p=0.01, p=0.001') +
  geom_hline(yintercept = c(-log10(0.05), -log10(0.01), -log10(0.001)), # creating horizontal lines to
    linetype = c("dotted", "longdash", "solid"),
    colour = c("black", "black", "black"),
    size = c(1, 1, 1)) +
  theme_bw(base_size = 24) +
  theme(
    legend.position = 'right',
    legend.background = element_rect(),
    plot.title = element_text(angle = 0, size = 16, face = 'bold', vjust = 1),
    plot.subtitle = element_text(angle = 0, size = 12, face = 'bold', vjust = 1),
    plot.caption = element_text(angle = 0, size = 11, face = 'bold', vjust = 1),
    axis.text.x = element_text(angle = 0, size = 11, face = 'bold', hjust = 1.10),
    axis.text.y = element_text(angle = 0, size = 11, face = 'bold', vjust = 0.5),
    axis.title = element_text(size = 11, face = 'bold'),
    axis.title.x = element_text(size = 11, face = 'bold'),
    axis.title.y = element_text(size = 11, face = 'bold'),
    axis.line = element_line(colour = 'black'),
    #Legend
    legend.key = element_blank(), # removes the border
    legend.key.size = unit(1, "cm"), # Sets overall area/size of the legend
    legend.text = element_text(size = 14, face = "bold"), # Text size
    title = element_text(size = 14, face = "bold")) +
  coord_flip()
```



```
ggplot2::ggsave("GOAnalysis_Th17cell_Fisher.png",
                device = NULL,
                height = 8.5,
                width = 12)

# We can see that the pathways upregulated in acne lesions are nucleosome assembly, mRNA splicing, intr
```

The process above for gene ontology is quite extensive, though. There is a great online platform called EnrichR where you can just plug in the top genes and get many different types of ontology plots.

```
Th17cellEnrichR <- FindMarkers(NewLymphocyteClusters, ident.1 = "Th17 cell", min.pct = 0.10, logfc.thres

# Turn into a table to get the gene names and rename the column.
Th17cellEnrichRtable <- data.table(Th17cellEnrichR, keep.rownames=TRUE)
```

```r
colnames(Th17cellEnrichRtable)[1] ="gene"

# Export into excel sheet.
# install.packages("writexl")
library("writexl")

# write_xlsx(Th17cellEnrichRtable,"C:\\Users\\tamto\\Desktop\\Th17cellEnrichRtable.xlsx")
```

Copy and paste the genes into https://maayanlab.cloud/Enrichr/ for analysis. Make sure you select for genes that are statistically significant (p_adj_val < 0.05).

Cells can transition from one functional state to another based on various stimuli or the microenvironment. (For example, a naive T cell can become a Th1, Th2, Th17, or Treg based on different stimuli.) Because we can analyze cell clusters from our single cell data, we can investigate cell trajectories too. Pseudotime analysis helps us determine which state each cell might be in based on their gene expression or the cells' progress through each transition state. Monocle is a powerful way to analyze trajectories because it learns the sequence of gene expression changes for each cell to place it within a trajectory.

Because this dataset doesn't really focus on a cell's transition over time (i.e. a stem cell data set or sequencing of cells over time), trajectory analysis is not as useful for our analysis, but it's good to know to apply for any future dataset that might need it.

Here is the site to help with installing the monocle3 package: https://cole-trapnell-lab.github.io/monocle3/docs/installation/

```r
# if (!requireNamespace("BiocManager", quietly = TRUE))
# install.packages("BiocManager")
# BiocManager::install(version = "3.14")

# BiocManager::install(c('BiocGenerics', 'DelayedArray', 'DelayedMatrixStats',
#                        'limma', 'lme4', 'S4Vectors', 'SingleCellExperiment',
#                        'SummarizedExperiment', 'batchelor', 'HDF5Array',
#                        'terra', 'ggrastr'))

# install.packages("devtools")
# devtools::install_github('cole-trapnell-lab/monocle3')

# install.packages("remotes")
# remotes::install_github("satijalab/seurat-wrappers")

library(monocle3)

# Useful way for getting started with monocle3: https://cole-trapnell-lab.github.io/monocle3/docs/getti

# Install SeuratWrappers and convert to monocle3 cell_data_set. We need to load the data into monocle3'

# remotes::install_github('satijalab/seurat-wrappers')
library(SeuratWrappers)

cds <- SeuratWrappers::as.cell_data_set(NewLymphocyteClusters)

# Now you can preprocess the data (dimension reduction, cell clustering, etc.). Monocle3 then "learns"

cds <- reduce_dimension(cds, preprocess_method = "PCA")
cds <- cluster_cells(cds)
cds <- learn_graph(cds, use_partition = FALSE)

# You can view the UMAP by plotting the cds and showing it on the trajectory graph. The funciton order_

plot_cells(cds,
           show_trajectory_graph = FALSE,
           color_cells_by = "partition")

cds <- order_cells(cds)

# After selecting the nodes of interest, you can plot it in various ways (i.e. get connect the nodes, l

plot_cells(cds,
           color_cells_by = "pseudotime",
           label_roots = F,
           label_leaves = T,
           label_branch_points = F
           )

# Here I chose the CD4 clusters. Based on the pseudotime plot I generated, it shows that the root cells

# Monocle3 is also useful for plotting where and when certain genes may be highly expressed (by compari

rowData(cds)$gene_name <- rownames(cds)
rowData(cds)$gene_short_name <- rowData(cds)$gene_name

# You can check to make sure a certain gene name is there by searching it in the row names of the cds t

term_genes <- c("TIGIT", "LAG3", "PDCD1", "CCR7", "SELL", "TCF7")

plot_cells(cds,
           genes = term_genes,
           label_cell_groups=TRUE,
           show_trajectory_graph=FALSE)

# Because our dataset does not have cells based on different time points, this analysis is not as relev

# Site for more monocle3 information: https://cole-trapnell-lab.github.io/monocle3/ . You can also do p
```

CONCLUSION

In this project, we went over various methods for analyzing scRNAseq data. Whether it be generating dot plots, volcano plots, or doing gene ontology analysis, there are so many ways for analyzing Seurat objects. There are also so many ways to visualize the data, and you can edit it to how you see fit based on your needs for data representation. This makes R such a powerful language to use for molecular biology analysis.