Graphical Analyses of Expanded Acne Dataset

TamTo

2023-10-07

Introduction

I'm working with the acne dataset I expanded from the last project (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.

```
knitr::opts_chunk$set(echo = TRUE)

library(tidyverse) # load the tidyverse
library(Seurat) # we need this to work with Seurat objects
library(data.table) # used to convert df to data table
library(scCustomize) # helps with visualization and aesthetics of single-cell data
library(ggrepel) # helps with text label positioning

# Check the working directory.
getwd()
```

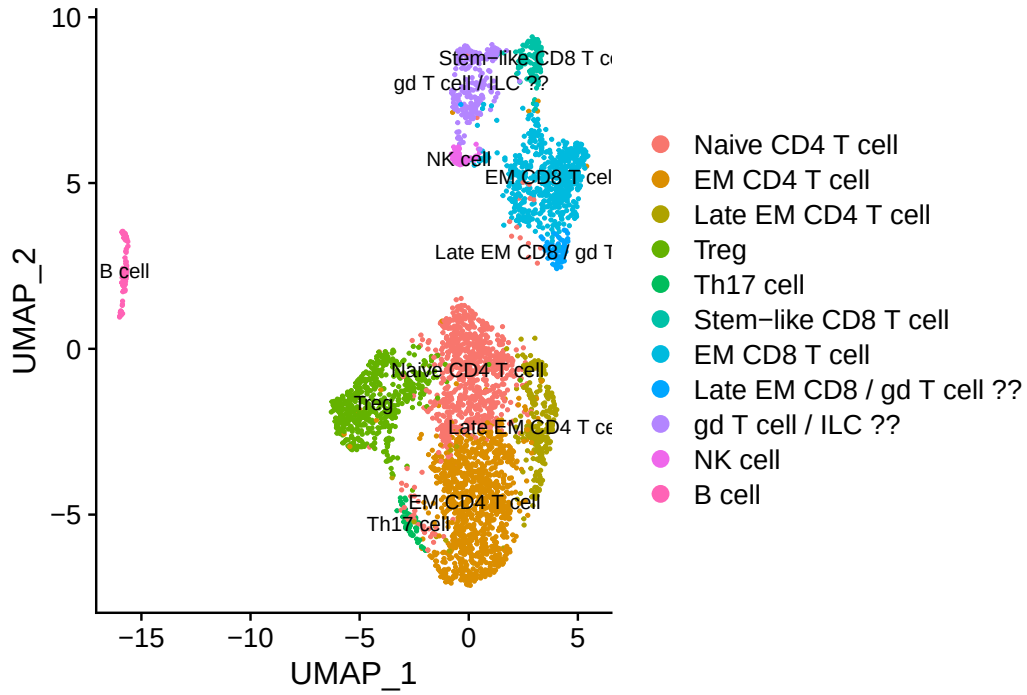
```
## [1] "/Users/tamto/Documents/GitHub/MolecularBiologyProjects/Projects/Project_AcneGraphicalAnalyses"
```

```
# Setting the wd.
setwd("/Users/tamto/Documents/GitHub/MolecularBiologyProjects/Projects/Project_AcneGraphicalAnalyses")

# Time to load the data in our global environment.
load("/Users/tamto/Desktop/NewLymphocyteClusters.Rdata")

# We can first view our object to know what we can work with in our analyses.
view(NewLymphocyteClusters)
```

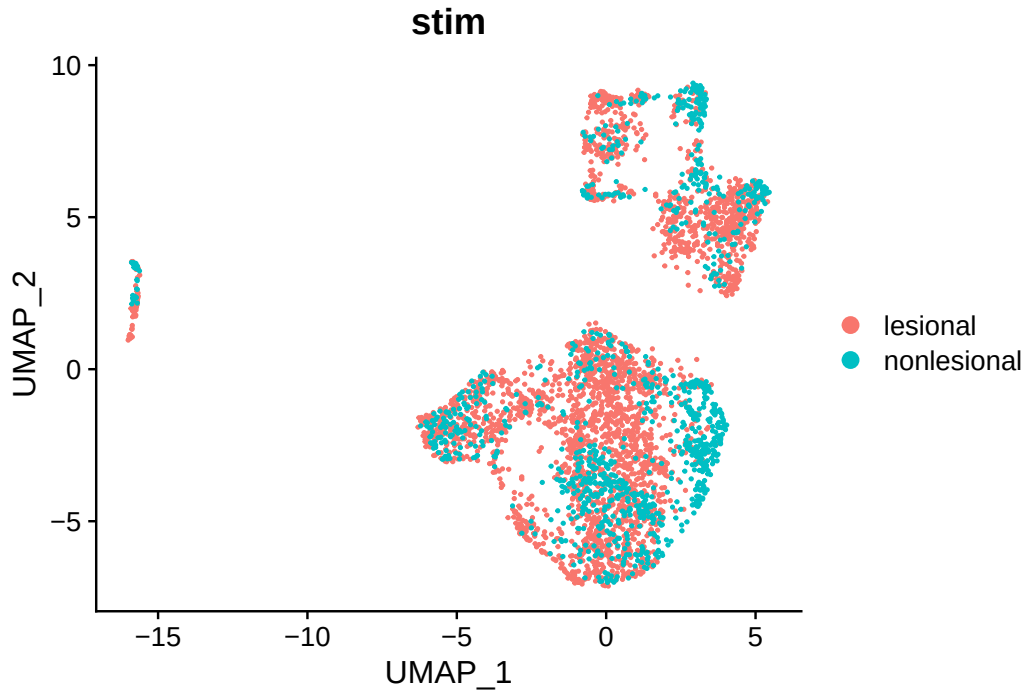
Visualizing Cell Types and Gene Expression by feature maps, dot plots, and violin plots}

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

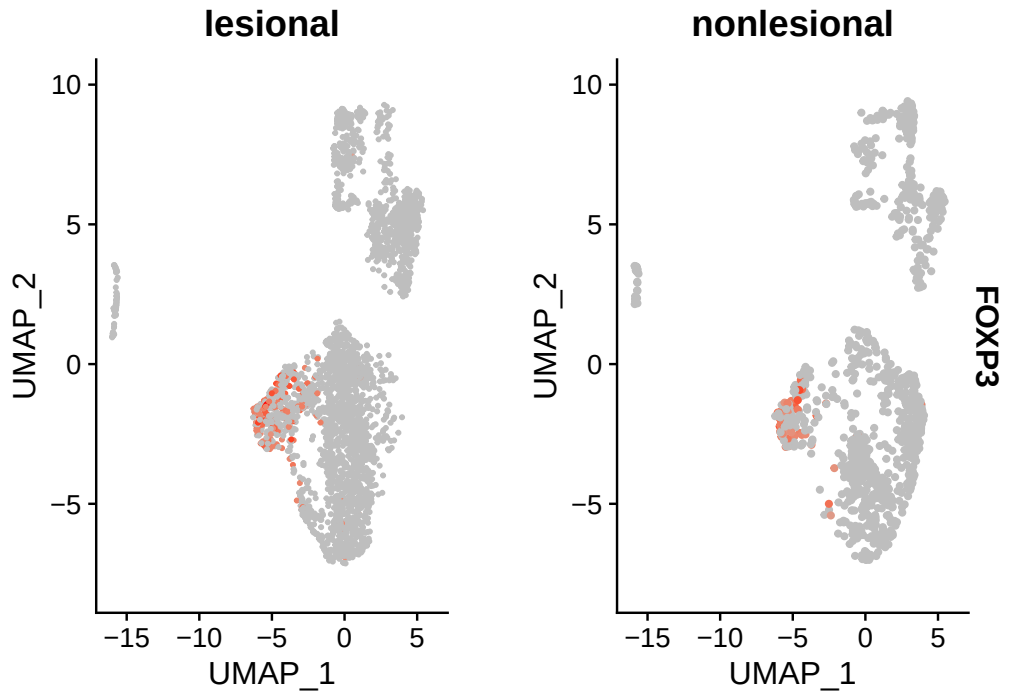
```
# View the UMAP of our Seurat object to look at our cell types again.
DimPlot(object = NewLymphocyteClusters, reduction = "umap", label = TRUE, label.size = 3)
```




```
# 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)
```

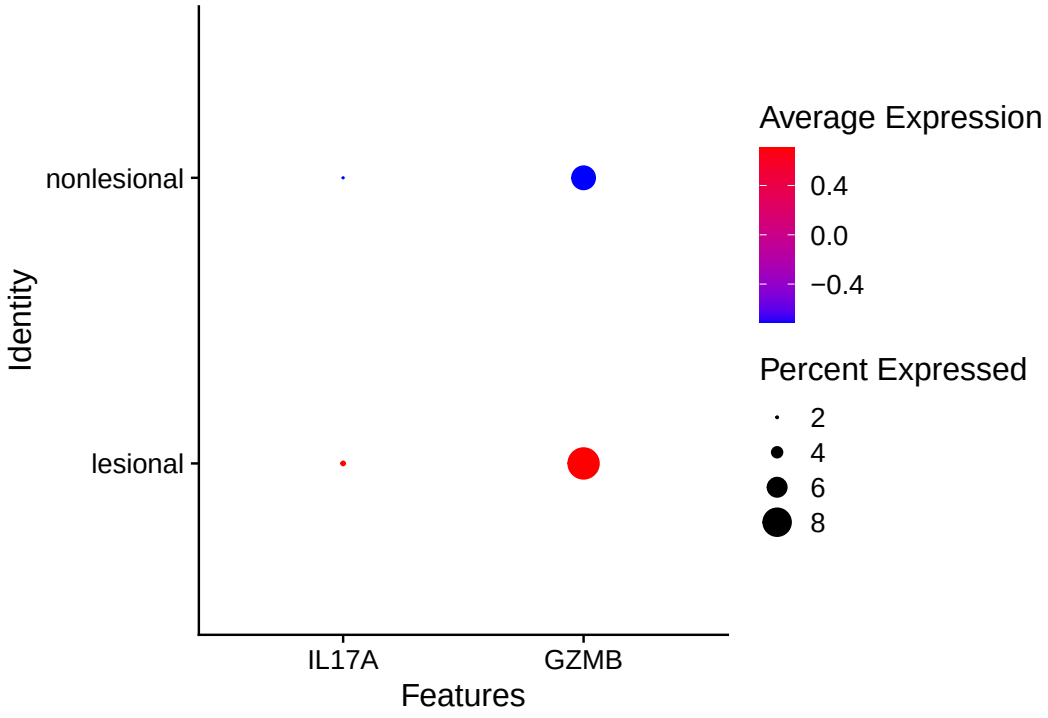



```
# View gene expression on the UMAP. Here, we see that FOXP3 is expressed mainly only in the Treg cluste
FeaturePlot(object = NewLymphocyteClusters, features = c("FOXP3"), split.by = "stim", cols = c("gray",
```

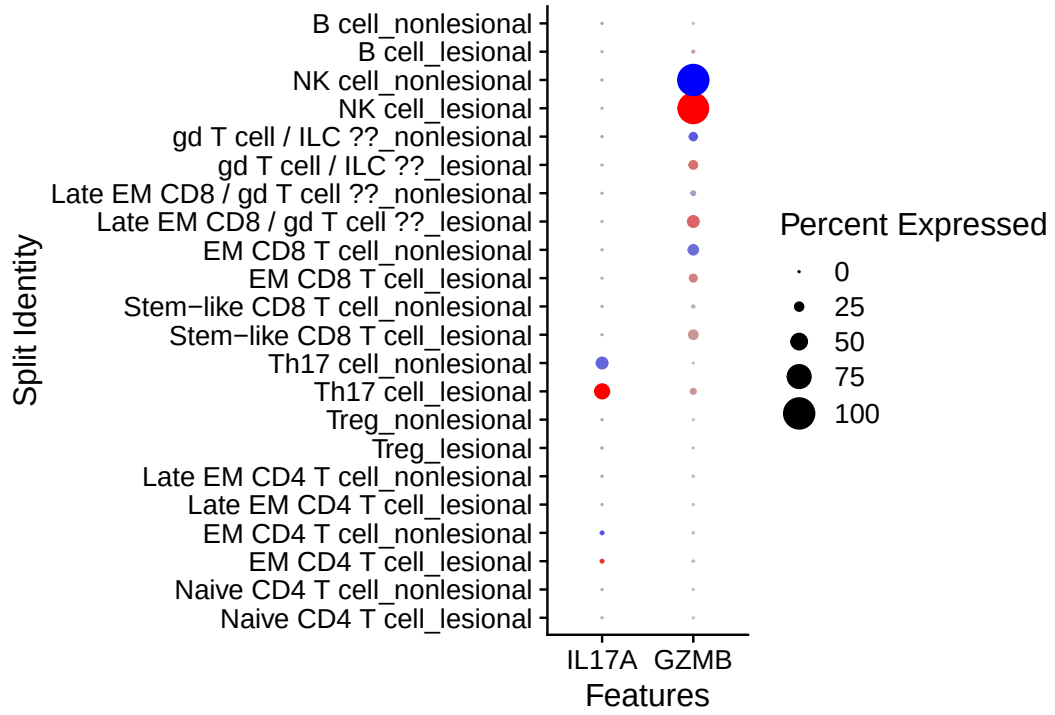



```
# View gene expression by dot plot. You can use group.by to view what genes may be more highly expresse
DotPlot(object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), group.by = "stim", cols = c("blu
```

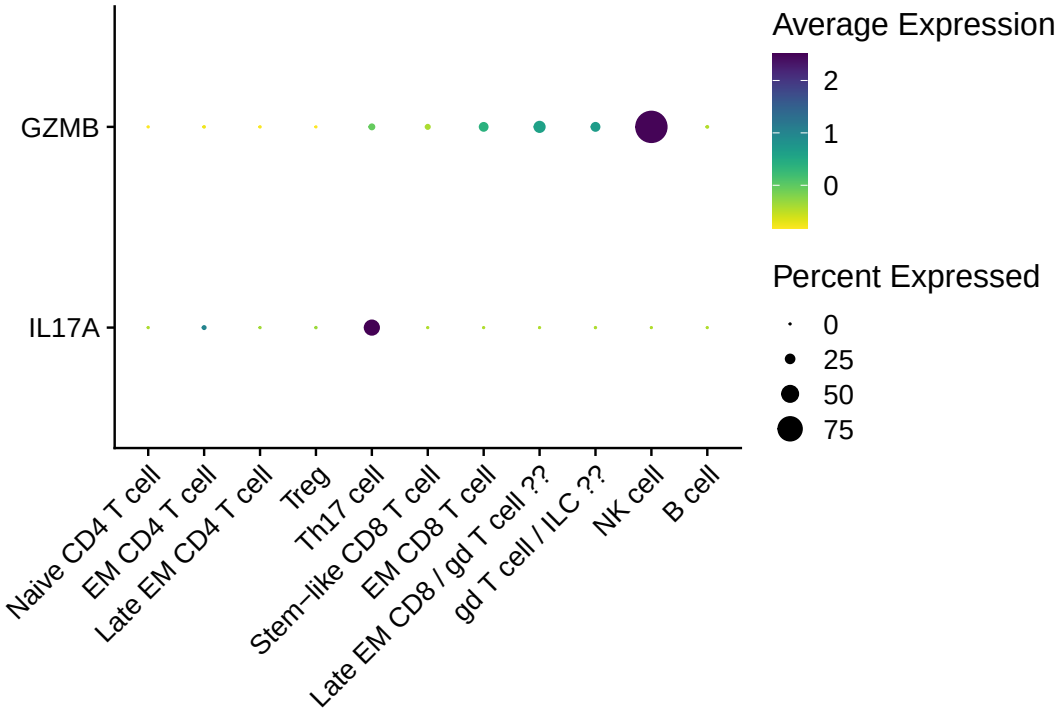
```
## Warning: Scaling data with a low number of groups may produce misleading
## results
```



```
# 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 =
```

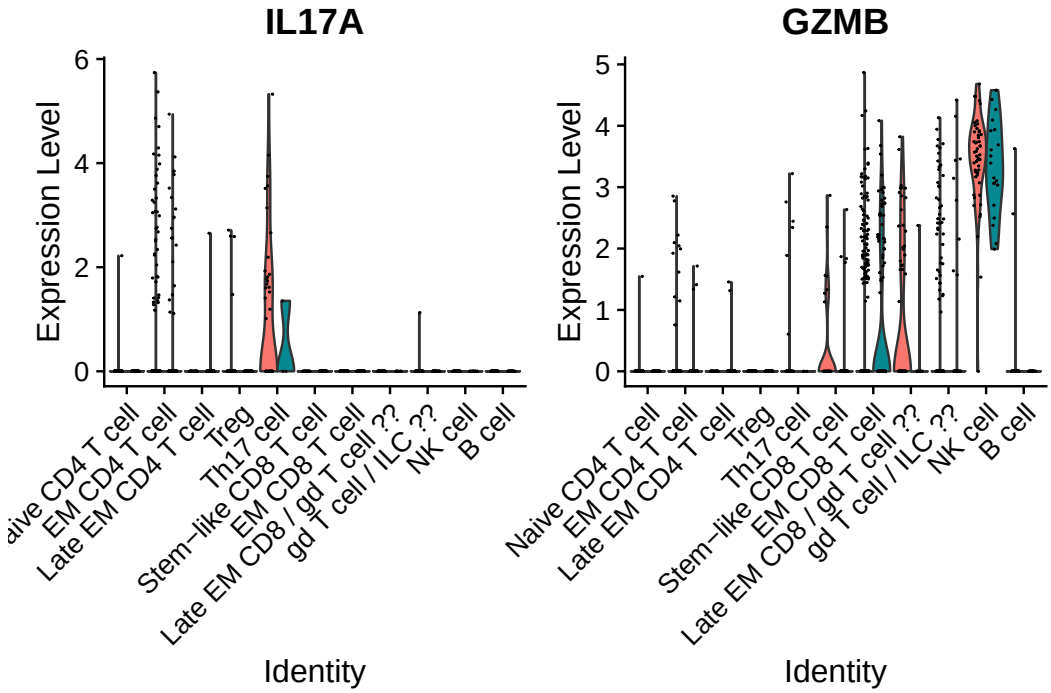



```
# To more aesthetically visualize which cell type expresses the gene, I like to use the scCustomize fun
DotPlot_scCustom(seurat_object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), flip_axes = TRUE
```

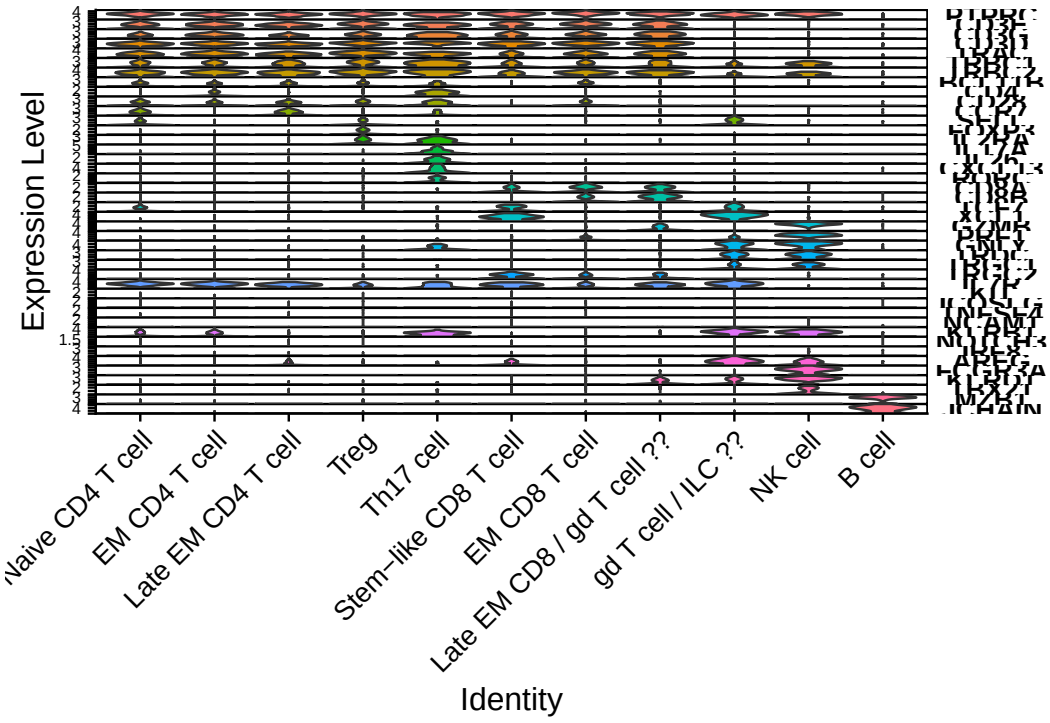


```
# We can visualize these genes by violin plots as well. This confirms what we saw in our dot plots.
VlnPlot(object = NewLymphocyteClusters, features = c("IL17A", "GZMB"), split.by = "stim", pt.size = .00
```

```
## The default behaviour of split.by has changed.
## Separate violin plots are now plotted side-by-side.
## To restore the old behaviour of a single split violin,
## set split.plot = TRUE.
##
## This message will be shown once per session.
```



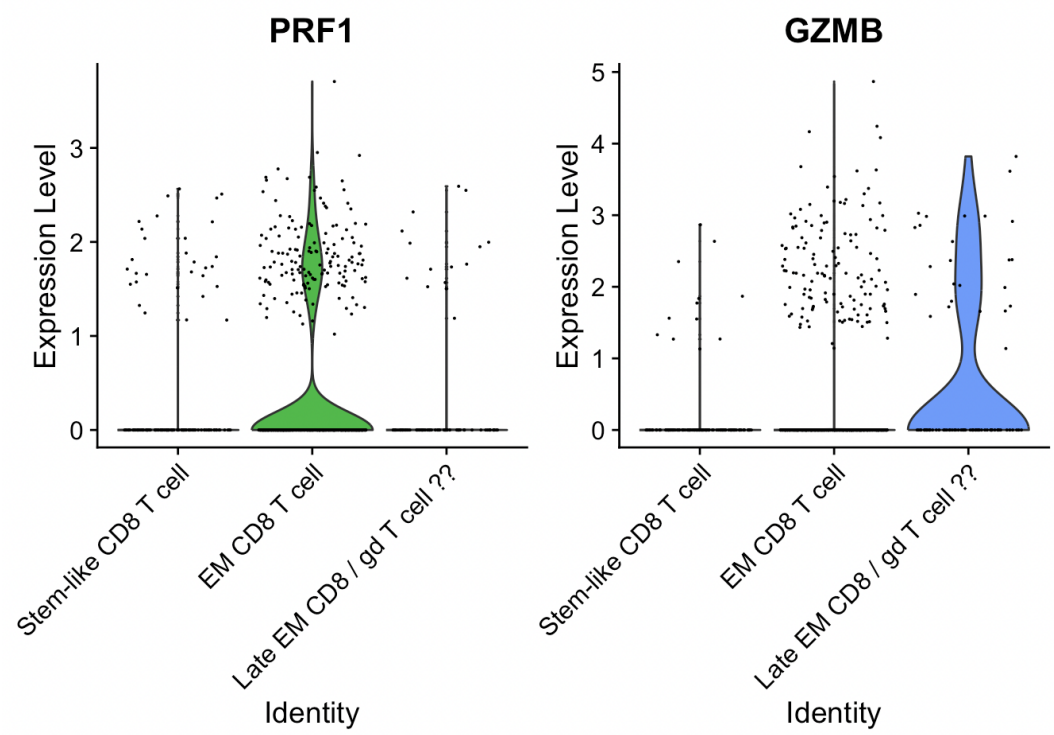

```
# If we want to visualize lots of genes in one graph, I recommend creating a stacked violin plot. Likew

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")
```



Subsetting Clusters

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"))
```



Cell Proportions

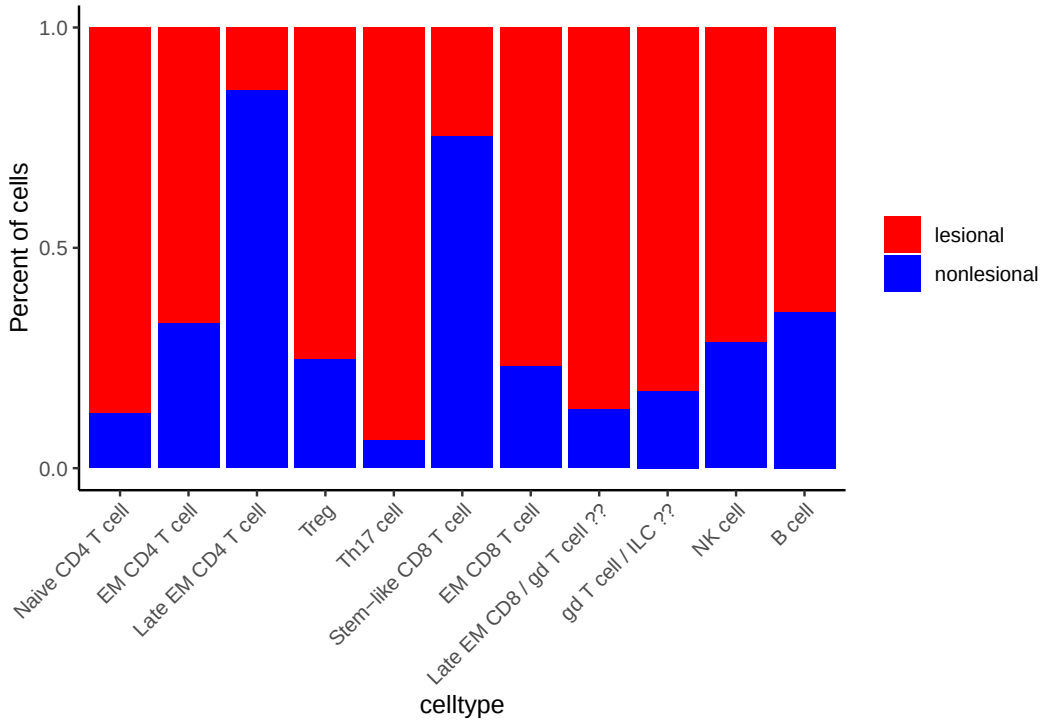
We can also visualize the data and compare cell proportions in each lesion (i.e. whether cell types may be differentially expressed in frequency). To do so, we can use the dittoSeq package to generate stacked bar plots. You can install the package through installing the BiocManager package (ensures the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

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

# BiocManager::install("dittoSeq")

library(BiocManager)
```

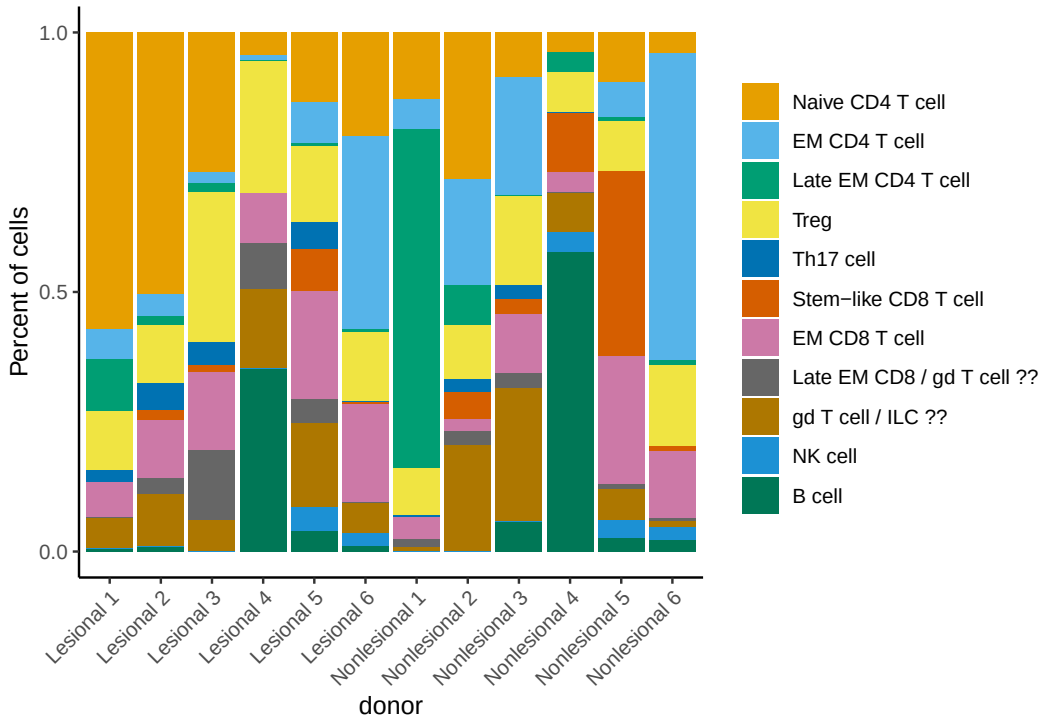
```
## Bioconductor version '3.17' is out-of-date; the current release version '3.18'
##   is available with R version '4.3'; see https://bioconductor.org/install
```

```
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"),
```

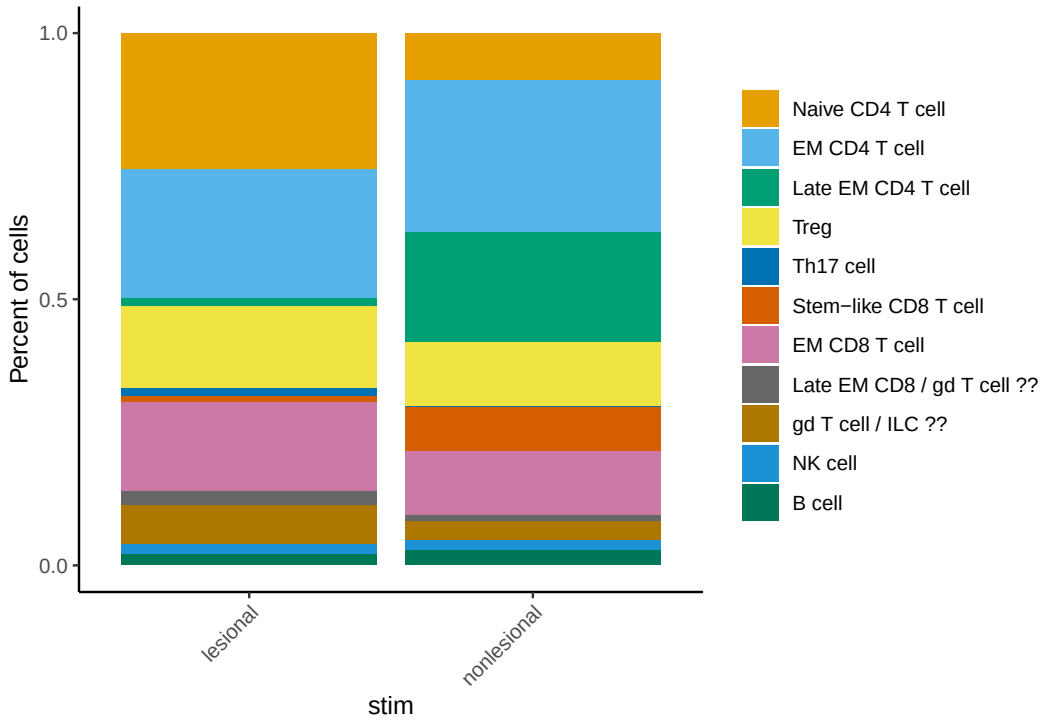
```
    x.reorder = c(7, 2, 5, 11, 10, 9, 3, 6, 4, 8, 1), # make sure to reorder the clusters for visual co
    )
```




```
# 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
    )
```




```
# I find it useful actually view the cell proportions by comparing lesional with nonlesional.

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



With this bar plot, we can see that the lesional has fewer Late EM CD4 T cells, more Th17 cells, fewer Stem-like CD8 T cells, etc. in frequency compared to the nonlesional.

More useful information on the dittoSeq package can be found here.

Volcano Plots

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.

Volcano Plots Using ggplot2

More information on how to edit volcano plots in ggplot2 can be found here.

```
# 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"
```

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

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

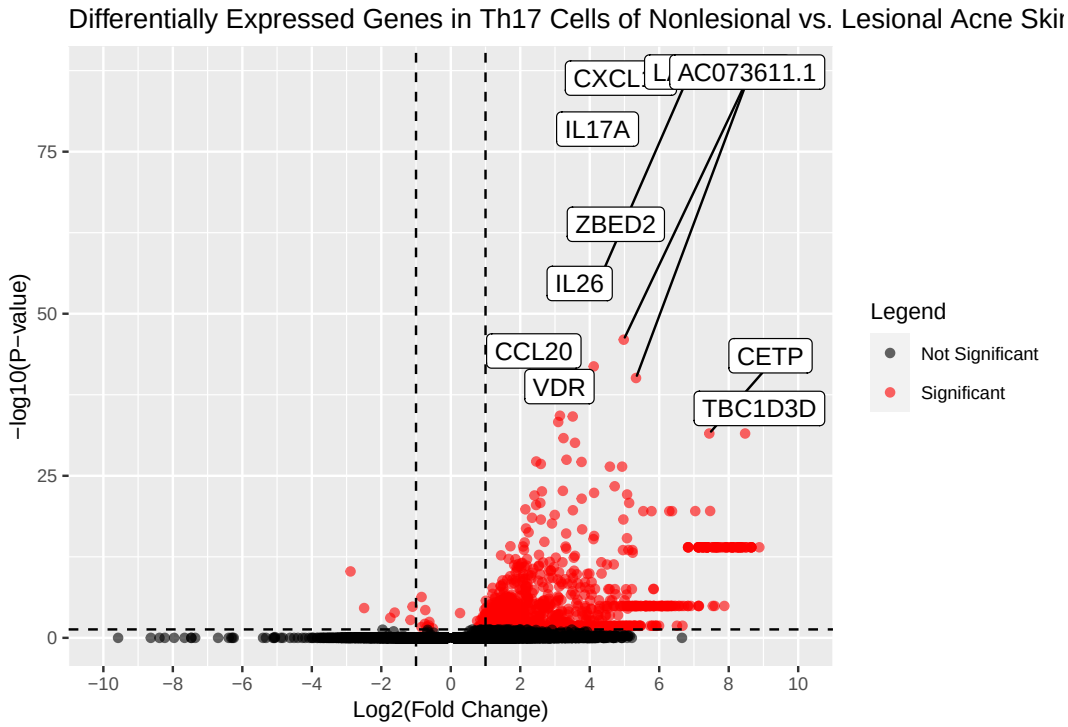
```
## [1] "gene"       "p_val"      "avg_log2FC" "pct.1"      "pct.2"
## [6] "p_val_adj"
```

```r
# 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 Cells of Nonlesional vs. Lesional Acne Skin") +
  scale_x_continuous(breaks = c(seq(-10, 10, 2)),
                     limits = c(-10, 10)) +
  geom_hline(yintercept = -log10(0.05),
             linetype = "dashed") +
  geom_vline(xintercept = c(log2(0.5), log2(2)),
             linetype = "dashed") +
  geom_label_repel(aes(label = gene))
```

```
## Warning: Removed 1 rows containing missing values (‘geom_point()‘).
```

```
## Warning: Removed 1 rows containing missing values (‘geom_label_repel()‘).
```

```
## Warning: ggrepel: 11037 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

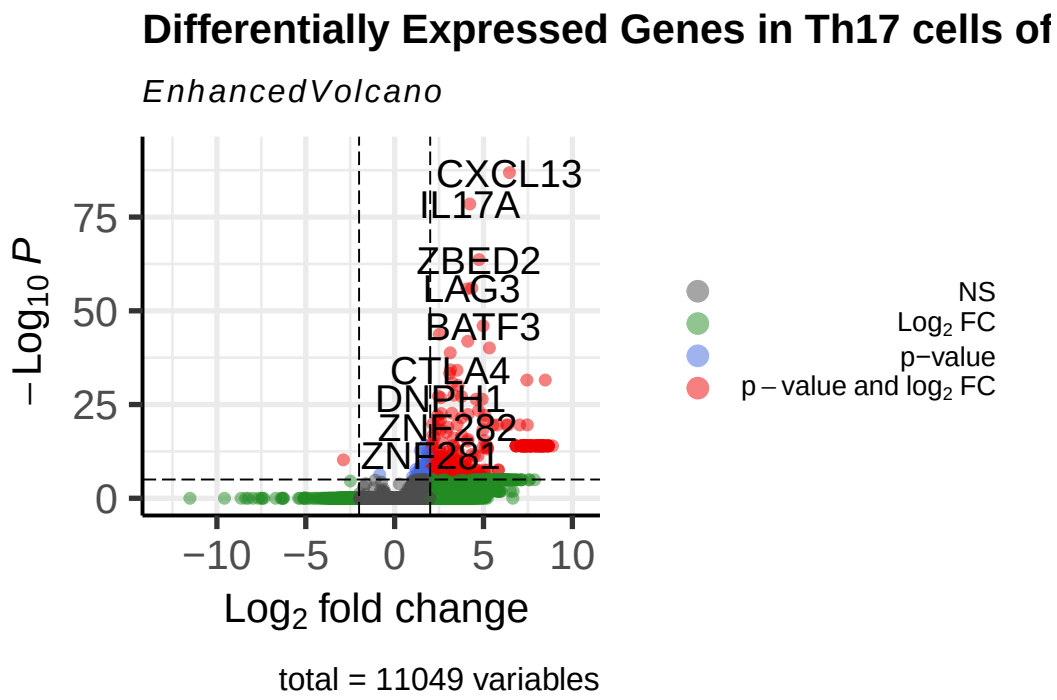



```
# Here, we now see that IL17, CCL20, CXCL13, IL16, etc. are some genes that are more highly expressed i
```

Volcano Plots Using EnhancedVolcano

Another method for making volcano plots to show differential gene expression is by using EnhancedVolcano.

```
# BiocManager::install('EnhancedVolcano')

library(EnhancedVolcano)

# In this case, we use the table we generated from before that we used for volcano plots with ggplot2,

Th17table_edit_rows <- Th17table %>%
  remove_rownames %>%
  column_to_rownames(var="gene")

EnhancedVolcano(Th17table_edit_rows,
                lab = rownames(Th17table_edit_rows),
                x = 'avg_log2FC',
                y = 'p_val_adj',
                title = 'Differentially Expressed Genes in Th17 cells of Nonlesional vs. Lesional Acne S
                pCutoff = 10e-6,
                FCcutoff = 2,
                pointSize = 2.0,
                labSize = 6.0,
                legendPosition = 'right',
                legendLabSize = 12,
                legendIconSize = 4.0
                )
```



EnhancedVolcano allows for easier adjustments to be made for data visualizations, but either way works for constructing volcano plots.

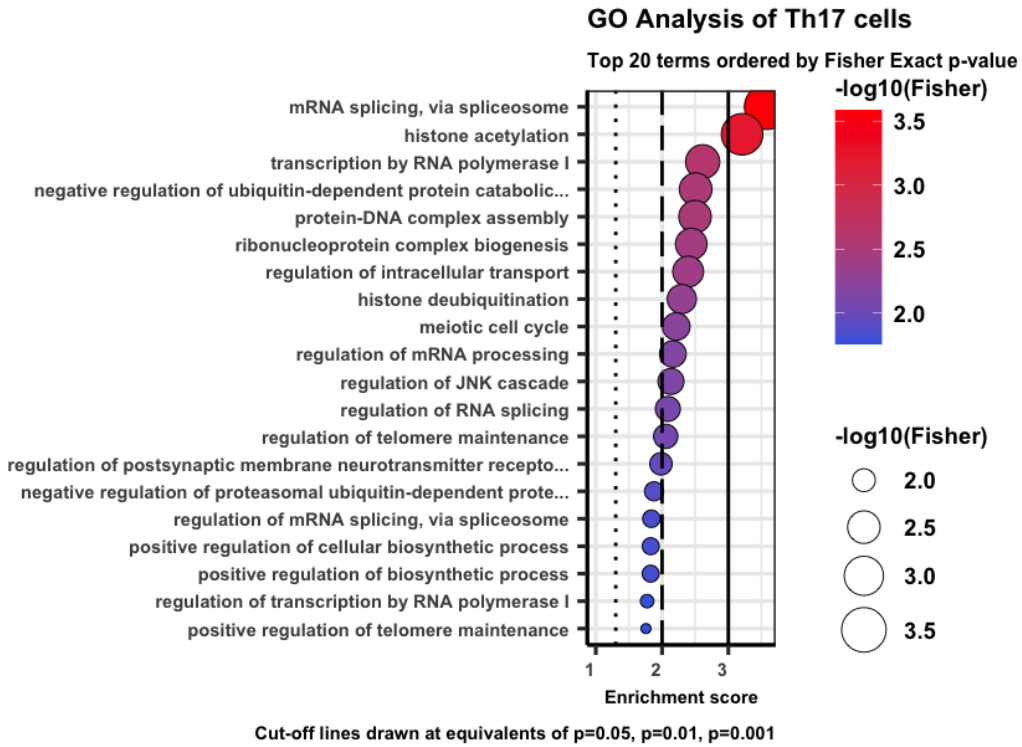
Gene Ontology

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)
```

```r
# Use the topGO package to compare it to [Gene Ontology](http://www.geneontology.org/). Here we are tak

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

GOdata

# 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)

# 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)

# 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)
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 EnrichR for analysis. Make sure you select for genes that are statistically significant (p_adj_val < 0.05).

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. In Part 2, I include how to do pseudotime analysis and how to use the CellChat package for cell-cell interaction analysis.