My Bioinformatics Internship Report: Part II

Universitätsmedizin Mannheim, Institut für Kliniche Chemie

Author: Frano Malinarich G. Supervisors: Prof.Dr. Neumaier and Dr. Ast

2024-09-27

Table of contents

PART II

Introduction

This report summarizes my internship in Bioinformatics called "Single-cell transcriptomic analysis in the identification and characterization of VIREM cells applying knowledge of R, Python and Linux", conducted at the laboratory of Prof. Dr. Neumaier and Dr. Ast (Bioinformatician) at the Universitätsmedizin Mannheim, Institut für kliniche Chemie. The goal of my internship was to identify and characterize a novel class of monocytic cells, capable of recombining and expressing antibody genes and alpha and beta chains, a function previously thought to be a unique feature of B and T cells, respectively. These innate cells are called "variable immunoreceptor–expressing myeloid cells" (VIREMs) Those expressing B cell receptor chains (heavy and light) are called B-VIREMs and those alpha and beta chains, T-VIREMs.

In brief, my internship consisted in identifying and characterizing VIREMs peripheral blood mononuclear cells (PBMCs) from healthy humans, starting from T and B cell clonotypic analysis using the V(D)J library, next, cell clustering by analyzing the gene expression library, followed by data integration of both libraries and, finally, finding possible VIREMs.

For the VDJ and Gene expression analyses, the dataset* was obtained from 10x Genomics dataset repository:

https://www.10xgenomics.com/datasets/human-pbmc-from-a-healthy-donor-10-k-cells-v-2-2-standard-5-0-0

*PBMCs of a healthy male donor aged 27.

To learn how to do VDJ clonotipic analysis using scRepertoire package, go to this website:

https://www.bioconductor.org/packages/release/bioc/vignettes/scRepertoire/inst/doc/vignette.html

To learn how to do Gene expression analyses on human PBMCs using Seurat package, go to this website:

https://satijalab.org/seurat/articles/pbmc3k_tutorial.

In this "Part II", I will continue with the data analysis of the gene-expression library, performing the following:

- Quality control analysis and cell filtering
- Data normalization and other data procedures
- PCA and determination of dimensionality
- Cell clustering
- Cell annotation and differential gene expression per cluster

Packages

```
# Load packages
if (!requireNamespace("pacman")) {
  install.packages("pacman")
}

pacman::p_load(conflicted, dplyr, tidyr, Seurat, scRepertoire, patchwork,
               ggplot2, ggrepel, pROC, flextable, ggraph, stringr)

conflict_prefer("filter", "dplyr")
conflict_prefer("select", "dplyr")
conflict_prefer("mutate", "dplyr")
conflict_prefer("slice", "dplyr")

# Activate libraries
suppressMessages(library(scRepertoire))
# install.packages("conflicted")
```

Setup

```
knitr::opts_chunk$set(fig.width = 7, fig.height = 4)  # Adjust figure size if applicable
knitr::opts_chunk$set(warning = FALSE, message = FALSE)
set_flextable_defaults(font.size = 6, font.color = "black", border.color = "black", border.width
 ↪  = 1, padding = 3)
```

Gene expression analysis

Cell clustering, cell annotation and differential expression gene analysis

The "filtered_feature_bc_matrix" uncompressed folder can be directly downloaded from the 10x Genomics website:

https://www.10xgenomics.com/datasets/human-pbmc-from-a-healthy-donor-10-k-cells-v-2-2-standard-5-0-0

Click on "Gene Expresion - Feature / cell matrix (filtered)" and get the compressed folder:

"sc5p_v2_hs_PBMC_10k_filtered_feature_bc_matrix.tar.gz". Then, by unzipping the folder, to get the "filtered_feature_bc_matrix" uncompressed folder. Inside this folder, there are three compressed folders. Do not uncompress these subfolders

The Read10X function reads in the data from a 10x Genomics experiment that is typically stored in three files: barcodes.tsv.gz, features.tsv.gz (or genes.tsv.gz), and matrix.mtx.gz.

Loading the gene expression data

```
# Define the data directory
data_dir <- "Data_2024_Internship/filtered_feature_bc_matrix"

# Read the PBMC dataset
pbmc.data <- Read10X(data.dir = data_dir)

# Check what data types are available (i.e., what is inside pbmc.data)
names(pbmc.data)
```

```
[1] "Gene Expression"  "Antibody Capture"
```

```
# Create the Seurat object with the Gene Expression data
pbmc <- CreateSeuratObject(counts = pbmc.data$`Gene Expression`, project =
 ↪  "pbmc_geneexpression", min.cells = 3, min.features = 200)
print(pbmc)
```

```
An object of class Seurat
20226 features across 10529 samples within 1 assay
Active assay: RNA (20226 features, 0 variable features)
 1 layer present: counts
```

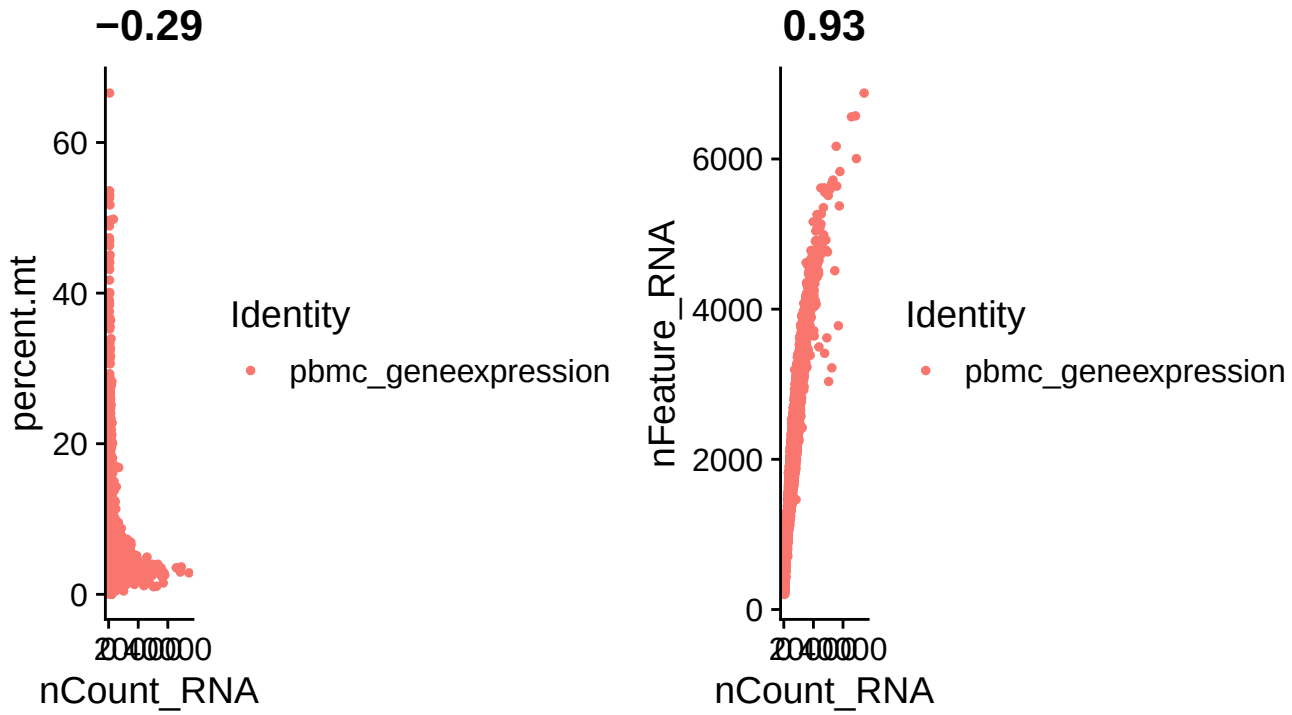
```
# min.cells = 3: This filters out genes that are detected in fewer than 3 cells.
# min.features = 200: This filters out cells that have fewer than 200 features (genes) detected.
 ↪  Cells with fewer than 200 genes are often low-quality cells or droplets that contain ambient
 ↪  RNA rather than actual single cells.
```

QC metrics and selecting cells for further analysis

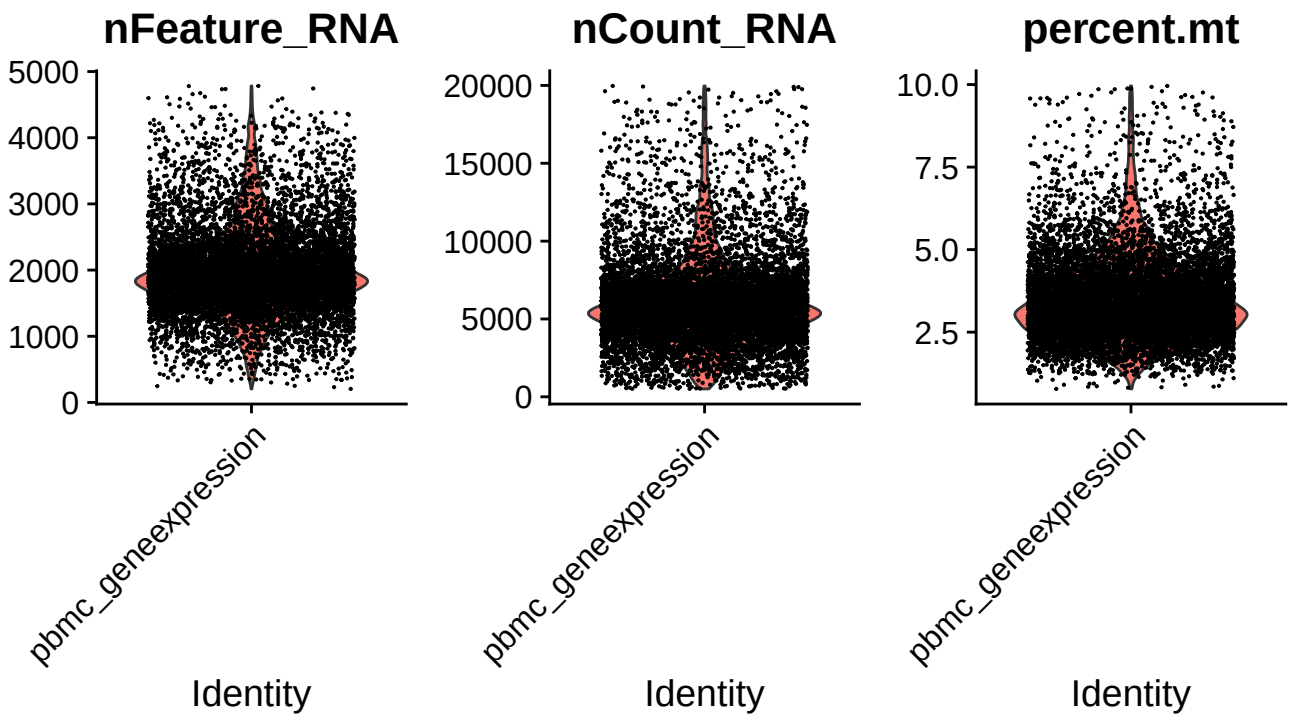
```
# Calculate the percentage of mitochondrial genes
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")
head(pbmc@meta.data, 5) # to see the whole metadata: View(pbmc@meta.data)
```

```
                          orig.ident nCount_RNA nFeature_RNA percent.mt
AAACCTGAGACAGACC-1 pbmc_geneexpression       4750         1981   4.252632
AAACCTGAGCGATAGC-1 pbmc_geneexpression       5767         1761   1.924744
AAACCTGAGCGGCTTC-1 pbmc_geneexpression       4990         1956   2.505010
AAACCTGAGGATCGCA-1 pbmc_geneexpression       6011         2309   3.693229
AAACCTGAGTCACGCC-1 pbmc_geneexpression       7267         2315   2.174212
```

```r
# Visualize QC metrics without normalization
plot <- VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)

# Correlation mitochondria content and number of RNA molecules
plot1 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "percent.mt")

# Correlation number of unique genes per cell and number of RNA molecules
plot2 <- FeatureScatter(pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
plot1 + plot2
```




```r
# High mitochondrial content can indicate stressed or dying cells.

# Filtering out bad cells based on specific criteria related to quality control metrics
 ↪  (nFeature_RNA and percent.mt)
pbmc <- subset(pbmc, subset = nFeature_RNA > 200 & nFeature_RNA < 5000 & percent.mt > 0.5 &
 ↪  percent.mt < 10 & nCount_RNA < 20000)
# Visualize the filtered dataset
qc_plot <- VlnPlot(pbmc, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
qc_plot
```




```
# After the filtering of bad cells, this is how the data now looks like
range(pbmc$nFeature_RNA) #205 4782
```

```
[1]  205 4782
```

```
range(pbmc$nCount_RNA) # 510 19965
```

```
[1]   510 19965
```

```
range(pbmc$percent.mt) # 0.781250 9.957806
```

```
[1] 0.781250 9.957806
```

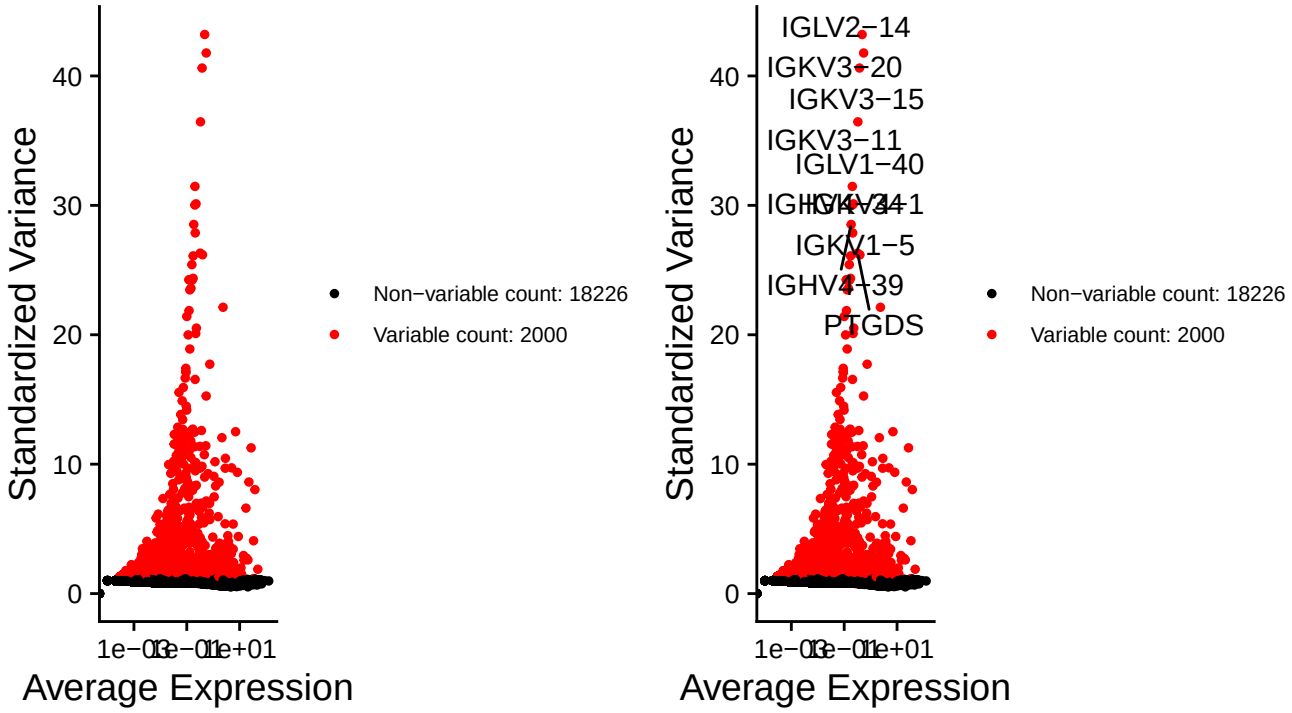
Data normalization, Identification of highly variable features and Data scaling

FindVariableFeatures allows the identification of genes with the highest variation in expression across cells in the Seurat object pbmc.

```
# Seurat object without normalization
# head(pbmc.data, 5) # --> counts
# CD3_TotalC     67  815  45

# Normalized filtered dataset. Use "LogNormalize"
pbmc <- NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000)
pbmc
```

```
An object of class Seurat
20226 features across 9999 samples within 1 assay
Active assay: RNA (20226 features, 0 variable features)
 2 layers present: counts, data
```

```r
# Check the RNA data from the Seurat normalized object.
# head(pbmc[["RNA"]]$data, 5)

# Find highly variable features (genes)
pbmc <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)

# Identify the 10 most highly variable genes
top10 <- head(VariableFeatures(pbmc), 10)

# plot variable features with and without labels
plot1 <- VariableFeaturePlot(pbmc)+
  theme(legend.text = element_text(size = 8), # Reduce legend text size
        axis.text.x = element_text(size = 10), # Adjust x-axis text size
        axis.text.y = element_text(size = 10)) # Adjust y-axis text size
plot2 <- LabelPoints(plot = plot1, points = top10,
                     repel = TRUE, xnudge = 0, ynudge = 0) +
  theme(legend.text = element_text(size = 8)) # Reduce legend text size
plot1 + plot2
```




```r
# Scaling pbmc dataset
all.genes <- rownames(pbmc)
pbmc <- ScaleData(pbmc, features = all.genes)
```

```
# View(pbmc@commands) # To see that Seurat object has been processed
```

Linear dimensional reduction (PCA)

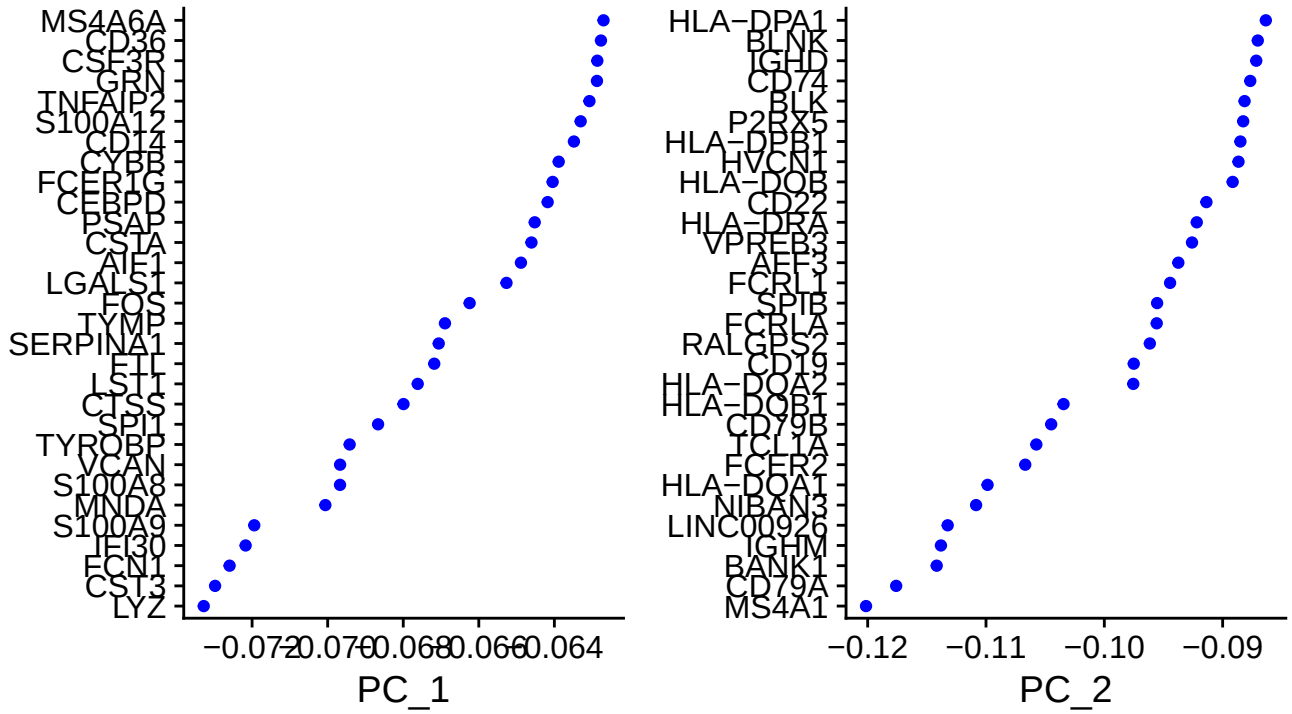
```
# Perform PCA on the scaled data.
pbmc <- RunPCA(pbmc, features = VariableFeatures(object = pbmc))
pbmc
```

```
An object of class Seurat
20226 features across 9999 samples within 1 assay
Active assay: RNA (20226 features, 2000 variable features)
 3 layers present: counts, data, scale.data
 1 dimensional reduction calculated: pca
```

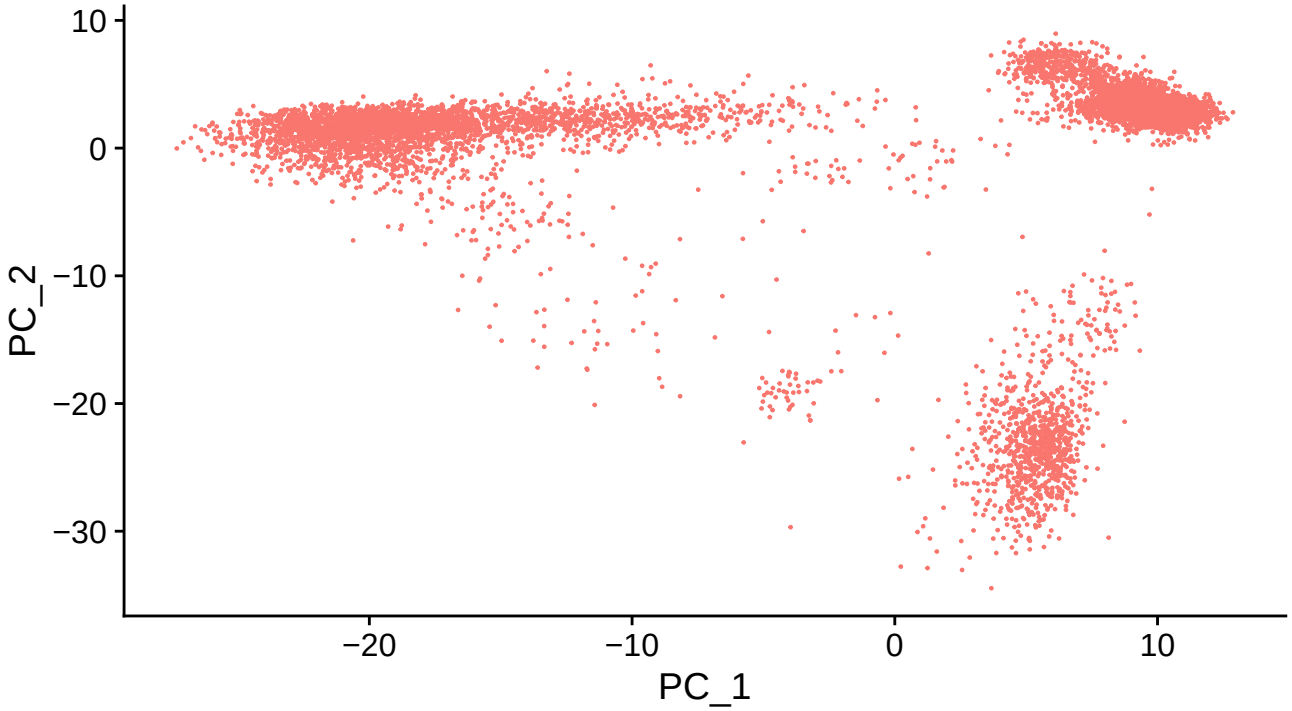
```
# Examine and visualize PCA results a few different ways
print(pbmc[["pca"]], dims = 1:5, nfeatures = 10)
```

```
PC_ 1
Positive:  IFITM1, CD3E, LTB, IL32, ETS1, TCF7, CD3D, LDHB, IL7R, PCED1B-AS1
Negative:  LYZ, CST3, FCN1, IFI30, S100A9, MNDA, S100A8, VCAN, TYROBP, SPI1
PC_ 2
Positive:  IL32, IFITM1, CD3E, CD7, GIMAP5, CD3D, CD247, ANXA1, IL7R, S100A4
Negative:  MS4A1, CD79A, BANK1, IGHM, LINC00926, NIBAN3, HLA-DQA1, FCER2, TCL1A, CD79B
PC_ 3
Positive:  LEF1, CCR7, TCF7, MAL, LDHB, CD3E, CAMK4, IL7R, TRABD2A, PIK3IP1
Negative:  NKG7, GNLY, CST7, KLRD1, GZMA, GZMB, PRF1, FGFBP2, KLRF1, SPON2
PC_ 4
Positive:  RPS18, CYBA, CD37, LSP1, VIM, ITGB2, JUN, JUNB, S100A6, S100A10
Negative:  CAVIN2, PRKAR2B, TUBB1, GNG11, PPBP, CLU, SPARC, PF4, GP1BB, MPIG6B
PC_ 5
Positive:  LILRA4, CLEC4C, SERPINF1, LRRC26, IL3RA, PLD4, TPM2, SCT, DNASE1L3, PPM1J
Negative:  LINC00926, MS4A1, FCER2, CD79A, CD79B, BANK1, CD19, FCRL1, FCRL3, PDLIM1
```

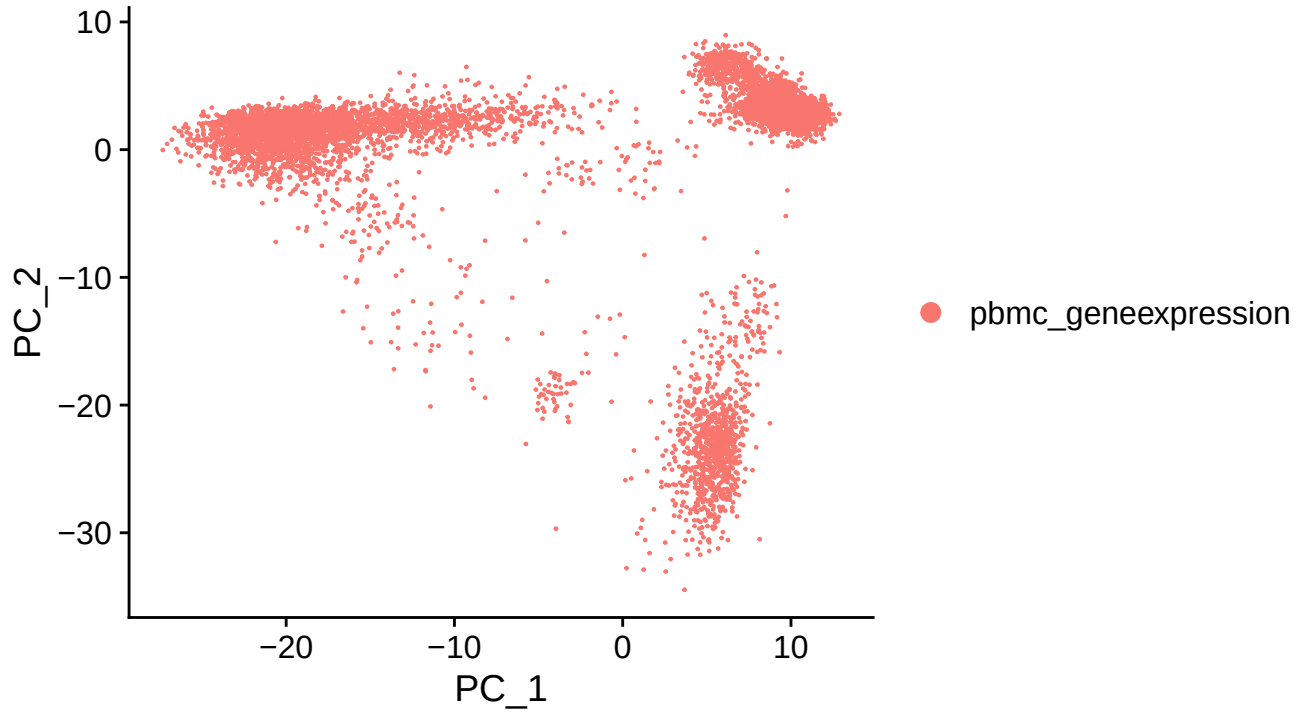
```
# Visualize the top genes contributing to PC_1 and PC_2
VizDimLoadings(pbmc, dims = 1:2, reduction = "pca")
```



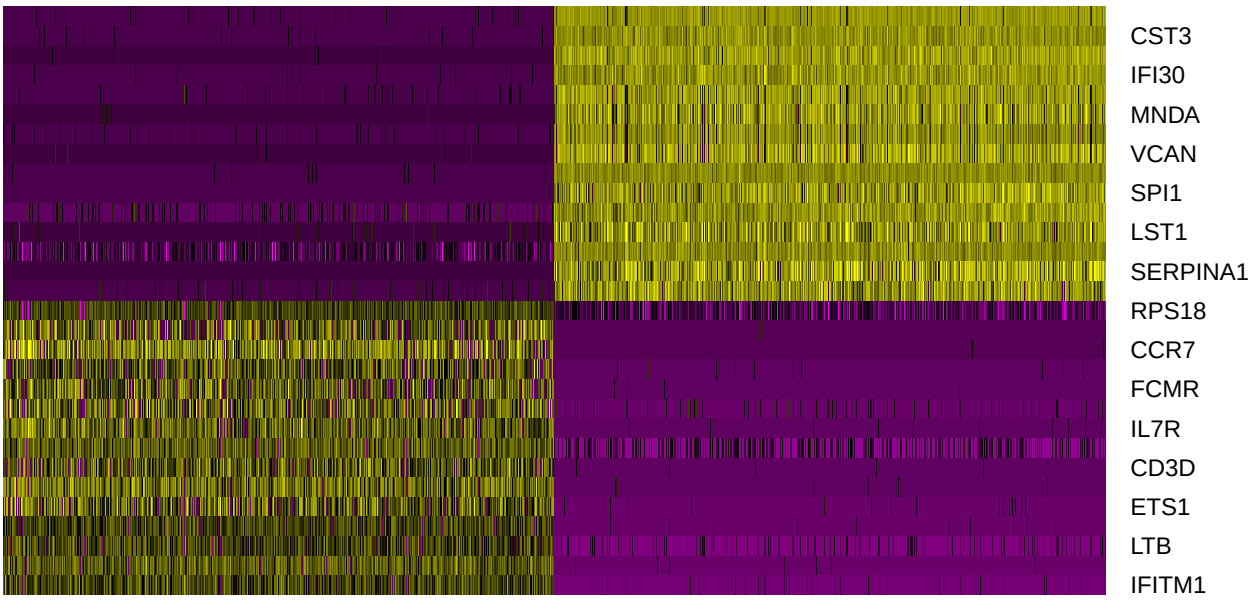

```
# Examine and visualize PCA results a few different ways
DimPlot(pbmc, reduction = "pca") + NoLegend()
```



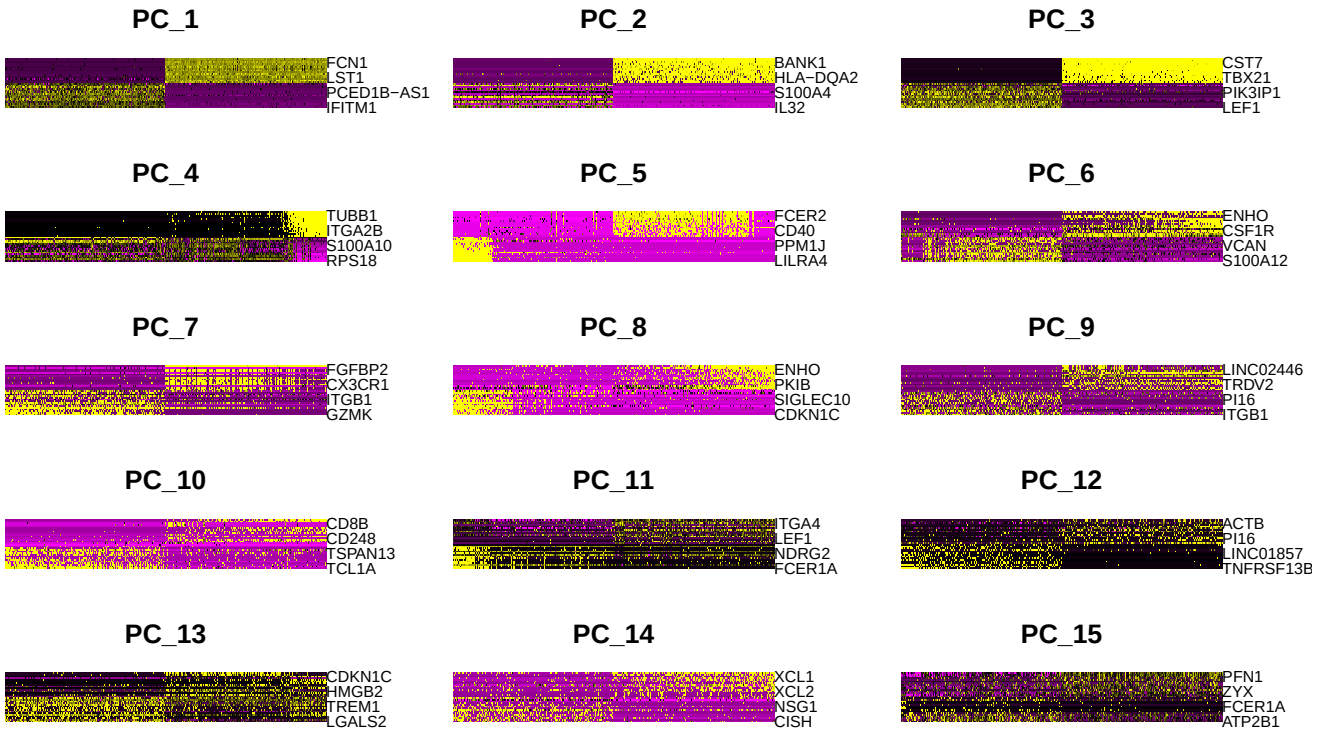
```
DimPlot(pbmc, reduction = "pca", dims = c(1,2))
```




```
# DimPlot(pbmc, reduction = "pca", dims = c(2,10))
# DimPlot(pbmc, reduction = "pca", dims = c(1,3))

# DimHeatmap(pbmc, dims = 1, cells = 500, balanced = TRUE)
DimHeatmap(pbmc, dims = 1, cells = 2638, balanced = TRUE)
```




```
# DimHeatmap(pbmc, dims = 2, cells = 2638, balanced = TRUE)
DimHeatmap(pbmc, dims = 1:15, cells = 500, balanced = TRUE)
```



```
# Rows: Represent individual genes.
# Columns: Represent individual cells.
# Color: The color intensity represents the expression level of each gene in each cell.
 ↪  Typically, a color scale is used, with one color representing low expression and another
 ↪  representing high expression. In your plot, it seems that yellow represents high expression
 ↪  and purple represents low expression.
# cells = 500 argument in your code indicates that only 500 cells are plotted.

# Extract the loadings for the first two principal components
loadings <- Loadings(pbmc, reduction = "pca")

# Extract the loadings for PC_1 and PC_2
pc1_loadings <- loadings[, "PC_1"]
pc2_loadings <- loadings[, "PC_2"]

# Get the top contributing genes for PC_1 and PC_2
top_genes_pc1 <- names(sort(abs(pc1_loadings), decreasing = TRUE)[1:30])
top_genes_pc2 <- names(sort(abs(pc2_loadings), decreasing = TRUE)[1:30])
top_genes_pc1
```

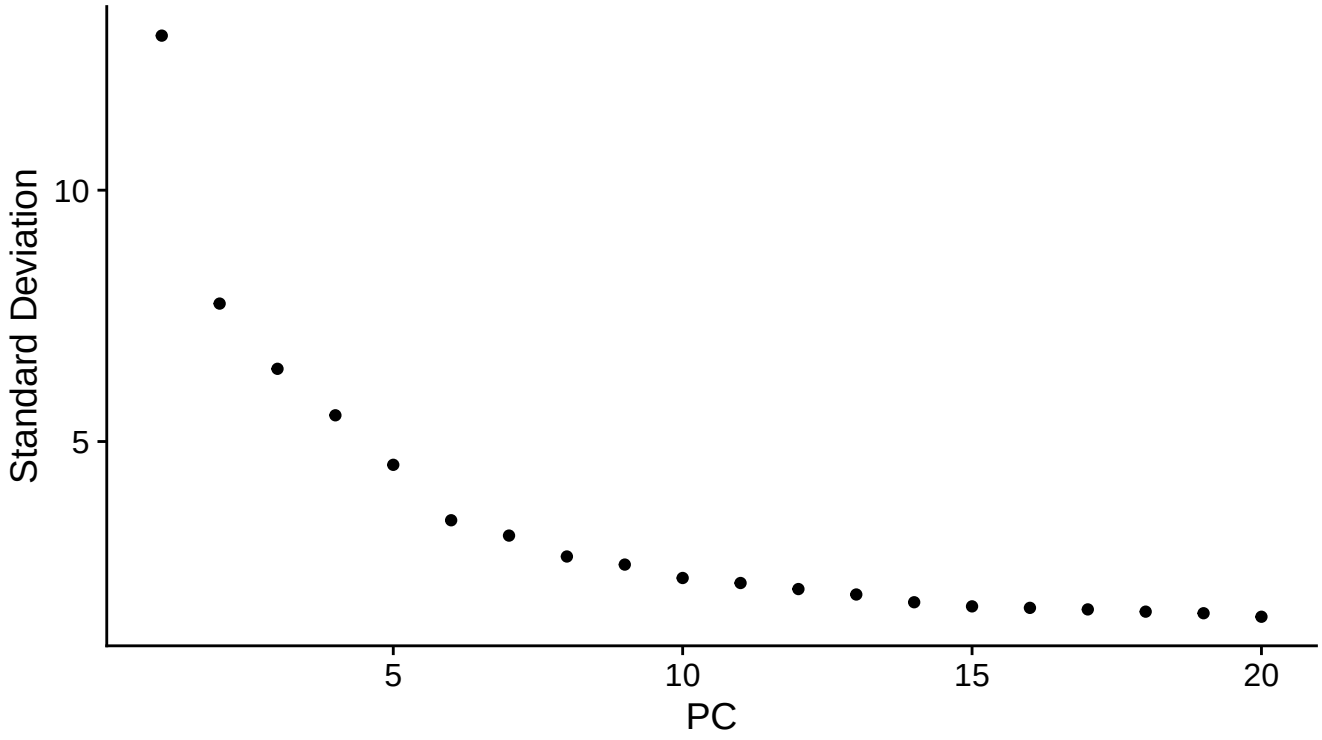
```
 [1] "LYZ"      "CST3"     "FCN1"     "IFI30"    "S100A9"   "MNDA"
 [7] "S100A8"   "VCAN"     "TYROBP"   "SPI1"     "CTSS"     "LST1"
[13] "FTL"      "SERPINA1" "TYMP"     "FOS"      "LGALS1"   "AIF1"
[19] "CSTA"     "PSAP"     "CEBPD"    "FCER1G"   "CYBB"     "CD14"
[25] "S100A12"  "TNFAIP2"  "GRN"      "CSF3R"    "CD36"     "MS4A6A"
```

```
top_genes_pc2
```

```
 [1] "MS4A1"     "CD79A"     "BANK1"     "IGHM"      "LINC00926" "NIBAN3"
 [7] "HLA-DQA1"  "FCER2"     "TCL1A"     "CD79B"     "HLA-DQB1"  "HLA-DQA2"
[13] "CD19"      "RALGPS2"   "FCRLA"     "SPIB"      "FCRL1"     "AFF3"
[19] "VPREB3"    "HLA-DRA"   "CD22"      "HLA-DOB"   "HVCN1"     "HLA-DPB1"
[25] "P2RX5"     "BLK"       "CD74"      "IGHD"      "BLNK"      "HLA-DPA1"
```

Determining the 'dimensionality' of the dataset

```
# Determine the number of PCs to retain for your analysis.
ElbowPlot(pbmc)
```




```
# Early PCs: The first few PCs often capture the major patterns of variation in the data, which
 ↪  are more likely to be biologically meaningful.
# Later PCs: As you move to later PCs, they explain a smaller and smaller proportion of the
 ↪  total variance. These PCs are more likely to capture noise and less relevant biological
 ↪  information.
```

Cell clustering

- **FindNeighbors**: Constructs a k-nearest neighbor graph based on the PCA (Principal Component Analysis) results (in this case, PCs 1-12). FindNeighbors calculates how "close" or "similar" the cells are to each other. These relationships are stored in a graph structure used for downstream clustering. The argument `dims = 1:12` indicates that the first 12 PCs are used, capturing most of the meaningful variation in gene expression.
- **FindClusters**: Groups cells into clusters based on the KNN graph. Once the neighbor graph is constructed, FindClusters uses a graph-based community detection algorithm to partition the cells into distinct clusters. Essentially, it groups together cells that are more similar (closer) to each other in terms of their gene expression profiles. The resolution parameter in FindClusters controls the granularity of clustering, with a higher resolution leading to more clusters.
- **RunUMAP**: Used to visualize the results in a 2D space. UMAP (Uniform Manifold Approximation and Projection) is a dimensionality reduction technique often used in single-cell RNA-seq analysis to visualize cell clusters in a lower-dimensional space.
- **Idents**: Retrieves the cluster identities of cells.

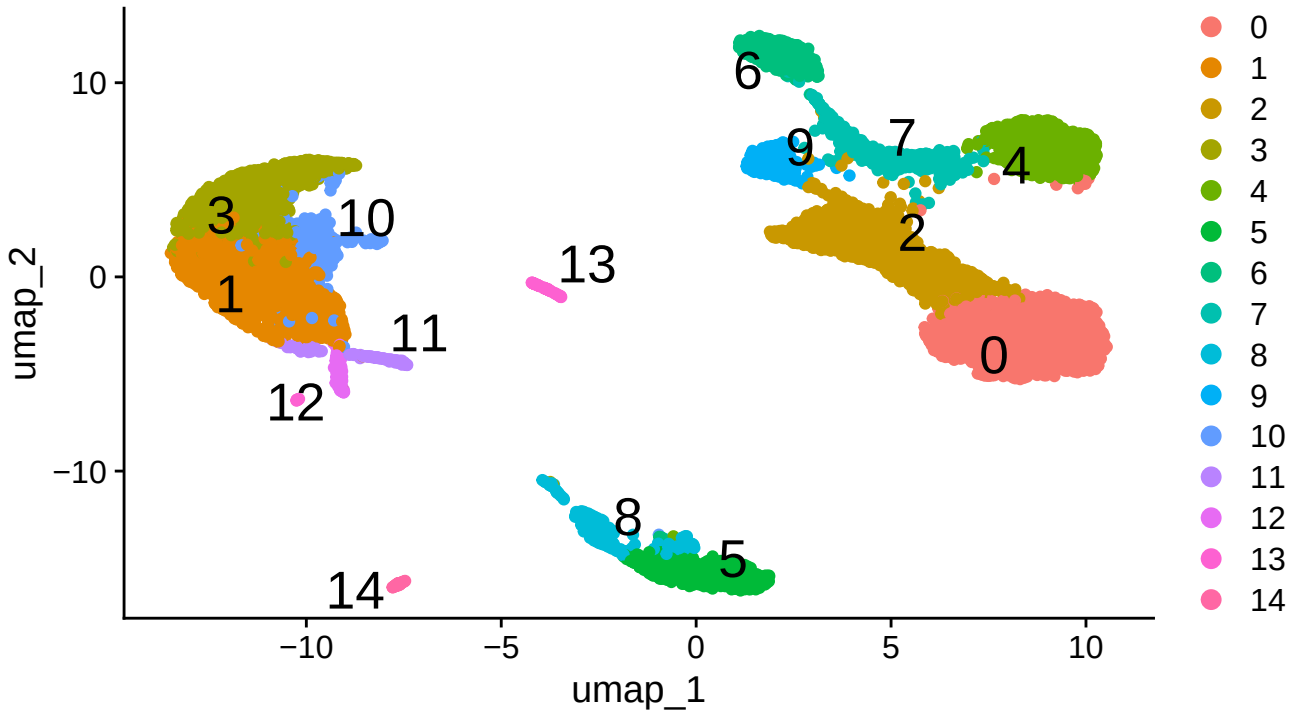
```
# Clustering analysis: find neighbors and clusters, then run the UMAP
pbmc <- FindNeighbors(pbmc, dims = 1:12)
pbmc <- FindClusters(pbmc, resolution = 0.4)
```

```
Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck

Number of nodes: 9999
Number of edges: 350454

Running Louvain algorithm...
Maximum modularity in 10 random starts: 0.9298
Number of communities: 15
Elapsed time: 2 seconds
```

```
pbmc <- RunUMAP(pbmc, dims = 1:12)
# pbmc <- RunTSNE(pbmc, dims = 1:20) # alternative visualization
# pbmc <- RunPCA(pbmc, dims = 1:20) # alternative visualization


# Look at cluster IDs of the first 5 cells
head(Idents(pbmc), 5)
```

```
AAACCTGAGACAGACC-1 AAACCTGAGCGATAGC-1 AAACCTGAGCGGCTTC-1 AAACCTGAGGATCGCA-1
                 1                  0                  1                  1
AAACCTGAGTCACGCC-1
                 2
Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

```
# Create UMAP plot without labels
umap1 <- DimPlot(pbmc, reduction = "umap", label = FALSE, pt.size = 1.5) # 'reduction = ' "tsne"
 ↪  or "pca"
# Add repelled labels with custom size
umap1 <- LabelClusters(umap1, id = "ident", repel = TRUE, size = 7)
umap1 # display the plot
```




```
head(pbmc@meta.data,5)
```

```
                       orig.ident nCount_RNA nFeature_RNA percent.mt
AAACCTGAGACAGACC-1 pbmc_geneexpression       4750         1981   4.252632
AAACCTGAGCGATAGC-1 pbmc_geneexpression       5767         1761   1.924744
AAACCTGAGCGGCTTC-1 pbmc_geneexpression       4990         1956   2.505010
AAACCTGAGGATCGCA-1 pbmc_geneexpression       6011         2309   3.693229
AAACCTGAGTCACGCC-1 pbmc_geneexpression       7267         2315   2.174212
                   RNA_snn_res.0.4 seurat_clusters
AAACCTGAGACAGACC-1               1               1
AAACCTGAGCGATAGC-1               0               0
AAACCTGAGCGGCTTC-1               1               1
AAACCTGAGGATCGCA-1               1               1
AAACCTGAGTCACGCC-1               2               2
```

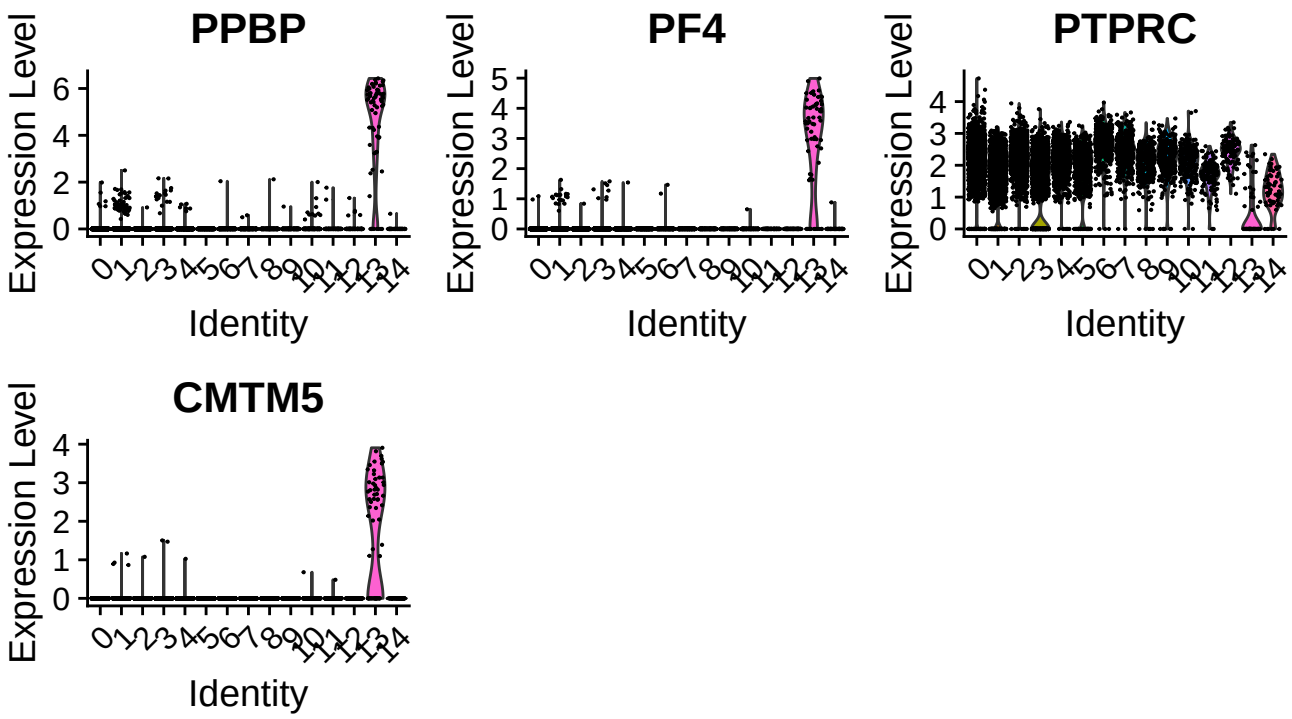
Assigning cell type identity to clusters and DEG analysis

Use well-known cell lineage-markers to distinguish immune cell populations and plot with VlnPlot.

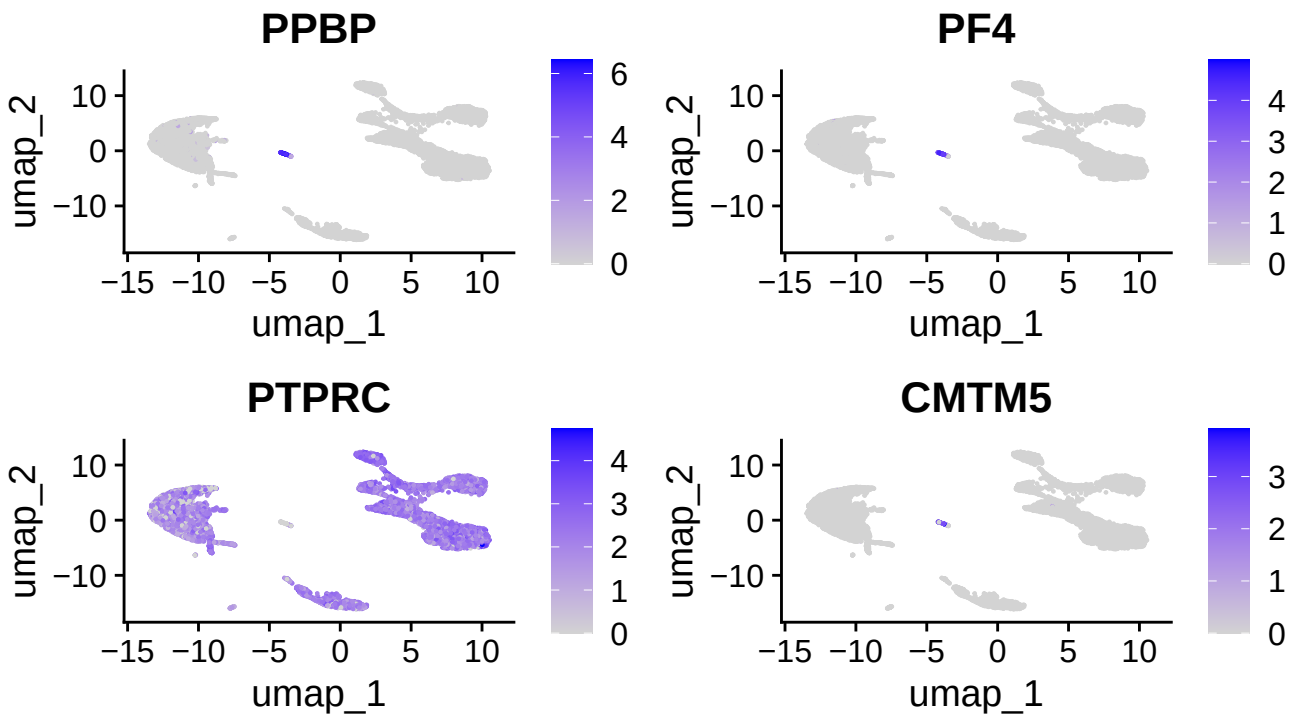
```
# Check genes using regular expression
genes <- pbmc.data$`Gene Expression`@Dimnames[[1]]
choose_genes <- grep("^IGHE", genes, perl = TRUE, value = TRUE)
choose_genes
```

```
[1] "IGHEP2" "IGHE"   "IGHEP1"
```

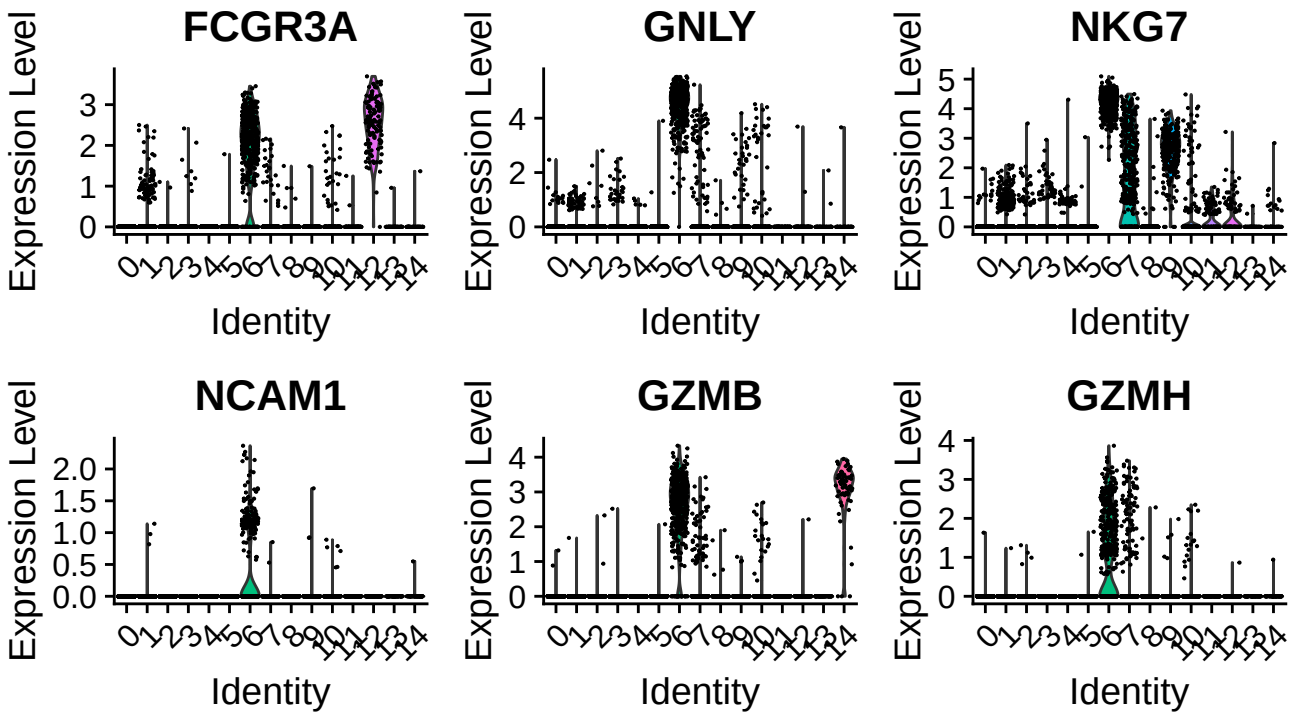
```
# Platelets
platelet <- VlnPlot(pbmc, features = c("PPBP", "PF4", "PTPRC", "CMTM5")) # Cluster 13
platelet # High expression of PPBP, PF4, CMTM5 and low expression of CD45 (PTPRC)
```



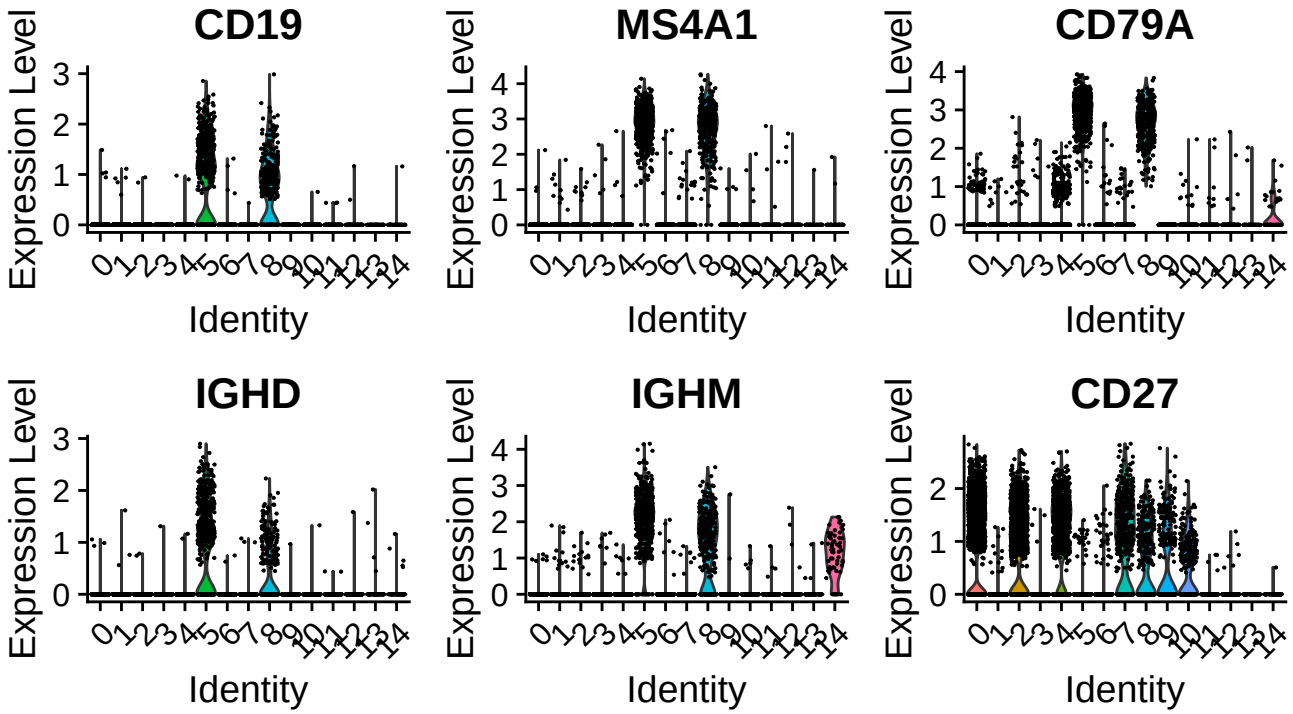

```
FeaturePlot(pbmc, features = c("PPBP", "PF4", "PTPRC", "CMTM5"))
```



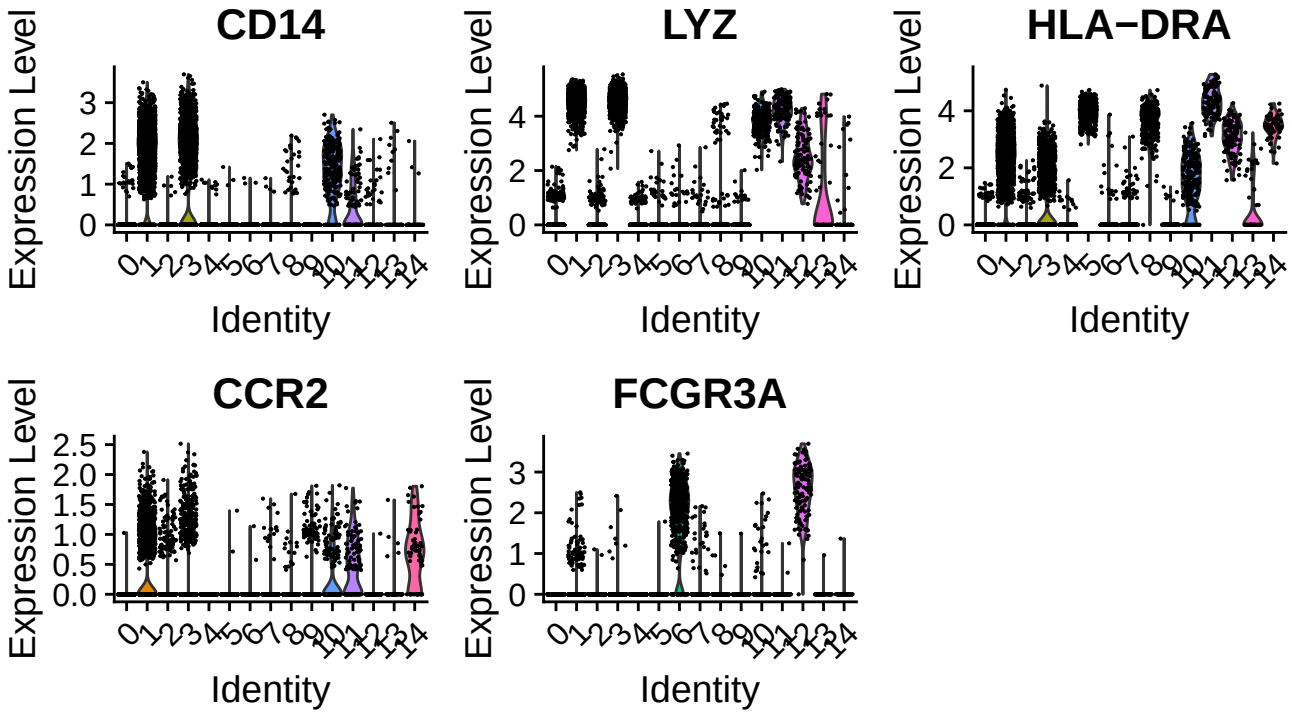

```
# NK
NK <- VlnPlot(pbmc, features = c("FCGR3A", "GNLY", "NKG7",
                                 "NCAM1", "GZMB","GZMH")) # Cluster 6
NK # NCAM1 --> CD56 / FCGR3A --> CD16
```



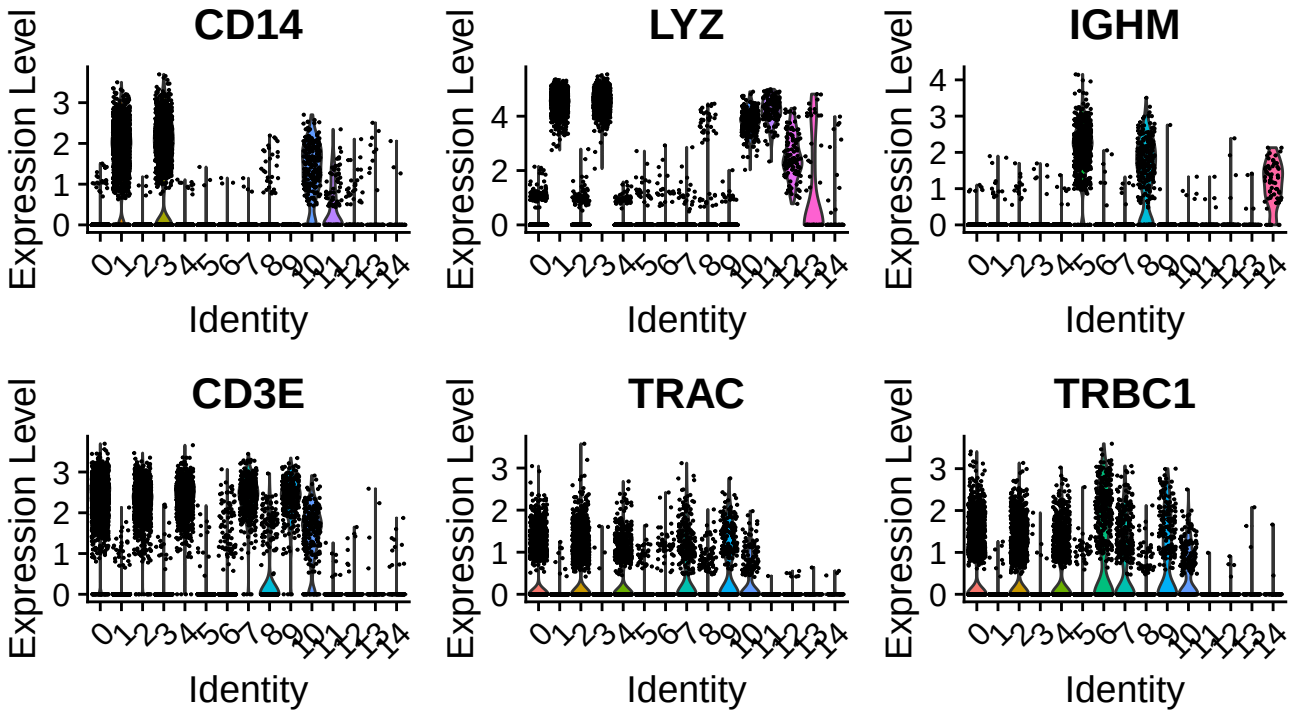

```
# FeaturePlot(pbmc, features = c("FCGR3A", "GNLY", "NKG7","NCAM1", "GZMB","GZMH"))
b_cell_common <- VlnPlot(pbmc, features = c("CD19","MS4A1", "CD79A",
                                            "IGHD", "IGHM","CD27")) # Clusters 5, 8
b_cell_common # MS4A1 --> CD20
```



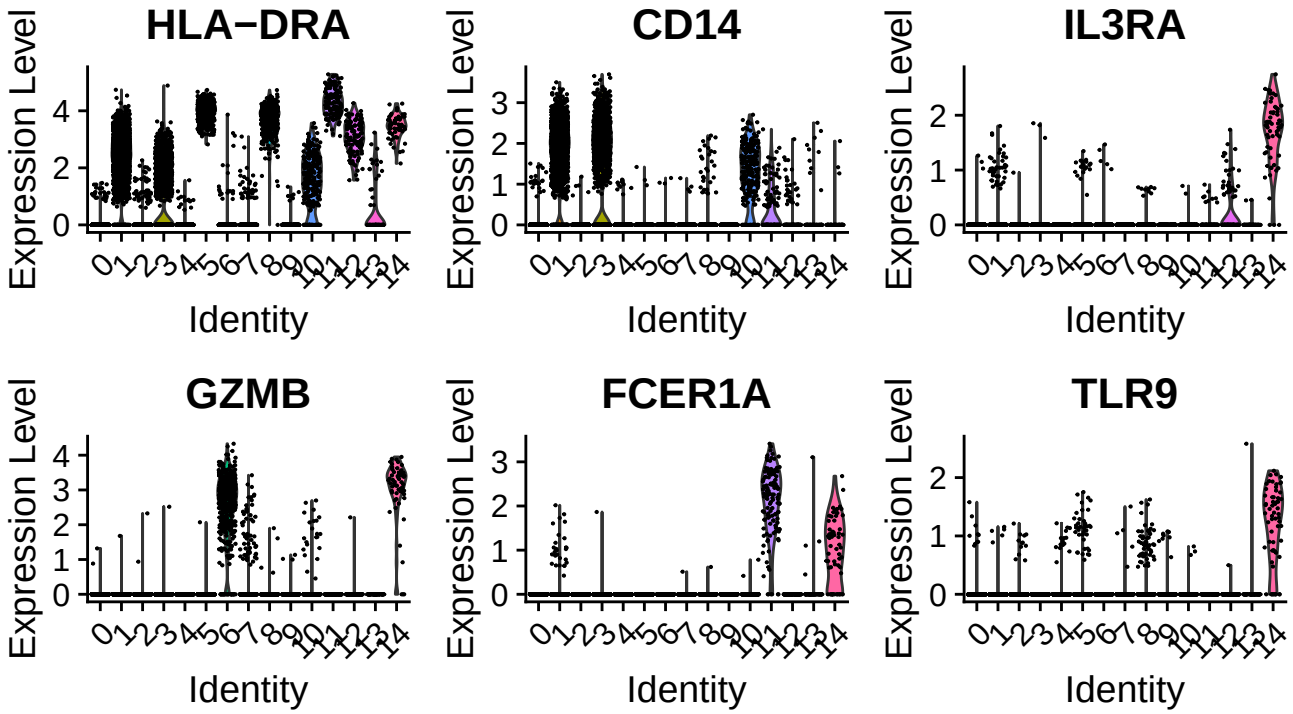

```
b_cell_naive <- VlnPlot(pbmc, features = c("CD19","MS4A1", "CD79A",
                                           "IGHD", "IGHM","CD27")) # Cluster 5. CD27neg
b_cell_active_memory <- VlnPlot(pbmc, features = c("CD19",
                     "IGHD", "IGHM","CD27")) # 8 IgDlow IgMpos, CD27pos.

monocyte_common <- VlnPlot(pbmc, features = c("CD14","LYZ","FCGR3A"))
mono_classic_1 <- VlnPlot(pbmc, features = c("CD14","LYZ","HLA-DRA",
                                           "CCR2", "FCGR3A")) # Cluster 1
mono_classic_1
```



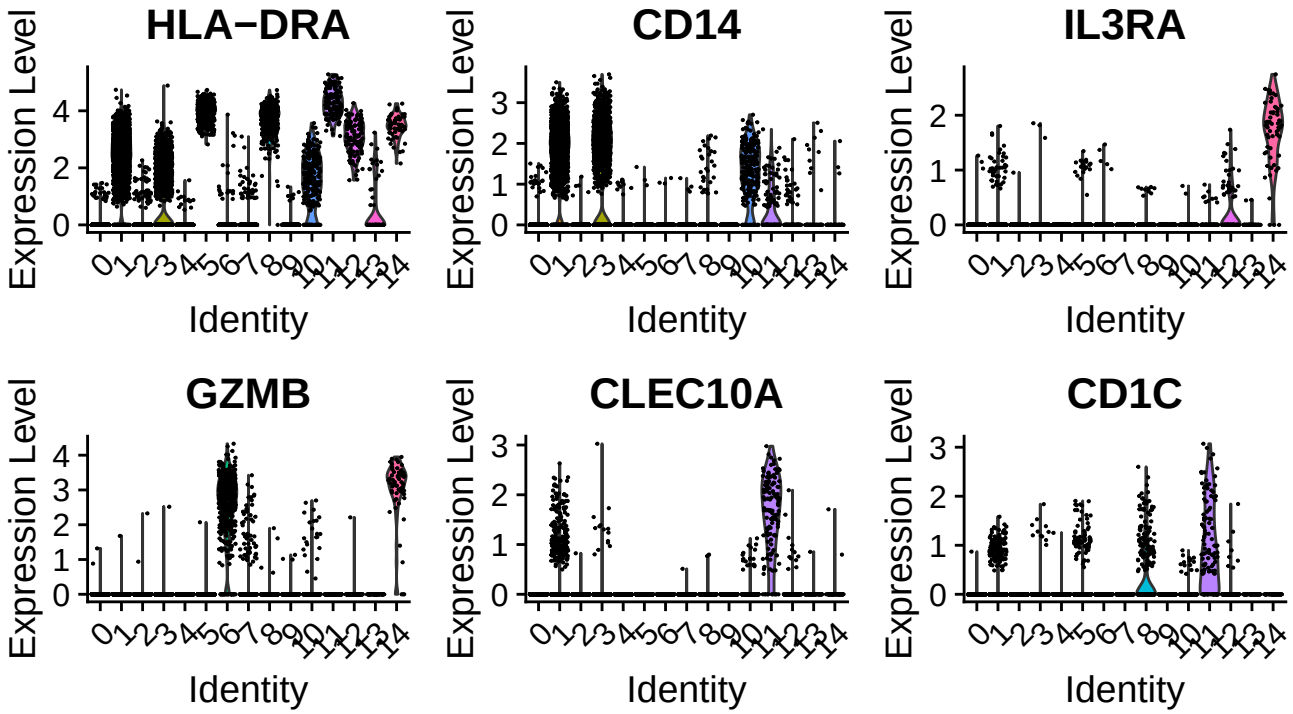

```
non_classic_mono_12 <- VlnPlot(pbmc, features = c("CD14","LYZ","FCGR3A",
                                                  "CCR2","ITGAM", "CD36")) # Cluster 12
# CD14pos, CD16 (FCGR3A)neg, CCR2pos, FCGR2Apos, HLA-DRAint
mono_trans1 <- VlnPlot(pbmc, features = c("CD14","LYZ","FCGR3A",
                                          "CCR2", "FCGR2A")) # Cluster 3
# CD14pos, CD16 (FCGR3A)neg, CCR2int/low, FCGR2Aint/low, HLA-DRAint
mono_trans2 <- VlnPlot(pbmc, features = c("CD14", "LYZ", "IGHM",
                                          "CD3E","TRAC","TRBC1")) # Cluster 10
mono_trans2
```



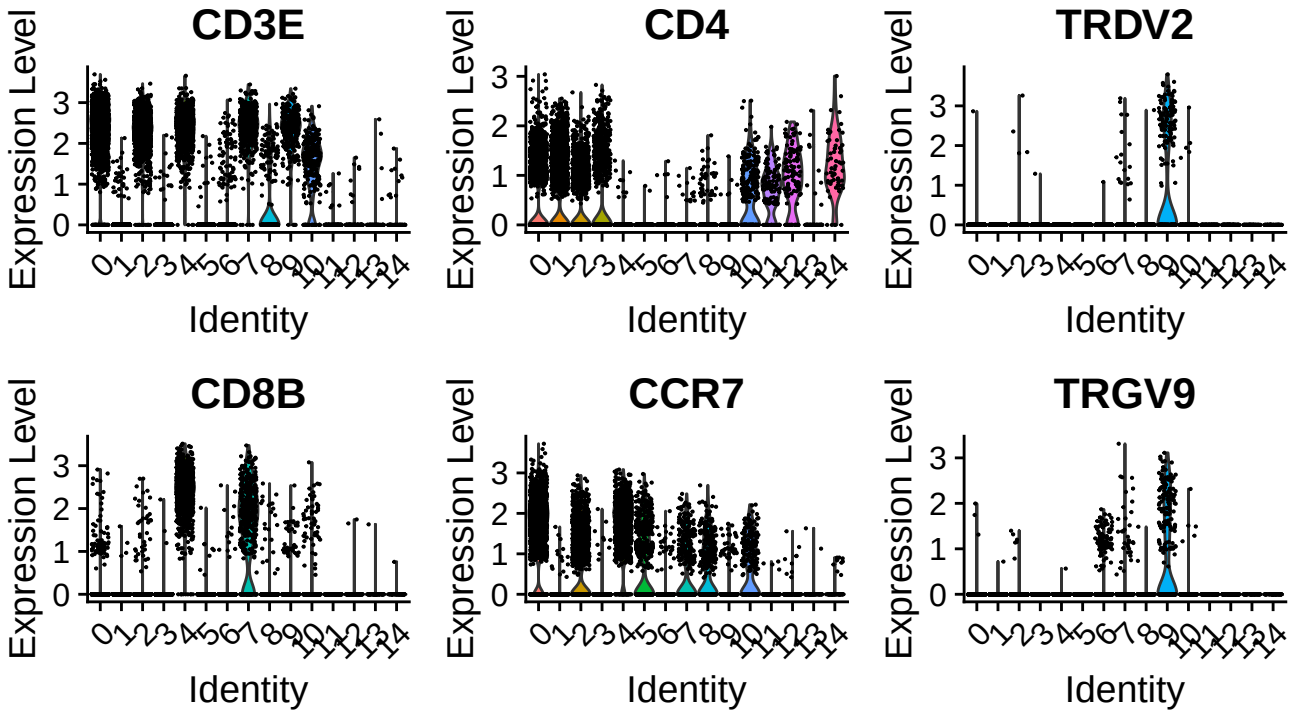

```
# CONCLUSION: Mono_trans2 are expressing TCR complex markers and CD4, but not markers of CD19
 ↪  nor Igs.

pdc_14 <- VlnPlot(pbmc, features = c("HLA-DRA", "CD14", "IL3RA",
                                     "GZMB", 'FCER1A', "TLR9")) # Cluster 14
pdc_14
```



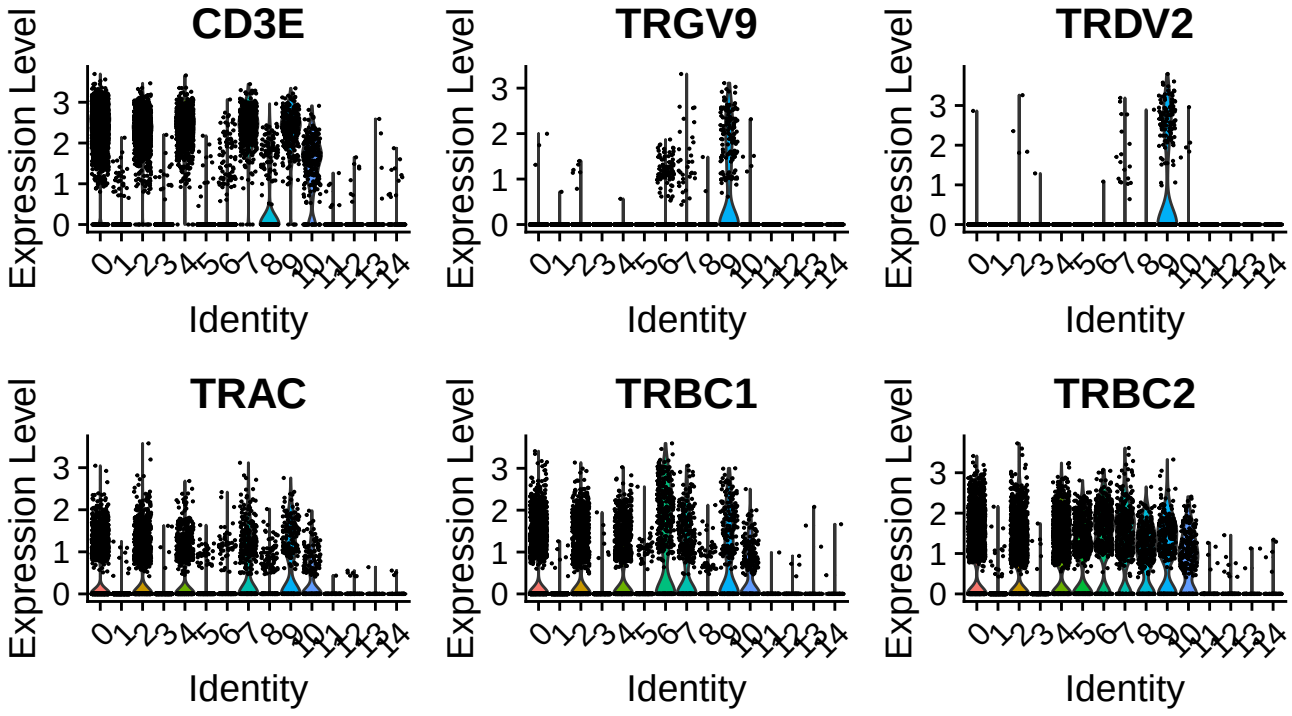

```
# Cluster 14: HLA-DRAhigh, CD14neg, IL3RApos, GZMBpos, CD1cneg, CLEC10Aneg, ITGAXneg
CD1c_DC_11 <- VlnPlot(pbmc, features = c("HLA-DRA", "CD14", "IL3RA",
                                         "GZMB", 'CLEC10A', "CD1C")) # Cluster 11
CD1c_DC_11
```



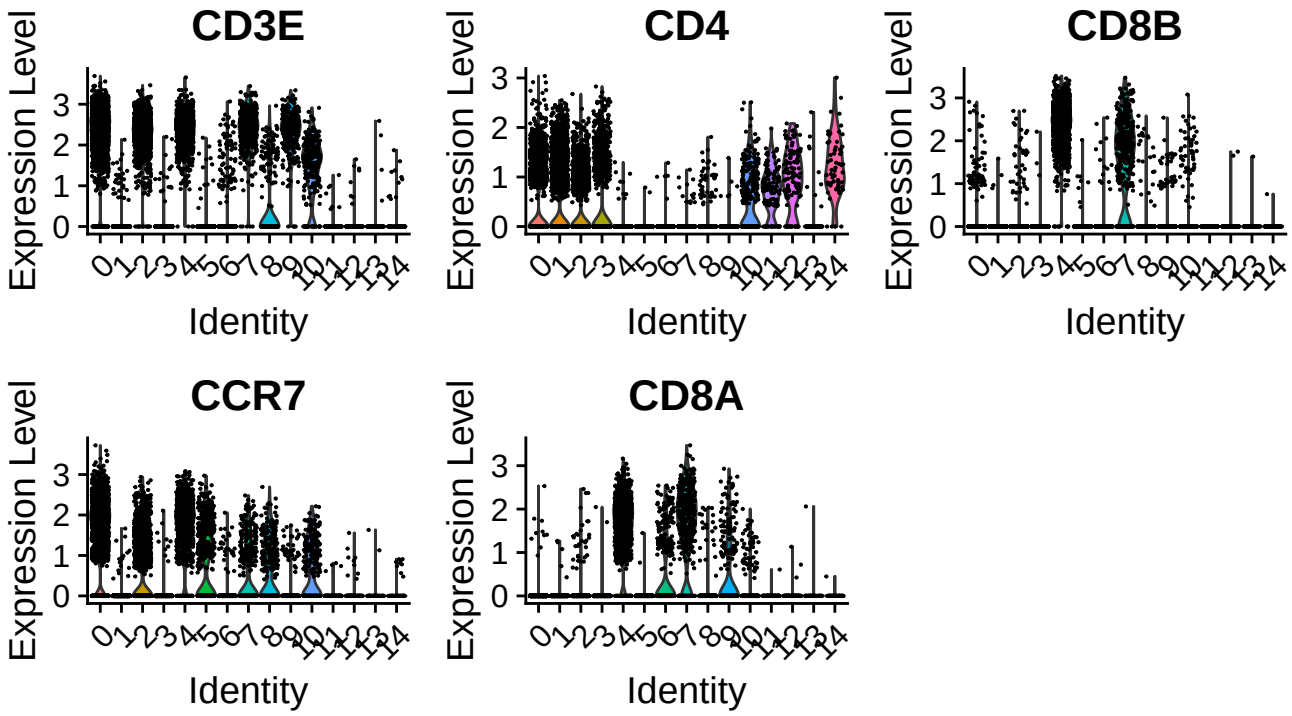

```
# Cluster 11: HLA-DRAhigh, CD14low, IL3RAneg, GZMBneg, CD1cpos, CLEC10Apos, ITGAXpos

# T cells
tcell_global <- VlnPlot(pbmc, features = c("CD3E", "CD4", "TRDV2",
                                           "CD8B","CCR7","TRGV9"))
tcell_global
```



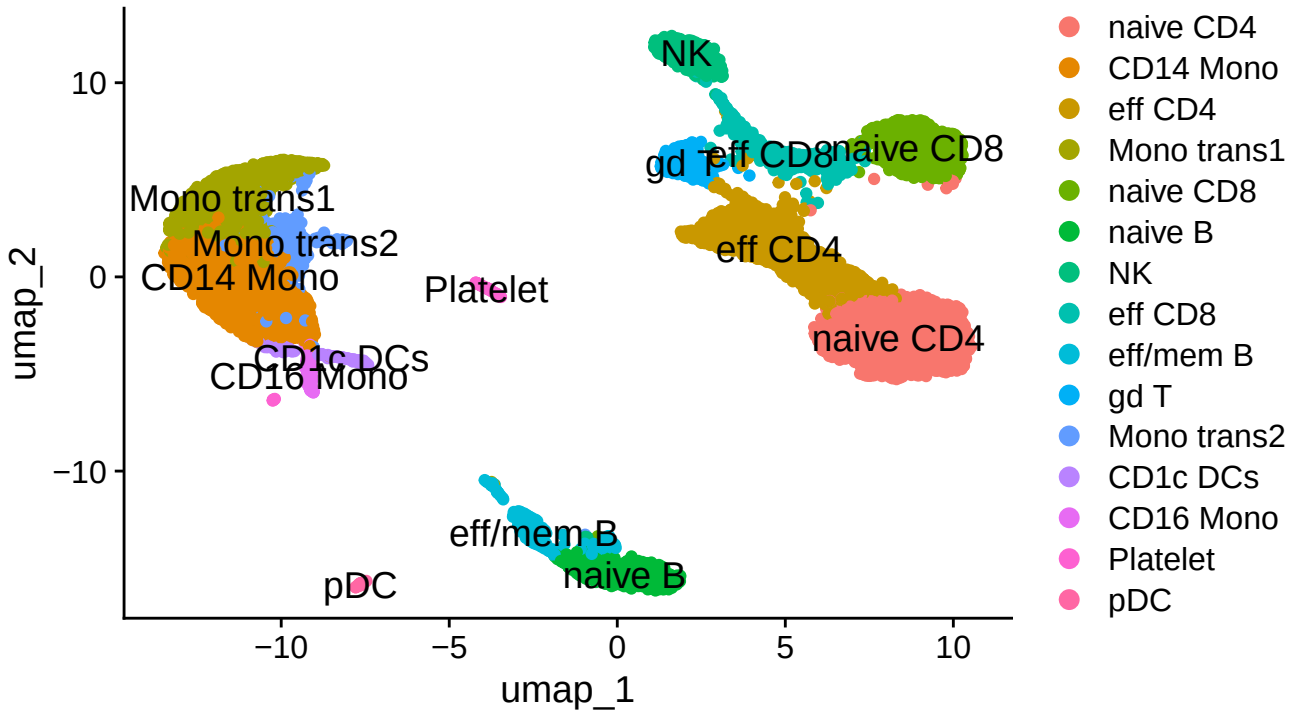

```
tcell_gd_9 <- VlnPlot(pbmc, features = c("CD3E", "TRGV9", "TRDV2",
                                         "TRAC", "TRBC1", "TRBC2")) # Cluster 9
tcell_gd_9
```



```
# gd T cell are also expressing TRAC and TRBC: This is wrong.

tcell_cd4_naive_0 <- VlnPlot(pbmc, features = c("CD3E", "CD4",
                                                "CD8B","CCR7", "CD8A")) # Cluster 0

tcell_cd8_naive_4 <- VlnPlot(pbmc, features = c("CD3E", "CD4",
                                                "CD8B","CCR7", "CD8A")) # Cluster 4

tcell_cd4_eff_2 <- VlnPlot(pbmc, features = c("CD3E", "CD4",
                                                "CD8B","CCR7", "CD8A")) # Cluster 2

tcell_cd8_eff_7 <- VlnPlot(pbmc, features = c("IFNG", "CD4", "IL2",
                                                "CD8B","CCR7", "PRF1")) # Cluster 7

tcell_cd4_naive_0
```




```
# CD8naive: IFNGlow, PRF1low, IL2neg, CCR7high
# PRF1: perforin-1

new.cluster.ids <- c("naive CD4","CD14 Mono","eff CD4","Mono trans1","naive CD8","naive B",
 ↪  "NK", "eff CD8","eff/mem B","gd T", "Mono trans2", "CD1c DCs", "CD16 Mono","Platelet","pDC")
names(new.cluster.ids) <- levels(pbmc)
pbmc <- RenameIdents(pbmc, new.cluster.ids)
umap2 <- DimPlot(pbmc, reduction = "umap", label = TRUE, pt.size = 1.5, label.size = 5) #+
 ↪  NoLegend()
umap2
```




```
# ggsave(filename = "Data_2024_Internship/umap_pbmc_annotations.png", height = 10, width = 16,
 ↪  plot = plot, dpi = 600)

# Identify highly upregulated-DEGs for each cluster, using ROC test
clusters.markers <- FindAllMarkers(pbmc,
                                   only.pos = TRUE, # show only upregulated-DEGs
                                   test.use = "roc",
                                   min.pct = 0.25, # Genes must be detected in at least 25% of
                                    ↪  cells
                                   return.thresh = 0.05 # Return markers with adjusted p-value <
                                    ↪  0.05
                                   )
clusters.markers %>%
  group_by(cluster) %>%
  dplyr::filter(avg_log2FC > 1) |>
  relocate(gene, .before = avg_diff)
```

```
# A tibble: 5,878 x 8
# Groups:   cluster [15]
   myAUC gene      avg_diff power avg_log2FC pct.1 pct.2 cluster
   <dbl> <chr>        <dbl> <dbl>      <dbl> <dbl> <dbl> <fct>
 1 0.861 TCF7         1.06  0.722       1.89 0.967 0.4   naive CD4
 2 0.844 LEF1         1.02  0.688       2.00 0.918 0.318 naive CD4
 3 0.834 SARAF        0.708 0.668       1.15 0.988 0.827 naive CD4
 4 0.825 CCR7         0.947 0.65        1.84 0.903 0.361 naive CD4
 5 0.815 LDHB         0.758 0.63        1.24 0.977 0.677 naive CD4
 6 0.8   CD3E         0.800 0.6         1.33 0.979 0.442 naive CD4
 7 0.768 NOSIP        0.713 0.536       1.32 0.887 0.541 naive CD4
```

```
 8 0.758 SELL         0.603 0.516       1.02 0.966 0.682 naive CD4
 9 0.757 LINC00861    0.750 0.514       1.59 0.802 0.36  naive CD4
10 0.755 CD3D         0.636 0.51        1.19 0.904 0.411 naive CD4
# i 5,868 more rows
```

```
# Filter markers based on the AUC threshold (e.g., AUC > 0.7)
filtered_markers <- clusters.markers %>%
  filter(avg_diff > 0.7) # This 'avg_diff' contains the AUC values for the ROC test

head(filtered_markers)
```

```
      myAUC  avg_diff power avg_log2FC pct.1 pct.2   cluster  gene
TCF7  0.861 1.0641872 0.722   1.890757 0.967 0.400 naive CD4  TCF7
LEF1  0.844 1.0213635 0.688   1.997637 0.918 0.318 naive CD4  LEF1
SARAF 0.834 0.7082457 0.668   1.145163 0.988 0.827 naive CD4 SARAF
CCR7  0.825 0.9469439 0.650   1.838185 0.903 0.361 naive CD4  CCR7
LDHB  0.815 0.7575300 0.630   1.242813 0.977 0.677 naive CD4  LDHB
CD3E  0.800 0.8001684 0.600   1.331712 0.979 0.442 naive CD4  CD3E
```

```
# pct.1 shows how frequently the gene is expressed in the cells of cluster 0.
# pct.2 shows how frequently the gene is expressed in the cells of all other clusters combined.

# write.csv("Data_2024_Internship/clusters_FindAllMarkers.csv", row.names = FALSE, quote =
 ↪  FALSE)
```

After finishing the V(D)J and gene-expression data analysis, save Seurat-object as RDS file. This file will contain only the gene-expression data analysis.

```
pbmc
```

```
An object of class Seurat
20226 features across 9999 samples within 1 assay
Active assay: RNA (20226 features, 2000 variable features)
 3 layers present: counts, data, scale.data
 2 dimensional reductions calculated: pca, umap
```

```
# Save the Seurat object 'pbmc' to a file
# saveRDS(pbmc, file = "Data_2024_Internship/pbmc_geneexp_clustering_3_10x.RDS")
```

The “Part III” of the report will show the integration of V(D)J with the analyzed-gene-expression data, followed by the identification of doublets and many other analyses.