scRNAseq Analysis

2024-06-16

Contents

---

scRNAseq Analysis Pipeline

1 Overview

This document outlines an automated pipeline for analyzing single-cell RNA sequencing (scRNA-seq) data using the 10x Genomics Chromium platform with 5' gene expression (GEX) captures. The analysis includes> 1. Loading data from `.h5` files

2. Performing quality control and normalization

3. Cell cycle scoring
4. Annotating cells using SingleR
5. Integrating multiple samples
6. Visualizing results
7. Identifying marker genes
8. Differential expression and marker selection
9. Cell type annotation using SingleR

2 0. Load the data

```
# Function to load data and create Seurat object
load_and_create_seurat_object <- function(file_path, project_name) {
  data <- Read10X_h5(file_path, use.names = TRUE, unique.features = TRUE)

  # Check and extract 'Gene Expression' modality if present
  if (is.list(data) && "Gene Expression" %in% names(data)) {
    rna_data <- data$`Gene Expression`
  } else if (is.list(data)) {
    rna_data <- data[[1]]
  } else {
    rna_data <- data
  }

  # Create Seurat object
  seurat_object <- CreateSeuratObject(counts = rna_data, project = project_name, min.cells = 3

  return(seurat_object)
}

# Load either PBMC or Tumor data
file_path <- "/home/mmerono/Downloads/sc5p_v2_hs_PBMC_10k_filtered_feature_bc_matrix.h5"
project_name <- "scRNAseq_analysis"
seurat_obj <- load_and_create_seurat_object(file_path, project_name)
```

```
## Genome matrix has multiple modalities, returning a list of matrices for this genome
```

3 1. Quality control (QC) and filtering

3.0.1 Compute QC Metrics

```
# Add number of genes per UMI for each cell to metadata
seurat_obj$log10GenesPerUMI <- log10(seurat_obj$nFeature_RNA) / log10(seurat_obj$nCount_RNA)
```

```r
# Compute percent mito ratio
seurat_obj$mitoRatio <- PercentageFeatureSet(object = seurat_obj, pattern = "^MT-") / 100
```

3.0.2 Ratio of mitochondiral

In single-cell RNA sequencing analysis, we aim to filter out specific cells to ensure data quality. First, we remove cells with a low number of genes or total molecules, as these potentially represent low-quality or empty cells, which can introduce noise into the dataset. Next, we exclude cells with an abnormally high number of genes or total molecules because these may be doublets or multiplets, where two or more cells were captured together, introducing bias to the data. Lastly, we filter out cells with a high percentage of mitochondrial genes, as this often indicates dying or low-quality cells, which can compromise the accuracy and reliability of the analysis.

```r
# Compute percent mito ratio
seurat_obj$mitoRatio <- PercentageFeatureSet(object = seurat_obj, pattern = "^MT-")
seurat_obj$mitoRatio <- seurat_obj@meta.data$mitoRatio / 100

# Create metadata dataframe
metadata <- seurat_obj@meta.data

# Add cell IDs to metadata
metadata$cells <- rownames(metadata)

# Rename columns
metadata <- metadata %>%
  dplyr::rename(seq_folder = orig.ident, nUMI = nCount_RNA, nGene = nFeature_RNA)

# Add metadata back to Seurat object
seurat_obj@meta.data <- metadata
```

3.0.3 Cell counts visualization

```r
# Visualize the correlation between genes detected and number of UMIs
p1 <- metadata %>%
  ggplot(aes(x = nUMI, y = nGene, color = mitoRatio)) +
  geom_point() +
  scale_colour_gradient(low = "gray90", high = "black") +
  stat_smooth(method = "lm", color = "blue") +
  scale_x_log10() +
  scale_y_log10() +
  theme_classic() +
  geom_vline(xintercept = 500, linetype = "dashed") +
  geom_hline(yintercept = 250, linetype = "dashed") +
  labs(
    title = "Correlation between Genes Detected and Number of UMIs",
    x = "Number of UMIs (log scale)",
```

```
    y = "Number of Genes (log scale)",
    color = "Mitochondrial Ratio"
  ) +
  theme(plot.title = element_text(hjust = 0.5))

# Save the plot
ggsave("plots/cell_counts_visualization.png", plot = p1)
```

```
## Saving 6 x 6 in image
## `geom_smooth()` using formula = 'y ~ x'
```

```
p1
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

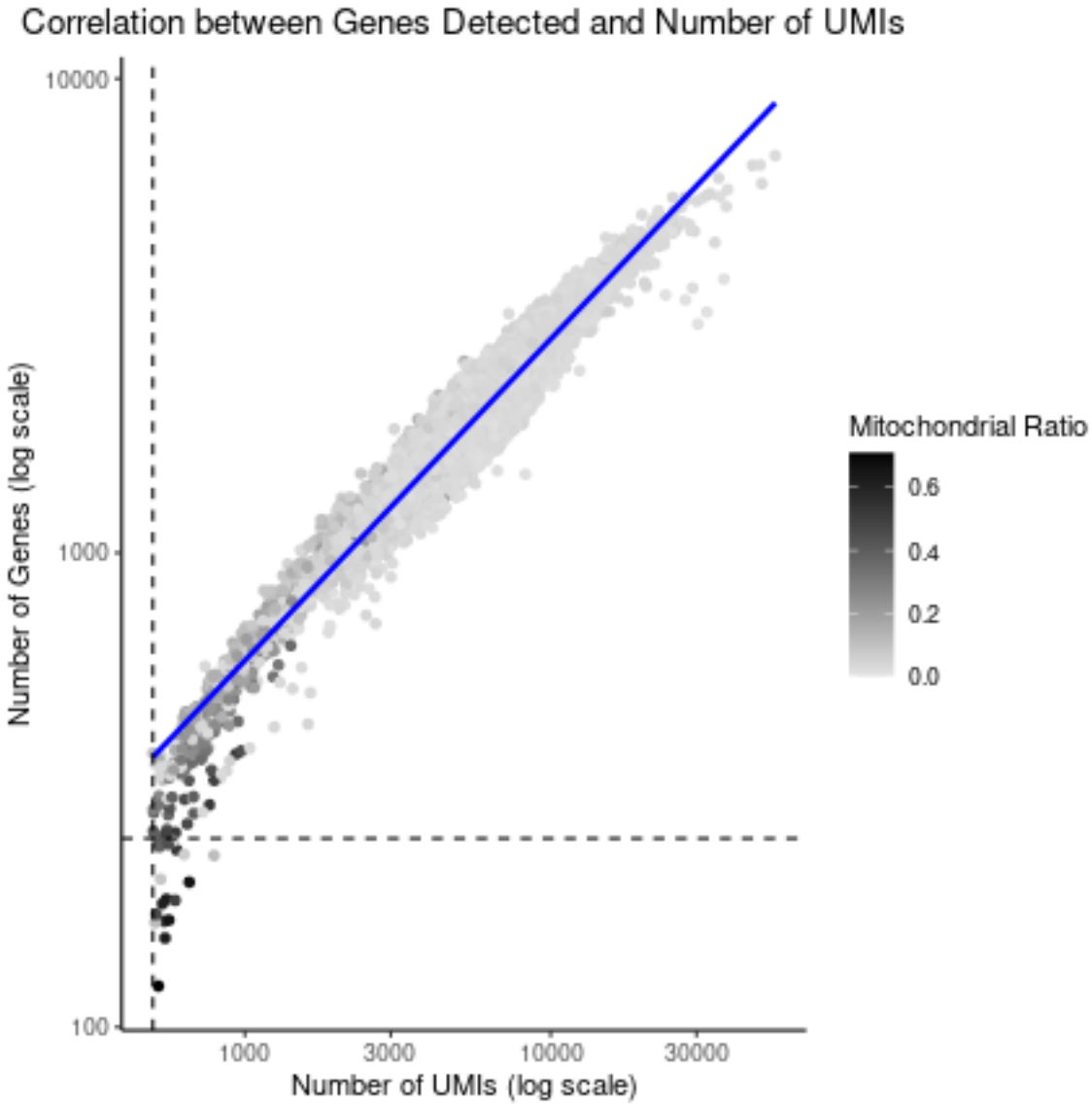


Figure 1: Figure 1: .

In the context of quality control (QC) metrics visualization in single-cell RNA sequencing analysis, we employ filtering criteria to refine the dataset. Firstly, we target cells with unique feature counts exceeding 2,500 or falling below 200, as these extremes may indicate technical artefacts or low-quality cells. Additionally, we identify cells with mitochondrial counts surpassing 5%, indicative of potential cell stress or poor quality, and subsequently remove them from the dataset to mitigate their adverse effects on data interpretation.

###Violin plot of QC metrics

```
# Visualize QC metrics as a violin plot
p2 <- VlnPlot(
  object = seurat_obj,
  features = c("nUMI", "nGene", "mitoRatio"),
  pt.size = 0.01,
  ncol = 3
) &
  theme(
    axis.text.x = element_text(angle = 0, hjust = 0.5),
    axis.title.x = element_blank()
  )
```

```
## Warning: Default search for "data" layer in "RNA" assay yielded no results;
## utilizing "counts" layer instead.
```

```
# Save the plot
 ggsave("plots/qc_violin_plot.png", plot = p2)
```

```
## Saving 6 x 6 in image
```

```
p2
```

Cell-level filtering

-nUMI $>$ 1000

-nGene $>$ 500

-log10GenesPerUMI $>$ 0.8

-mitoRatio $<$ 0.2

```
# Filter out low-quality cells using selected thresholds
filtered_pbmc <- subset(
  x = seurat_obj,
  subset = (nUMI >= 500) &
    (nGene >= 250) &
    (log10GenesPerUMI > 0.80) &
    (mitoRatio < 0.20)
)
```

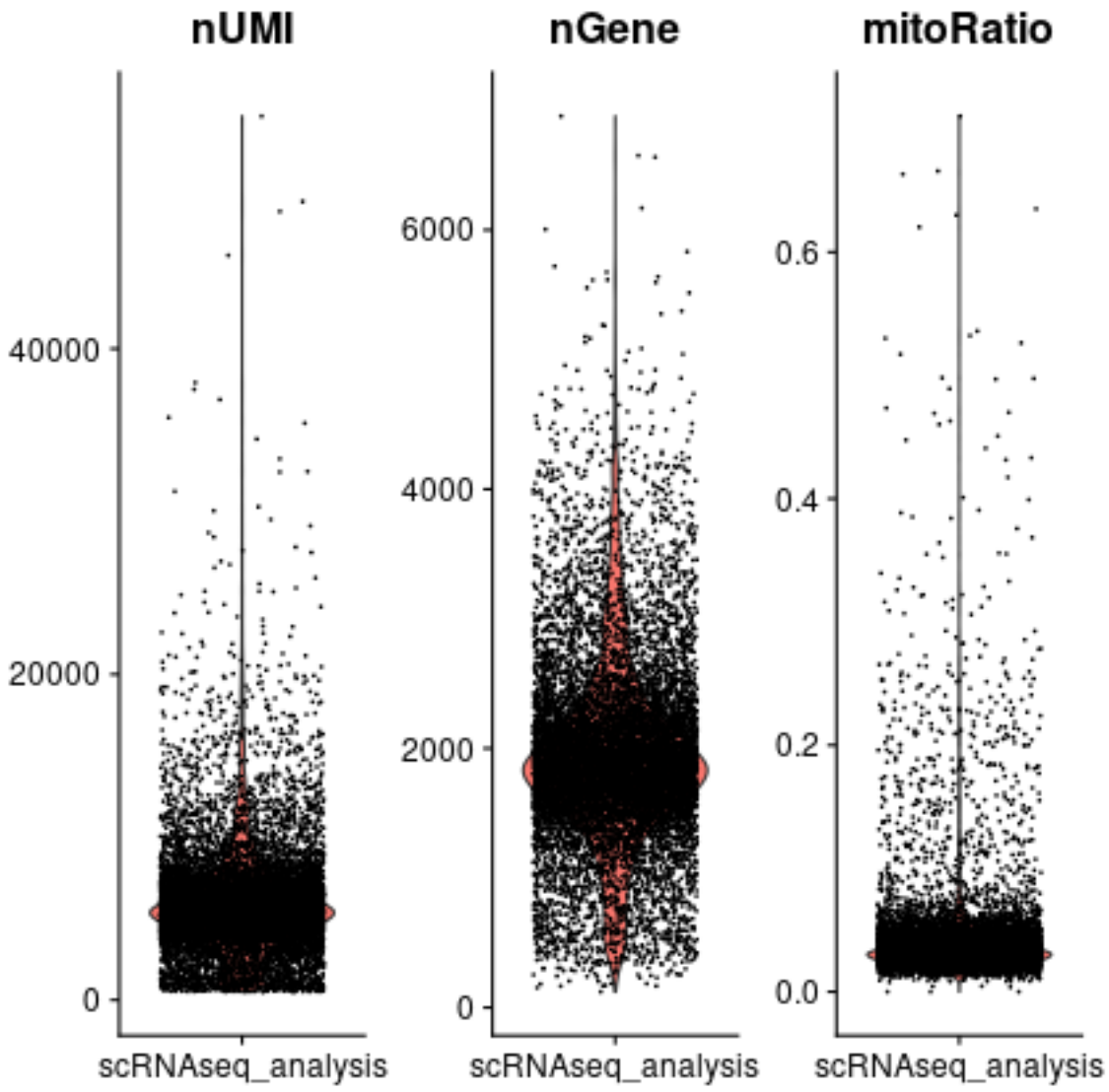


Figure 2: Figure 2: .

4 2. Normalizing the data

After removing unwanted cells from the dataset, the next step is to normalize the data.

```
# Standard log normalization for RNA
filtered_pbmc <- NormalizeData(filtered_pbmc, normalization.method = "LogNormalize", scale.fac
```

```
## Normalizing layer: counts
```

4.1 Cell cycle scoring

Calculate cell cycle scores. Seurat has build-in list, cc.genes.updated.2019, that defines genes involved in cell cycle.

```
s.genes <- cc.genes.updated.2019$s.genes
g2m.genes <- cc.genes.updated.2019$g2m.genes

filtered_pbmc <- CellCycleScoring(filtered_pbmc, s.features = s.genes, g2m.features = g2m.genes
table(filtered_pbmc[[]]$Phase)
```

```
##
##   G1  G2M    S
## 3207 3774 3357
```

5 3. Identification of highly variable features (feature selection)

```
filtered_pbmc <- FindVariableFeatures(filtered_pbmc, selection.method = "vst", nfeatures = 200
```

```
## Finding variable features for layer counts
```

6 4. Scaling the data

Remove unwanted sources of variation.

```
# Scale the counts
filtered_pbmc <- ScaleData(filtered_pbmc)
```

```
## Centering and scaling data matrix
```

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

# Plot variable features with and without labels
var_features <- VariableFeaturePlot(filtered_pbmc)
lab_points <- LabelPoints(plot = var_features, points = top10, repel = TRUE)
```

```
## When using repel, set xnudge and ynudge to 0 for optimal results
```

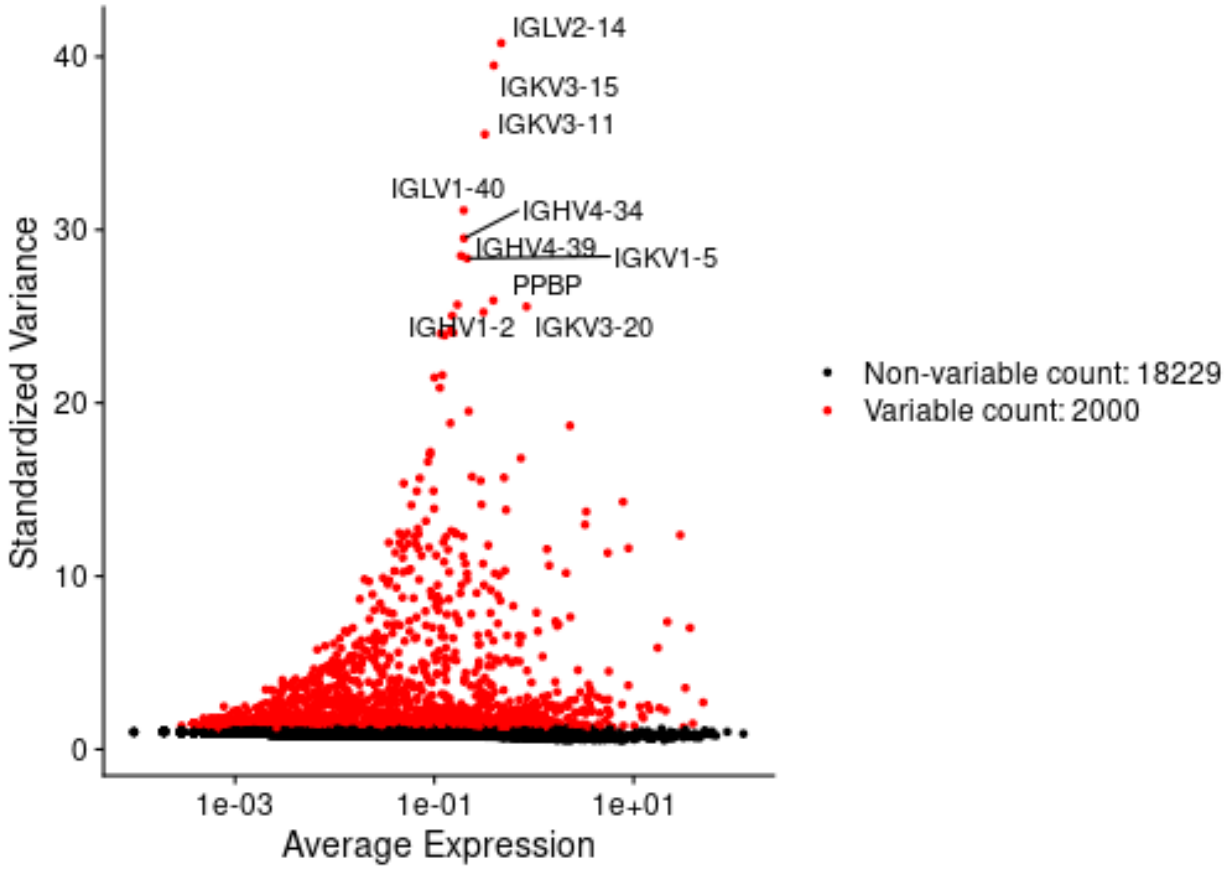
```
# Save the plots
ggsave("plots/variable_features.png", plot = var_features)
```

```
## Saving 7 x 5 in image
```

```
ggsave("plots/variable_features_labeled.png", plot = lab_points)
```

```
## Saving 7 x 5 in image
```

```
lab_points
```



7 5. Performing linear dimensional reduction

Next, we perform PCA on the scaled data.

By default we see 2000 most variable genes.

We also split the figure by cell cycle phase, to evaluate similarities and/or differences.

```
filtered_pbmc <- RunPCA(filtered_pbmc, features = VariableFeatures(object = filtered_pbmc))
```

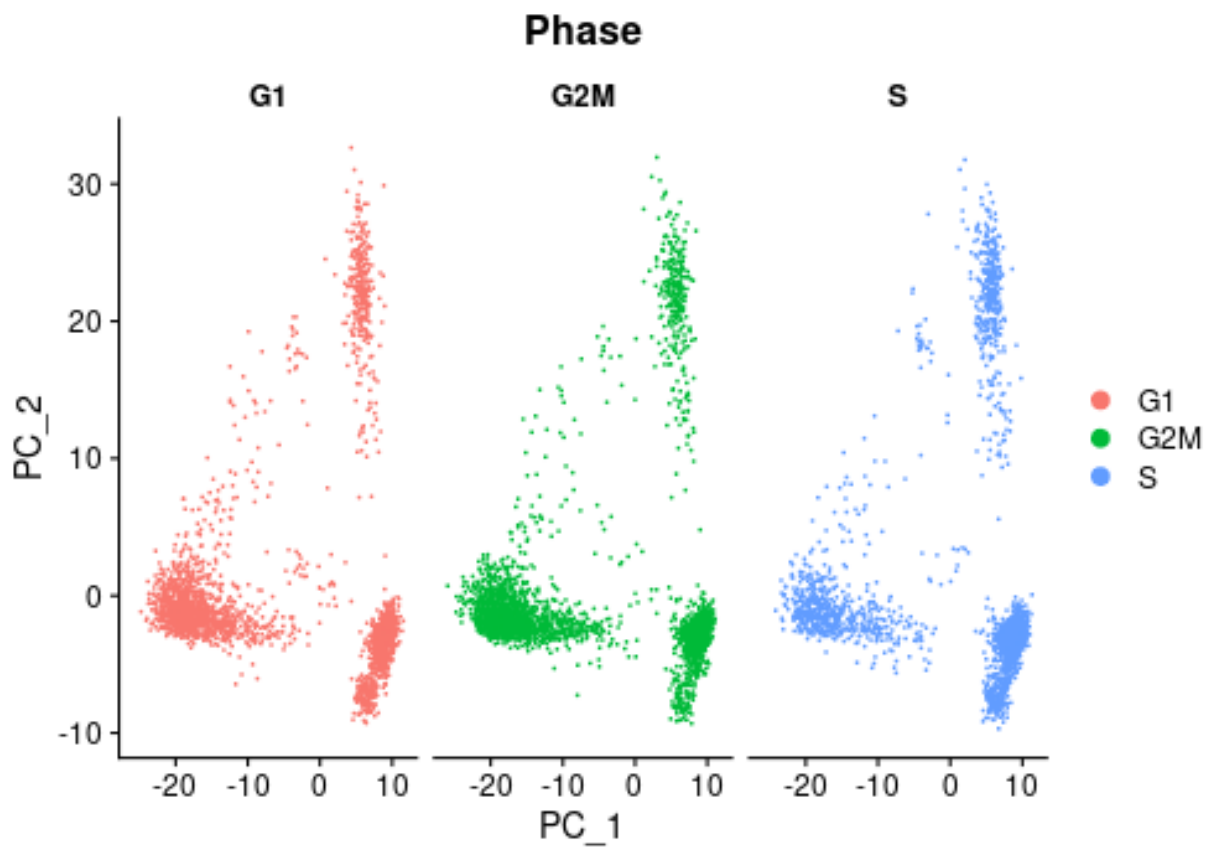
```
## PC_ 1
## Positive:  IFITM1, LTB, CD3E, IL32, TCF7, CD3D, IL7R, FCMR, CD7, CCR7
##     TRBC2, LEF1, CD27, LIME1, CD247, ARL4C, CD2, GZMM, MAL, AQP3
##     TRBC1, PIM2, CDC25B, ITM2A, INPP4B, CTSW, CNN2, TRAC, NELL2, HIST1H4C
```

```
## Negative:  LYZ, CST3, FCN1, IFI30, S100A9, MNDA, S100A8, VCAN, TYROBP, SPI1
##     CTSS, LST1, SERPINA1, TYMP, FOS, AIF1, FTL, LGALS1, CSTA, PSAP
##     CEBPD, FCER1G, CD14, CYBB, S100A12, TNFAIP2, GRN, CSF3R, CD36, MS4A6A
## PC_ 2
## Positive:  MS4A1, CD79A, BANK1, IGHM, LINC00926, NIBAN3, HLA-DQA1, FCER2, TCL1A, CD79B
##     HLA-DQB1, CD19, HLA-DQA2, SPIB, RALGPS2, FCRLA, FCRL1, HLA-DRA, AFF3, VPREB3
##     CD22, HLA-DPB1, HVCN1, P2RX5, CD74, HLA-DOB, BLK, HLA-DPA1, BLNK, IGHD
## Negative:  IL32, IFITM1, CD7, CD3E, CD247, CD3D, IL7R, ANXA1, GZMM, TCF7
##     S100A4, CTSW, CD2, LEF1, ARL4C, IFITM2, S100A6, SRGN, PRF1, S100A10
##     ITGB2, LIME1, KLRK1, NKG7, TRBC1, GZMA, CST7, SAMHD1, MT2A, GAPDH
## PC_ 3
## Positive:  NKG7, CST7, GNLY, GZMB, GZMA, KLRD1, PRF1, FGFBP2, KLRF1, SPON2
##     CLIC3, TBX21, HOPX, SH2D1B, FCGR3A, S1PR5, ADGRG1, PTGDR, IL2RB, CCL5
##     CCL4, PRSS23, MATK, XCL2, GZMH, TTC38, AKR1C3, CX3CR1, FCRL6, CTSW
## Negative:  LEF1, CCR7, TCF7, MAL, IL7R, CD3E, CD3D, CD27, LTB, RGCC
##     NELL2, LRRN3, VIM, INPP4B, C20orf204, MYC, CD40LG, AIF1, TSHZ2, BEX3
##     H1FX, AQP3, PIM2, PASK, CD28, COTL1, AL138963.4, S100A12, S100A8, LINC02446
## PC_ 4
## Positive:  TMSB10, CYBA, CD37, LSP1, ITGB2, NKG7, EFHD2, PRF1, GZMA, CST7
##     GNLY, KLRD1, VIM, IFITM2, FGFBP2, CD74, KLRF1, HOPX, SPON2, S100A4
##     S100A6, TBX21, CX3CR1, CD81, S1PR5, S100A10, FCGR3A, PPIB, SH2D1B, IL2RB
## Negative:  CAVIN2, PRKAR2B, TUBB1, GNG11, PPBP, SPARC, MPIG6B, GP1BB, PF4, CLU
##     GP9, ITGA2B, TREML1, AC147651.1, CMTM5, TRIM58, MMD, PTGS1, GMPR, ITGB3
##     CA2, MYL9, ALOX12, MTURN, AL731557.1, SEPTIN5, ACRBP, C2orf88, SH3BGRL2, MAP3K7CL
## PC_ 5
## Positive:  LILRA4, CLEC4C, SERPINF1, LRRC26, IL3RA, PLD4, PPM1J, ITM2C, SCT, DNASE1L3
##     TPM2, TNFRSF21, AC097375.1, GAS6, SLC35F3, PACSIN1, DERL3, PTPRS, TLR9, MAP1A
##     SMPD3, MZB1, LGMN, AC007381.1, LINC00996, FCER1A, ZFAT, SCAMP5, LAMP5, CCDC50
## Negative:  LINC00926, FCER2, MS4A1, CD79A, BANK1, CD79B, FCRL1, CD19, FCRL3, CD22
##     IGHD, VPREB3, COL19A1, PAX5, CD40, CD200, HLA-DOB, PLPP5, KCNG1, LINC02397
##     GNLY, FGFBP2, KLRF1, PDLIM1, P2RX5, IGHV5-78, KLRD1, BLK, SWAP70, RALGPS2
```

```
# Plot the PCA colored by cell cycle phase
p3 <- DimPlot(filtered_pbmc, reduction = "pca", group.by = "Phase", split.by = "Phase")

# Save the plot
ggsave("plots/pca_phase.png", plot = p3)
```

```
## Saving 7 x 5 in image
```

```
p3
```



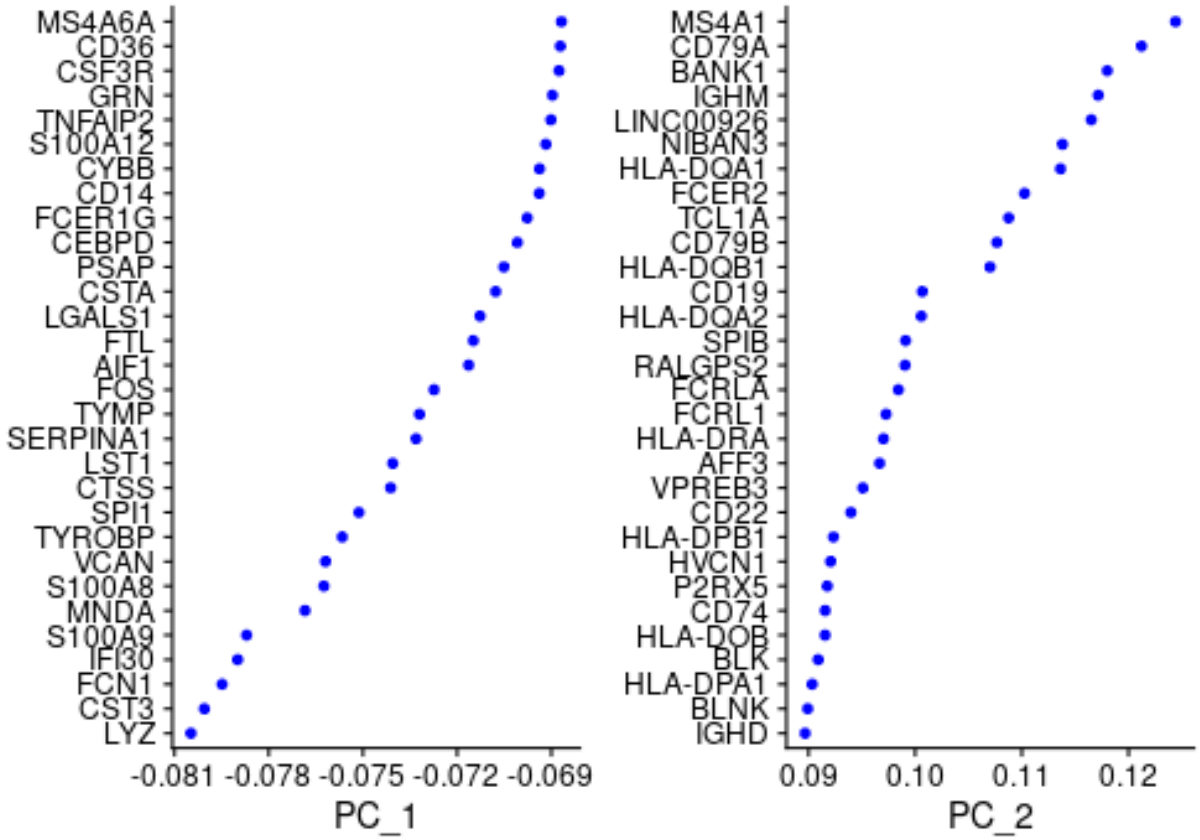

```
print(filtered_pbmc[["pca"]], dims = 1:5, nfeatures = 5)
```

```
## PC_ 1
## Positive:  IFITM1, LTB, CD3E, IL32, TCF7
## Negative:  LYZ, CST3, FCN1, IFI30, S100A9
## PC_ 2
## Positive:  MS4A1, CD79A, BANK1, IGHM, LINC00926
## Negative:  IL32, IFITM1, CD7, CD3E, CD247
## PC_ 3
## Positive:  NKG7, CST7, GNLY, GZMB, GZMA
## Negative:  LEF1, CCR7, TCF7, MAL, IL7R
## PC_ 4
## Positive:  TMSB10, CYBA, CD37, LSP1, ITGB2
## Negative:  CAVIN2, PRKAR2B, TUBB1, GNG11, PPBP
## PC_ 5
## Positive:  LILRA4, CLEC4C, SERPINF1, LRRC26, IL3RA
## Negative:  LINC00926, FCER2, MS4A1, CD79A, BANK1
```

```
p4 <- VizDimLoadings(filtered_pbmc, dims = 1:2, reduction = "pca")

# Save the plot
ggsave("plots/pca_loadings.png", plot = p4)
```

```
## Saving 7 x 5 in image
```

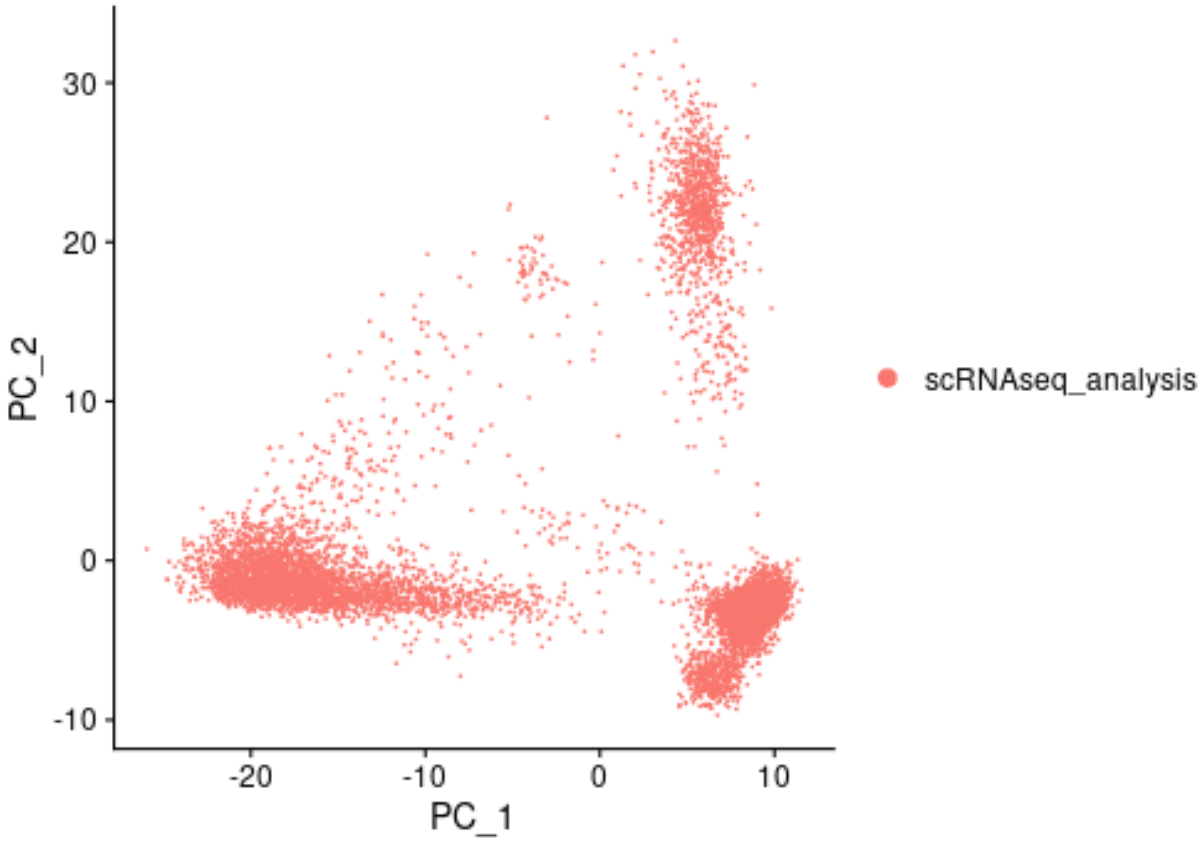
p4



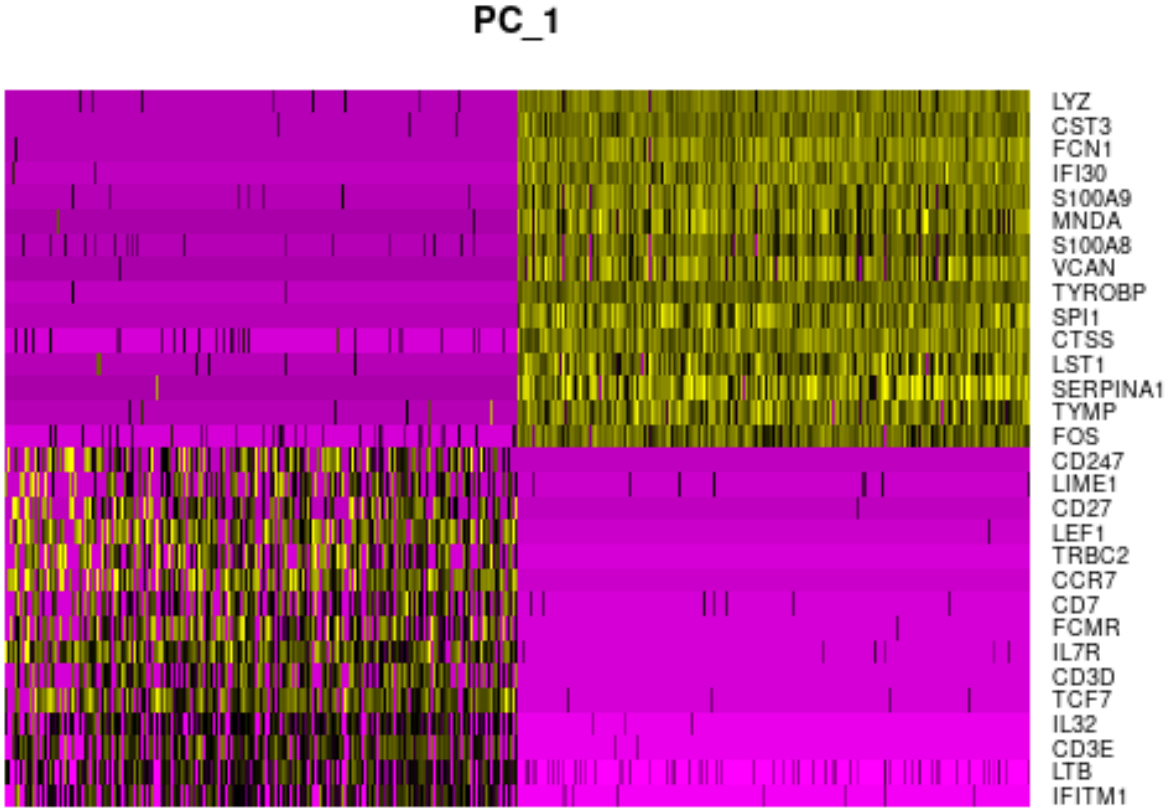

```r
p5 <- DimPlot(filtered_pbmc, reduction = "pca")

# Save the plot
ggsave("plots/pca_plot.png", plot = p5)
```

```
## Saving 7 x 5 in image
```

```r
p5
```




```
p6 <- DimHeatmap(filtered_pbmc, dims = 1, cells = 500, balanced = TRUE)
```



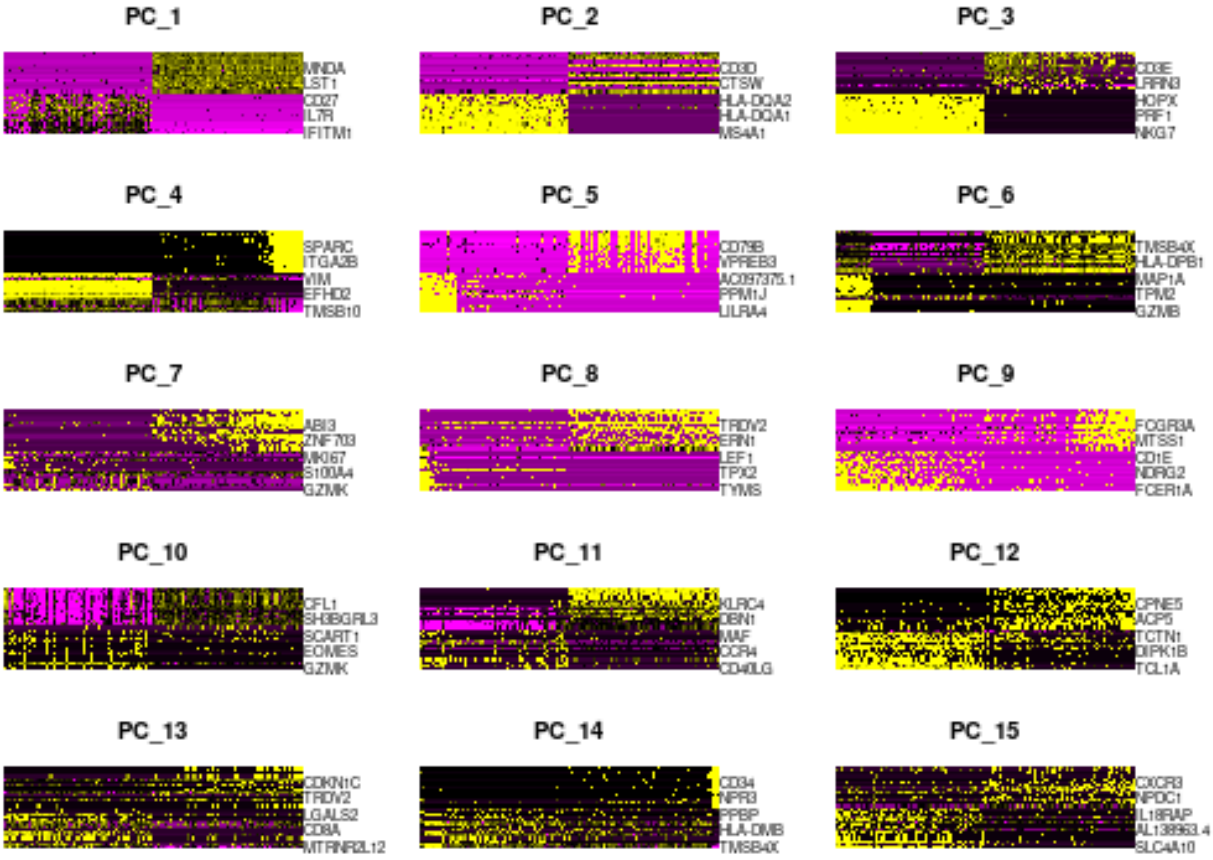

```
# Save the plot
ggsave("plots/pca_heatmap1.png", plot = p6)
```

```
## Saving 7 x 5 in image
```

```
p6
```

```
## NULL
```

```
p7 <- DimHeatmap(filtered_pbmc, dims = 1:15, cells = 500, balanced = TRUE)
```




```
# Save the plot
ggsave("plots/pca_heatmap2.png", plot = p7)
```

```
## Saving 7 x 5 in image
```

```
p7
```

```
## NULL
```

7.0.1 Determine the dimensionality of the dataset

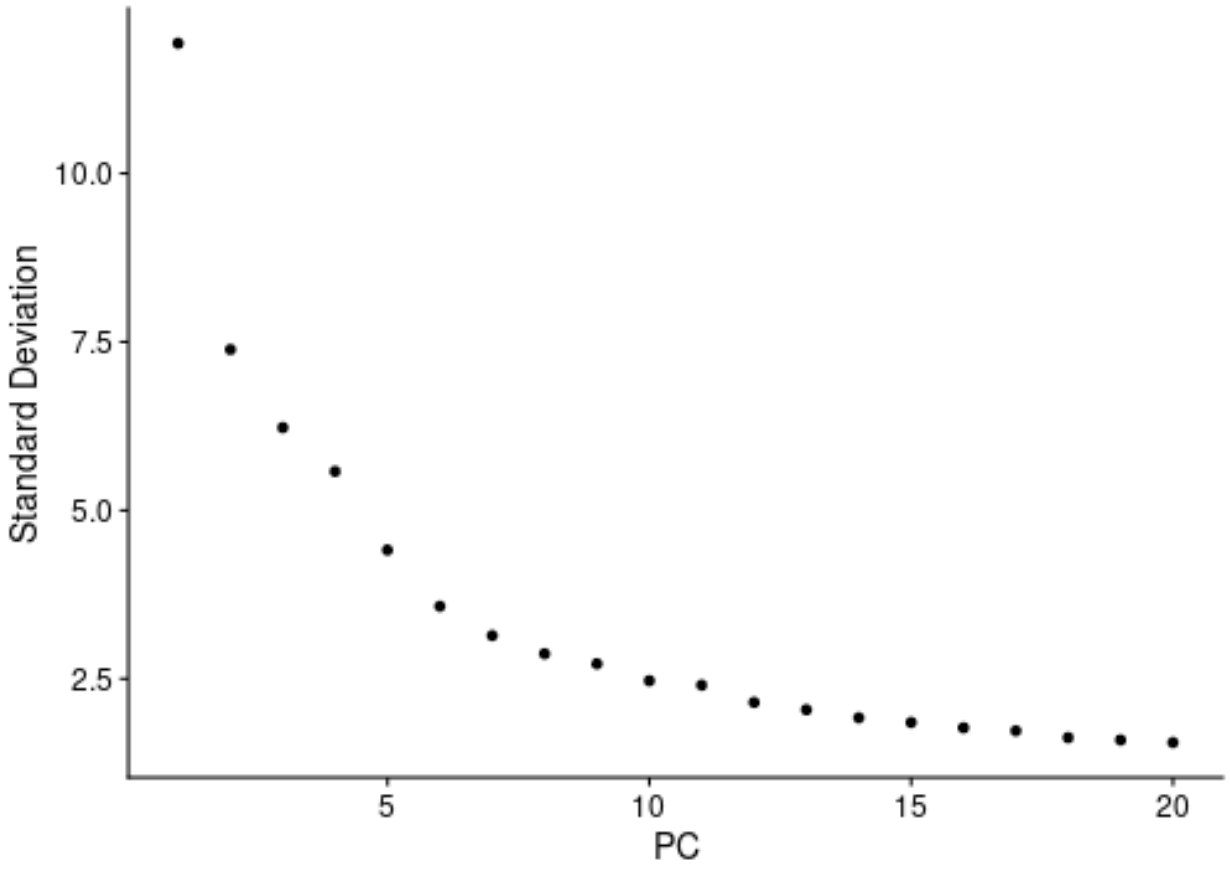
To determine the number of significant principal components (PCs):

```
p8 <- ElbowPlot(filtered_pbmc)
```

```
# Save the plot
ggsave("plots/elbow_plot.png", plot = p8)
```

```
## Saving 7 x 5 in image
```

```
p8
```



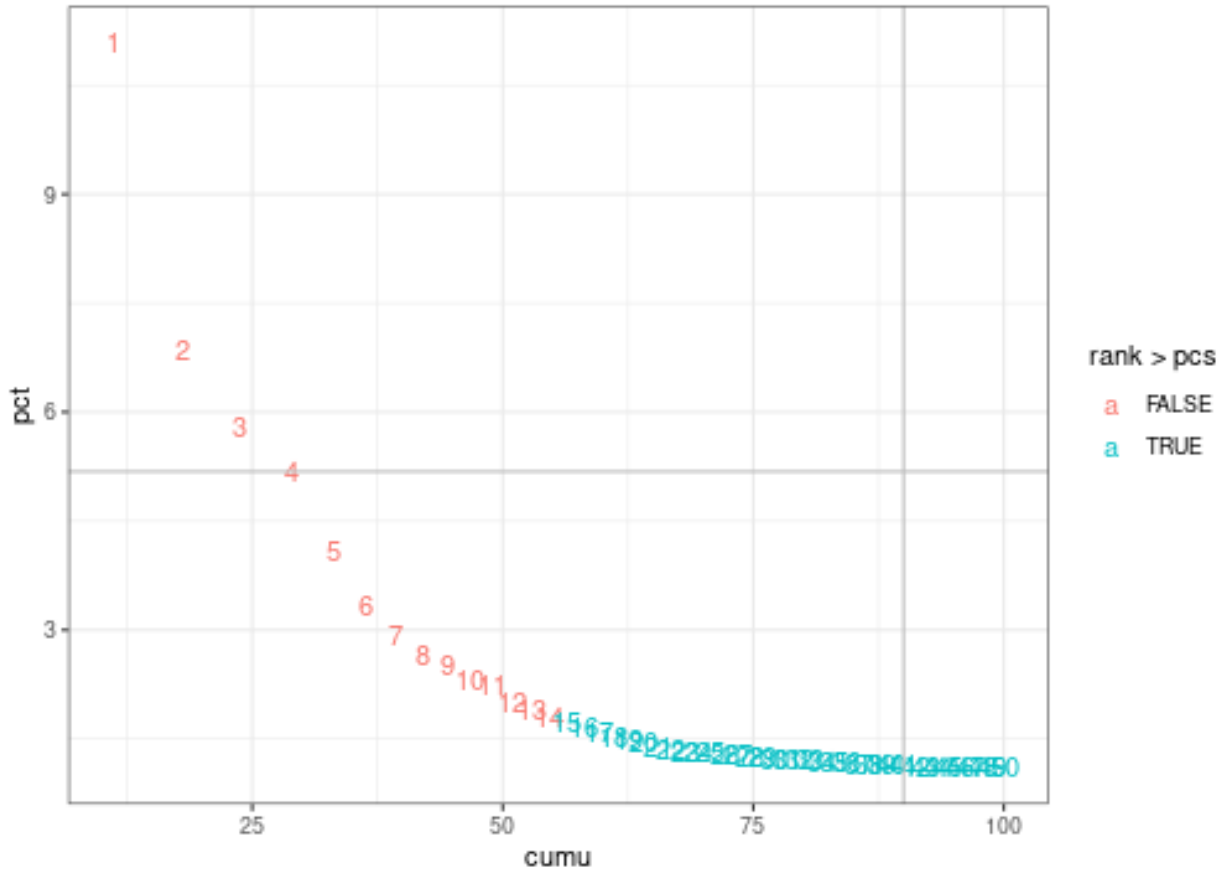
Where the "elbow" appears is usually the threshold for identifying the majority of the variation. This method can be a bit subjective about where to locate the "elbow". To help identifying we can use this plot:

```
# Determine percent of variation associated with each PC
pct <- filtered_pbmc[["pca"]]@stdev / sum(filtered_pbmc[["pca"]]@stdev) * 100

# Calculate cumulative percents for each PC
cumu <- cumsum(pct)

# Determine which PC exhibits cumulative percent greater than 90% and % variation associated w
co1 <- which(cumu > 90 & pct < 5)[1]

# Determine the difference between variation of PC and subsequent PC
co2 <- sort(which((pct[1:length(pct) - 1] - pct[2:length(pct)]) > 0.1), decreasing = TRUE)[1] +

# Minimum of the two calculations
pcs <- min(co1, co2)

# Create a dataframe with values
plot_df <- data.frame(pct = pct, cumu = cumu, rank = 1:length(pct))
opt_dims <- plot_df$rank[plot_df$rank > pcs][1]
```

```
# Elbow plot to visualize
p9 <- ggplot(plot_df, aes(cumu, pct, label = rank, color = rank > pcs)) +
  geom_text() +
  geom_vline(xintercept = 90, color = "grey") +
  geom_hline(yintercept = min(pct[pct > 5]), color = "grey") +
  theme_bw()

# Save the plot
ggsave("plots/elbow_plot_annotated.png", plot = p9)
```

```
## Saving 7 x 5 in image
```

```
p9
```



Developed by members of the teaching team at the Harvard Chan Bioinformatics Core (HBC).

Seurat recommends a default resolution of 0.8 for typical single-cell datasets. A higher resolution may be more suitable for larger datasets and vice versa.

8 6. Cluster the cells

```
filtered_pbmc <- FindNeighbors(filtered_pbmc, dims = 1:opt_dims)
```

```
## Computing nearest neighbor graph

## Computing SNN
```

```
filtered_pbmc <- FindClusters(filtered_pbmc, resolution = 0.8)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 10338
## Number of edges: 370196
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8917
## Number of communities: 19
## Elapsed time: 1 seconds
```

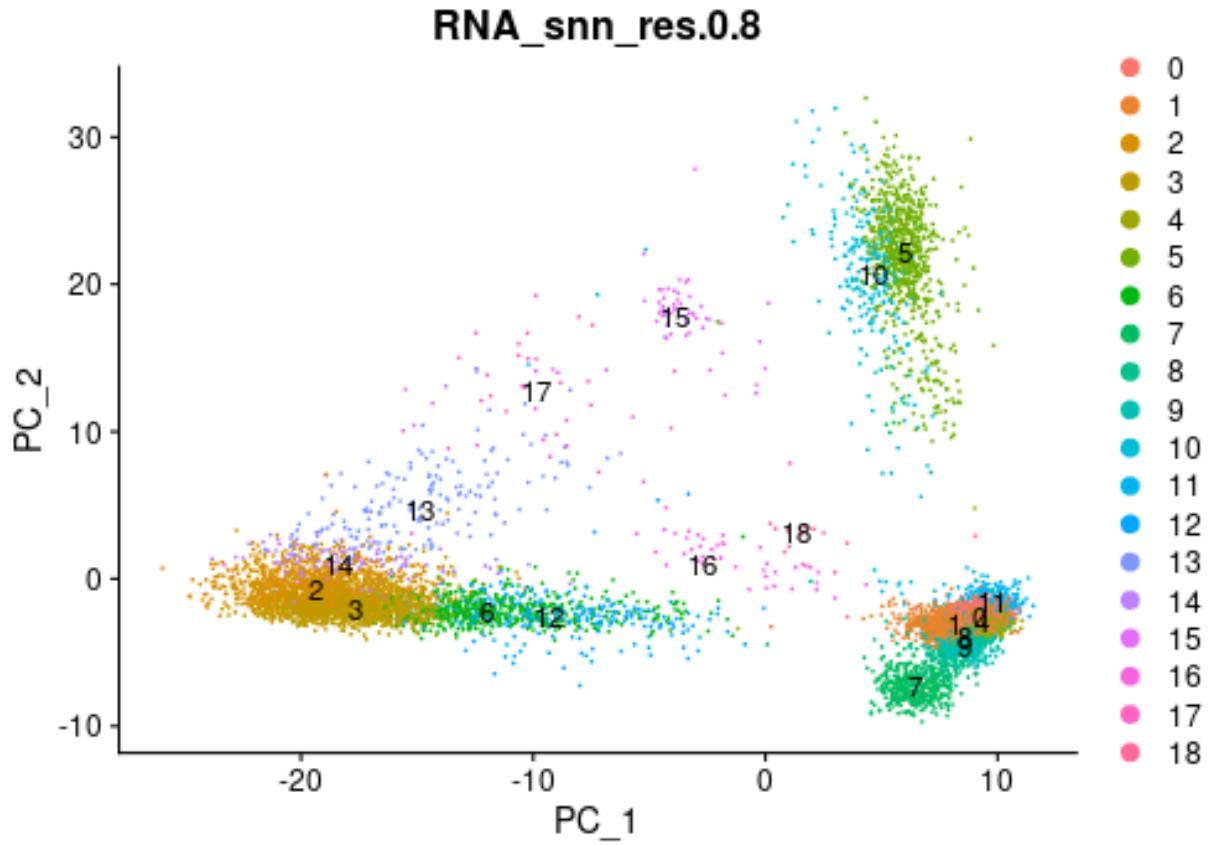
```
# View cluster IDs of the first 5 cells
head(Idents(filtered_pbmc), 5)
```

```
## AAACCTGAGACAGACC-1 AAACCTGAGCGATAGC-1 AAACCTGAGCGGCTTC-1 AAACCTGAGGATCGCA-1
##                  2                  0                  2                  2
## AAACCTGAGTCACGCC-1
##                  1
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
```

```
p10 <- DimPlot(filtered_pbmc, group.by = "RNA_snn_res.0.8", label = TRUE)

# Save the plot
ggsave("plots/cluster_plot.png", plot = p10)
```

```
## Saving 7 x 5 in image
```

```
p10
```



The lower the resolution the less clusters it will group the data in. High resolution will try to find more clusters. Cells are clustered based on their gene expression profile.

Since gene expression of the basis of differentiation, different clusters usually represents different cell types.

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

```
## AAACCTGAGACAGACC-1 AAACCTGAGCGATAGC-1 AAACCTGAGCGGCTTC-1 AAACCTGAGGATCGCA-1
##                  2                  0                  2                  2
## AAACCTGAGTCACGCC-1
##                  1
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
```

9 7. Non-linear dimantisonal reduction (tSNE/UMAP)

By default, the clusters are numbered by the number of cells with cluster 0 having the largest number of cells.

9.1 umap

```
filtered_pbmc <- RunUMAP(filtered_pbmc, dims = 1:opt_dims)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlatio
## This message will be shown once per session
```

```
## 13:12:06 UMAP embedding parameters a = 0.9922 b = 1.112
```

```
## 13:12:06 Read 10338 rows and found 15 numeric columns
```

```
## 13:12:06 Using Annoy for neighbor search, n_neighbors = 30
```

```
## 13:12:06 Building Annoy index with metric = cosine, n_trees = 50
```

```
## 0%   10   20   30   40   50   60   70   80   90   100%
```

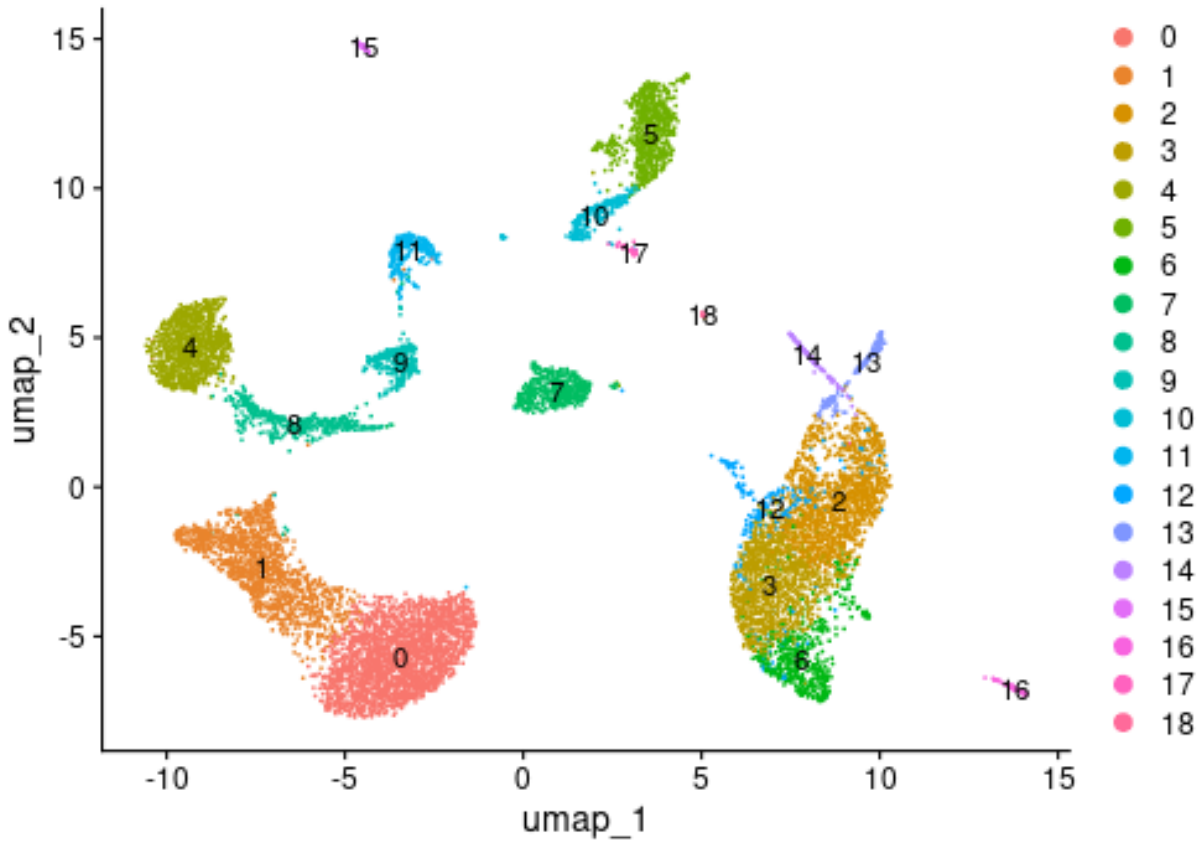
```
## [----|----|----|----|----|----|----|----|----|----|
```

```
## **************************************************|
## 13:12:07 Writing NN index file to temp file /tmp/Rtmp5VvQnY/file8176c5845da43
## 13:12:07 Searching Annoy index using 1 thread, search_k = 3000
## 13:12:09 Annoy recall = 100%
## 13:12:10 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors =
## 13:12:11 Initializing from normalized Laplacian + noise (using RSpectra)
## 13:12:11 Commencing optimization for 200 epochs, with 436340 positive edges
## 13:12:17 Optimization finished
```

```
p11 <- DimPlot(filtered_pbmc, reduction = "umap", label = TRUE)

# Save the plot
ggsave("plots/umap_plot.png", plot = p11)
```

```
## Saving 7 x 5 in image
```

```
p11
```

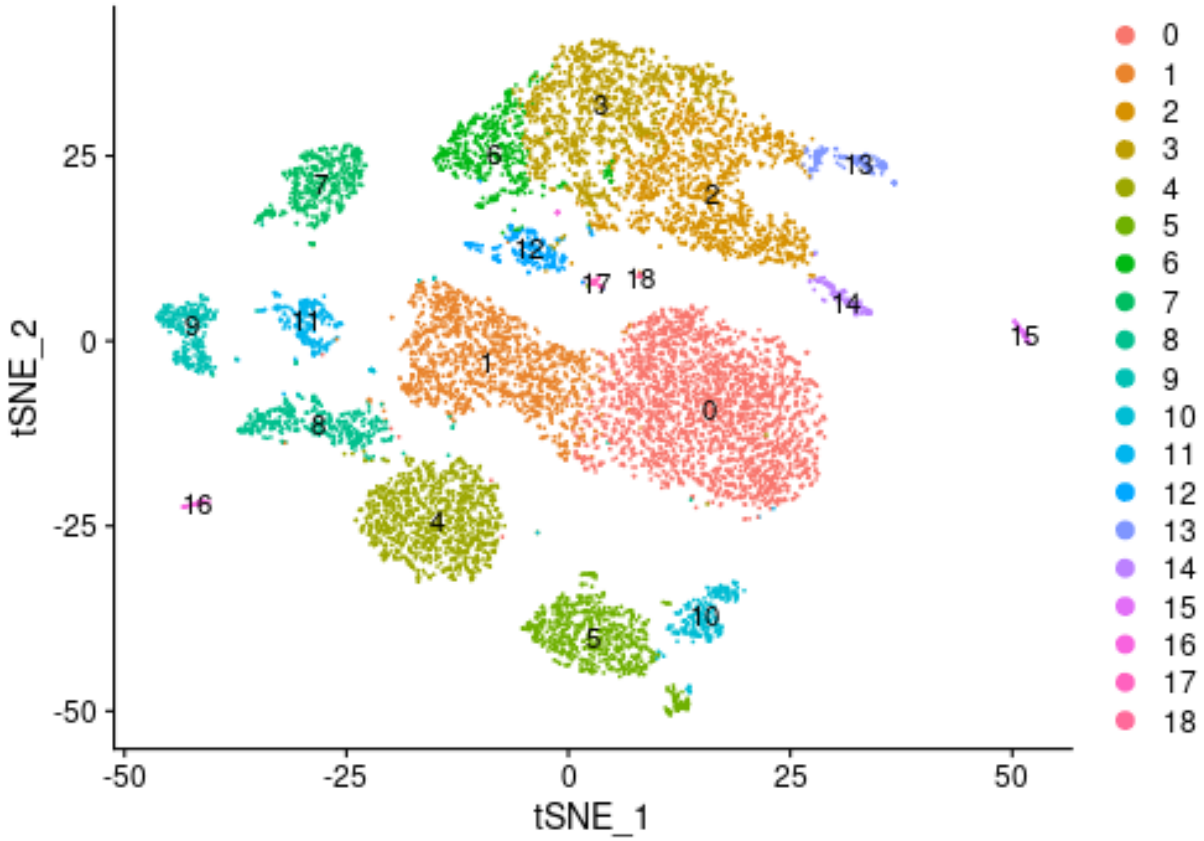


9.2 tsne

```
filtered_pbmc <- RunTSNE(filtered_pbmc, dims = 1:20)
p12 <- TSNEPlot(object = filtered_pbmc, label = TRUE)

# Save the plot
ggsave("plots/tsne_plot.png", plot = p12)
```

```
## Saving 7 x 5 in image
```

```
p12
```

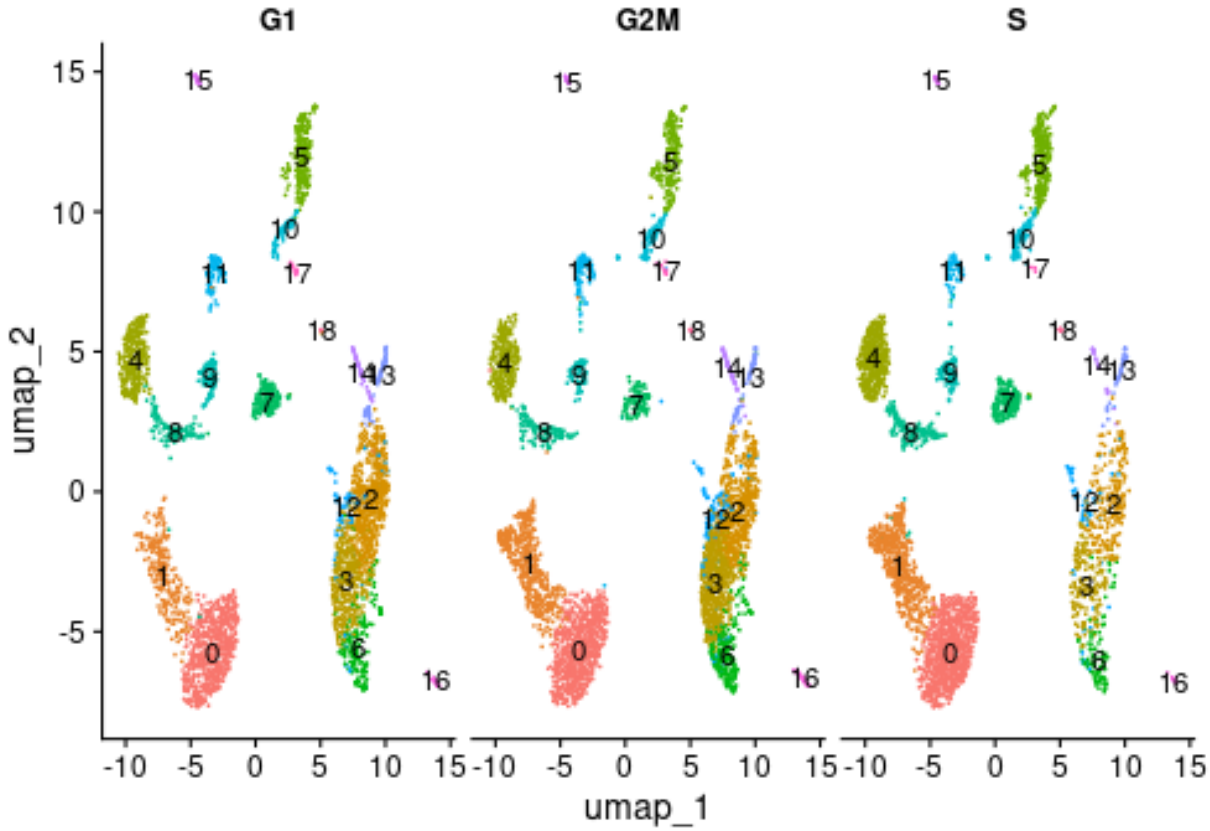


Done following SANGER singel cell workflow.

```
# Explore whether clusters segregate by cell cycle phase
p13 <- DimPlot(filtered_pbmc, label = TRUE, split.by = "Phase") + NoLegend()

# Save the plot
ggsave("plots/phase_seg_plot.png", plot = p13)
```

```
## Saving 7 x 5 in image
```

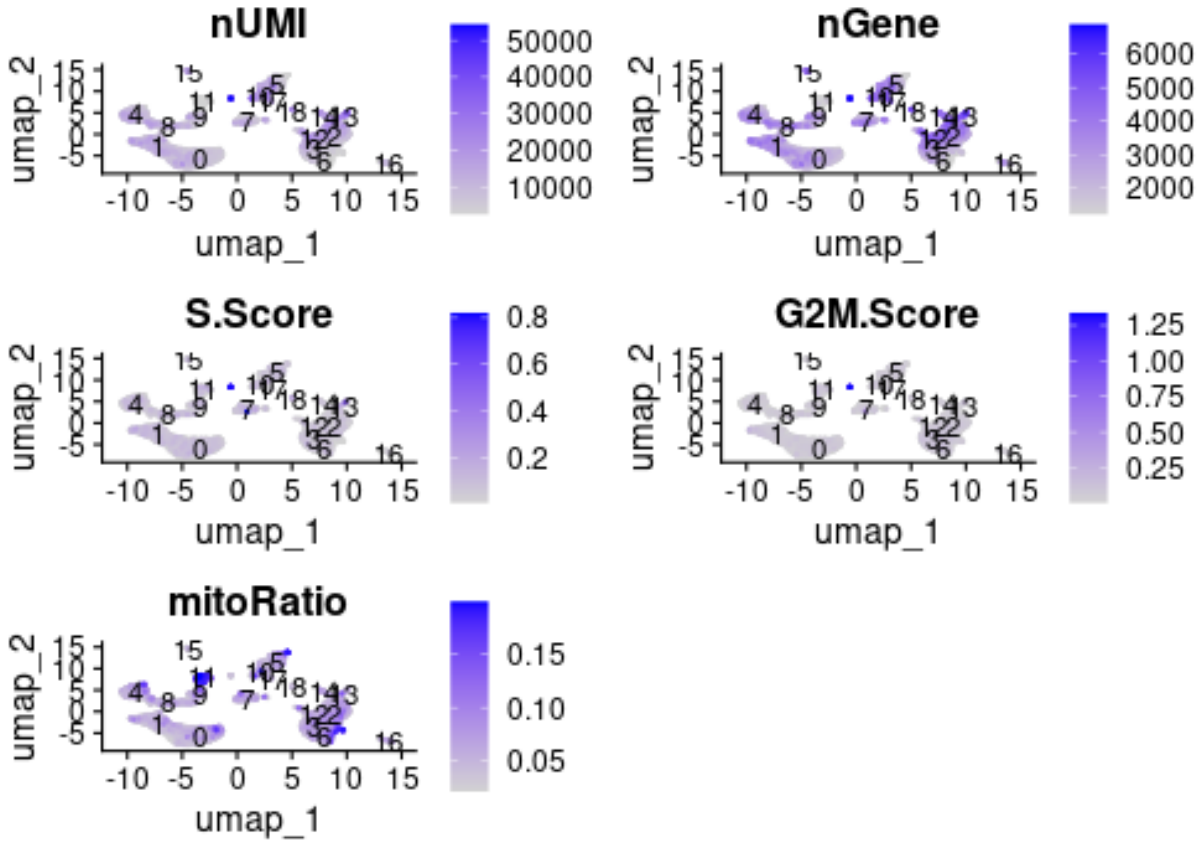
```
p13
```



9.3 Cluster segregation by QC metrics

The parameter min.cutoff of q10 is translated as 10% of the cells with the lowest expression of the gene will not exhibit any purple shading. This parameter is applied because when plotting, the order of the arguments will plot the positive cells above the negative cells and with min.cutoff argument we can determine the threshold for shading.

```
metrics <-  c("nUMI", "nGene", "S.Score", "G2M.Score", "mitoRatio")

p14 <- FeaturePlot(filtered_pbmc,
            reduction = "umap",
            features = metrics,
            pt.size = 0.4,
            order = TRUE,
            min.cutoff = 'q10',
            label = TRUE)

# Save the plot
ggsave("plots/feature_plot.png", plot = p14)

## Saving 7 x 5 in image

p14
```



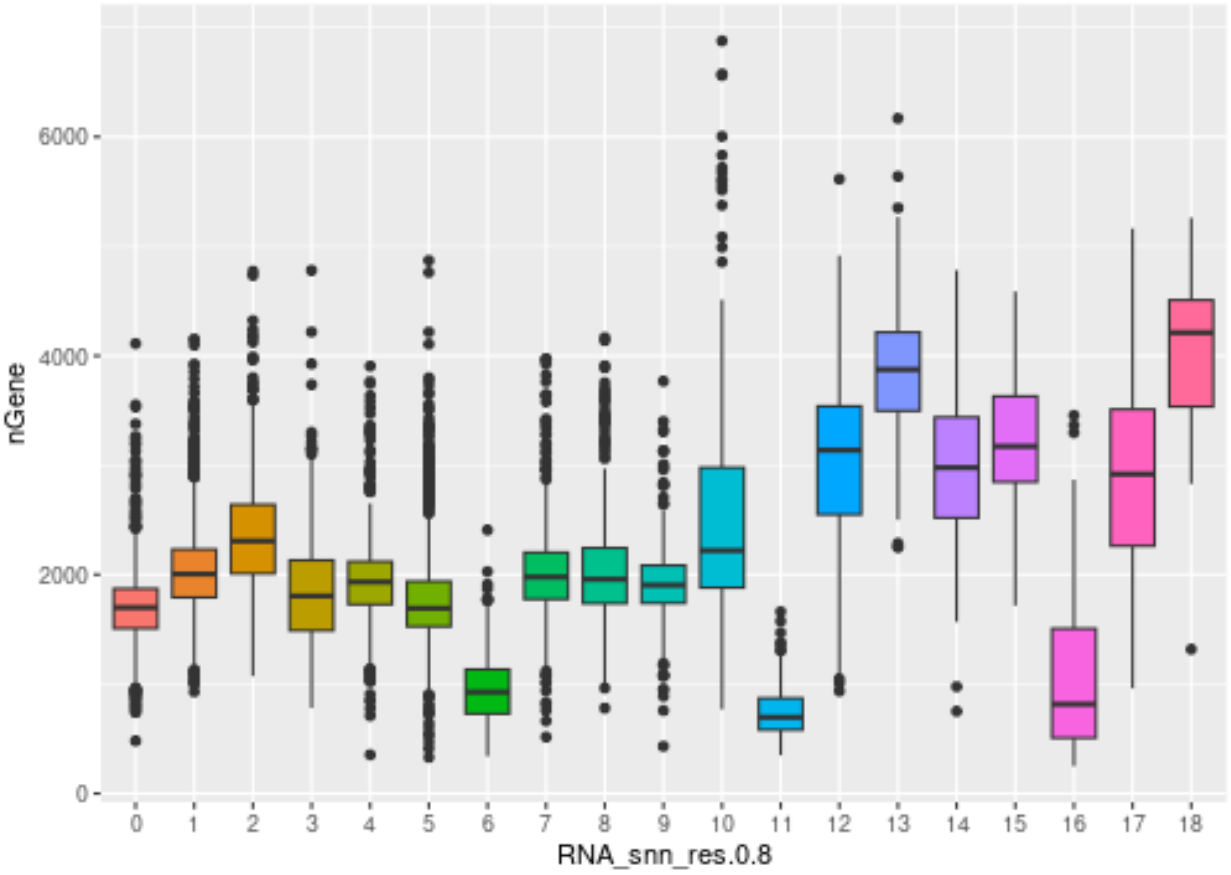
Done following hbctraining workflow.

##Boxplot of nGene per cluster

```
p15 <- ggplot(filtered_pbmc@meta.data) +
  geom_boxplot(aes(x = RNA_snn_res.0.8, y = nGene, fill = RNA_snn_res.0.8)) +
  NoLegend()

# Save the plot
ggsave("plots/nGene_boxplot.png", plot = p15)
```

```
## Saving 7 x 5 in image
```

```
p15
```



10 8. Differential expression and marker selection

Differential gene expression allows us to define gene markers specific to each cluster.

```
DefaultAssay(filtered_pbmc) <- "RNA"
filtered_pbmc <- NormalizeData(filtered_pbmc)
```

```
## Normalizing layer: counts
```

```
filtered_pbmc <- FindVariableFeatures(filtered_pbmc, selection.method = "vst", nfeatures = 200
```

```
## Finding variable features for layer counts
```

```
all.genes <- rownames(filtered_pbmc)
filtered_pbmc <- ScaleData(filtered_pbmc, features = all.genes)
```

```
## Centering and scaling data matrix
```

```
## Warning: Different features in new layer data than already exists for
## scale.data
```

Find markers for every cluster by comparing it to all remaining cells, while reporting only the positive ones. The test used: Wilcoxon Rank Sum.

```
all.markers <- FindAllMarkers(filtered_pbmc, only.pos = TRUE, min.pct = 0.5, logfc.threshold =
```

```
## Calculating cluster 0
## For a (much!) faster implementation of the Wilcoxon Rank Sum Test,
## (default method for FindMarkers) please install the presto package
## --------------------------------------------
## install.packages('devtools')
## devtools::install_github('immunogenomics/presto')
## --------------------------------------------
## After installation of presto, Seurat will automatically use the more
## efficient implementation (no further action necessary).
## This message will be shown once per session
## Calculating cluster 1
## Calculating cluster 2
## Calculating cluster 3
## Calculating cluster 4
## Calculating cluster 5
## Calculating cluster 6
## Calculating cluster 7
## Calculating cluster 8
## Calculating cluster 9
## Calculating cluster 10
## Calculating cluster 11
## Calculating cluster 12
## Calculating cluster 13
## Calculating cluster 14
## Calculating cluster 15
## Calculating cluster 16
## Calculating cluster 17
## Calculating cluster 18
table(all.markers$cluster)
##
##    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
##  190  142  598  433  158  175  169  443  130  175  253   87  179  823  828 1046
##   16   17   18
##  296  436 1378
top3_markers <- as.data.frame(all.markers %>% group_by(cluster) %>% top_n(n = 3, wt = avg_log2
top3_markers
```

```
##            p_val avg_log2FC pct.1 pct.2     p_val_adj cluster       gene
## 1   0.000000e+00  1.8494078 0.922 0.321  0.000000e+00       0       LEF1
## 2   0.000000e+00  2.0267240 0.661 0.209  0.000000e+00       0        MAL
## 3   0.000000e+00  1.8952301 0.565 0.185  0.000000e+00       0      TRAT1
## 4   0.000000e+00  2.3516192 0.817 0.250  0.000000e+00       1       AQP3
## 5  4.694613e-303  2.0148682 0.769 0.309 9.496733e-299       1      ITGB1
## 6  7.108404e-242  2.0920652 0.564 0.167 1.437959e-237       1     GPR183
## 7   0.000000e+00  3.2214159 0.909 0.127  0.000000e+00       2     LGALS2
## 8   0.000000e+00  3.1404242 0.553 0.064  0.000000e+00       2   TMEM176B
## 9   0.000000e+00  3.6756538 0.520 0.048  0.000000e+00       2       PID1
## 10  0.000000e+00  3.0208504 0.986 0.214  0.000000e+00       3    S100A12
## 11  0.000000e+00  2.9398611 1.000 0.293  0.000000e+00       3     S100A9
## 12  0.000000e+00  3.1815253 0.612 0.080  0.000000e+00       3     CYP1B1
## 13  0.000000e+00  4.7656333 0.990 0.066  0.000000e+00       4       CD8B
## 14  0.000000e+00  3.5941504 0.943 0.082  0.000000e+00       4       CD8A
## 15  0.000000e+00  5.1407883 0.768 0.034  0.000000e+00       4  LINC02446
## 16  0.000000e+00  7.2261767 0.933 0.012  0.000000e+00       5      TCL1A
## 17  0.000000e+00  6.4686598 0.891 0.021  0.000000e+00       5      FCER2
## 18  0.000000e+00  6.5260654 0.647 0.013  0.000000e+00       5       IGHD
## 19 6.238346e-306  3.0263411 0.992 0.347 1.261955e-301       6     S100A8
## 20 1.061347e-131  2.7763829 0.622 0.219 2.146999e-127       6       RGS2
## 21  1.718461e-95  2.7133474 0.503 0.181  3.476274e-91       6       RBP7
## 22  0.000000e+00  8.0215278 0.903 0.008  0.000000e+00       7      KLRF1
## 23  0.000000e+00  8.7861760 0.745 0.004  0.000000e+00       7     SH2D1B
## 24  0.000000e+00  7.9224906 0.596 0.004  0.000000e+00       7     PRSS23
## 25  0.000000e+00  3.5351653 0.875 0.105  0.000000e+00       8       CCL5
## 26 7.948990e-240  3.2435539 0.555 0.076 1.608001e-235       8      CXCR3
## 27 2.636578e-218  3.1614914 0.506 0.067 5.333533e-214       8      KLRC4
## 28  0.000000e+00  4.6348346 0.977 0.106  0.000000e+00       9      KLRB1
## 29  0.000000e+00  7.6860891 0.578 0.004  0.000000e+00       9    SLC4A10
## 30  0.000000e+00  4.8403765 0.512 0.021  0.000000e+00       9    TRAV1-2
## 31  0.000000e+00  5.5189084 0.600 0.030  0.000000e+00      10     JCHAIN
## 32  0.000000e+00  5.5907591 0.576 0.013  0.000000e+00      10       AIM2
## 33  0.000000e+00  7.8326409 0.544 0.003  0.000000e+00      10  TNFRSF13B
## 34  9.527043e-28  2.8636919 0.556 0.409  1.927226e-23      11       AAK1
## 35  4.046628e-26  2.9784009 0.523 0.363  8.185924e-22      11     BCL11B
## 36  1.550118e-23  2.9249776 0.523 0.391  3.135733e-19      11       NKTR
## 37  4.056651e-50  0.9458402 0.602 0.170  8.206198e-46      12      STAB1
## 38  2.414341e-41  0.9508349 0.572 0.178  4.883971e-37      12    CLEC12A
## 39  2.634484e-31  0.9018058 0.500 0.169  5.329298e-27      12     ZNF467
## 40  0.000000e+00  8.3902588 0.882 0.008  0.000000e+00      13     FCER1A
## 41  0.000000e+00  6.2398957 0.864 0.025  0.000000e+00      13    CLEC10A
## 42  0.000000e+00  7.4608633 0.669 0.006  0.000000e+00      13       ENHO
## 43  0.000000e+00  7.2190791 0.615 0.015  0.000000e+00      14     CDKN1C
## 44  0.000000e+00  8.4043944 0.582 0.002  0.000000e+00      14       HES4
## 45  0.000000e+00  5.3104486 0.590 0.018  0.000000e+00      14     NEURL1
## 46  0.000000e+00 14.8130528 0.984 0.000  0.000000e+00      15     LRRC26
## 47  0.000000e+00 11.5556003 0.852 0.000  0.000000e+00      15        SCT
```

```
## 48  0.000000e+00 13.7058434 0.738 0.000  0.000000e+00      15 AC097375.1
## 49  0.000000e+00 15.5206370 0.684 0.000  0.000000e+00      16 AP001189.1
## 50  0.000000e+00 15.1124072 0.544 0.000  0.000000e+00      16      LTBP1
## 51  0.000000e+00 14.1309560 0.509 0.000  0.000000e+00      16 AC090409.1
## 52  1.804256e-49  3.2903916 0.625 0.048  3.649829e-45      17       PAX5
## 53  1.550939e-48  2.8329368 0.844 0.088  3.137394e-44      17     NIBAN3
## 54  6.459055e-43  3.2451295 0.656 0.060  1.306602e-38      17       CD22
## 55  0.000000e+00 13.9184710 0.833 0.000  0.000000e+00      18       NPR3
## 56  0.000000e+00 13.8490035 0.500 0.000  0.000000e+00      18       CPA3
## 57  0.000000e+00 14.2839346 0.500 0.000  0.000000e+00      18 AC011139.1
```

```
top3_markers <- as.data.frame(all.markers %>% group_by(cluster) %>% top_n(n = 3, wt = avg_log2
top3_markers
```

```
##            p_val avg_log2FC pct.1 pct.2     p_val_adj cluster       gene
## 1   0.000000e+00  1.8494078 0.922 0.321  0.000000e+00       0       LEF1
## 2   0.000000e+00  2.0267240 0.661 0.209  0.000000e+00       0        MAL
## 3   0.000000e+00  1.8952301 0.565 0.185  0.000000e+00       0      TRAT1
## 4   0.000000e+00  2.3516192 0.817 0.250  0.000000e+00       1       AQP3
## 5  4.694613e-303  2.0148682 0.769 0.309 9.496733e-299       1      ITGB1
## 6  7.108404e-242  2.0920652 0.564 0.167 1.437959e-237       1     GPR183
## 7   0.000000e+00  3.2214159 0.909 0.127  0.000000e+00       2     LGALS2
## 8   0.000000e+00  3.1404242 0.553 0.064  0.000000e+00       2   TMEM176B
## 9   0.000000e+00  3.6756538 0.520 0.048  0.000000e+00       2       PID1
## 10  0.000000e+00  3.0208504 0.986 0.214  0.000000e+00       3    S100A12
## 11  0.000000e+00  2.9398611 1.000 0.293  0.000000e+00       3     S100A9
## 12  0.000000e+00  3.1815253 0.612 0.080  0.000000e+00       3     CYP1B1
## 13  0.000000e+00  4.7656333 0.990 0.066  0.000000e+00       4       CD8B
## 14  0.000000e+00  3.5941504 0.943 0.082  0.000000e+00       4       CD8A
## 15  0.000000e+00  5.1407883 0.768 0.034  0.000000e+00       4  LINC02446
## 16  0.000000e+00  7.2261767 0.933 0.012  0.000000e+00       5      TCL1A
## 17  0.000000e+00  6.4686598 0.891 0.021  0.000000e+00       5      FCER2
## 18  0.000000e+00  6.5260654 0.647 0.013  0.000000e+00       5       IGHD
## 19 6.238346e-306  3.0263411 0.992 0.347 1.261955e-301       6     S100A8
## 20 1.061347e-131  2.7763829 0.622 0.219 2.146999e-127       6       RGS2
## 21  1.718461e-95  2.7133474 0.503 0.181  3.476274e-91       6       RBP7
## 22  0.000000e+00  8.0215278 0.903 0.008  0.000000e+00       7      KLRF1
## 23  0.000000e+00  8.7861760 0.745 0.004  0.000000e+00       7     SH2D1B
## 24  0.000000e+00  7.9224906 0.596 0.004  0.000000e+00       7     PRSS23
## 25  0.000000e+00  3.5351653 0.875 0.105  0.000000e+00       8       CCL5
## 26 7.948990e-240  3.2435539 0.555 0.076 1.608001e-235       8      CXCR3
## 27 2.636578e-218  3.1614914 0.506 0.067 5.333533e-214       8      KLRC4
## 28  0.000000e+00  4.6348346 0.977 0.106  0.000000e+00       9      KLRB1
## 29  0.000000e+00  7.6860891 0.578 0.004  0.000000e+00       9    SLC4A10
## 30  0.000000e+00  4.8403765 0.512 0.021  0.000000e+00       9    TRAV1-2
## 31  0.000000e+00  5.5189084 0.600 0.030  0.000000e+00      10     JCHAIN
## 32  0.000000e+00  5.5907591 0.576 0.013  0.000000e+00      10       AIM2
## 33  0.000000e+00  7.8326409 0.544 0.003  0.000000e+00      10  TNFRSF13B
## 34  9.527043e-28  2.8636919 0.556 0.409  1.927226e-23      11       AAK1
```

```
## 35  4.046628e-26  2.9784009 0.523 0.363  8.185924e-22      11     BCL11B
## 36  1.550118e-23  2.9249776 0.523 0.391  3.135733e-19      11       NKTR
## 37  4.056651e-50  0.9458402 0.602 0.170  8.206198e-46      12      STAB1
## 38  2.414341e-41  0.9508349 0.572 0.178  4.883971e-37      12    CLEC12A
## 39  2.634484e-31  0.9018058 0.500 0.169  5.329298e-27      12     ZNF467
## 40  0.000000e+00  8.3902588 0.882 0.008  0.000000e+00      13     FCER1A
## 41  0.000000e+00  6.2398957 0.864 0.025  0.000000e+00      13    CLEC10A
## 42  0.000000e+00  7.4608633 0.669 0.006  0.000000e+00      13       ENHO
## 43  0.000000e+00  7.2190791 0.615 0.015  0.000000e+00      14     CDKN1C
## 44  0.000000e+00  8.4043944 0.582 0.002  0.000000e+00      14       HES4
## 45  0.000000e+00  5.3104486 0.590 0.018  0.000000e+00      14     NEURL1
## 46  0.000000e+00 14.8130528 0.984 0.000  0.000000e+00      15     LRRC26
## 47  0.000000e+00 11.5556003 0.852 0.000  0.000000e+00      15        SCT
## 48  0.000000e+00 13.7058434 0.738 0.000  0.000000e+00      15 AC097375.1
## 49  0.000000e+00 15.5206370 0.684 0.000  0.000000e+00      16 AP001189.1
## 50  0.000000e+00 15.1124072 0.544 0.000  0.000000e+00      16      LTBP1
## 51  0.000000e+00 14.1309560 0.509 0.000  0.000000e+00      16 AC090409.1
## 52  1.804256e-49  3.2903916 0.625 0.048  3.649829e-45      17       PAX5
## 53  1.550939e-48  2.8329368 0.844 0.088  3.137394e-44      17     NIBAN3
## 54  6.459055e-43  3.2451295 0.656 0.060  1.306602e-38      17       CD22
## 55  0.000000e+00 13.9184710 0.833 0.000  0.000000e+00      18       NPR3
## 56  0.000000e+00 13.8490035 0.500 0.000  0.000000e+00      18       CPA3
## 57  0.000000e+00 14.2839346 0.500 0.000  0.000000e+00      18 AC011139.1
```

11 9. Cell type annotation using SingleR

We can try automatic annotation with SingleR, using reference dataset from celldex package

```
monaco.ref <- celldex::MonacoImmuneData()
```

```
## see ?celldex and browseVignettes('celldex') for documentation
```

```
## loading from cache
```

```
## see ?celldex and browseVignettes('celldex') for documentation
```

```
## loading from cache
```

convert our Seurat object to single cell experiment for convinience:

```
sce <- as.SingleCellExperiment(DietSeurat(filtered_pbmc))
sce
```

```
## class: SingleCellExperiment
## dim: 20229 10338
## metadata(0):
## assays(3): counts logcounts scaledata
## rownames(20229): AL627309.1 AL627309.5 ... AC007325.4 AC007325.2
## rowData names(0):
## colnames(10338): AAACCTGAGACAGACC-1 AAACCTGAGCGATAGC-1 ...
```

```
##   TTTGTCATCTCTGAGA-1 TTTGTCATCTTCGAGA-1
## colData names(14): seq_folder nUMI ... seurat_clusters ident
## reducedDimNames(0):
## mainExpName: RNA
## altExpNames(0):
```

```
monaco.main <- SingleR(test = sce,assay.type.test = 1,ref = monaco.ref,labels = monaco.ref$labe
monaco.fine <- SingleR(test = sce,assay.type.test = 1,ref = monaco.ref,labels = monaco.ref$labe
```

```
table(monaco.main$pruned.labels)
```

```
##
##         B cells    CD4+ T cells    CD8+ T cells Dendritic cells       Monocytes
##             857            3306            1519             267            3128
##        NK cells     Progenitors         T cells
##             429              54             465
```

```
table(monaco.fine$pruned.labels)
```

```
##
##    Central memory CD8 T cells           Classical monocytes
##                           224                          2823
##   Effector memory CD8 T cells             Exhausted B cells
##                            57                            22
##     Follicular helper T cells        Intermediate monocytes
##                           381                           167
##         Low-density basophils                   MAIT cells
##                             2                           220
##       Myeloid dendritic cells                 Naive B cells
##                           208                           645
##             Naive CD4 T cells             Naive CD8 T cells
##                          2265                          1108
##          Natural killer cells       Non classical monocytes
##                           442                            57
##   Non-switched memory B cells            Non-Vd2 gd T cells
##                           130                            32
##                 Plasmablasts  Plasmacytoid dendritic cells
##                             7                            56
##              Progenitor cells       Switched memory B cells
##                            50                            58
##           T regulatory cells Terminal effector CD4 T cells
##                           230                             3
## Terminal effector CD8 T cells                     Th1 cells
##                            23                           105
##                Th1/Th17 cells                   Th17 cells
##                           175                           198
##                     Th2 cells                Vd2 gd T cells
##                           214                           193
```

Add the annotations to the Seurat Object metadata.

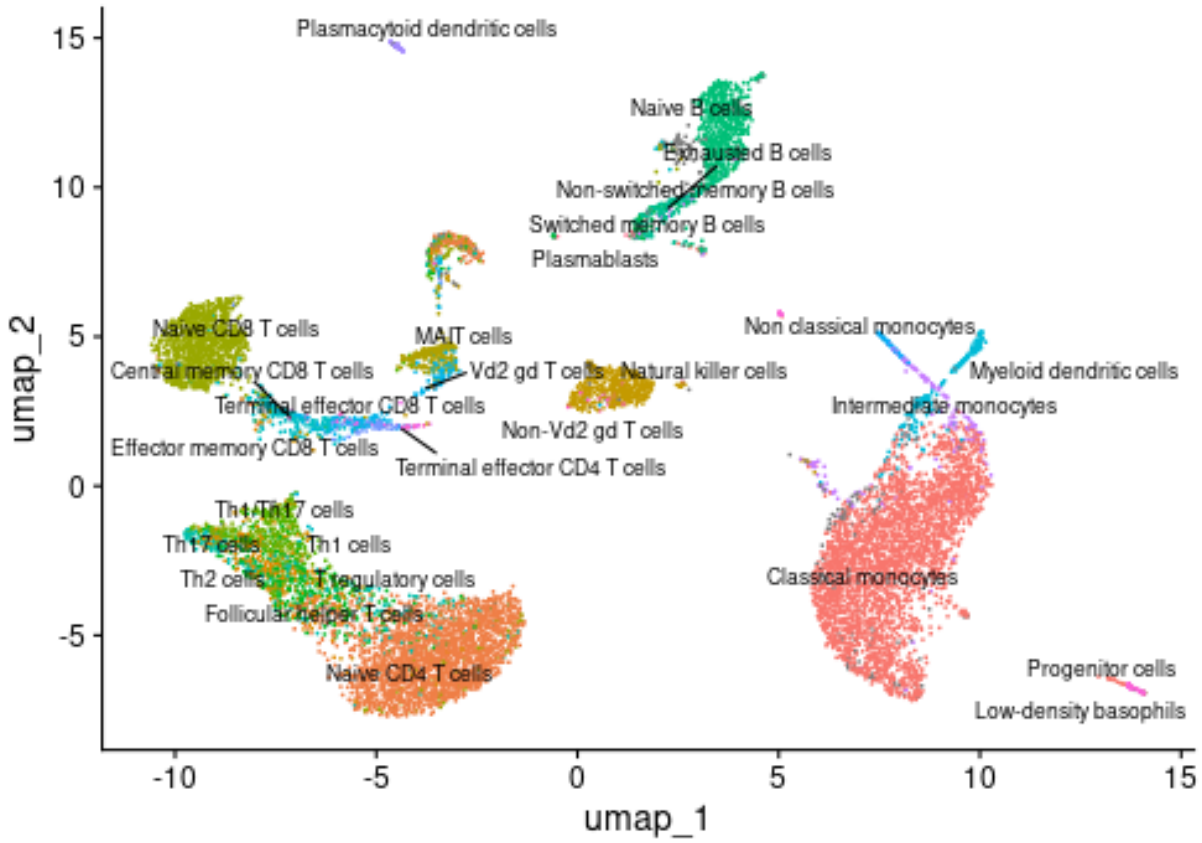
```
filtered_pbmc@meta.data$monaco.main <- monaco.main$pruned.labels
filtered_pbmc@meta.data$monaco.fine <- monaco.fine$pruned.labels
```

```
srat <- SetIdent(filtered_pbmc, value = "monaco.fine")
p16 <- DimPlot(srat, label = TRUE, repel = TRUE, label.size = 3) + NoLegend()
```

```
# Save the plot
ggsave("plots/annotation_plot.png", plot = p16)
```

```
## Saving 7 x 5 in image
```

```
p16
```



Extract top 10 cell types and their top 10 expressed genes

```
# Get the top 10 cell types
top_cell_types <- names(sort(table(filtered_pbmc@meta.data$monaco.main), decreasing = TRUE))[1
```

```
# Find top 10 expressed genes for each cell type
top_genes <- lapply(top_cell_types, function(cell_type) {
  cells <- which(filtered_pbmc@meta.data$monaco.main == cell_type)
  avg_exp <- rowMeans(GetAssayData(filtered_pbmc, slot = "data")[, cells])
  names(sort(avg_exp, decreasing = TRUE))[1:10]
})
```

```
## Warning: The `slot` argument of `GetAssayData()` is deprecated as of SeuratObject 5.0.0.
## i Please use the `layer` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# Combine the list of top genes
top_genes <- unique(unlist(top_genes))

# Create dot plot based on RNA expression
p17 <- DotPlot(filtered_pbmc, features = top_genes, assay = "RNA") +
  ggtitle("Top 10 Cell Types and Their Top 10 Expressed Genes") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

```
## Warning: The following requested variables were not found: NA
```

```
# Save the plot
ggsave("plots/dot_plot_top_genes.png", plot = p17)
```

```
## Saving 7 x 5 in image
```

```
# Display the plot
print(p17)
```

11.1 Differential expression analysis

```
# Find markers
oupMarker <- FindAllMarkers(filtered_pbmc, only.pos = TRUE, logfc.threshold = 1.0, min.pct = 0
```

```
## Calculating cluster 0
## Calculating cluster 1
## Calculating cluster 2
## Calculating cluster 3
## Calculating cluster 4
## Calculating cluster 5
## Calculating cluster 6
## Calculating cluster 7
## Calculating cluster 8
## Calculating cluster 9
## Calculating cluster 10
## Calculating cluster 11
## Calculating cluster 12
## Calculating cluster 13
## Calculating cluster 14
## Calculating cluster 15
## Calculating cluster 16
## Calculating cluster 17
## Calculating cluster 18
```

```
oupMarker <- data.table(oupMarker)
oupMarker$pct.diff = oupMarker$pct.1 - oupMarker$pct.2
oupMarker <- oupMarker[, c("cluster","gene","avg_log2FC","pct.1","pct.2","pct.diff","p_val","p
```

```
# Check if known genes are in the marker gene list
knownGenes <- c("CD34","CRHBP","GATA1", "CD14","IRF8","CD19","CD4","CD8B","GNLY")
oupMarker[gene %in% knownGenes]
```

```
##     cluster   gene avg_log2FC pct.1 pct.2 pct.diff         p_val     p_val_adj
##      <fctr> <char>      <num> <num> <num>    <num>         <num>         <num>
##  1:       2   CD14   2.309717 0.936 0.189    0.747  0.000000e+00  0.000000e+00
##  2:       2    CD4   1.321495 0.719 0.321    0.398 8.705648e-176 1.761066e-171
##  3:       3   CD14   2.445486 0.914 0.201    0.713  0.000000e+00  0.000000e+00
##  4:       4   CD8B   4.765633 0.990 0.066    0.924  0.000000e+00  0.000000e+00
##  5:       5   CD19   5.429974 0.682 0.022    0.660  0.000000e+00  0.000000e+00
```

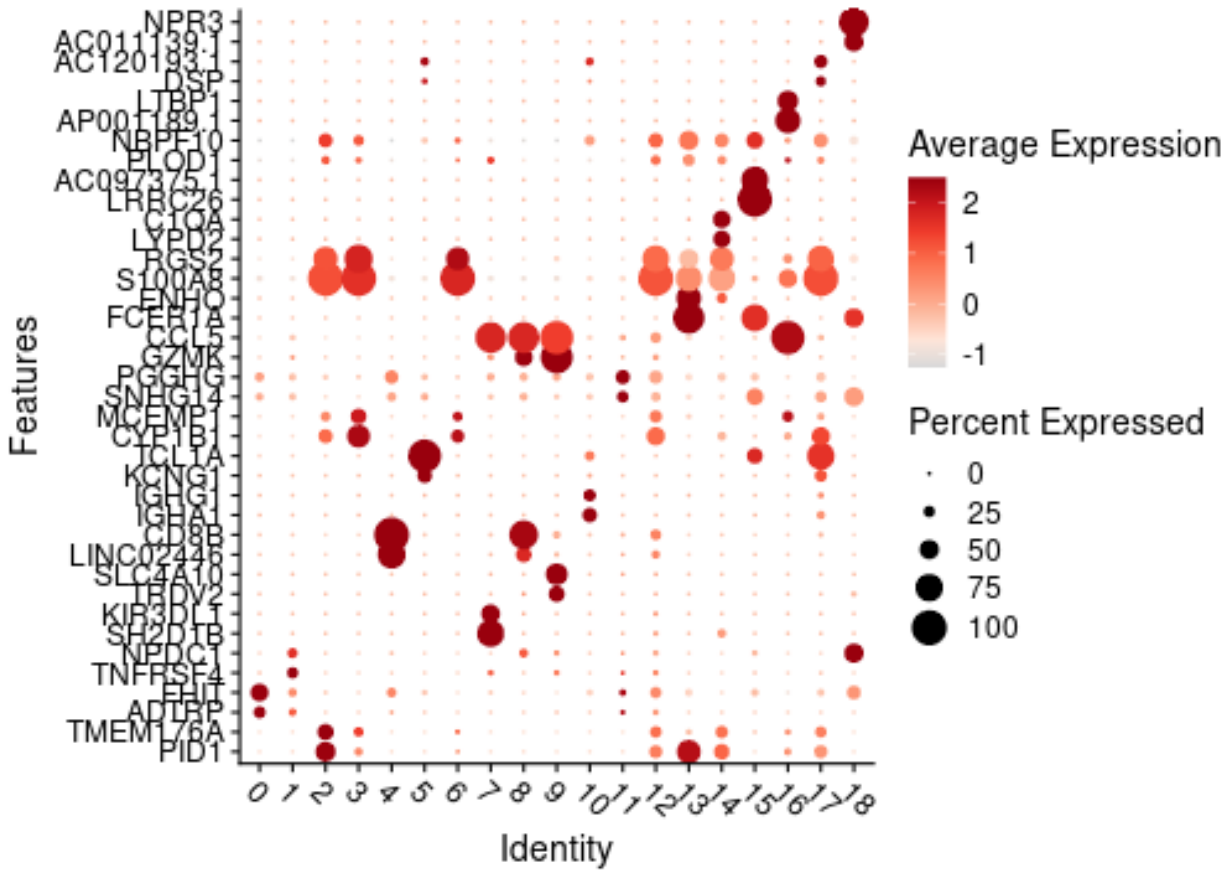
```
##  6:       5   IRF8   2.308674 0.677 0.146    0.531 1.975186e-299 3.995604e-295
##  7:       6   CD14   1.756712 0.639 0.260    0.379  9.458151e-92  1.913289e-87
##  8:       7   GNLY   7.803666 0.998 0.028    0.970  0.000000e+00  0.000000e+00
##  9:       8   CD8B   2.360840 0.801 0.129    0.672 1.416960e-275 2.866369e-271
## 10:      10   CD19   3.416086 0.660 0.053    0.607 1.839517e-306 3.721158e-302
## 11:      10   IRF8   1.733308 0.656 0.170    0.486  1.574271e-86  3.184592e-82
## 12:      13   IRF8   1.510344 0.692 0.174    0.518  3.328580e-52  6.733383e-48
## 13:      15   IRF8   5.456532 1.000 0.177    0.823  3.688025e-89  7.460505e-85
## 14:      15    CD4   1.569806 0.918 0.365    0.553  1.332525e-18  2.695566e-14
## 15:      17   CD19   2.272918 0.688 0.065    0.623  6.906437e-41  1.397103e-36
## 16:      17   IRF8   1.215874 0.688 0.180    0.508  2.307764e-12  4.668376e-08
## 17:      18   CD34  10.724063 0.917 0.000    0.917  0.000000e+00  0.000000e+00
## 18:      18  CRHBP   8.819145 0.583 0.001    0.582  0.000000e+00  0.000000e+00
## 19:      18  GATA1   7.608127 0.250 0.001    0.249 1.658341e-141 3.354657e-137
```

```
# Get top genes for each cluster and do dot plot / violin plot
oupMarker$cluster = factor(oupMarker$cluster, levels = unique(filtered_pbmc$seurat_clusters))
oupMarker = oupMarker[order(cluster, -avg_log2FC)]
genes.to.plot <- unique(oupMarker[cluster %in% unique(filtered_pbmc$seurat_clusters), head(.SD
```

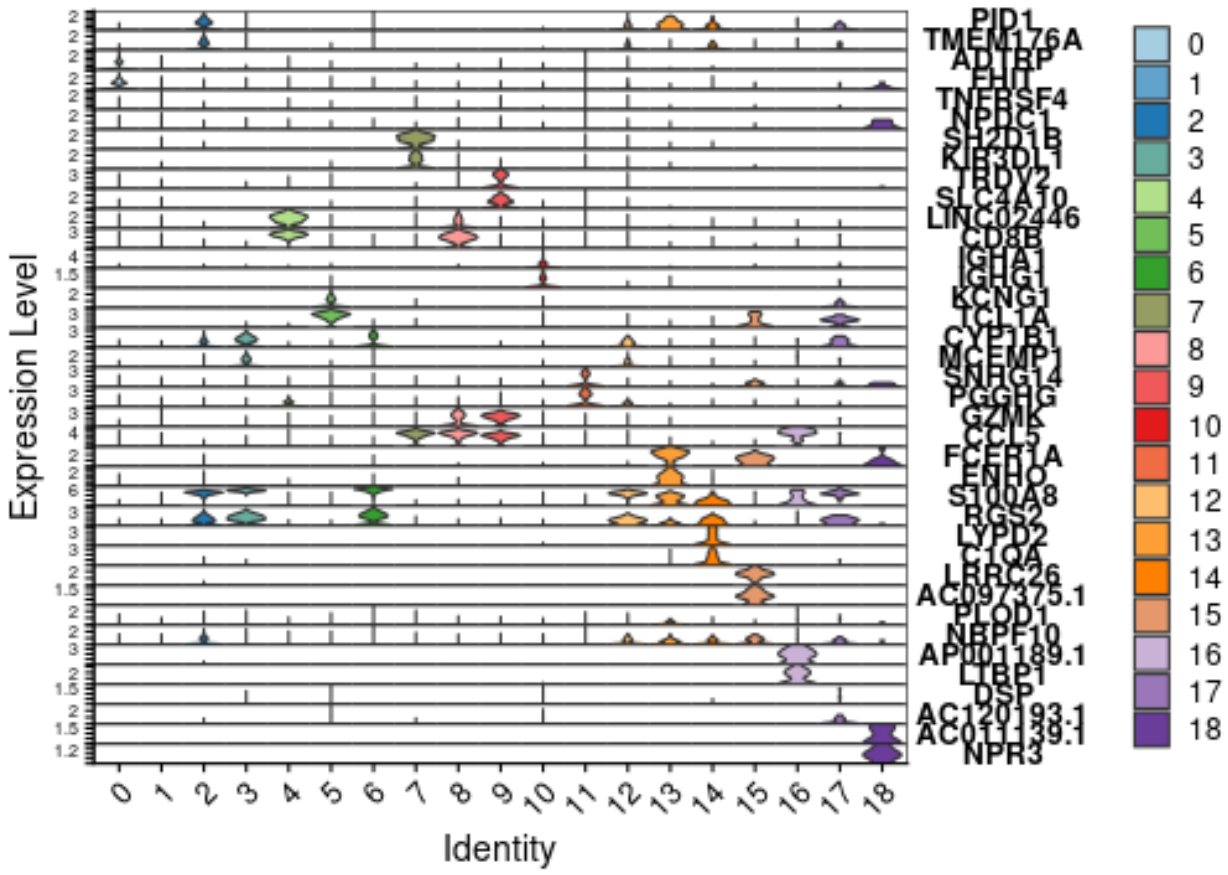
```
# Set color for gene expression
colGEX = c("grey85", brewer.pal(7, "Reds"))

p1 <- DotPlot(filtered_pbmc, group.by = "seurat_clusters", features = genes.to.plot) +
  coord_flip() + scale_color_gradientn(colors = colGEX) +
  theme(axis.text.x = element_text(angle = -45, hjust = 0))
```

```
## Scale for colour is already present.
## Adding another scale for colour, which will replace the existing scale.
```

```
# Setup color palettes
nClust <- uniqueN(Idents(filtered_pbmc))
colCls <- colorRampPalette(brewer.pal(n = 10, name = "Paired"))(nClust)

p2 <- VlnPlot(filtered_pbmc, group.by = "seurat_clusters", fill.by = "ident", cols = colCls, f

p1
```


Features
NPR3
AC011139.1
AC120193.1
DSP
LTBP1
AP001189.1
NBPF10
PLOD1
AC097375.1
LRRC26
C1QA
LYPD2
RGS2
S100A8
ENHO
FCER1A
CCL5
GZMK
PGGHG
SNHG14
MCEMP1
CYP1B1
TCL1A
KCNG1
IGHG1
IGHA1
IGHA2
CD8B
LINC02446
SLC4A10
TRDV2
KIR3DL1
SH2D1B
NPDC1
TNFRSF4
EHD1
ADTRP
TMEM176A
PID1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Identity
Average Expression
2
1
0
-1
Percent Expressed
0
25
50
75
100

p2



A work by Marta Meroño

mmeronorafel@gmail.com