all_alevin

Syrus

9/9/2024

Loading libraries

```
suppressPackageStartupMessages({
    library(fishpond)
    library(tximport)
    library(devtools)
    library(ggplot2)
    library(Seurat)
    library(tidyverse)
    library(gridExtra)
    library(celldex)
    library(SingleR)
    library(pheatmap)
})
```

```
assympotomatic.files <- file.path("~/assymptomatic_alevin/alevin/quants_mat.gz")
severe.files <- file.path("~/symptomatic_severe_alevin/alevin/quants_mat.gz")
moderate.files <- file.path("~/symptomatic_moderate_alevin/alevin/quants_mat.gz")
recovered.files <- file.path("~/symptomatic_severe_recovered_alevin/alevin/quants_mat.gz")
healthy.files <- file.path("~/healthy_controls_alevin/alevin/quants_mat.gz")
file.exists(c(assympotomatic.files, severe.files, moderate.files, recovered.files, healthy.files))
```

```
## [1] TRUE TRUE TRUE TRUE TRUE
```

```
assympotomatic.txi <- tximport(files = assympotomatic.files, type = "alevin")
```

```
## reading in alevin gene-level counts across cells with 'eds'
```

```
severe.txi <- tximport(files = severe.files, type = "alevin")
```

```
## reading in alevin gene-level counts across cells with 'eds'
```

```
moderate.txi <- tximport(files = moderate.files, type = "alevin")
```

```
## reading in alevin gene-level counts across cells with 'eds'
```

```
recovered.txi <- tximport(files = recovered.files, type = "alevin")
```

```
## reading in alevin gene-level counts across cells with 'eds'
```

```
healthy.txi <- tximport(files = healthy.files, type = "alevin")
```

```
## reading in alevin gene-level counts across cells with 'eds'
```

```
severe <- CreateSeuratObject(severe.txi$counts, project = "severe")
moderate <- CreateSeuratObject(moderate.txi$counts, project = "moderate")
recovered <- CreateSeuratObject(recovered.txi$counts, project = "recovered")
```

```
assympotomatic <- CreateSeuratObject(assympotomatic.txi$counts,
                                     project = "assympotomatic")
healthy <- CreateSeuratObject(healthy.txi$counts, project = "healthy")

object <- merge(healthy, y = c(assympotomatic, moderate, severe, recovered),
                add.cell.ids = c("healthy", "assympotomatics", "moderate",
                                 "severe", "recovered"),
                project = "covid_19_norms")
object
```

```
## An object of class Seurat
## 35940 features across 8333 samples within 1 assay
## Active assay: RNA (35940 features, 0 variable features)
##  5 layers present: counts.healthy, counts.assympotomatic, counts.moderate, counts.severe, counts.rec
```

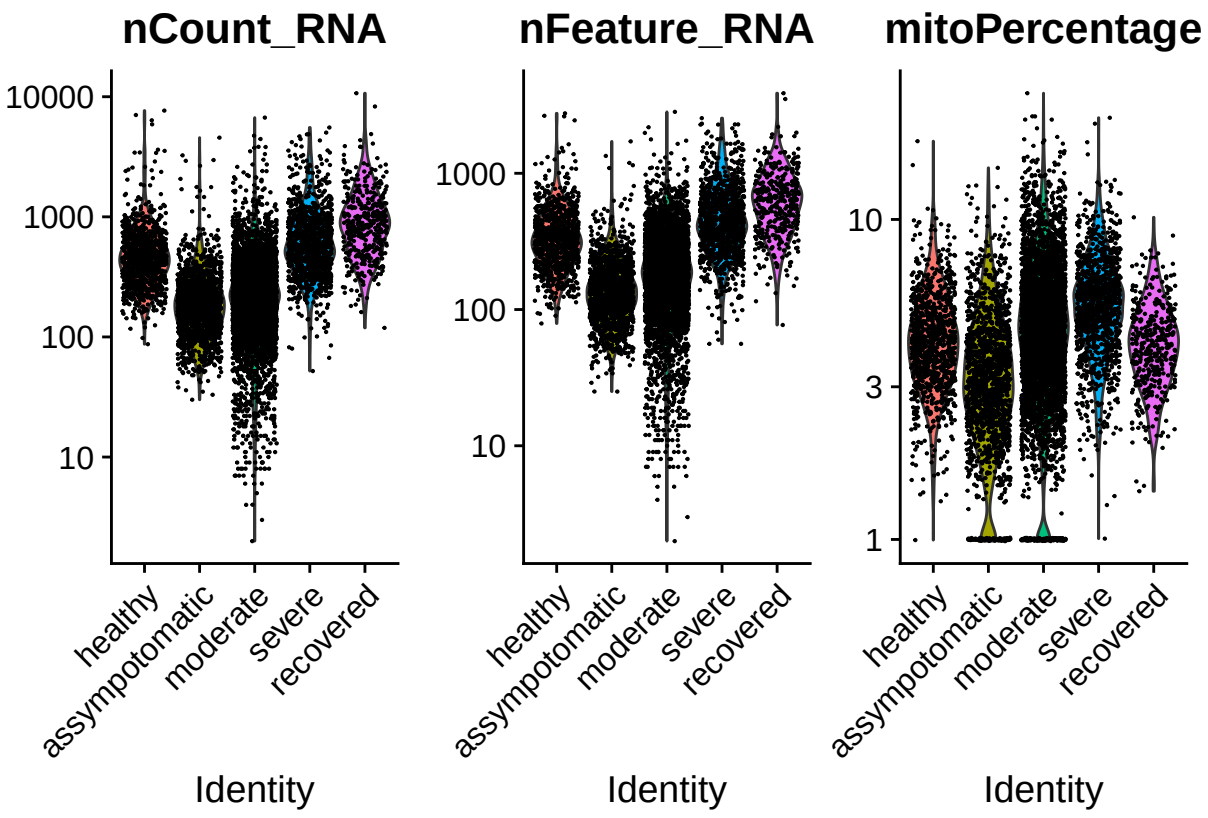
```
#View(object@meta.data)
# Create a sample column
object$sample <- rownames(object@meta.data)
# Split sample column
object@meta.data <- separate(object@meta.data, col = "sample",
                             into = c("Patient", "Barcode"), sep = "_")
```

```
# Calculate mitochodrial percentage
object$mitoPercentage <- PercentageFeatureSet(object, pattern = "^MT")
#View(object@meta.data)
```

Quality control visualisation

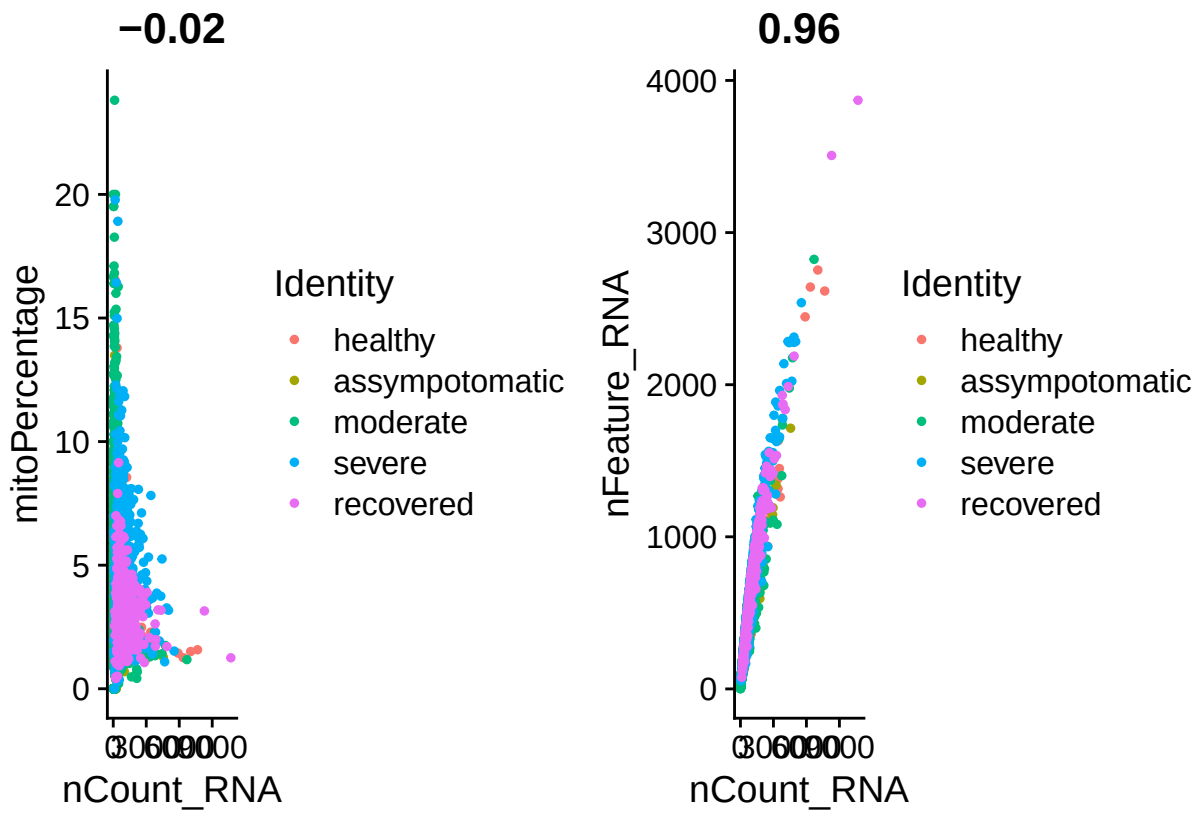
```
# Quality control
#VlnPlot(object, features = "nCount_RNA", pt.size = 0.1, log = TRUE)
#VlnPlot(object, features = "nFeature_RNA", pt.size = 0.1, log = TRUE)
#VlnPlot(object, "mitoPercentage", pt.size = 0.1)

VlnPlot(object, features = c("nCount_RNA", "nFeature_RNA", "mitoPercentage"),
        pt.size = 0.1, log = TRUE)
```

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



Using featurescatter to visualise the relatioship between features

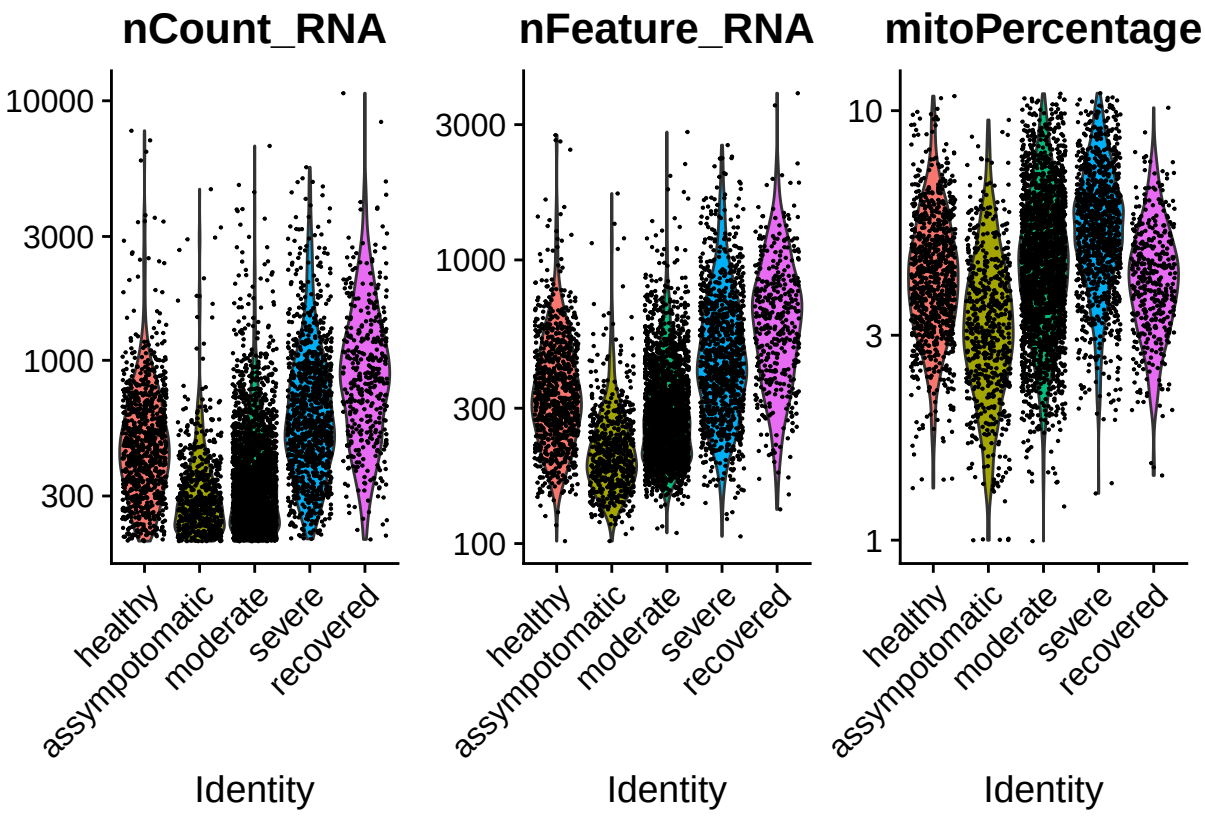
```
p1 <- FeatureScatter(object, feature1 = "nCount_RNA", feature2 = "mitoPercentage")
p2 <- FeatureScatter(object, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")

p1 + p2
```



Filtering

```
object_filtered <- subset(object,
                          subset = mitoPercentage < 10 & nFeature_RNA > 100 &
                            nCount_RNA > 200)
# After filtering
VlnPlot(object_filtered, features = c("nCount_RNA", "nFeature_RNA",
                                      "mitoPercentage"),
        pt.size = 0.1, log = TRUE)
```

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




```
object
```

```
## An object of class Seurat
## 35940 features across 8333 samples within 1 assay
## Active assay: RNA (35940 features, 0 variable features)
##  5 layers present: counts.healthy, counts.assympotomatic, counts.moderate, counts.severe, counts.rec
```

```
object_filtered
```

```
## An object of class Seurat
## 35940 features across 5303 samples within 1 assay
## Active assay: RNA (35940 features, 0 variable features)
##  5 layers present: counts.healthy, counts.assympotomatic, counts.moderate, counts.severe, counts.rec
```

Normalizing

After removing unwanted cells from the dataset, the next step is to normalize the data. By default, we employ a global-scaling normalization method "LogNormalize" that normalizes the feature expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result. This method relies on the assumption that each cell contains the same number of RNA molecules. Results are stored in object[["RNA"]]$data.

```
#object_filtered <- NormalizeData(object = object_filtered, normalization.method = "LogNormalize", scal
object_filtered <- NormalizeData(object_filtered)
```

```
## Normalizing layer: counts.healthy
```

```
## Normalizing layer: counts.assympotomatic
```

```
## Normalizing layer: counts.moderate
## Normalizing layer: counts.severe
## Normalizing layer: counts.recovered
```

Identifing highly variable features

We then subset features or gene that are highly expressed in some cells and lowly expressed in others. These highlight biological signals in single-cell datasets. By default, this method returns 2000 features per dataset.

```
#object_filtered <- FindVariableFeatures(object_filtered, selection.method = "vst", nfeatures = 2000)
object_filtered <- FindVariableFeatures(object = object_filtered)
```

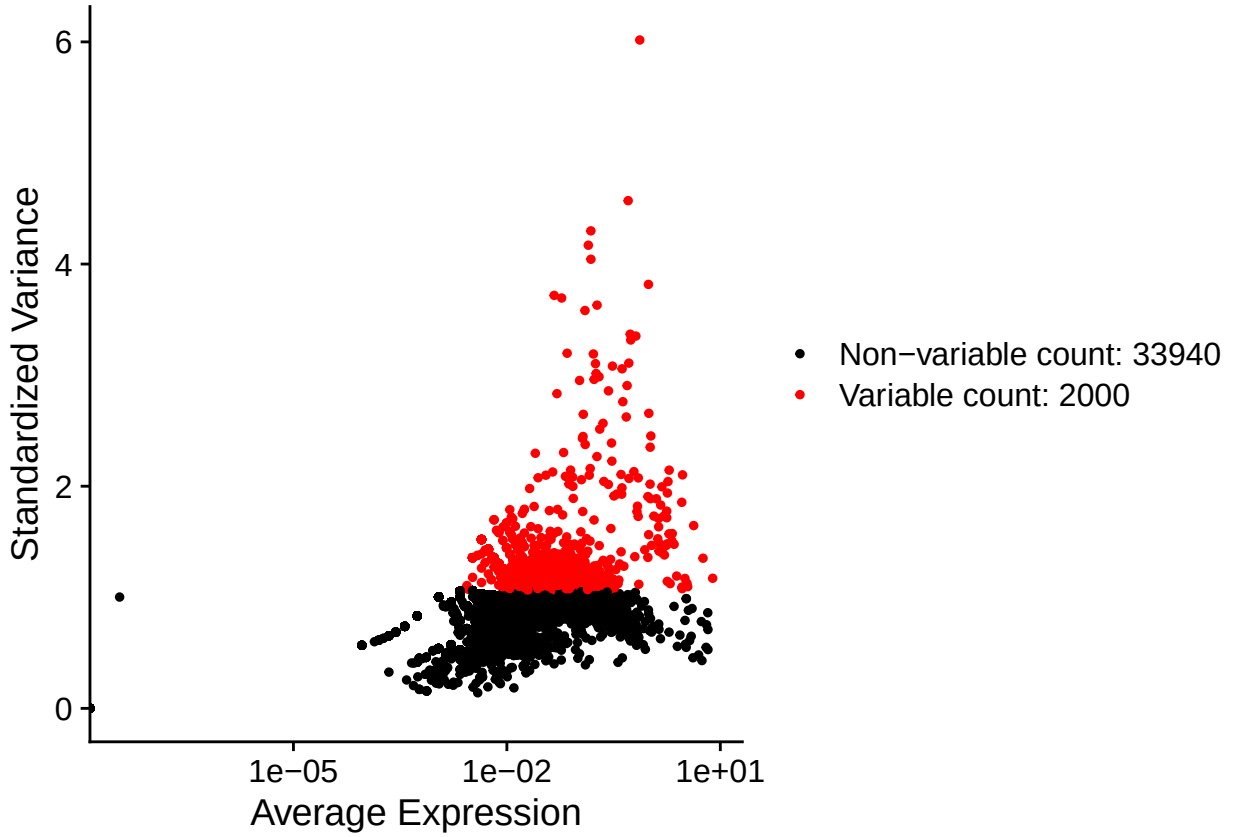
```
## Finding variable features for layer counts.healthy
## Finding variable features for layer counts.assympotomatic
## Finding variable features for layer counts.moderate
## Finding variable features for layer counts.severe
## Finding variable features for layer counts.recovered
```

```
# Top 10 variable features
Top10 <- head(VariableFeatures(object_filtered), 10)
Top10
```

```
##  [1] "IGHV5-10-1" "IGKV1D-13"  "E2F7"       "IGHA2"      "IGHV1-58"
##  [6] "IGKV6D-21"  "IGLV1-41"   "IGKV3D-20"  "IGLV3-21"   "AC145029.2"
```

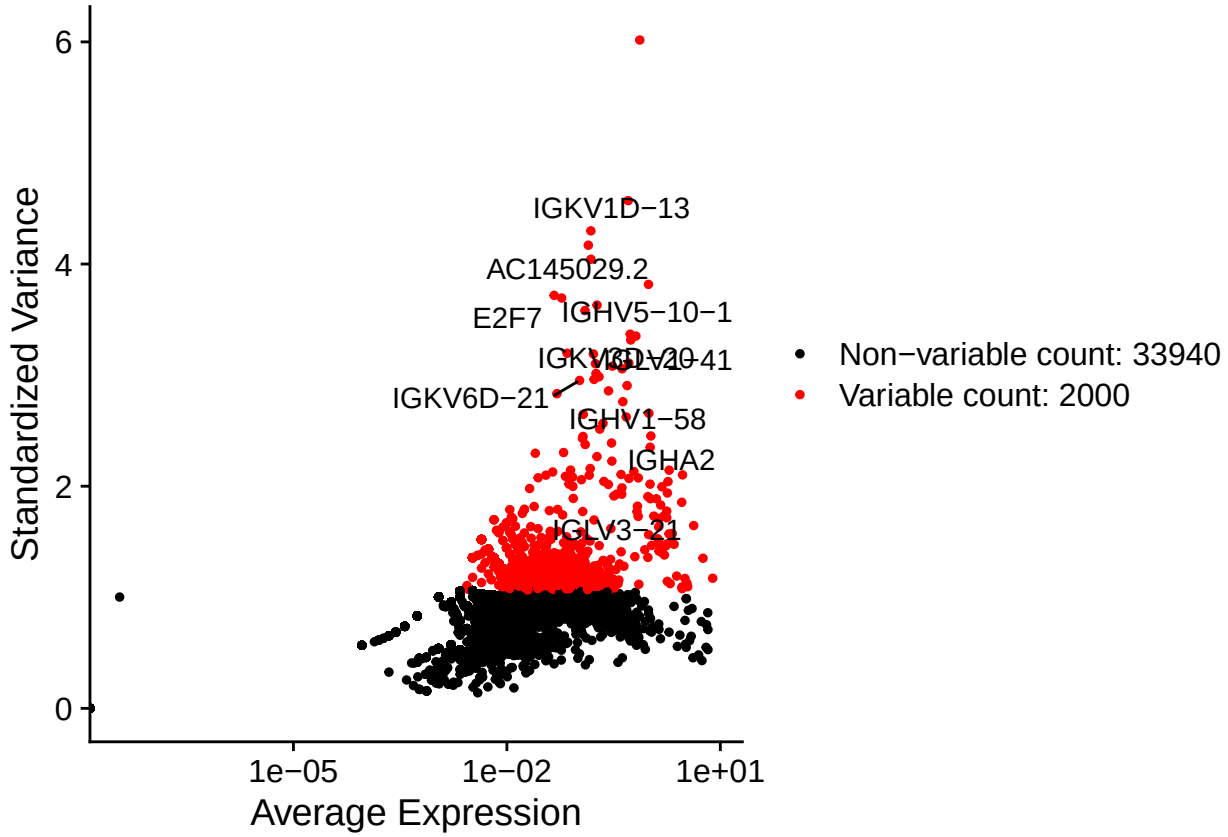
```
# PLoting variable features with labeled top 10 highly variable features
p3 <- VariableFeaturePlot(object_filtered)
p4 <- LabelPoints(plot = p3, points = Top10, repel = TRUE, xnudge = 0, ynudge = 0)

p3
```

```
## Warning in scale_x_log10(): log-10 transformation introduced infinite values.
```




```
p4
```

```
## Warning in scale_x_log10(): log-10 transformation introduced infinite values.
```




```
#p3 + p4
```

Scaling the dataset

We perform linear transformation the variable features before dimensional reduction techniques because; it shifts the expression of each gene so that the mean expression across cells is zero and also the variance across cells is one - so that highly expressed genes don't dominate. Results are stored in object_filtered[["RNA]]$scale.data

```
#all.genes.list <- rownames(object_filtered)
#object_filtered <- ScaleData(object_filtered, features = all.genes.list)
object_filtered <- ScaleData(object = object_filtered)
```

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

```
# ScaleData function can also be used to remove unwanted sources of variation ie heterogeneity associat
#object_filtered <- ScaleData(object_filtered, vars.to.regress = "mitoPercentage")
```

Perfoming linear dimensional reduction

Only the variable features are used as input as Seurat outputs a list of gene with the most positive (correlation) and negative loadings (anti-correlation) across the dataset.

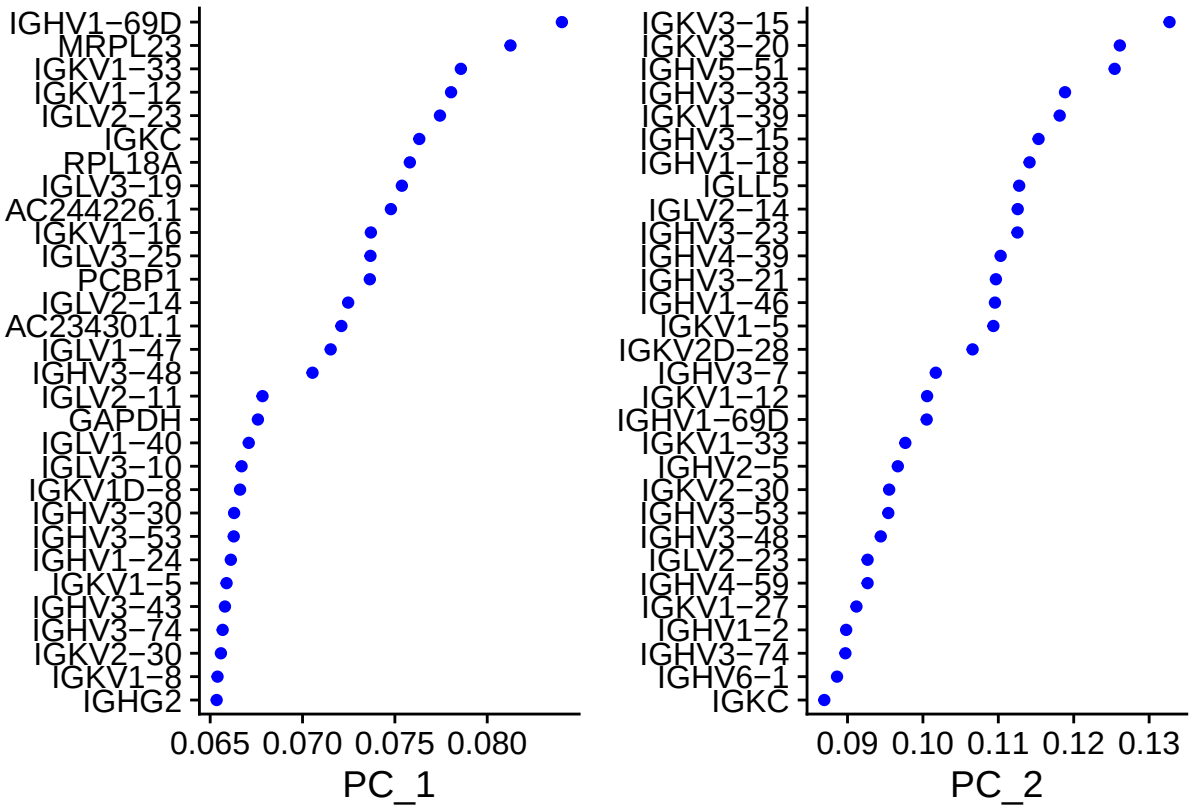
```
#object_filtered <- RunPCA(object_filtered, features = VariableFeatures(object = object_filtered))
object_filtered <- RunPCA(object = object_filtered)
```

```
## PC_ 1
## Positive:  IGHV1-69D, MRPL23, IGKV1-33, IGKV1-12, IGLV2-23, IGKC, RPL18A, IGLV3-19, AC244226.1, IGKV
##     IGLV3-25, PCBP1, IGLV2-14, AC234301.1, IGLV1-47, IGHV3-48, IGLV2-11, GAPDH, IGLV1-40, IGLV3-10
##     IGKV1D-8, IGHV3-30, IGHV3-53, IGHV1-24, IGKV1-5, IGHV3-43, IGHV3-74, IGKV2-30, IGKV1-8, IGHG2
## Negative:  IGLC1, IGKV3D-15, IGKV2D-24, E2F7, ADAMTS1, ZNF66, KLHL26, CCDC77, SAP25, CABP4
##     EPB41L4A, TPM2, SARDH, KIT, FCRL5, RPP38, SGF29, FN1, ZFP41, FAM200A
##     NAA16, KIF1C, CCDC17, CEP57L1, MUTYH, ASPDH, CNR2, FAM153A, OPRL1, IGIP
## PC_ 2
## Positive:  IGKV3-15, IGKV3-20, IGHV5-51, IGHV3-33, IGKV1-39, IGHV3-15, IGHV1-18, IGLL5, IGLV2-14, IG
##     IGHV4-39, IGHV3-21, IGHV1-46, IGKV1-5, IGKV2D-28, IGHV3-7, IGKV1-12, IGHV1-69D, IGKV1-33, IGHV2-5
##     IGKV2-30, IGHV3-53, IGHV3-48, IGLV2-23, IGHV4-59, IGKV1-27, IGHV1-2, IGHV3-74, IGHV6-1, IGKC
## Negative:  IFI30, CALR, LRRC25, FTH1, STX11, LRP1, PECAM1, WARS, MYO9B, PFN1
##     EIF3A, ITGAL, LILRB2, LSP1, PLCB2, MT-ATP6, DDX39B, ARPC1B, MT-CO2, EIF4A1
##     RPL3, PAG1, BIN2, POLR2A, ACADVL, SPI1, CCDC88B, SND1, LYN, CCDC88C
## PC_ 3
## Positive:  IGHV3-43, IGHV3-13, GSE1, DDX39B, ZAP70, IGHV1-2, PLCB2, SP3, SEC24C, AC119396.1
##     AC233755.2, ITGAL, CCDC88B, FLII, AC129492.1, IGHV3-74, PTK2B, IGHV1-18, EIF3A, PIK3R5
##     ARHGAP4, IGKV1-9, ESYT1, TNFAIP3, IGHV3-21, SMG1P1, WDFY4, NLRP1, IGKV2-29, KMT2D
## Negative:  FTL, MT-CO2, MRPL23, RPL18A, RPL10, S100A9, RPS12, RPS18, GAPDH, RPS19
##     RPL19, RPS2, PCBP1, IGLV3-25, RPS19BP1, GABARAP, RPS3A, RPL41, TMSB10, RPS27A
##     RPS4X, MT-ATP6, RPL7A, RPL26, TESC, CST3, FTH1, NT5C, IGHG1, RPL18
## PC_ 4
## Positive:  IKBKG, TESC, MRPL23, CCDC88B, ZAP70, NOP53, IGKV3D-20, AC145029.2, RPS4X, IGLC6
##     IGLV8-61, IGHA1, NT5C, IGLC3, ARHGAP4, RPS12, ZNF276, AC004556.1, IGHG4, MAD1L1
##     IGKV1D-13, IGLV4-60, GABARAP, IGHV2-70D, IGLV3-16, IGLV1-40, RPL26, DNAAF2, CHMP1A, IGLV1-44
## Negative:  TRIM58, MPIG6B, ITGA2B, LTBP1, F13A1, AC244226.1, LIMS1, THBS1, MMRN1, PROS1
##     PRKAR2B, TUBB1, IGLV3-25, STON2, STX11, IGHV1-3, MCTP1, ZNF185, IGHV1-69-2, PECAM1
##     RAB27B, VCL, ENDOD1, ITGB3, TAL1, IGKV1D-33, IGLV3-10, IGKV1-33, PTGS1, CLIC4
## PC_ 5
## Positive:  IGLV6-57, IGLV3-1, IGHG1, IGLV1-47, IGLV4-69, IGLV3-10, IGHV5-10-1, IGLV3-21, IGHV2-70, IG
##     IGHG2, IGLV3-25, IGLV2-23, IGKV2-24, IGLV2-8, IGHV6-1, IGHG3, IGKV2-30, IGLV3-19, IGLC3
##     IGHV3-72, IGKC, IGHV3-20, IGHA2, IGKV1-16, IGHV1-3, IGLV7-46, IGLV2-11, IGHV2-70D, IGKV1-17
## Negative:  IGHV1-2, IGKV1D-8, IGHV1-69-2, IGHV1-69, IGHV3-74, IGKV1-27, IGHV3-43, IGKV2-29, IGKV1D-17
##     IGHV3-13, IGHV4-59, IGHV3-53, IGHV3-66, IGHV4-31, IGKV2D-29, AC234301.1, AC145029.2, IGHM, IGHV3-
##     IGHV1-18, IGHV4-61, IGHV3-49, IGHV2-26, IGKV1D-33, IGKV3-20, IGKV1-6, IGKV1D-43, IGKV1-33, IGKV1-
```

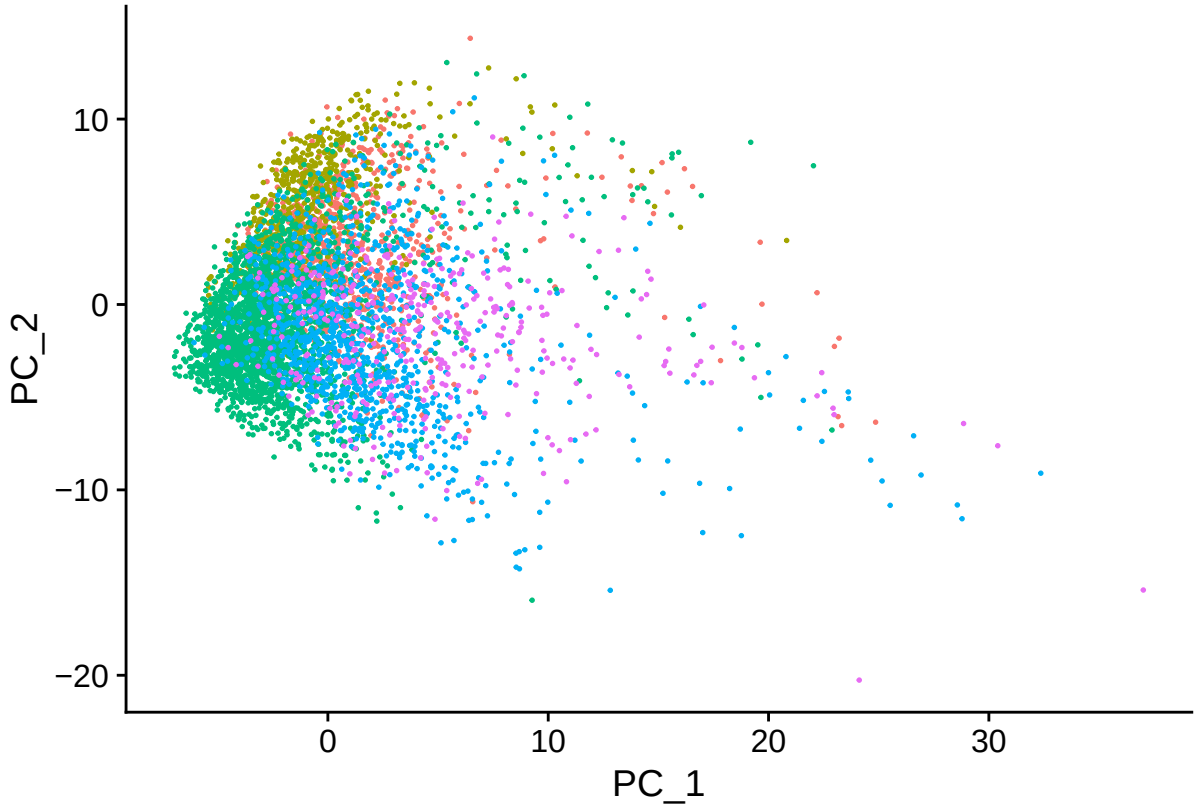
```r
# Visualise a few
print(object_filtered[["pca"]], dims = 1:5, nfeatures = 5)
```

```
## PC_ 1
## Positive:  IGHV1-69D, MRPL23, IGKV1-33, IGKV1-12, IGLV2-23
## Negative:  IGLC1, IGKV3D-15, IGKV2D-24, E2F7, ADAMTS1
## PC_ 2
## Positive:  IGKV3-15, IGKV3-20, IGHV5-51, IGHV3-33, IGKV1-39
## Negative:  IFI30, CALR, LRRC25, FTH1, STX11
## PC_ 3
## Positive:  IGHV3-43, IGHV3-13, GSE1, DDX39B, ZAP70
## Negative:  FTL, MT-CO2, MRPL23, RPL18A, RPL10
## PC_ 4
## Positive:  IKBKG, TESC, MRPL23, CCDC88B, ZAP70
## Negative:  TRIM58, MPIG6B, ITGA2B, LTBP1, F13A1
## PC_ 5
## Positive:  IGLV6-57, IGLV3-1, IGHG1, IGLV1-47, IGLV4-69
## Negative:  IGHV1-2, IGKV1D-8, IGHV1-69-2, IGHV1-69, IGHV3-74
```

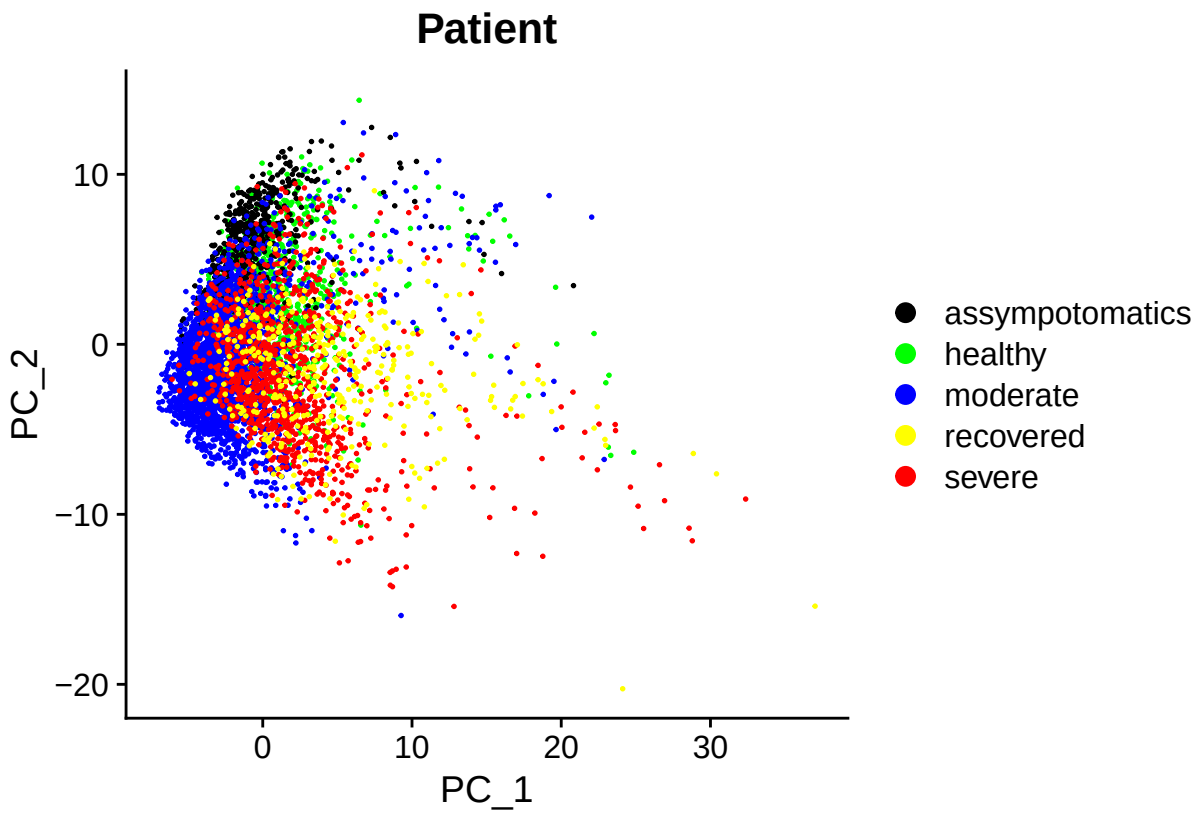
```
# Visualising the first two pcs
VizDimLoadings(object_filtered, dims = 1:2, reduction = "pca")
```




```
# Cells within the first two pcs
plot_pca_1 <- DimPlot(object_filtered, reduction = "pca" ) + NoLegend()
plot_pca_1
```

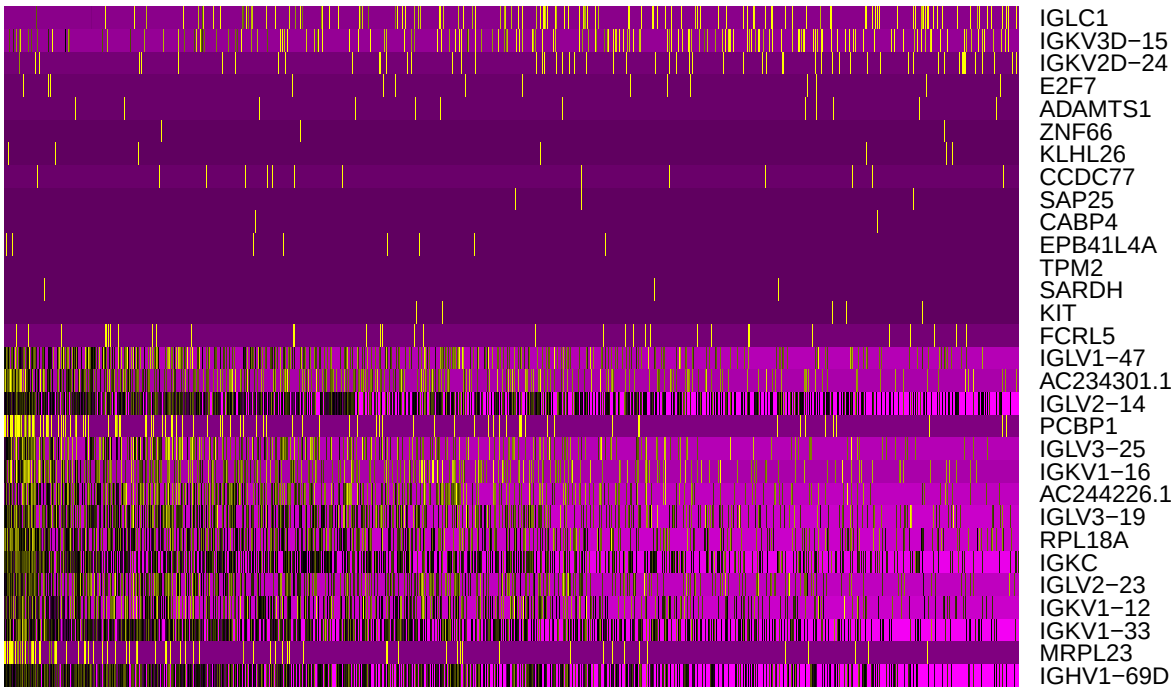



```
plot_pca_2 <- DimPlot(object_filtered, reduction = "pca",
                      group.by = "Patient",
                      cols = c("black", "green", "blue", "yellow", "red"))
plot_pca_2
```

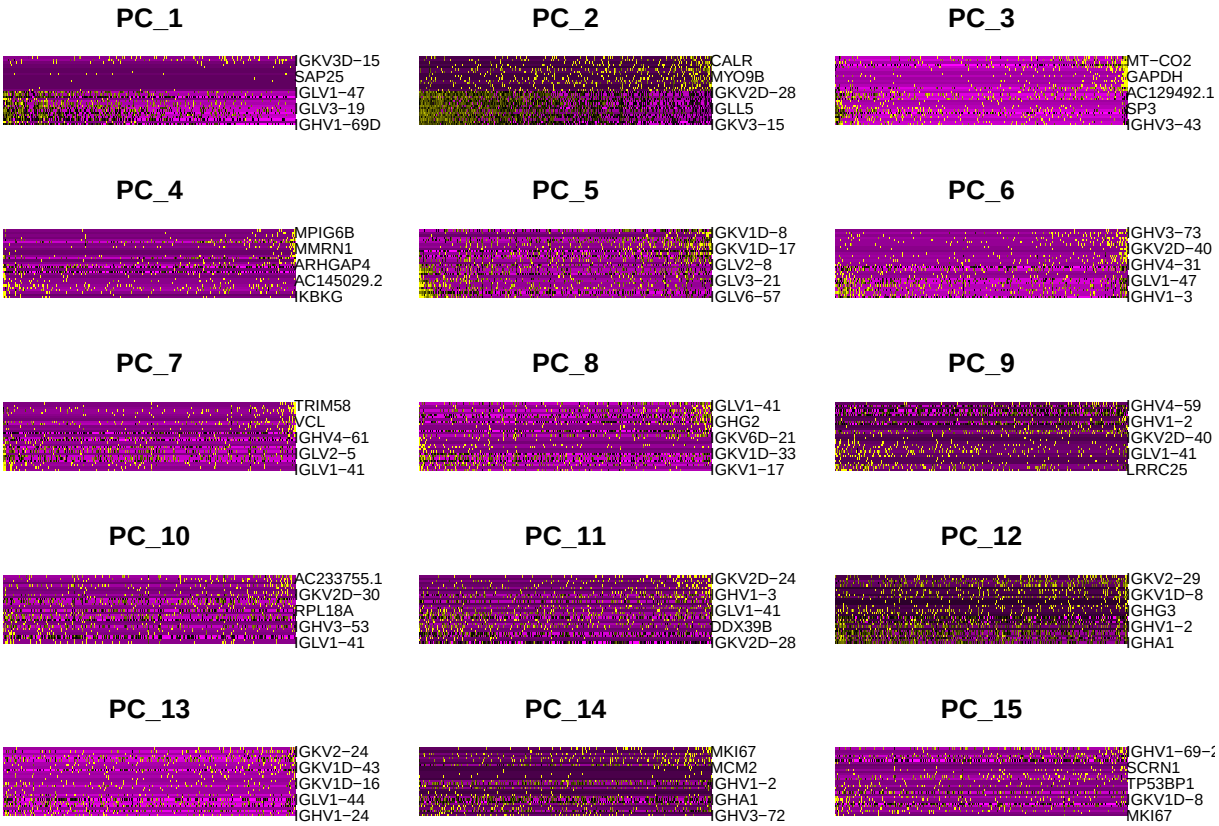


Plotting heatmaps Heatmaps enable easy exploration of the primary sources of heterogeneity in a dataset and are useful when deciding which PCs to include for downstream analysis.

```
DimHeatmap(object_filtered) #, dims = 1:15, cells = 500, balanced = TRUE)
```

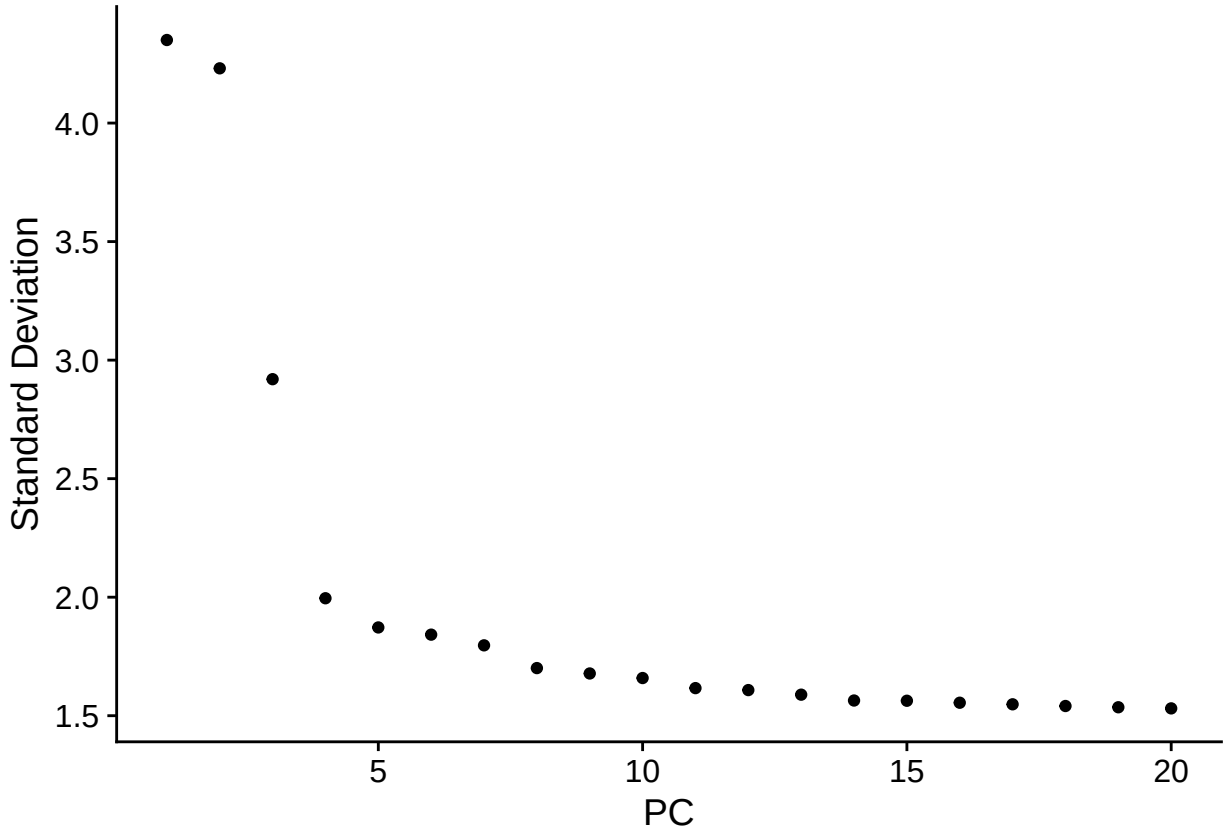



```
DimHeatmap(object_filtered, dims = 1:15) #, nfeatures = 10, ncol = 2)
```



Determining the dimensionality of the dataset The elbow plot helps rank the PCs based on the percentage of variance explained by each one

```
ElbowPlot(object_filtered)
```



Clustering the cells Seurat uses the graph based clustering approach for example the K-nearest neighbor (KNN) graph which draws edged on cells with similar features expressions and then connects these into communities. We first construct a KNN graph based on the euclidean distances in PCA space and then weight the edges based on the shared overlap in their local neighborhoods.

```
object_filtered <- FindNeighbors(object = object_filtered, dims = 1:15)
```

```
## Computing nearest neighbor graph
```

```
## Computing SNN
```

```
object_filtered <- FindClusters(object = object_filtered, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 5303
## Number of edges: 172629
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8286
## Number of communities: 13
## Elapsed time: 1 seconds
```

```
# Looking at the cluster IDs of the first 5 cells
head(Idents(object_filtered), 5)
```

```
## healthy_ATAGACCAGGGT healthy_CTAACTTAGGAG healthy_AGATCTGCAACC
##                    2                    1                    3
## healthy_ATGCGATAGTGT healthy_AGCGGTCTCAGG
```

```
##                   8                   1
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12
```

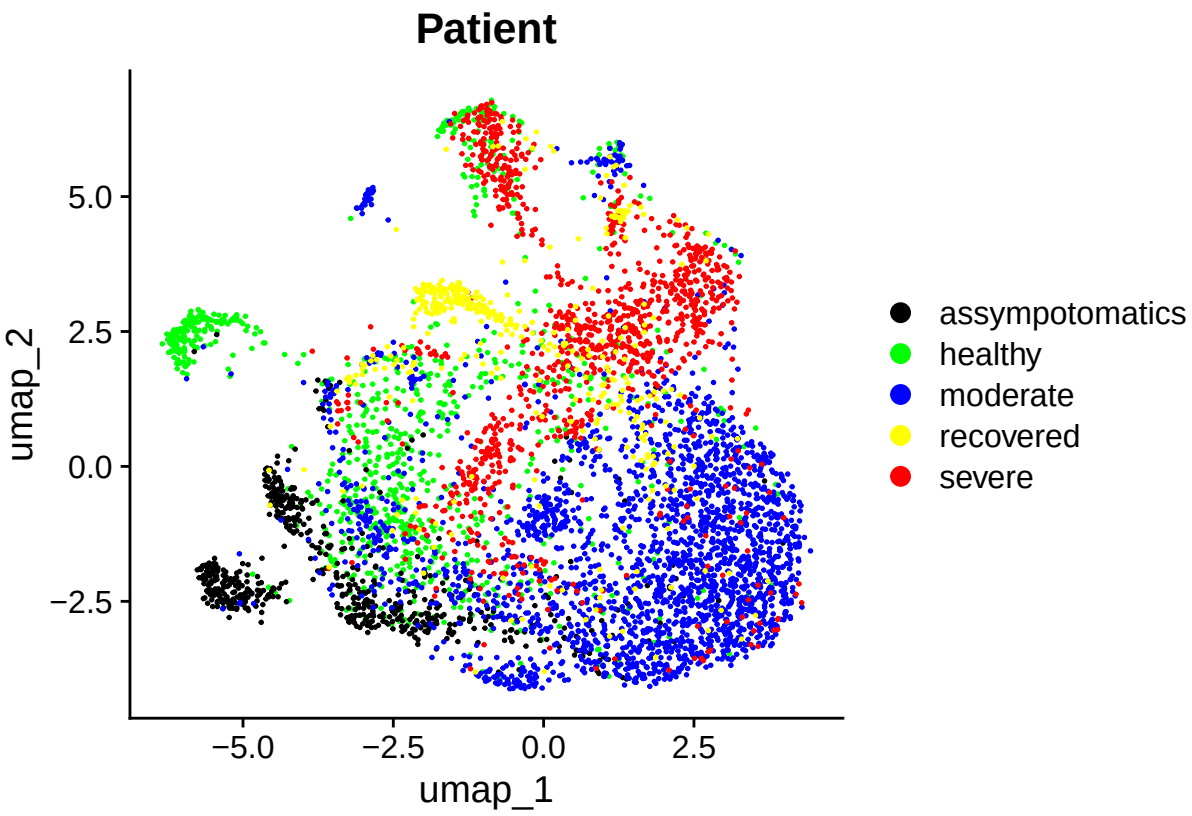
Running non-linear dimesional reduction (UMAP/tSNE)

These methods aim at preserving local distances in the dataset but often don't preserve more global relationships so we avoid drawing biological conclusions solely on the visualization techniques.

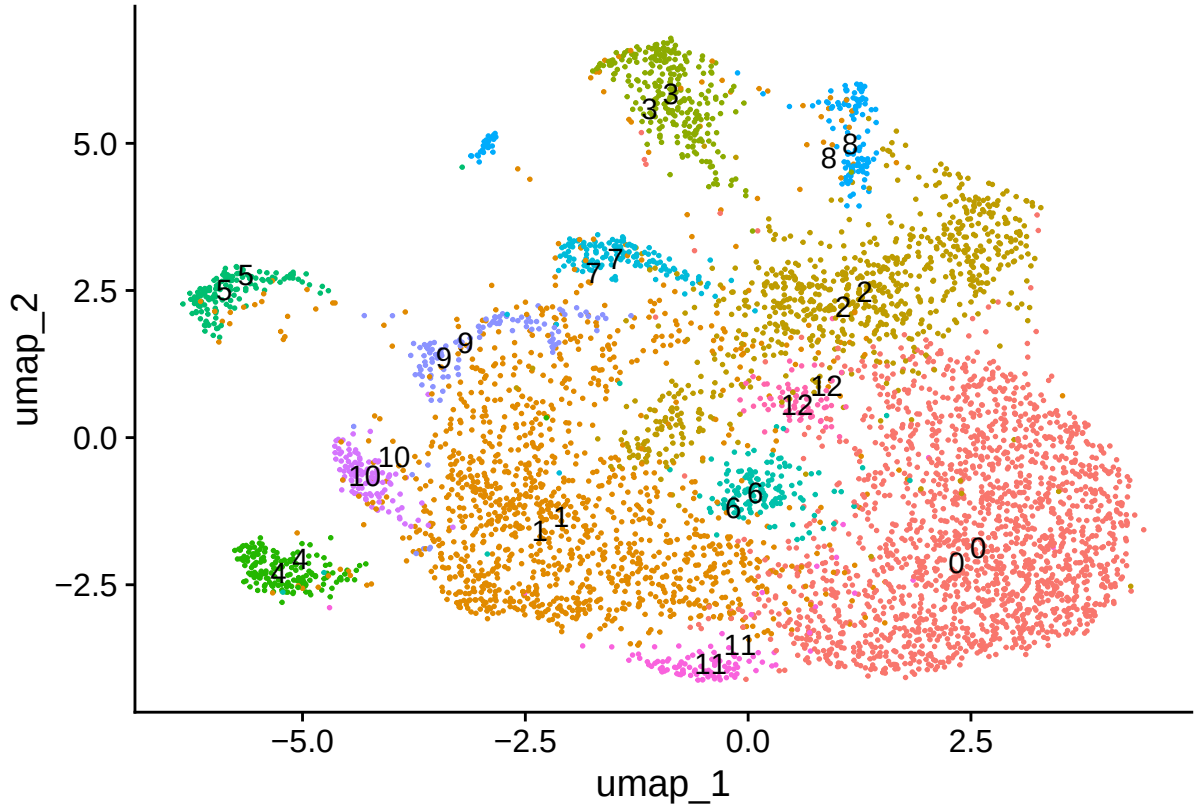
```
object_filtered <- RunUMAP(object = object_filtered, dims = 1:15)
```

```
## Warning: The default method for RunUMAP has changed from calling Python UMAP via reticulate to the R
## To use Python UMAP via reticulate, set umap.method to 'umap-learn' and metric to 'correlation'
## This message will be shown once per session
## 12:37:35 UMAP embedding parameters a = 0.9922 b = 1.112
## 12:37:35 Read 5303 rows and found 15 numeric columns
## 12:37:35 Using Annoy for neighbor search, n_neighbors = 30
## 12:37:35 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 12:37:37 Writing NN index file to temp file /tmp/Rtmp93TxMJ/file5b4a32615bd7
## 12:37:37 Searching Annoy index using 1 thread, search_k = 3000
## 12:37:40 Annoy recall = 100%
## 12:37:42 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 12:37:46 Initializing from normalized Laplacian + noise (using RSpectra)
## 12:37:46 Commencing optimization for 500 epochs, with 227990 positive edges
## 12:38:05 Optimization finished
```

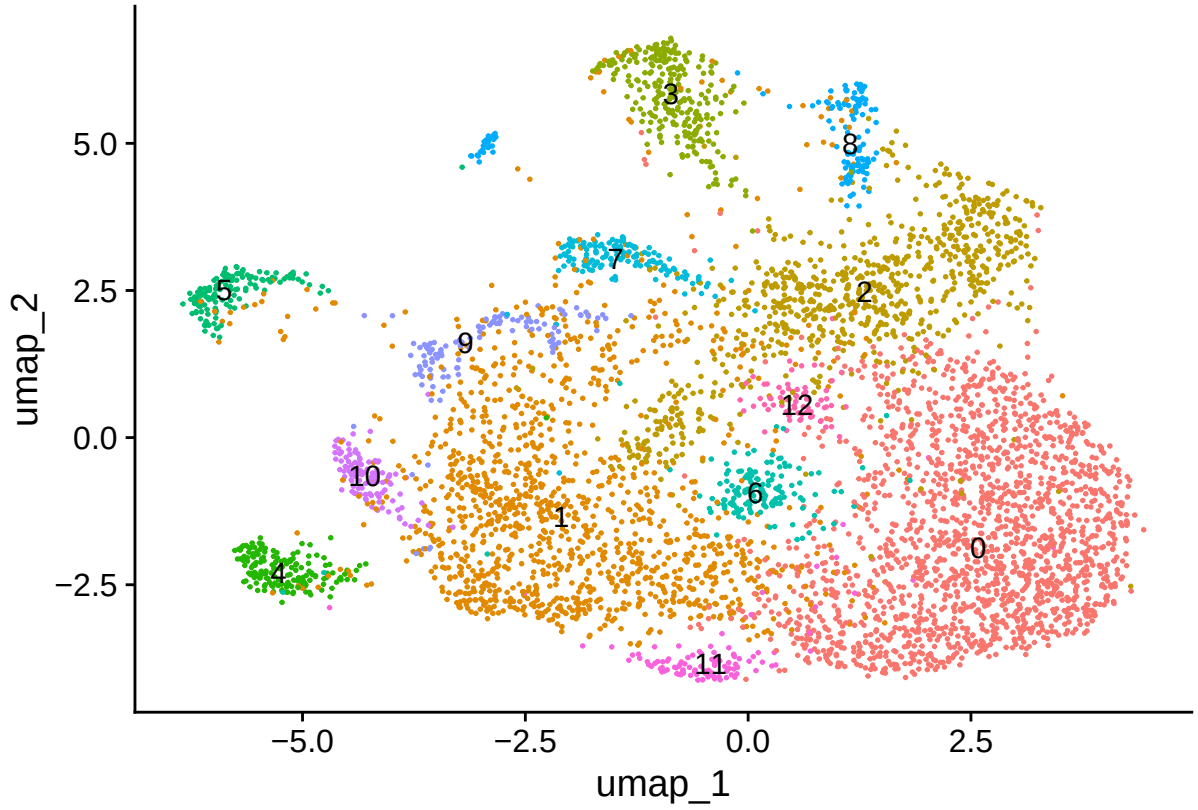
```
# Checking wheather cells cluster according to source
plot_patient <- DimPlot(object_filtered, reduction = "umap",
                        group.by = "Patient",
                        cols = c("black", "green", "blue", "yellow", "red"))
plot_patient
```




```
# Individual clusters
plot_umap <- DimPlot(object_filtered, reduction = "umap", label = TRUE) + NoLegend()
LabelClusters(plot_umap, id = 'ident')
```




```
plot_umap
```




```
# or DimPlot(object_filtered, reduction = "umap", label = T, repel = T) + NoLegend()
```

Annotating clusters using an automatic reference based method

```
# Annotating the clusters
ref <- celldex::HumanPrimaryCellAtlasData()
#View(as.data.frame(colData(ref)))

# Joining the layers
object_filtered <-  JoinLayers(object_filtered)

# Running singleR with default
object_counts <- GetAssayData(object_filtered, slot = "counts")
```

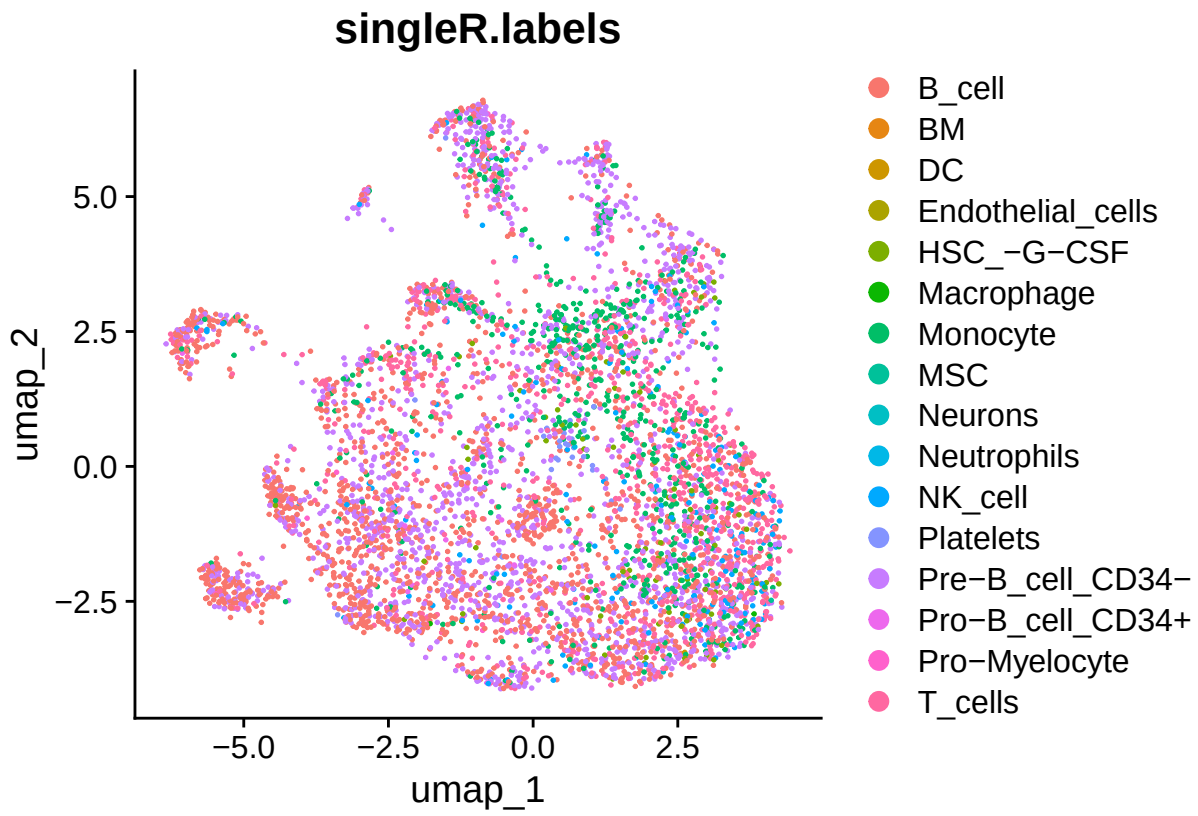
```
## Warning: The `slot` argument of `GetAssayData()` is deprecated as of SeuratObject 5.0.0.
## i Please use the `layer` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
predicted <- SingleR(test = object_counts, ref = ref, labels = ref$label.main)
predicted
```

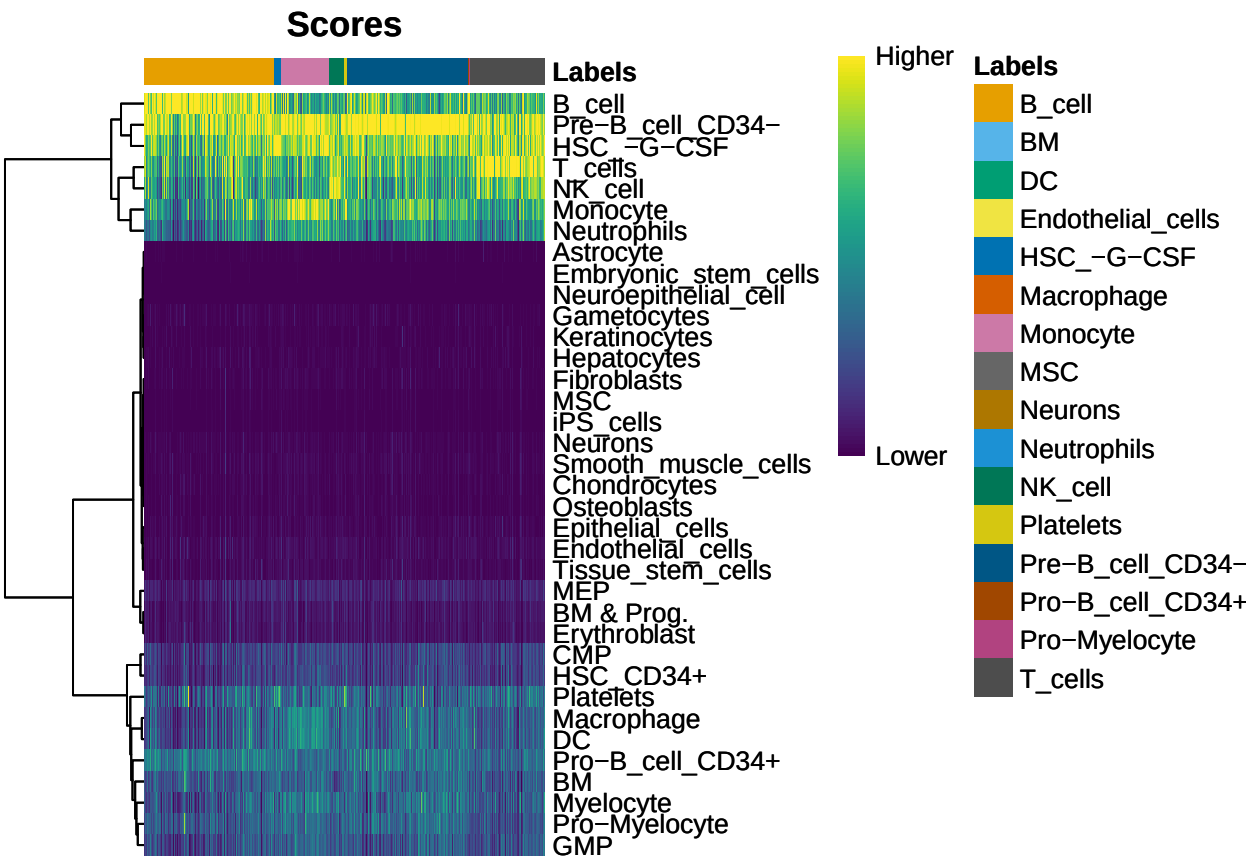
```
## DataFrame with 5303 rows and 4 columns
##                                                scores           labels
##                                              <matrix>      <character>
```

```
## healthy_ATAGACCAGGGT     0.0205400:0.125080:0.1037109:... Pre-B_cell_CD34-
## healthy_CTAACTTAGGAG     0.0441058:0.140348:0.1020277:...          T_cells
## healthy_AGATCTGCAACC     0.0191665:0.111898:0.0958721:... Pre-B_cell_CD34-
## healthy_ATGCGATAGTGT     0.0520049:0.154640:0.1243195:...           B_cell
## healthy_AGCGGTCTCAGG     0.0265472:0.129488:0.0970986:... Pre-B_cell_CD34-
## ...                                                  ...              ...
## recovered_CAGCTAATCTCG 0.01083635:0.1466898:0.1185828:...          T_cells
## recovered_ATAACGCCACAA 0.03818074:0.1253874:0.0697047:...           B_cell
## recovered_GACGGCTGTATT 0.00414312:0.0915006:0.0644146:...           B_cell
## recovered_TTGTAGGTCACC 0.02670675:0.0997304:0.0768512:...          T_cells
## recovered_AGACGTTCACAC 0.03760389:0.1225688:0.1007350:... Pro-B_cell_CD34+
##                        delta.next    pruned.labels
##                         <numeric>      <character>
## healthy_ATAGACCAGGGT    0.1080267 Pre-B_cell_CD34-
## healthy_CTAACTTAGGAG    0.0233073          T_cells
## healthy_AGATCTGCAACC    0.0683287 Pre-B_cell_CD34-
## healthy_ATGCGATAGTGT    0.0621443           B_cell
## healthy_AGCGGTCTCAGG    0.3147247 Pre-B_cell_CD34-
## ...                           ...              ...
## recovered_CAGCTAATCTCG 0.01039498          T_cells
## recovered_ATAACGCCACAA 0.07097879           B_cell
## recovered_GACGGCTGTATT 0.05530626           B_cell
## recovered_TTGTAGGTCACC 0.00393914          T_cells
## recovered_AGACGTTCACAC 0.10007657 Pro-B_cell_CD34+
```

```
object_filtered$singleR.labels <- predicted$labels[match(rownames(object_filtered@meta.data),
                                                         rownames(predicted))]
DimPlot(object_filtered, reduction = "umap", group.by = "singleR.labels")
```




```
# Annotation diagnostics
plotScoreHeatmap(predicted)
```




```
# Based on the deltas
plotDeltaDistribution(predicted)
```

```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

```
## Warning in max(data$density, na.rm = TRUE): no non-missing arguments to max;
## returning -Inf
```

```
## Warning: Computation failed in `stat_ydensity()`.
## Caused by error in `$<-.data.frame`:
## ! replacement has 1 row, data has 0
```

```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

```
## Warning in max(data$density, na.rm = TRUE): no non-missing arguments to max;
## returning -Inf
```

```
## Warning: Computation failed in `stat_ydensity()`.
## Caused by error in `$<-.data.frame`:
## ! replacement has 1 row, data has 0
```

```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

```
## Warning in max(data$density, na.rm = TRUE): no non-missing arguments to max;
## returning -Inf
```

```
## Warning: Computation failed in `stat_ydensity()`.
```

```
## Caused by error in `$<-.data.frame`:
## ! replacement has 1 row, data has 0
```

```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

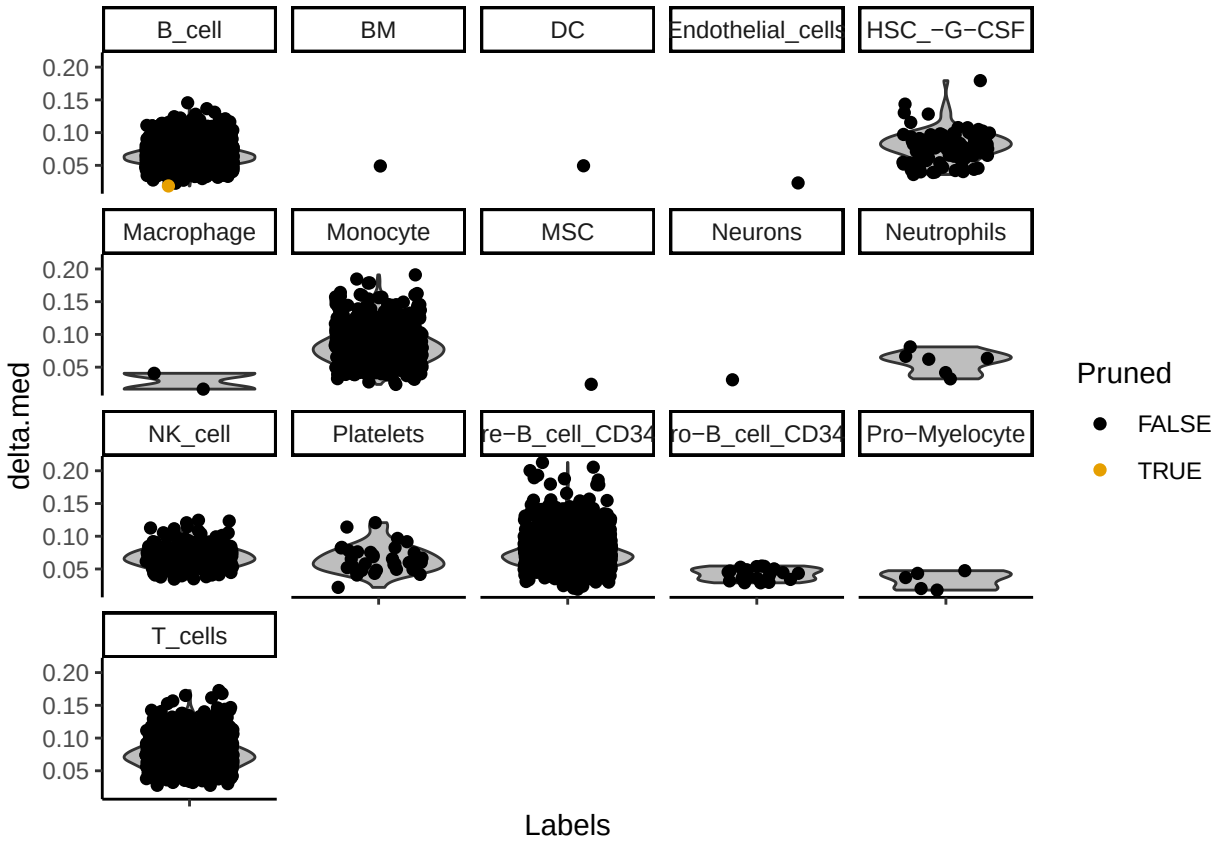
```
## Warning in max(data$density, na.rm = TRUE): no non-missing arguments to max;
## returning -Inf
```

```
## Warning: Computation failed in `stat_ydensity()`.
## Caused by error in `$<-.data.frame`:
## ! replacement has 1 row, data has 0
```

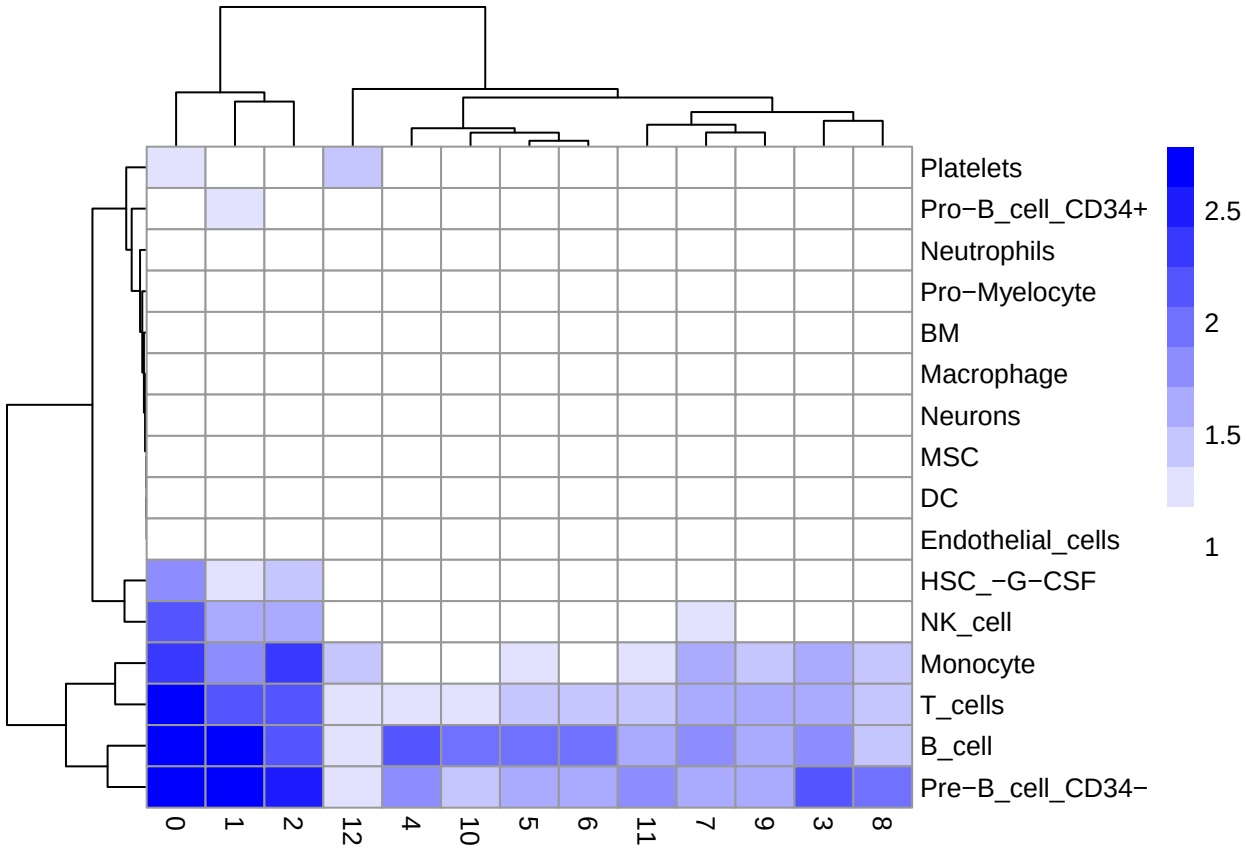
```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

```
## Warning in max(data$density, na.rm = TRUE): no non-missing arguments to max;
## returning -Inf
```

```
## Warning: Computation failed in `stat_ydensity()`.
## Caused by error in `$<-.data.frame`:
## ! replacement has 1 row, data has 0
```



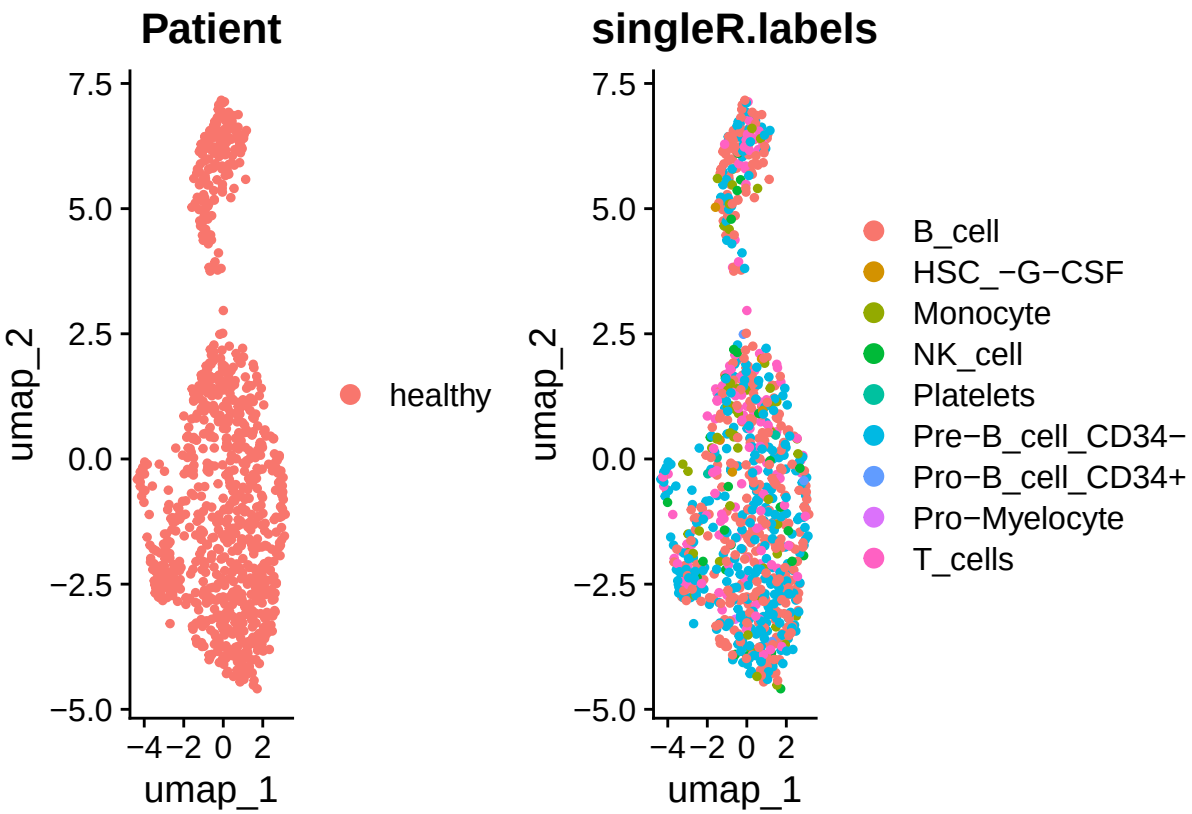

```
# Comparing to unsupervised clustering
tab <- table(Assigned=predicted$labels, clusters=object_filtered$seurat_clusters)
pheatmap(log10(tab+10), color = colorRampPalette(c("white", "blue"))(10))
```




```
# Creating subsets for
healthy_subset <- subset(object_filtered, Patient == "healthy")
healthy_umap <- RunUMAP(healthy_subset, dims = 1:15)
```

```
## 12:47:14 UMAP embedding parameters a = 0.9922 b = 1.112
## 12:47:14 Read 909 rows and found 15 numeric columns
## 12:47:14 Using Annoy for neighbor search, n_neighbors = 30
## 12:47:14 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 12:47:14 Writing NN index file to temp file /tmp/Rtmp93TxMJ/file5b4a3fe7dd4e
## 12:47:14 Searching Annoy index using 1 thread, search_k = 3000
## 12:47:15 Annoy recall = 100%
## 12:47:16 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 12:47:20 Initializing from normalized Laplacian + noise (using RSpectra)
## 12:47:20 Commencing optimization for 500 epochs, with 38010 positive edges
## 12:47:24 Optimization finished
```

```
DimPlot(healthy_umap, group.by = c("Patient", "singleR.labels"))
```



Finding differentially expressed features

By default FindAllMarkers() identifies positive and negative markers of a single cluster (ident.1)

```
# find all markers of cluster 1
cluster.markers <- FindAllMarkers(object_filtered)
```

```
## 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
```

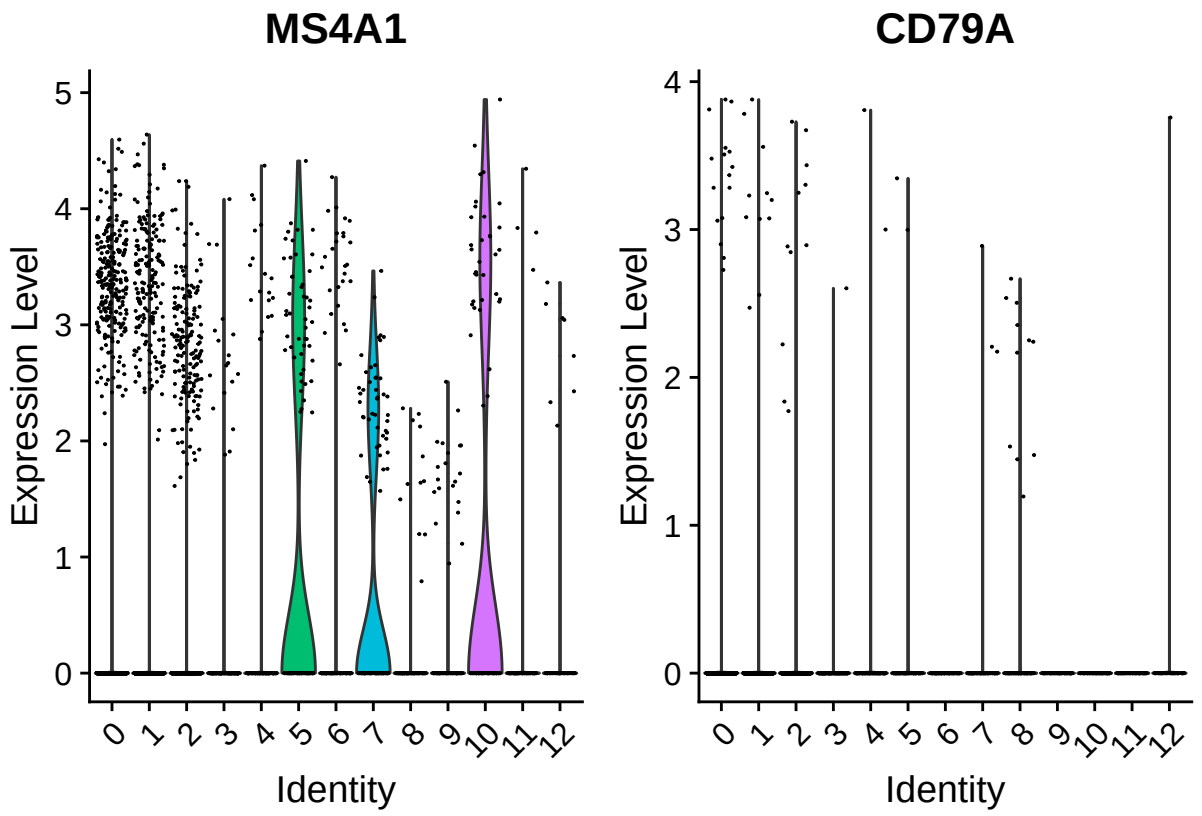
```
head(cluster.markers, n = 5)
```

```
##                   p_val avg_log2FC pct.1 pct.2     p_val_adj cluster      gene
## IGHV1-69D 2.976114e-247  -2.162915 0.301 0.760 1.069615e-242       0 IGHV1-69D
## IGKV1-33  2.330098e-241  -2.320460 0.266 0.743 8.374373e-237       0  IGKV1-33
## IGKV3-15  1.446565e-240  -1.690562 0.523 0.886 5.198953e-236       0  IGKV3-15
## IGKV3-11  6.782437e-235  -1.635834 0.515 0.876 2.437608e-230       0  IGKV3-11
## IGKV1-5   9.350101e-235  -1.680317 0.416 0.850 3.360426e-230       0   IGKV1-5
```

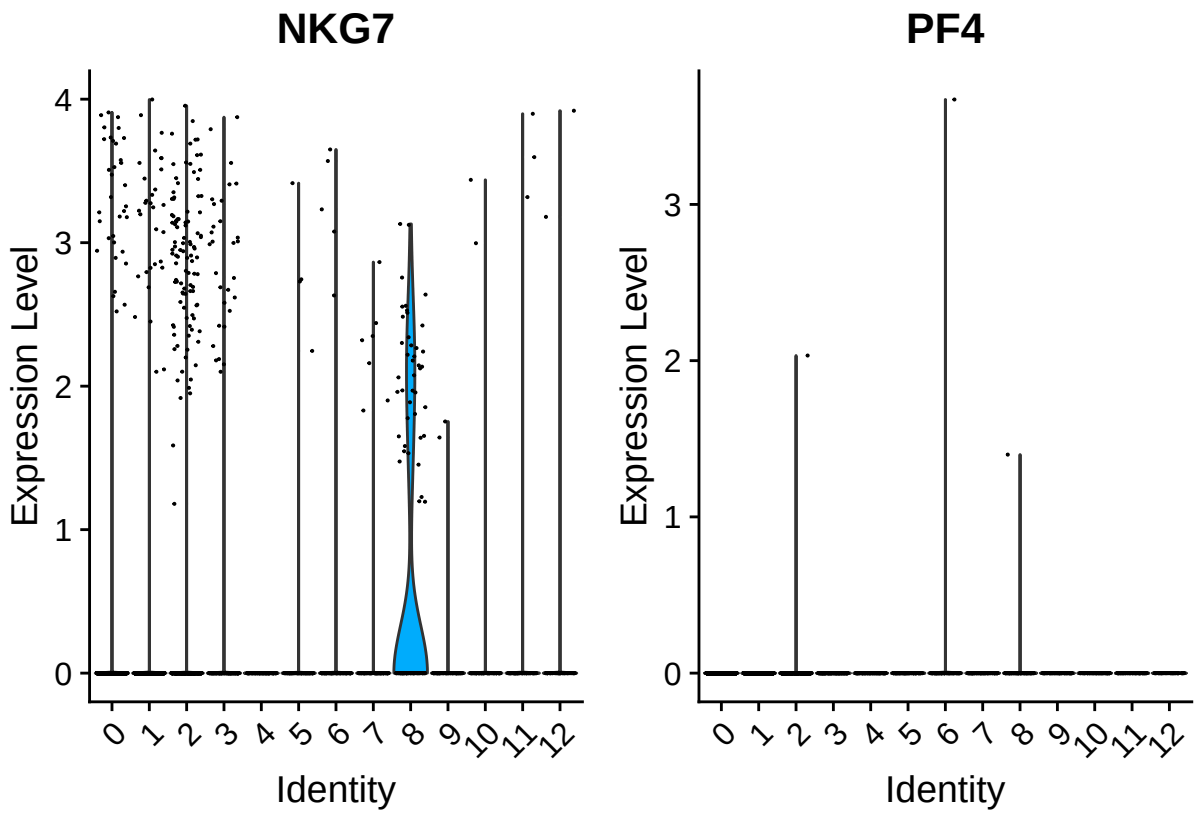
```
cluster.markers %>% group_by(cluster) %>% filter(avg_log2FC > 1)
```

```
## # A tibble: 6,152 x 7
## # Groups:   cluster [13]
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 7.05e-  3       1.54 0.013 0.006 1   e+  0 0       UBE2H
##  2 9.80e-  3       1.34 0.028 0.017 1   e+  0 0       THEM4
##  3 5.19e-141       1.01 0.966 0.802 1.87e-136 1       IGKV3-20
##  4 4.13e-115       1.04 0.905 0.703 1.49e-110 1       IGKV3-11
##  5 2.02e-102       1.15 0.782 0.514 7.24e- 98 1       IGKV1-33
##  6 2.23e- 96       1.18 0.694 0.418 8.03e- 92 1       IGHV3-15
##  7 8.02e- 92       1.14 0.749 0.496 2.88e- 87 1       IGKV1D-33
##  8 9.60e- 90       1.01 0.921 0.71  3.45e- 85 1       IGKV3-15
##  9 9.73e- 70       1.16 0.854 0.673 3.50e- 65 1       IGKV2D-28
## 10 9.96e- 61       1.24 0.516 0.295 3.58e- 56 1       IGHV3-53
## # i 6,142 more rows
```

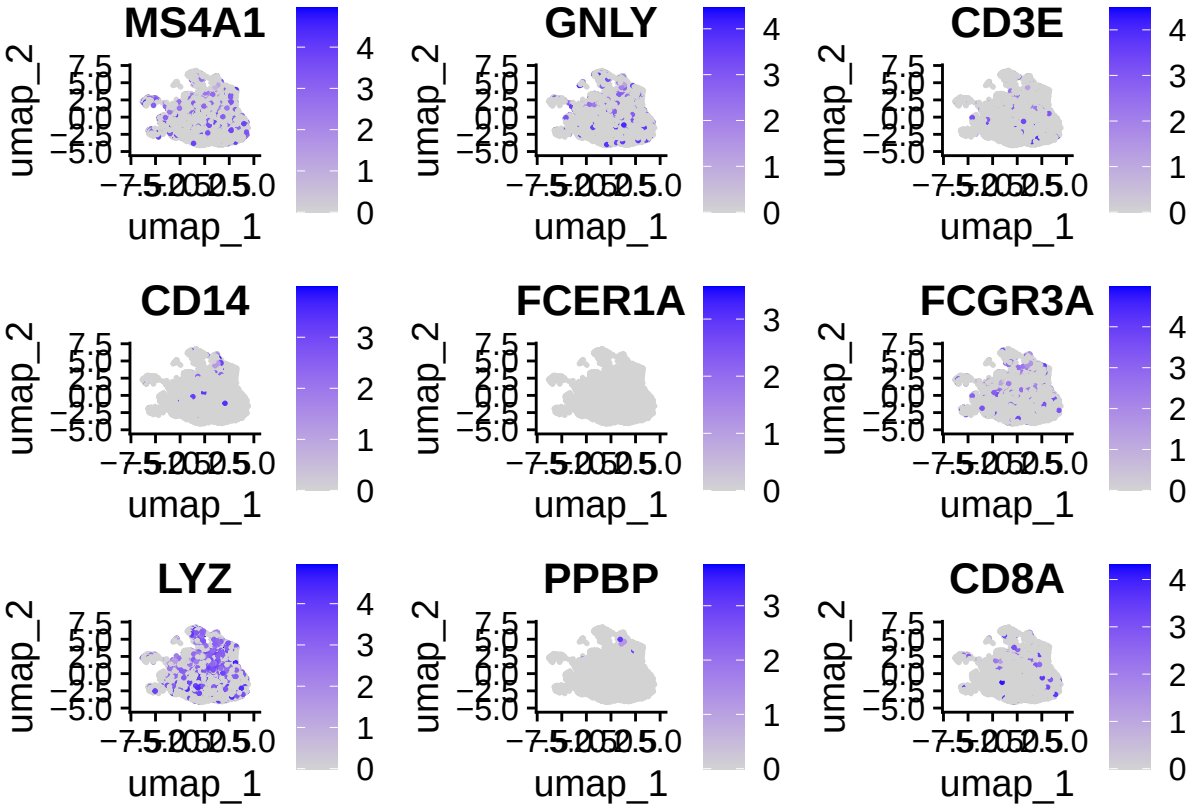
```
# Markers for b cells
VlnPlot(object_filtered, features = c("MS4A1", "CD79A"))
```




```
# Markers for raw counts
VlnPlot(object_filtered, features = c("NKG7", "PF4"), log = FALSE)
```

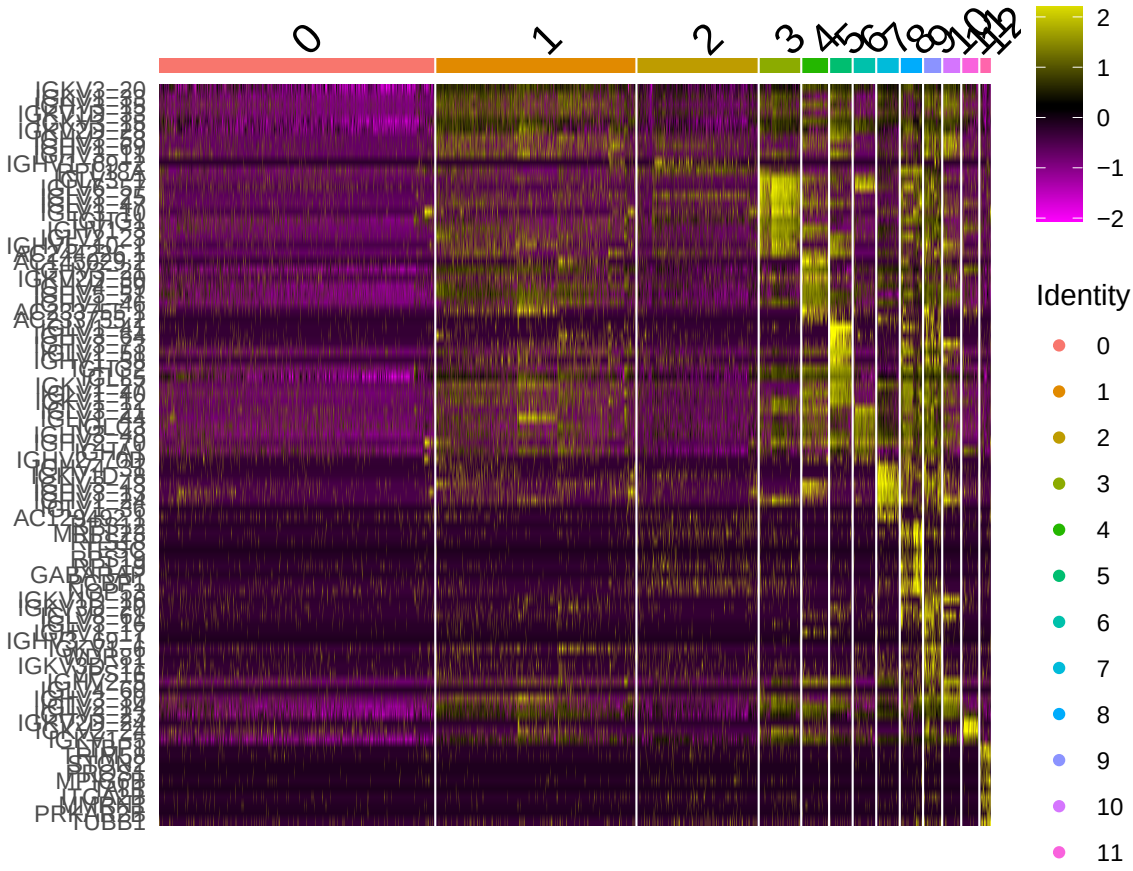



```
FeaturePlot(object_filtered, features = c("MS4A1", "GNLY", "CD3E", "CD14", "FCER1A", "FCGR3A", "LYZ", "
```




```
cluster.markers %>% group_by(cluster) %>% filter(avg_log2FC > 1) %>%
  slice_head(n = 10) %>% ungroup() -> Top10
DoHeatmap(object_filtered, features = Top10$gene)
```

```
## Warning in DoHeatmap(object_filtered, features = Top10$gene): The following
## features were omitted as they were not found in the scale.data slot for the RNA
## assay: RBM20, EPB41L1, CCDC188, ADGRG3, RPL21P16, IGKV2OR2-7D, AC009286.3,
## AC234301.4, IGHV7-81, RPS9, JSRP1, ATP5F1E, SH3D19, CR759762.1, BEST1, RPL11,
## CR388220.1, IGKV3-11, THEM4, UBE2H
```


0
1
2
3
4
5
6
7
8
9
10
11
2
1
0
−1
−2
Identity
0
1
2
3
4
5
6
7
8
9
10
11