RNA_SequenceAnalysis

Antonio Pano

2022-11-22

https://www.10xgenomics.com/resources/datasets/20-k-human-pbm-cs-5-ht-v-2-0-2-high-6-1-0 (https://www.10xgenomics.com/resources/datasets/20-k-human-pbm-cs-5-ht-v-2-0-2-high-6-1-0)

Human peripheral blood mononuclear cells (PBMCs) of a healthy male donor aged 30-35 were obtained by 10x Genomics from AllCells.

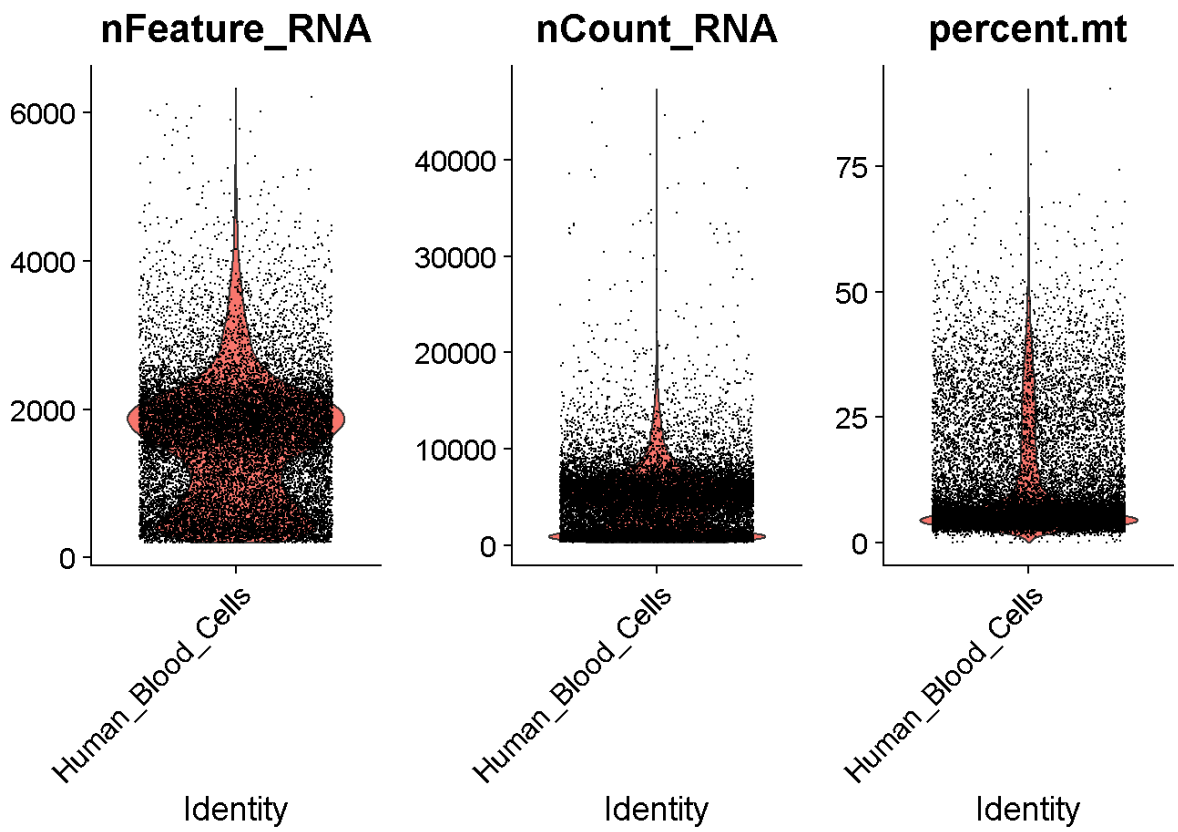
Gene expression and V(D)J libraries were generated from ~33,000 cells (18,470 cells recovered) as described in the Chromium Next GEM Single Cell 5' HT Reagent Kits v2 User Guide (CG000421) using the Chromium and sequenced on an Illumina NovaSeq 6000 to a read depth of approximately 25,000 mean reads per cell for Gene Expression, 15,000 mean reads per cell for TCR Amplified, and 25,000 mean reads per cell for BCR Amplified libraries.

```
library(tidyr)
library(dplyr)
library(Seurat)
library(ggplot2)
```

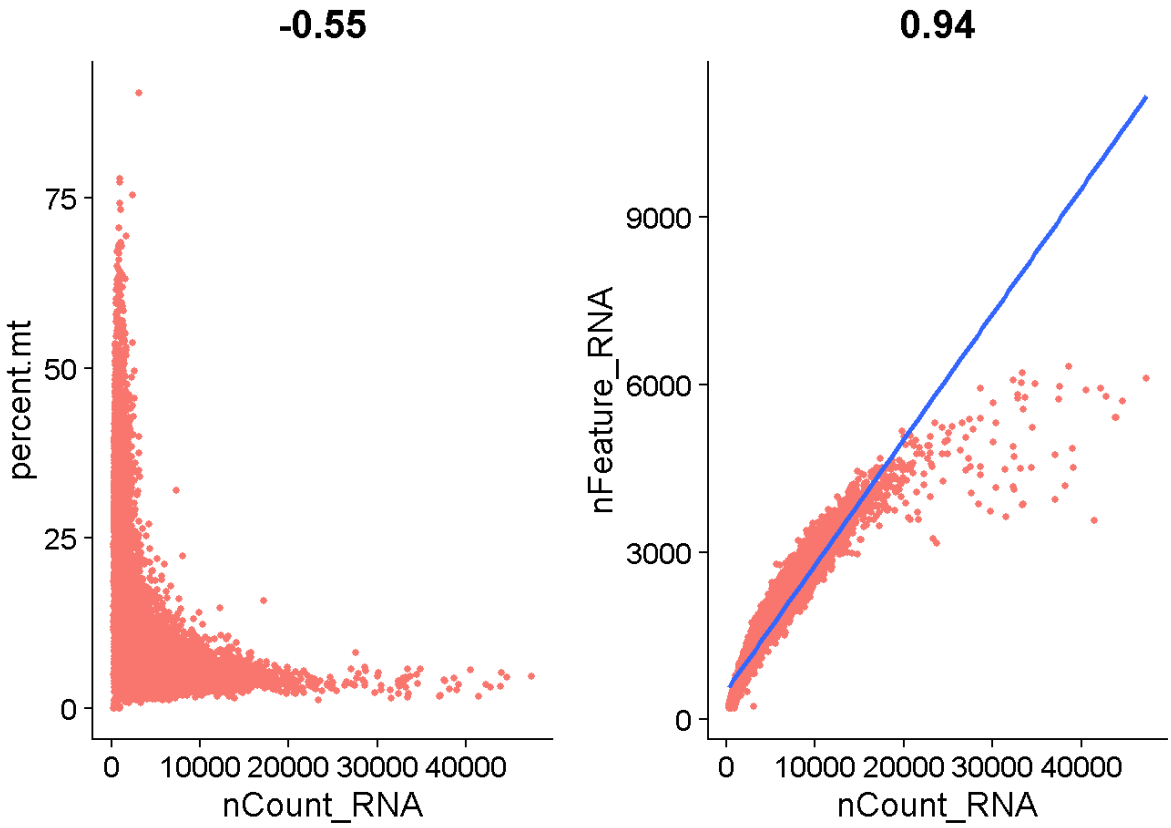
```
#loading matrix
Human <- Read10X_h5(filename = "20k_PBMC_5pv2_HT_nextgem_Chromium_X_Multiplex_count_raw_feature_bc_matrix.h
5")


#21517 features across 19574 samples within 1 assay
#Active assay: RNA (21517 features, 0 variable features)
human_obj <- CreateSeuratObject(counts = Human, project = "Human_Blood_Cells",
                   min.cells = 3, min.features = 200)
```

Quality Control

```
human_obj[["percent.mt"]] <- PercentageFeatureSet(human_obj, pattern = "^MT")

human_obj@meta.data %>% View

VlnPlot(human_obj, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```




```
plot1 <- FeatureScatter(human_obj, feature1 = "nCount_RNA", feature2 = "percent.mt") + theme(legend.position
= "none")
plot2 <- FeatureScatter(human_obj, feature1 = "nCount_RNA", feature2 = "nFeature_RNA") + geom_smooth(method
 = "lm")+ theme(legend.position = "none")
plot1 + plot2
```



Filtering

```
human_obj <- subset(human_obj, subset = nFeature_RNA > 150 & nFeature_RNA < 4500 & percent.mt <= 25)
```

Normalization

```
human_obj <- NormalizeData(human_obj, normalization.method = "LogNormalize", scale.factor = 1000)
```

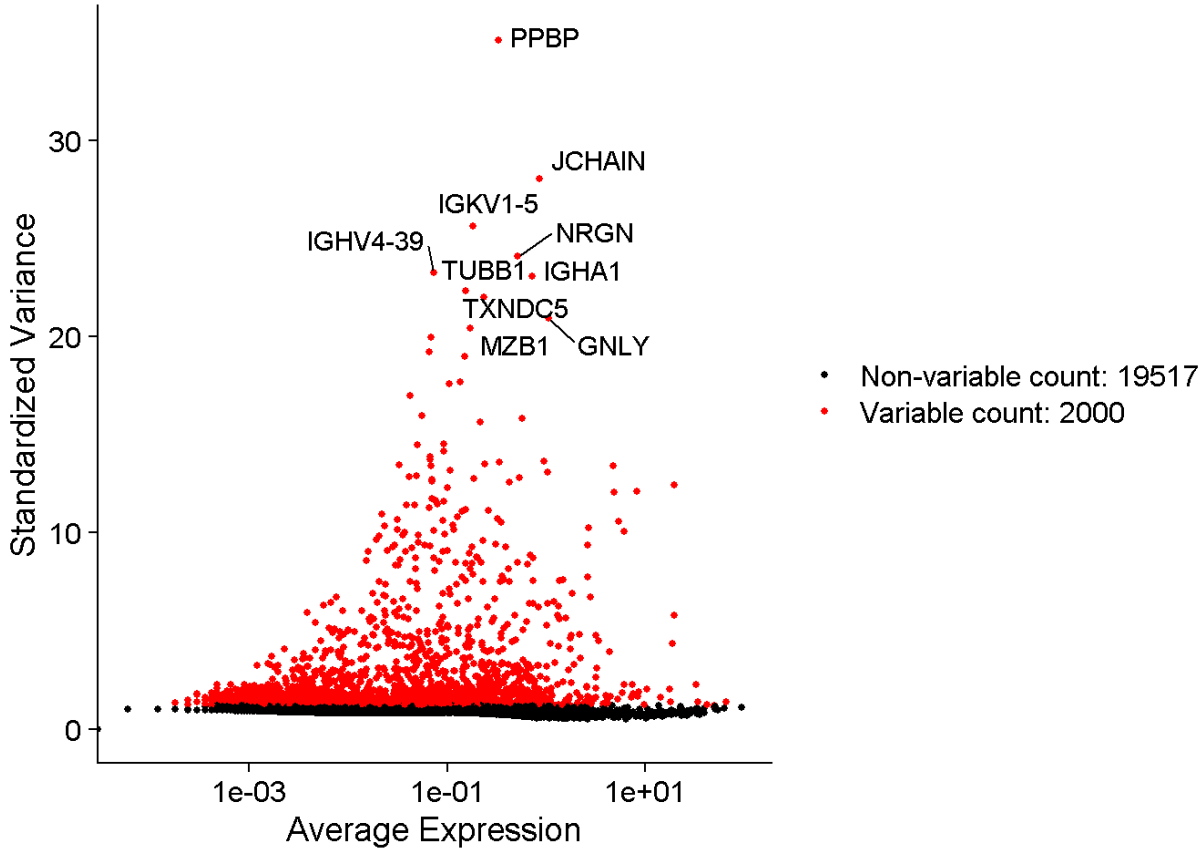
Identifying Highly Variable Features

Finding features which exhibit high cell-to-cell variation in the dataset (i.e, they are highly expressed in some cells, and lowly expressed in others)

```
human_obj <- FindVariableFeatures(human_obj, selection.method = "vst", nfeatures = 2000)
```

Top 10 most highly variable genes

```
top10 <- head(VariableFeatures(human_obj), 10)

plot1 <- VariableFeaturePlot(human_obj)
LabelPoints(plot = plot1, points = top10, repel = TRUE)
```



Scaling

- Scaling is done to make sure that cell clustering occurs because of natural causes and not axis “weight” similar to KNN.
- ScaleData() Function:
- Shifts the expression of each gene, so that the mean expression across cells is 0
- Scales the expression of each gene, so that the variance across cells is 1 (This step gives equal weight in downstream analyses, so that highly-expressed genes do not dominate)
- The results of this are stored in pbmc[[“RNA”]]@scale.data

```
all_genes <- rownames(human_obj)
human_obj <- ScaleData(human_obj, features = all_genes)
```

Dimensionality Reduction (Principal Component Analysis)

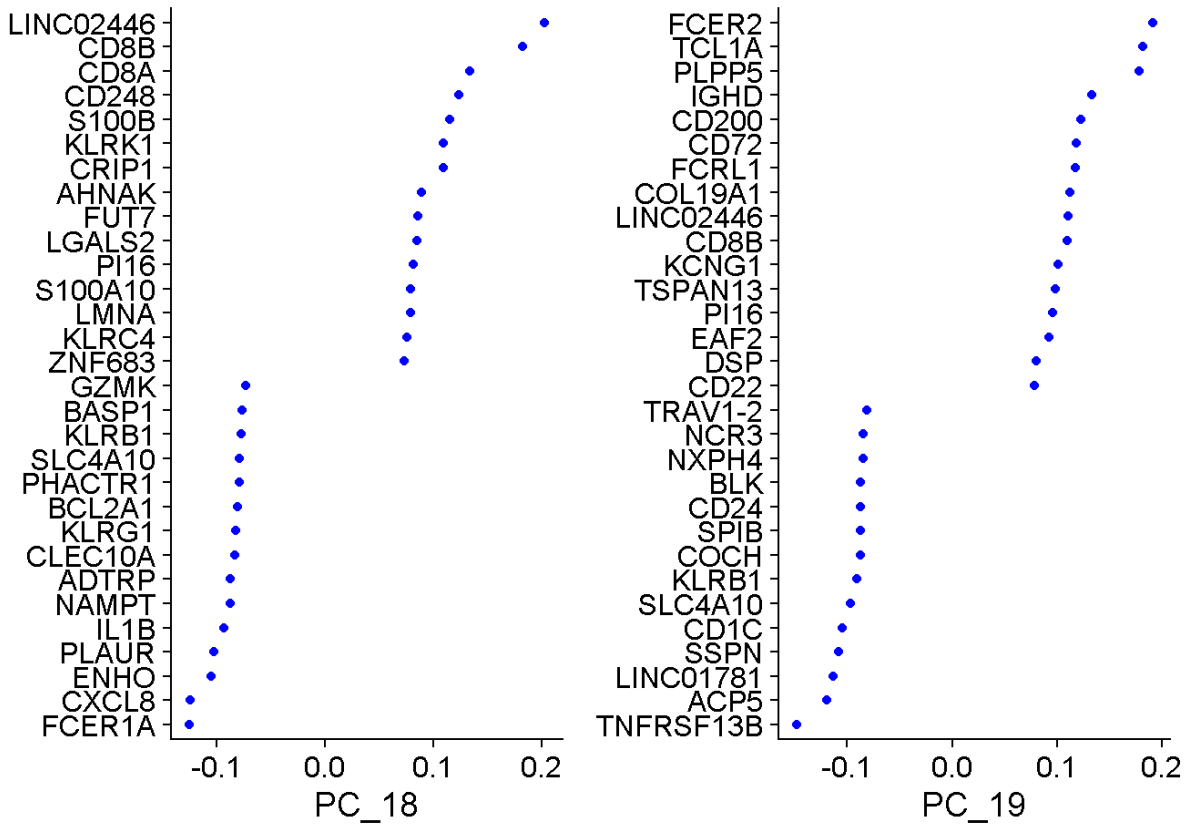
- RunPCA will throw an error if the data hasn’t been scaled.

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

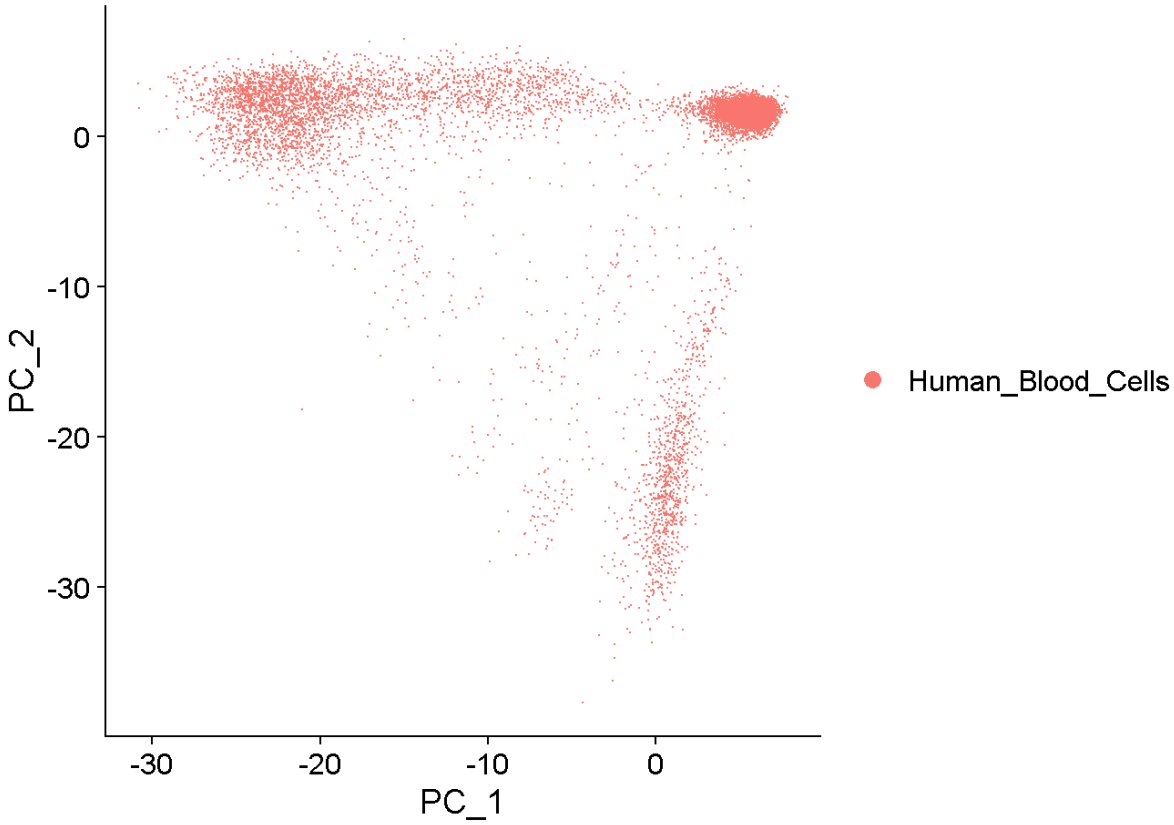
```
##### There are different ways of visualizing PCA:

### Through text
print(human_obj[["pca"]], dims = 1:5, nfeatures = TRUE)
```

```
## PC_ 1
## Positive:  IL32
## Negative:  CST3
## PC_ 2
## Positive:  IL32
## Negative:  CD79A
## PC_ 3
## Positive:  PPBP
## Negative:  CD37
## PC_ 4
## Positive:  ATP2B1-AS1
## Negative:  PFN1
## PC_ 5
## Positive:  NKG7
## Negative:  RPLP1
```

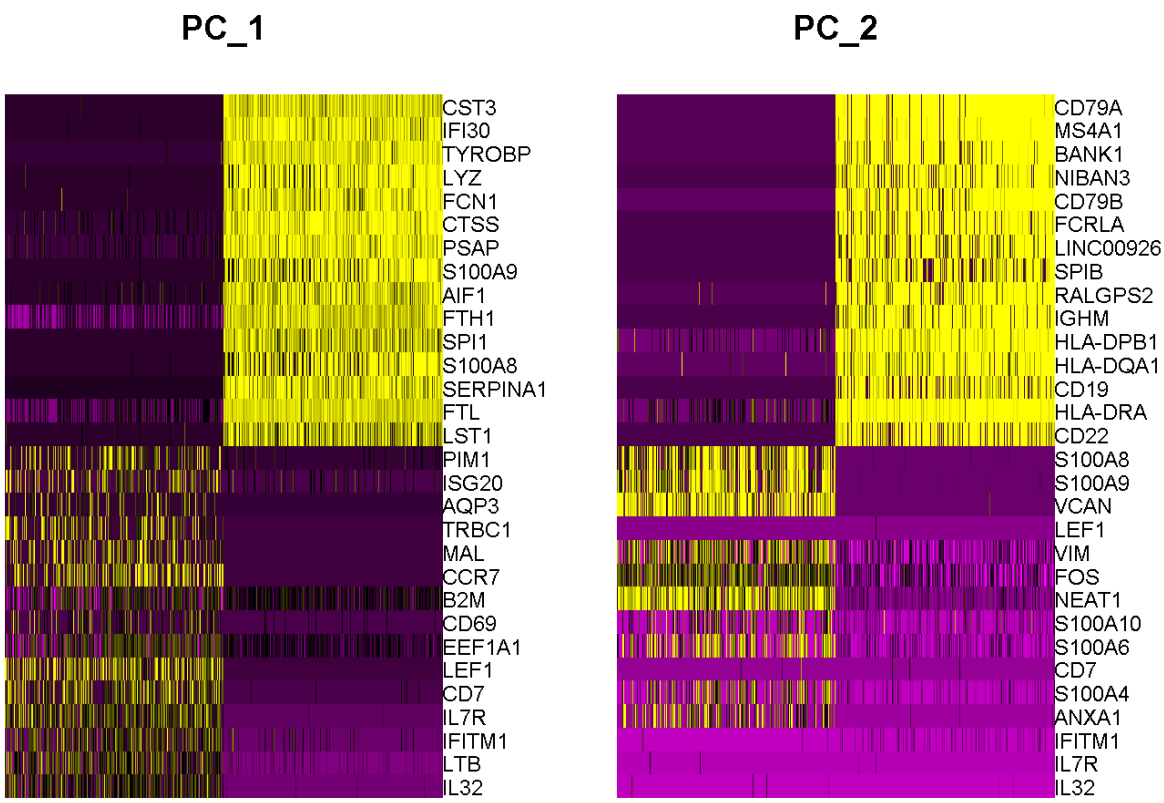
```
### Through Dimension Loadings
VizDimLoadings(human_obj, dims = 18:19, reduction = "pca")
```



```
### Through a Dimension Plot
DimPlot(human_obj, reduction = "pca")
```




```
### Through a Heatmap
# dims = "How many Principal Components?"
# cells = num of cells to plot
Seurat::DimHeatmap(human_obj, dims = 1:2, cells = 500, balanced = TRUE)
```



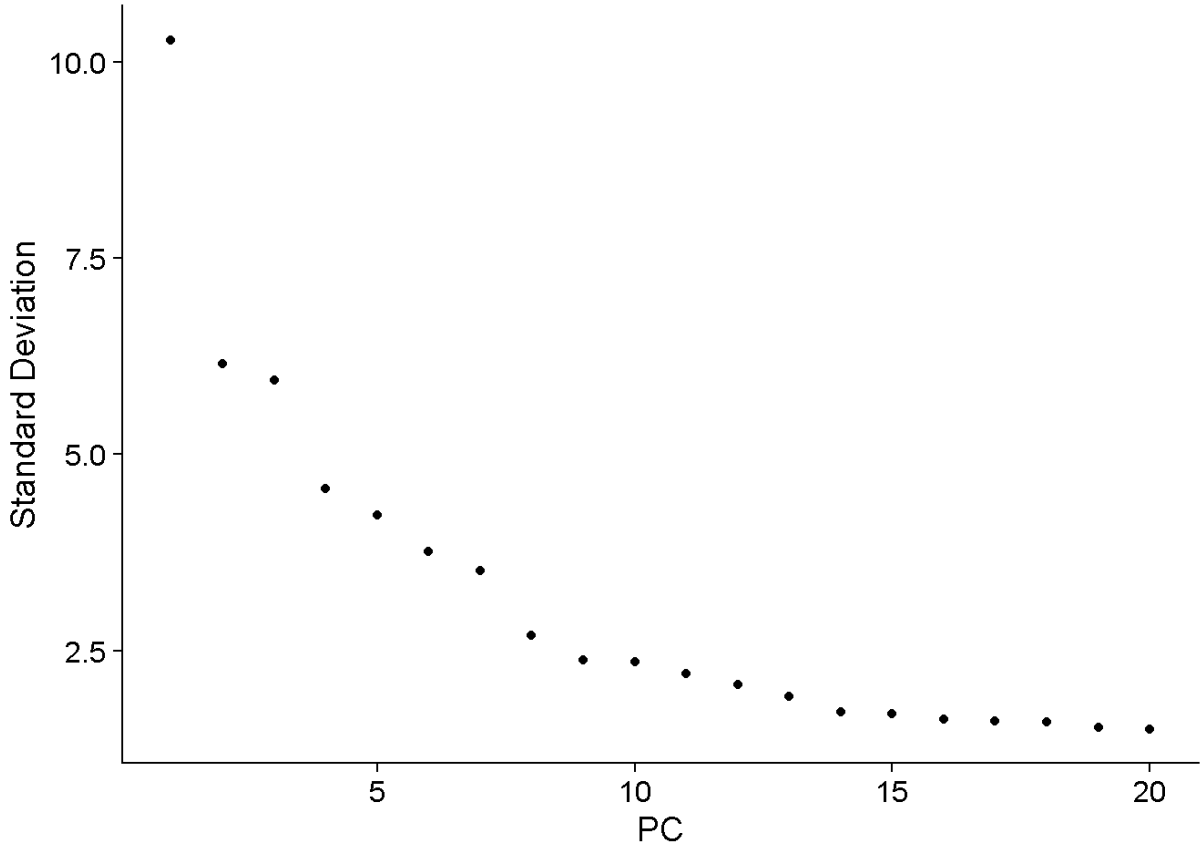
Determining Dimensionality (How many PCs should we use?)

Determining dimensionality to use PCs that capture majority of the signal in this downstream analysis.

Elbow Plot: A ranking of principle components based on the percentage of variance explained by each one . Better to use elbow on the highr side of the x-axis because the effects could be large, otherwise (huge loss of signal from one PC to the next).

Jack Straw Plot: Provides a visualization tool for comparing the distribution of p-values for each PC with a uniform distribution (dashed line). ‘Significant’ PCs will show a strong enrichment of features with low p-values (solid curve above the dashed line). You choose PCs based off those that don’t “drop off”.

```
ElbowPlot(human_obj)
```




```
#human_obj <- JackStraw(human_obj, num.replicate = 100)
#human_obj <- ScoreJackStraw(human_obj, dims = 1:20)
```

Clustering the cells

- First, constructing a KNN graph based on the euclidean distance in PCA space, and refine the edge weights between any two cells based on the shared overlap in their local neighborhoods (Jaccard similarity). This step is performed using the FindNeighbors() function. Takes as input the previously defined dimensionality of the dataset.
- Second, we apply Modularity Optimization techniques to iteratively group cells together, with the goal of optimizing the standard modularity function. The FindClusters() function implements this procedure, and contains a resolution parameter that sets the 'granularity' of the downstream clustering, with increased values leading to a greater number of clusters.
- The clusters can be found using the Idents() function.

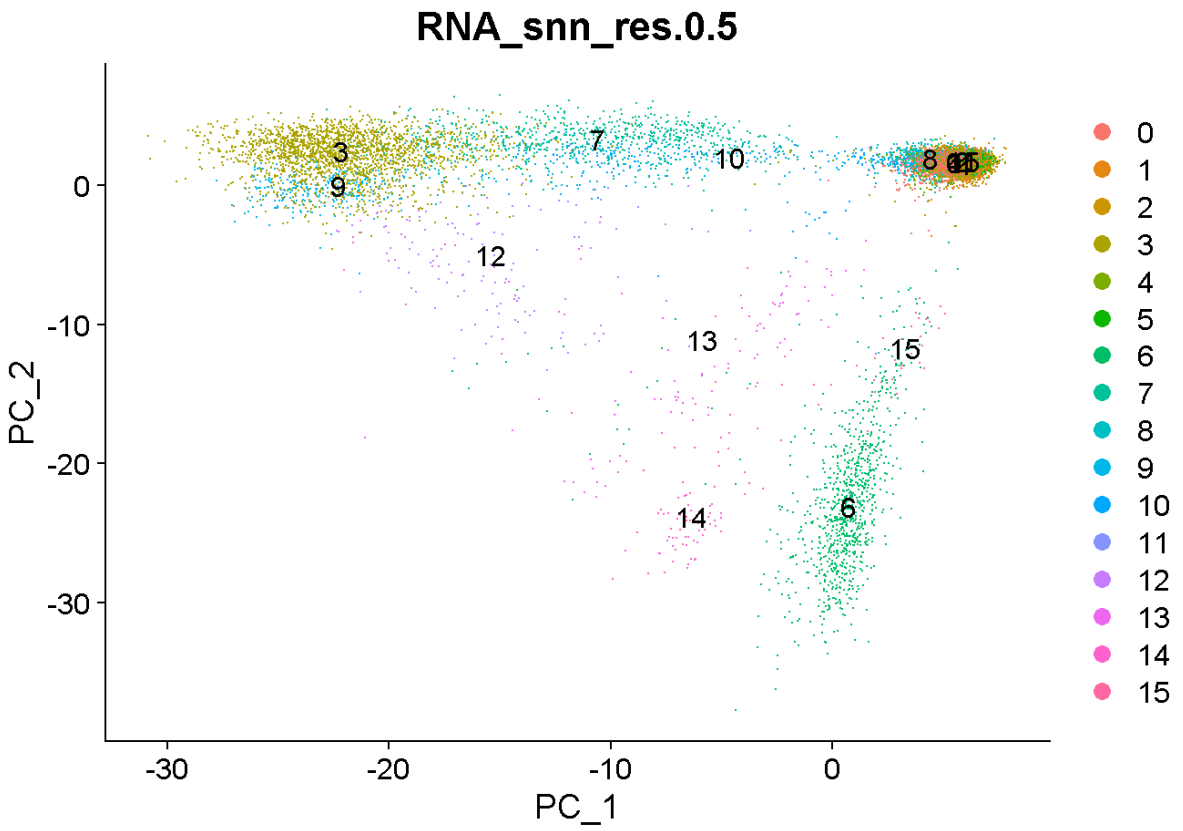
```
human_obj <- FindNeighbors(human_obj, dims = 1:15)
human_obj <- FindClusters(human_obj, resolution = c(0, 0.3, 0.5, 0.7, 1))
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 16804
## Number of edges: 586593
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 1.0000
## Number of communities: 2
## Elapsed time: 4 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 16804
## Number of edges: 586593
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9361
## Number of communities: 15
## Elapsed time: 4 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 16804
## Number of edges: 586593
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9136
## Number of communities: 16
## Elapsed time: 6 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 16804
## Number of edges: 586593
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8941
## Number of communities: 17
## Elapsed time: 6 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 16804
## Number of edges: 586593
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8698
## Number of communities: 21
## Elapsed time: 5 seconds
```

```
head(Idents(human_obj), 5)
```

```
## AAACCTGAGAAACCTA-1 AAACCTGAGAAGGTGA-1 AAACCTGAGACTTGAA-1 AAACCTGAGCCAGGAT-1
##                  0                 13                  5                  7
## AAACCTGAGGGCACTA-1
##                  4
## Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```

```
#View(human_obj@meta.data)

DimPlot(human_obj, group.by = "RNA_snn_res.0.5", label = TRUE)
```

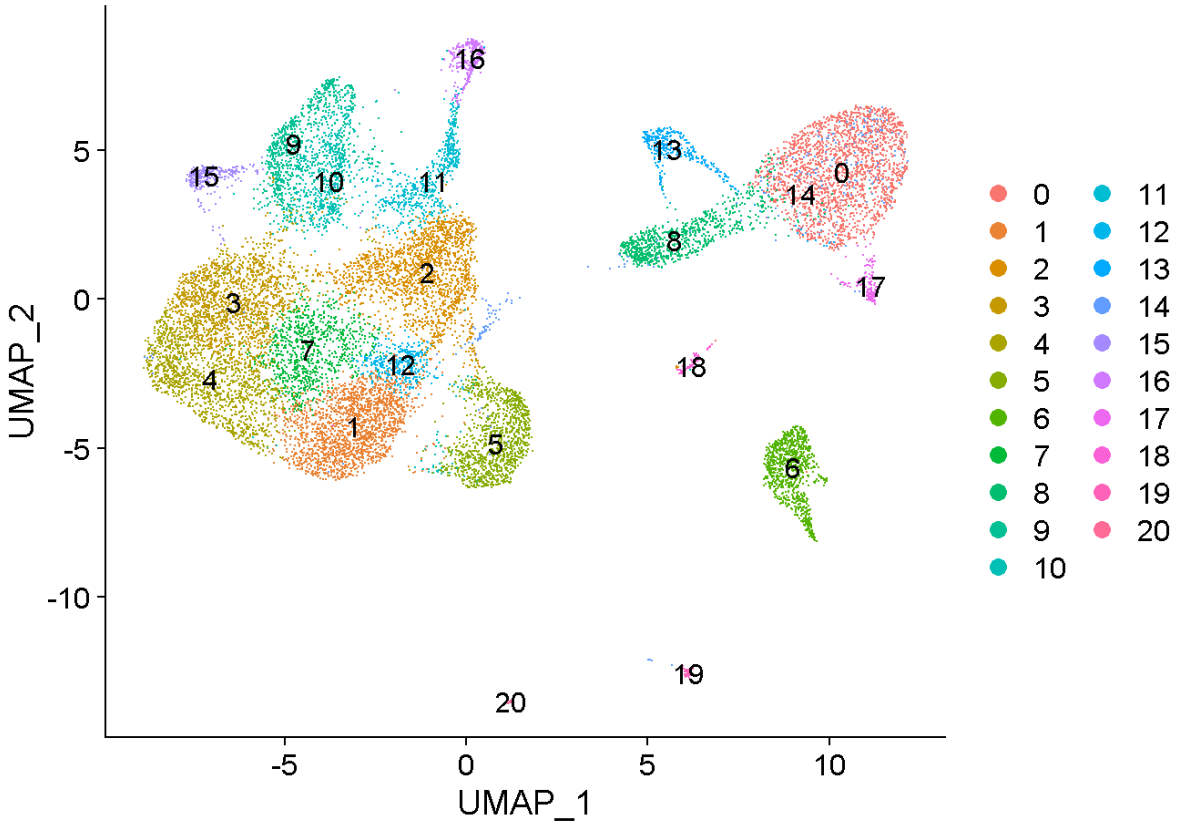



```
### Setting identity clusters

#Idents(human_obj)
#Idents(human_obj) <- "RNA_snn_res.0.1"
#Idents(human_obj)
```

Non-Linear Dimensionality Reduction (UMAP)

The goal of this algorithm is to learn the underlying manifold of the data in order to place similar cells together in low-dimensional space

```
library(reticulate)
py_install(packages = "umap-learn")


# Using the same PCs as previously.
human_obj <- RunUMAP(human_obj, dims = 1:15)

DimPlot(human_obj, reduction = "umap", label = TRUE)
```



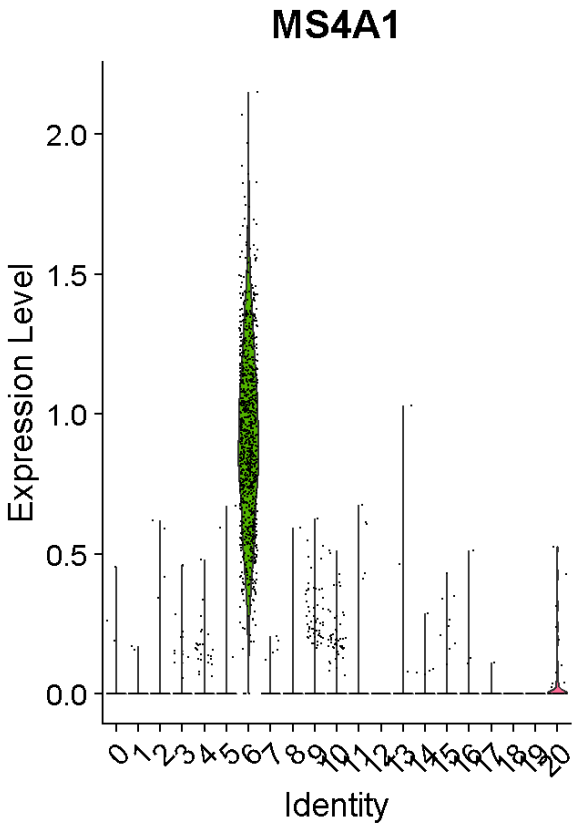
Saving the Seraut Object

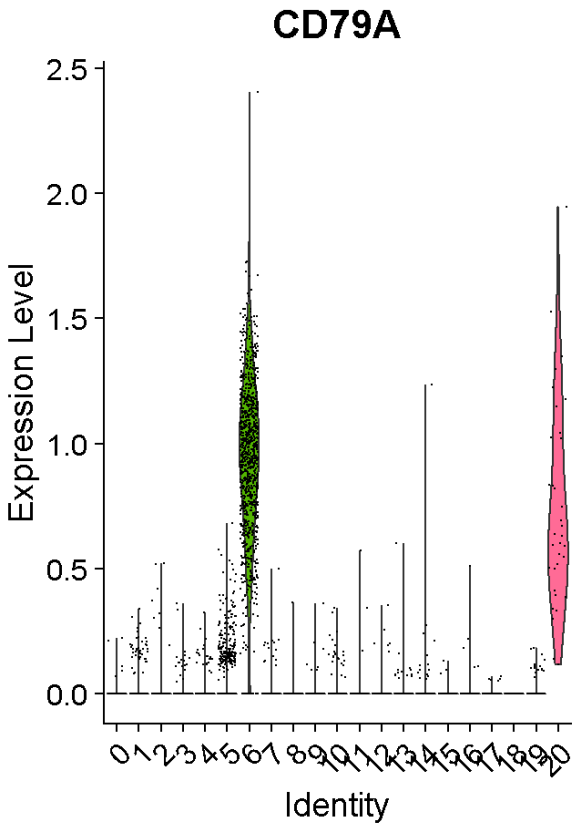
```
#saveRDS(human_obj, file = "PBMC_OBJECT.rds")
```

Finding Differentially Expressed Features

```
# Finding all markers of cluster 2
cluster2.markers <- FindMarkers(human_obj, ident.1 = 2, min.pct = 0.25)

# Finding all markers distinguishing cluster 5 from clusters 0 and 3
cluster5.markers <- FindMarkers(human_obj, ident.1 = 5, ident.2 = c(0, 3), min.pct = 0.25)
```

```
VlnPlot(human_obj, features = c("MS4A1", "CD79A"))
```


MS4A1
Expression Level
0.0
0.5
1.0
1.5
2.0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Identity


CD79A
Expression Level
0.0
0.5
1.0
1.5
2.0
2.5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Identity