# RNA_SequenceAnalysis

Antonio Pano

2022-11-22

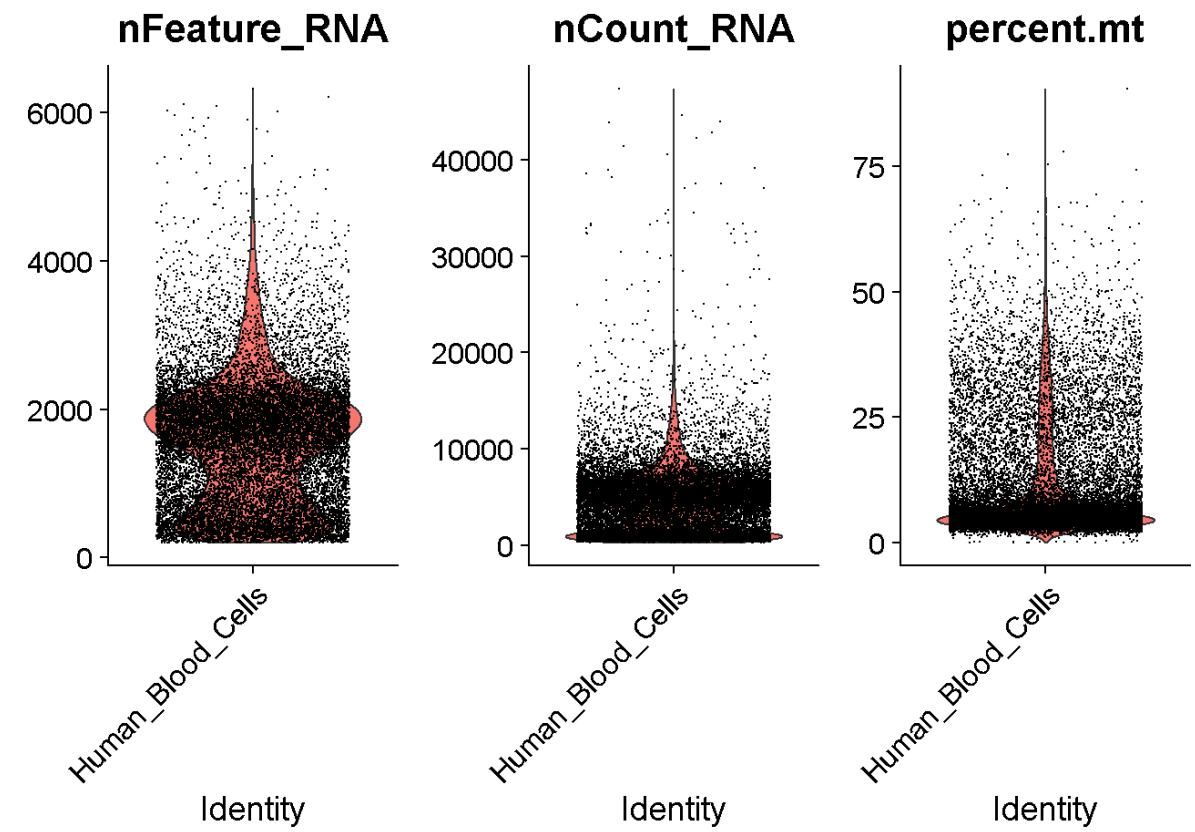
# https://www.10xgenomics.com/resources/datasets/20-k-human-pbm-cs-5-ht-v-2-0-2-high-6-1-0 (https://www.10xgenomics.com/resources/datasets/20-k-human-pbm-cs-5-ht-v-2-0-2-high-6-1-0)

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

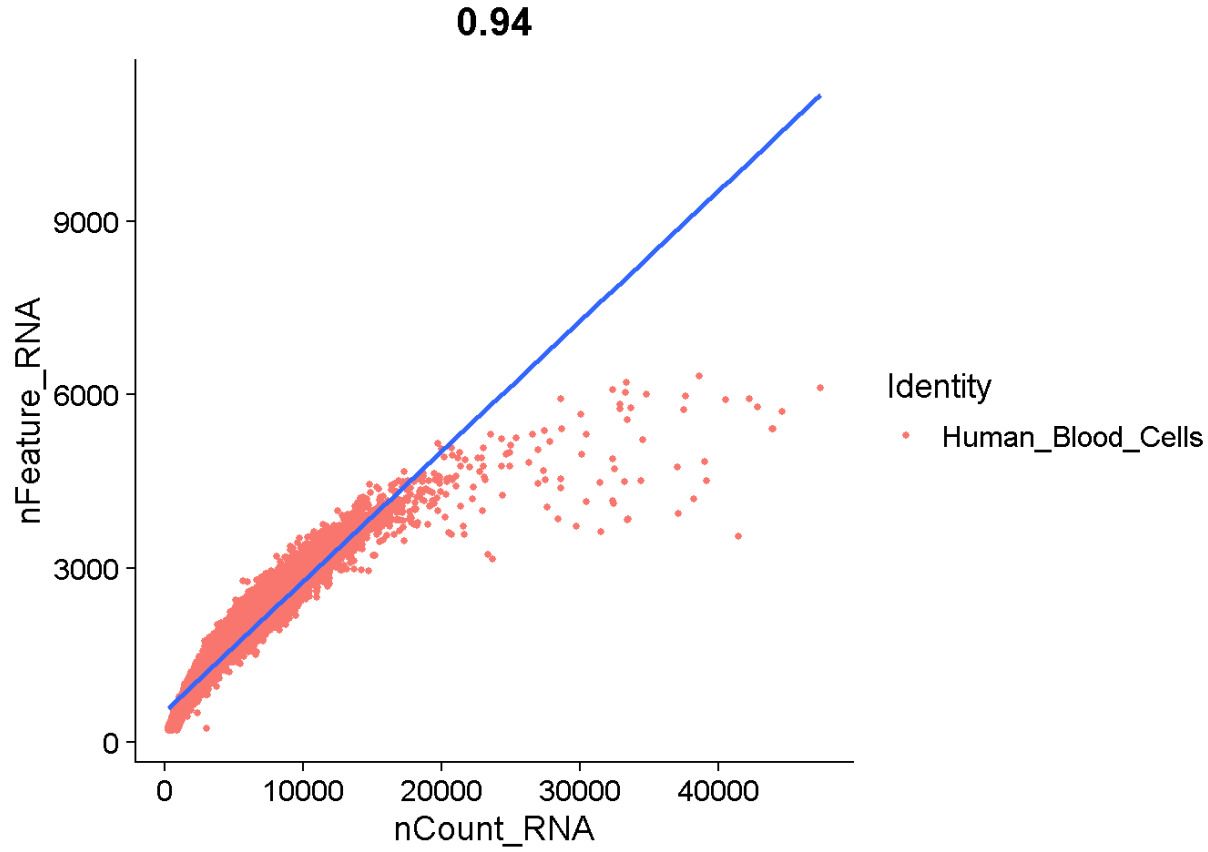
```
#loading matrix
Human <- Read10X_h5(filename = "20k_PBMC_5pv2_HT_nextgem_Chromium_X_Multiplex_count_raw_feature_bc_matrix.h
5")


#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)







#Lungs <- Read10X_h5(filename = "20k_NSCLC_DTC_3p_nextgem_Multiplex_count_raw_feature_bc_matrix.h5")
```

# 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)
```




```
FeatureScatter(human_obj, feature1 = "nCount_RNA", feature2 = "nFeature_RNA") +
  geom_smooth(method = "lm")
```



# Filtering

```
human_obj <- subset(human_obj, subset = nFeature_RNA > 400 & nFeature_RNA < 2650 & percent.mt < 7)
```

# Normalization

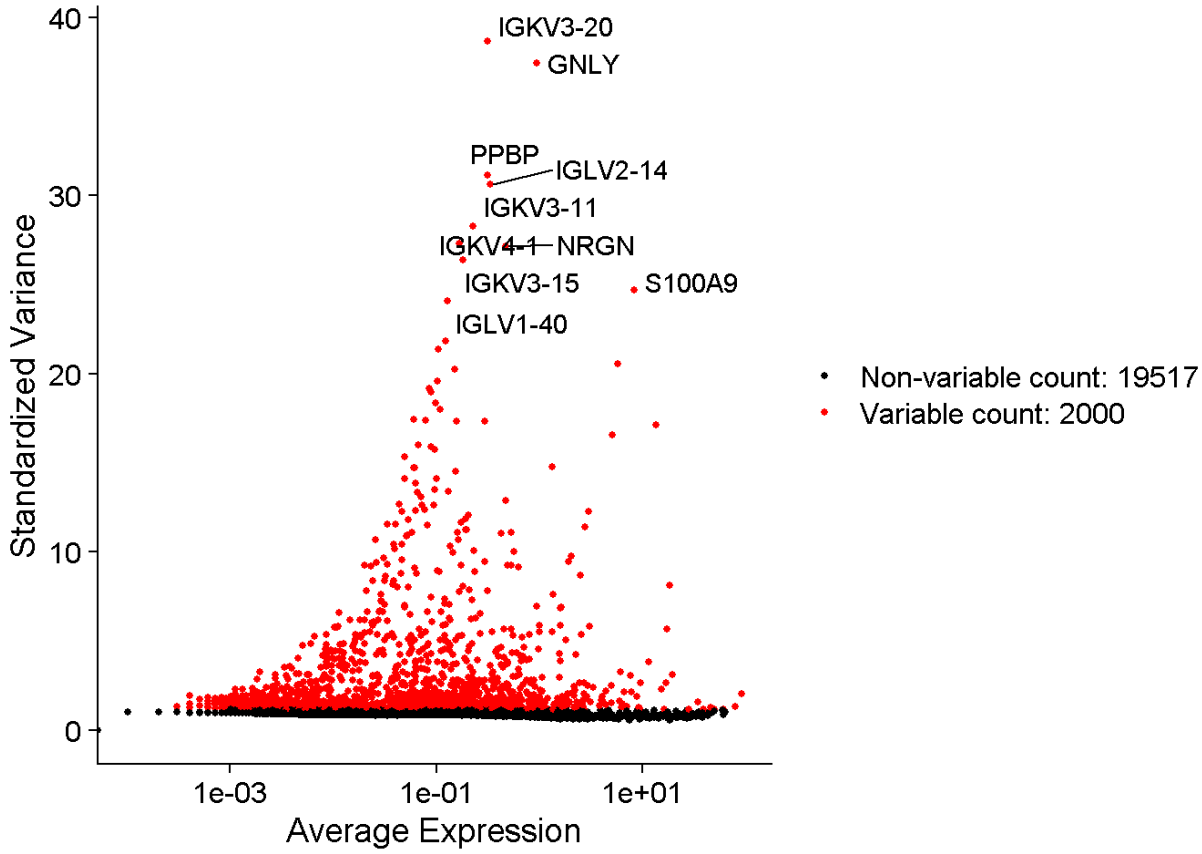
```
human_obj <- NormalizeData(human_obj, normalization.method = "LogNormalize", scale.factor = 1000)
```

# Identifying Highly Variable Features

```
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.

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

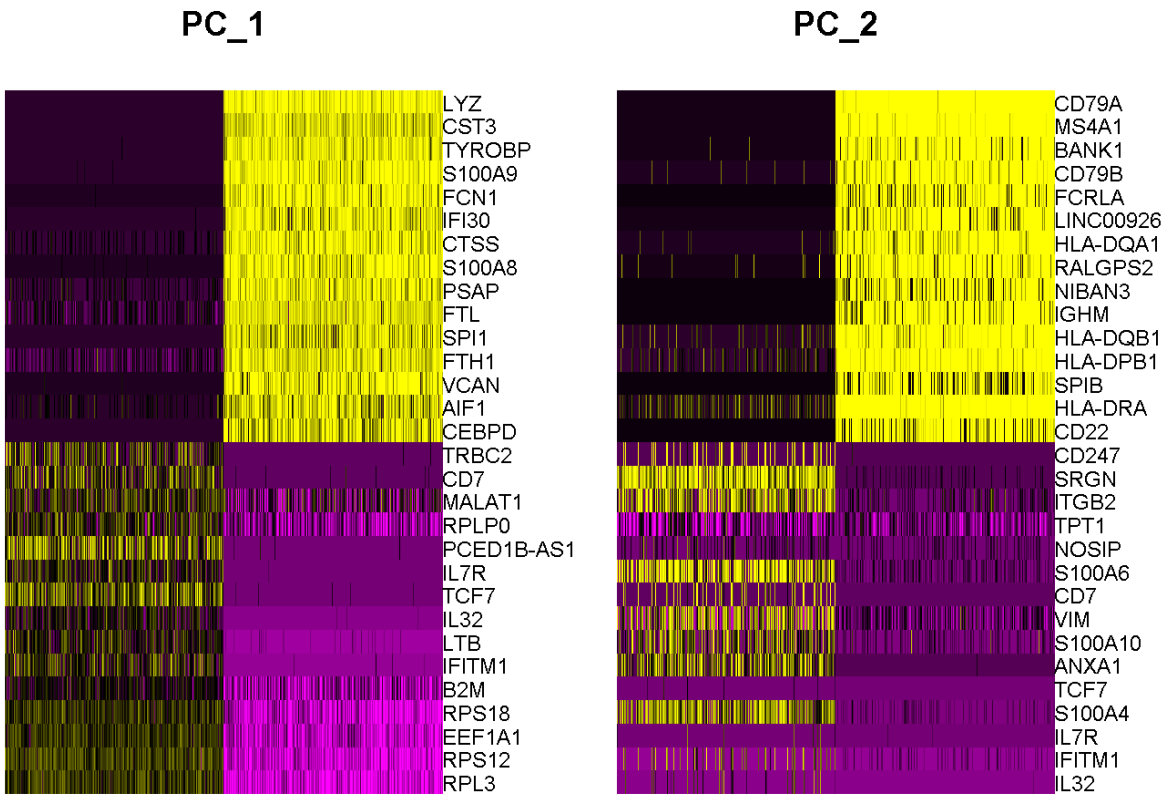
# Dimensionality Reduction (Principal Component Analysis)

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

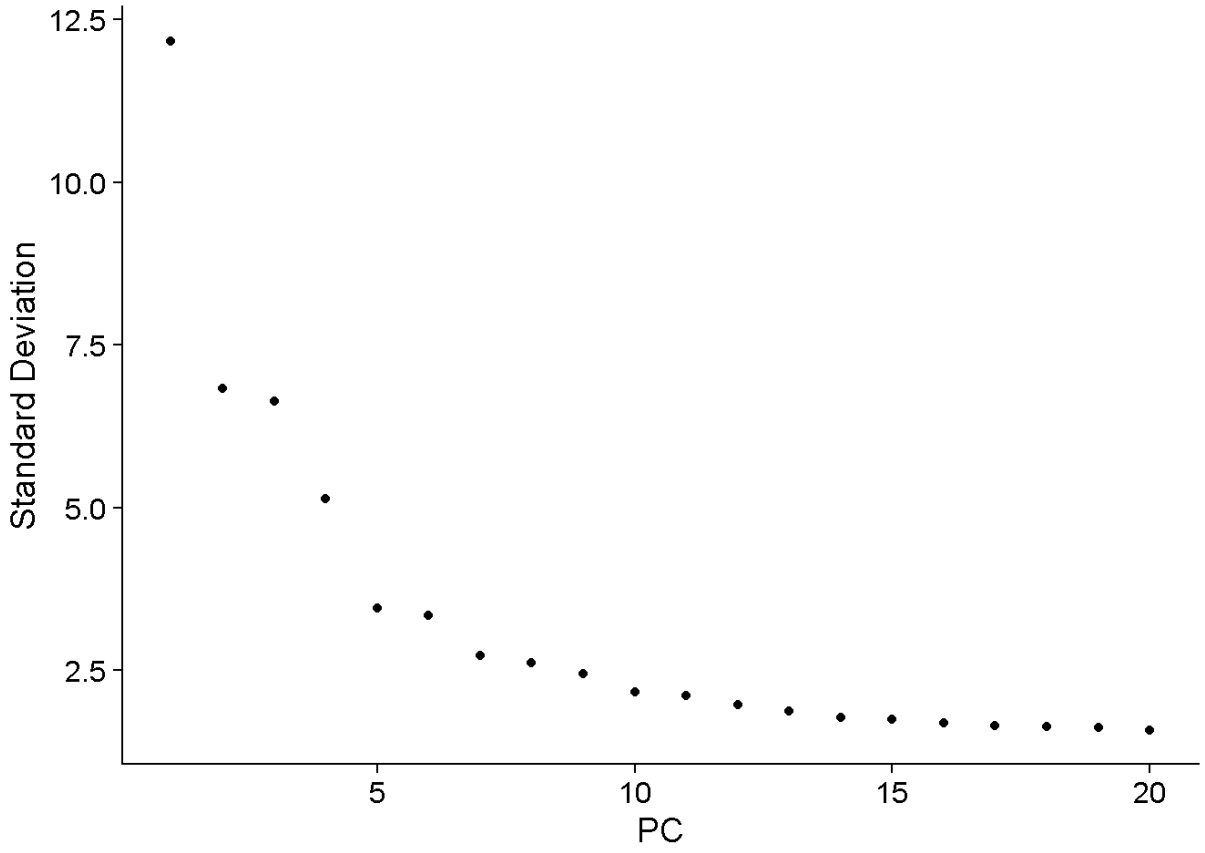
```
human_obj <- RunPCA(human_obj, features = VariableFeatures(object = human_obj))

# Visualizing the PCA plot
print(human_obj[["pca"]], dims = 1:5, nfeatures = TRUE)
```

```
## PC_ 1
## Positive:  RPL3
## Negative:  LYZ
## PC_ 2
## Positive:  IL32
## Negative:  CD79A
## PC_ 3
## Positive:  PPBP
## Negative:  EEF1A1
## PC_ 4
## Positive:  TPT1
## Negative:  NKG7
## PC_ 5
## Positive:  CRIP1
## Negative:  CD7
```

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




```
# Determining the dimensionality of the data to use only the PC's that capture the majority of the signal in this downstream analysis. Better to use elbow on the higher side of the x-axis bc the effects could be large.
ElbowPlot(human_obj)
```

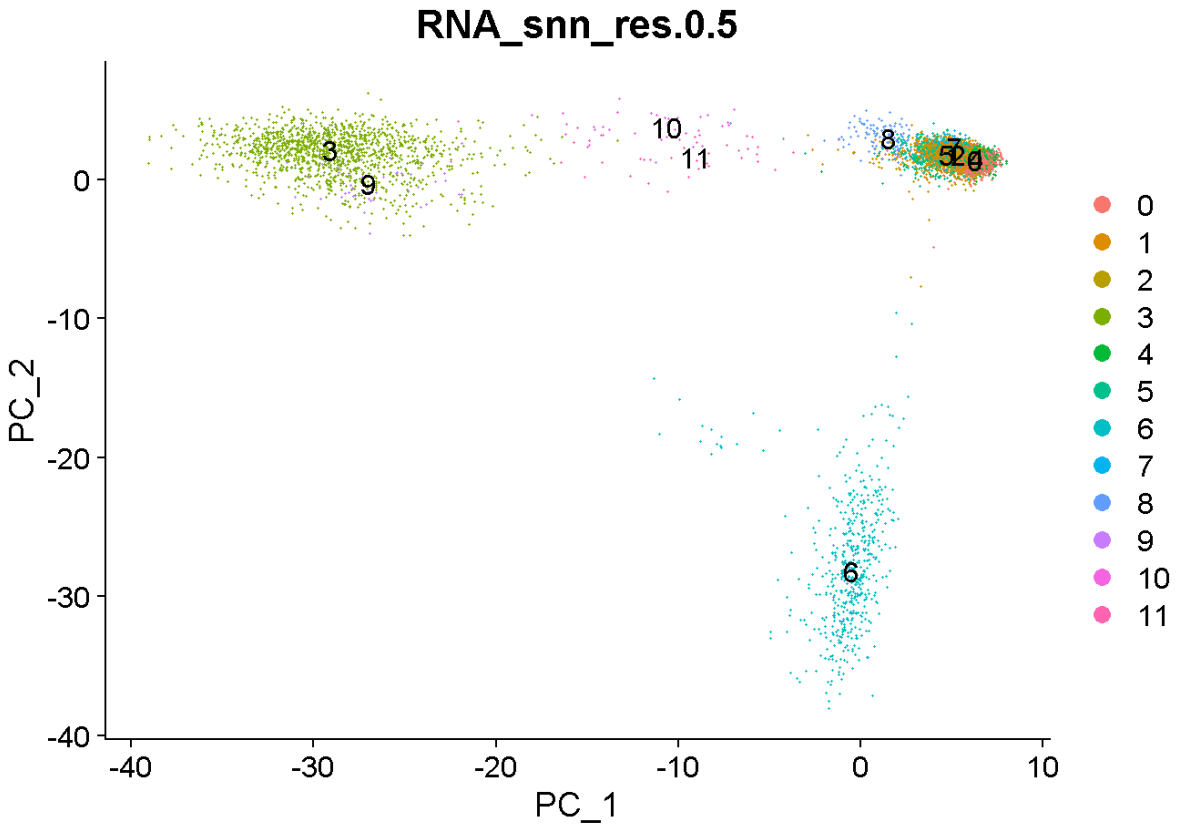


# Clustering

```
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: 9780
## Number of edges: 339344
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 1.0000
## Number of communities: 1
## Elapsed time: 2 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 9780
## Number of edges: 339344
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9222
## Number of communities: 10
## Elapsed time: 2 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 9780
## Number of edges: 339344
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8904
## Number of communities: 12
## Elapsed time: 3 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 9780
## Number of edges: 339344
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8669
## Number of communities: 15
## Elapsed time: 3 seconds
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 9780
## Number of edges: 339344
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8377
## Number of communities: 18
## Elapsed time: 3 seconds
```

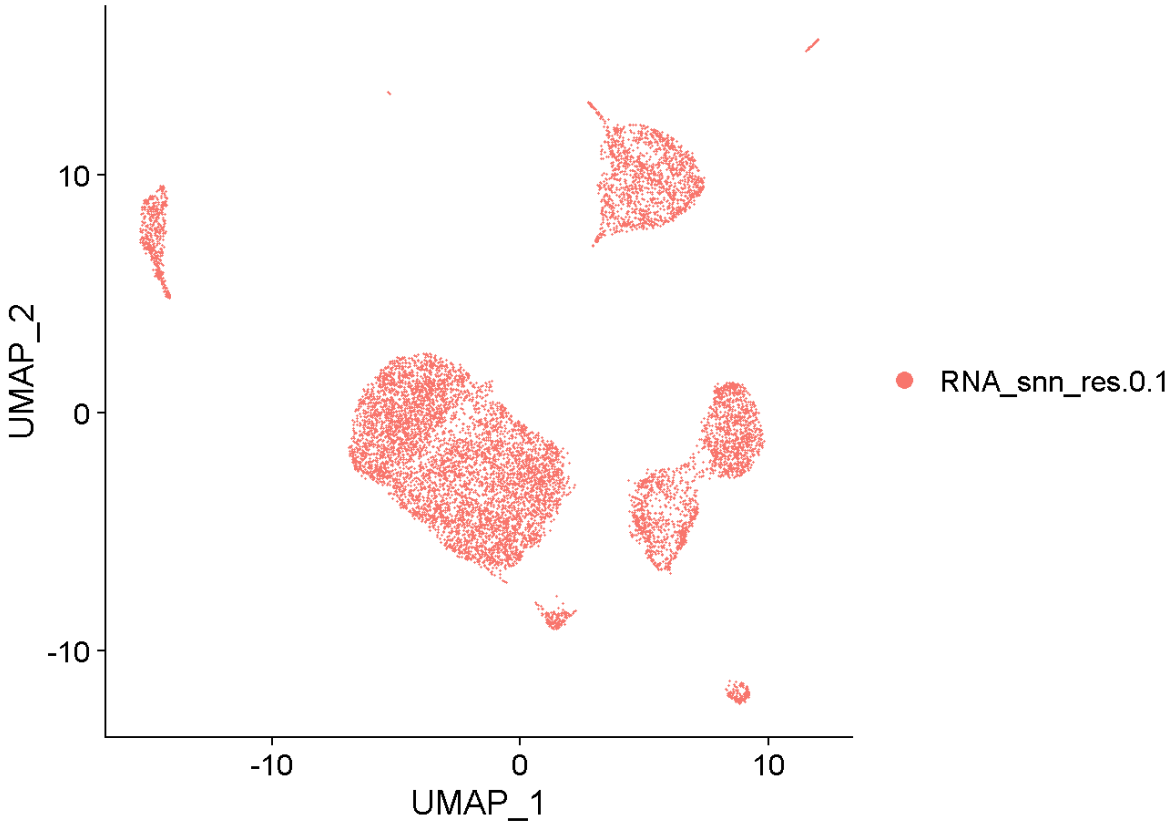
```
View(human_obj@meta.data)

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)

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

human_obj <- RunUMAP(human_obj, dims = 1:15)

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


10
0
-10
UMAP_2
-10
0
10
UMAP_1
RNA_snn_res.0.1