Seurat

Parissa Amin

2022-10-04

Creating Seurat object

Loading packages and data to create Seurat object.

```
library(Seurat)
```

```
## Loading required package: SeuratObject
```

```
## Loading required package: sp
```

```
## Warning: package 'sp' was built under R version 4.2.3
```

```
## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)
```

```
##
## Attaching package: 'SeuratObject'
```

```
## The following objects are masked from 'package:base':
##
##     intersect, saveRDS
```

```
## Loading Seurat v5 beta version
## To maintain compatibility with previous workflows, new Seurat objects will use the previous object s
## To use new Seurat v5 assays please run: options(Seurat.object.assay.version = 'v5')
```

```
library(Matrix)
```

```
## Warning: package 'Matrix' was built under R version 4.2.3
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.2.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.2.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library (patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.2.3
```

```
metadata <- read.csv("D:/scRNASeq/Data/metadata.csv", header = TRUE, row.names = 1)

gene_names <- read.table("D:/scRNASeq/Data/genes.tsv", header = FALSE, col.names =
                           "GeneName", stringsAsFactors = FALSE)$GeneName

mtx_matrix<-readMM("D:/scRNASeq/Data/matrix.mtx")

rownames(mtx_matrix) <- gene_names
colnames(mtx_matrix) <- rownames(metadata)
seurat_obj <- CreateSeuratObject(counts = Matrix::Matrix(as.matrix(mtx_matrix),sparse = T), meta.data =
```

```
## Warning in asMethod(object): sparse->dense coercion: allocating vector of size
## 22.2 GiB
```

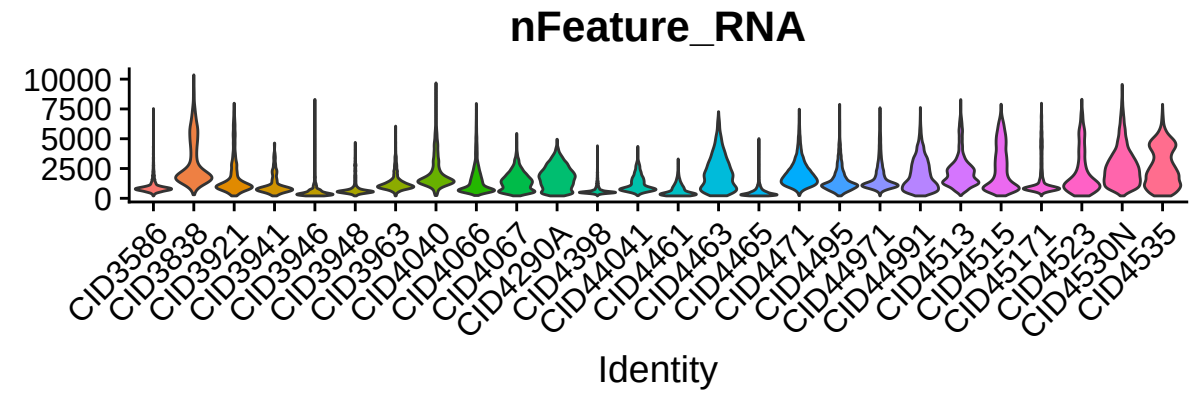
#Quality Control

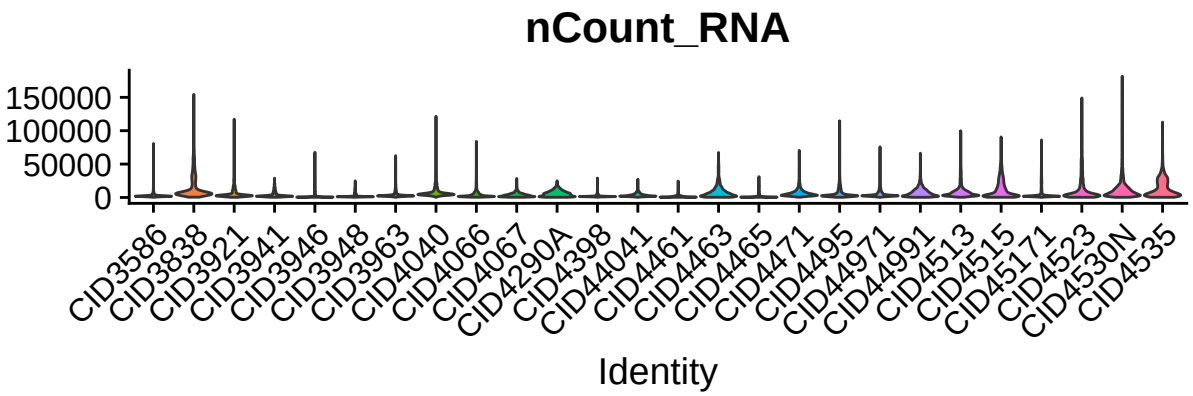
While the data is already processed, we perform a brief quality control.

```
# Violin Plot
VlnPlot(object = seurat_obj, features = c('nFeature_RNA','nCount_RNA','pct.Ribo',
                                          'pct.mito'), group.by = 'orig.ident',pt.size = 0,ncol=1)
```

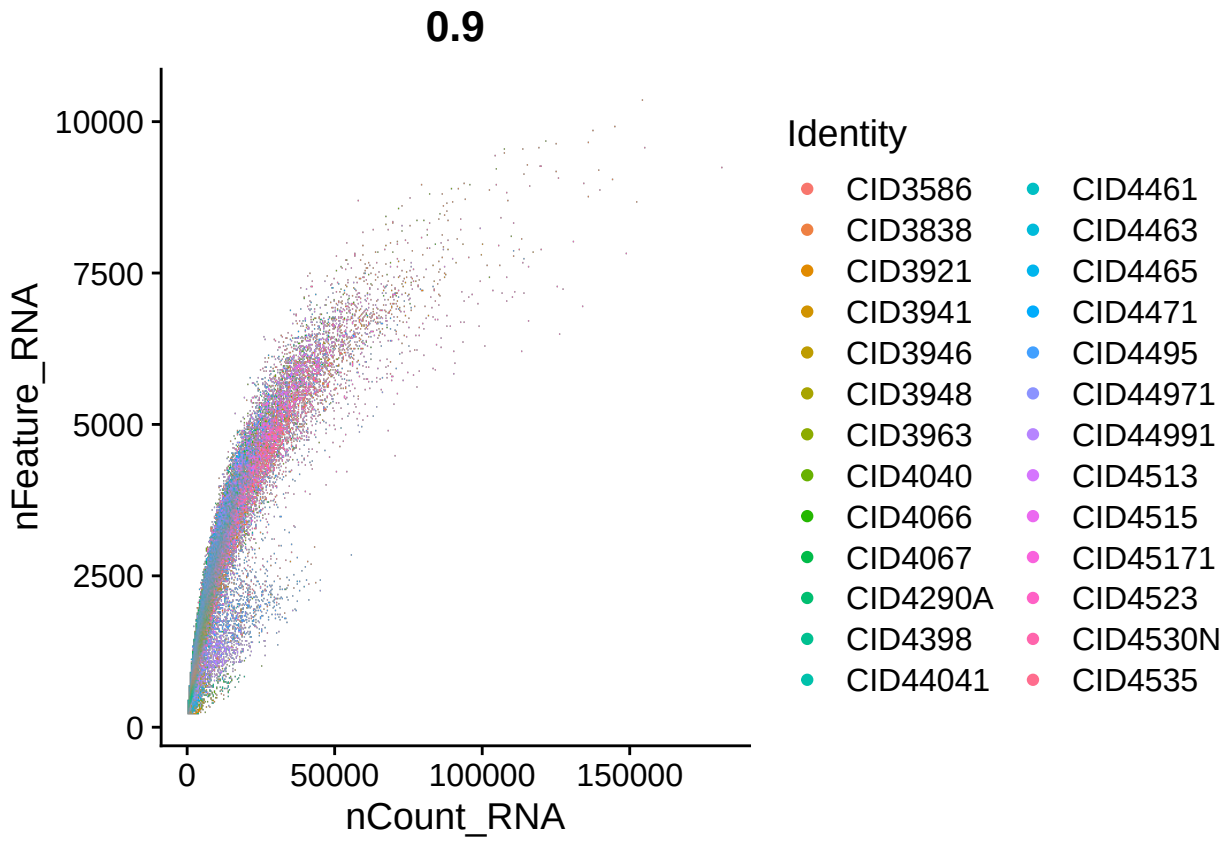
```
## Warning: The following requested variables were not found: pct.Ribo, pct.mito
```

```
## Rasterizing points since number of points exceeds 100,000.
## To disable this behavior set `raster=FALSE`
## Rasterizing points since number of points exceeds 100,000.
## To disable this behavior set `raster=FALSE`
```






```
# FeatureScatter can be used to visualize feature-feature relationships such as number
#of genes ("nFeature_RNA") vs number of UMIs ("nCount_RNA")
FeatureScatter(seurat_obj, feature1 = "nCount_RNA", feature2 = "nFeature_RNA",group.by = 'orig.ident')
```

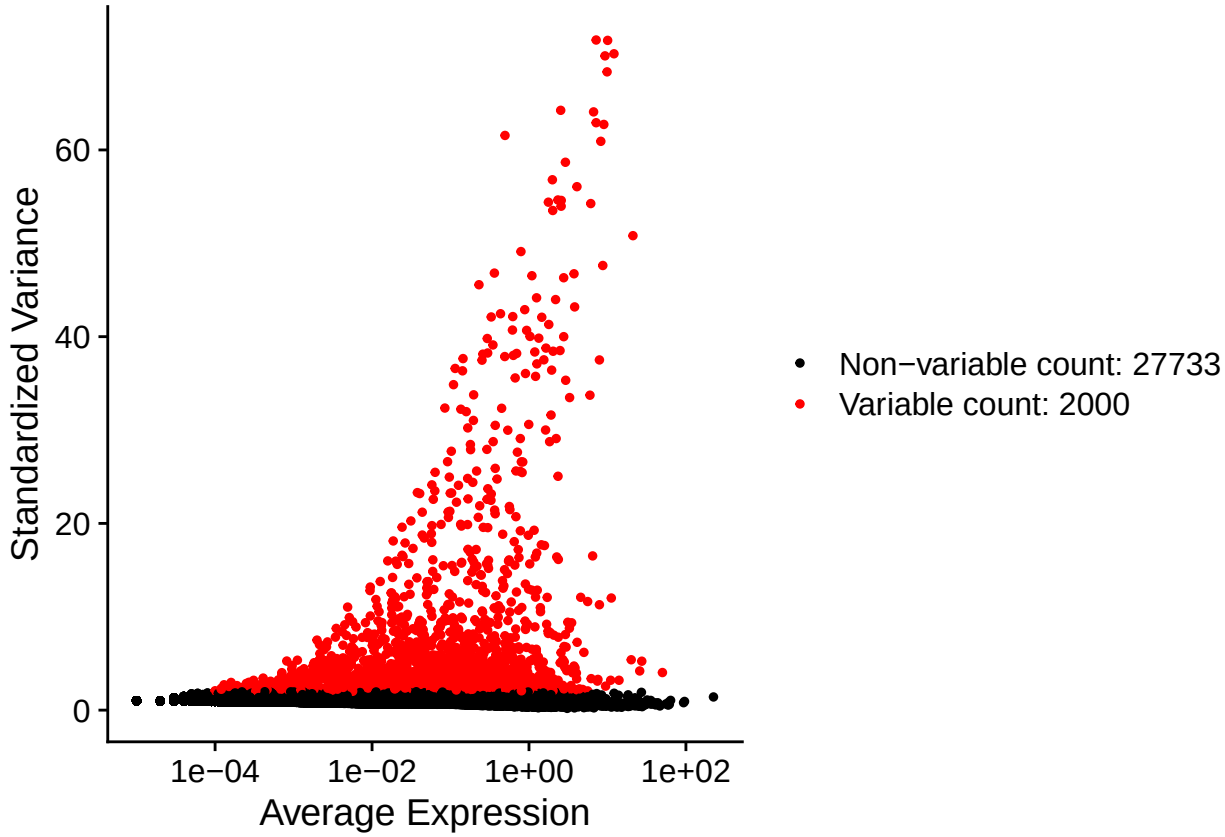


#Identifying Variable Genes

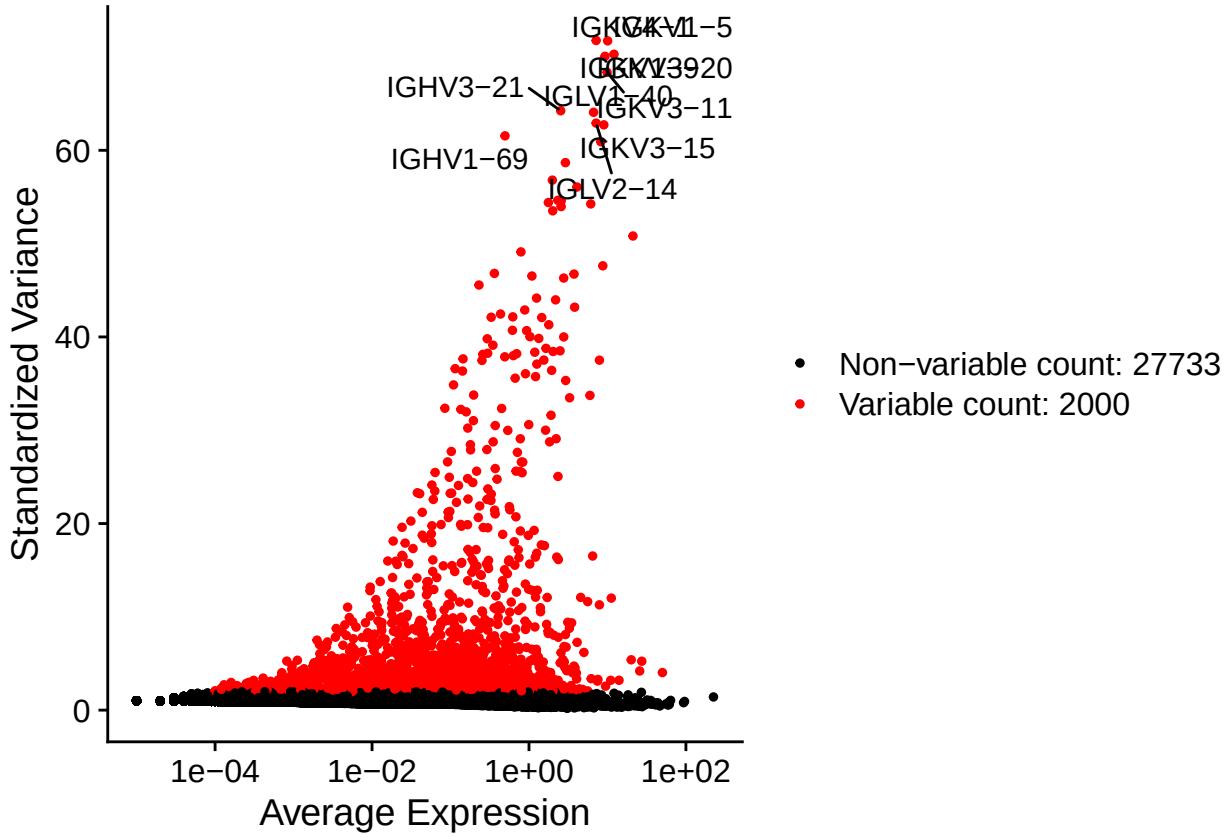
```
seurat_obj <- FindVariableFeatures(seurat_obj,
                                   selection.method = "vst", nfeatures = 2000)

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

# plot variable features with and without labels
plot1 <- VariableFeaturePlot(seurat_obj)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
```

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

```
plot1
```


Standardized Variance
60
40
20
0
1e−04
1e−02
1e+00
1e+02
Average Expression
Non−variable count: 27733
Variable count: 2000

```
plot2
```



#Scaling data

My personal PC doesnt run the full scaling using

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

So I went on with the recommended substitute:

```
seurat_obj <- ScaleData(seurat_obj)
```

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

#Perform linear dimensional reduction

Next we perform PCA on the scaled data. By default, only the previously determined variable features are used as input, but can be defined using features argument if one wishes to choose a different subset.

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

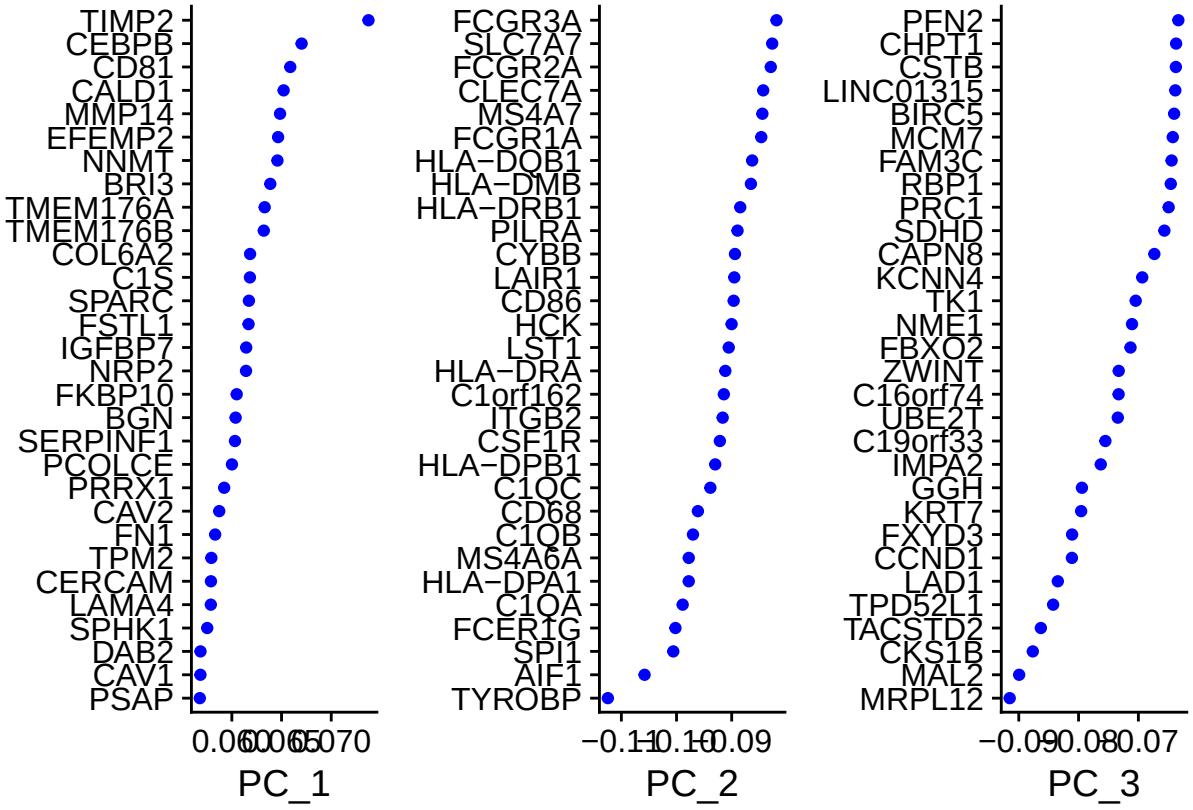
```
## PC_ 1
## Positive:  TIMP2, CEBPB, CD81, CALD1, MMP14, EFEMP2, NNMT, BRI3, TMEM176A, TMEM176B
##     COL6A2, C1S, SPARC, FSTL1, IGFBP7, NRP2, FKBP10, BGN, SERPINF1, PCOLCE
##     PRRX1, CAV2, FN1, TPM2, CERCAM, LAMA4, SPHK1, DAB2, CAV1, PSAP
## Negative:  GZMK, ZNF683, GZMH, CD40LG, RP11-291B21.2, CTLA4, GNLY, CD79A, KLRC1, IFNG
##     GZMB, CCR7, XCL1, TRDC, MZB1, TNFRSF18, MS4A1, TRBV7-9, TRGC2, FOXP3
##     XCL2, TRBV20-1, TRAV13-1, TNFRSF17, TRAV8-2, KIAA0125, IGHG1, TRBV19, TRAV29DV5, TRBV5-1
## PC_ 2
## Positive:  IGFBP7, CALD1, NNMT, SPARC, BGN, FSTL1, EFEMP2, PCOLCE, PRRX1, C1S
##     COL6A2, TPM2, MYL9, LAMA4, SPARCL1, MXRA8, COL6A3, FKBP10, LAMB1, LHFP
##     DKK3, C1R, MFGE8, PLAC9, PDGFRB, NID1, MAP1B, RCN3, THY1, GGT5
```

```
## Negative:  TYROBP, AIF1, SPI1, FCER1G, C1QA, HLA-DPA1, MS4A6A, C1QB, CD68, C1QC
##     HLA-DPB1, CSF1R, ITGB2, C1orf162, HLA-DRA, LST1, HCK, CD86, LAIR1, CYBB
##     PILRA, HLA-DRB1, HLA-DMB, HLA-DQB1, FCGR1A, MS4A7, CLEC7A, FCGR2A, SLC7A7, FCGR3A
## PC_ 3
## Positive:  IGFBP7, SPARC, SERPINF1, BGN, SPARCL1, MXRA8, FSTL1, PRRX1, PCOLCE, PLAC9
##     NID1, TIMP2, FBLN2, LHFP, PDGFRB, A2M, OLFML3, DCN, EFEMP2, PODN
##     COL6A3, THY1, GGT5, CFH, CRISPLD2, COL5A1, ANGPTL2, COL3A1, COL6A2, LUM
## Negative:  MRPL12, MAL2, CKS1B, TACSTD2, TPD52L1, LAD1, CCND1, FXYD3, KRT7, GGH
##     IMPA2, C19orf33, UBE2T, C16orf74, ZWINT, FBXO2, NME1, TK1, KCNN4, CAPN8
##     SDHD, PRC1, RBP1, FAM3C, MCM7, BIRC5, LINC01315, CSTB, CHPT1, PFN2
## PC_ 4
## Positive:  RAMP2, EMCN, ESAM, PECAM1, ADGRL4, PLVAP, CYYR1, CLEC14A, CXorf36, CD34
##     PALMD, MMRN2, AQP1, VWF, RAMP3, PTPRB, CD93, EGFL7, CDH5, GNG11
##     JAM2, SPARCL1, IL3RA, ADAMTS9, SPRY1, A2M, MCTP1, THSD7A, IL33, APOLD1
## Negative:  COL6A3, MXRA8, DCN, LUM, CERCAM, CTHRC1, FBLN1, COL3A1, SFRP2, ADAM12
##     C1S, COL5A1, MFAP2, COL1A1, ISLR, CTSK, RCN3, PODN, MXRA5, AEBP1
##     SERPINF1, PDGFRL, FAP, EFEMP2, COL6A1, CCDC80, COL6A2, COL5A2, COL1A2, THBS2
## PC_ 5
## Positive:  LEMD1, FSCN1, KCNN4, MRAS, CAV2, MSLN, CASC15, MIA, FAM3C, MKI67
##     SGOL1, XDH, KRT6B, IMPA2, OBP2B, RARRES1, TK1, KLK6, GJB3, KIF23
##     ASF1B, CKS1B, CALB2, PLCXD3, PRC1, CENPF, C6orf15, LDHB, SERPINB5, BIRC5
## Negative:  AGR3, STARD10, MAGED2, SLC39A6, H2AFJ, ZNF703, MLLT4, ARMT1, C1orf64, TSPAN13
##     AFF3, AC018816.3, ELP2, ACADSB, DNAJC12, TMEM150C, FSIP1, DHRS2, ESR1, KCNJ3
##     RP11-53O19.1, MRPS30, STC2, GJA1, GFRA1, TRH, GSTM3, CHPT1, TNNT1, BCAS1
```

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

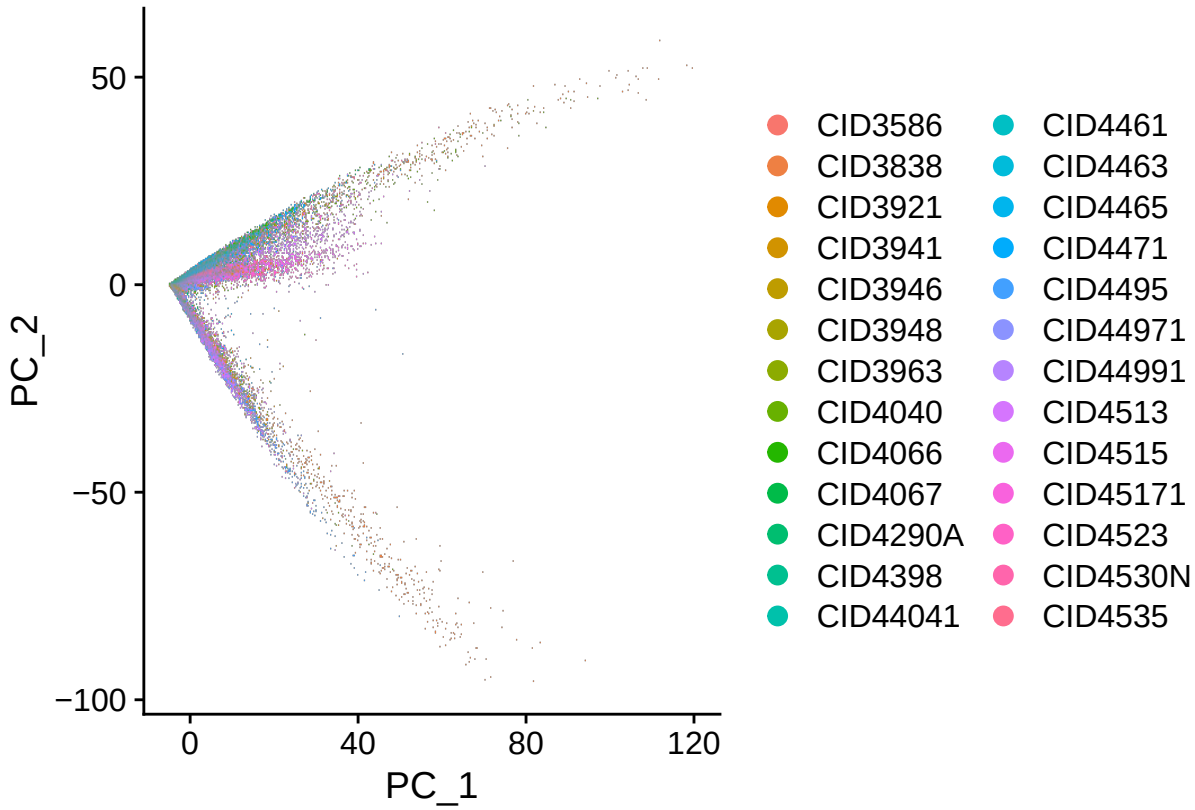
```
## PC_ 1
## Positive:  TIMP2, CEBPB, CD81, CALD1, MMP14
## Negative:  GZMK, ZNF683, GZMH, CD40LG, RP11-291B21.2
## PC_ 2
## Positive:  IGFBP7, CALD1, NNMT, SPARC, BGN
## Negative:  TYROBP, AIF1, SPI1, FCER1G, C1QA
## PC_ 3
## Positive:  IGFBP7, SPARC, SERPINF1, BGN, SPARCL1
## Negative:  MRPL12, MAL2, CKS1B, TACSTD2, TPD52L1
## PC_ 4
## Positive:  RAMP2, EMCN, ESAM, PECAM1, ADGRL4
## Negative:  COL6A3, MXRA8, DCN, LUM, CERCAM
## PC_ 5
## Positive:  LEMD1, FSCN1, KCNN4, MRAS, CAV2
## Negative:  AGR3, STARD10, MAGED2, SLC39A6, H2AFJ
```

```
VizDimLoadings(seurat_obj, dims = 1:3, reduction = "pca", ncol=3)
```

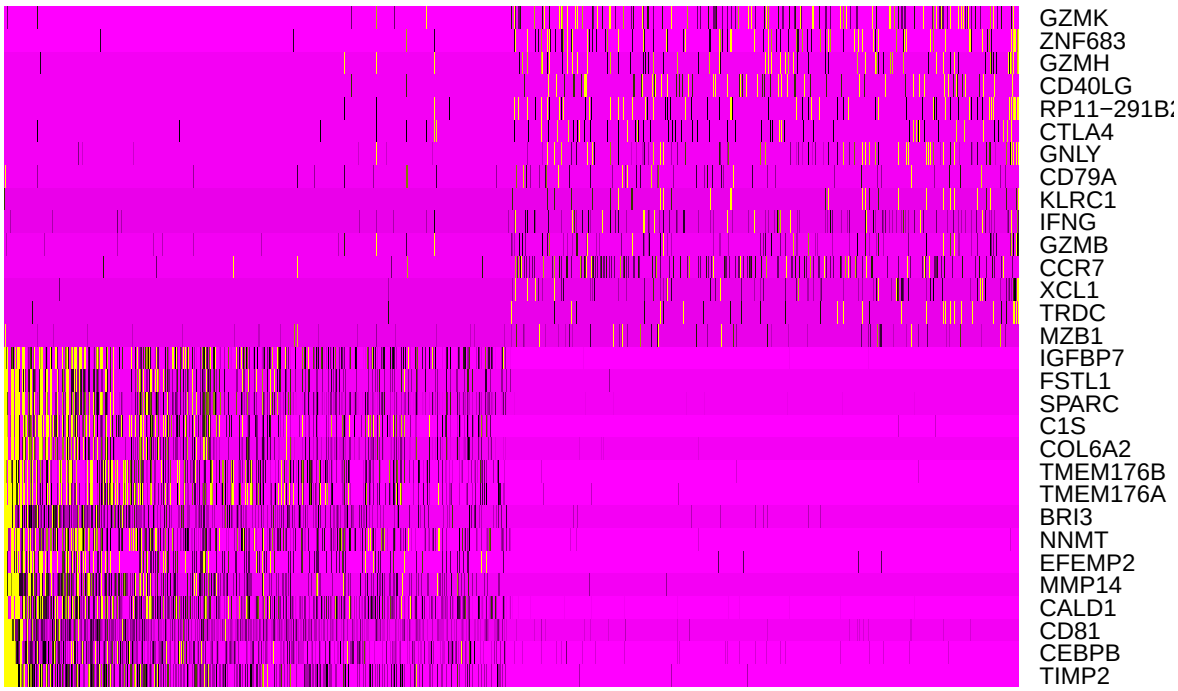



```
DimPlot(seurat_obj, reduction = "pca")
```

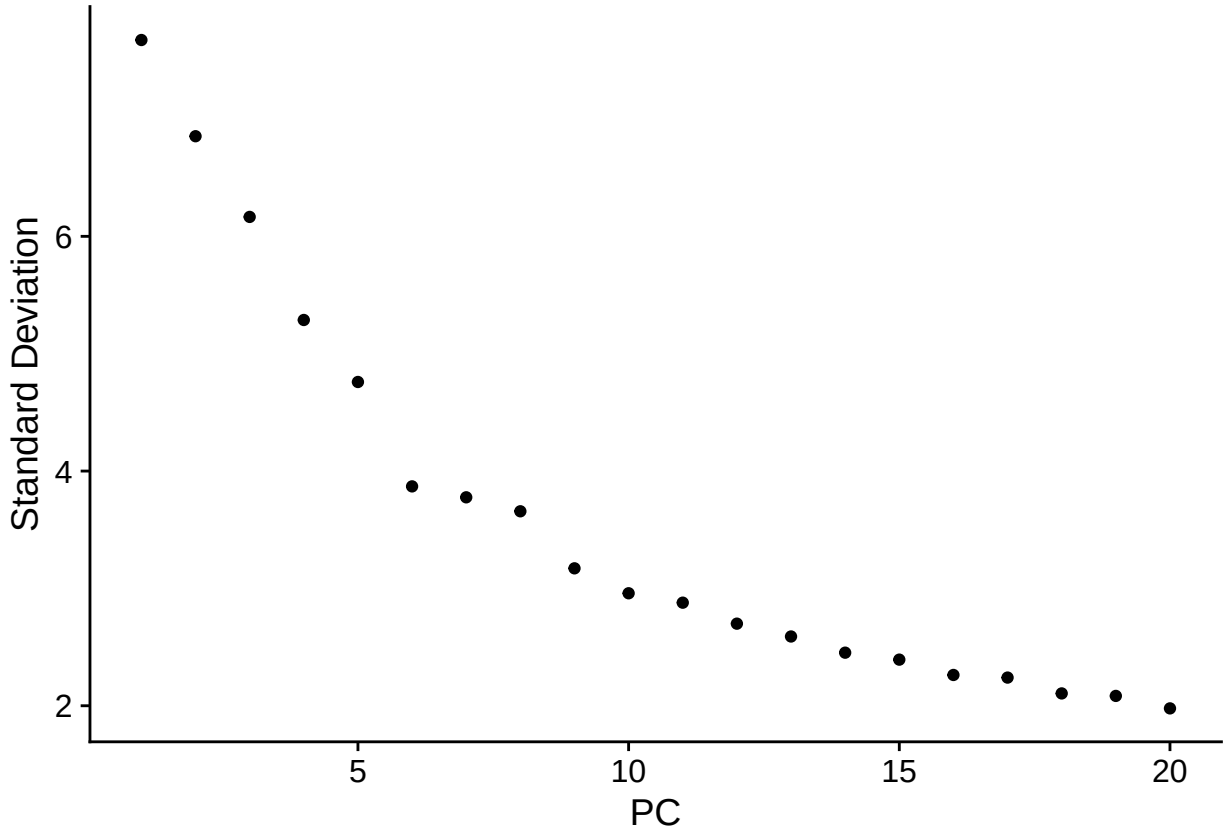
```
## Rasterizing points since number of points exceeds 100,000.
## To disable this behavior set `raster=FALSE`
```




```
DimHeatmap(seurat_obj, dims = 1, cells = 50000, balanced = TRUE)
```




```
ElbowPlot(seurat_obj)
```



#Find clusters

I have tried different values for dim in seurat_obj <-FindNeighbors(seurat_obj, dims = 1:30) and it significantly alters the results! I continued using dim=1:30

```
seurat_obj <- FindNeighbors(seurat_obj, dims = 1:30)
```

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

```
## Computing SNN
```

```
seurat_obj <- FindClusters(seurat_obj, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 100064
## Number of edges: 3491188
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.9558
## Number of communities: 33
## Elapsed time: 36 seconds
```

```
## 1 singletons identified. 32 final clusters.
```

```
head(Idents(seurat_obj), 5)
```

```
## CID3586_AAGACCTCAGCATGAG CID3586_AAGGTTCGTAGTACCT CID3586_ACCAGTAGTTGTGGCC
##                        4                       20                       20
## CID3586_ACCCACTAGATGTCGG CID3586_ACTGATGGTCAACTGT
```

```
##                       20                       20
## 32 Levels: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ... 31
```

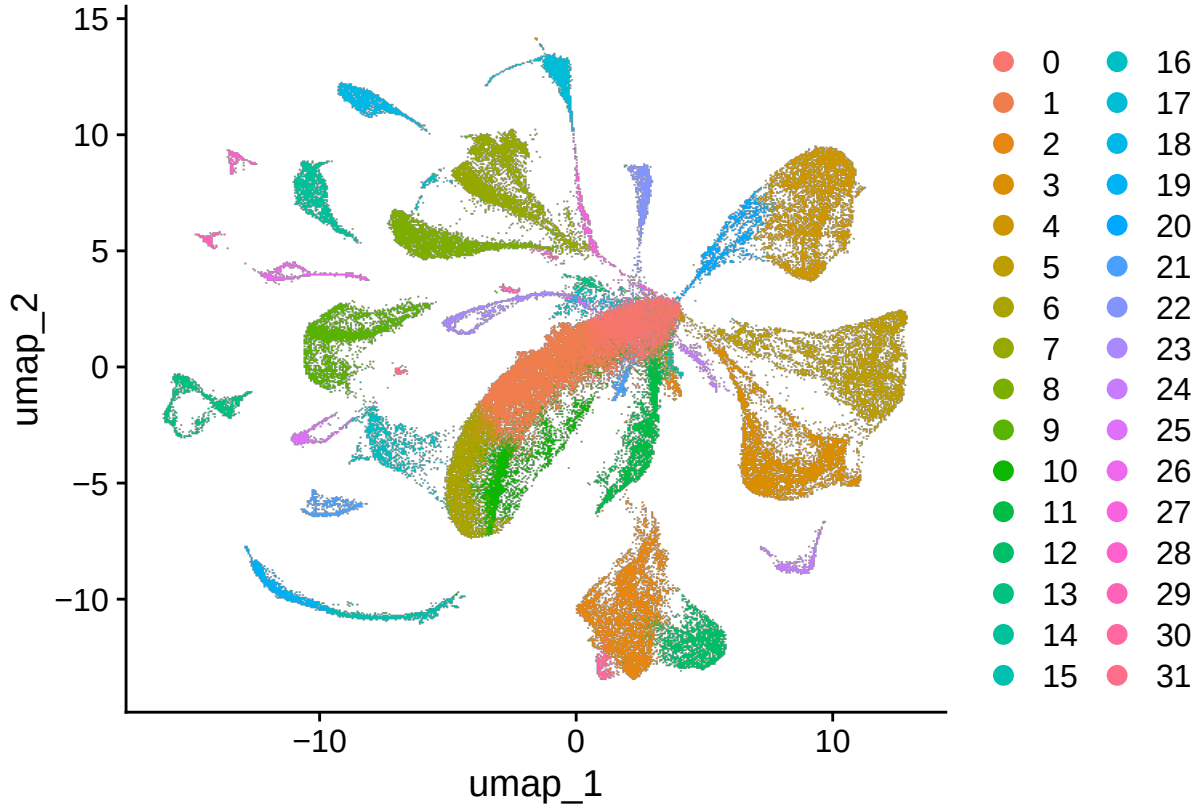
#UMAP

```
seurat_obj <- RunUMAP(seurat_obj, dims = 1:30)
```

```
## 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
## 10:37:23 UMAP embedding parameters a = 0.9922 b = 1.112
## Found more than one class "dist" in cache; using the first, from namespace 'spam'
## Also defined by 'BiocGenerics'
## 10:37:23 Read 100064 rows and found 30 numeric columns
## 10:37:23 Using Annoy for neighbor search, n_neighbors = 30
## Found more than one class "dist" in cache; using the first, from namespace 'spam'
## Also defined by 'BiocGenerics'
## 10:37:23 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 10:37:53 Writing NN index file to temp file C:\Users\PY\AppData\Local\Temp\RtmpK22ZgR\file2cb810764c
## 10:37:53 Searching Annoy index using 1 thread, search_k = 3000
## 10:38:32 Annoy recall = 84.63%
## 10:38:33 Commencing smooth kNN distance calibration using 1 thread with target n_neighbors = 30
## 10:38:39 Initializing from normalized Laplacian + noise (using RSpectra)
## 10:39:05 Commencing optimization for 200 epochs, with 4630342 positive edges
## 10:41:02 Optimization finished
```

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

```
## Rasterizing points since number of points exceeds 100,000.
## To disable this behavior set `raster=FALSE`
```



I continued using dim=1:30 here too.

#Markers for each cluster

Find markers for every cluster compared to all remaining cells, report only the positive ones

```
seurat_obj.markers <- FindAllMarkers(seurat_obj, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25
```

```
## 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
## Calculating cluster 19
## Calculating cluster 20
## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =
## counts, : No features pass logfc.threshold threshold; returning empty
## data.frame
## Calculating cluster 21
## Calculating cluster 22
## Calculating cluster 23
## Calculating cluster 24
## Calculating cluster 25
## Calculating cluster 26
## Calculating cluster 27
## Warning in FindMarkers.default(object = data.use, slot = data.slot, counts =
## counts, : No features pass logfc.threshold threshold; returning empty
## data.frame
## Calculating cluster 28
## Calculating cluster 29
## Calculating cluster 30
## Calculating cluster 31
```

```
seurat_obj.markers %>%
  group_by(cluster) %>%
  slice_max(n = 2, order_by = avg_log2FC)
```

```
## # A tibble: 102 x 7
## # Groups:   cluster [30]
##        p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##        <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 0              2.81  0.309 0.074 0         0       KLRB1
##  2 0              1.14  0.285 0.103 0         1       SYTL3
##  3 0              0.418 0.289 0.069 0         1       GZMM
##  4 2.36e- 18    Inf     0.391 0.349 7.01e- 14 2       IFI27
##  5 1.09e-284    416.    0.57  0.383 3.24e-280 2       ISG15
##  6 0            Inf     0.366 0.008 0         3       SFRP4
##  7 0            Inf     0.954 0.044 0         3       LUM
##  8 0            Inf     0.978 0.059 0         3       DCN
##  9 0            Inf     0.619 0.053 0         3       POSTN
## 10 0            Inf     0.373 0.029 0         3       MMP11
## # i 92 more rows
```

Saving the markers:

```
saveRDS(seurat_obj.markers, file = "seurat_obj.markers.rds")
```

Conlusion

I was able to identify cellular clusters in scRNA seq atlas of breast tumors. For each cluster, the corresponding marker genes were also identified. The question of whether these findings are biologically valid or not remains to be explored. For example, one needs to explore if the clusters identified here are representing a distinct cell type with their corresponding marker genes.

Additionally, I made multiple decisions about values of parameters that have significantly affected the results. As such, the identified clusters might not represent the real structure of the data and existing clusters (cell types) within it.