scRNAseq analysis

Li Sun

6/28/2020

Data source

Original paper

Anoop P. Patel, Itay Tirosh, John J. Trombetta, Alex K. Shalek, Shawn M. Gillespie, Hiroaki Wakimoto, Daniel P. Cahill, Brian V. Nahed, William T. Curry, Robert L. Martuza, David N. Louis, Orit Rozenblatt-Rosen, Mario L. Suvà, Aviv Regev, and Bradley E. Bernstein Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma Science. 2014 Jun 20; 344(6190): 1396–1401. doi: 10.1126/science.1254257.

Original abstract

"Human cancers are complex ecosystems composed of cells with distinct phenotypes, genotypes and epigenetic states, but current models do not adequately reflect tumor composition in patients. We used single cell RNA-seq to profile 430 cells from five primary glioblastomas, which we found to be inherently variable in their expression of diverse transcriptional programs related to oncogenic signaling, proliferation, complement/immune response and hypoxia. We also observed a continuum of stemness-related expression states that enabled us to identify putative regulators of stemness in vivo. Finally, we show that established glioblastoma subtype classifiers are variably expressed across individual cells within a tumor and demonstrate the potential prognostic implications of such intratumoral heterogeneity. Thus, we reveal previously unappreciated heterogeneity in diverse regulatory programs central to glioblastoma biology, prognosis, and therapy."

Single cell sample preparation

Single cells transcriptome were generated using cell sorting and SMART-Seq from 6 freshly dissected human glioblastomas (MGH26,264,28, 29, 30, 31). MGH26, MGH30, and MGH31 have significant expression of EGFR. 96–192 cells were generated from each tumor sample and in total 672 cells were profiled. After filtering out low quality cells and genes with less coverage, 430 cells with around 6000 genes were kept. Count matrix is downloaded from here. This matrix is log2(TPM + 1). Seurat starts with count data or TPM.

Purpose

Purpose of this project is to re-analyze the single cell RNAseq data using Seurat workflow.

```
ltmpf <- 'Glioblastoma_expressed_genes.txt'
l2tmp <- read.table(ltmpf, sep = '\t', header = TRUE, row.names = 1)
# Convert log2(TPM + 1) to TPM
count <- ceiling(2**l2tmp - 1)
# Create Seurat object
so <- CreateSeuratObject(count, project = 'Glioblastoma')
so
```

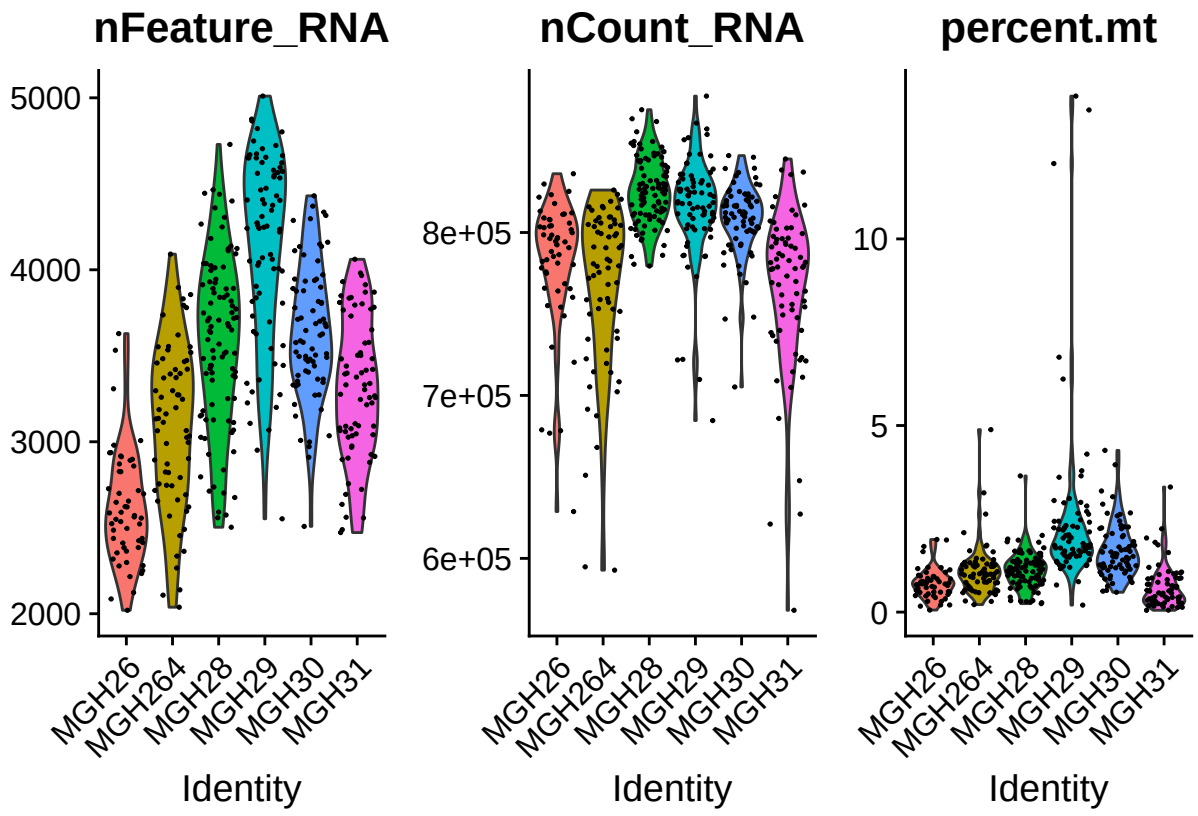
```
## An object of class Seurat
```

```
## 5948 features across 430 samples within 1 assay
## Active assay: RNA (5948 features, 0 variable features)
```

Preprocessing

Mitocondria gene percentage. High mt-genes indicates dying or low quality celss.

```
so[['percent.mt']] <- PercentageFeatureSet(so, pattern = '^MT-')
VlnPlot(so, features = c("nFeature_RNA", "nCount_RNA", "percent.mt"), ncol = 3)
```



Filtering

Remove cells with more than 5% mt gene. nFeature seems reasonable and no filtering needed. The count slot and data slot at this point of time store the same 'sparse matrix' in dgCMatrix format.

```
so <- subset(so, subset = percent.mt <5)
```

Normalizing the data

LogNormalize TPM. Scale factor = 10000. This step update data slot of seurat object. Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. This is then natural-log transformed using log1p.

```
so <- NormalizeData(so, normalization.method = 'LogNormalize', scale.factor = 10000)
```

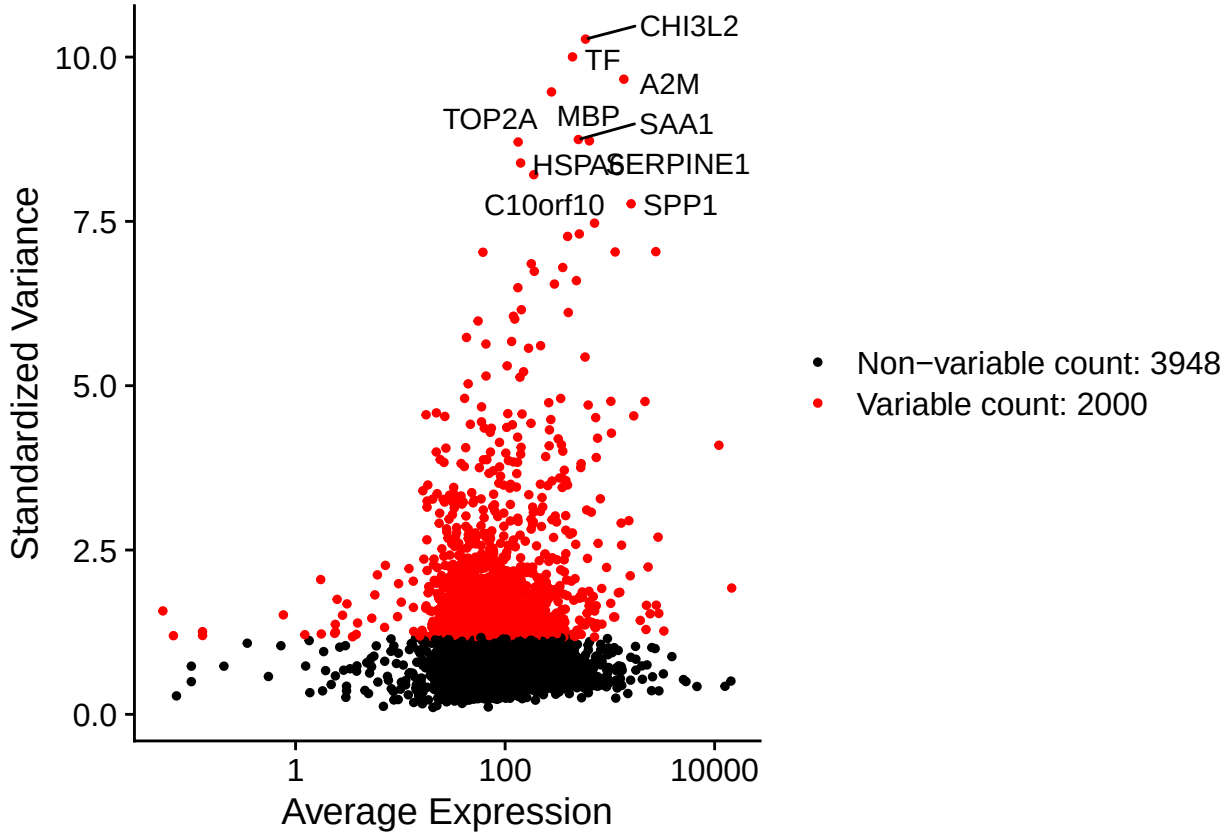
Identifying highly variable features (feature selection)

This step is lowering number of features to a much smaller number while maintain most of the variations. To choose most informative/variable features/genes, we cannot directly order genes according to their variation accross cells. because this highly expressed genes normally have much larger variance comparing to lowly expressed genes. Thus the relationship between expression and variantion must be taken into account. We use vst method described here.

```
so <- FindVariableFeatures(so, selection.method = 'vst', nfeatures = 2000)
top10 <- head(VariableFeatures(so), 10)
plot1 <- VariableFeaturePlot(so)
plot2 <- LabelPoints(plot = plot1, points = top10, repel = TRUE)
```

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

```
plot2
```



Scaling data for dimension reduction

Make sure columns (each feature) has mean 0 and variance 1. Scaled data is in normal matrix in slot @data@scale.data.

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

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

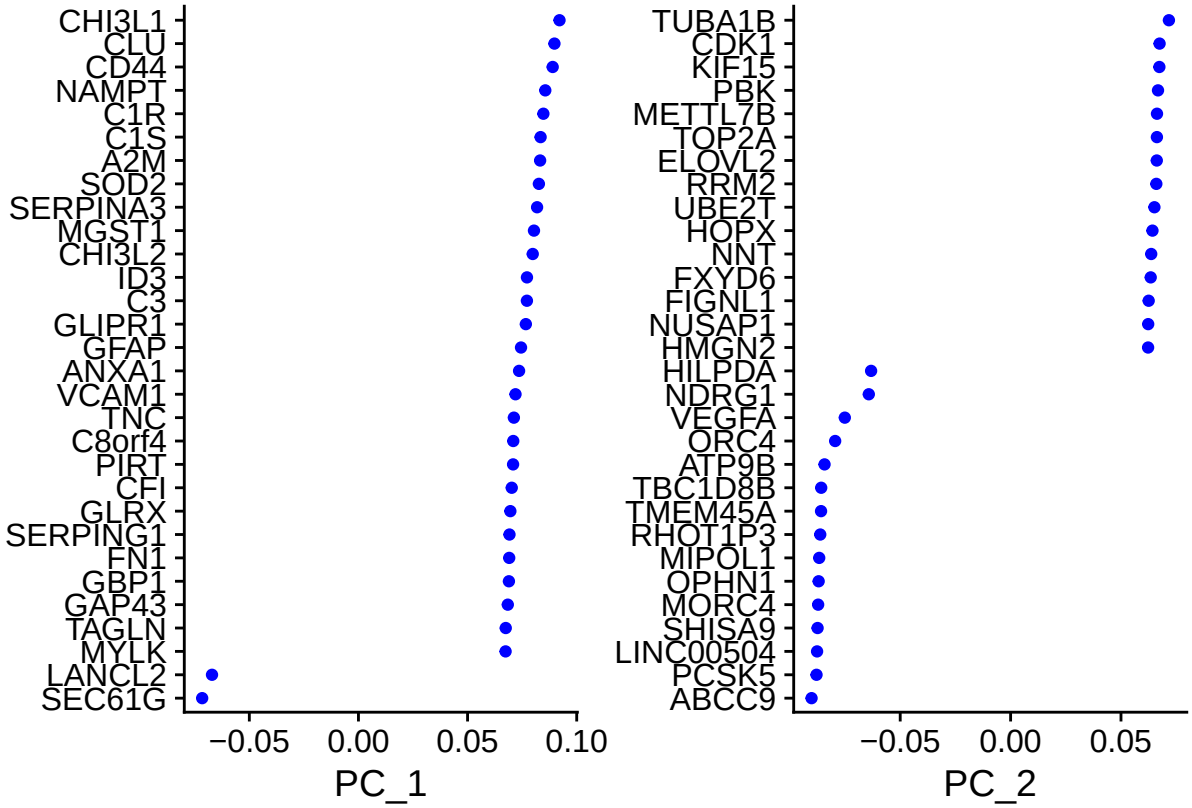
Linear dimensional reduction

PCA is used to reduce dimension of our normalized count data, only using 2000 top variable features. PCA results are stored in @reductions. VizDimLoadings is showing gene with highest absolute loadings with each PC. DimPlot is showing first two PCs in scatter plot with coloring according to @meta.data$orig.ident. DimHeatmap is ploted as following: 1. For each PC, the top nFeatures genes with largest absolute loadings (half positive, half negative, when nFeatures is odd, it will only show nfeatures -1) were selected from @pca@feature.loadings. 2. For the same PC, the top cells with largest embedding of that PC were chose. And then the scaled expression data of chosen genes and cells were ploted as Heatmap.

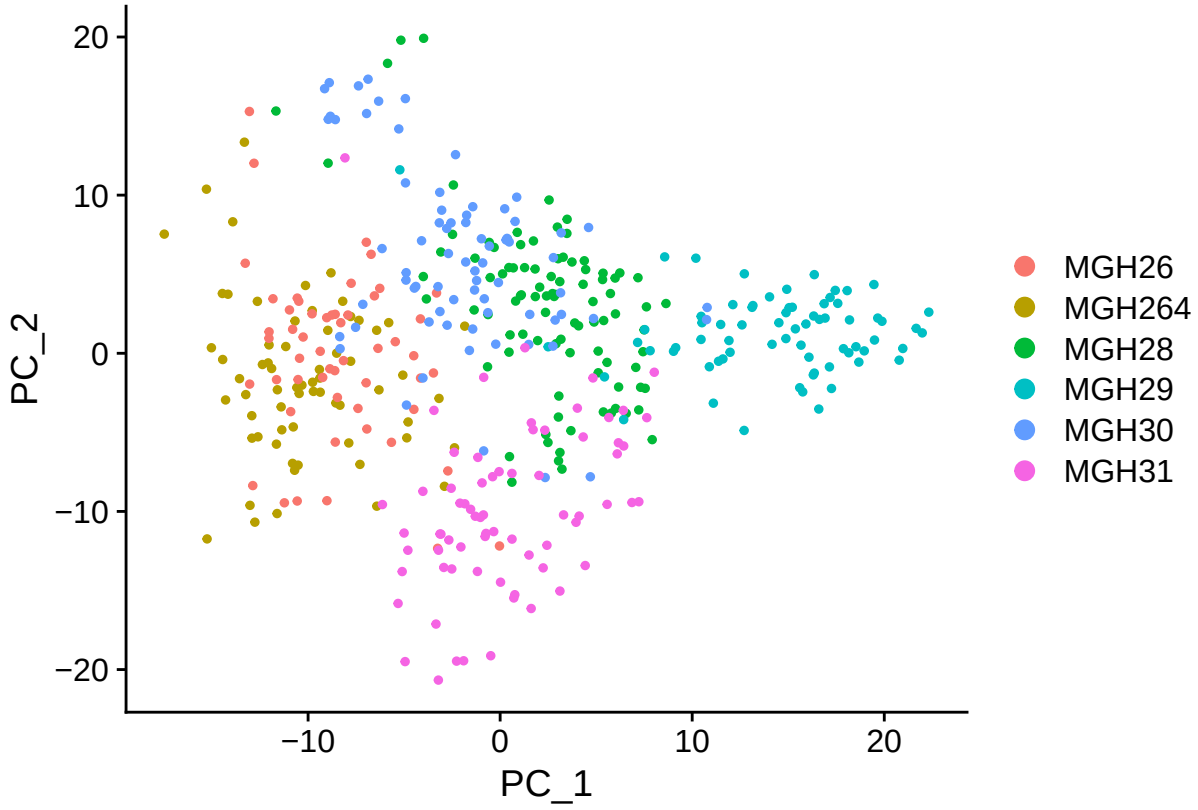
```
so <- RunPCA(so, features = VariableFeatures(object = so))
```

```
## PC_ 1
## Positive:  CHI3L1, CLU, CD44, NAMPT, C1R, C1S, A2M, SOD2, SERPINA3, MGST1
##     CHI3L2, ID3, C3, GLIPR1, GFAP, ANXA1, VCAM1, TNC, C8orf4, PIRT
##     CFI, GLRX, SERPING1, FN1, GBP1, GAP43, TAGLN, MYLK, F3, GEM
## Negative:  SEC61G, LANCL2, EGFR, MIR3917, TCF12, BEX1, HMGB1, MAP2, NRXN1, ARL4A
##     TRIB2, LINC00504, SCG3, PCMTD2, MIPOL1, SHISA9, CD82, GRIA2, TBC1D8B, CDK6
##     ATP9B, TM4SF1, RHOT1P3, HIST1H4C, MORC4, ABCC9, TMEM45A, HMGN2, CHD7, POSTN
## PC_ 2
## Positive:  TUBA1B, CDK1, KIF15, PBK, METTL7B, TOP2A, ELOVL2, RRM2, UBE2T, HOPX
##     NNT, FXYD6, FIGNL1, NUSAP1, HMGN2, SLC1A3, AURKB, KPNA2, FANCI, BIRC5
##     CCNB2, KIF4A, MLF1IP, KIF22, TPX2, NCAPG2, HIF1A, OXCT1, NLGN3, CKAP2
## Negative:  ABCC9, PCSK5, LINC00504, SHISA9, MORC4, OPHN1, MIPOL1, RHOT1P3, TMEM45A, TBC1D8B
##     ATP9B, ORC4, VEGFA, NDRG1, HILPDA, FAM197Y6, TF, ZNF395, IGFBP5, UGT8
##     CLDN11, GBP4, ANKRD28, MIR143HG, GLDN, FTL, ENO2, TTN, CCDC141, RP9P
## PC_ 3
## Positive:  SCG2, FABP5, LGALS3, PGM2L1, BCAT1, MT2A, CPNE8, AKAP12, CA12, TIMP1
##     IDS, CTSC, SPP1, IGFBP3, NRP1, SLC2A3, CTSB, COL1A2, EFEMP1, MT1X
##     SERPINE1, ARSJ, ARRDC3, SMYD2, METTL7B, GBE1, IQGAP1, CLDN12, CXCL14, BSCL2
## Negative:  SCRG1, BCAN, S100B, C1orf61, FXYD6, VCAN, LHFPL3, AQP4, FAM107A, C21orf62
##     PMP2, TUBB2B, PSAT1, ID3, C3orf70, MARCKS, LIX1, SLC38A3, MASP1, AHCYL1
##     NKAIN4, PIRT, LANCL2, SEC61G, NCALD, PGAM2, MYBPC1, CHI3L2, GFAP, SIRT2
## PC_ 4
## Positive:  TIMP1, AGT, SCG2, POSTN, FABP7, FKBP1A, TSPAN7, EFEMP1, SEC61G, IL1RAP
##     SMYD2, BMP5, IGFBP3, GAP43, DUSP6, LANCL2, COL1A2, SPRY4, PEG10, METTL7B
##     CPVL, FABP5, CECR1, CNR1, B2M, CNOT2, CTSB, C1orf61, SAT1, CD68
## Negative:  TOP2A, KIF15, NUSAP1, SPAG5, PBK, KIF4A, CDK1, RRM2, AURKB, BIRC5
##     FANCD2, TPX2, DHFR, BUB1, UBE2T, CCNB2, XRCC2, CENPF, KIF20A, MLF1IP
##     POLQ, NCAPG2, CLSPN, RACGAP1, DSN1, GINS1, FANCI, CCNB1, FIGNL1, KIAA0101
## PC_ 5
## Positive:  CNDP1, KLK6, HHATL, CLDN11, SPOCK3, RNASE1, ENPP2, PLP1, PTGDS, CNTN2
##     SLC31A2, SEPP1, CNTNAP4, SEPT4, MBP, TF, GLDN, QDPR, ERMN, PIP4K2A
##     SLC24A2, TMEM144, MTUS1, PLEKHB1, PCDH9, CNP, FGF1, KCNMB4, HEPN1, ABCA8
## Negative:  HILPDA, H2AFZ, PGK1, VEGFA, MCM7, ADM, TREM1, SEC61G, IGFBP5, ID2
##     NRN1, PCNA, ERO1L, HMGB2, HDAC2, AKAP12, GLIPR1, C8orf4, KIAA0101, RRM2
##     HIST1H4C, MAD2L1, PTTG1, CENPF, SOD2, PFKP, BIRC5, SHMT2, KIF4A, GBE1
```

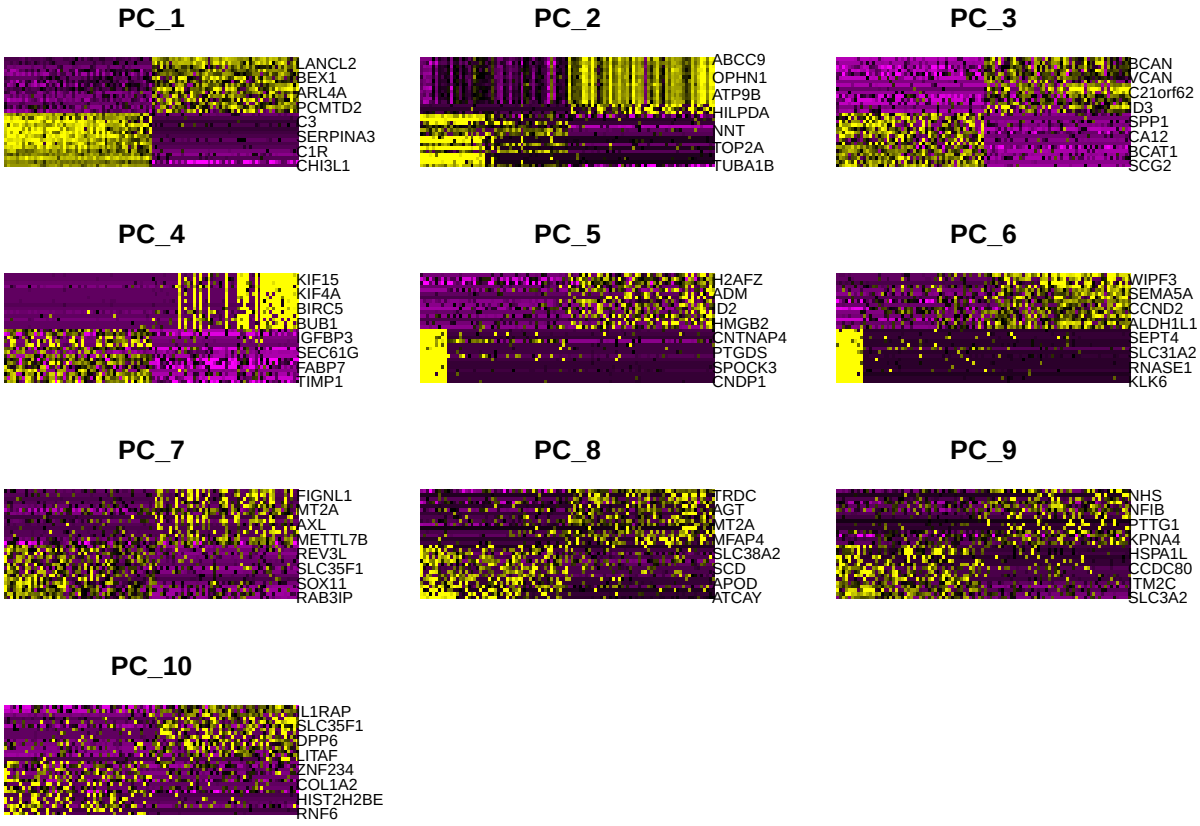
```
# Visualizing PCA results
VizDimLoadings(so, dims = 1:2, reduction = 'pca')
```




```
DimPlot(so, reduction = 'pca')
```




```
DimHeatmap(so, cells=100, dims = 1:10, balanced = TRUE)
```



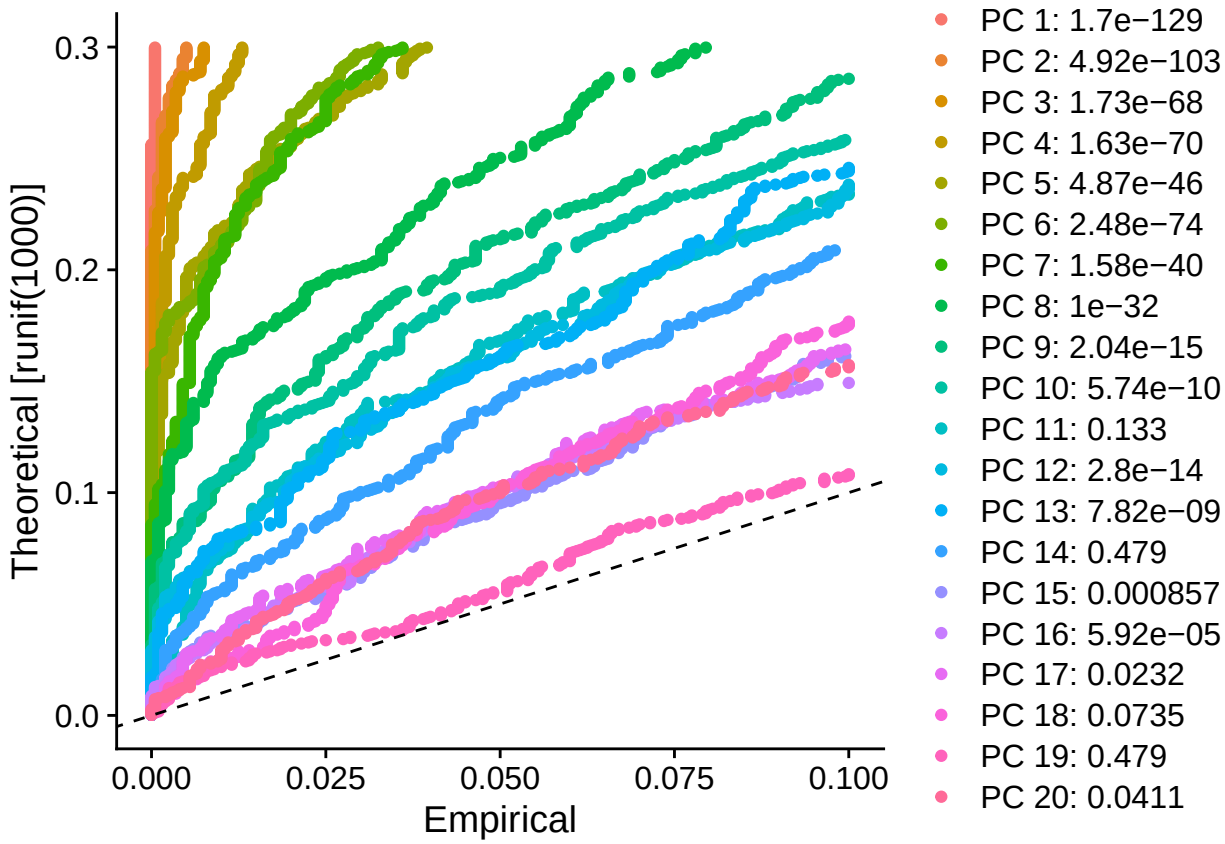
Determine Dimensionality of the data

To find the dimensionality or choose the optimal number of PCs, we could use JackStraw test. JackStraw is a method to test significance of association between each gene to any subset of PCs. Directly calculated association between gene expression across cells and any or any subset of "unit PCs" (the row vector of matrix V transpose from SVD) using F test (ANOVA or nested linear models) has 2 problem. Firstly, matrix V transpose is an noisy estimation of true latent variables. Secondly, PCA picks any variation in the dataset, including the noise, so it is overfitting. To deal with these problems, JackStraw method took a small subset of the data (1% is the default in Seurat), permute each selected rows. And the recalculate PCA and F statistics of the permuted rows. This process is repeated until sufficient number of null F statistics are obtained. Then all the F statistics of original un-permuted rows/genes were compared to this null distribution to get p values. Note, if you are permuting s rows each time, and for B times. Then you are estimating null distribution using s x B values. If you fix s x B = 10000, using s = 1 and B = 10000 gives you best estimation but slowest computation. On the other hand, if you use s = 10000 and B = 1, you get fastest computation time but most conservative biased estimation. JaskStraw results are stored in @reductions$pca@jackstraw. By using this method, you can get all the gene p values on each PC. In the regard of choosing number of top PCs for following analysis. We could choose the PCs with most low p values. This could be visualized by plotting QQ plot of gene wise p-values compared with a uniform distribution. ElbowPlot is a traditional method to visualize importance of each top PCs. It use the @stdev values in @reductions$pca. In general, determing dimensionality is subjective. Using more dimensions could help identifying rarer clusters, but it could also bring more noisy. However, author of Seurat believes higher dimension might be less worse than lower dimension. And it is a good practice to try several different dimensions in the following analysis and compare, if possible.

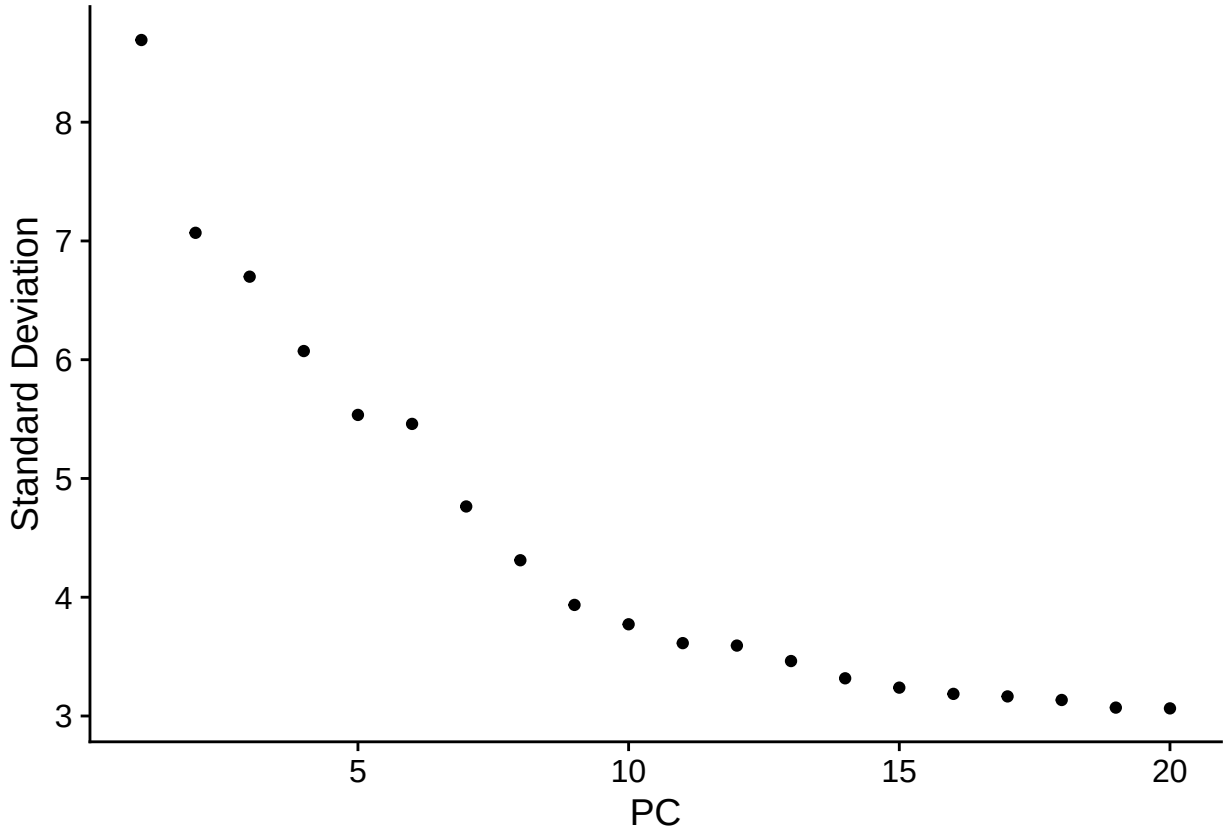
```
so <- JackStraw(so)
so <- ScoreJackStraw(so, dims = 1:20)
```

```
JackStrawPlot(so, dims=1:20)
```

```
## Warning: Removed 30418 rows containing missing values (geom_point).
```




```
ElbowPlot(so)
```



Choose 10 components.

Cluster cells

After number of PCs chosen, for the following analysis, we use the 10 values to represent each cell. First, we want cluster all cells using these 10 PCs.

Following process should be based on here. But it seems different. Clustering is done using Shared Nearest Neighbor (SNN). SNN is a secondary NN algorithm based on KNN. When KNN is done, SNN recalculate similarity between pair of points using their shared neighbors, as such: 1. if there is no shared neighbor, there is no similarity/connection between two points. 2. if there is shared neighbor, difference between k and the highest averaged ranking of the shared neighbors. Based on this sparse similarity matrix/graph, quasi-clique (some clusters could be overlapping) is found using some algorithm. Then significantly overlapping quasi-cliques are merged. Points within more than 1 clusters were reassinged to the 'best' cluster.

```
so <- FindNeighbors(so, dims = 1:10)
```

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

```
## Computing SNN
```

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

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 425
## Number of edges: 11586
##
## Running Louvain algorithm...
```

```
## Maximum modularity in 10 random starts: 0.8558
## Number of communities: 6
## Elapsed time: 0 seconds
```

```
library(igraph)
```

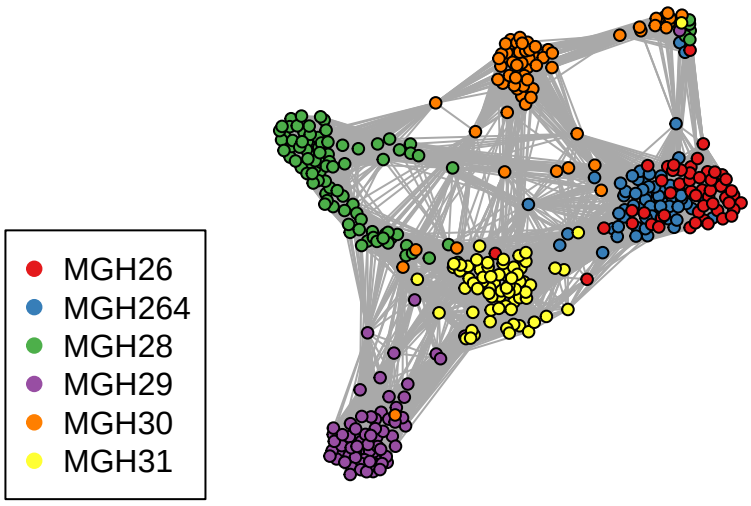
```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

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

```
## The following object is masked from 'package:base':
##
##     union
```

```
g = graph.adjacency(as.matrix(so@graphs$RNA_snn), mode="undirected",weighted=TRUE,diag=FALSE)
# Simplfy the adjacency object
g <- simplify(g, remove.multiple=TRUE, remove.loops=TRUE)
cellgroup = factor(sapply(rownames(so@graphs$RNA_snn), function(x)strsplit(x,'_')[[1]][1]))
library(RColorBrewer)
coul  <- brewer.pal(6, "Set1")
plot(g, vertex.size=5, vertex.color=coul[as.numeric(cellgroup)], vertex.label='',main = 'Visualize SNN')
legend("bottomleft", legend=levels(cellgroup), col = coul, pch=19)
```



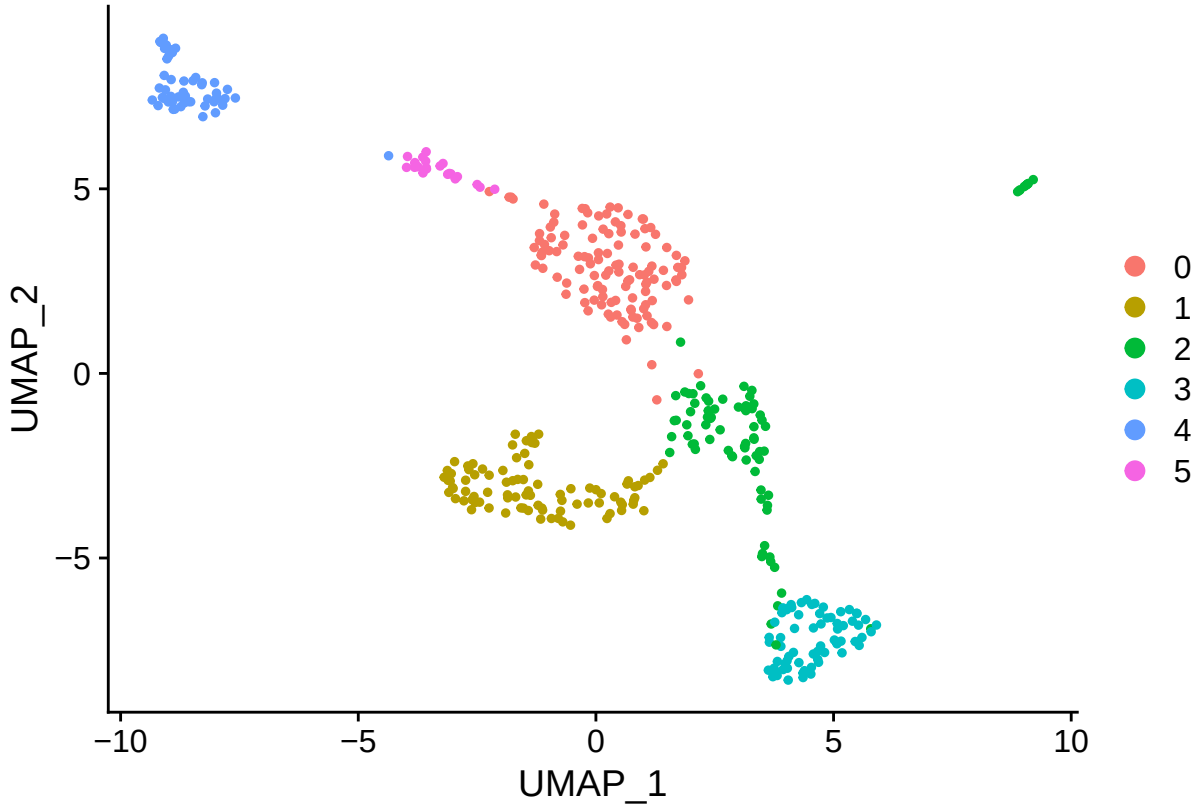
Non-linear dimension reduction

There are multiple non-linear dimension reduction method. Such as tSNE, UMAP. These method are mainly used to visualize cell clusters in 2D space.These methods are desired because of superior visualization character in 2D space. Such as tSNE, which conserve local relationships while increases distances of larger distances. This is a desired behavior for visualizing high dimensional data in low dimension. Also, the clusters showed are normally coincident with clusters we get using SNN. (I don't understand yet why these methods were not used in clustering?) Seurat author suggest using the same PCs as input as we did for above clustering process.

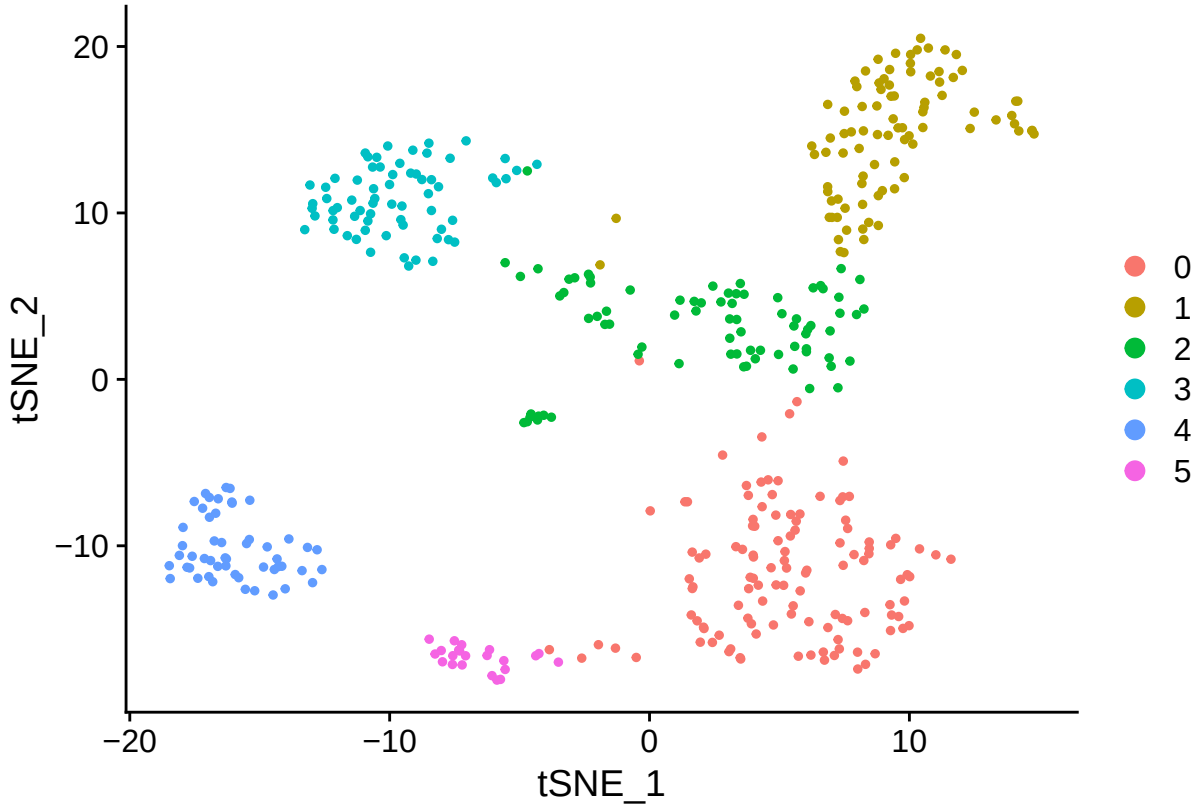
```
so <- RunUMAP(so, dims = 1:10)
```

```
## 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
## 23:20:59 UMAP embedding parameters a = 0.9922 b = 1.112
## 23:20:59 Read 425 rows and found 10 numeric columns
## 23:20:59 Using Annoy for neighbor search, n_neighbors = 30
## 23:20:59 Building Annoy index with metric = cosine, n_trees = 50
## 0%   10   20   30   40   50   60   70   80   90   100%
## [----|----|----|----|----|----|----|----|----|----|
## **************************************************|
## 23:20:59 Writing NN index file to temp file C:\Users\sunli\AppData\Local\Temp\RtmpMvw8ax\fileb1943de
## 23:20:59 Searching Annoy index using 1 thread, search_k = 3000
## 23:20:59 Annoy recall = 100%
## 23:21:00 Commencing smooth kNN distance calibration using 1 thread
## 23:21:00 Initializing from normalized Laplacian + noise
## 23:21:00 Commencing optimization for 500 epochs, with 14790 positive edges
## 23:21:02 Optimization finished
```

```
so <- RunTSNE(so, dims = 1:10)
DimPlot(so, reduction = "umap")
```




```
DimPlot(so, reduction = "tsne")
```



Finding marker features of each cluster

Differential expression is always 1 vs another, or pairwise. Thus, we will do a series of comparisons among all clusters. In Seurat, this is done using function FindAllMarkers.

```
so.markers <- FindAllMarkers(so, 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
```

```
so.markers %>% group_by(cluster) %>% top_n(n = 2, wt = avg_log2FC)
```

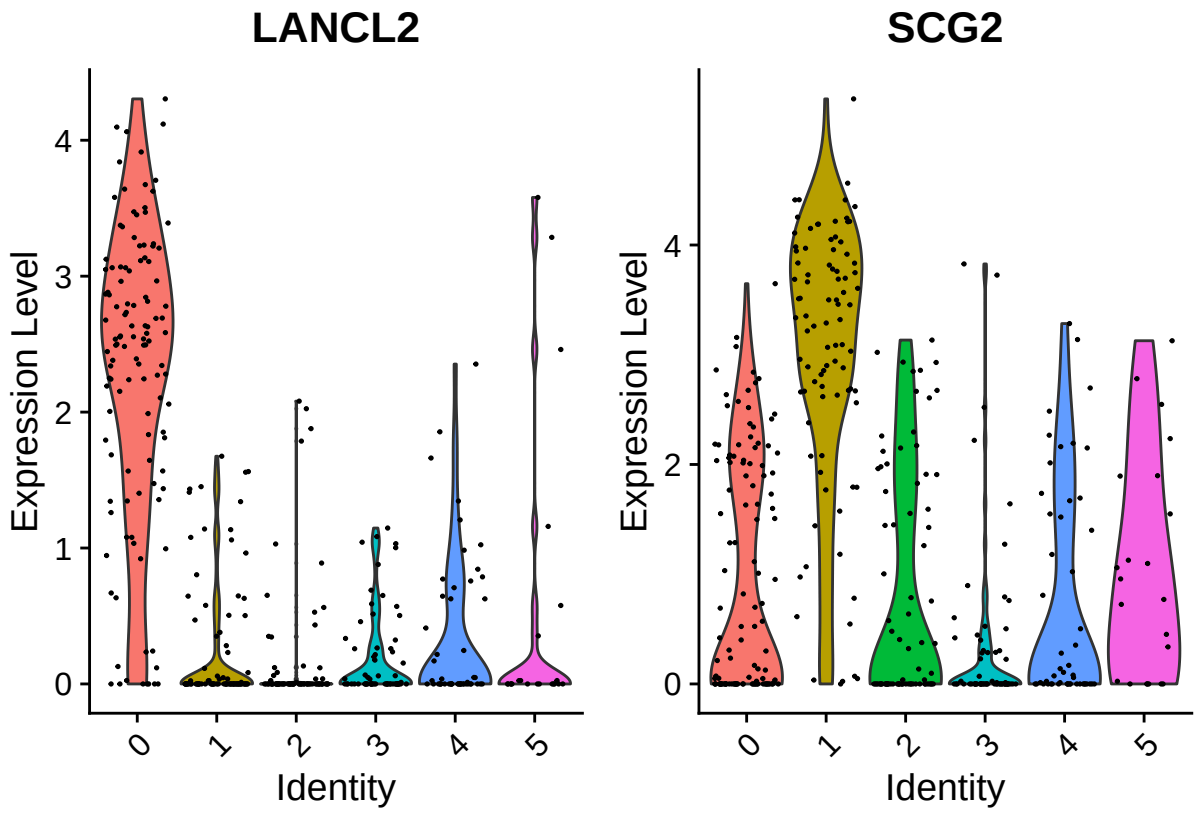
```
## Registered S3 method overwritten by 'cli':
##   method     from
##   print.boxx spatstat
```

```
## # A tibble: 12 x 7
## # Groups:   cluster [6]
##       p_val avg_log2FC pct.1 pct.2 p_val_adj cluster gene
##       <dbl>      <dbl> <dbl> <dbl>     <dbl> <fct>   <chr>
##  1 2.11e-47       3.23 0.941 0.384  1.26e-43 0       LANCL2
```

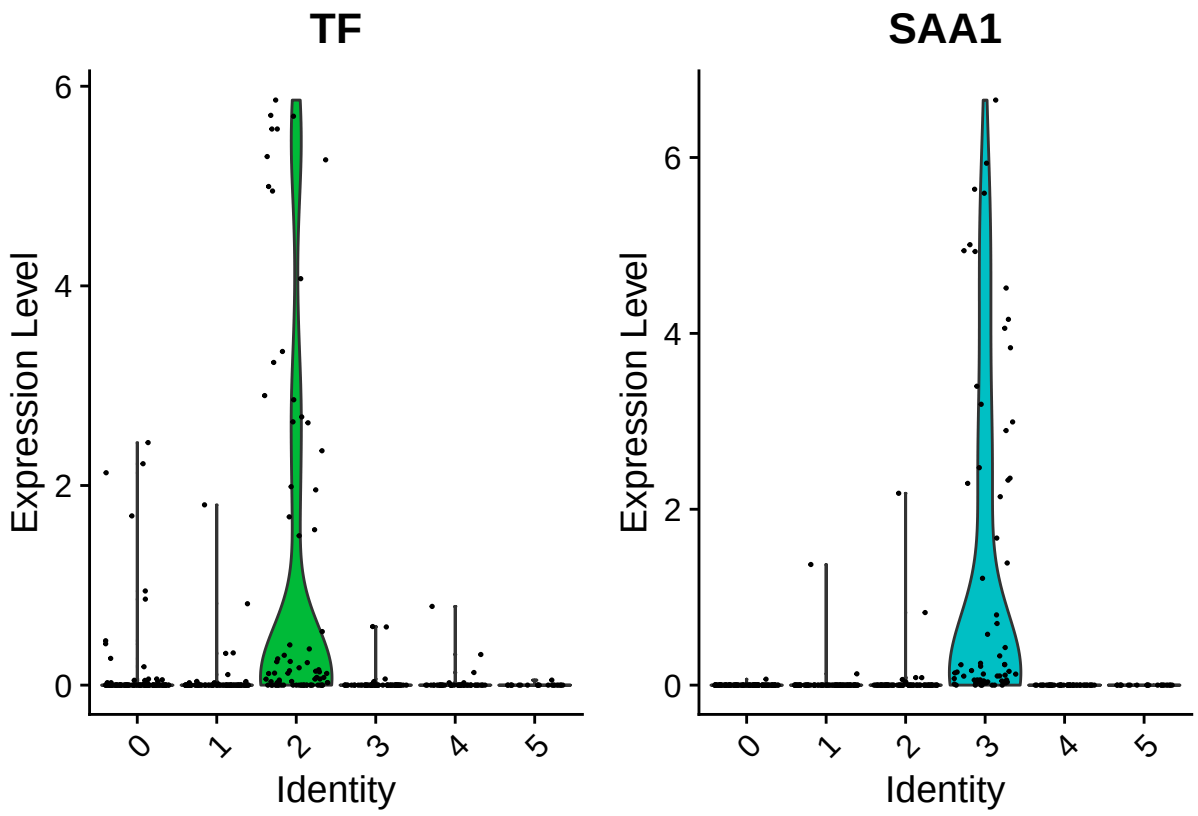
```
##  2 1.54e-46       4.14 0.966 0.86   9.14e-43 0       SEC61G
##  3 4.58e-34       3.03 0.989 0.562  2.72e-30 1       SCG2
##  4 1.24e-24       3.48 0.978 0.75   7.37e-21 1       AGT
##  5 5.17e-28       4.77 0.658 0.118  3.08e-24 2       TF
##  6 1.02e- 4       3.84 0.557 0.303  6.06e- 1 2       PLP1
##  7 5.12e-70       5.25 0.894 0.025  3.04e-66 3       SAA1
##  8 6.46e-60       4.80 1     0.109  3.85e-56 3       CHI3L2
##  9 7.71e-32       2.51 1     0.249  4.58e-28 4       FIGNL1
## 10 6.36e-30       3.11 1     0.457  3.78e-26 4       ATP1A2
## 11 3.58e-28       4.86 1     0.171  2.13e-24 5       TOP2A
## 12 1.02e-20       3.71 0.955 0.213  6.04e-17 5       CCNB1
```

Visualizing feature expressions:

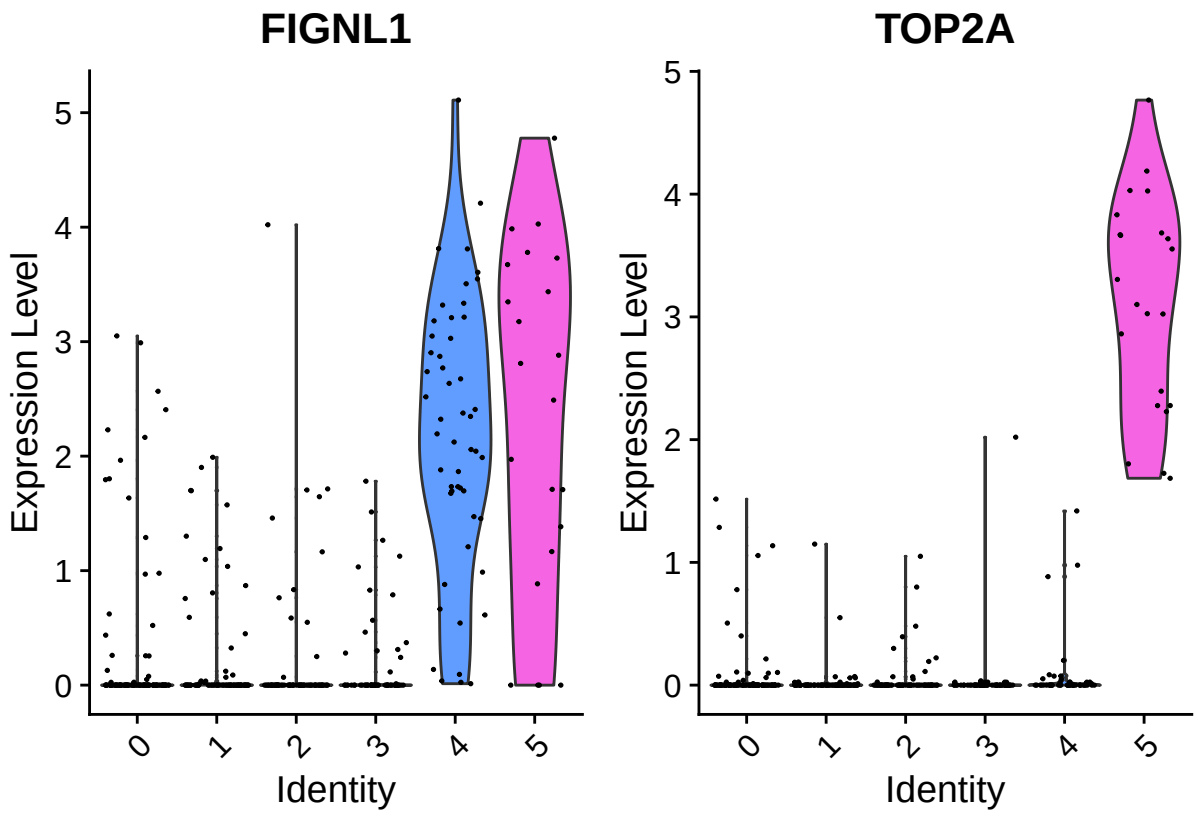
```
VlnPlot(so, features = c("LANCL2", "SCG2"))
```




```
VlnPlot(so, features = c("TF", "SAA1"))
```

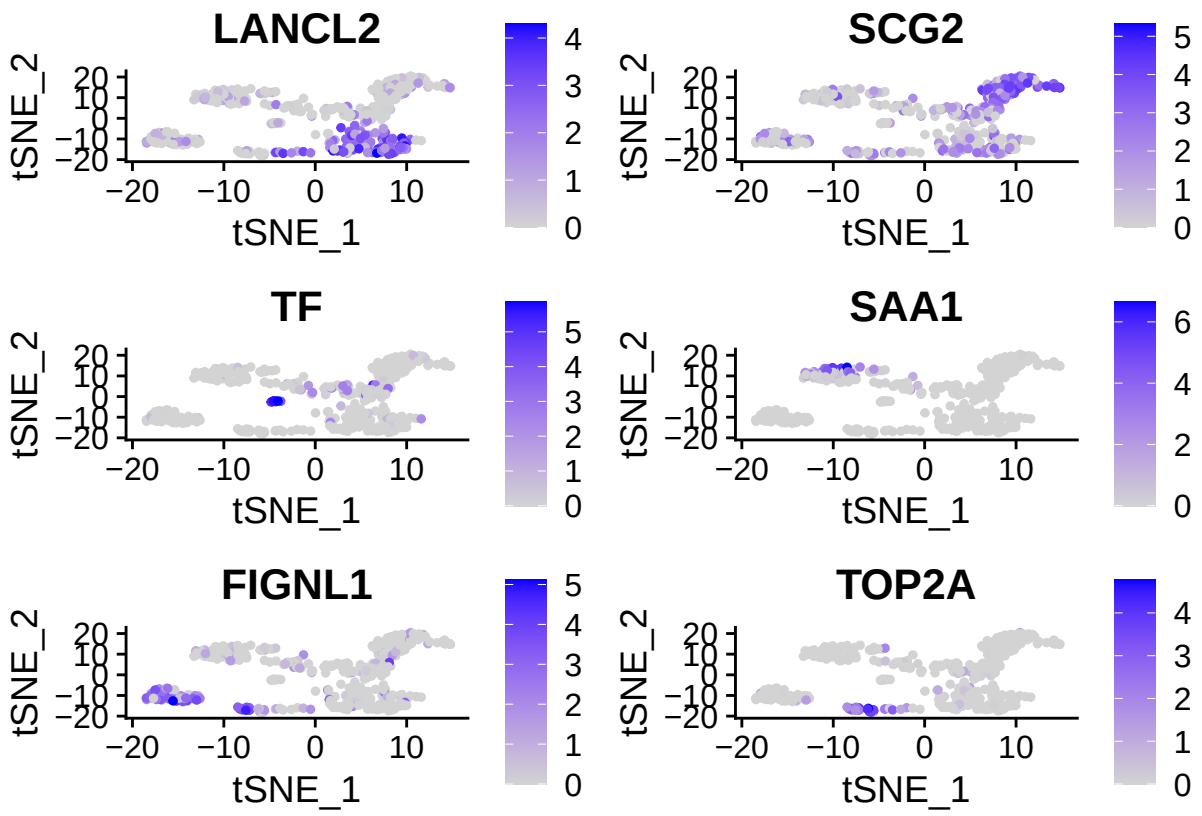



```
VlnPlot(so, features = c("FIGNL1", "TOP2A"))
```



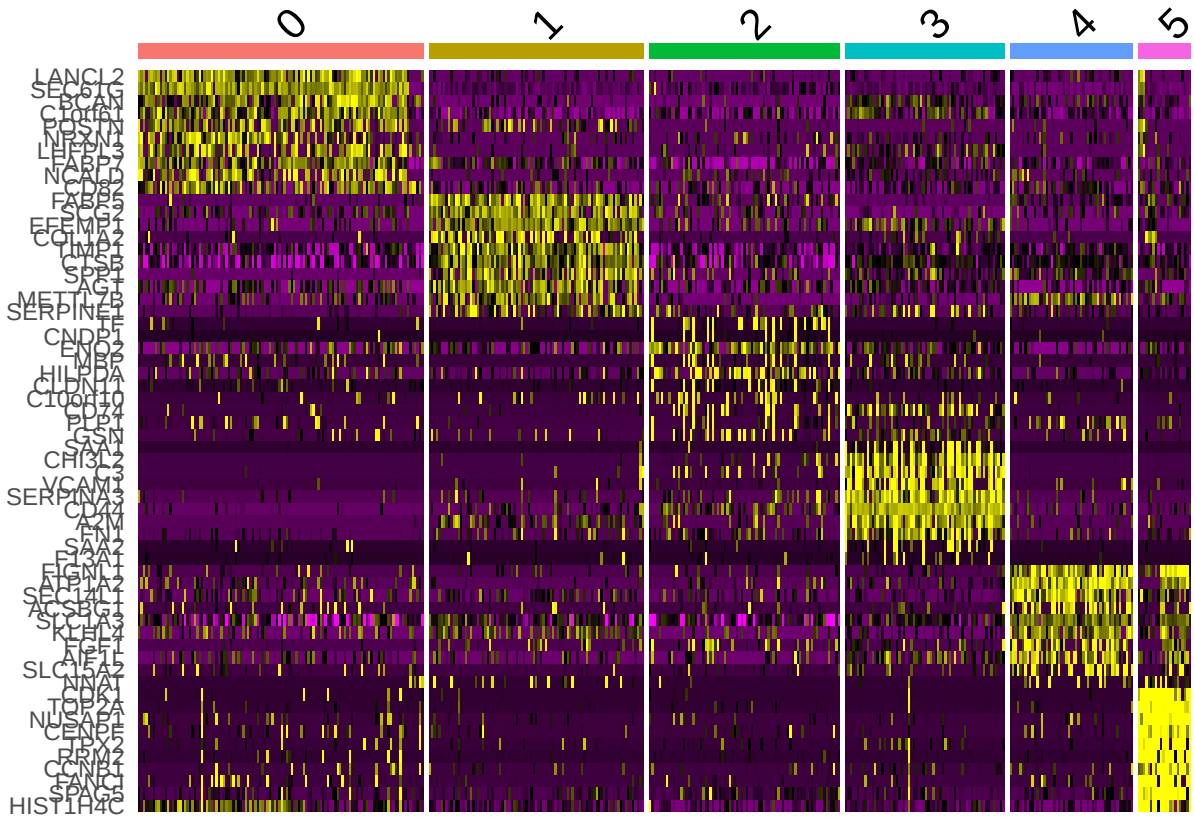
Visualizing feature expressions on clustering:

```
FeaturePlot(so, features = c("LANCL2", "SCG2", "TF", "SAA1", "FIGNL1", "TOP2A"), reduction='tsne')
```



Plot top 10 markers from each cluster

```
top10 <- so.markers %>% group_by(cluster) %>% top_n(n = 10, wt = avg_log2FC)
DoHeatmap(so, features = top10$gene) + NoLegend()
```



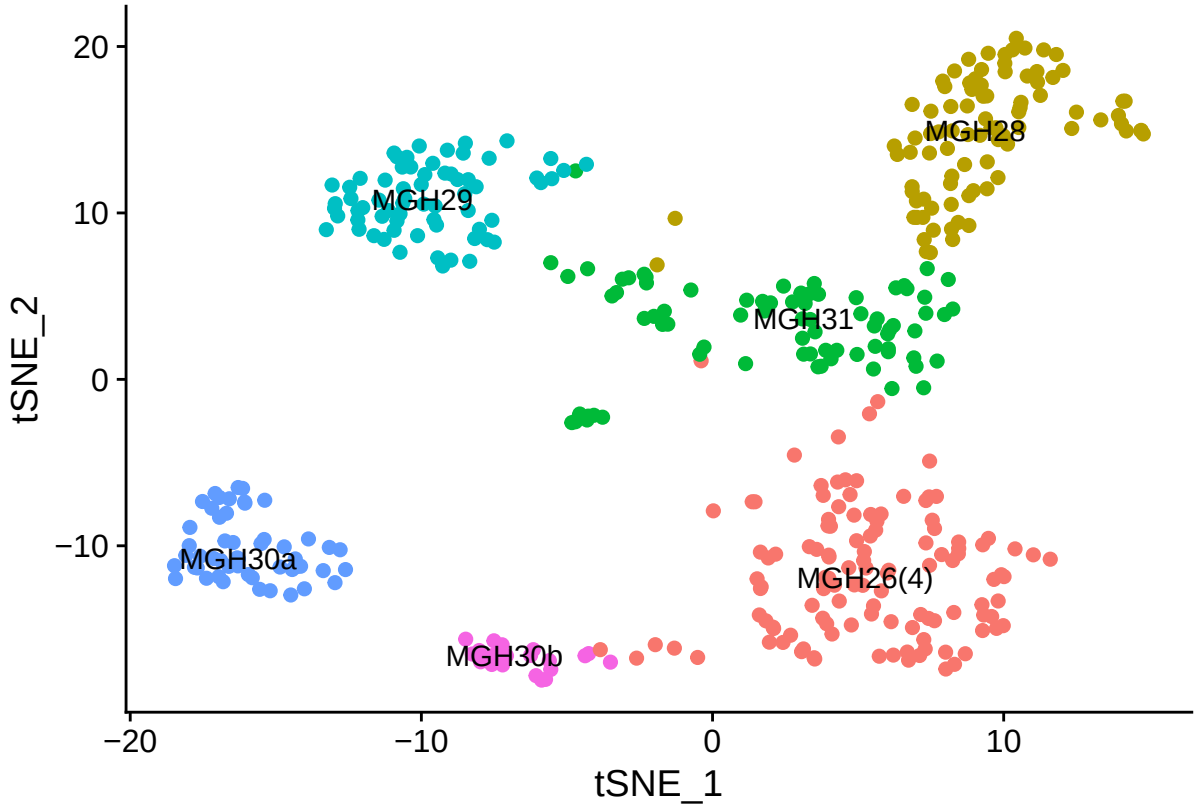
Assigning cell type identity to clusters

This is knowledge-based process. Cannot be done solely by Seurat.But in this dataset, we have already known the cell types (cell origin in our case). So let's see how was our clustering job.

```
table(so@meta.data$orig.ident, so@active.ident)
```

```
##
##            0  1  2  3  4  5
##   MGH26   50  0  2  0  0  1
##   MGH264 62  0  1  0  0  2
##   MGH28    0 86  3  0  0  5
##   MGH29    0  0  4 65  0  1
##   MGH30    5  2  2  1 51 12
##   MGH31    1  1 67  0  0  1
```

```
# It looks good
# lets change give the SNN found groups a name based on above table results
new.cluster.ids <- c('MGH26(4)', 'MGH28', 'MGH31', 'MGH29', 'MGH30a', 'MGH30b')
names(new.cluster.ids) <- levels(so)
so <- RenameIdents(so, new.cluster.ids)
DimPlot(so, reduction = "tsne", label=TRUE, pt.size = 2) + NoLegend()
```



This is a run-through of most Seurat workflow of processing and analyzing single cell RNAseq dataset from TPM count matrix. More to come (slingshot)!