Using TF-Gene Target network to perform clustering analysis

Weixu Wang

Preprocessing the dataset

Before running ICAnetTF, we first need to preprocess the dataset. Here I used the preprocessing guideline provided by SCENIC [1].

```
setwd("/mnt/data2/wangweixu/mouseData")
library(doParallel)
```

```
## Loading required package: foreach
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(Seurat)
library(cowplot)
```

```
##
## ************************************************************
```

```
## Note: As of version 1.0.0, cowplot does not change the
```

```
##   default ggplot2 theme anymore. To recover the previous
```

```
##   behavior, execute:
##   theme_set(theme_cowplot())
```

```
## ************************************************************
```

```
library(ica)
library(ICAnet)

mouseData <- read.table("/mnt/data4/weixu/MCTA/immuneAtlas/Mouse/GSE60361_C1-3005-Expression.txt",header=TRUE,row.names=1)
cellid <- read.csv("/mnt/data4/weixu/MCTA/immuneAtlas/Mouse/cellid.csv",header=TRUE)

mouseData <- as.matrix(mouseData)
mouseData <- mouseData[rowSums(mouseData)>90,]
##filtering the mouse dataset

mouseData <- CreateSeuratObject(counts = mouseData, project = "mouse.brain", min.cells = 0.01*ncol(mouseData))
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')
```

```
mouseData$celltype <- cellid[,3]
mouseData$batch <- rep("mouseData",ncol(mouseData))
mouseData <- NormalizeData(mouseData, normalization.method = "LogNormalize", scale.factor = 10000)
```

Setup the Motif-Gene network

In this tutorial, we will introduce how to incorporate TF-Gene target network to analyze the mouse brain dataset sequenced by Zeisel et al. There are 3005 single cells that were sequenced on the Illumina HiSeq 2000. The raw data can be accessed through GEO database with the accession number GSE60361. Different from SCENIC [1], ICAnet first creates a Motif-Gene network, then performs target enrichment analysis for each Motif on each ICA-component. Meanwhile, it also considers whether each motif-associated transcriptional factor is 'activated'. In this step, ICAnet creates Motif-Gene network based on Motif-gene association matrix provided by RcisTarget [2].

```
library(RcisTarget)
Motif_Net <- TF_Net_Generate("/mnt/data2/wangweixu/mm9-500bp-upstream-10species.mc8nr.feather",cutoff=1)
```

```
## Loading TF,motif annotation dataset...
## Trimming the interaction...
## Generating TF-Gene-Net...
## Done
```

We note that it may takes a long time to download the whole TF-motif database. For the users who want to implement our procedure quickly, we provided Motif_Net object generate from above functions, users could download it from https://github.com/WWXkenmo/ICAnet_external_data (https://github.com/WWXkenmo/ICAnet_external_data) , and load the data through following command

```
Motif_Net <- readRDS("/mnt/data2/wangweixu/mouseData/Motif_Net.RDS")
```

Important parameters are as follows:

- *cutoff*: This parameter is to select the appropriate target for each TF through performing Z-score transformation on column or row of the matrix, and those interactions which transformed value < cutoff

were removed. Therefore, an appropriate cutoff needs to be determined carefully. The default value is 1.

This procedure will generate a binary motif-gene relationship. Therefore, each motif has corresponding associate gene sets.

Running ICAnetTF using motif-gene network as input

Then, we run ICAnetTF to decompose gene expression matrix into a number of independent components. After running ICAnetTF, the Seurat obj will be assigned an assay “IcaNet_TF”, which stores TF-regulon activity matrix.

```
Ica.mouse <- ICAcomputing(mouseData,ICA.type="JADE",RMT=TRUE,two.stage=FALSE)
```

```
## [1] "batch 1 Indepdent Component Analysis"
## [1] "emmm...centering..."
```

```
## Centering data matrix
```

```
## [1] "Done Centering"
## [1] "Using RMT to estimate number of module"
```

```
## Loading required package: coop
```

```
## Loading required package: rARPACK
```

```
## [1] "RMT estimate 32 expression programm"
```

```
mouseData  <- RunICAnetTF(mouseData,Ica.mouse$ica.pooling,W.top.TFs=3,W.top.genes=2.5,aucMaxRank=600,Motif_Net=Motif_Net,TF_motif_annot=motifAnnotations_mgi_v8)
```

```
## Running on TF-Gene Network
## Running on the 1st component
## Identify 13 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 2st component
## Identify 6 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 3st component
## Identify 7 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 4st component
## Identify 10 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 5st component
## Identify 12 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 6st component
## Identify 15 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 7st component
## Identify 12 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 8st component
## Identify 12 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 9st component
## Identify 12 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 10st component
## Identify 19 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 11st component
## Identify 7 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 12st component
## Identify 4 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 13st component
## Identify 19 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 14st component
## Identify 18 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
```

```
## Running on the 15st component
## Identify 11 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 16st component
## Identify 6 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 17st component
## Identify 18 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 18st component
## Identify 14 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 19st component
## Identify 9 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 20st component
## Identify 11 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 21st component
## Identify 15 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 22st component
## Identify 15 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 23st component
## Identify 12 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 24st component
## Identify 7 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 25st component
## Identify 15 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 26st component
## Identify 21 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 27st component
## Identify 13 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 28st component
## Identify 15 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 29st component
## Identify 8 activated TF
```

```
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 30st component
## Identify 16 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 31st component
## Identify 18 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
##
## Running on the 32st component
## Identify 2 activated TF
## perform modularity test[1] "Starting Monte Carlo Runs"
## [1] "num:691"
```

```
## Quantiles for the number of genes detected by cell:
## (Non-detected genes are shuffled at the end of the ranking. Keep it in mind when choosing th
e threshold for calculating the AUC).
```

```
##     min      1%      5%     10%     50%    100%
##  748.00 1031.32 1410.40 1726.00 3605.00 8059.00
```

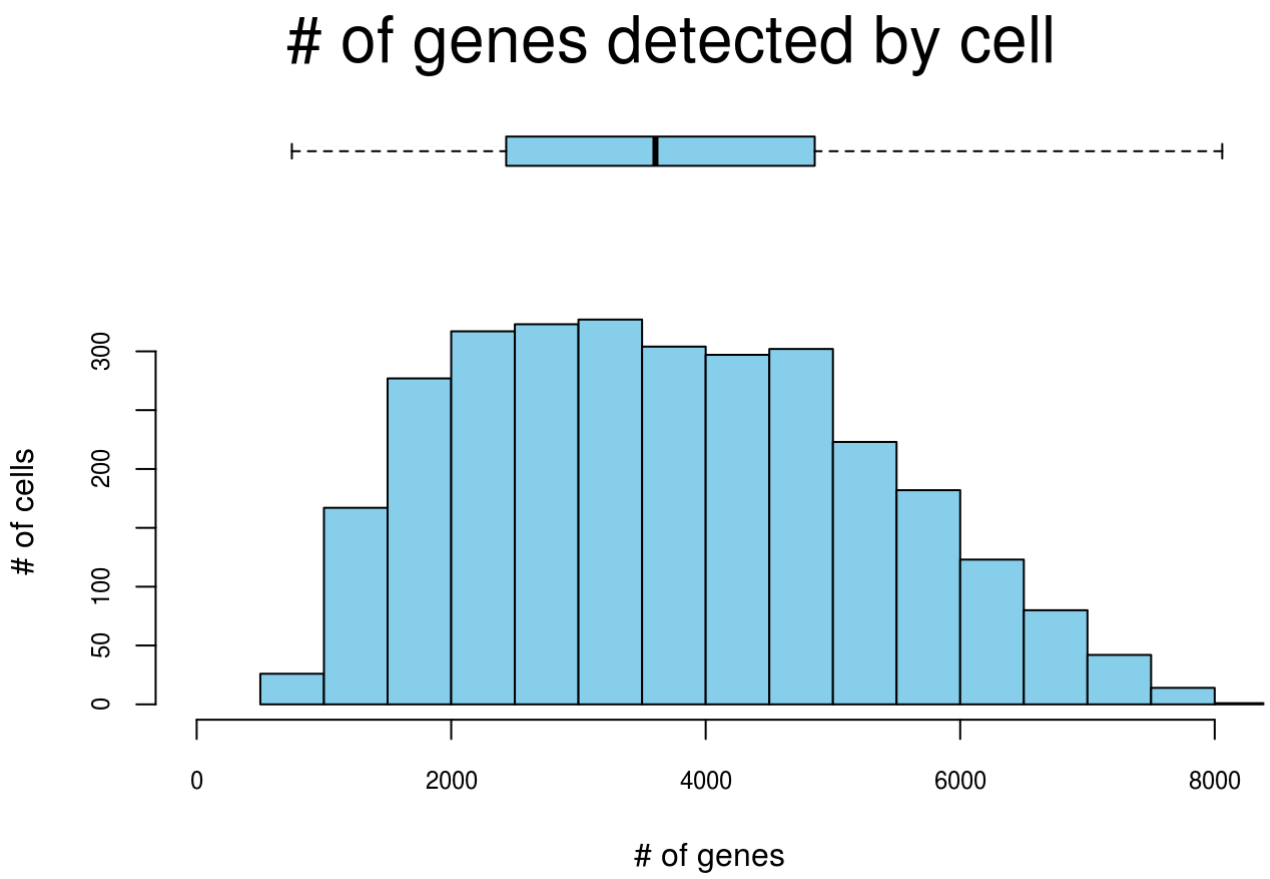
```
## Using 6 cores.
```

```
## Using 6 cores.
```

```
## Warning: Feature names cannot have underscores ('_'), replacing with dashes
## ('-')
```

```
## Warning: Keys should be one or more alphanumeric characters followed by an
## underscore, setting key from icanet_tf_ to icanettf_
```

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



Important parameters are as follows:

- *W.top.TFs* The weight threshold to define “activated” TFs whose attribute values are used to compare the similarity between different ICs.
- *W.top.genes* The weight threshold to define “activated” genes whose attribute values are used to compare the similarity between different ICs.
- *TF_motif_annot* The TF-binding motif matrix provided by RcisTarget [1].
- *cutoff* Integer value to determine the significant TF-regulons. The default value is 0.01.

Meanwhile, ICAntTF also returns the details of module information including each module’s TF activity value and its corresponding statistical significance. The statistical significance is calculated from Monte Carlo randomization test, representing if the average gene activate score on each TF-module is significantly higher/lower than the average activate score calculated from randomly sampled genes. The more significant of a module, the higher possibility that this module is activated. We could check the module information through the following commands.

```
moduleInfor <- mouseData@misc$IcaNet_geneSets_TF_moduleInfor
head(moduleInfor)
```

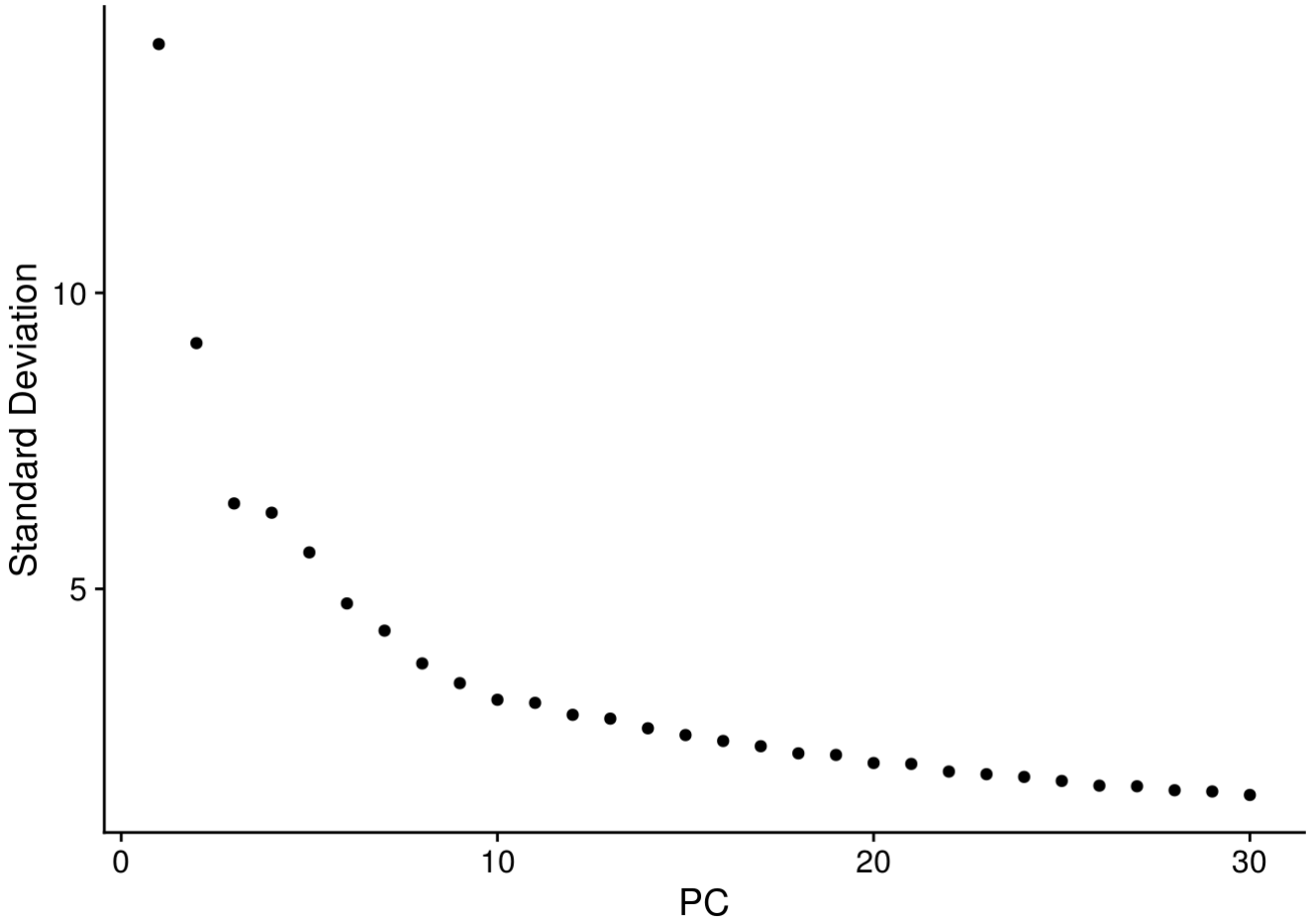
```
##                                                gene     moduleSize
## ICAnet-1-4-Bcl11a+hocomoco__BC11A_HUMAN.H10MO.C "Bcl11a" "4"
## ICAnet-1-4-Bcl11a+transfac_pro__M04833          "Bcl11a" "4"
## ICAnet-1-8-Bcl11b+transfac_pro__M05953          "Bcl11b" "8"
## ICAnet-1-3-Foxp1+cisbp__M0721                   "Foxp1"  "3"
## ICAnet-1-3-Foxp1+cisbp__M0727                   "Foxp1"  "3"
## ICAnet-1-3-Foxp1+cisbp__M0748                   "Foxp1"  "3"
##                                                motif_list
## ICAnet-1-4-Bcl11a+hocomoco__BC11A_HUMAN.H10MO.C "hocomoco__BC11A_HUMAN.H10MO.C"
## ICAnet-1-4-Bcl11a+transfac_pro__M04833          "transfac_pro__M04833"
## ICAnet-1-8-Bcl11b+transfac_pro__M05953          "transfac_pro__M05953"
## ICAnet-1-3-Foxp1+cisbp__M0721                   "cisbp__M0721"
## ICAnet-1-3-Foxp1+cisbp__M0727                   "cisbp__M0727"
## ICAnet-1-3-Foxp1+cisbp__M0748                   "cisbp__M0748"
##                                                modularity
## ICAnet-1-4-Bcl11a+hocomoco__BC11A_HUMAN.H10MO.C "-0.104492175183937"
## ICAnet-1-4-Bcl11a+transfac_pro__M04833          "-0.0687305917832474"
## ICAnet-1-8-Bcl11b+transfac_pro__M05953          "-0.050023316903042"
## ICAnet-1-3-Foxp1+cisbp__M0721                   "-0.105103253837964"
## ICAnet-1-3-Foxp1+cisbp__M0727                   "-0.108083841105661"
## ICAnet-1-3-Foxp1+cisbp__M0748                   "-0.108220231624089"
##                                                significance TF_activity_all
## ICAnet-1-4-Bcl11a+hocomoco__BC11A_HUMAN.H10MO.C "0"          "3.75981675707898"
## ICAnet-1-4-Bcl11a+transfac_pro__M04833          "0.01"       "3.75981675707898"
## ICAnet-1-8-Bcl11b+transfac_pro__M05953          "0"          "7.08914595413014"
## ICAnet-1-3-Foxp1+cisbp__M0721                   "0.01"       "4.3601223827535"
## ICAnet-1-3-Foxp1+cisbp__M0727                   "0"          "4.3601223827535"
## ICAnet-1-3-Foxp1+cisbp__M0748                   "0"          "4.3601223827535"
```

Information of each module information is stored in *mouseData@misc$IcaNet_geneSets_TF_moduleInfor (mailto:mouseData@misc$IcaNet_geneSets_TF_moduleInfor)*, which contains five column. The first column represents the transcriptional factor that regulates this module. The second column represents the number of genes in the module. The third column contains motif list. The forth column contains modularity score of each module, which represents the average IC attributes values of the member genes in the module. The fifth column contains the module significance of each TF-regulon, which is calculated from Monte Carlo test. The last column is the activity score of each TF.

Using TF-regulon predict by ICAnetTF to perform cell clustering

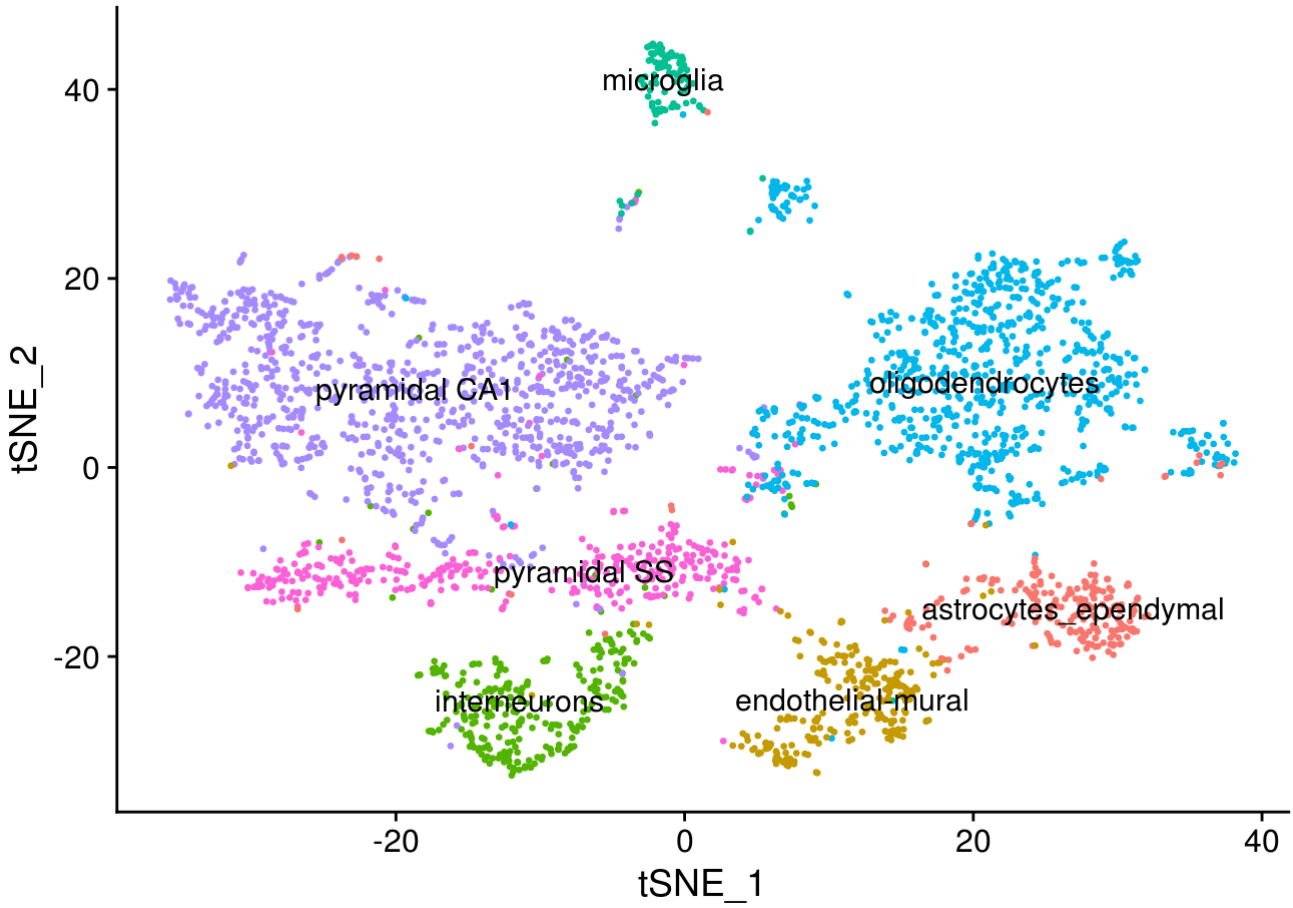
As ICAnet scans the enrichments of all possible TF-motif regulation targets, it tends to detect much more possible TF-regulons than SCENIC identified. We use these modules to perform cell clustering.

```
mouseData <- RunPCA(mouseData,npcs = 30,features=rownames(mouseData),verbose = F)
ElbowPlot(mouseData,ndims=30)
```




```
mouseData <- RunTSNE(mouseData, reduction = "pca", dims = 1:20, reduction.name = "tsne",  reduction.key = "tSNE_")
DimPlot(mouseData, reduction = 'tsne',  group.by = 'celltype',label=1)+NoLegend()
```

```
## Warning: Using `as.character()` on a quosure is deprecated as of rlang 0.3.0.
## Please use `as_label()` or `as_name()` instead.
## This warning is displayed once per session.
```



This figure represent t-SNE visualization of cells, each cell is colored according to cell type id.

We could notice that the test cells are largely grouped according to cell identity. To show its clustering performance, we run Louvian clustering algorithm to cluster these cells.

```
library(mclust)
```

```
## Package 'mclust' version 5.4.5
## Type 'citation("mclust")' for citing this R package in publications.
```

```
mouseData <- FindNeighbors(mouseData, dims = 1:20,reduction="pca")
```

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

```
## Computing SNN
```

```
mouseData <- FindClusters(mouseData, resolution = 0.2,algorithm=2)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 3005
## Number of edges: 107095
##
## Running Louvain algorithm with multilevel refinement...
## Maximum modularity in 10 random starts: 0.9396
## Number of communities: 7
## Elapsed time: 0 seconds
```

```
adjustedRandIndex(as.numeric(as.factor(mouseData$celltype)),as.numeric(mouseData@active.ident))
```

```
## [1] 0.8698687
```

We could find that ICAnetTF reaches highly accurate clustering performance (ARI > 0.85), which is comparable and even surpasses the clustering performance of SCENIC [1] and SC3 [2]. Notably, compared with SCENIC, ICAnet can identify activated TF-regulon within 10 minutes.

Reference

[1] Aibar S, González-Blas C B, Moerman T, et al. SCENIC: single-cell regulatory network inference and clustering[J]. Nature methods, 2017, 14(11): 1083-1086. [2] Kiselev V Y, Kirschner K, Schaub M T, et al. SC3: consensus clustering of single-cell RNA-seq data[J]. Nature methods, 2017, 14(5): 483-486.