scRNA Seq Cell Subgroup Tutorial

Yufei Wang

2023-01-16

In this tutorial we will cover the pipeline for single cell RNA-Seq raw data to various kinds of visualization involving Dimensionality Reduction & Clustering & Cell Type Annotation. Here, we use GSE132257 as example.

Create Seurat objects based on the expression matrix after GEO processing

Here we will use two forms of file: "GEO processed raw UMI count matrix" and "GEO processed 10X cell_annotation"

```
datadir = "GSE132257_GEO_processed_protocol_and_fresh_frozen_raw_UMI_count_matrix.txt"
metadata = "GSE132257_processed_protocol_and_fresh_frozen_cell_annotation.txt"
outdir = "./Dimention"
nfeature<-100
ncount<-200
mito<-10
ribo<-100
red<-10
```

Read gene-sampleID matrix and metadata

```
library(data.table)
data<-fread(datadir,sep = "\t", header = T,check.names = F)
data<-data.frame(data,check.names = F, row.names = 1)

metadata<-fread(metadata,sep = "\t", header = T,check.names = F)
metadata<-data.frame(metadata,check.names = F, row.names = 1)
```

Create SeuratObject

Remember to attach SeuratObject by using "library(Seurat)"

```
library(Seurat)
```

```
## Attaching SeuratObject
```

```
project = CreateSeuratObject(counts = data,metadata = metadata)
```

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

```
project = AddMetaData(object = project, metadata = metadata)
```

Data cleaning

```
table(grepl("^RP[SL][[:digit:]]",rownames(project)))
```

```
##
## FALSE  TRUE
## 33595    99
```

```
project[["mito.per"]]  = PercentageFeatureSet(project, pattern = "^MT-")
project[["ribo.per"]]  = PercentageFeatureSet(project, pattern = "^RP[SL][[:digit:]]")
project[["redcell.per"]]  = PercentageFeatureSet(project, pattern = "^HB-")

project <- subset(project,
                  subset = nFeature_RNA >= nfeature &
                    nCount_RNA >= ncount &
                    mito.per <= mito &
                    ribo.per <= ribo &
                    redcell.per<=red )
```

Data dimensionality reduction and clustering

Main process: Normalize –> FindVariableFeatures –> ScaleData –> RunPCA –> FindNeighbors –> FindClusters –> RunTSNE/RunUMAP

Log Normalize the data

```
project<- NormalizeData(object = project, scale.factor = 10000)
```

Find DEG

```
project<- FindVariableFeatures(object = project, selection.method = "vst", nfeatures = 2000, mean.cutof
```

Standardize

```
project <- ScaleData(project,verbose = FALSE,features=rownames(project))
```

PCA

```
project <- RunPCA(project, npcs = 10, verbose = FALSE)
```

Create figure

```
project <- FindNeighbors(project, reduction = "pca", dims = 1:10)
```

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

Cluster

```
project <- FindClusters(project, resolution = 0.8)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
```

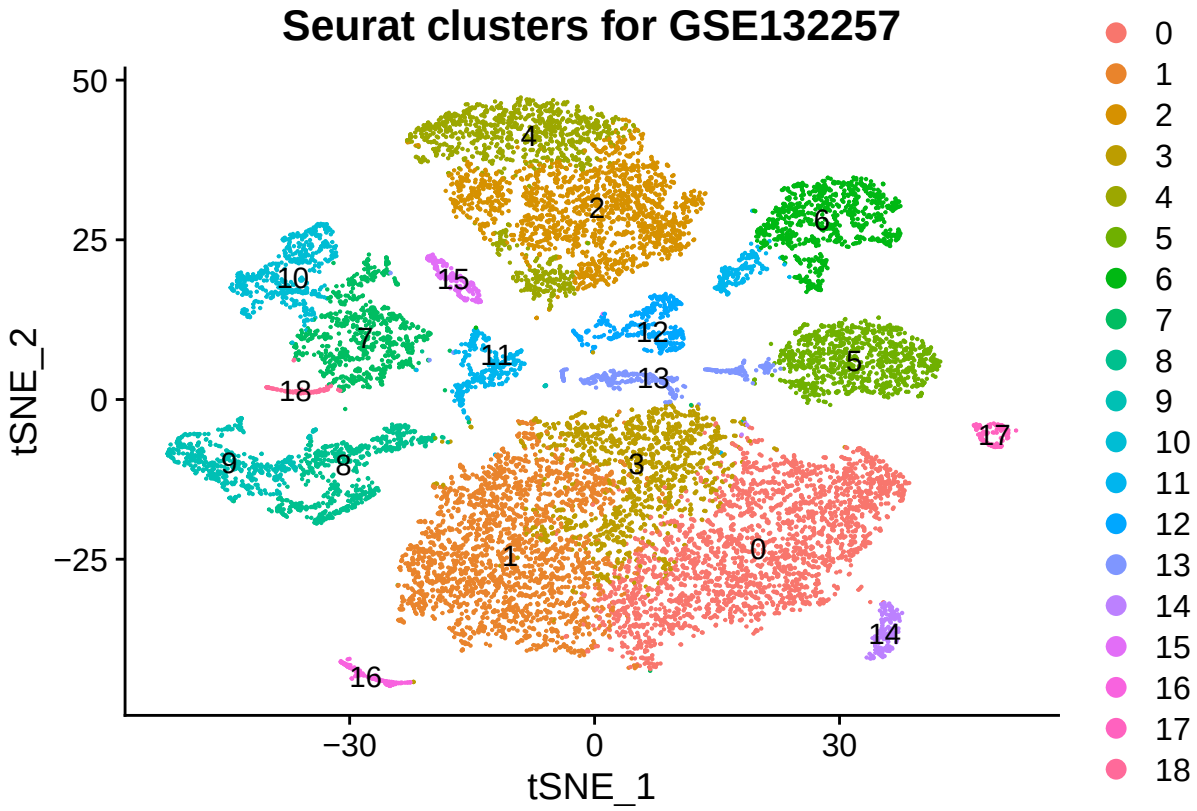
```
##
## Number of nodes: 15697
## Number of edges: 490752
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8921
## Number of communities: 19
## Elapsed time: 2 seconds
```

Dimension reduction: tsne/umap (Here we use tsne)

```
project <- RunTSNE(object = project, reduction = "pca",dims = 1:10)
```

Now our data is ready for fancy plots! (without annotation)

```
library(ggplot2)
DimPlot(object=project,dims = c(1, 2), reduction ="tsne",
        group.by = c("seurat_clusters"),label = TRUE)+
  ggtitle("Seurat clusters for GSE132257")
```



SingleR for Cell Annotation

SingleR can annotate cell subtype for a target data using a well-annotated reference data set. References can be accessed through celldex package. Then, we change the cluster names from being numeric as above, to specific cell type as predicted by the SingleR function.

```
library(SingleR)
```

```
## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Loading required package: matrixStats

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgsPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgsPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##     rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##     rowWeightedSds, rowWeightedVars

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: BiocGenerics

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:data.table':
##
##     first, second

## The following objects are masked from 'package:base':
##
##     expand.grid, I, unname
```

```
## Loading required package: IRanges
##
## Attaching package: 'IRanges'
## The following object is masked from 'package:data.table':
##
##     shift
## Loading required package: GenomeInfoDb
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Attaching package: 'Biobase'
## The following object is masked from 'package:MatrixGenerics':
##
##     rowMedians
## The following objects are masked from 'package:matrixStats':
##
##     anyMissing, rowMedians
##
## Attaching package: 'SummarizedExperiment'
## The following object is masked from 'package:SeuratObject':
##
##     Assays
## The following object is masked from 'package:Seurat':
##
##     Assays
library(celldex)
##
## Attaching package: 'celldex'
## The following objects are masked from 'package:SingleR':
##
##     BlueprintEncodeData, DatabaseImmuneCellExpressionData,
##     HumanPrimaryCellAtlasData, ImmGenData, MonacoImmuneData,
##     MouseRNAseqData, NovershternHematopoieticData
ref <- HumanPrimaryCellAtlasData()
## snapshotDate(): 2022-10-31
## see ?celldex and browseVignettes('celldex') for documentation
## loading from cache
## see ?celldex and browseVignettes('celldex') for documentation
## loading from cache
```

```
pred.GSE132257 <- SingleR(test = project@assays$RNA@data, ref = ref, assay.type.test = 1,
                          labels = ref$label.main, clusters = project@active.ident)
```

The following code was failed but still needs new try. "new.cluster.ids.GSE132257 <- pred.GSE132257$pruned.labels" "names(new.cluster.ids.GSE132257) <- levels(project)" "project <- RenameIdents(project, new.cluster.ids.GSE132257)"

So far, I used a a little more complicated method:

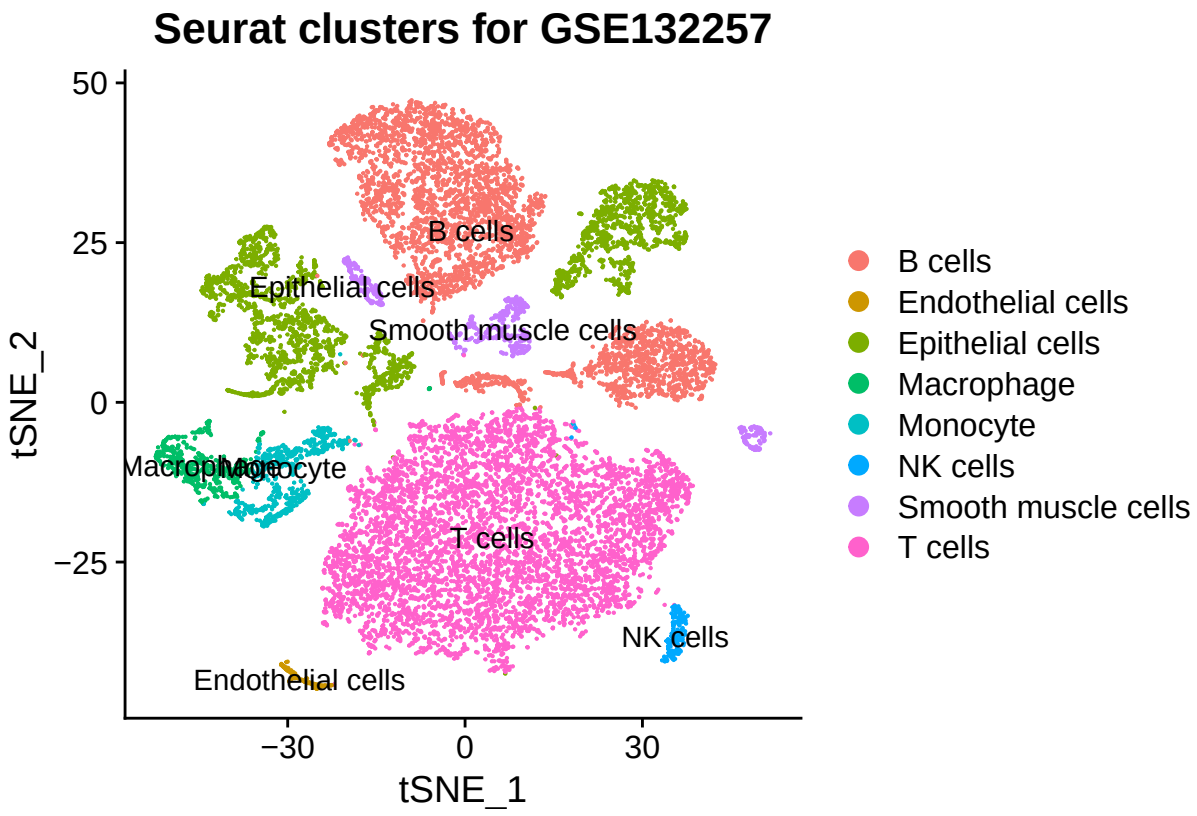
```
GSE132257.celltype <- c("0"="T cells",
                        "1"="T cells",
                        "2"="B cells",
                        "3"= "T cells",
                        "4"= "B cells",
                        "5"= "B cells",
                        "6"= "Epithelial cells",
                        "7"= "Epithelial cells",
                        "8"= "Monocyte",
                        "9"= "Macrophage",
                        "10"= "Epithelial cells",
                        "11"= "Epithelial cells",
                        "12"= "Smooth muscle cells",
                        "13"= "B cells",
                        "14"= "NK cells",
                        "15"= "Smooth muscle cells",
                        "16"= "Endothelial cells",
                        "17"= "Smooth muscle cells",
                        "18"= "Epithelial cells")
```

Code above: see the name of "pred.GSE132257" and annotate them manually.

```
project[['cell_type']] = unname(GSE132257.celltype[project@meta.data$seurat_clusters])
```

Now we can plot the tsne again, but this time with annotated labels!

```
DimPlot(object=project,dims = c(1, 2), reduction ="tsne",
        group.by = c("cell_type"),label = TRUE)+
  ggtitle("Seurat clusters for GSE132257")
```


Seurat clusters for GSE132257
50
25
0
−25
tSNE_2
B cells
Epithelial cells
Smooth muscle cells
Macrophage
Monocyte
T cells
NK cells
Endothelial cells
−30
0
30
tSNE_1
B cells
Endothelial cells
Epithelial cells
Macrophage
Monocyte
NK cells
Smooth muscle cells
T cells

The End