10X data analysis pipeline

Part1.Testing data source

1.Website

All test data are downloaded from 10X genomics website:

https://support.10xgenomics.com/single-cell-gene-expression/datasets

2.Dataintros

Each dataset has a brief introduction, below is our test one:

1k Brain Cells from an E18 Mouse

Single Cell Gene Expression Dataset by Cell Ranger 2.1.0

Cells from a combined cortex, hippocampus and sub ventricular zone of an E18 mouse.

- 931 cells detected
- Sequenced on Illumina HiSeq2500 with approximately 56,000 reads per cell
- 26bp read1 (16bp Chromium barcode and 10bp UMI), 98bp read2 (transcript), and 8bp I7 sample barcode
- Analysis run with —cells=2000
- Analysis run with reference version 1.2.0

Published on November 8, 2017

Part2.Installation Instructions

1.10X Cellranger

1)Official support：

support@10xgenomics.com

2)Minimum requirements：

Cell Ranger pipelines run on Linux systems that meet these minimum requirements:

- 8-core Intel or AMD processor (16 recommended)
- 64GB RAM (128GB recommended)
- 1TB free disk space
- 64-bit CentOS/RedHat 5.5 or Ubuntu 10.04

3)Cellranger 2.1（just unzip to use）：

wget -O cellranger-2.1.0.tar.gz "http://cf.10xgenomics.com/releases/cell-exp/cellranger-2.1.0.tar.gz?Expires=1519742068&Policy=eyJTdGF0ZW1lbnQiOlt7IlJlc291cmNlIjoiaHR0cDovL2NmLjEweGdlbm9taWNzLmNvbS9yZWxlYXNlcy9jZWxsLWV4cC9jZWxscmFuZ2VyLTIuMS4wLnRhci5neiIsIkNvbmRpdGlvbiI6eyJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTUxOTc0MjA2OH19fV19&Signature=asbBQSCEXuOKkh3Qb4JOLqYL~Wpg2k7r2XEuwg7VpE6-ThR59Daw1qHVwwxHSbAD8bo4FoXdizyFWKMchAjqefoFPe~52xWOB4f3ZgX7Ey1gLtN04D-dAfAzMqJew9wF5d2fyav~~0UsSTThsKYWZ2wRKp9BQPvmY8J2BMCbwIE~7rmGty~1BVgm697nzDOGiyQ2HHHrFU0H3dMBLhdfTpsI23EdKgQvattQHDFotcheC~1jrqIU4F-QewrSII3pef0IrRBwsWvjDvoUXi-u8OpYx7V9LjrgMztWKW-MhTBf0EkmJGlTUCpsAbWwUCLIASn5N1llH9OQeigL23QVxw__&Key-Pair-Id=APKAI7S6A5RYOXBWRPDA"

4)Refrence file：

GRCh38 wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-GRCh38-1.2.0.tar.gz

Hg19 wget http://cf.10xgenomics.com/supp/cell-exp/refdata-cellranger-hg19-1.2.0.tar.gz

2.Seurat

1)Requirements

1. Install R (v >= 3.3). We recommend using R 3.4.1
2. Recommended : Install R Studio

If you have installed a previous version of Seurat, restart your R session before loading the new version.

2)Install from CRAN

Seurat is now available on CRAN for all platforms. To install, run:

```
# Enter commands in R (or R studio, if installed)
install.packages('Seurat')
library(Seurat)
```

Part3.Analysis procedures

1.From fastqs to expression matrix(By Cellranger)

1)The input files

It is highly likely that these files were initially processed with bcl2fastq, so you will need to rename the files in the following format, once you track down their origin:

[Sample Name]_S1_L00[Lane Number]_[Read Type]_001.fastq.gz

Where Read Type is one of:

- I1: Sample index read (optional)
- R1: Read 1
- R2: Read 2

Details can be found in below network:

https://support.illumina.com/help/BaseSpace_OLH_009008/Content/Source/Informatics/BS/NamingConvention_FASTQ-files-swBS.htm

2)The command lines

Usage: [ ] can be ignored

```
cellranger count
--id=ID \
--fastqs=PATH \
  [--sample=PREFIX] \
--transcriptome=DIR \
  [options]
```

Arguments:

id : A unique run id, used to name output folder [a-zA-Z0-9_-]+.

fastqs : Path of folder containing fastq files.

sample : Prefix of the filenames of FASTQs to select.(can be deprecated if only one sample or you want to count all the samples in the folder)

transcriptome : Path of folder containing 10x-compatible reference.

options : Extra options that can tune the program(often as default)

2.Multiple analysis of expression matrix(By Seurat)

All below procedures must obey the same orders cause they have forward dependences.

1)Setup the Seurat Object

We start by reading in the data. All features in Seurat have been configured to work with sparse matrices which results in significant memory and speed savings for Drop-seq/inDrop/10x data.

```
install.packages('Seurat')
library(Seurat)
library(dplyr)
library(Matrix)
#Load the dataset
mouse.data <- Read10X(data.dir="F:/HCA/sample3v3/outs/filtered_gene_bc_matrices/mm10")
mouse <- CreateSeuratObject(raw.data = mouse.data, min.cells = 3, min.genes = 200, project = "10X_mouse")
```

Input

Raw expression matrix.

Arguments

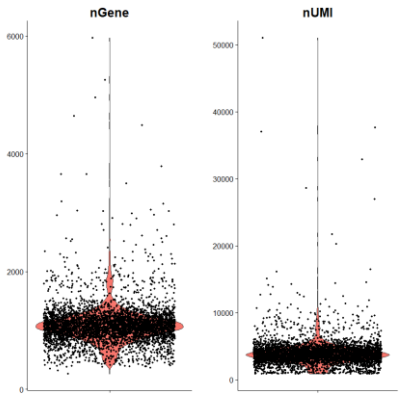
raw.data	Raw input data
project	Project name (string)
min.cells	Include genes with detected expression in at least this many cells. Will subset the raw.data matrix as well. To reintroduce excluded genes, create a new object with a lower cutoff.
min.genes	Include cells where at least this many genes are detected.

Value

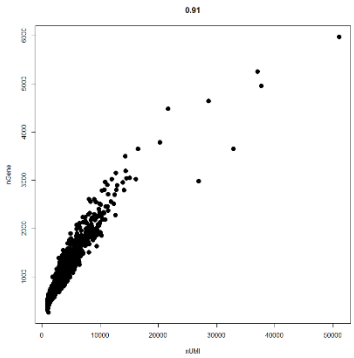
Returns a Seurat object with the raw data stored in object@raw.data. object@data, object@meta.data, object@ident, also initialized.

2)QC and selecting cells for further analysis

```
VlnPlot(object = mouse, features.plot = c("nGene", "nUMI"), nCol = 2)
```




```
GenePlot(object = mouse, gene1 = "nUMI", gene2 = "nGene")
```




```
mouse <- FilterCells(object = mouse, subset.names = "nGene", low.thresholds = c(200), high.thresholds = c(2500))
```

Input

Original Seurat object which hasn't been QC and filtered.

Arguments

object	Seurat object
subset.names	Parameters to subset on. Eg, the name of a gene, PC1, a column name in object@meta.data, etc. Any argument that can be retreived using FetchData.
low.thresholds	Low cutoffs for the parameters (default is -Inf)
high.thresholds	High cutoffs for the parameters (default is Inf)

Value

Returns a Seurat object containing only the relevant subset of cells

3)Normalizing the data

After removing unwanted cells from the dataset, the next step is to normalize the data. By default, we employ a global-scaling normalization method "LogNormalize" that normalizes the gene expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result.

```
pbmc <- NormalizeData(object = mouse, normalization.method = "LogNormalize",    scale.factor = 10000)
```

Input

Seurat object after QC and filtering.

Arguments

object	Seurat object
normalization.method	Method for normalization. Default is log-normalization (LogNormalize). More methods to be added very shortly.
scale.factor	Sets the scale factor for cell-level normalization

Value

Returns object after normalization. Normalized data is stored in data or scale.data slot, depending on the method

4)Detection of variable genes across the single cells

```
pbmc <- FindVariableGenes(object = mouse, mean.function = ExpMean, dispersion.function = LogVMR, x.low.cutoff = 0.0125, x.high.cutoff = 3, y.cutoff = 0.5)
```

Input

Seurat object after QC and filtering.

Arguments

object	Seurat object
mean.function	Function to compute x-axis value (average expression). Default is to take the mean of the detected (i.e. non-zero) values
dispersion.function	Function to compute y-axis value (dispersion). Default is to take the standard deviation of all values/
x.low.cutoff	Bottom cutoff on x-axis for identifying variable genes
x.high.cutoff	Top cutoff on x-axis for identifying variable genes
y.cutoff	Bottom cutoff on y-axis for identifying variable genes

Value

Seurat object with variable genes selected.

5)Scaling the data and removing unwanted sources of variation

```
pbmc <- ScaleData(object = mouse, vars.to.regress = c("nUMI", "percent.mito"))
```

Input

Seurat object with variable genes selected.

Arguments

object	Seurat object
vars.to.regress	Variables to regress out (previously latent.vars in RegressOut). For example, nUMI, or percent.mito.

Value

Returns a seurat object with object@scale.data updated with scaled and/or centered data.

6)Perform linear dimensional reduction

pbmc <- RunPCA(object = mouse, pc.genes = mouse@var.genes, do.print = TRUE, pcs.print = 1:5, genes.print = 5)

Input

Seurat object with object@scale.data updated with scaled and/or centered data.

Arguments

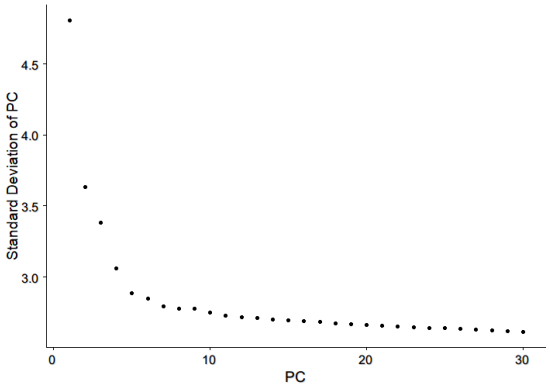
object	Seurat object
pc.genes	Genes to use as input for PCA. Default is object@var.genes
do.print	Print the top genes associated with high/low loadings for the PCs
pcs.print	PCs to print genes for
genes.print	Number of genes to print for each PC

Value

Returns Seurat object with the PCA calculation stored in object@dr$pca.

7)Determine statistically significant principal components

PCElbowPlot(object = mouse)



Input

Seurat object with the PCA calculation stored in object@dr$pca.

Arguments

object	Seurat object

Value

Returns ggplot object.

8)Cluster the cells

pbmc <- FindClusters(object = mouse, reduction.type = "pca", dims.use = 1:10, resolution = 0.6, print.output = 0, save.SNN = TRUE)

Input

Seurat object with the PCA calculation stored in object@dr$pca.

Arguments

object	Seurat object
reduction.type	Name of dimensional reduction technique to use in construction of SNN graph. (e.g. "pca", "ica")
dims.use	A vector of the dimensions to use in construction of the SNN graph (e.g. To use the first 10 PCs, pass 1:10)
print.output	Whether or not to print output to the console
distance.matrix	Build SNN from distance matrix (experimental)
save.SNN	Saves the SNN matrix associated with the calculation in object@snn
resolution	Value of the resolution parameter, use a value above (below) 1.0 if you want to obtain a larger (smaller) number of communities.

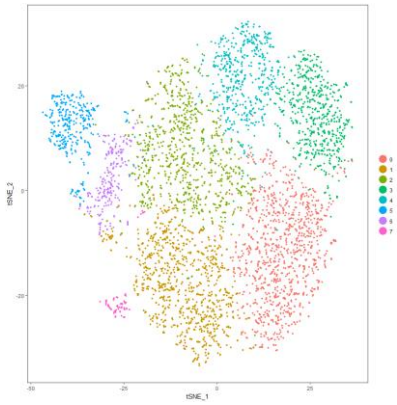
Value

Returns a Seurat object and optionally the SNN matrix, object@ident has been updated with new cluster info.

9)Run Non-linear dimensional reduction (tSNE)

pbmc <- RunTSNE(object = pbmc, dims.use = 1:10, do.fast = TRUE)

TSNEPlot(object = pbmc)



Input

Seurat object with the PCA calculation stored in object@dr$pca and cluster info stored in object@ident.

Arguments

object	Seurat object
dims.use	Which dimensions to use as input features
do.fast	If TRUE, uses the Barnes-hut implementation, which runs faster, but is less flexible. TRUE by default.

Value

Returns a Seurat object with a tSNE embedding in object@dr$tsne@cell.embeddings.

10)Finding differentially expressed genes (cluster biomarkers)

cluster1.markers <- FindMarkers(object = mouse, ident.1 = 1, min.pct = 0.25)

print(x = head(x = cluster.markers, n = 5))

```
>  print(x = head(x = cluster1.markers, n = 5))
              p_val avg_logFC pct.1 pct.2     p_val_adj
LMNA  1.942881e-248 1.5136711 0.765 0.243 3.054209e-244
ANXA1 1.363992e-158 0.8632822 0.929 0.661 2.144196e-154
ZFP36 1.988707e-119 0.6441415 0.972 0.798 3.126248e-115
FTH1  3.194966e-112 0.4525044 1.000 0.980 5.022487e-108
KLF6   4.653247e-99 0.5354614 0.970 0.777  7.314904e-95
```

Input

Seurat object with the PCA calculation stored in object@dr$pca and cluster info stored in object@ident.

Arguments

object	Seurat object
ident.1	Identity class to define markers for
min.pct	only test genes that are detected in a minimum fraction of min.pct cells in either of the two populations. Meant to speed up the function by not testing genes that are very infrequently expressed. Default is 0.1

Value

Matrix containing a ranked list of putative markers, and associated statistics (p-values, ROC score, etc.)

Part4.Summary and results

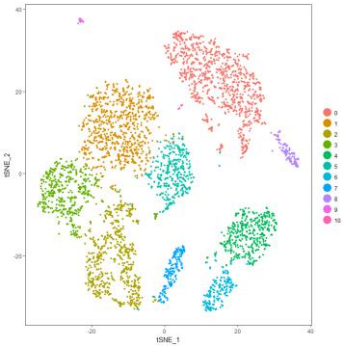
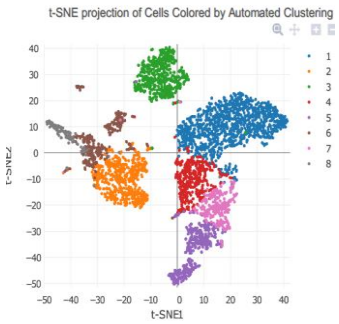
1.All source code can be found in https://paste.ubuntu.com/p/fJFqg2zkpH/

2.Analysis results:

Cellranger official pipeline do provide some analysis result, here I compared it with the Seurat results:

Here we used a more diverse dataset to do the validation.

Data can be found in https://support.10xgenomics.com/single-cell-gene-expression/datasets/2.1.0/pbmc4k

	Result by Seurat	Reference
tSNE		

Cluster marker	`> print(x = head(x = cluster3.markers, n = 5))`
	`p_val avg_logFC pct.1 pct.2`
	`TNFRSF4 5.419837e-133 0.5377856 0.255 0.015 9.02`
	`IL7R 1.022800e-130 1.0846075 0.841 0.344 1.70`
	`LTB 6.580635e-118 0.9294000 0.998 0.751 1.09`
	`ITGB1 6.792568e-118 1.0527865 0.648 0.202 1.13`
	`IL32 6.116023e-110 0.8940391 0.969 0.565 1.01`	

		Cluster 1		Cluster 2		Cluster 3		Cluster 4	
Gene ID	Gene name	L2FC	p-value	L2FC	p-value	L2FC	p-value	L2FC	p-value
ENSG00000150093	ITGB1	-1.49	1e-05	-0.56	4e-01	-0.75	4e-01	2.02	6e-09
ENSG00000168685	IL7R	1.03	4e-04	-3.10	2e-09	-3.32	9e-10	1.85	1e-07
ENSG00000111796	KLRB1	-1.39	2e-08	-3.27	7e-09	-3.88	6e-10	1.55	2e-04
ENSG00000008517	IL32	0.04	1e+00	-3.16	3e-10	-3.35	1e-10	1.54	3e-05
ENSG00000227507	LTB	0.57	1e-01	-2.67	4e-08	0.95	3e-02	1.43	1e-04

We can find that the tSNE both have clear borders, because they used different cluster method and the randomness of tSNE so these two pictures are not identical. But about the find markers, they do have great similarity. Here we can find IL7R, ITGB1, LTB, IL32 are same 4 of 5.

Fengchen

20180302