scRNAseq analysis using Rmarkdown

Cankun Wang

07/14/2020

How to use this repository

GitHub for this templtate: Click here to open

Step 1: Download the whole repository.

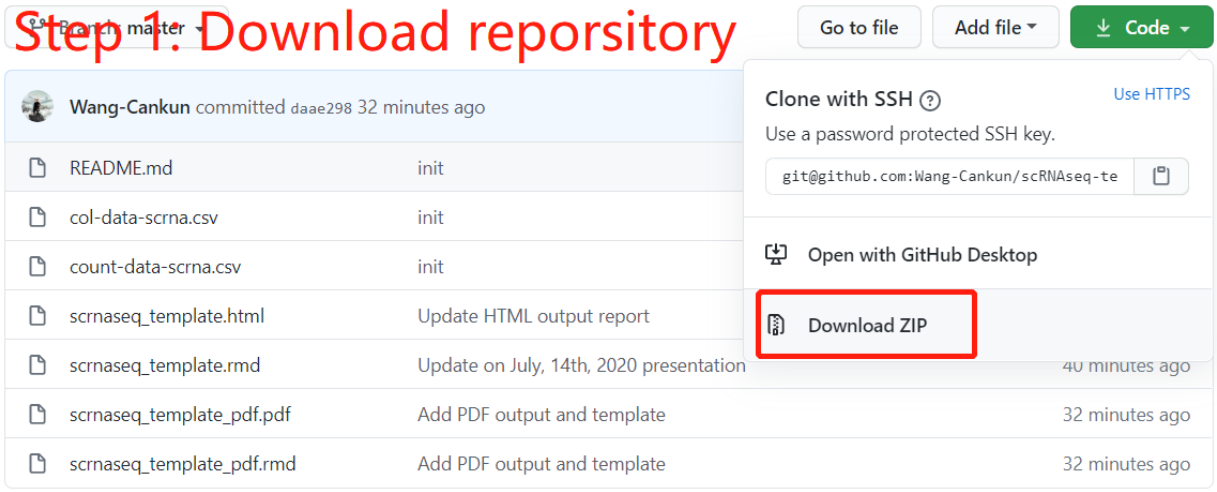


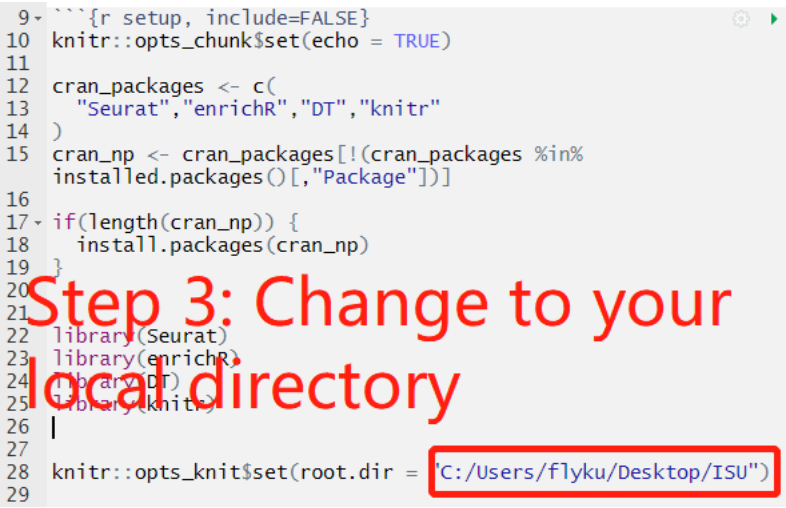
Figure 1: Download repo

Step 2: Unzip files to your local directory.

You will find these files in your directory

- count-data-scrna.csv: example gene expression matrix file.
- col-data-scrna.csv: example gene metadata file.
- scrnaseq_template_html.rmd: Rmarkdown template to generate HTML report.
- scrnaseq_template_html.html: Example HTML report.
- scrnaseq_template_pdf.rmd: Rmarkdown template to generate PDF report.
- scrnaseq_template_pdf.html: Example PDF report.

Step 3 (in Rmarkdown template): Change the working directory to your local directory.




```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)

cran_packages <- c(
  "Seurat","enrichR","DT","knitr"
)
cran_np <- cran_packages[!(cran_packages %in%
installed.packages()[,"Package"])]

if(length(cran_np)) {
  install.packages(cran_np)
}

library(Seurat)
library(enrichR)
library(DT)
library(knitr)

knitr::opts_knit$set(root.dir = 'C:/Users/flyku/Desktop/ISU')
```

Goal

The primary goal of this study is to identify gene expression difference between conditions using a toy dataset.

Rmarkdown reference can be found in here: Reference

Mathematics in Rmarkdown

Seurat - Guided Clustering Tutorial

Run Seurat strandard workflow

Load dataset

Using raw read count as input.

```
##          OoCyte_1 OoCyte_2 OoCyte_3 Zygote_1 Zygote_2 Zygote_3
## CREB3L1       953     1249     1414     1169      790      527
## GPR98        3889     6625     6059     3332     4434     7063
## RTN1         9423    12507    10256     3759     2757     1499
## KBTBD8        180      285      141     1113      675     1178
## ZEB1        11760    14258    10912    12724     8528    14400
## FAT4            0        0        0        0        0        0
## BANK1       18373    15971    11918    10526     8703    14819
## PIR           379       79       47        0      237        0
## KIAA1199        0      635        0      805      245      561
## SORL1        5898     5279     2714     2234     1519     1849
##          X2_Cell_embryo_1_Cell_1 X2_Cell_embryo_1_Cell_2
## CREB3L1                     1613                    1366
## GPR98                       2916                    4079
## RTN1                       10850                   11434
## KBTBD8                      1115                    1278
```
```

```
## ZEB1                         16406                    16338
## FAT4                             0                        0
## BANK1                         5464                     8777
## PIR                            253                      237
## KIAA1199                       439                        0
## SORL1                         1316                     2065
##          X2_Cell_embryo_2_Cell_1 X2_Cell_embryo_2_Cell_2
## CREB3L1                       1744                     1480
## GPR98                         6220                     5468
## RTN1                          5341                     4754
## KBTBD8                        1051                     1198
## ZEB1                         10348                    13699
## FAT4                             0                        0
## BANK1                         9110                    12009
## PIR                              0                      300
## KIAA1199                       757                      511
## SORL1                         1500                     1216
```

```
##                                Cluster Time Condition
## OoCyte_1                        OoCyte    1   Control
## OoCyte_2                        OoCyte    1   Control
## OoCyte_3                        OoCyte    1   Control
## Zygote_1                        Zygote    1   Control
## Zygote_2                        Zygote    1   Control
## Zygote_3                        Zygote    1   Control
## X2_Cell_embryo_1_Cell_1 2_Cell_embryo    2   Control
## X2_Cell_embryo_1_Cell_2 2_Cell_embryo    2   Control
## X2_Cell_embryo_2_Cell_1 2_Cell_embryo    2   Control
## X2_Cell_embryo_2_Cell_2 2_Cell_embryo    2   Control
```

Load dataset in Seurat

```
# choose whahtever column as design
obj <- CreateSeuratObject(counts = counts, project = "scRNAseq", min.cells = 3, min.features = 200)
obj <- AddMetaData(object = obj, metadata = meta)
```

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 feature expression measurements for each cell by the total expression, multiplies this by a scale factor (10,000 by default), and log-transforms the result. Normalized values are stored in obj[["RNA"]]@data.

```
obj <- NormalizeData(obj)
```

Identification of highly variable features (feature selection)

We next calculate a subset of features that exhibit high cell-to-cell variation in the dataset (i.e, they are highly expressed in some cells, and lowly expressed in others). We and others have found that focusing on these genes in downstream analysis helps to highlight biological signal in single-cell datasets.

Our procedure in Seurat3 is described in detail here, and improves on previous versions by directly modeling the mean-variance relationship inherent in single-cell data, and is implemented in the FindVariableFeatures function. By default, we return 2,000 features per dataset. These will be used in downstream analysis, like PCA.

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

Scaling the data

Next, we apply a linear transformation ('scaling') that is a standard pre-processing step prior to dimensional reduction techniques like PCA. The ScaleData function:

- Shifts the expression of each gene, so that the mean expression across cells is 0
- Scales the expression of each gene, so that the variance across cells is 1
- This step gives equal weight in downstream analyses, so that highly-expressed genes do not dominate
- The results of this are stored in obj[["RNA"]]@scale.data

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

Perform linear dimensional reduction (PCA)

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 you wish to choose a different subset.

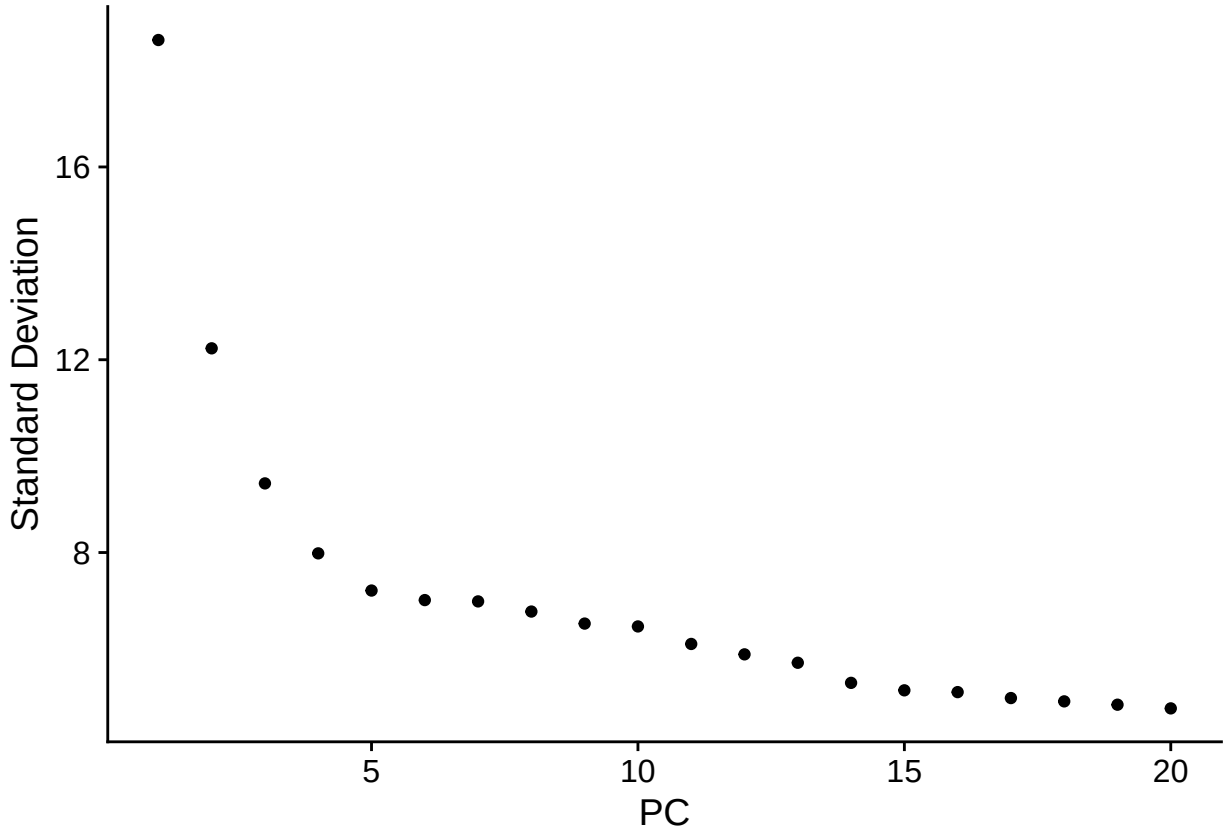
```
obj <- RunPCA(obj, features = VariableFeatures(object = obj))
```

Determine the 'dimensionality' of the dataset

To overcome the extensive technical noise in any single feature for scRNA-seq data, Seurat clusters cells based on their PCA scores, with each PC essentially representing a 'metafeature' that combines information across a correlated feature set. The top principal components therefore represent a robust compression of the dataset. However, how many componenets should we choose to include? 10? 20? 100?

In Macosko et al, we implemented a resampling test inspired by the JackStraw procedure. We randomly permute a subset of the data (1% by default) and rerun PCA, constructing a 'null distribution' of feature scores, and repeat this procedure. We identify 'significant' PCs as those who have a strong enrichment of low p-value features.

```
ElbowPlot(obj)
```



Cluster the cells

Seurat v3 applies a graph-based clustering approach, building upon initial strategies in (Macosko et al). Importantly, the distance metric which drives the clustering analysis (based on previously identified PCs) remains the same. However, our approach to partioning the cellular distance matrix into clusters has dramatically improved. Our approach was heavily inspired by recent manuscripts which applied graph-based clustering approaches to scRNA-seq data [SNN-Cliq, Xu and Su, Bioinformatics, 2015] and CyTOF data [PhenoGraph, Levine et al., Cell, 2015]. Briefly, these methods embed cells in a graph structure - for example a K-nearest neighbor (KNN) graph, with edges drawn between cells with similar feature expression patterns, and then attempt to partition this graph into highly interconnected 'quasi-cliques' or 'communities'.

As in PhenoGraph, we first construct a KNN graph based on the euclidean distance in PCA space, and refine the edge weights between any two cells based on the shared overlap in their local neighborhoods (Jaccard similarity). This step is performed using the FindNeighbors function, and takes as input the previously defined dimensionality of the dataset (first 10 PCs).

To cluster the cells, we next apply modularity optimization techniques such as the Louvain algorithm (default) or SLM [SLM, Blondel et al., Journal of Statistical Mechanics], to iteratively group cells together, with the goal of optimizing the standard modularity function. The FindClusters function implements this procedure, and contains a resolution parameter that sets the 'granularity' of the downstream clustering, with increased values leading to a greater number of clusters. We find that setting this parameter between 0.4-1.2 typically returns good results for single-cell datasets of around 3K cells. Optimal resolution often increases for larger datasets. The clusters can be found using the Idents function.

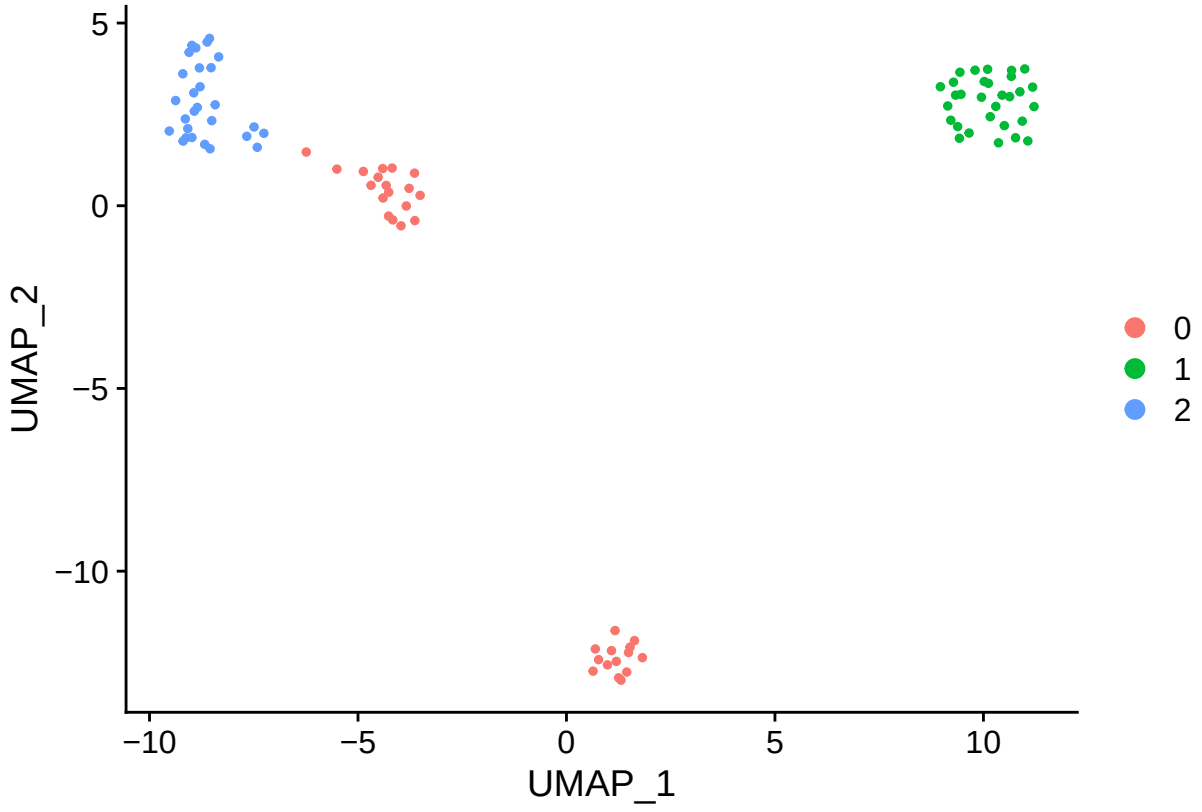
PCA plot colored by time

```
obj <- FindNeighbors(obj, dims = 1:10)
obj <- FindClusters(obj, resolution = 0.5)
```

```
## Modularity Optimizer version 1.3.0 by Ludo Waltman and Nees Jan van Eck
##
## Number of nodes: 90
## Number of edges: 1396
##
## Running Louvain algorithm...
## Maximum modularity in 10 random starts: 0.8195
## Number of communities: 3
## Elapsed time: 0 seconds
```

Run non-linear dimensional reduction (UMAP/tSNE)

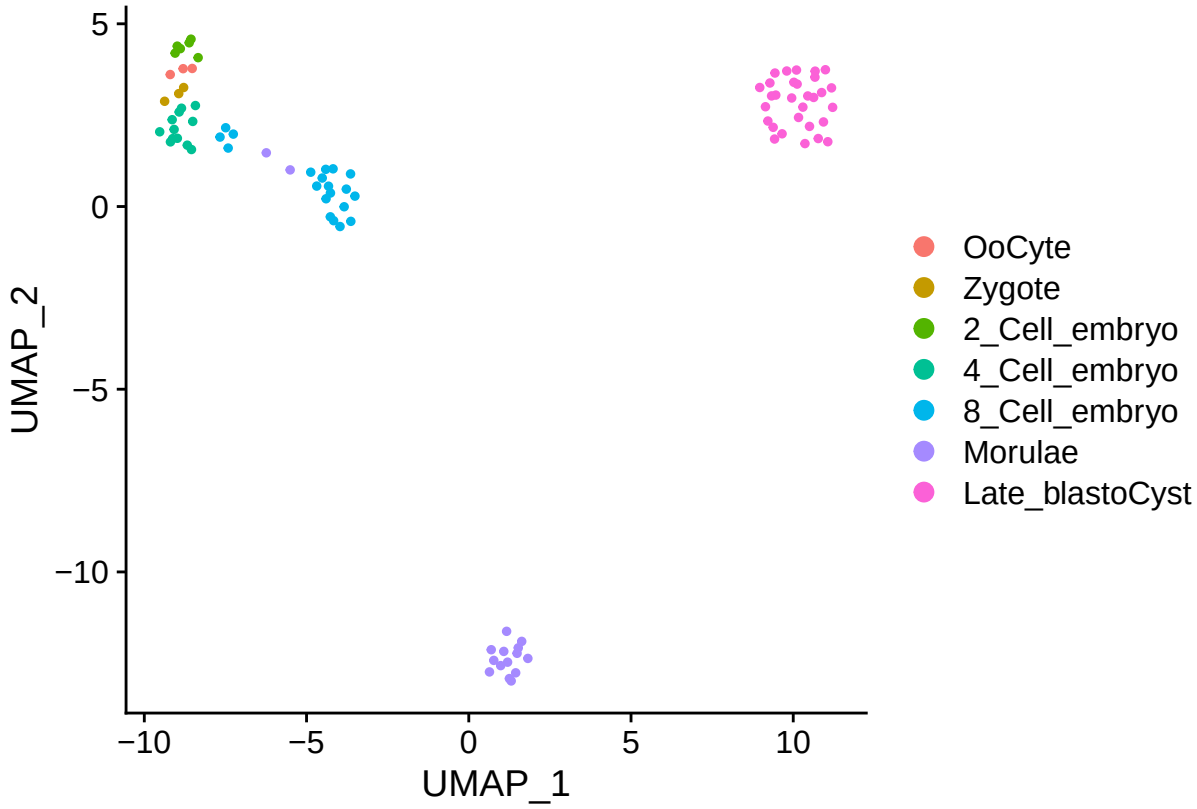
```
obj <- RunUMAP(obj, dims = 1:10)
DimPlot(obj, reduction = "umap")
```



Plot UMAP colored by custom metadata

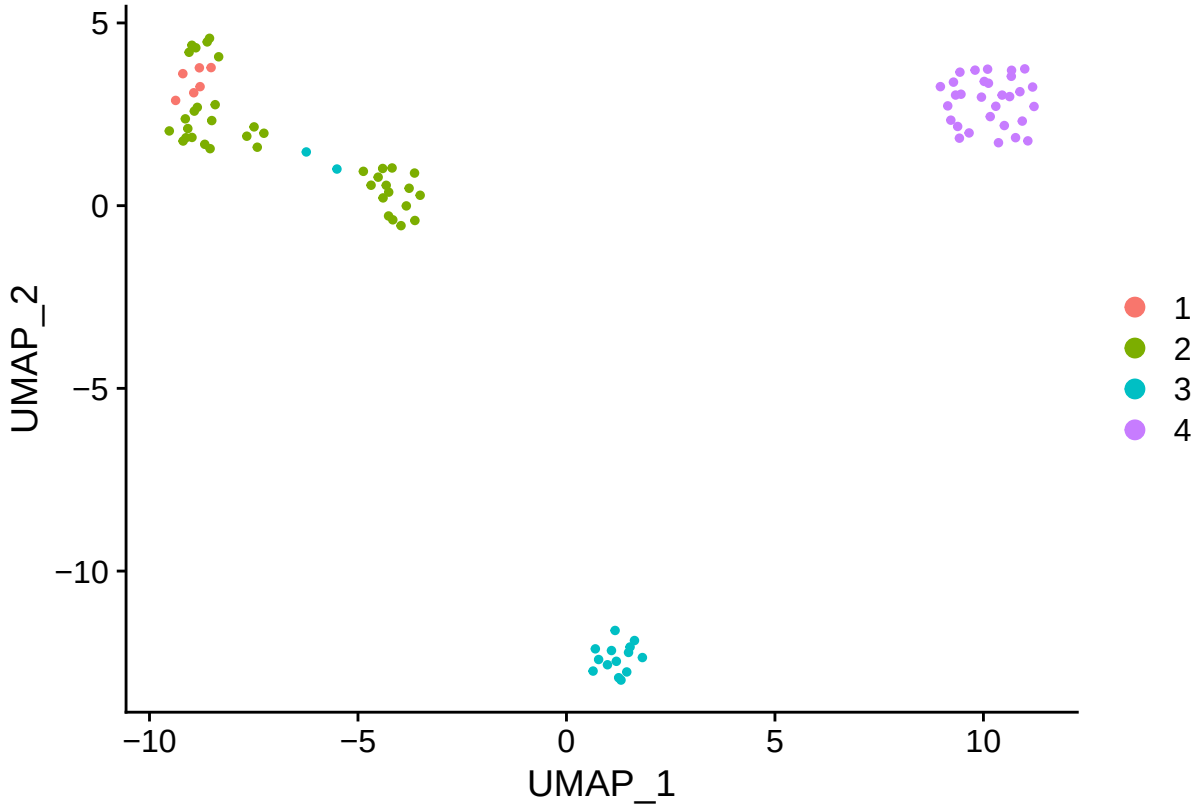
UMAP plot colored by cell types

```
Idents(obj) <- obj$Cluster
DimPlot(obj, reduction = "umap")
```



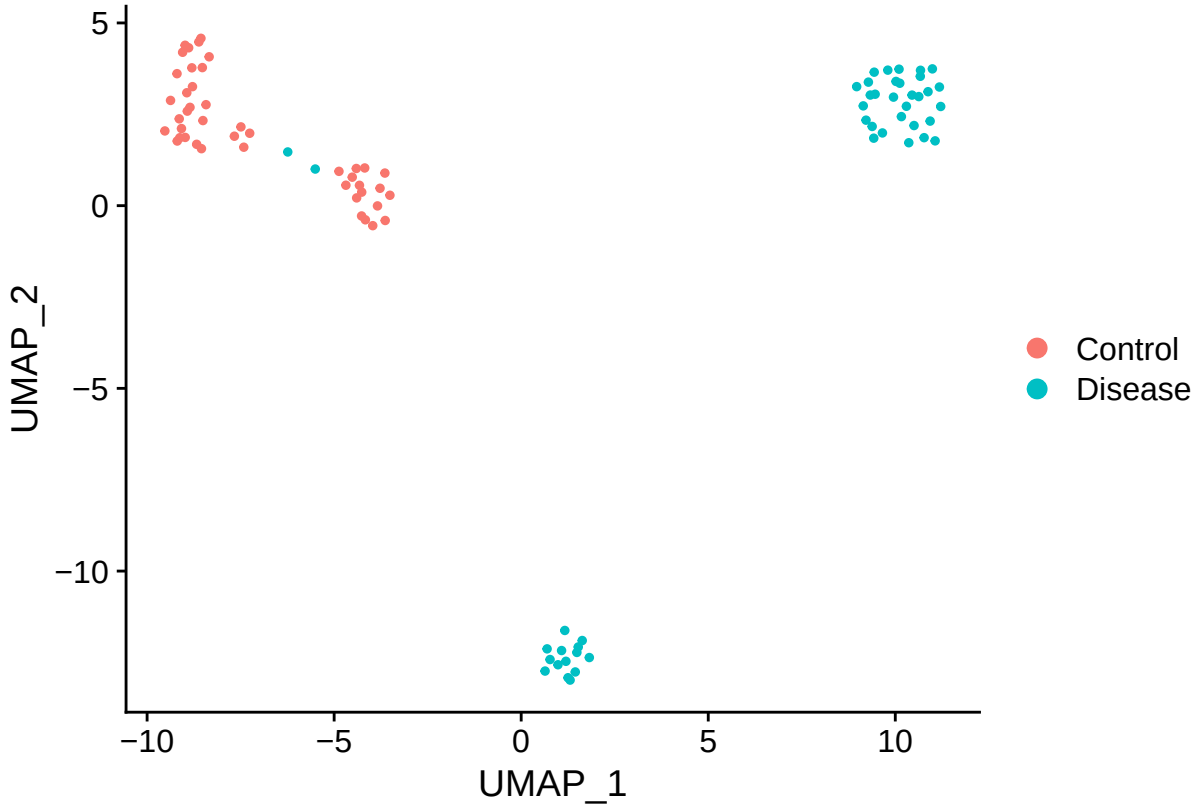
UMAP plot colored by time

```
Idents(obj) <- obj$Time
DimPlot(obj, reduction = "umap")
```



UMAP plot colored by condition

```
Idents(obj) <- obj$Condition
DimPlot(obj, reduction = "umap")
```



Finding differentially expressed features

Find cell type specific genes between control and disease group

```
Idents(obj) <- obj$Condition
cts_de_genes <- FindMarkers(obj, ident.1 = "Control", ident.2 = "Disease")

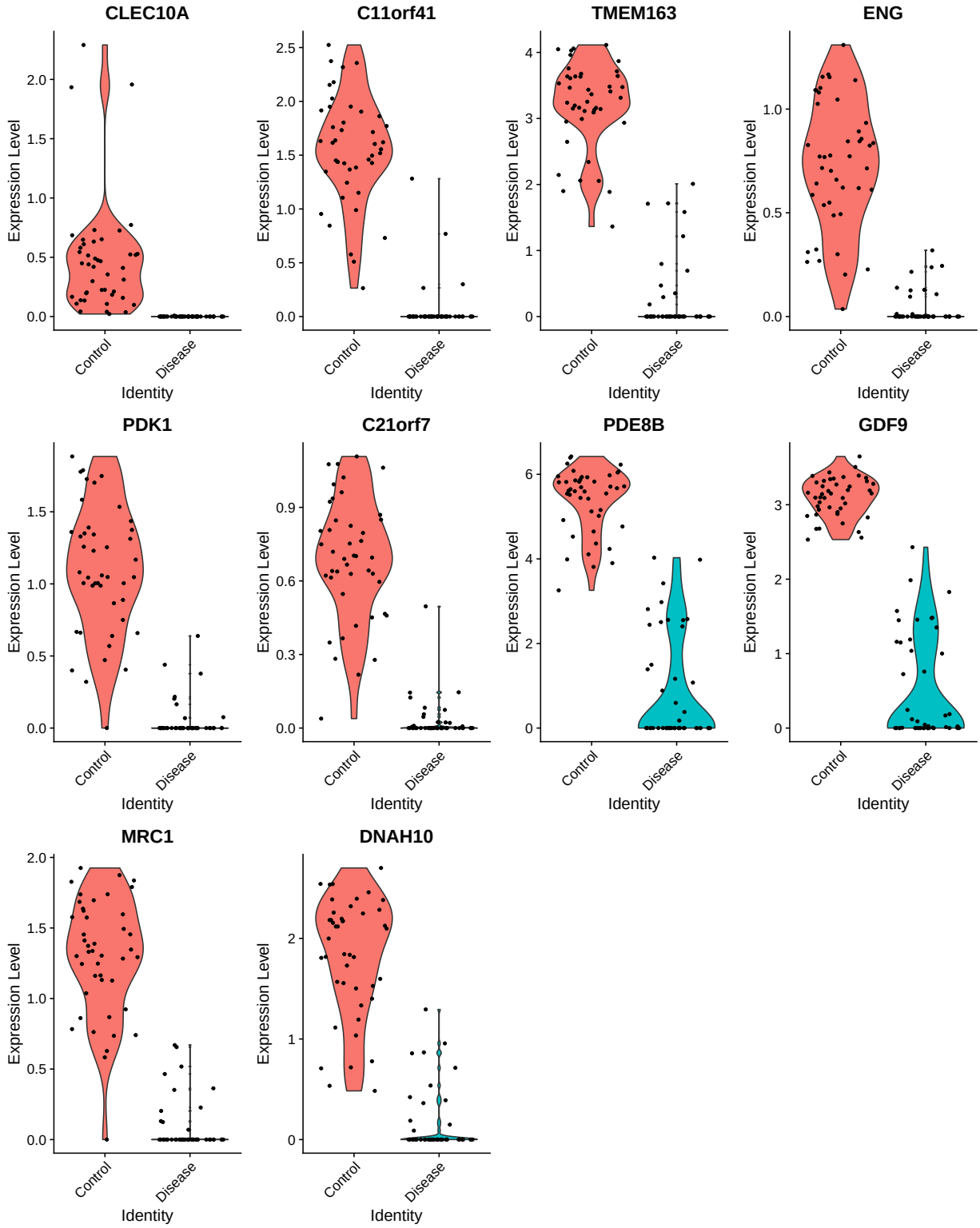
# top 50 DE genes
kable(cts_de_genes[1:50,])
```

	p_val	avg_logFC	pct.1	pct.2	p_val_adj
CLEC10A	0	0.6535568	1.000	0.022	0
C11orf41	0	1.5716403	1.000	0.087	0
TMEM163	0	2.9257802	1.000	0.239	0
ENG	0	0.7380097	1.000	0.239	0
PDK1	0	1.1270502	0.977	0.174	0
C21orf7	0	0.6862599	1.000	0.304	0
PDE8B	0	3.7330361	1.000	0.413	0
GDF9	0	2.3430204	1.000	0.587	0
MRC1	0	1.2680825	0.977	0.239	0
DNAH10	0	1.7650057	1.000	0.261	0
TMEM200A	0	1.5258964	0.909	0.000	0
FBLN5	0	1.6605395	0.955	0.109	0
FAM46C	0	2.5273579	1.000	0.500	0

	p_val	avg_logFC	pct.1	pct.2	p_val_adj
TOX2	0	2.3288898	1.000	0.370	0
LYPD1	0	0.8254808	1.000	0.304	0
TESC	0	1.3170113	0.977	0.217	0
COX4I2	0	0.4068809	1.000	0.283	0
GPR137B	0	1.6367645	0.955	0.109	0
PTTG1	0	1.7710574	1.000	1.000	0
ALPL	0	-3.1992330	0.227	0.978	0
TCTN1	0	0.9041435	0.977	0.130	0
TRIM36	0	1.2468499	1.000	0.348	0
PRR15	0	1.1594417	1.000	0.326	0
LAMA1	0	-3.1948661	0.409	1.000	0
ABCC6	0	1.4735055	1.000	0.348	0
MTP18	0	-0.4862883	0.295	0.978	0
ELOVL4	0	1.3220434	0.977	0.283	0
RND1	0	0.9051316	0.977	0.239	0
CRMP1	0	1.9563517	0.955	0.130	0
DEPDC7	0	1.8240991	1.000	0.391	0
NME1-NME2	0	-1.6608055	1.000	1.000	0
SLC37A4	0	-0.6649210	0.318	0.978	0
KALRN	0	2.7026124	0.977	0.261	0
CD36	0	0.6631757	0.977	0.304	0
PARVG	0	0.5623636	0.886	0.022	0
KRT18	0	-3.2098391	1.000	1.000	0
ST6GAL1	0	-3.2228267	0.341	0.978	0
LRRK1	0	1.0659505	0.909	0.087	0
MCTP1	0	2.2007034	1.000	0.304	0
PRODH	0	-1.3270007	0.250	1.000	0
ELMO1	0	0.8721151	1.000	0.522	0
HSF2BP	0	2.3913629	1.000	0.652	0
SERHL2	0	1.5584741	1.000	0.609	0
HSPA12A	0	0.6791781	0.864	0.022	0
SLCO3A1	0	2.0170152	0.955	0.174	0
NR2F6	0	-1.2644049	0.409	1.000	0
S100A11	0	-0.5290551	0.614	1.000	0
RPL13AP20	0	-0.3179660	1.000	1.000	0
AKAP12	0	-1.9997725	0.455	1.000	0
PVRL2	0	-1.7457315	0.386	0.978	0

```
#DT::datatable(cts_de_genes, extensions = c('FixedColumns','Buttons'),
#              options = list(
#                pageLength = 5,
#                scrollX = TRUE,
#                scrollCollapse = TRUE,
#                dom = 'Bfrtip',
#                buttons = c('copy', 'csv', 'excel')
#              ))
```

Violin plot for top 10 DEGs



Functional enrichment analysis for DEGs

GO Biological Process

```
# This select genes of adj.p.value < 0.05,
# sometimes people use different threshold, like adding log2foldchange threshold.


enriched_combined <- enrichr(rownames(cts_de_genes[which(cts_de_genes$p_val_adj < 0.05
                                                         & abs(cts_de_genes$avg_logFC) > 1.5),]),dbs)
```

```
## Uploading data to Enrichr... Done.
##   Querying GO_Molecular_Function_2018... Done.
##   Querying GO_Cellular_Component_2018... Done.
##   Querying GO_Biological_Process_2018... Done.
##   Querying KEGG_2019_Human... Done.
## Parsing results... Done.
```

```
kable(head(enriched_combined$GO_Biological_Process_2018,n=20)[,c(-3,-5,-6,-7,-9)])
```

Term	Overlap	Adjusted.P.value	Combined.Score
membrane raft assembly (GO:0001765)	3/6	0.0367183	822.36560
membrane raft organization (GO:0031579)	3/10	0.1078483	419.64442
membrane assembly (GO:0071709)	3/23	0.9907331	134.93264
amino acid import across plasma membrane (GO:0089718)	2/7	1.0000000	271.96295
regulation of endothelial cell migration (GO:0010594)	4/69	1.0000000	52.08753
positive regulation of cell-matrix adhesion (GO:0001954)	3/33	1.0000000	80.50138
regulation of sequestering of calcium ion (GO:0051282)	2/9	1.0000000	195.18295
Cdc42 protein signal transduction (GO:0032488)	2/9	1.0000000	195.18295
Rho protein signal transduction (GO:0007266)	4/72	1.0000000	48.70299
cholesterol biosynthetic process (GO:0006695)	3/35	1.0000000	73.85980
secondary alcohol biosynthetic process (GO:1902653)	3/36	1.0000000	70.86179
sterol biosynthetic process (GO:0016126)	3/40	1.0000000	60.61073
vesicle budding from membrane (GO:0006900)	2/12	1.0000000	132.68425
prostanoid metabolic process (GO:0006692)	2/12	1.0000000	132.68425
nucleotide metabolic process (GO:0009117)	2/13	1.0000000	119.00904
cellular response to acid chemical (GO:0071229)	4/91	1.0000000	33.34696
L-alpha-amino acid transmembrane transport (GO:1902475)	2/15	1.0000000	97.81109
Ras protein signal transduction (GO:0007265)	6/223	1.0000000	19.37798
positive regulation of cell-substrate adhesion (GO:0010811)	3/52	1.0000000	40.67555
regulation of sprouting angiogenesis (GO:1903670)	2/19	1.0000000	70.36431

```
## output top 20 enriched terms
#DT::datatable(head(enriched_combined$GO_Biological_Process_2018,n=20)[,c(-3,-5,-6,-7,-9)],
#              extensions = c('FixedColumns','Buttons'),
#              options = list(
#                pageLength = 5,
#                scrollX = TRUE,
#                scrollCollapse = TRUE,
#                dom = 'Bfrtip',
```

```
#                 buttons = c('copy', 'csv', 'excel')
#               ))
```

GO Cellular Component

```
kable(head(enriched_combined$GO_Cellular_Component_2018,n=20)[,c(-3,-5,-6,-7,-9,-9)])
```

Term	Overlap	Adjusted.P.value	Combined.Score
actin cytoskeleton (GO:0015629)	8/294	0.5931489	25.028831
focal adhesion (GO:0005925)	8/356	0.9625108	16.995574
perinuclear region of cytoplasm (GO:0048471)	8/378	0.9135826	14.967925
pigment granule (GO:0048770)	2/21	1.0000000	61.069717
melanosome (GO:0042470)	2/21	0.8816241	61.069717
membrane raft (GO:0045121)	4/119	0.8035784	21.135574
cytoskeleton (GO:0005856)	9/520	0.8350038	10.420019
serine/threonine protein kinase complex (GO:1902554)	2/25	0.7725355	47.543941
alpha DNA polymerase:primase complex (GO:0005658)	1/6	1.0000000	73.143198
clathrin vesicle coat (GO:0030125)	1/6	1.0000000	73.143198
HFE-transferrin receptor complex (GO:1990712)	1/8	1.0000000	49.986469
clathrin coat of trans-Golgi network vesicle (GO:0030130)	1/8	1.0000000	49.986469
apical dendrite (GO:0097440)	1/8	1.0000000	49.986469
SCF ubiquitin ligase complex (GO:0019005)	2/54	1.0000000	14.667201
NMDA selective glutamate receptor complex (GO:0017146)	1/9	1.0000000	42.669594
nuclear matrix (GO:0016363)	2/59	1.0000000	12.691086
fibrillar center (GO:0001650)	3/131	1.0000000	8.509127
polymeric cytoskeletal fiber (GO:0099513)	4/221	1.0000000	6.483701
spindle pole centrosome (GO:0031616)	1/11	1.0000000	32.467308
secretory granule lumen (GO:0034774)	5/317	1.0000000	5.549493

```
#DT::datatable(head(enriched_combined$GO_Cellular_Component_2018,n=20)[,c(-3,-5,-6,-7,-9)],
#extensions = c('FixedColumns','Buttons'),
#               options = list(
#                 pageLength = 5,
#                 scrollX = TRUE,
#                 scrollCollapse = TRUE,
#                 dom = 'Bfrtip',
#                 buttons = c('copy', 'csv', 'excel')
#               ))
```

GO Molecular Function

```
kable(head(enriched_combined$GO_Molecular_Function_2018,n=20)[,c(-3,-5,-6,-7,-9)])
```

Term	Overlap	Adjusted.P.value	Combined.Score
phospholipase inhibitor activity (GO:0004859)	2/8	1.000000	228.142858
cyclic-nucleotide phosphodiesterase activity (GO:0004112)	2/10	1.000000	169.597477

Term	Overlap	Adjusted.P.value	Combined.Score
tropomyosin binding (GO:0005523)	2/14	1.000000	107.543403
transaminase activity (GO:0008483)	2/14	1.000000	107.543403
calcium ion binding (GO:0005509)	7/284	1.000000	18.454783
calcium-dependent phospholipid binding (GO:0005544)	3/47	0.901937	47.516290
cadherin binding involved in cell-cell adhesion (GO:0098641)	2/19	1.000000	70.364312
3',5'-cyclic-nucleotide phosphodiesterase activity (GO:0004114)	2/20	1.000000	65.448022
protein binding involved in cell-cell adhesion (GO:0098632)	2/21	1.000000	61.069717
core promoter proximal region DNA binding (GO:0001159)	2/22	1.000000	57.149674
carboxylic acid transmembrane transporter activity (GO:0046943)	3/64	1.000000	29.327706
GTPase regulator activity (GO:0030695)	6/275	1.000000	12.775555
metal ion binding (GO:0046872)	8/442	1.000000	10.582974
calmodulin-dependent protein kinase activity (GO:0004683)	2/27	1.000000	42.503233
protein homodimerization activity (GO:0042803)	10/664	1.000000	7.982032
non-membrane spanning protein tyrosine kinase activity (GO:0004715)	2/43	1.000000	21.065193
arginine transmembrane transporter activity (GO:0015181)	1/6	1.000000	73.143198
L-lysine transmembrane transporter activity (GO:0015189)	1/6	1.000000	73.143198
prostaglandin E receptor activity (GO:0004957)	1/6	1.000000	73.143198
L-ornithine transmembrane transporter activity (GO:0000064)	1/6	1.000000	73.143198

```
#DT::datatable(head(enriched_combined$GO_Molecular_Function_2018,n=20)[,c(-3,-5,-6,-7,-9)],
#extensions = c('FixedColumns','Buttons'),
#              options = list(
#                pageLength = 5,
#                scrollX = TRUE,
#                scrollCollapse = TRUE,
#                dom = 'Bfrtip',
#                buttons = c('copy', 'csv', 'excel')
#              ))
```

KEGG pathway

```
kable(head(enriched_combined$KEGG_2019_Mouse,n=20)[,c(-3,-5,-6,-7,-9)])
```

|| || || ||

```
#DT::datatable(head(enriched_combined$KEGG_2019_Mouse,n=20)[,c(-3,-5,-6,-7,-9)],
#extensions = c('FixedColumns','Buttons'),
#              options = list(
#                pageLength = 5,
#                scrollX = TRUE,
#                scrollCollapse = TRUE,
#                dom = 'Bfrtip',
#                buttons = c('copy', 'csv', 'excel')
#              ))
```

Session Infomation

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=Chinese (Simplified)_China.936
## [2] LC_CTYPE=Chinese (Simplified)_China.936
## [3] LC_MONETARY=Chinese (Simplified)_China.936
## [4] LC_NUMERIC=C
## [5] LC_TIME=Chinese (Simplified)_China.936
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.28   DT_0.13      enrichR_2.1  Seurat_3.1.5
##
## loaded via a namespace (and not attached):
##   [1] nlme_3.1-147        tsne_0.1-3          RcppAnnoy_0.0.16
##   [4] RColorBrewer_1.1-2  httr_1.4.1          sctransform_0.2.1
##   [7] tools_4.0.0         R6_2.4.1            irlba_2.3.3
##  [10] KernSmooth_2.23-17  uwot_0.1.8          lazyeval_0.2.2
##  [13] colorspace_1.4-1    withr_2.2.0         npsurv_0.4-0.1
##  [16] tidyselect_1.0.0    gridExtra_2.3       curl_4.3
##  [19] compiler_4.0.0      plotly_4.9.2.1      labeling_0.3
##  [22] scales_1.1.1        lmtest_0.9-37       ggridges_0.5.2
##  [25] pbapply_1.4-2       rappdirs_0.3.1      stringr_1.4.0
##  [28] digest_0.6.25       rmarkdown_2.1       pkgconfig_2.0.3
##  [31] htmltools_0.4.0     highr_0.8           limma_3.44.1
##  [34] htmlwidgets_1.5.1   rlang_0.4.6         farver_2.0.3
##  [37] zoo_1.8-8           jsonlite_1.6.1      ica_1.0-2
##  [40] dplyr_0.8.5         magrittr_1.5        patchwork_1.0.0
##  [43] Matrix_1.2-18       Rcpp_1.0.4.6        munsell_0.5.0
##  [46] ape_5.3             reticulate_1.15     lifecycle_0.2.0
##  [49] stringi_1.4.6       yaml_2.2.1          MASS_7.3-51.6
##  [52] Rtsne_0.15          plyr_1.8.6          grid_4.0.0
##  [55] parallel_4.0.0      listenv_0.8.0       ggrepel_0.8.2
##  [58] crayon_1.3.4        lattice_0.20-41     cowplot_1.0.0
##  [61] splines_4.0.0       pillar_1.4.4        igraph_1.2.5
##  [64] rjson_0.2.20        future.apply_1.5.0  reshape2_1.4.4
##  [67] codetools_0.2-16    leiden_0.3.3        glue_1.4.0
##  [70] evaluate_0.14       lsei_1.2-0.1        data.table_1.12.8
##  [73] png_0.1-7           vctrs_0.3.1         gtable_0.3.0
##  [76] RANN_2.6.1          purrr_0.3.4         tidyr_1.0.3
##  [79] future_1.17.0       assertthat_0.2.1    ggplot2_3.3.0
##  [82] xfun_0.13           rsvd_1.0.3          RSpectra_0.16-0
```

```
## [85] survival_3.1-12     viridisLite_0.3.0   tibble_3.0.1
## [88] cluster_2.1.0       globals_0.12.5      fitdistrplus_1.0-14
## [91] ellipsis_0.3.0      ROCR_1.0-11
```