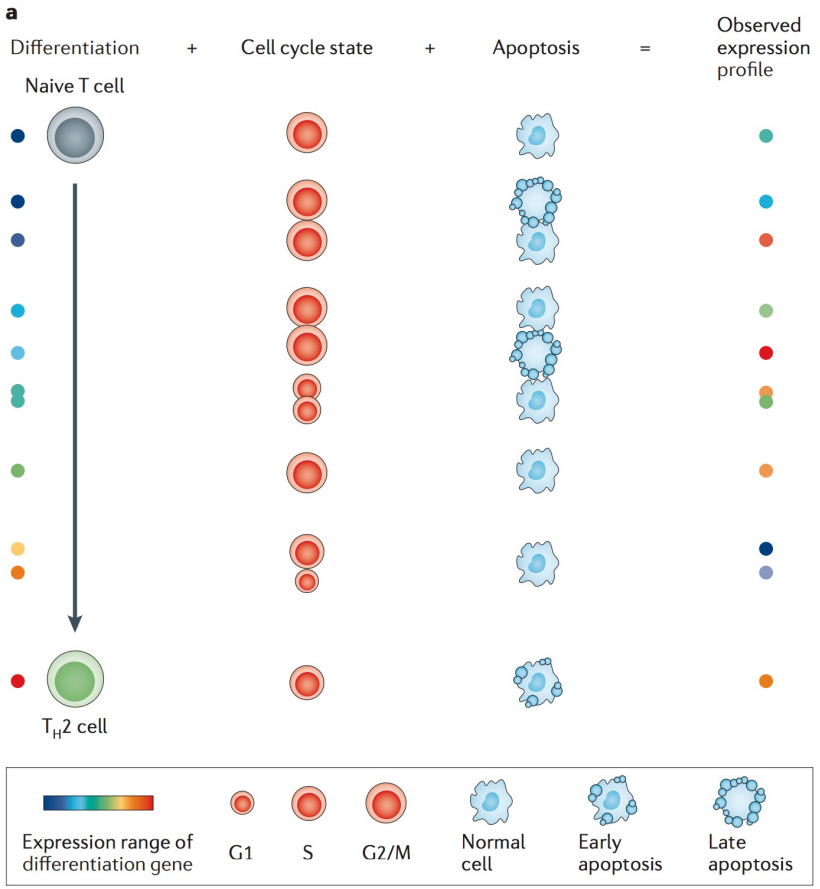
Introduction to biological analyzing methods

scRNA-Seq vs Bulk/tissue RNA-Seq

- Bulk RNA-Seq data (tissue)
  - > Mixed sum of signals from individual cells
  - > Large number of cell state specific expression are averaged
  - > H_0: gene A in two conditions have the same expression level
  - > 10^1~10^3 samples
- scRNA-seq data
  - > An outcome of the regulation signal in the current cell
  - > Cell type specific transcriptomics heterogeneity
  - > Differed regulatory states
  - > 10^2~10^6 Cells



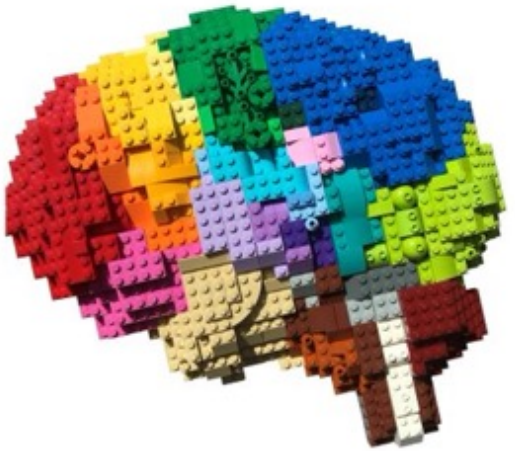
Oliver Stegle et al 2014

scRNA-Seq vs Bulk/tissue RNA-Seq

bulk RNA-seq

single-cell RNA-seq

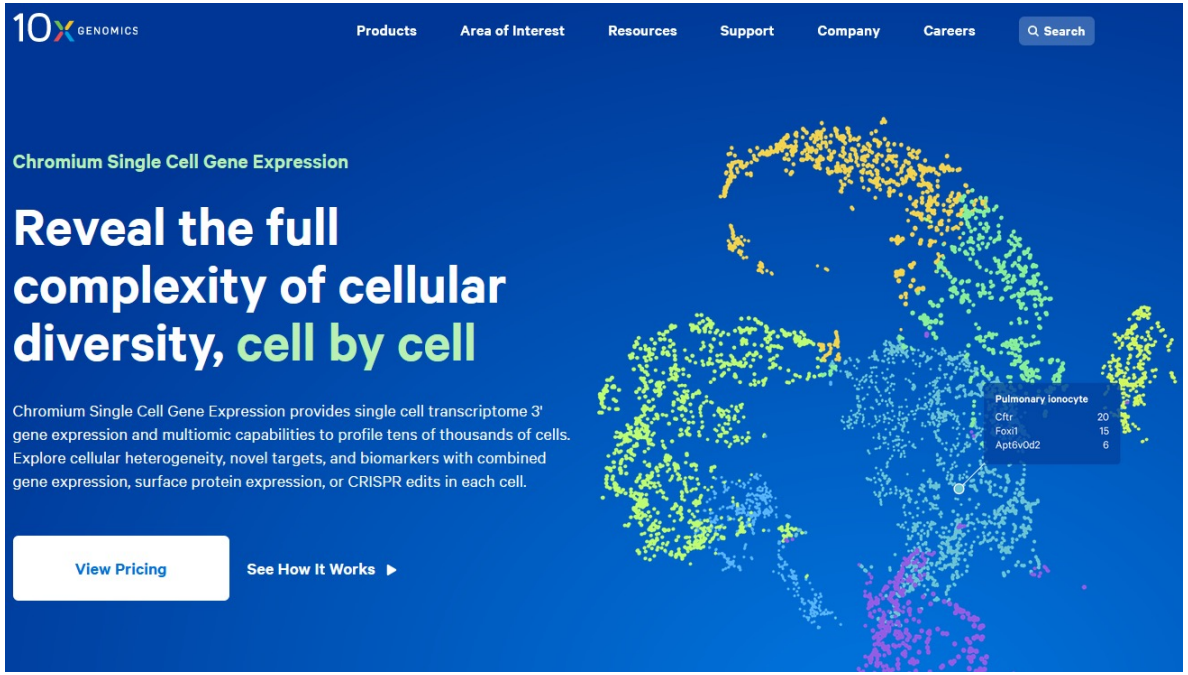
spatial transcriptomics



functional tissue

scRNA-seq platform

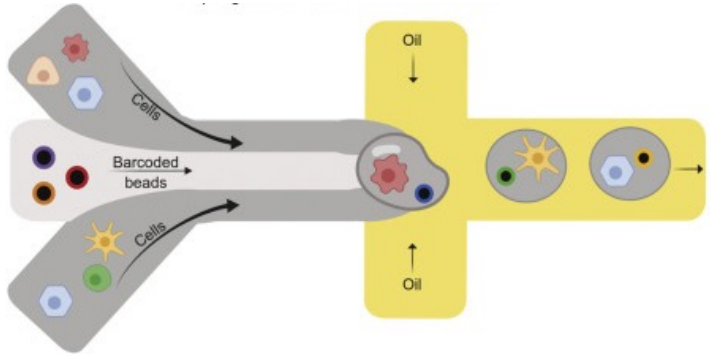
- Currently, essentially 10X Genomics Chromium Single Cell Gene Expression platform dominates the market and is considered to be *de facto* standard for single cell RNA-seq experiments.



https://www.10xgenomics.com/products/single-cell-gene-expression

Sequencing and output data

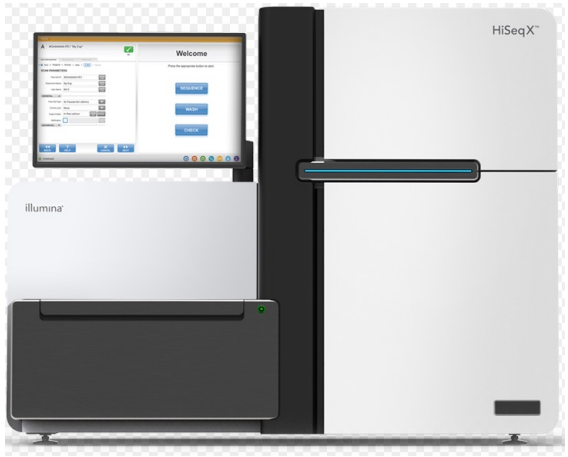
Single cell isolation



(UMI) counts

	Cell1	Cell2	...	CellN
Gene1	3	2	.	13
Gene2	2	3	.	1
Gene3	1	14	.	18
...	.	.	.	.
...	.	.	.	.
...	.	.	.	.
GeneM	25	0	.	0

Short read sequencing

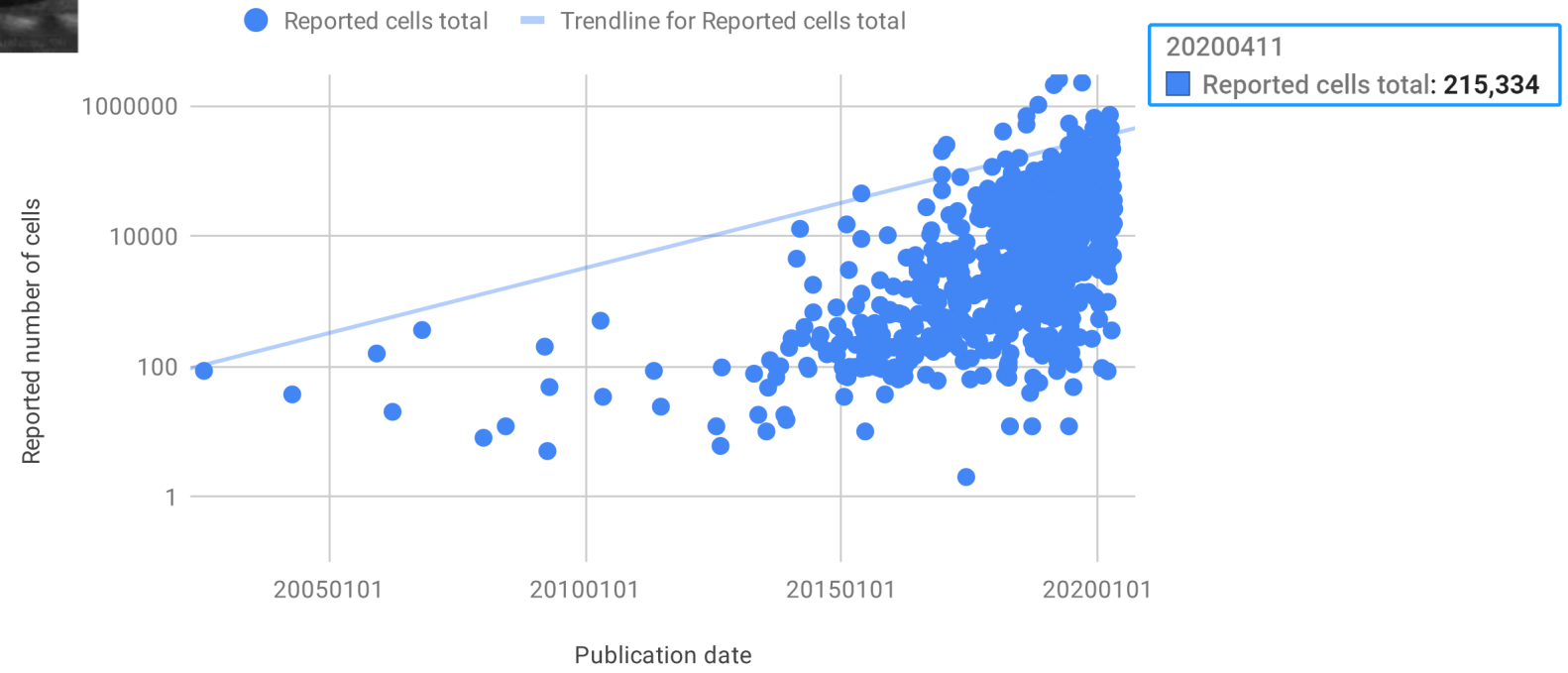


Single-cell data is the ideal big data

The Curse of Dimensionality:

Inability to perform meaningful mathematics
when the number of features, **p**,
is much greater than the number of samples, **n**

In single cell studies, we can generate
p~20,000 genes and **n~200,000 cells**



www.nxn.se/single-cell-studies/gui

Online resources

scRNA-Seq tool archives

https://www.scrna-tools.org/ (681 tools, 0703/2020)
https://github.com/seandavi/awesome-single-cell

scRNA-Seq database

https://www.ebi.ac.uk/gxa/sc/home (EBI)
https://singlecell.broadinstitute.org/single_cell
https://panglaodb.se/index.html (Neuron)
http://biocc.hrbmu.edu.cn/CancerSEA/ (Cancer)

scRNA-Seq online courses/tutorial

https://scrnaseq-course.cog.sanger.ac.uk/website/index.html
https://canvas.harvard.edu/courses/49497
https://osca.bioconductor.org/index.html
https://youtu.be/DY3FHQTNjJo

Single-cell common data format

General format

1 Expression matrix

	Cell1	Cell2	Cell3…
Xkr4	12	0	10
Gm1992	3	0	5
Gm37381	0	0	32
Rp1	11	22	0
Sox17	0	10	0
Gm37323	3	0	0
…	…	…	…

Output of 10X

2 10X folder

Barcodes

```
AAACCTGAGTACTTGC-1
AAACCTGCAAGTTGTC-1
AAACCTGTCAGGTTCA-1
AAACGGGAGAGAGCTC-1
AAACGGGAGCTACCGC-1
AAACGGGCAACGCACC-1
```

Features

```
ENSMUSG00000051951      Xkr4    Gene Expression
ENSMUSG00000089699      Gm1992  Gene Expression
ENSMUSG00000102343      Gm37381 Gene Expression
ENSMUSG00000025900      Rp1     Gene Expression
ENSMUSG00000025902      Sox17   Gene Expression
ENSMUSG00000104328      Gm37323 Gene Expression
```

Sparse matrix

```
31057 3980 6231653
31054 1 1
30980 1 3
30978 1 1
30975 1 1
30974 1 14
```

3 HDF5

```
(root)
├─ barcode_idx
├─ barcode_info [HDF5 group]
│   ├─ genomes
│   └─ pass_filter
├─ barcodes
├─ count
├─ feature_idx
├─ features     [HDF5 group]
│   ├─ _all_tag_keys
│   ├─ target_sets [for Targeted Gene Expression or Fixed RNA Profiling]
│   │    └─ [target set name]
│   ├─ feature_type
│   ├─ genome
│   ├─ id
│   ├─ name
│   ├─ pattern [Feature Barcode only]
│   ├─ read [Feature Barcode only]
│   └─ sequence [Feature Barcode only]
├─ gem_group
├─ library_idx
├─ library_info
├─ metrics_json [HDF5 dataset; see below]
├── probe_idx          ---------------------|
├── probes [HDF5 group]                     |
│   ├── feature_id                          | [For Fixed RNA Profiling,
│   ├── feature_name                        |
│   ├── probe_id                           |
│   └── region         ---------------------|
├─ umi
└─ umi_type
```

HDF5 detail can be found at: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/output/molecule_info

Section 2: Create a SeuratObject


```
# 2.1 read in 10X folder
pbmc.data<- Read10X(”./data1")

# 2.1 (optional) read in 10X h5 file; You can also read csv or txt files
pbmc.data <- Read10X_h5("5k_pbmc_v3_filtered_feature_bc_matrix.h5")

# 2.2 create Seurat Object
pbmc <- CreateSeuratObject(counts =  pbmc.data, min.cells = 3,min.features = 200)
pbmc
```

```
An object of class Seurat
13714 features across 2700 samples within 1 assay
Active assay: RNA (13714 features, 0 variable features)
 2 layers present: counts, data
```

About Seurat object:

There are Object Oriented Programming (OOP) system, including S3 & S4.

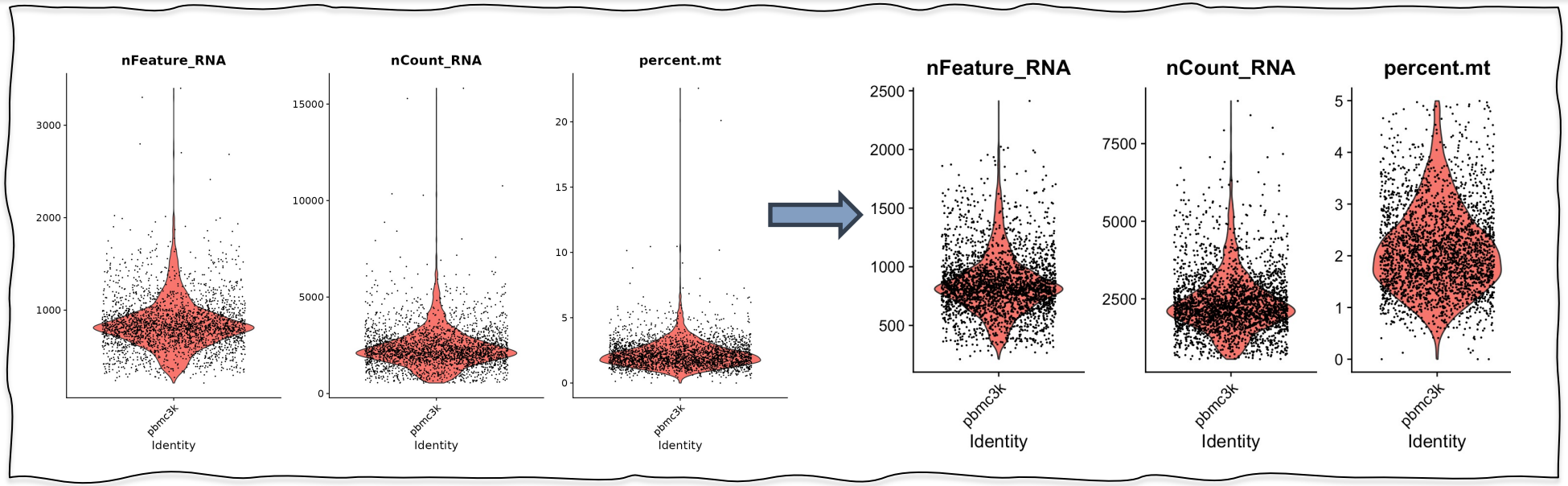
- S4 is more complex, formal, and strict than S3.
- S4 is suitable for a bigger
- Seurat belong to S4.

For advanced User: Seurat object structure.

https://github.com/satijalab/seurat/wiki

Section 3: processing data (QC, normalization, feature selection)


```
# 3.1 Quality control (QC)
# 3.1.1 calculate mitochondrial gene percentage for each cell. (^Mt- or ^mt-))
pbmc[["percent.mt"]] <- PercentageFeatureSet(pbmc, pattern = "^MT-")

# 3.1.2 Plot QC matrics
VlnPlot(pbmc, features = c("nFeature_RNA","nCount_RNA","percent.mt"),
                                ncol = 3)

pbmc <- subset(pbmc, subset = percent.mt < 5 & nFeature_RNA > 200 & nFeature_RNA < 2500)
```

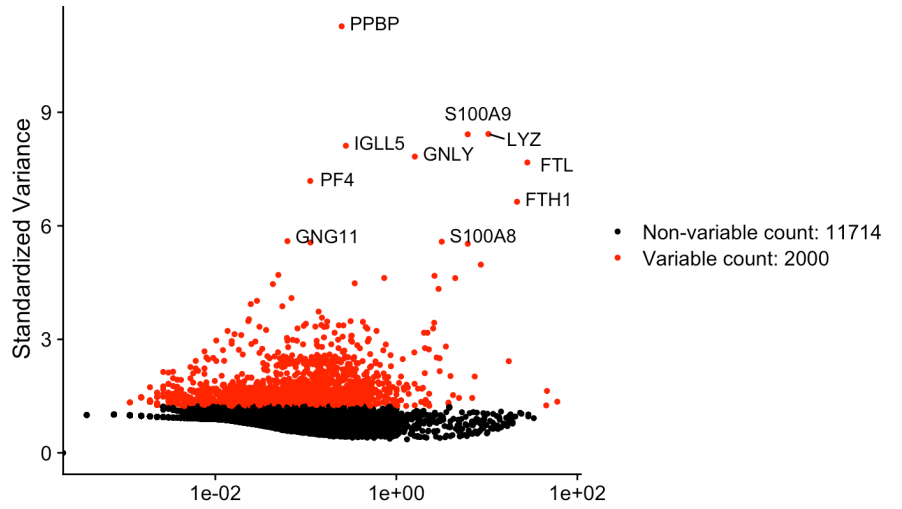


Ilicic, T., Kim, J.K., Kolodziejczyk, A.A. *et al.* Classification of low quality cells from single-cell RNA-seq data. *Genome Biol* **17,** 29 (2016). https://doi.org/10.1186/s13059-016-0888-1

Section 3: Processing data (QC, normalization, feature selection)


```
# 3.2 normalization (Or check tutorial for scTransform V2)
pbmc <-
 NormalizeData(pbmc, normalization.method = "LogNormalize", scale.factor = 10000
)
# 3.3 feature selection
pbmc  <- FindVariableFeatures(pbmc, selection.method = "vst", nfeatures = 2000)

top10 <- head(VariableFeatures(pbmc), 10)
plot1 <- VariableFeaturePlot(pbmc)
LabelPoints(plot = plot1, points = top10, repel = TRUE)
```

Feature selection refers to excluding uninformative genes such as those which exhibit no meaningful biological variation across samples.



Townes, F.W., Hicks, S.C., Aryee, M.J. *et al.* Feature selection and dimension reduction for single-cell RNA-Seq based on a multinomial model. *Genome Biol* **20,** 295 (2019). https://doi.org/10.1186/s13059-019-1861-6

Section 3: Processing data (QC, normalization, feature selection)


```
# 3.4 scaling data and remove unwanted variation
pbmc <- ScaleData(object =  pbmc, vars.to.regress = c("percent.mt"))
```

Scaling data :

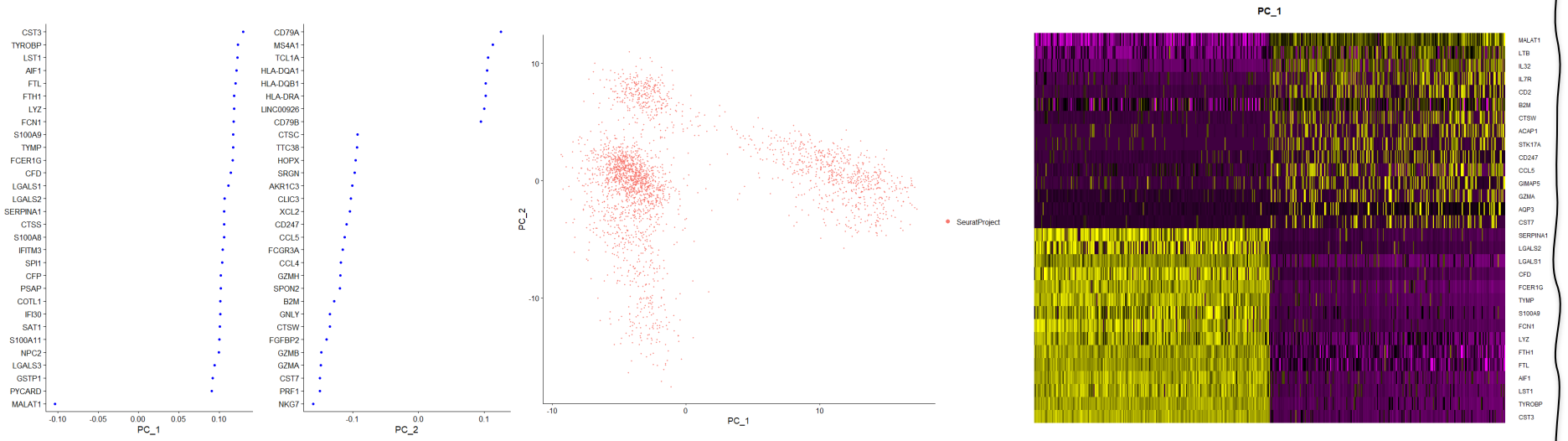
- Standard step before doing PCA.
- Variables (genes) shift to be zero centered.
- Variables (genes) should be scaled to have unit variance.

Regress out unwanted source: mt, ribo, etc.

Regressing these signals out of the analysis can improve downstream dimensionality reduction and clustering.

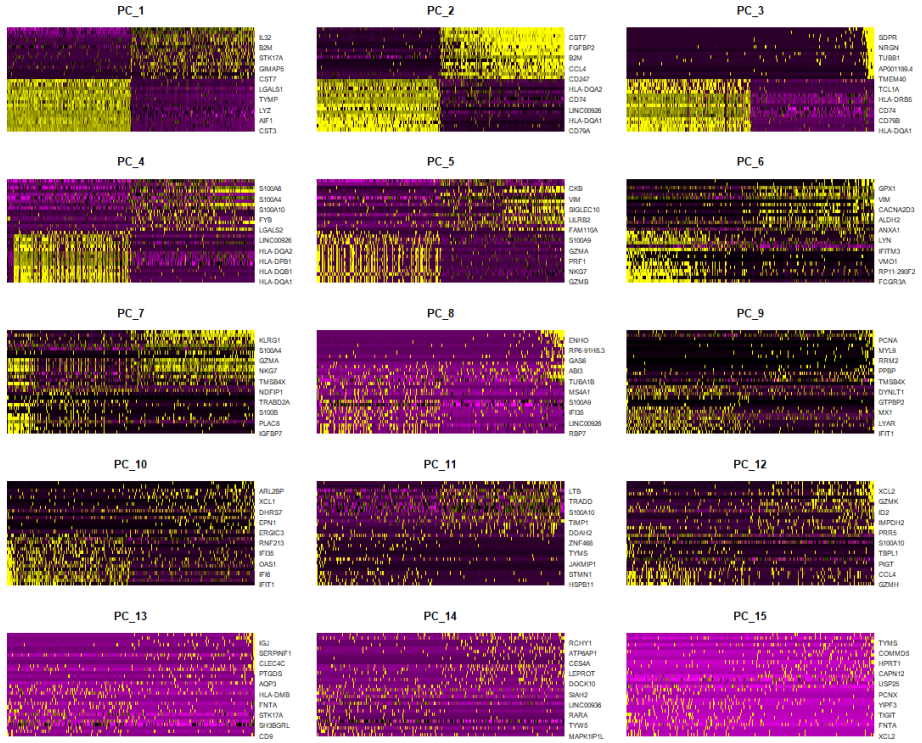
Section 4: linear dimension reduction

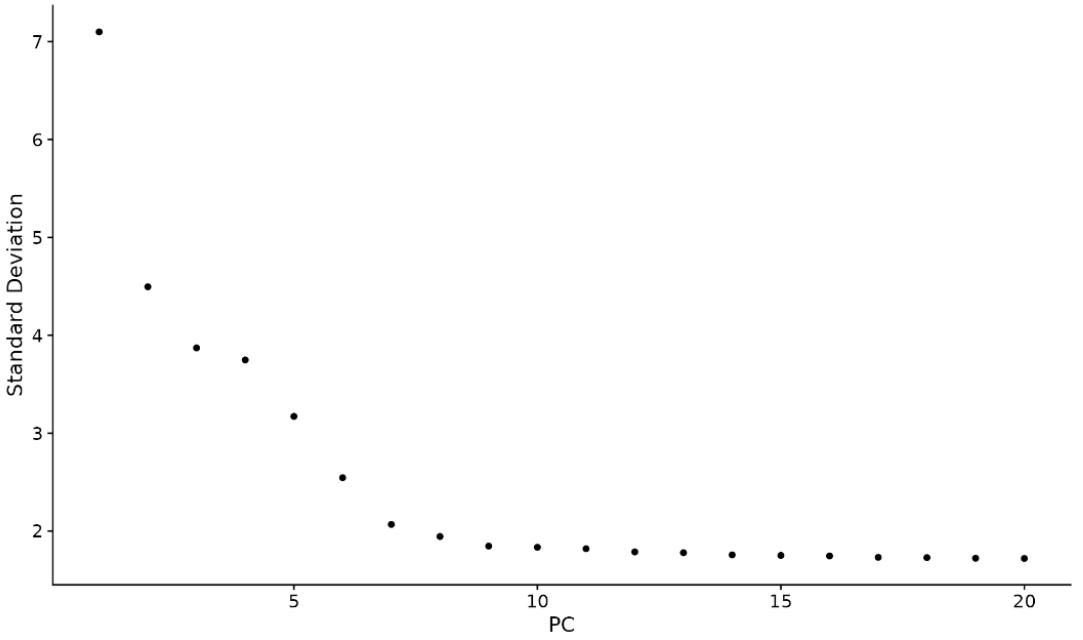

```
# 4.1 linear dimension reduction
SeuratObject <- RunPCA(pbmc, features = VariableFeatures(object = pbmc))
# 4.2 visualize top genes associated with reduction components
VizDimLoadings(pbmc, dims = 1:2, reduction = "pca")
# 4.3 visualize PCA results
DimPlot(object =  pbmc, reduction = "pca")
# 4.4 easy exploration of the primary sources of heterogeneity in a dataset
DimHeatmap(pbmc, dims = 1, cells = 500, balanced = TRUE)
```



Section 4: choosing reasonable PCs for further analysis

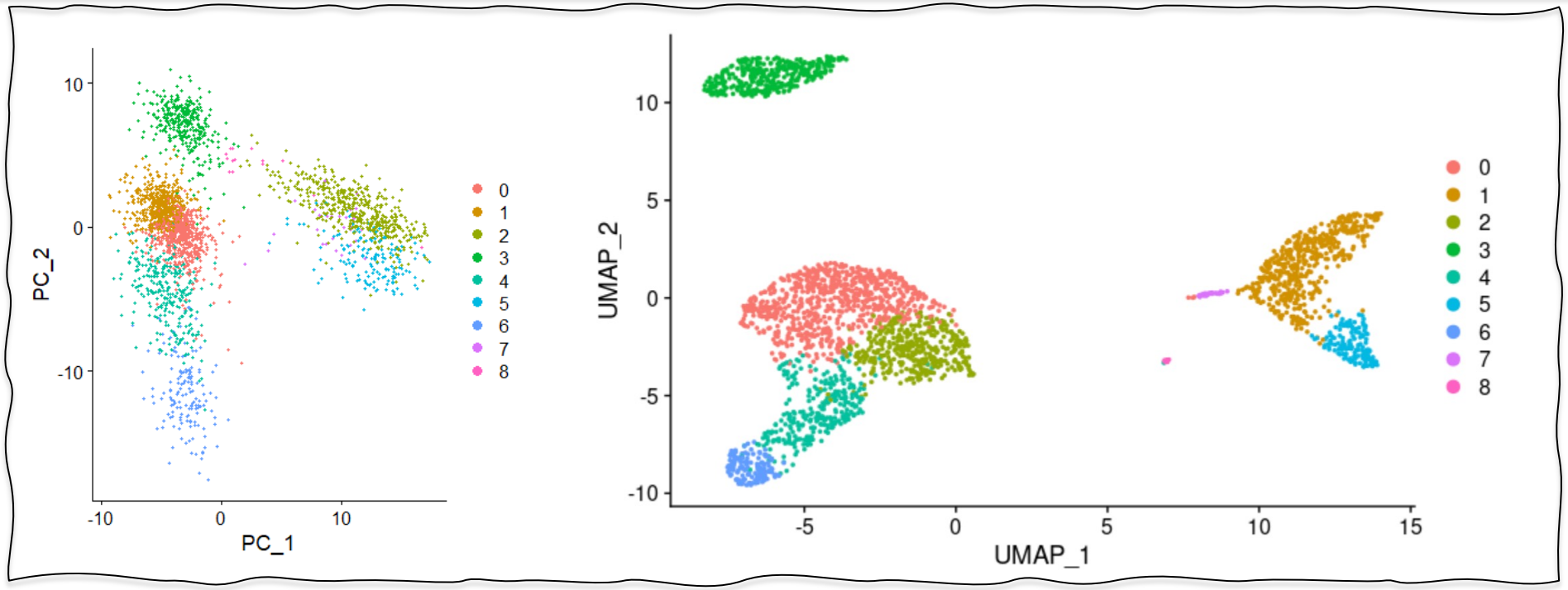

```
# method I: use DimHeapmap
DimHeatmap(pbmc, dims = 1:15, cells = 500, balanced = TRUE)

# method II: Elbow plot
ElbowPlot(pbmc)
```



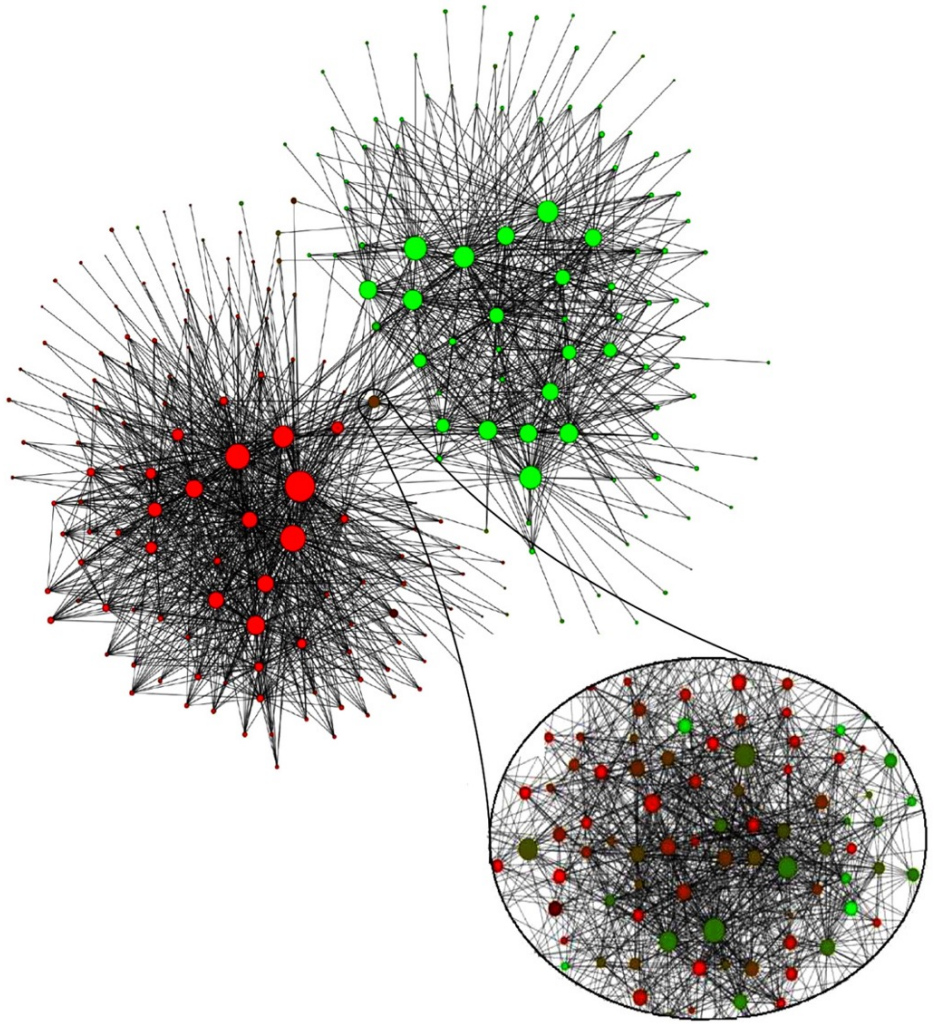


Section 5: UMAP & clustering


```
# 5.1 construct a KNN graph based on the euclidean distance in PCA space
pbmc  <- FindNeighbors(pbmc, dims = 1:7)
# 5.2 cluster based on Louvain alogorithm
pbmc  <- FindClusters(pbmc, resolution = 0.5)
# 5.3 non-linear dimensional reduction (UMAP)
pbmc  <- RunUMAP(pbmc, dims = 1:7)
# 5.4 visualization on UMAP
DimPlot(pbmc, reduction = "umap")
# use Idents(pbmc)<-"XX" to select the active meta information for plot.
```



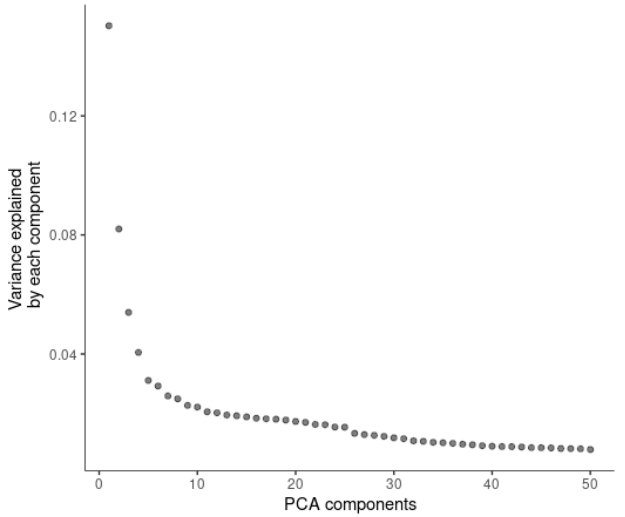
Louvain algorithm

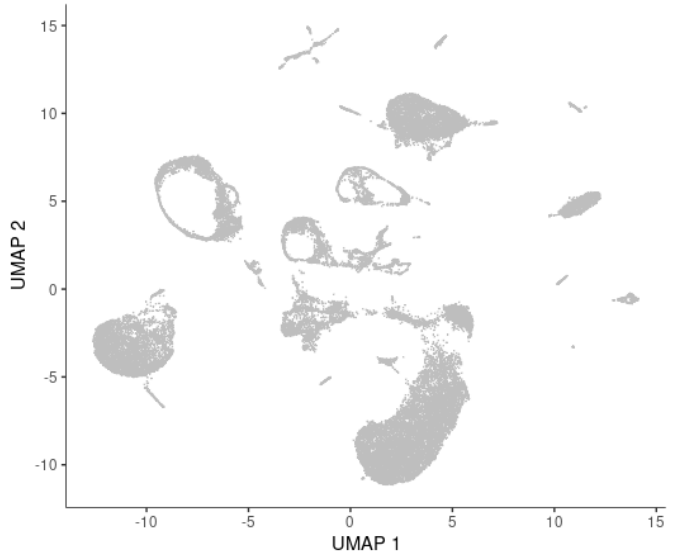


Selecting marker genes from the clusters

Processing and dimension reduction

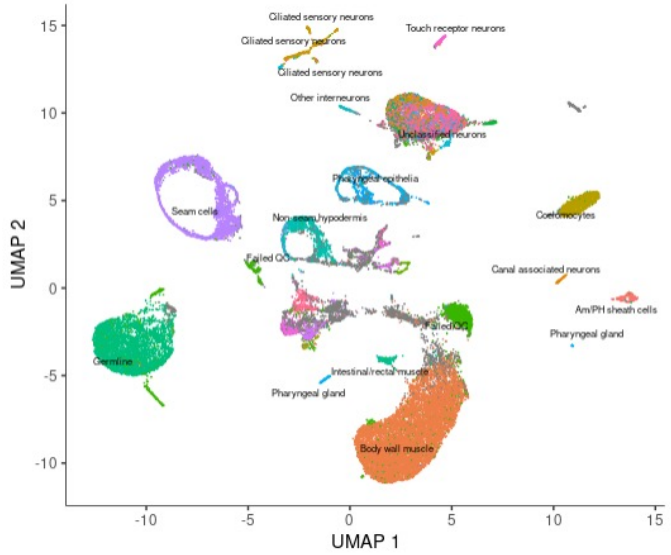

```
# preprocess and find low dimensions
cds <- preprocess_cds(cds, num_dim = 50)
# cds <- preprocess_cds(cds, method = c("PCA",
"LSI"),num_dim = 50, norm_method = c("log", "size_only",
"none"), use_genes = NULL,pseudo_count = NULL,scaling
= TRUE)

# Check elbow plot
plot_pc_variance_explained(cds)

# Reduce for 2 UMAP dimensions
cds <- reduce_dimension(cds, max_components = 2,

            reduction_method = "UMAP",

            preprocess_method = "PCA")

# Observe UMAP
plot_cells(cds)
```

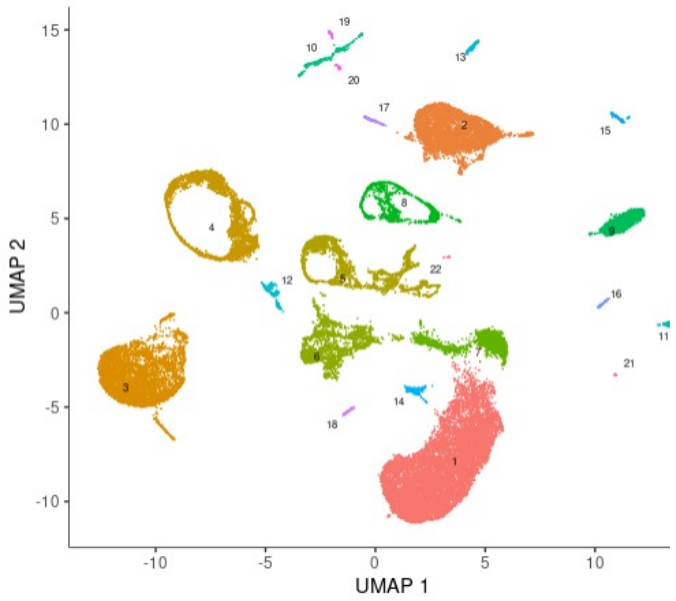


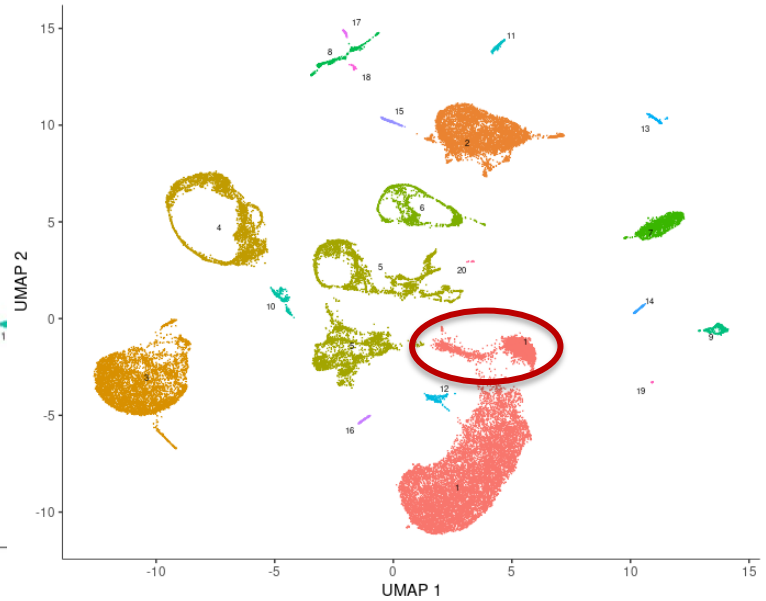


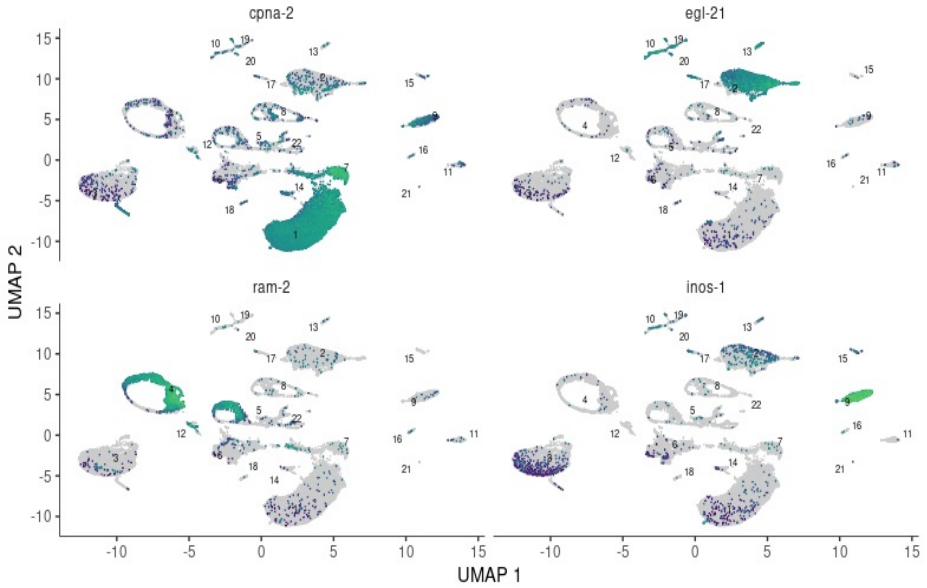
Clustering and UMAP visualization


```
# cluster cells
cds = cluster_cells(cds, resolution=1e-5)
# select genes to check feature plot
int_genes <- c("che-1","hlh-17","nhr-6","dmd-6","ceh-36","ham-1")
# more plots
plot_cells(cds, color_cells_by="cao_cell_type")
plot_cells(cds, color_cells_by="cluster")
plot_cells(cds, color_cells_by="partition")
plot_cells(cds, genes=int_genes)
```



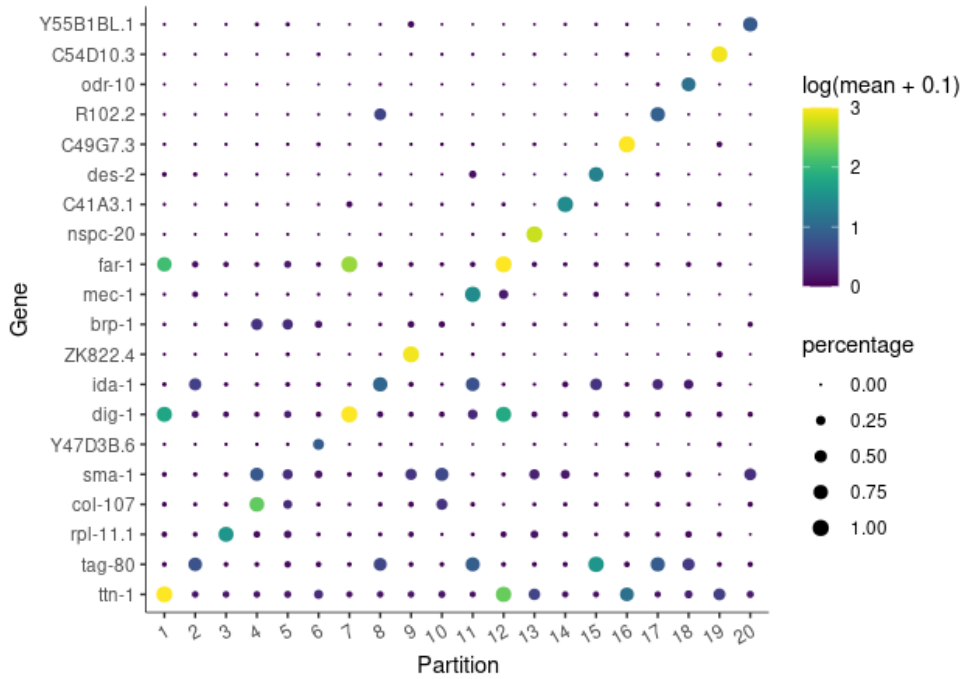






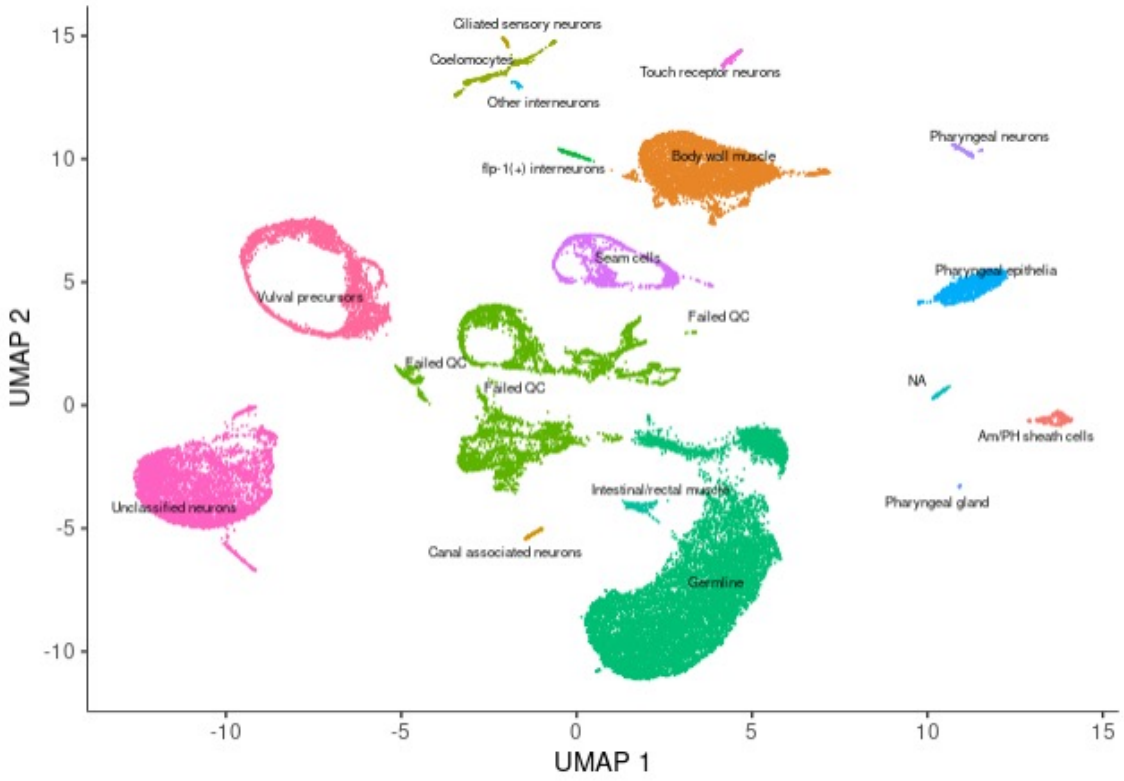
Find differentially expressed genes


```
# find DE genes for each "partition" (merged cell cluster)
marker_test_res <- top_markers(cds, group_cells_by="partition", reference_cells=1000, cores=8)
# select top DE genes
top_specific_markers <- marker_test_res %>%
                            filter(fraction_expressing >= 0.10) %>%
                            group_by(cell_group) %>%
                            top_n(1, pseudo_R2)
# remove repeated DE genes among clusters
top_specific_marker_ids <- unique(top_specific_markers %>%
                                      pull(gene_id))
# draw plot
plot_genes_by_group(cds, top_specific_marker_ids,
                            group_cells_by="partition",
                            ordering_type="maximal_on_diag",
                            max.size=3)
```



Annotate your data


```
# Create a new column in colData(cds)
colData(cds)$assigned_cell_type <- as.character(partitions(cds))
# Assign cell labels
colData(cds)$assigned_cell_type <- dplyr::recode(colData(cds)$anno_cell_type,
                                  "1"="Germline",
                                   "2"="Body wall muscle",
                                     "3"="Unclassified neurons

# Check your relabeled UMAP
plot_cells(cds, group_cells_by="partition",
                        color_cells_by="anno_cell_type")
```



singleR

Reference-Based Single-Cell RNA-Seq Annotation

Performs unbiased cell type recognition from single-cell RNA sequencing data, by leveraging reference transcriptomic datasets of pure cell types to infer the cell of origin of each single cell independently.

Installation

```
devtools::install_github('dviraran/SingleR')
# this might take long, though mostly because of the installation of Seurat.
```

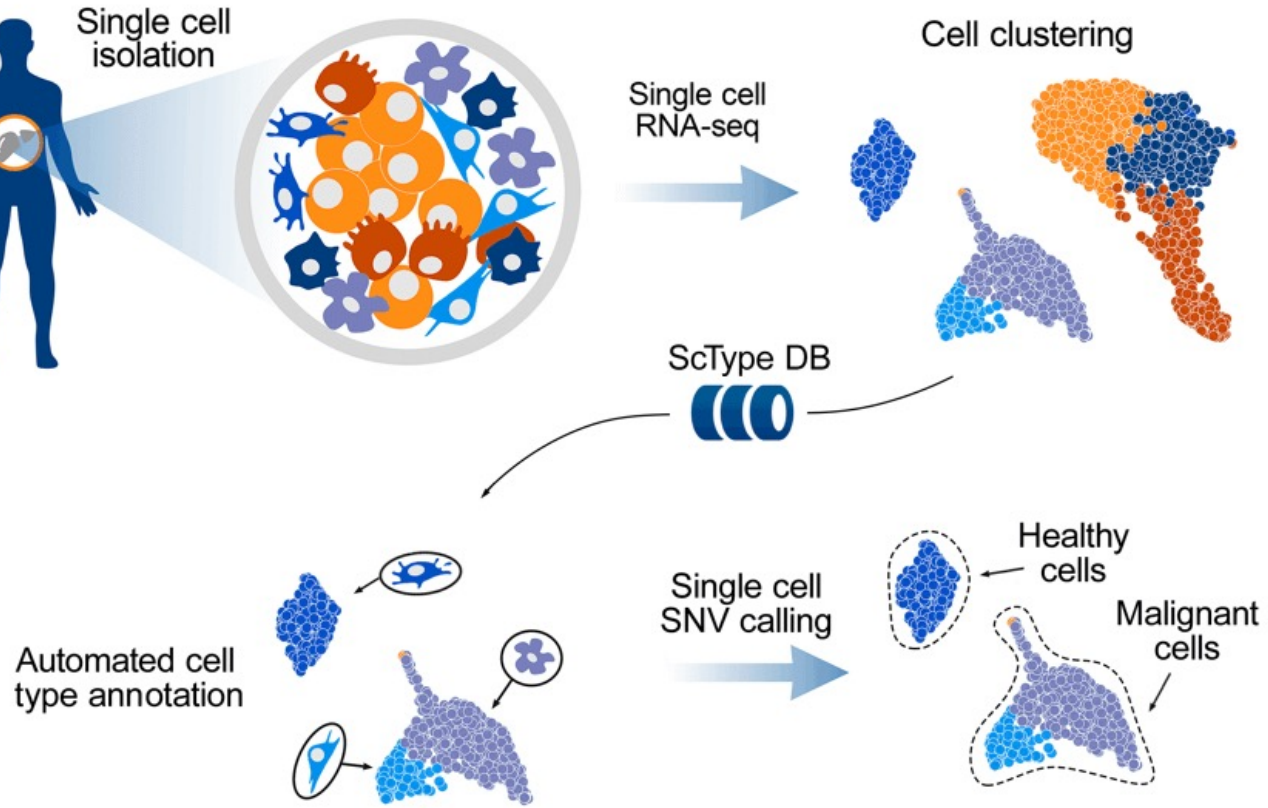
Aran D, Looney AP, Liu L, Wu E, Fong V, Hsu A, Chak S, Naikawadi RP, Wolters PJ, Abate AR, Butte AJ, Bhattacharya M (2019). “Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage.” *Nat. Immunol.*, **20**, 163-172. doi:10.1038/s41590-018-0276-y.

```
library(SingleR)

# Simplest use is running the wrapper function that creates both a SingleR and Seurat object:

# counts.file maybe a tab delimited text file, 10X directory or a matrix. annot is a tab delimited
# text file or a data.frame with the original identities. normalize.gene.length should be true if
# the data comes from a full-length platform. min.genes, min.cells, npca and regress.out are passed
# to Seurat to create a Seurat object object:
singler = CreateSinglerSeuratObject(counts.file, annot, project.name,
  min.genes = 500, technology, species = "Human" (or "Mouse"), citation,
  normalize.gene.length = F, min.cells = 2, npca = 10
  regress.out = "nUMI", reduce.seurat.object = T)

# The object can then be saved and uploaded to the SingleR web-app for further analysis and visualization or using functions available in the SingleR package (see vignette).
save(singler,file=paste0(project.name,'.RData')
```

scType

ScType is a computational platform, which enables data-driven, fully-automated and ultra-fast cell-type identification based solely on given scRNA-seq data, combined with our comprehensive cell marker database as background information.

https://sctype.app/



Citation: Ianevski, A., Giri, A.K. & Aittokallio, T. Fully-automated and ultra-fast cell-type identification using specific marker combinations from single-cell transcriptomic data. Nat Commun 13, 1246 (2022). https://doi.org/10.1038/s41467-022-28803-w

GPTCelltype – Use GPT-4 to annotate cell type

nature methods

Explore content ∨ About the journal ∨ Publish with us ∨

nature › nature methods › brief communications › article

Brief Communication | Open access | Published: 25 March 2024

Assessing GPT-4 for cell type annotation in single-cell RNA-seq analysis

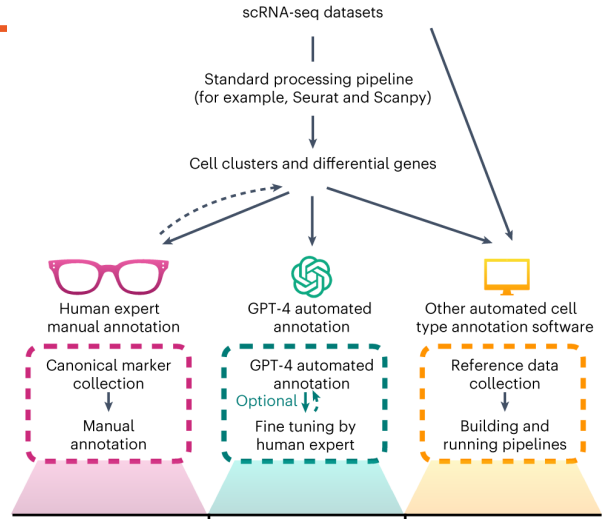
Wenpin Hou & Zhicheng Ji

Nature Methods (2024) | Cite this article

1 Citations | 290 Altmetric | Metrics

- They demonstrate that the large language model GPT-4 can accurately annotate cell types using marker gene information in single-cell RNA sequencing analysis.
- When evaluated across hundreds of tissue and cell types, GPT-4 generates cell type annotations exhibiting strong concordance with manual annotations.
- This capability can considerably reduce the effort and expertise required for cell type annotation

a



Human expert manual annotation	GPT-4 automated annotation	Other automated cell type annotation software
Requires strong biology expertise	*No* biology expertise required	No biology expertise required
No coding expertise required	*No* coding expertise required	Requires medium coding expertise
Reference scRNA-seq not needed	Reference scRNA-seq *not* needed	May need to collect reference scRNA-seq
Significant effort required	*Minimum* effort required	Moderate effort required
Fits into existing pipeline	Fits into *existing* pipeline	Standalone pipeline
Tissue type not restricted	Tissue type *not* restricted	Tissue types restricted to reference data
Allows adjusting granularity	Allows *adjusting* granularity	Single granularity level

b

Identify cell types of human prostate cells using the following markers. Identify one cell type for each row. Only provide the cell type name.

CR2, CD24, FAS, CXCR3, CD1c
KLK3, KRT8, KLK2, MSMB, ACPP, KLK1, KLK4
MMRN1, FLT4, RELN, CCL21, PROXl, LYVE1
TPSAB1, FCER1A, TPSB2, KIT, CD69, HDC
ACTA2, MY01B, ACTA2, ANPEP, DES, MCAM, PDGFRB, CSPG4

1. Dendritic cells
2. Luminal epithelial cells
3. Lymphatic endothelial cells
4. Mast cells
5. Pericytes

Be more specific

1. Myeloid dendritic cells (mDCs)
2. Prostate secretory luminal epithelial cells
3. Prostate lymphatic endothelial cells
4 . Prostate tissue-resident mast cells
5. Prostate vascular pericytes

c

Identify cell types of human prostate cells using the following markers. Identify one cell type for each row. Only provide the cell type name. Some could be a mixture of multiple cell types. Some could be unknown cell types.

KLK3, KRT8, KLK2, MSMB, ACPP, KLK1, KLK4
MMRN1, FLT4, RELN, CCL21, PROX1, LYVE1
CD69, IL7R,, CD3D, CD3E, CD3G,ACTA2, MYO1B, ACTA2, ANPEP, PDGFRB, CSPG4
DDX49,LOC105371196,MTND1P30,LOC105373682,TAGLN2,ZNF836,ZNF677,COILP1

1. Prostate epithelial cells
2. Lymphatic endothelial cells
3. T cell and smooth muscle cell mixture
4. Unknown cell type

Section 6: Find DEG based on clustering results


```
# 6.1 find DEG between two selected clusters
DefaultAssay(pbmc) <- "RNA"
cluster1.markers <- FindMarkers(pbmc, ident.1 = 1, min.pct = 0.25)
cluster5.markers <-
 FindMarkers(pbmc, ident.1 = 5, ident.2 = c(0, 3), min.pct = 0.25)

# 6.2 look at the DEG result
cluster1.markers[1:5,]
```

	P_val	Avg_logFC	Pct.1	Pct.2	P_val_adj
RPS12	7.015738e-111	0.4734381	1.000	0.992	2.296812e-106
RPL32	1.862033e-109	0.4149342	0.998	0.995	6.095923e-105
RPS27	5.612393e-107	0.4651555	0.998	0.992	1.837385e-102
RPS6	6.336980e-106	0.4362345	1.000	0.995	2.074601e-101
CYBA	3.091515e-98	-1.0682534	0.625	0.906	1.012100e-93

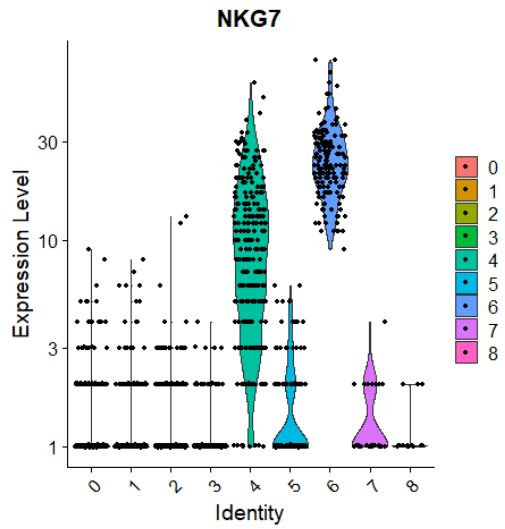
Section 6: Find DEG based on clustering results

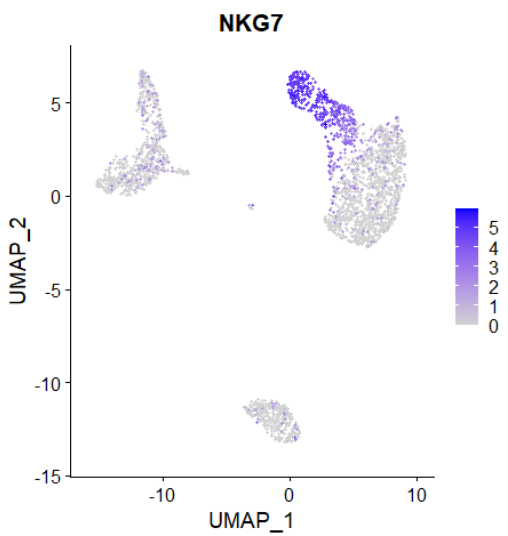

```
# 6.3 find DEG for each clustering in one code
pbmc.markers <-
 FindAllMarkers(pbmc, only.pos = TRUE, min.pct = 0.25, logfc.thresh
old = 0.25)
# 6.4 look at the DEG result
pbmc.markers[1:5,]
```

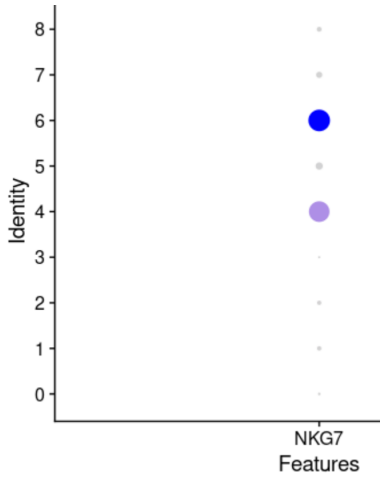
	P_val	Avg_logFC	Pct.1	Pct.2	P_val_adj	cluster	P_val
LTB	6.861872e-105	0.8834625	0.969	0.623	2.246440e-100	0	LTB
LDHB	3.652889e-103	0.7429271	0.954	0.590	1.195883e-98	0	LDHB
IL32	9.407421e-100	0.7935960	0.924	0.439	3.079802e-95	0	IL32
RPS121	7.015738e-111	0.4734381	1.000	0.992	2.296812e-106	1	RPS12
RPL32	1.862033e-109	0.4149342	0.998	0.995	6.095923e-105	1	RPL32

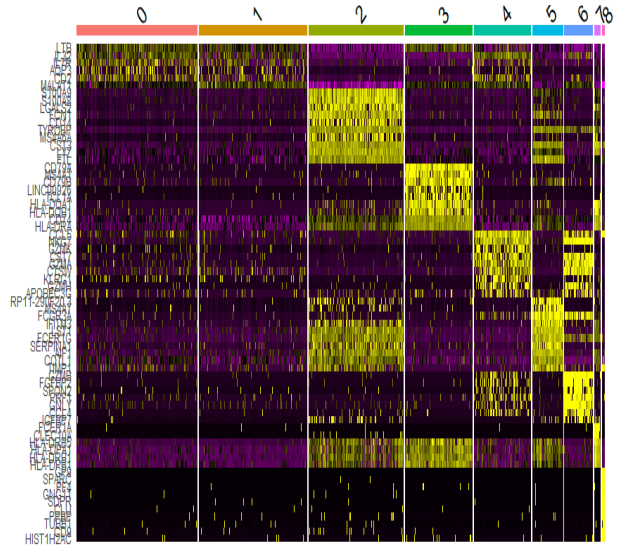
Section 7: Gene expression visualization


```
# 7.1 violin plot visualizes interested gene distribution in each clusters.
VlnPlot(pbmc, features = c("NKG7"), slot = "counts", log = TRUE)
# 7.2 color cells with expression value on UMAP
FeaturePlot(pbmc, features = c("NKG7"))
# 7.3 Gene expression across clusters on dot plot
DotPlot(pbmc, features = c("NKG7"))
# 7.4 visualize DEG gene in heatmap
top10 <-  pbmc.markers %>% group_by(cluster) %>% top_n(n = 10, wt = avg_logFC)
DoHeatmap(SeuratObject, features = top10$gene) + NoLegend()
```



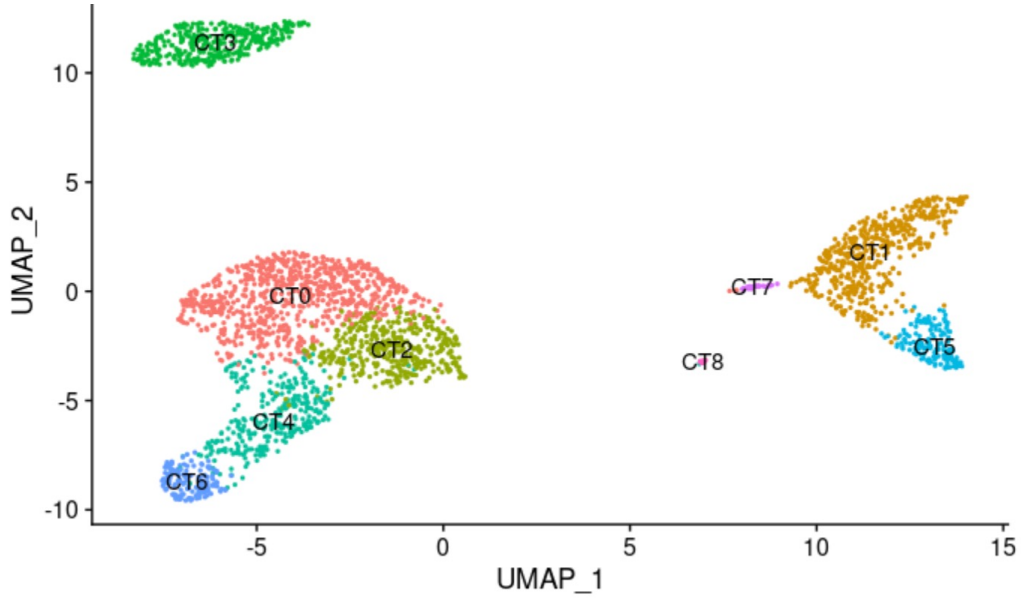






Section 8: Manual cell type annotation


```
new.cluster.ids <- c(”CT1", "CT2", "CT3", "CT4", "CT5", "CT6", "CT7”, "CT8")

names(new.cluster.ids) <- levels(SeuratObject)

SeuratObject <- RenameIdents(SeuratObject, new.cluster.ids)

DimPlot(SeuratObject, reduction = "umap", label = TRUE, pt.size = 0.5) +
NoLegend()
```



Multiple scRNA-seq data integration and query mapping

https://satijalab.org/seurat/articles/integration_introduction.html
https://satijalab.org/seurat/articles/integration_mapping.html

```
# Install new packages
#devtools::install_github('satijalab/seurat-data')
#library(SeuratData)

#InstallData("panc8")

# Load Seurat example data
data("panc8")
pancreas.list <- SplitObject(panc8, split.by = "tech")
pancreas.list <- pancreas.list[c("celseq", "celseq2", "fluidigmc1", "smartseq2")]

# standard preprocessing steps
for (i in 1:length(pancreas.list)) {
        pancreas.list[[i]] <- NormalizeData(pancreas.list[[i]])
        pancreas.list[[i]] <- FindVariableFeatures(pancreas.list[[i]],
selection.method =                                                    "vst",
nfeatures = 2000)
}
```

Here is what we do normally


```
# normally, we use Read10X or read.table or read.csv
dat.1 <- Read10X(data.dir = "")
dat.2 <- Read10X(data.dir = "")

# create Seurat objects
dat.1.obj <- CreateSeuratObject(counts = dat.1, project = “celseq", min.cells = 3,
min.features = 200)
dat.2.obj <- CreateSeuratObject(counts = dat.2, project = “celseq2", min.cells = 3,
min.features = 200)

#Give a group name ("treatment") and sample labels (stil and ctrl) to both data
dat.1.obj$tech="celseq"
dat.2.obj$tech="celseq2"

#create a list of Seurat objects and perform preprocessing
data.list <- lapply(X = list(dat.1.obj, dat.2.obj), FUN = function(x) {
  # making a mitochondria percentage column (mt/MT/Mt based on gtf)
  x[["percent.mt"]] <- PercentageFeatureSet(x, pattern = "^mt-")
  # Filtering, normalization and feature detection
  x <- subset(x, subset = nFeature_RNA > 200 & nFeature_RNA < 4000 & percent.mt < 10)
  x <- NormalizeData(x)
  x <- FindVariableFeatures(x, selection.method = "vst", nfeatures = 2000)
})
```

Type 1: Data integration and cell clustering


```
# Back to the example
# create a list
reference.list <- pancreas.list[c("celseq", "celseq2", "smartseq2")]

# select features that are repeatedly variable across datasets for integration
# NOTE: this is optional and not recommended for more data integration
features <- SelectIntegrationFeatures(object.list = reference.list)

# find integration anchors …
pancreas.anchors <- FindIntegrationAnchors(object.list = reference.list, dims = 1:30)

# … and integrate
data.combined <- IntegrateData(anchorset = pancreas.anchors, anchor.features = 2000)

# switch to integration assay
DefaultAssay(pancreas.integrated) <- "integrated"
```

Dimension reduction and cell clustering


```
# Standard workflow for visualization and clustering
pancreas.integrated <- ScaleData(pancreas.integrated, verbose = FALSE)
pancreas.integrated <- RunPCA(pancreas.integrated, npcs = 30, verbose = FALSE)
pancreas.integrated <- RunUMAP(pancreas.integrated, reduction = "pca", dims = 1:30,
verbose = FALSE)

# For this example, we do not perform cell clustering as the cell type labels have
predicted already and provided. Yet, in your case, please do that.
data.combined <- FindNeighbors(data.combined, reduction = "pca", dims = 1:15)
data.combined <- FindClusters(data.combined, resolution = 0.5)
```

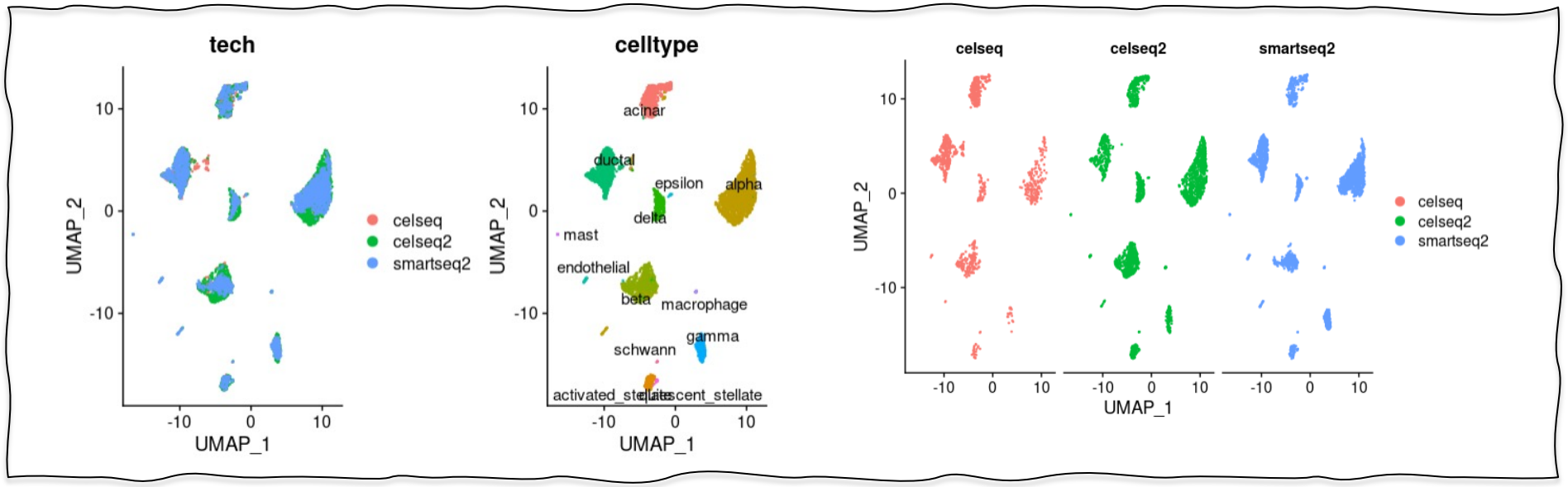
```
pancreas.integrated  Large Seurat (878.7 Mb)
..@ assays :List of 2
.. ..$ RNA :Formal class 'Assay' [package "Seura
.. .. .. ..@ counts :Formal class 'dgCMatrix' [p
.. .. .. .. .. ..@ i : int [1:30410029] 19 22 28
.. .. .. .. .. ..@ p : int [1:5684] 0 1986 6195
.. .. .. .. .. ..@ Dim : int [1:2] 34363 5683
.. .. .. .. .. ..@ Dimnames:List of 2
.. .. .. .. .. .. ..$ : chr [1:34363] "A1BG-AS1"
.. .. .. .. .. .. ..$ : chr [1:5683] "D101_5" "D
.. .. .. .. .. ..@ x : num [1:30410029] 1 1 1 2.
.. .. .. .. .. ..@ factors : list()
.. .. .. ..@ data :Formal class 'dgCMatrix' [pac
```

```
..@ meta.data :'data.frame': 5683 obs. of 8 var
.. ..$ orig.ident : chr [1:5683] "D101" "D101"
.. ..$ nCount_RNA : num [1:5683] 4616 29002 670
.. ..$ nFeature_RNA : int [1:5683] 1986 4209 24
.. ..$ tech : chr [1:5683] "celseq" "celseq" "c
.. ..$ replicate : chr [1:5683] "celseq" "celse
.. ..$ assigned_cluster: chr [1:5683] NA NA NA
.. ..$ celltype : chr [1:5683] "gamma" "acinar"
.. ..$ dataset : chr [1:5683] "celseq" "celseq"
..@ active.assay: chr "integrated"
..@ active.ident: Factor w/ 3 levels "celseq","
```

```
.. ..$ umap:Formal class 'DimReduc' [package "SeuratObject"] w
.. .. .. ..@ cell.embeddings : num [1:5683, 1:2] 3.538 -0.728
.. .. .. .. ..- attr(*, "scaled:center")= num [1:2] 1.156 0.27
.. .. .. .. ..- attr(*, "dimnames")=List of 2
.. .. .. .. .. ..$ : chr [1:5683] "D101_5" "D101_7" "D101_10"
.. .. .. .. .. ..$ : chr [1:2] "UMAP_1" "UMAP_2"
```

UMAP visualization

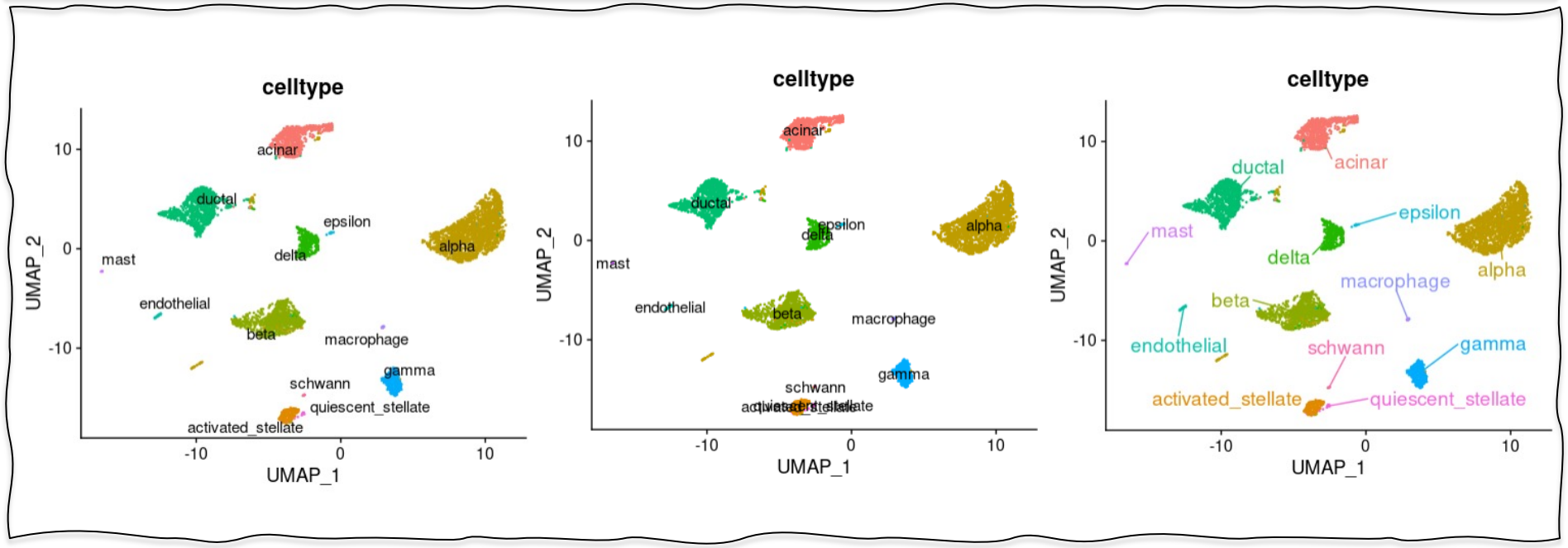

```
# Number of cells in each condition
table(pancreas.integrated@meta.data$tech)
table(pancreas.integrated@meta.data$tech, pancreas.integrated@meta.data$celltype)

# Visualization
p1 <- DimPlot(pancreas.integrated, reduction = "umap", group.by = "tech")
p2 <- DimPlot(pancreas.integrated, reduction = "umap", group.by = "celltype", label =
TRUE, repel = TRUE) + NoLegend()
p3 <- DimPlot(pancreas.integrated, reduction = "umap", split.by = " tech")
p1 + p2 + p3
```




```
celseq   celseq2 smartseq2
  1004      2285      2394
```

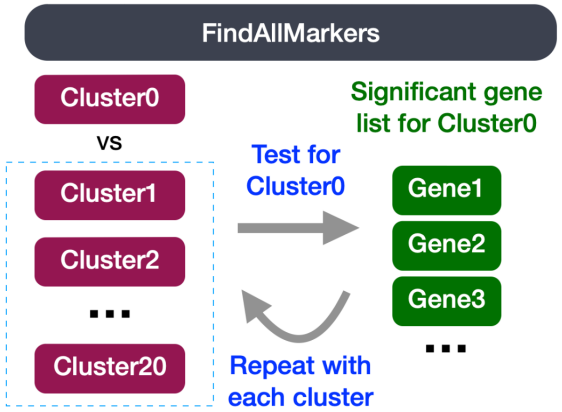
More about UMAP

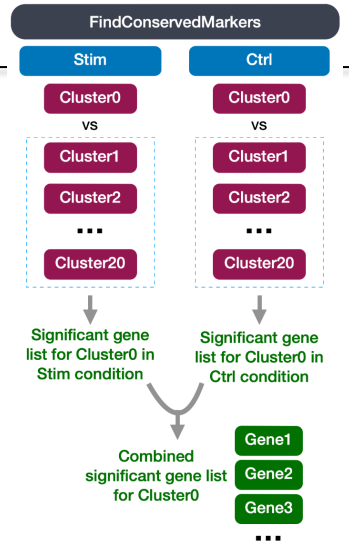

```
p2 <- DimPlot(pancreas.integrated, reduction = "umap", group.by = "celltype", label =
TRUE, repel = TRUE) + NoLegend()

p4 <- DimPlot(pancreas.integrated, reduction = "umap", group.by = "celltype", label =
TRUE, repel = FALSE)

p5 <- DimPlot(pancreas.integrated, reduction = "umap", group.by = "celltype")
LabelClusters(p3, id = "celltype", color = unique(ggplot_build(p3)$data[[1]]$colour),
size = 5, repel = T, box.padding = 1.25)
```



DEG analysis

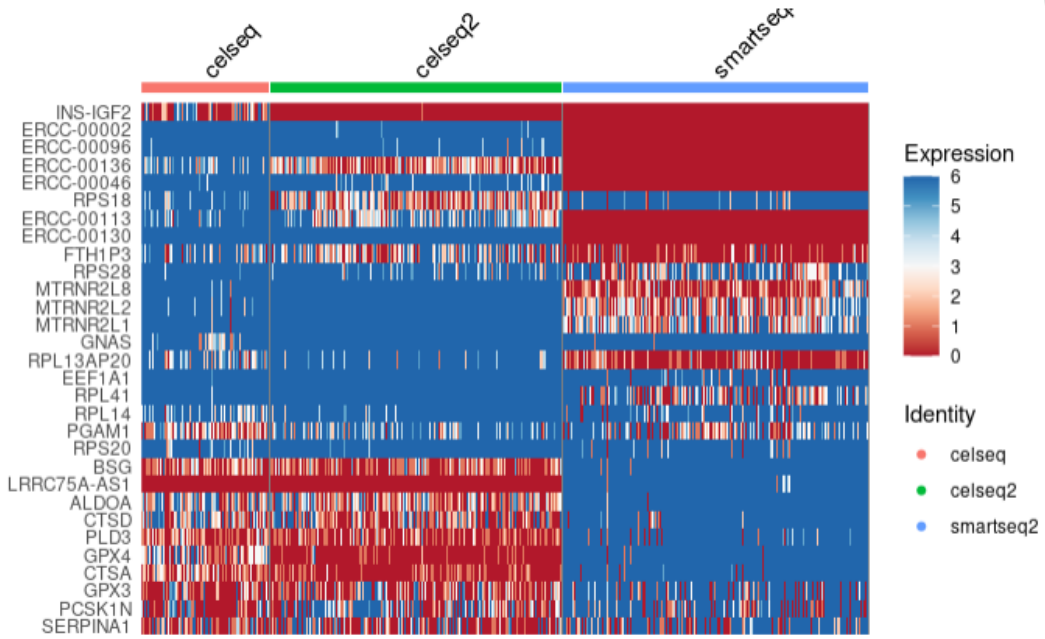

```
# Remember to switch to raw data for DEG
DefaultAssay(pancreas.integrated) <- "RNA"

# Find DEGs in celseq
celseq.markers <- FindMarkers(pancreas.integrated, ident.1 = "celseq",group.by = "tech",
        logfc.threshold = 0.25, only.pos = TRUE)

# Find DEGs in alpha
Idents(pancreas.integrated) <- "celltype"
alpha.markers <- FindMarkers(pancreas.integrated, ident.1 = "acinar", logfc.threshold =
        0.25, only.pos = TRUE)

# Find conserved DEGs among techs
all.markers <- FindConservedMarkers(pancreas.integrated, ident.1 = c("acinar"),
        grouping.var = "tech")
```





DEG analysis


```
# Find DEGs for all techs
Idents(pancreas.integrated) <- "tech"
all.markers <- FindAllMarkers(pancreas.integrated, logfc.threshold = 0.25, only.pos =
                                                                    TRUE)
top_10_marker <- all.markers %>% group_by(cluster) %>% top_n(n =10, avg_log2FC)
head(top_10_marker)

# Draw heatmap
DoHeatmap(pancreas.integrated,features = top_10_marker$gene,slot="counts", size = 4) +
scale_fill_gradientn(colors = RColorBrewer::brewer.pal(n = 9, name = "RdBu"))
```



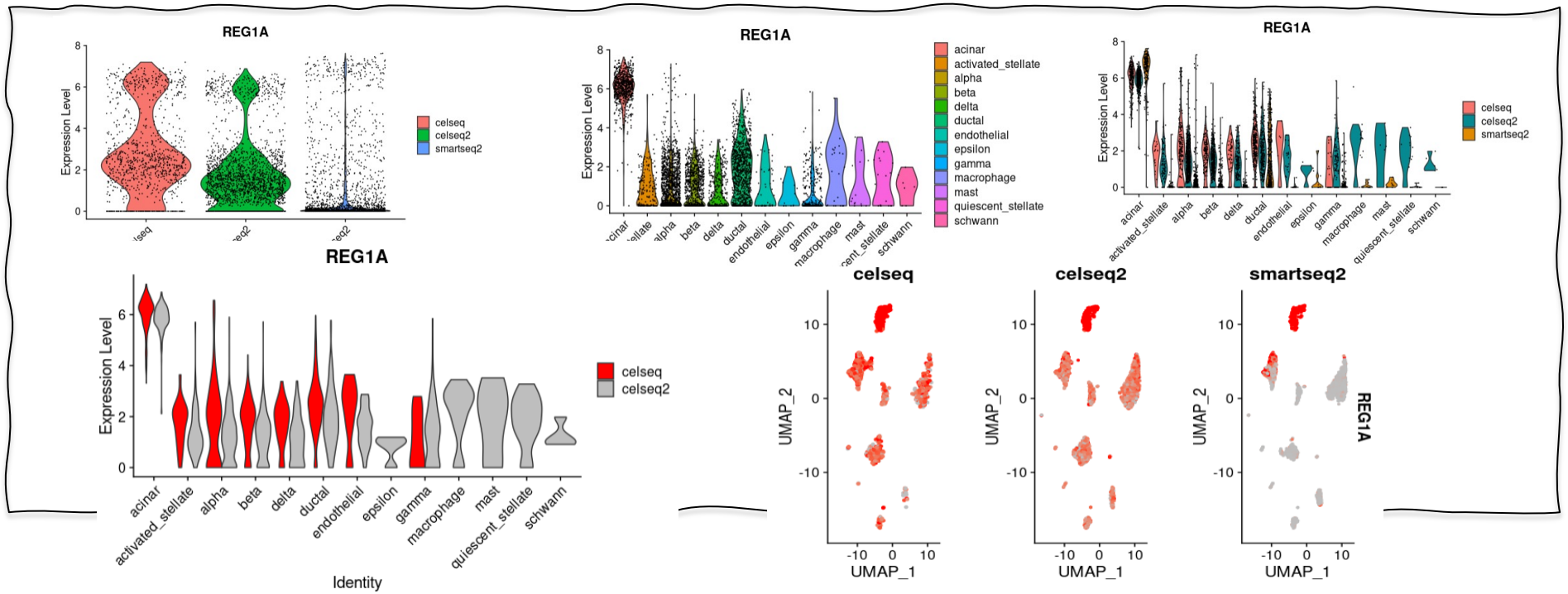
Note: the output table of FindAllMarkers have gene names edited into:

Gene.1_1 … cluster 1
Gene.1_2 … cluster 8
…

You want to use the last column for actual gene names when you perform pathway analysis.

Marker gene visualizations


```
Idents(pancreas.integrated) <- "tech"
VlnPlot(pancreas.integrated, features =c("REG1A"))
VlnPlot(pancreas.integrated, features =c("REG1A"), group.by = "celltype")
VlnPlot(pancreas.integrated, features = c("REG1A"),split.by = "tech")

pancreas.integrated.sub <- subset(pancreas.integrated, idents=c("celseq","celseq2"))
VlnPlot(pancreas.integrated.sub, features = c("REG1A"),group.by = "celltype",
                    split.by = "tech", cols=c("red","grey","blue"), pt.size = 0)
FeaturePlot(pancreas.integrated, features = c("REG1A"), split.by = "tech",
                    max.cutoff = 3, cols = c("grey", "red"))
```



Type 2: Reference mapping


```
pancreas.query <- pancreas.list[["fluidigmc1"]]

pancreas.anchors <- FindTransferAnchors(reference = pancreas.integrated,
                                        query = pancreas.query, dims = 1:30,
reference.reduction = "pca")

predictions <- TransferData(anchorset = pancreas.anchors,
                                                    refdata =
pancreas.integrated$celltype, dims = 1:30)

pancreas.query <- AddMetaData(pancreas.query, metadata = predictions)

# check if the mapped (prediction) align well wit
pancreas.query$prediction.match <- pancreas.query
pancreas.query$celltype
```

	predicted.id <chr>	prediction.score.gamma <dbl>
10th_C10_S104	alpha	0.00000000
10th_C11_S96	beta	0.00000000
10th_C13_S61	beta	0.00000000
10th_C14_S53	beta	0.00000000
10th_C16_S105	alpha	0.00000000
10th_C17_S97	beta	0.00000000

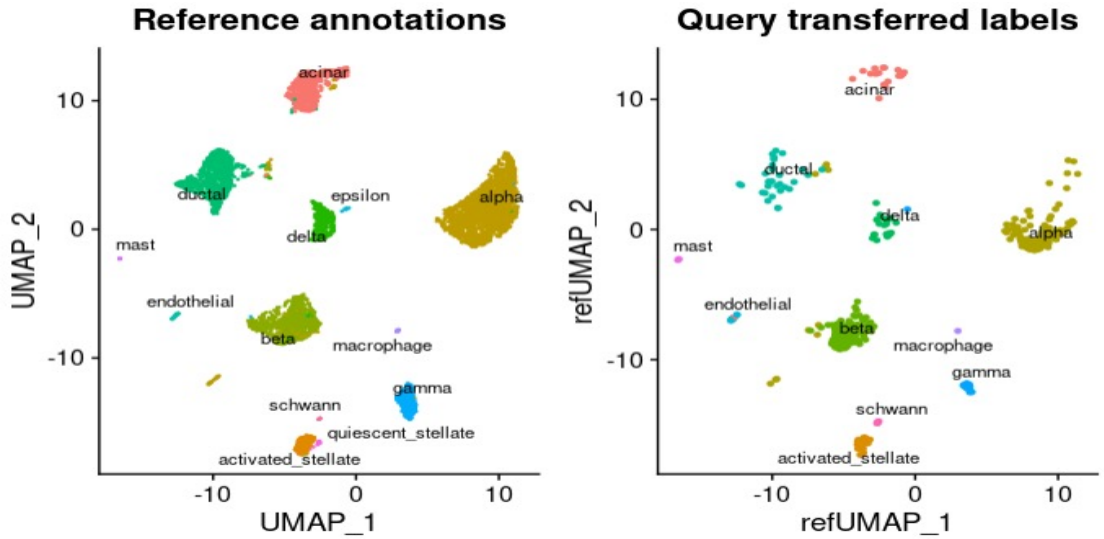
```
table(pancreas.query$prediction.match)
```

```
10th_C10_S104  10th_C11_S96  10th_C13_S61  10th_C14_S53 10th_C16_S105  10th_C17_S97
        FALSE          TRUE          TRUE          TRUE          TRUE          TRUE
 10th_C19_S62   10th_C1_S59  10th_C20_S54  10th_C23_S98  10th_C24_S90  10th_C28_S91
         TRUE          TRUE          TRUE          TRUE          TRUE          TRUE
```

```
FALSE  TRUE
   21   617
```

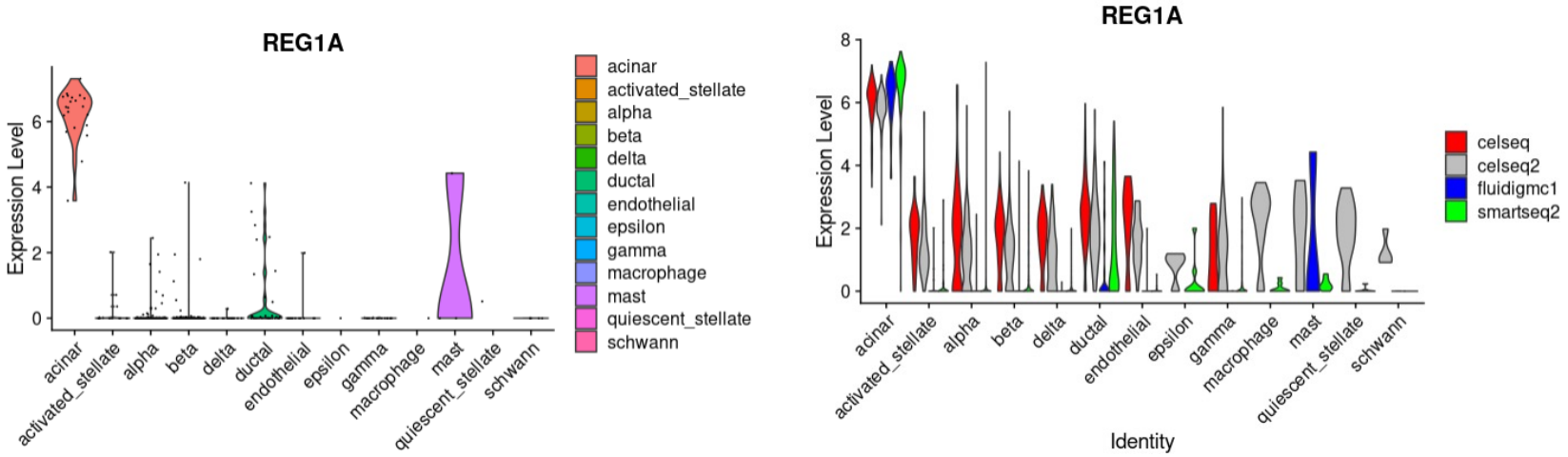
UMAP visualization


```
DefaultAssay(pancreas.integrated) <- "RNA"
pancreas.integrated <- RunUMAP(pancreas.integrated, dims = 1:30, reduction = "pca",
        return.model = TRUE)
pancreas.query <- MapQuery(anchorset = pancreas.anchors, reference = pancreas.integrated,
        query = pancreas.query, refdata = list(celltype = "celltype"),
reference.reduction =       "pca", reduction.model = "umap")
p1 <- DimPlot(pancreas.integrated, reduction = "umap", group.by = "celltype", label =
        TRUE, label.size = 3,    repel = TRUE) + NoLegend() + ggtitle("Reference
        annotations")
p2 <- DimPlot(pancreas.query, reduction = "ref.umap", group.by = "predicted.celltype",
        label = TRUE,    label.size = 3, repel = TRUE) + NoLegend() + ggtitle("Query
        transferred labels")
p1 + p2
```



More marker gene visualizations


```
VlnPlot(pancreas.query, features =c("REG1A"), group.by = "celltype")

pancreas.merge <- merge(pancreas.integrated,pancreas.query)
VlnPlot(pancreas.merge, features =c("REG1A"), group.by = "celltype", split.by =
        "tech",cols=c("red","grey","blue","green"), pt.size = 0)

markers.to.plot <- c("CD3D", "CREM", "HSPH1", "SELL", "GIMAP5", "CACYBP", "GNLY",
        "NKG7", "CCL5", "CD8A", "MS4A1", "CD79A")
DotPlot(pancreas.merge, features = markers.to.plot, cols = c("blue", "red“, ”yellow”,
        ”green”), dot.scale = 8, split.by = “tech") +   RotatedAxis()
```



Sankey plot


```
# create framework of two labels
sankey.dat <-
         data.frame(source=pancreas.query$predicted.id,target=pancreas.query$celltype,
         value=rep(1, length(pancreas.query$celltype)))
sankey.dat$new <- paste(sankey.dat$source, sankey.dat$target)

# create a connecting data frame of label pair frequencies
sankey.link <- aggregate(value~new, sankey.dat,sum)
sankey.link <- separate(sankey.link, col = new, into = c("source", "target"), sep = " ")
sankey.link$target <- paste(sankey.link$target, " ", sep="")

# create a node data frame of all unique labels
sankey.nodes <- data.frame(name=c(as.character(sankey.link$source),
         as.character(sankey.link$target)) %>% unique())
```

Sankey.dat

	source	target	value	new
10th_C10_S104	alpha	ductal	1	alpha ductal
10th_C11_S96	beta	beta	1	beta beta
10th_C13_S61	beta	beta	1	beta beta
10th_C14_S53	beta	beta	1	beta beta

Sankey.link

	source	target	value
1	acinar	acinar	21
2	acinar	endothelial	1
3	activated_stellate	activated_stellate	16
4	activated_stellate	quiescent_stellate	1
5	alpha	alpha	240

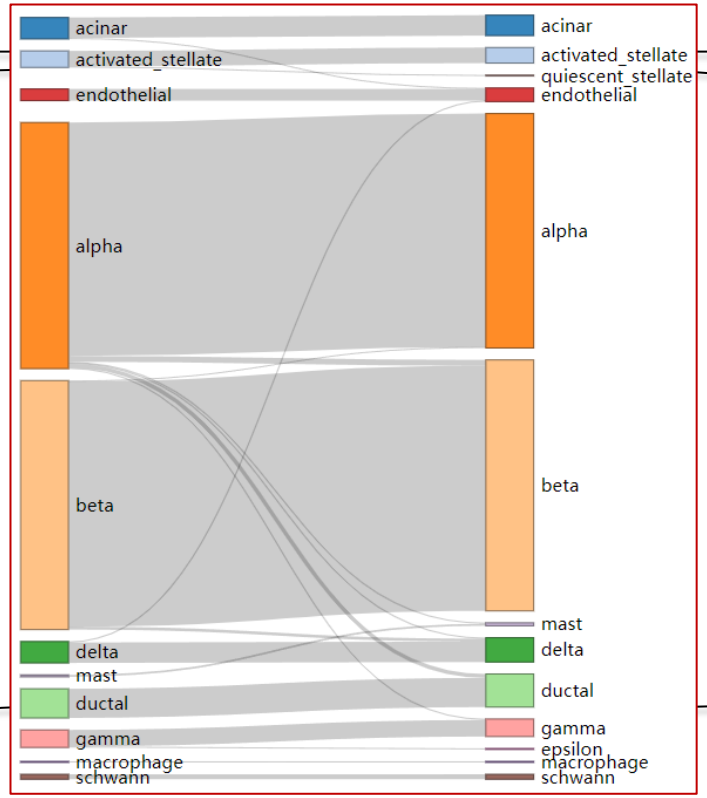
Sankey.node

	name
1	acinar
2	activated_stellate
3	alpha
4	beta

Sankey plot


```
# transfer target and source names to node numbers
sankey.link$IDsource <- match(sankey.link$source, sankey.nodes$name)-1
sankey.link$IDtarget <- match(sankey.link$target, sankey.nodes$name)-1

p <- sankeyNetwork(Links = sankey.link, Nodes = sankey.nodes,
                                    Source = "IDsource", Target = "IDtarget",
                                    Value = "value", NodeID = "name",
                                    sinksRight=FALSE, fontSize=15,
nodeWidth=40,nodePadding=10)
p
```

	source	target	value	IDsource	IDtarget
1	acinar	acinar	21	0	11
2	acinar	endothelial	1	0	12
3	activated_stellate	activated_stellate	16	1	13
4	activated_stellate	quiescent_stellate	1	1	14
5	alpha	alpha	240	2	15
6	alpha	beta	6	2	16



Some notes for single-cell data integration

The data integration should be done when:

- You want to increase your sample (cell) pool
- You want to compare two scRNA-seq samples
- You want to jointly analyze scMulti-omics data (we will introduce more in the following classes)

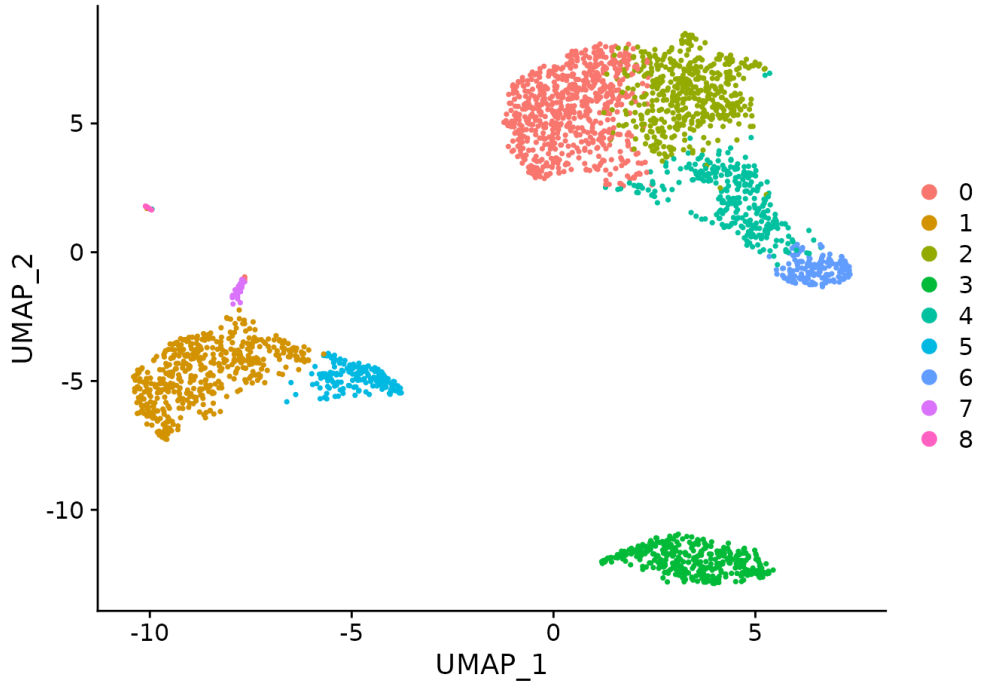
When use reference mapping:

- Be sure to choose a annotated reference data similar to your query data (you don't want to use a PBMC reference to annotate your tumor samples)
- Choose a reference data with cells more than your query data.
- Do not 100% trust your reference mapping annotations. Always check marker gene expressions.
- Other tools for automated annotation: singleR, scANVI
- Check here for more: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7873570/

Data visualization on single cells

Dimension reduction: UMAP/tSNE

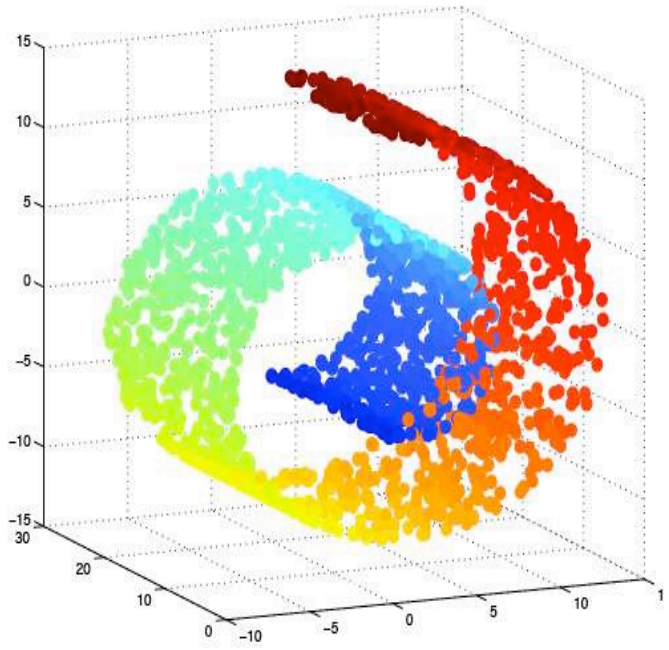
- UMAP: Nonlinear dimension reduction
- tSNE is also used instead of UMAP
- To learn the underlying manifold of the data to place similar cells together in low-dimensional space.
- Intuitively, we expect cells belong to the same cluster co-localize.



Dimension reduction: UMAP/tSNE

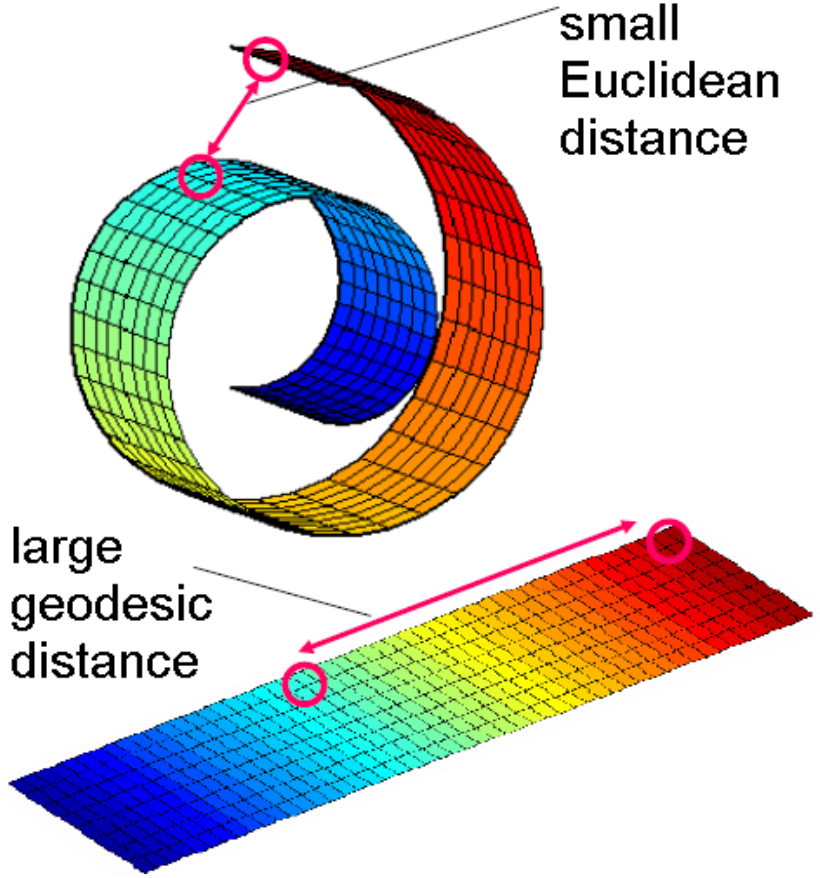
https://www.bettycrocker.com/recipes/chocolate-swiss-roll/eaaacebf-2d31-41ad-b547-d8f37f5afa44

Dimension reduction: UMAP/tSNE



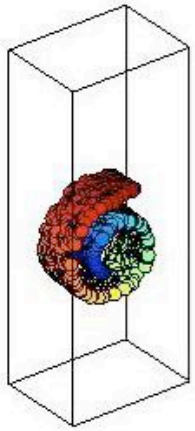
https://www.cs.utah.edu/~piyush/teaching/25-10-slides.pdf

Dimension reduction: UMAP/tSNE



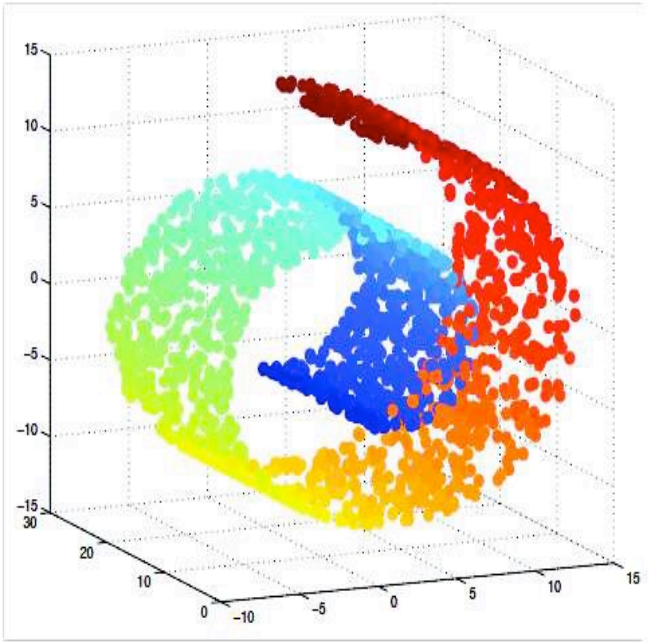
https://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/slides/ManifoldLearning.pdf

Dimension reduction: UMAP/tSNE



https://www.cs.cmu.edu/~bapoczos/Classes/ML10715_2015Fall/slides/ManifoldLearning.pdf

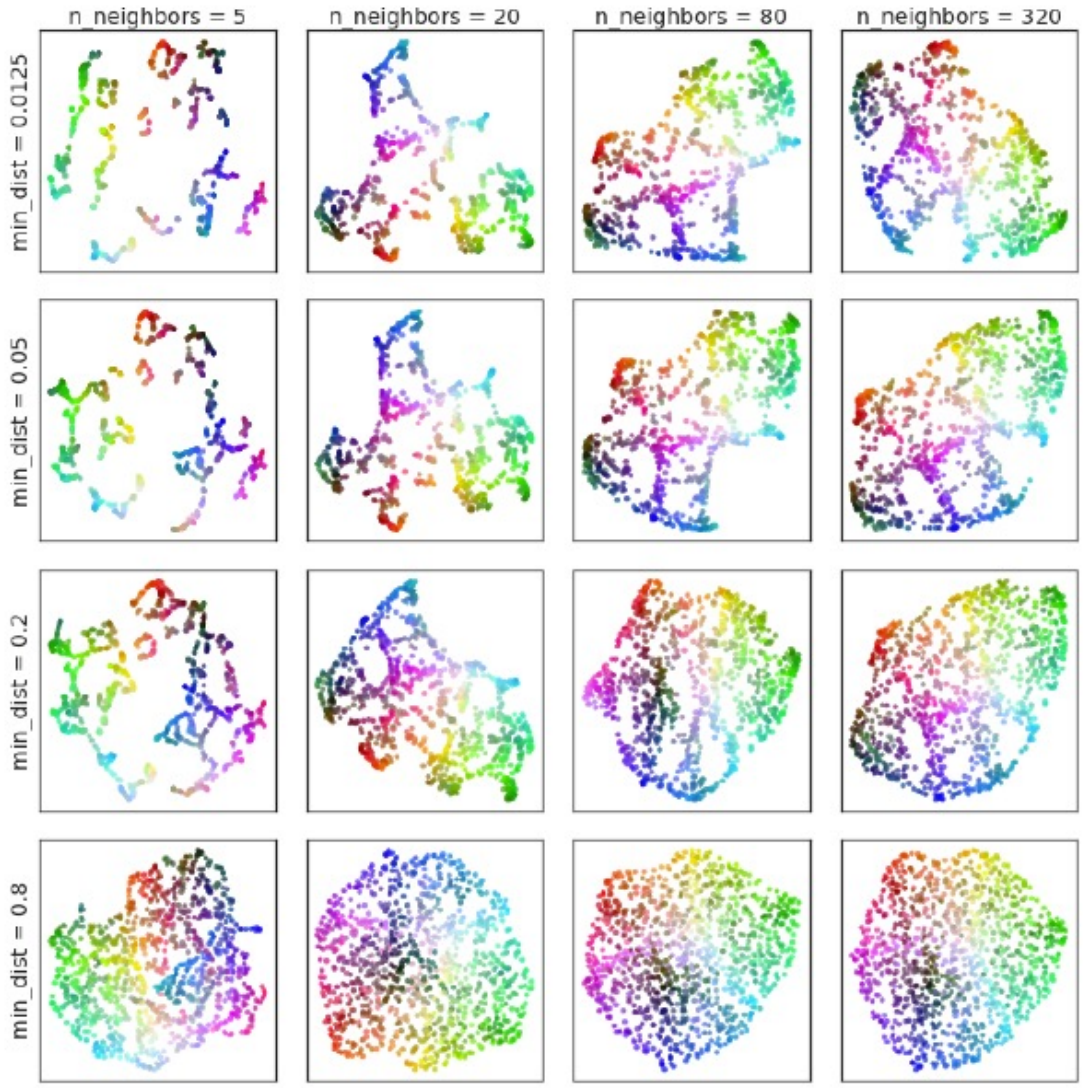
Dimension reduction: UMAP/tSNE



https://www.cs.utah.edu/~piyush/teaching/25-10-slides.pdf

Dimension reduction: UMAP/tSNE

- UMAP also has tuning parameters
- Intuitive explanation:
- https://pair-code.github.io/understanding-umap/



McInnes et al. (2018) arXiv:1802.03426v2 [stat.ML]