Single Cell Genomics

Bill Flynn, Computational Scientist, JAX

bill.flynn@jax.org

MEDS6498, Spring 2020

Goals for this lecture

1. Understand the technologies and analysis methods available in the single cell field to critically examine publications.
2. Understand the **challenges** these methods face. A good review: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-1926-6 (https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-1926-6)
3. For single cell RNA sequencing:
    - Understand the analysis decisions and common pitfalls involved with typical experiments
    - Know where to start and what tools are available if you are faced with analysis of this data.

Structure of the lecture

1. History lesson
2. Overview of Single cell RNA-sequencing
3. Current technologies
4. Analysis workshop
5. Other single cell technologies

Slides available at: https://wflynny.github.io/MEDS6498-2020/ (https://wflynny.github.io/MEDS6498-2020/)

What is a cell?

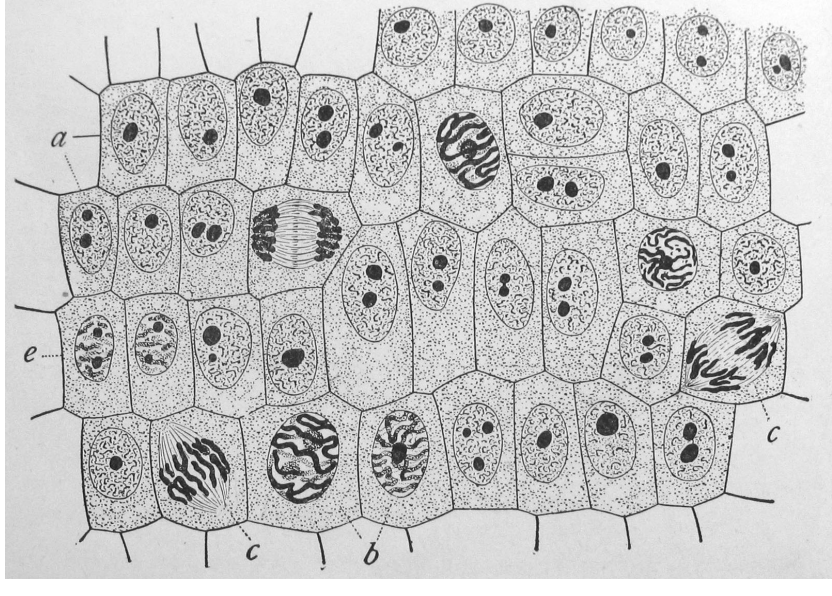
> ***That there is one universal principle of development for the elementary part of organisms, however different, and that this principle is the formation of cells.***

Microscopical researches into the accordance in the structure and growth of animals and plants
Theodor Schwann (1810--1882)

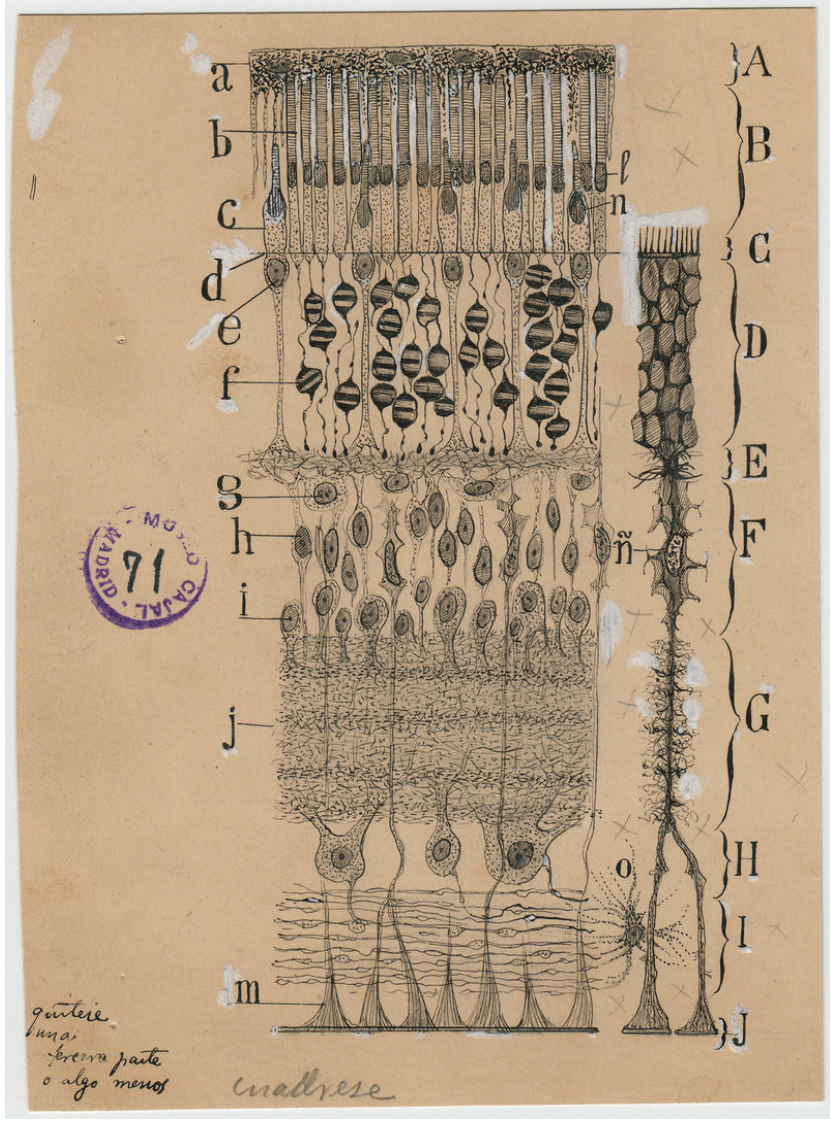
Understanding what makes a cell a cell

One of the major goals of Cell Biology has been to uncover and characterize the diverse set of cells that are life's building blocks.

For a long time, this was done manually with microscopes.



Early cataloging of retina cells, Ramon y Cajal



Advancements in multiple areas improved ability to describe cells

- Molecular biology --> improvements in isolation and dissociation
  - enzymes, laboratory techniques
  - discovery of DNA, RNA
- Microscopy --> better morphological descriptions and throughput

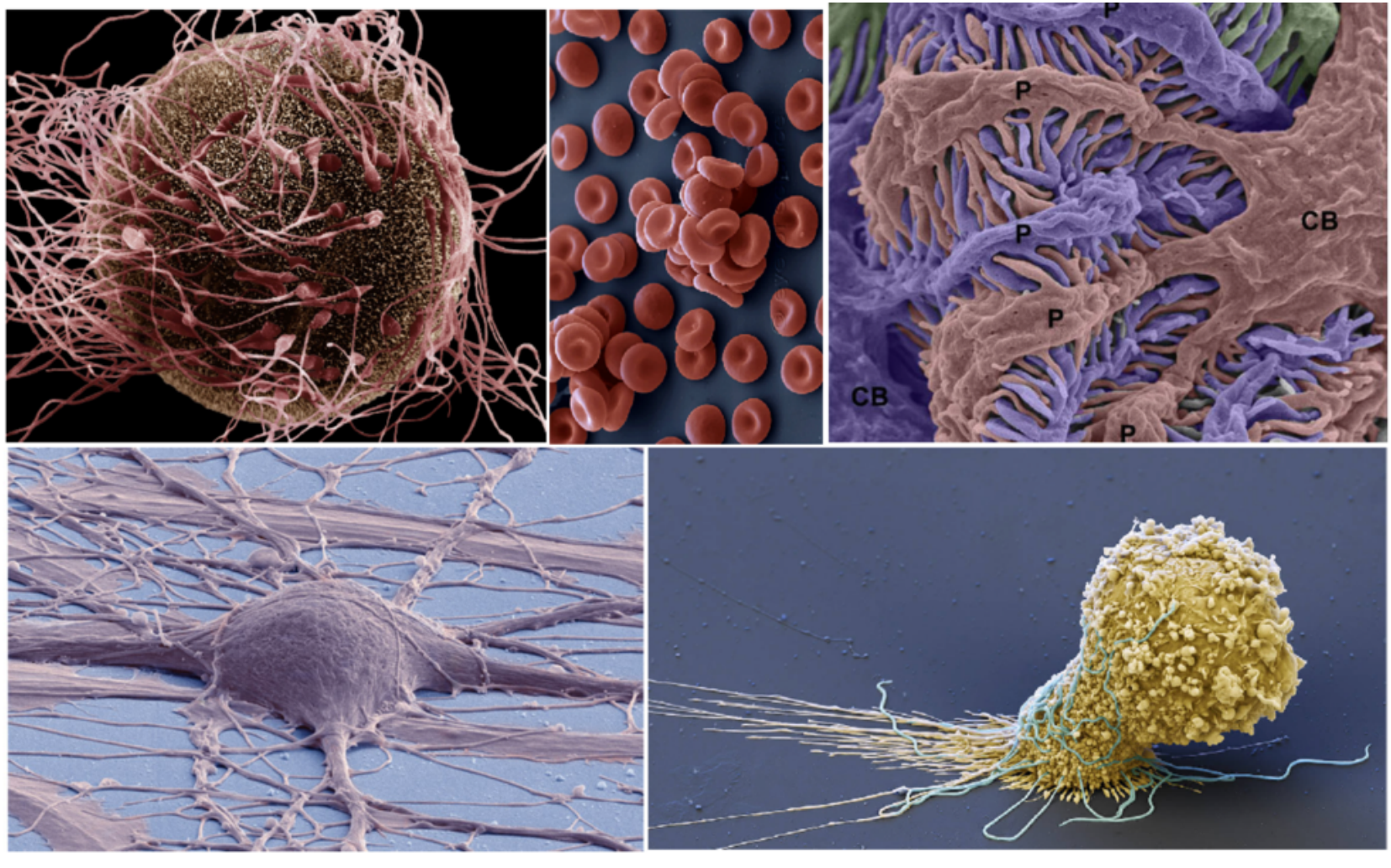
Advancements in multiple areas improved ability to describe cells

- Molecular biology --> improvements in isolation and dissociation
  - enzymes, laboratory techniques
  - discovery of DNA, RNA
- Microscopy --> better morphological descriptions and throughput

More recent advancements have led to more ways to describe cells:

- Sequencing allows us to characterize both interior and exterior of cells
- Microfluidic valves and droplets dramatically increase the throughput with which to capture and probe individual cells

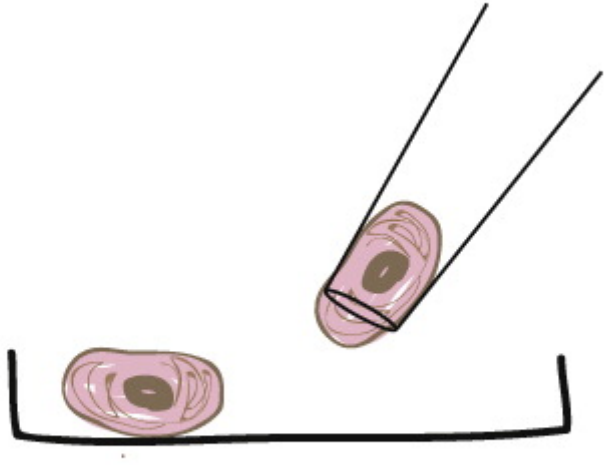
Many sizes, shapes, functions



This functional, morphological, and spatial heterogeneity implies heterogeneity within and between types of cells

How do we profile single cells?

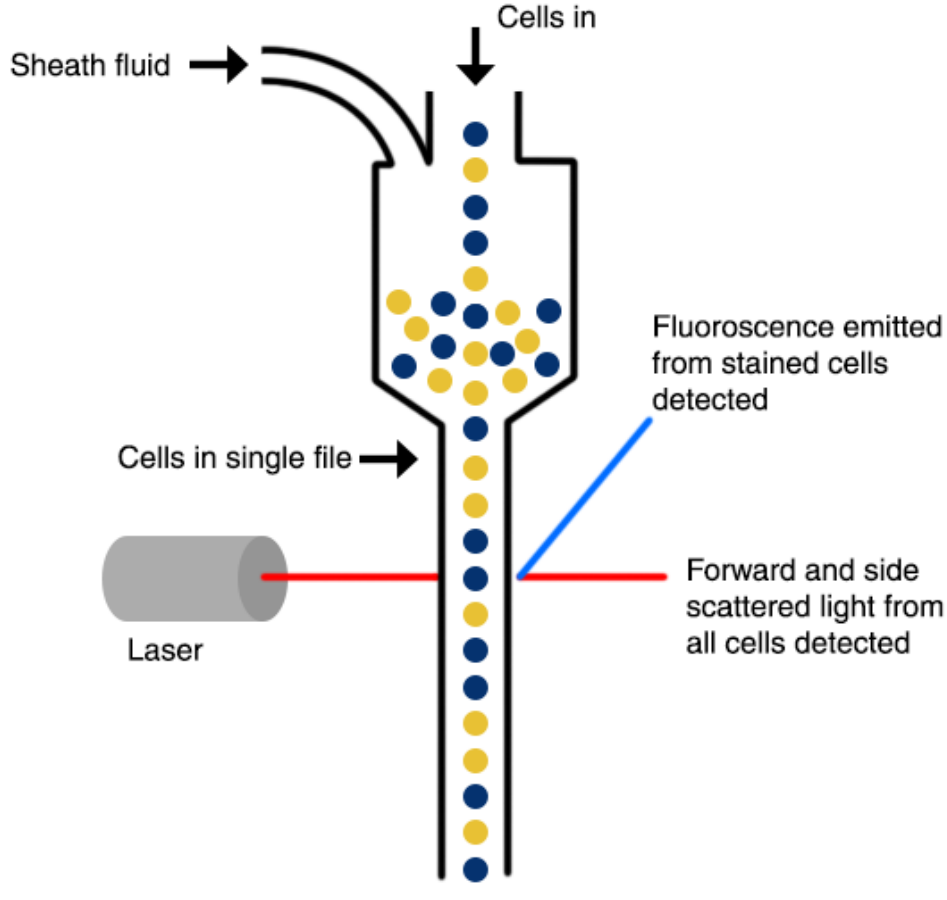
Started by micropipetting individual cells



Extremely labor-intensive, yielding on the order of a dozen cells.

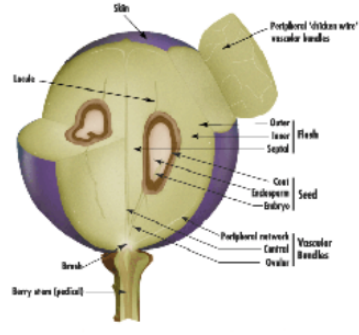
This was ***the only*** process up until about 50 years ago, and is still utilized today.

Then graduated to flow cytometry



Available since the 1970s. Currently 28 fluorescent probes or 40 mass spectrometry-based probes available.

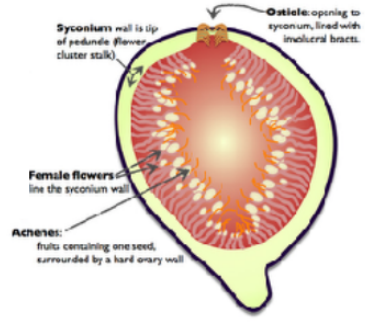
How to increase the measurable set of features?



a grape

measure more features

a blue fruit



a blueberry

In order to profile cells more deeply, we must explore the properties that define a cell from the ***inside***.

The transcriptome defines a cell

The transcriptome defines a cell

A Cell's Function is Determined By the Repertoire of Genes It Expresses

The transcriptome defines a cell

A Cell's Function is Determined By the Repertoire of Genes It Expresses

Characterization of a cell from the inside

- Each cell type expresses a specific fraction of genes, these define its function.
- ~15,000--150,000 mRNA molecules per cell
- Levels of gene-specific mRNA expression are characteristic of cell type (i.e. not just binary on/off)

The transcriptome defines a cell

A Cell's Function is Determined By the Repertoire of Genes It Expresses

Characterization of a cell from the inside

- Each cell type expresses a specific fraction of genes, these define its function.
- ~15,000--150,000 mRNA molecules per cell
- Levels of gene-specific mRNA expression are characteristic of cell type (i.e. not just binary on/off)

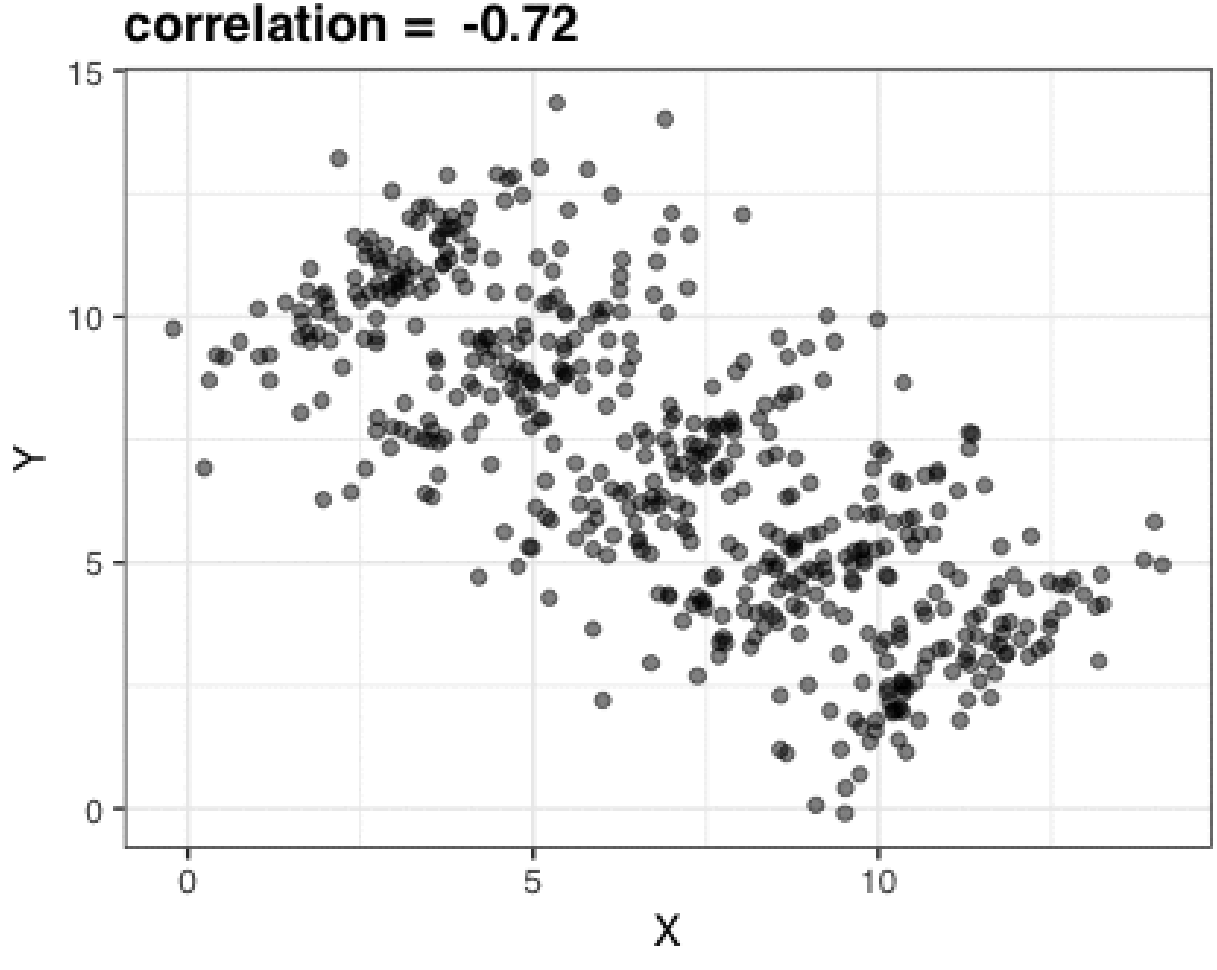
Single Cell Transcriptomics *is* Cell Biology

With the genome sequenced and annotated, we can define the unknown

- Limited set of genes (~30,000) to define all cell types in the human and mouse genome
- Unique combinations of these genes will define all cell types
- Single cell transcriptomics thus provides an unbiased approach to define all cell types
- Defining the transcriptomes of individual cell types will provide insight into this cell's function
- Identify rare or unknown cells

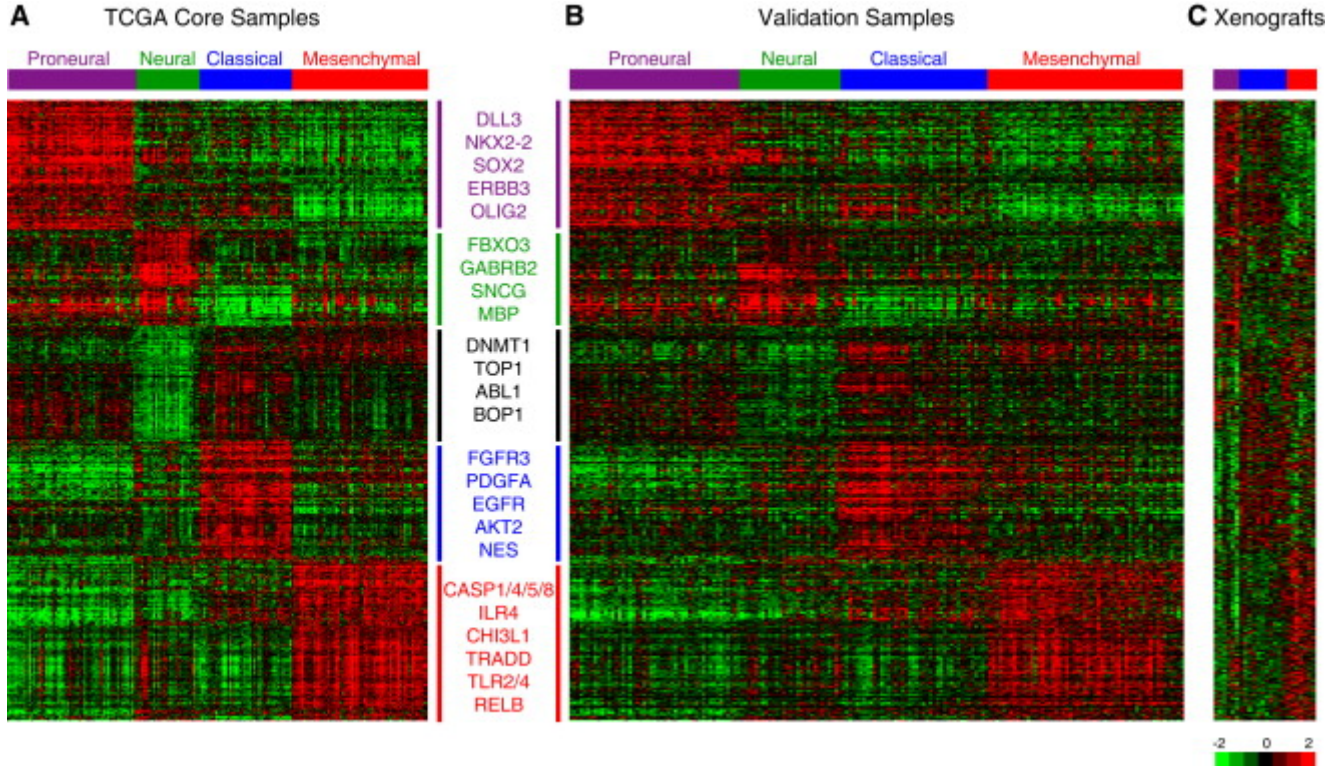
Can't we just use bulk RNA-seq to understand cellular makeup?

Signals about cellular makeup from bulk RNA-seq can be misleading



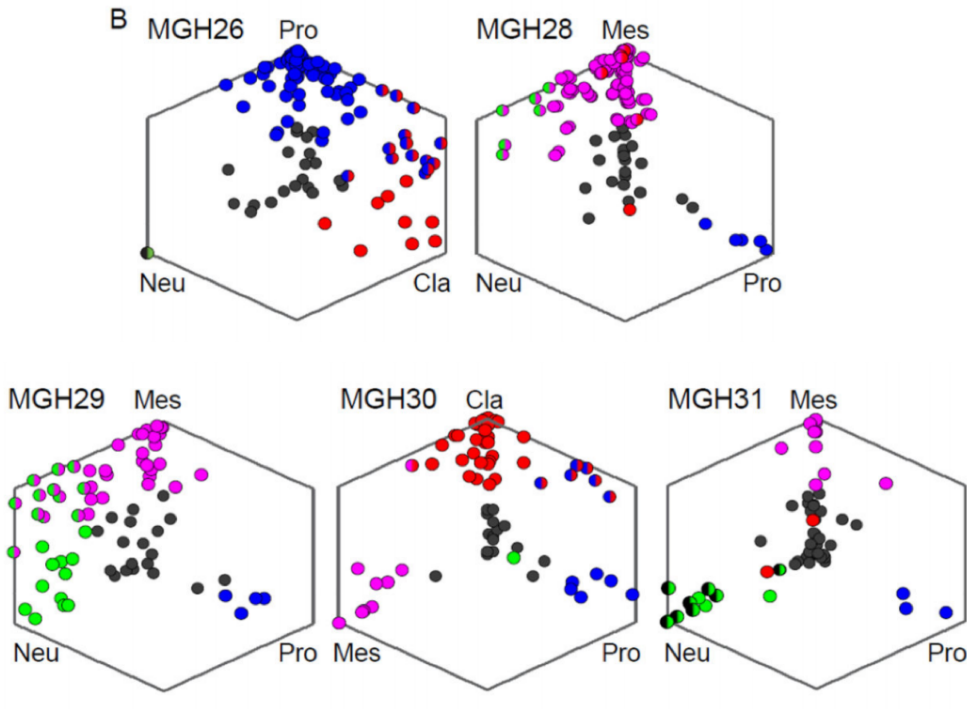
An example: molecular subtypes of glioblastoma

Bulk gene signatures define 4 molecular subtypes in GBM.



Verhaak et al Cancer Cell 2010
(https://linkinghub.elsevier.com/retrieve/pii/S1535610809004322)

An example: molecular subtypes of glioblastoma



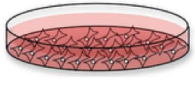
When applied to the single cell transcriptomes of 5 patients, tumors are made up of a mix of cells from each subtype.
Some cells even express multiple subtype signatures.

Patel et al. Science 2014 (http://science.sciencemag.org/content/344/6190/1396.full)

How is high throughput sequencing of single cells done?

Overview scRNA-seq workflow

Sample/tissue Dissociation

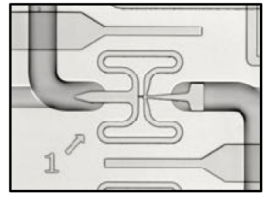


OR

Enrichment (or not)

Cell capture, lysis, RT, DNA amplification

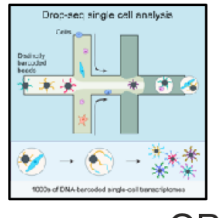
1. C1-based (**96 to 1,600** cells)



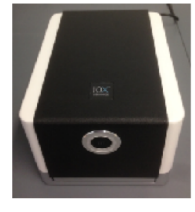
OR

2. Droplet-based (**1,000s** of cells)

Drop-seq

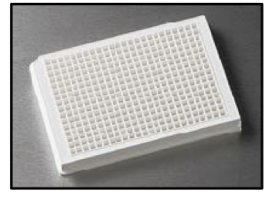


10XGenomics



OR

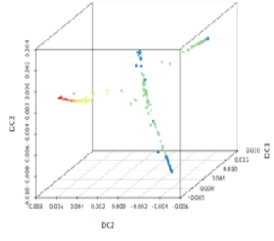
3. Plate or Tube-based (**1 to 1,000s** of cells)



OR

Sequence

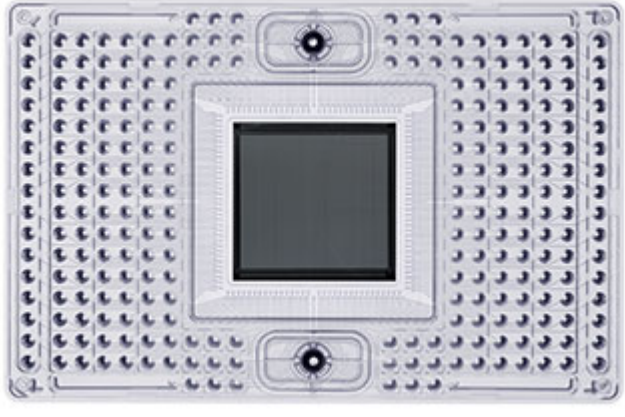
informatics



The first step is isolating cells from one another.

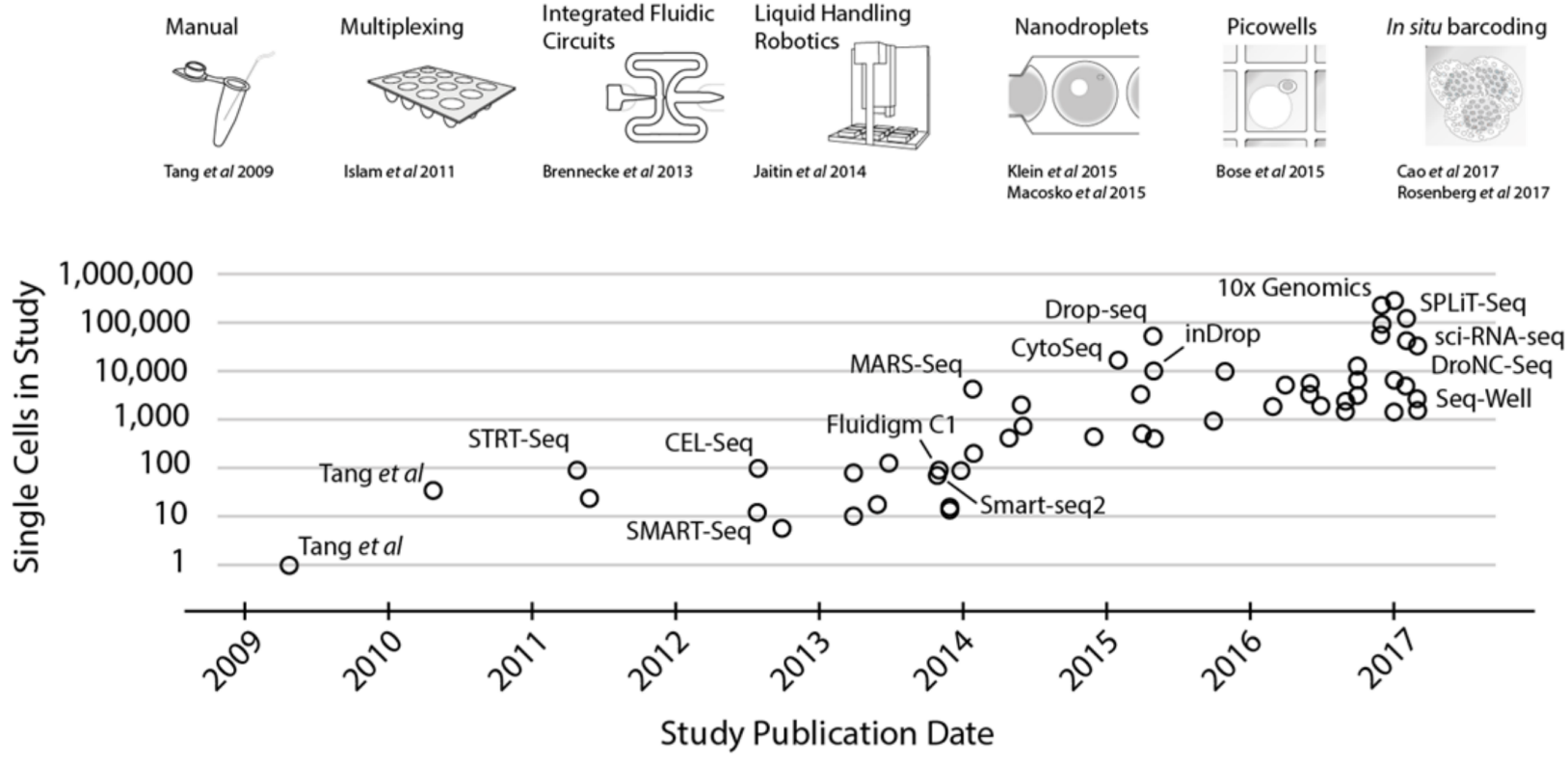
The first step is isolating cells from one another.

First generation microfluidics - Fluidigm C1



Captures up to 96 cells on a chip.

Exponentional scaling of single cell gene expression methods



adapted from arXiv:1704.01379 (https://arxiv.org/abs/1704.01379)

Droplet-based technologies enable thousands of cells to be sequenced

Droplet-based technologies enable thousands of cells to be sequenced

Capture of cells in 1nL droplets

Droplet-based technologies enable thousands of cells to be sequenced

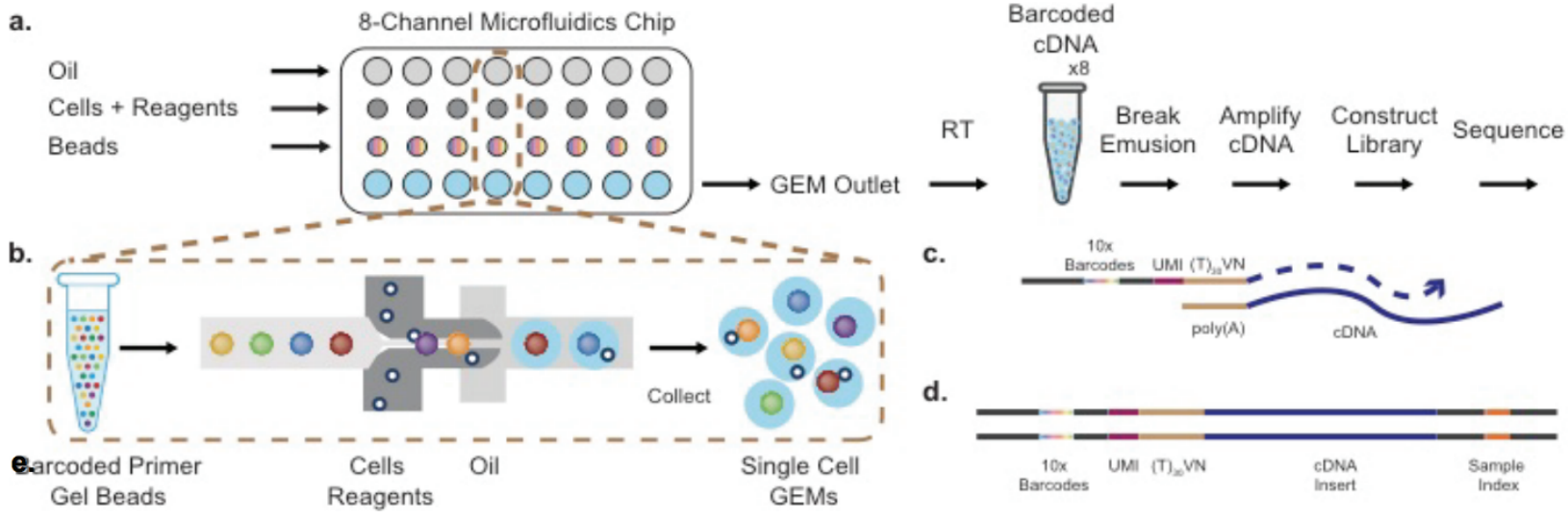
Capture of cells in 1nL droplets

Gel beads contain reagents for creation of cDNA.

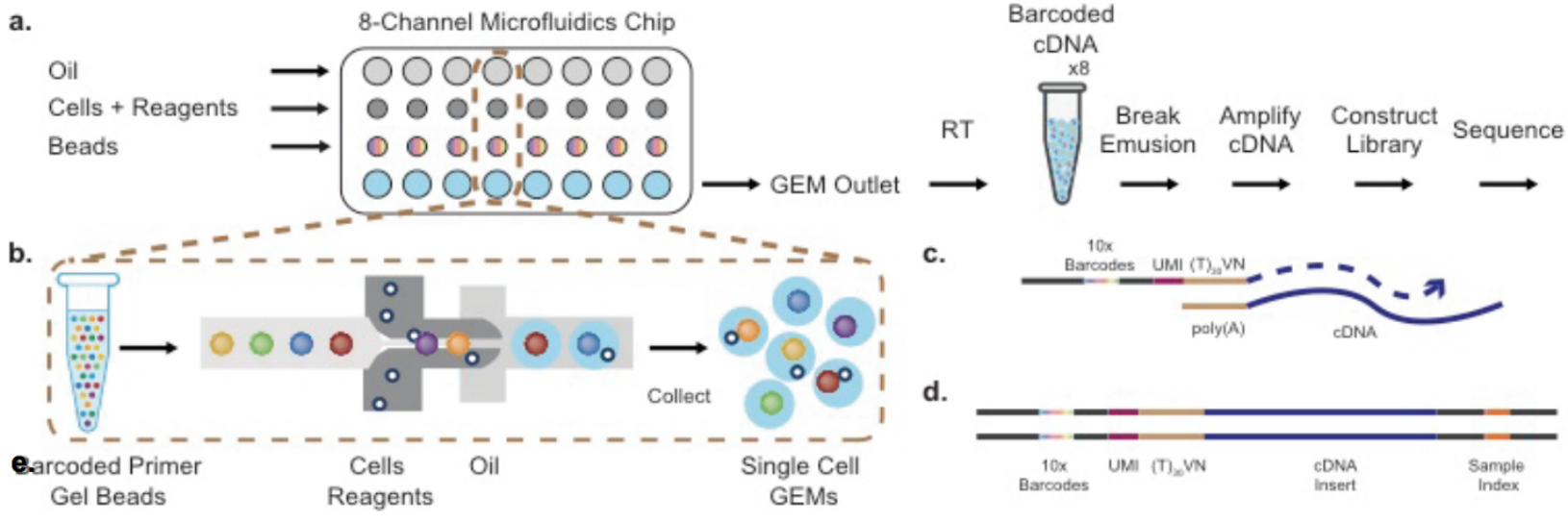
Major challenge: conquering double-Poisson loading of droplets.

Solution: severely limit the number of cells versus reagents. Leads to fewer "multiplets" but a lot of empty droplets and wasted reagents.

10X Genomics Chromium System

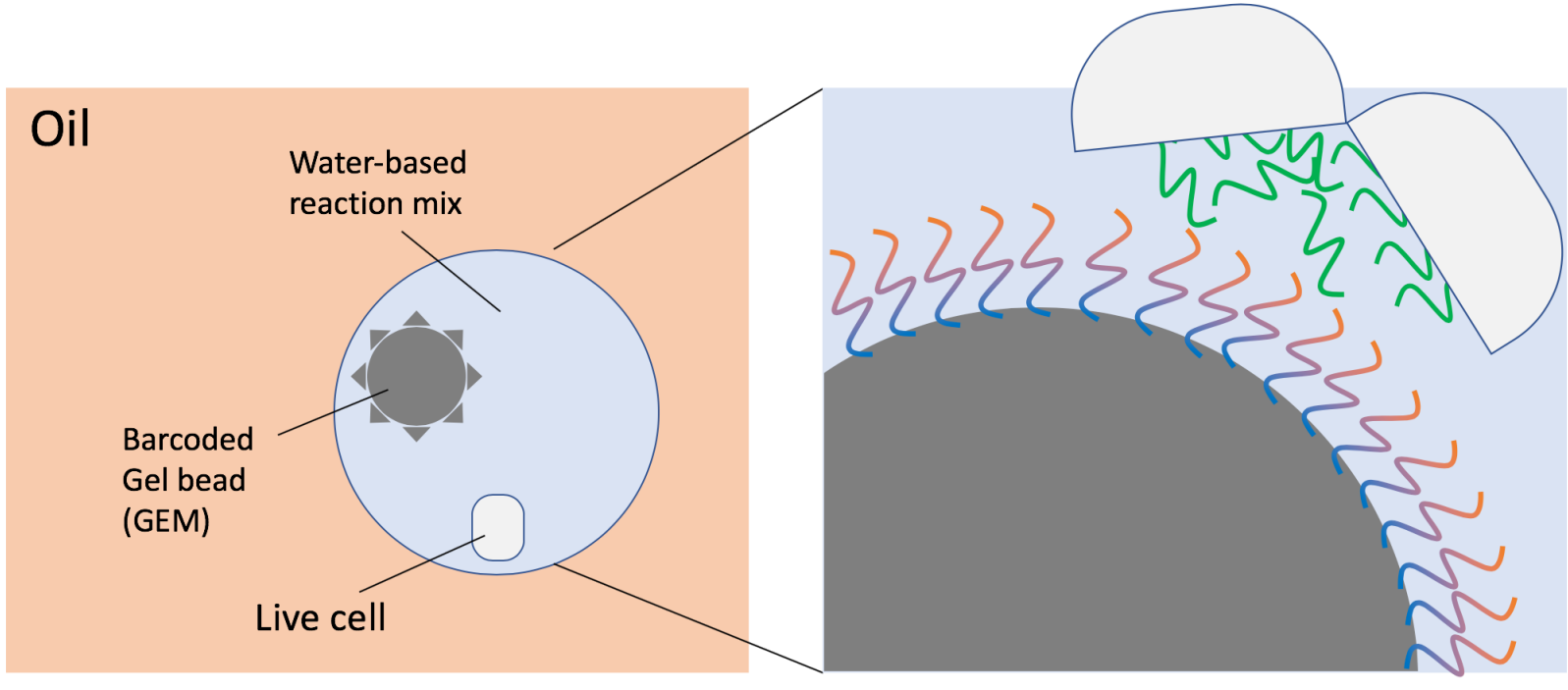


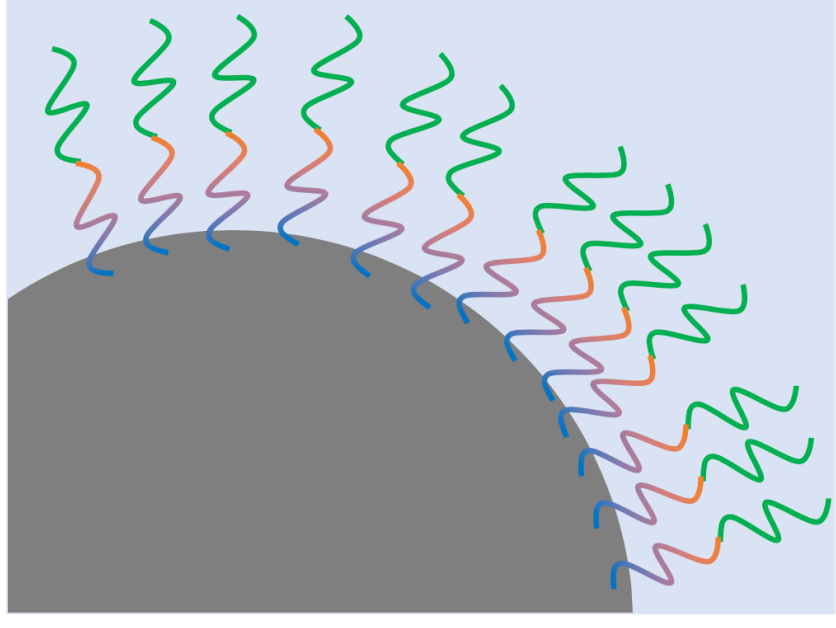
10X Genomics Chromium System

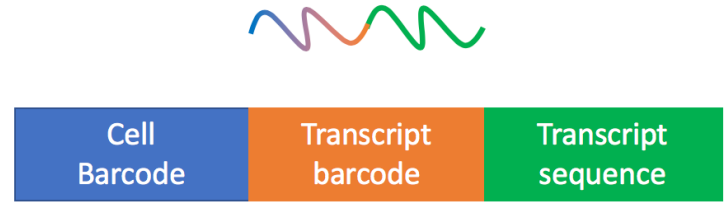


- Gel bead in Emulsion (GEM)
- High GEM fill ratio (~90% of droplets contain beads)
- Poisson Loading of Cells in GEMs
- Beads dissolve for efficient, liquid phase biochemistry
- Cell lysis starts immediately following encapsulation
- Up to 8 channels/samples processed in parallel
- Up to 6,000 cells/channel, 48,000 cells/chip
- ~50% cell processing efficiency

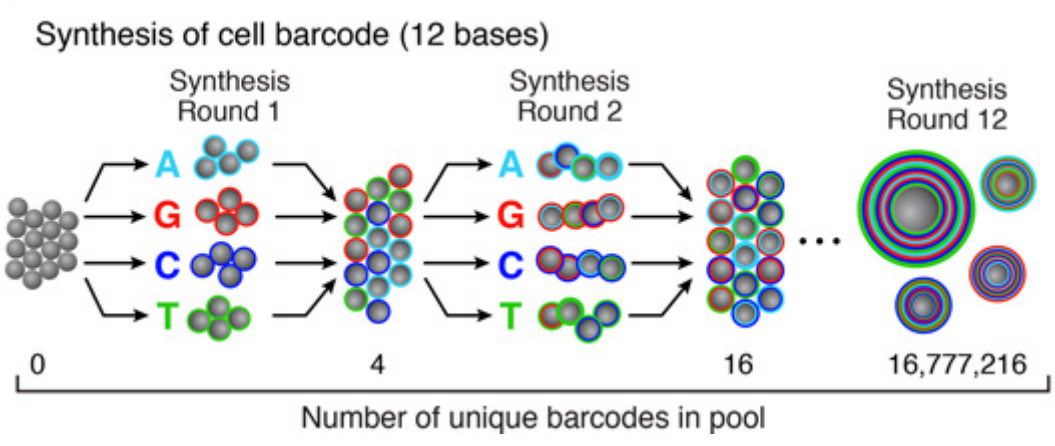
Zoomed in


Oil
Water-based
reaction mix
Barcoded
Gel bead
(GEM)
Live cell

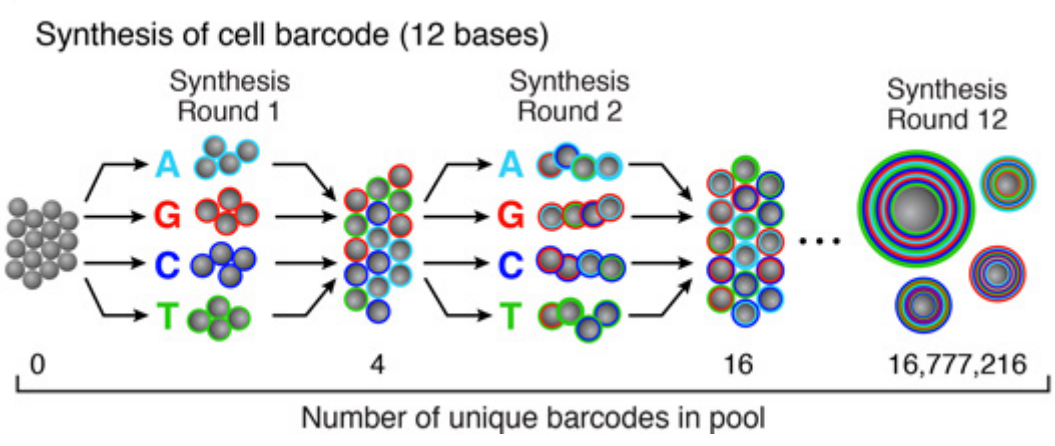



Cell Barcode
Transcript barcode
Transcript sequence

Transcripts from each cell are labeled with a unique, cell-specific barcode



Transcripts from each cell are labeled with a unique, cell-specific barcode



Each transcript is labeled with a randomized barcode

Unique Molecular Identifier (UMI) which help control PCR amplification bias.

Each molecule from all cells has a unique (cell barcode, UMI) pair, allowing them to be pooled together for sequencing.

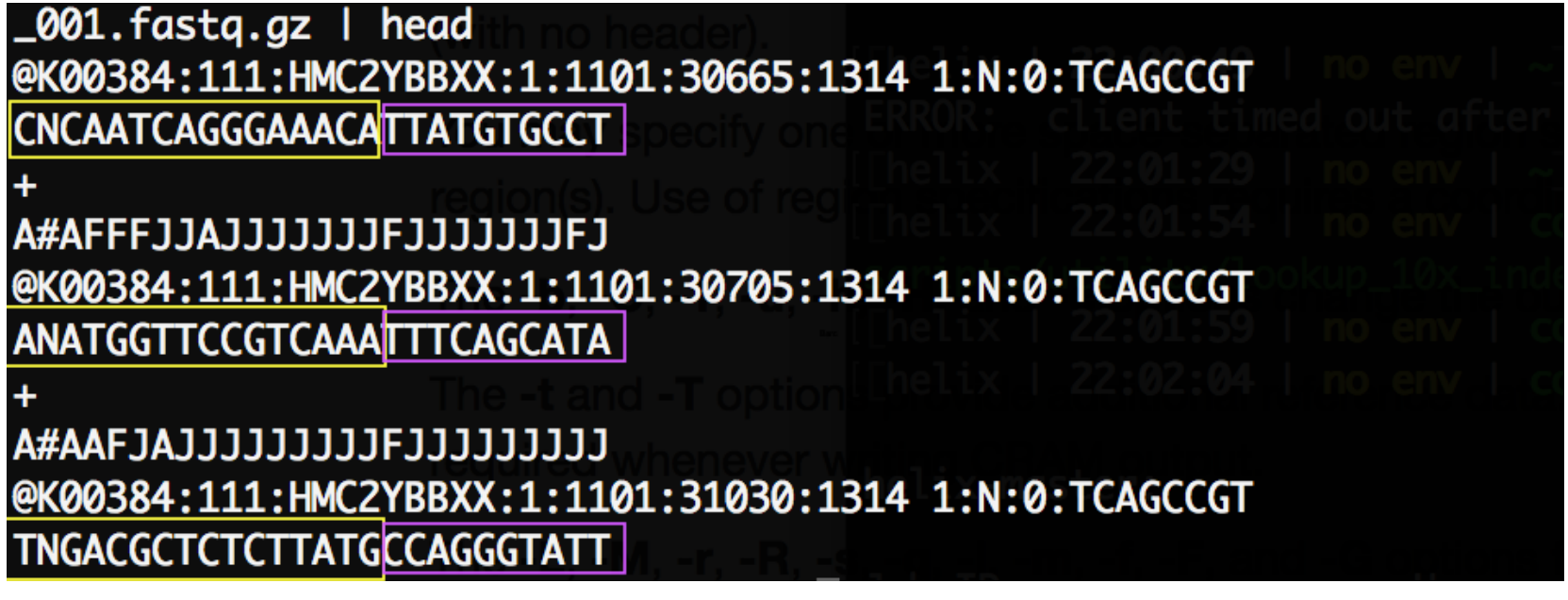
During sequencing, paired end reads are used to capture these barcodes and the mRNA transcript

Sequencing with asymmetric paired-end reads. 1 short technical read, 1 longer biological read.

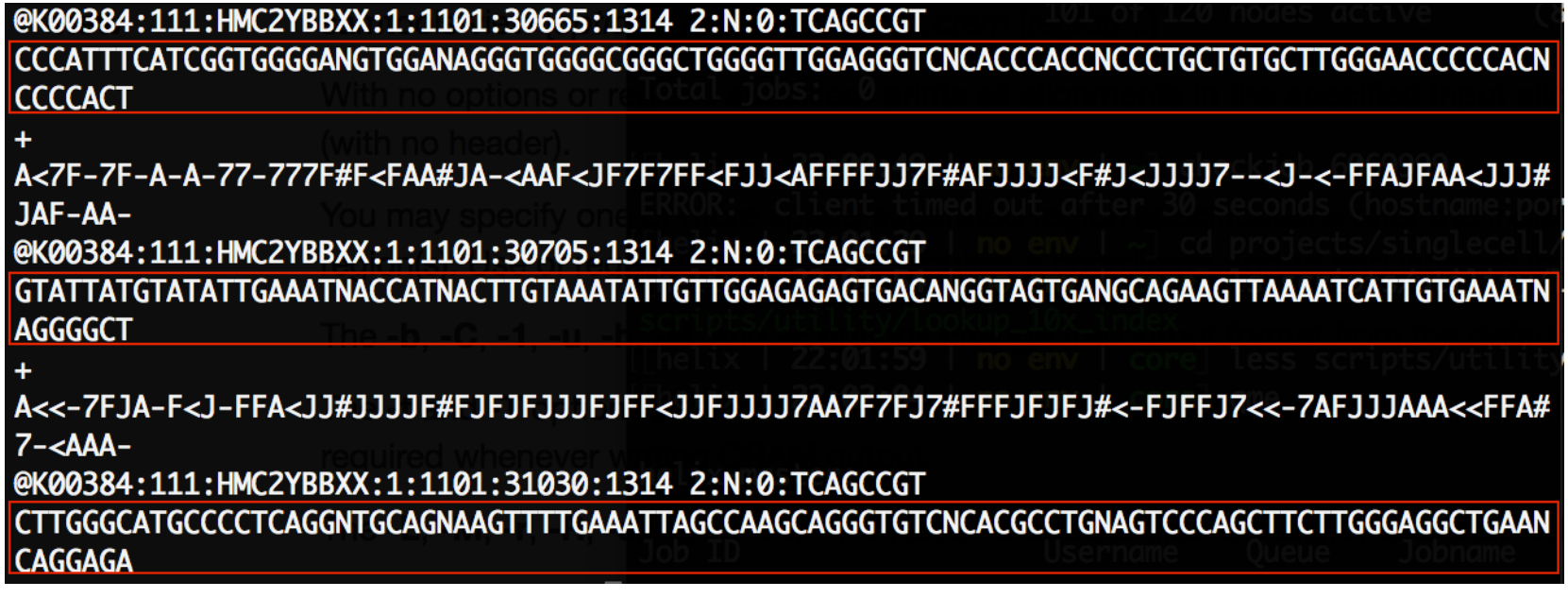
During sequencing, paired end reads are used to capture these barcodes and the mRNA transcript

Sequencing with asymmetric paired-end reads. 1 short technical read, 1 longer biological read.

Read 1 (technical)



Read 2 (biological)

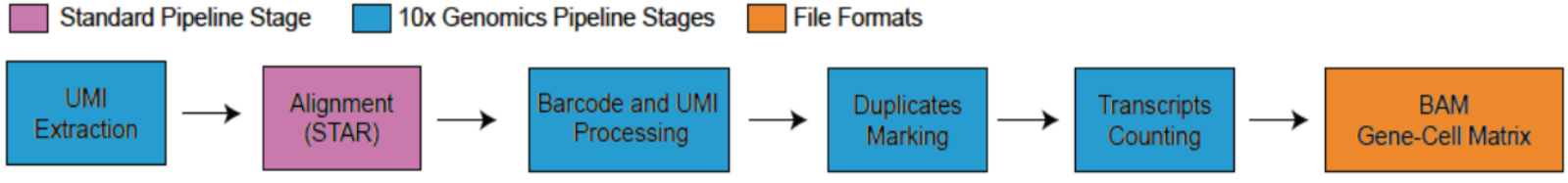


Carries some fragment of the transcript. In this case, it's the 3' end.

From this point, it's all informatics

Reads -> Cell x Gene count matrix -> Insights

Typical processing pipeline



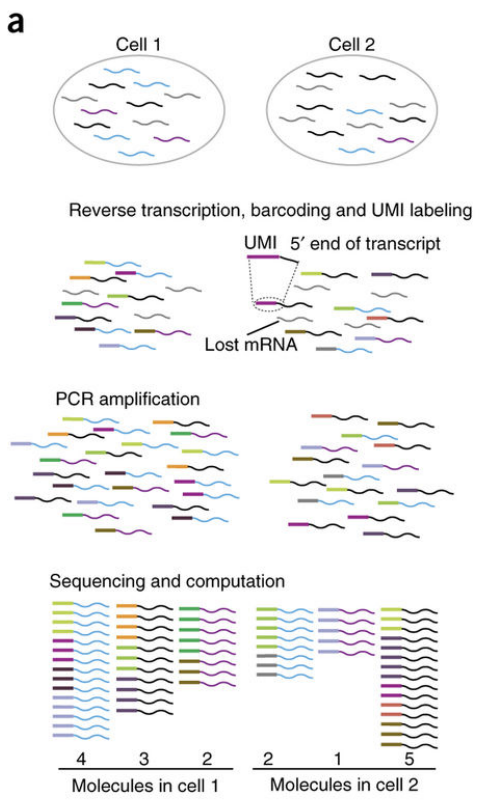
This is the 10X Genomics "CellRanger" pipeline, but there are alternatives that all do roughly the same thing:

- CellRanger (https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger)
- DropSeqTools (https://github.com/broadinstitute/Drop-seq/releases)
- Salmon Alevin (https://salmon.readthedocs.io/en/latest/alevin.html)
- Kallisto Bustools (https://www.kallistobus.tools/)
- STARsolo (https://github.com/alexdobin/STAR/blob/master/docs/STARsolo.md)

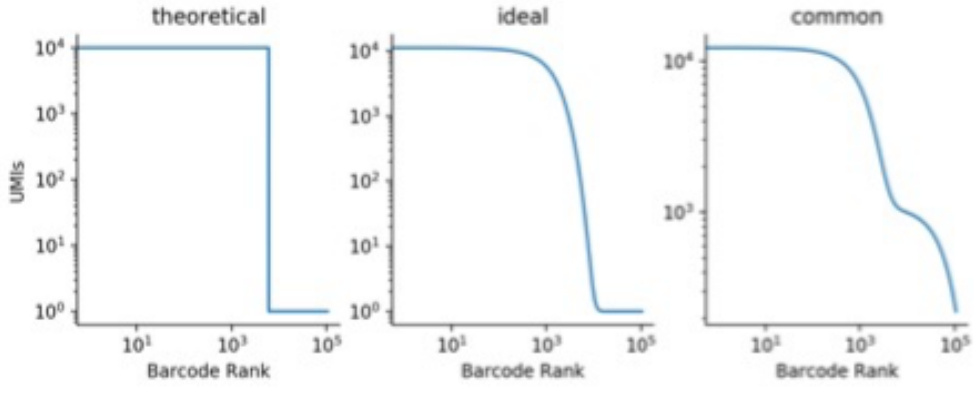
Reads are aligned

```
NB501381:38:H72W3BGXY:1:21207:16427:4789        272     1       11911   0       50M6S   *
0       0       TCATGTGTATTTGCTGTCTCTTAGCCCAGACTTCCCGTGTCCTTTCCACCCCCCCA /EEAEEEEEEEEEEA<EEAEEEE
EEEEEEEEEEEAEEEEAEEEEEEEEEE//A</A       NH:i:8  HI:i:3  AS:i:49 nM:i:0  RE:A:I  BC:Z:CTTATGTA
   QT:Z:AAAAAEEE    CR:Z:AGCGATACCGCAAT     CY:Z:AAAAAEEEEAEEEA     CB:Z:AGCGATACCGCAAT-1   UR:Z
:CGTCATCGCG UY:Z:AAAA/EEEE/ UB:Z:CGTCATCGCG RG:Z:PR16002:MissingLibrary:1:H72W3BGXY:1
NB501381:38:H72W3BGXY:3:11604:5488:17233        272     1       11913   0       55M1S   *
0       0       ATGTGTATTTGCTGGCTCTTAGCCCAGACTTCCCGTGTCCTTTCCACCGGGCCTTA /EEAA///6EAE/</EEE/EEEE
EA/E/AEEAEAE/EEEEE/EE/AEEEAEAAAAA       NH:i:5  HI:i:4  AS:i:52 nM:i:1  RE:A:I  BC:Z:AACGCCCT
   QT:Z:AAA/AAEA    CR:Z:AGCGATACCGCAAT     CY:Z:A/////EA//AA//     CB:Z:AGCGATACCGCAAT-1   UR:Z
:CGTCATCGCG UY:Z:AA/AAEE/A/ UB:Z:CGTCATCGCG RG:Z:PR16002:MissingLibrary:1:H72W3BGXY:3
NB501381:38:H72W3BGXY:3:13608:26685:16497        272     1       11915   0       55M1S   *
0       0       GTGTATTTGCTGTCTCTTAGCCCAGACTTCCCGTGTCCTTTCCACCGGGCCTTTGT EEEEE/<EEEEE/EEEEEEEEEE
EEEEAEEEEEEE6EEE6EEEEEEEEEEEAAAAA       NH:i:5  HI:i:2  AS:i:54 nM:i:0  RE:A:I  BC:Z:TGACGAGC
   QT:Z:AAAAAEEE    CR:Z:AGCGATACCGCAAT     CY:Z:AAAAAAEEEEEAEE     CB:Z:AGCGATACCGCAAT-1   UR:Z
:CGTCATCGCG UY:Z:A//AAEEEEE UB:Z:CGTCATCGCG RG:Z:PR16002:MissingLibrary:1:H72W3BGXY:3
```

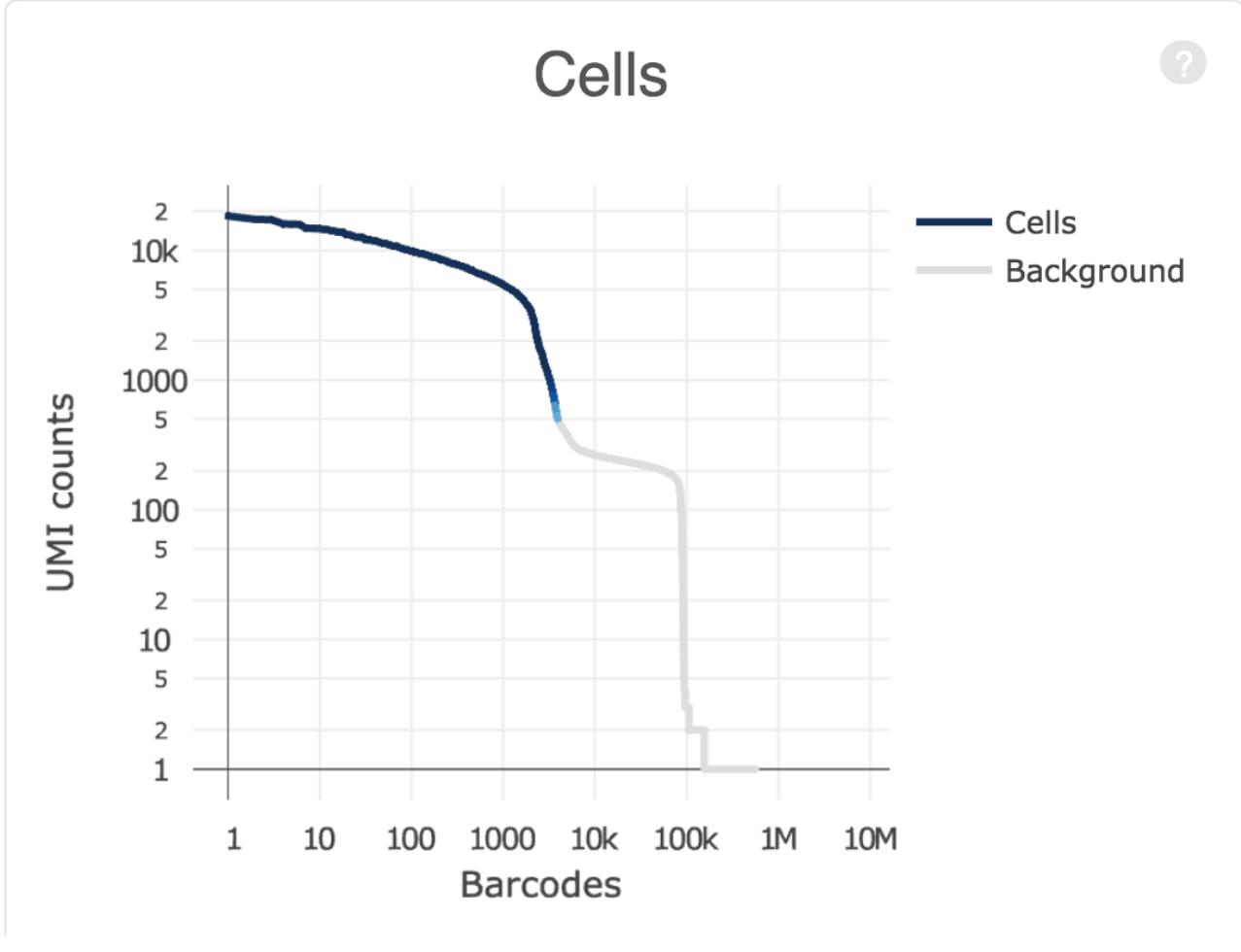
De-duplicating UMIs



Distinguishing cell from barcodes



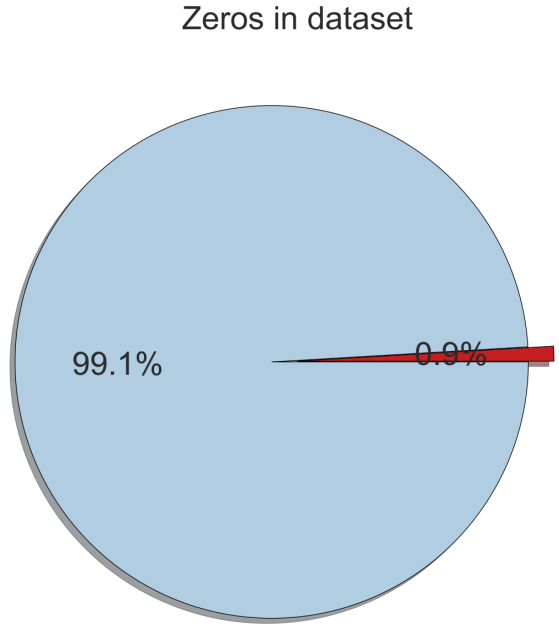
Common cell-calling distribution



Yields a *digital* expression matrix

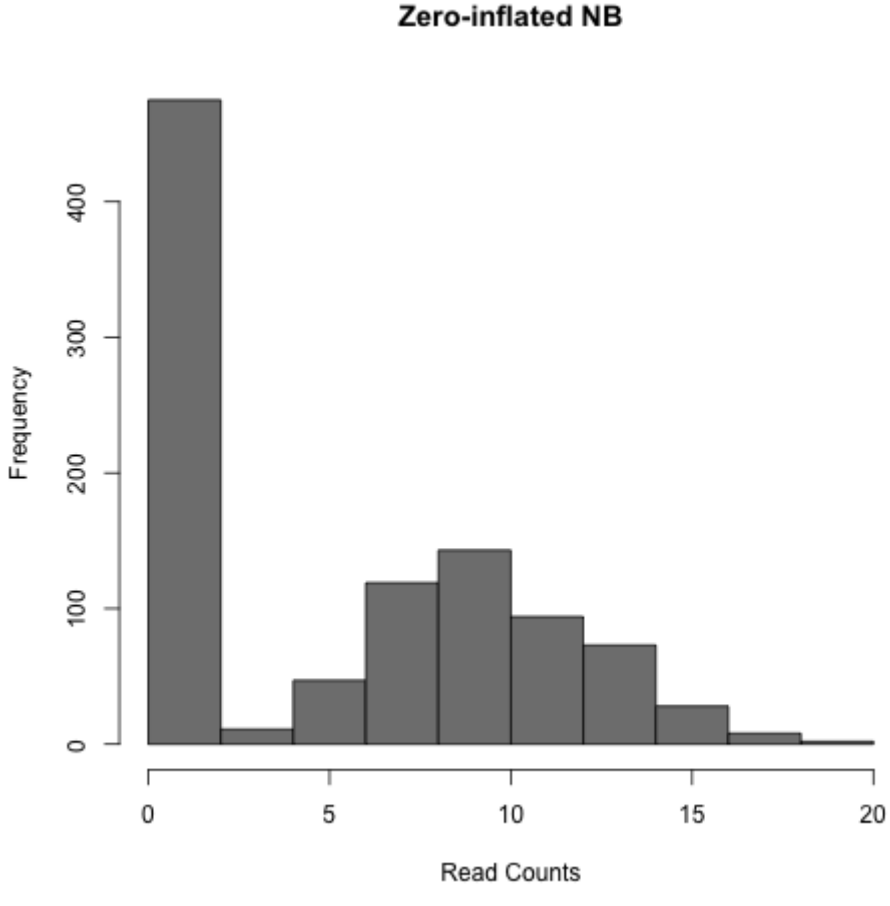
	$Cell_1$	$Cell_2$	…	$Cell_n$
$gene_a$				
$gene_b$				
$gene_c$				
⋮				
$gene_n$				

However, this matrix is extremely sparse

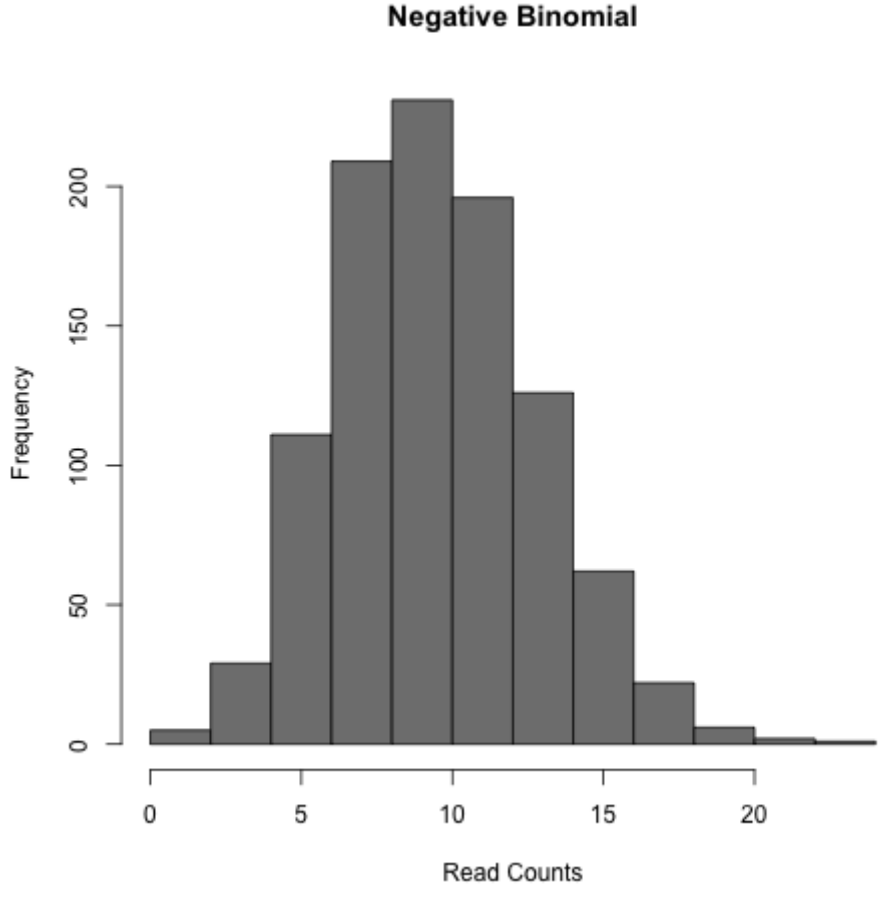


Overcoming this sparsity is a major challenge in the field.

scRNA-seq data is described by a zero-inflated negative binomial distribution



In comparison, bulk RNA-seq is described by a negative binomial distribution



Analysis overview

Analysis overview

Going from matrix to insights

The goal of every scRNA-seq experiment is to describe the heterogeneity of cell types (and within cell types) within the sample population sequenced.

Analysis overview

Going from matrix to insights

The goal of every scRNA-seq experiment is to describe the heterogeneity of cell types (and within cell types) within the sample population sequenced.

This analytical process typically involves the following steps:

- Filtering low quality cells
- Filtering really sparsely expressed genes
- Normalize gene expression
- Feature selection for clustering
- Dimensionality reduction
- Cluster identification
- Marker gene identification
- Cell type inference

Each of these steps involves making choices that affect downstream processing.

Faced with challenges

How to...

- discriminate lowly expressing cells from ambient mRNA?
- reliably call expressed genes?
- cluster cells without bias?
- id cell types without clustering?
- infer celltypes from marker genes? without biological insight?

Faced with challenges

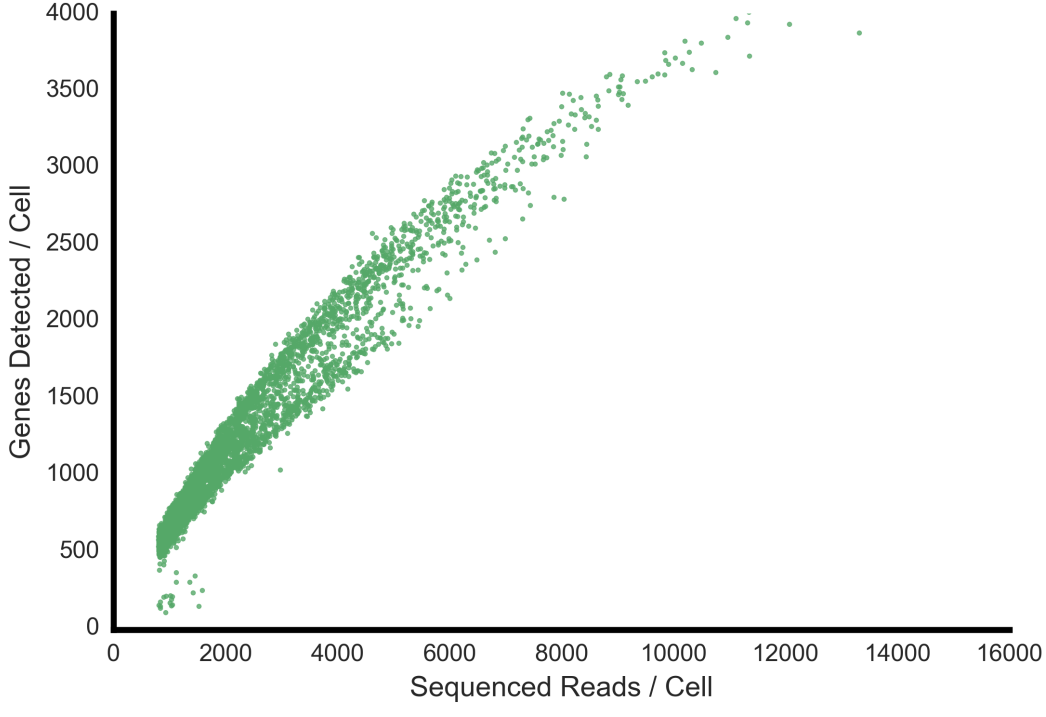
How to...

- discriminate lowly expressing cells from ambient mRNA?
- reliably call expressed genes?
- cluster cells without bias?
- id cell types without clustering?
- infer celltypes from marker genes? without biological insight?

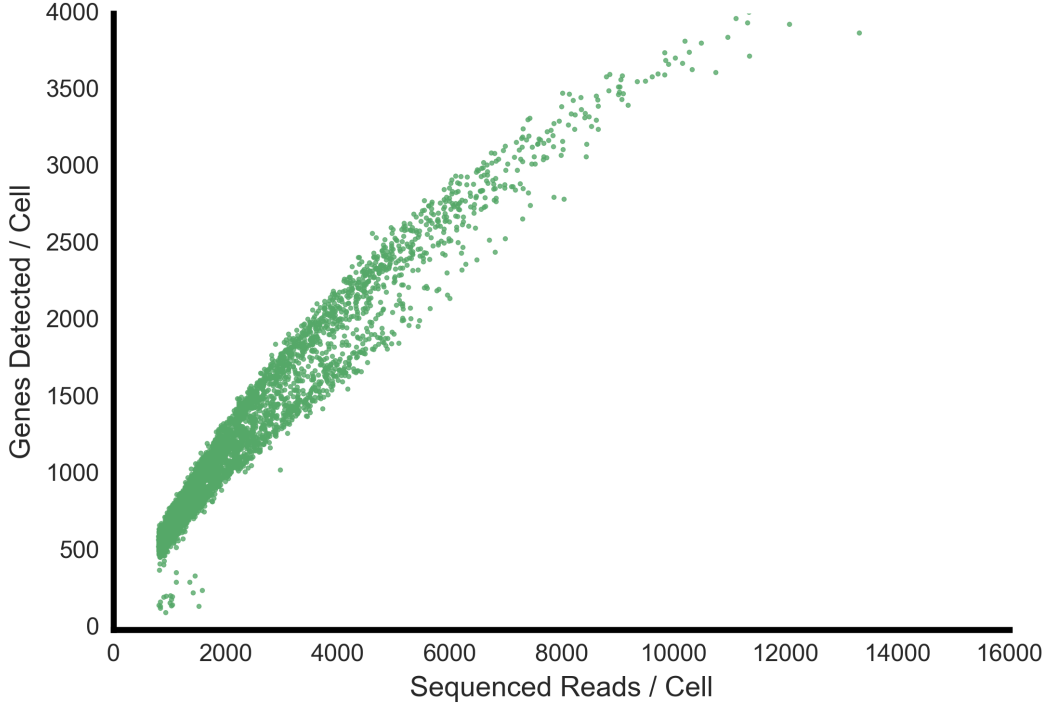
Philosophical challenges

- How to define a "cell type"?
- Do cells that cluster together always belong to the same "type"?

How variable is the data



How variable is the data



- Low quality cells with `<250` genes per cell
- (sub)linear relationship between sequencing depth and genes detected per cell

Normalization

The number of transcripts captured and subsequent reads generated in each cell is stochastic. It is common to correct for that using ***Library size*** normalization which divides the counts of each gene in a cell by that cell's total UMI count (the library size). Counts are then often multiplied by a scaling factor.

$$\text{counts}^{norm}_{cell_i,gene_j} = \frac{\text{counts}_{cell_i,gene_j}}{\sum_{gene_j} \text{counts}_{cell_i,gene_j}} * \text{scaling\_factor}$$

Normalization

The number of transcripts captured and subsequent reads generated in each cell is stochastic. It is common to correct for that using ***Library size*** normalization which divides the counts of each gene in a cell by that cell's total UMI count (the library size). Counts are then often multiplied by a scaling factor.

$$\text{counts}^{\text{norm}}_{\text{cell}_i,\text{gene}_j} = \frac{\text{counts}_{\text{cell}_i,\text{gene}_j}}{\sum_{\text{gene}_j}\text{counts}_{\text{cell}_i,\text{gene}_j}} * \text{scaling\_factor}$$

Options for this scaling factor are:

- 1,000,000 (resultant is counts per million (CPM))
- the median library size across cells
- literally any arbitrary number (like 10,000 or 2^14)

Normalization

The number of transcripts captured and subsequent reads generated in each cell is stochastic. It is common to correct for that using ***Library size*** normalization which divides the counts of each gene in a cell by that cell's total UMI count (the library size). Counts are then often multiplied by a scaling factor.

$$\text{counts}^{\text{norm}}_{\text{cell}_i,\text{gene}_j} = \frac{\text{counts}_{\text{cell}_i,\text{gene}_j}}{\sum_{\text{gene}_j}\text{counts}_{\text{cell}_i,\text{gene}_j}} * \text{scaling\_factor}$$

Options for this scaling factor are:

- 1,000,000 (resultant is counts per million (CPM))
- the median library size across cells
- literally any arbitrary number (like 10,000 or 2^14)

Additionally, we can account for any variation in the variance of genes (highly expressed genes have higher variance in expression) by performing a sublinear transform (like `log` or `sqrt`) to make the variances more normally distributed.

$$\log(\text{counts}^{\text{norm}}_{\text{cell}_i,\text{gene}_j} + 1)$$

Dimensionality reduction

Even after filtering, many genes (typically ~10,000 - 15,000) remain expressed in at least a few cells. This is a very large multi-dimensional space.

Effective analysis in spaces with more than a few hundred dimensions is difficult (Curse of Dimensionality). In particular, distances between points tend toward uniformity, making finding clusters or determining differences challenging.

What is the best way to proceed?

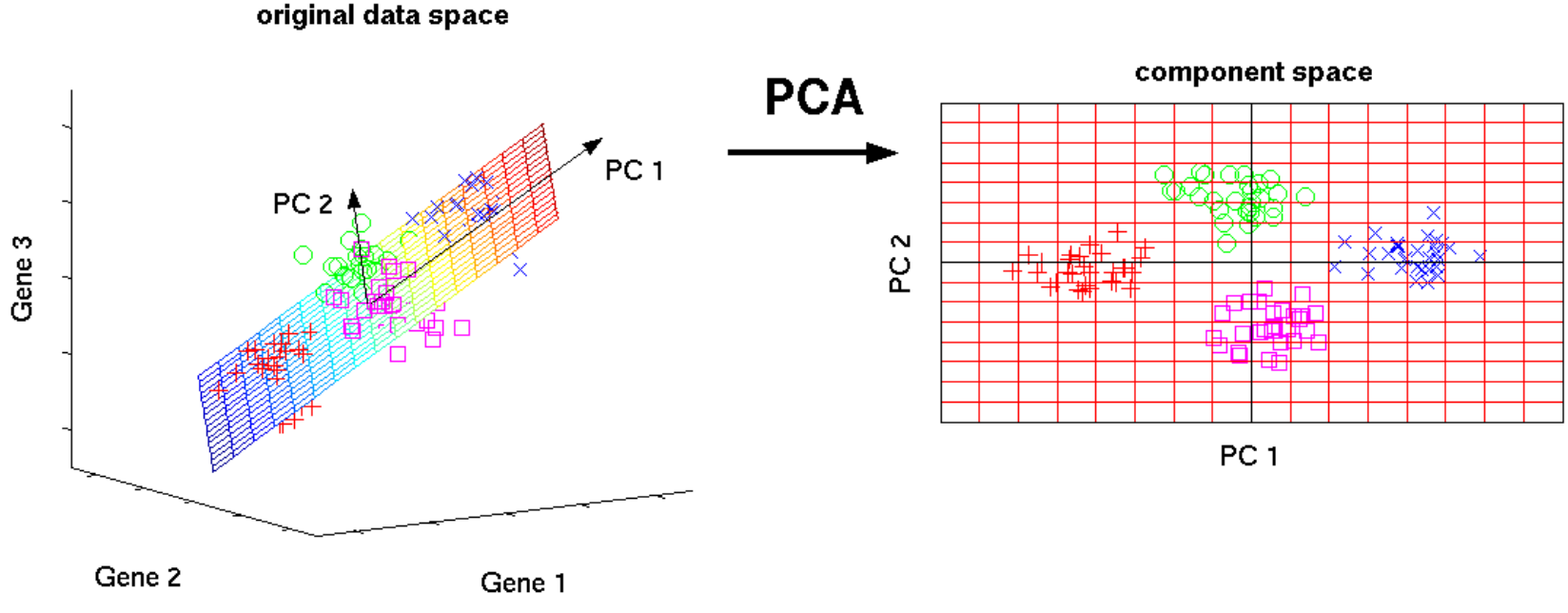
Gene selection

Perhaps the most important step in single cell transcriptome analysis is to select which subset of genes is used to represent the original high dimensional space of the data. This greatly affects the outcome of dimensionality reduction.

Some common ways:

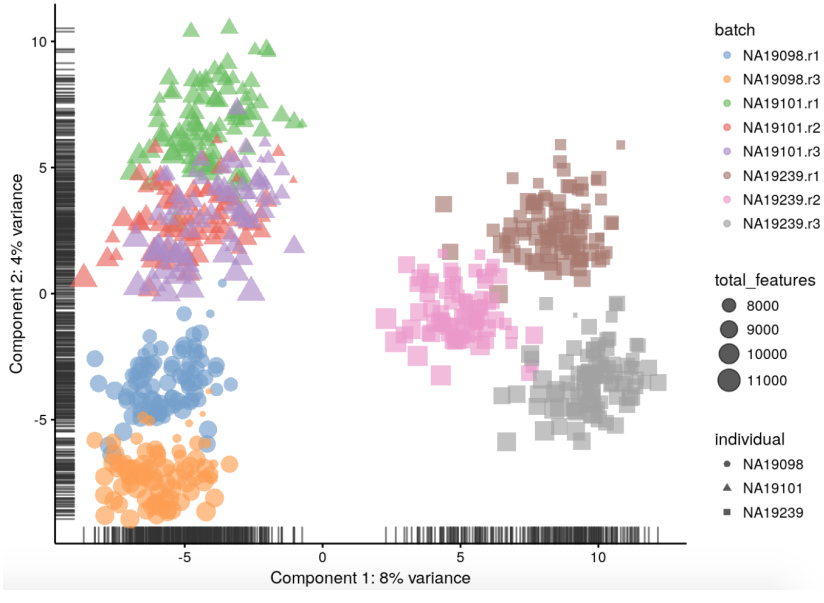
- All expressed genes after filters (~15,000)
- Filter genes expressed in > 95% of cells and < 5% of cells
- Top variable genes
- Overdispersed genes (ratio of variance to mean)
- Highly expressed in 1 cell

First step - Principal Component Analysis

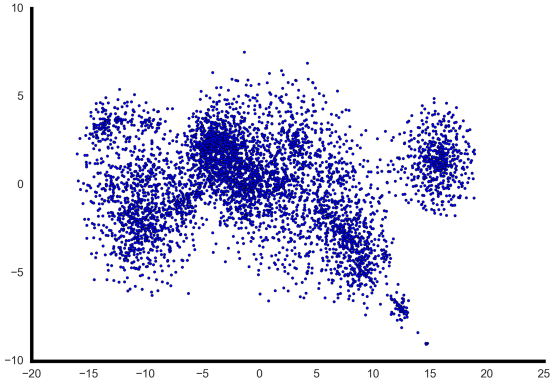


Martin Hemberg's scRNA-seq course (https://hemberg-lab.github.io/scRNA.seq.course/index.html)

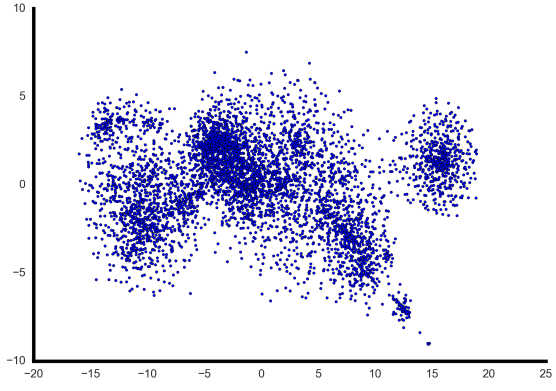
PCA on normalized counts



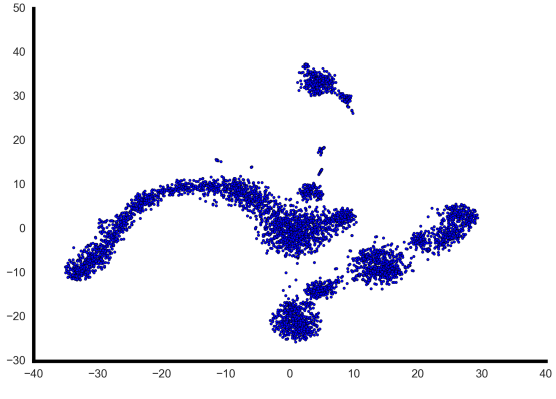
Using high variance genes



Using high variance genes

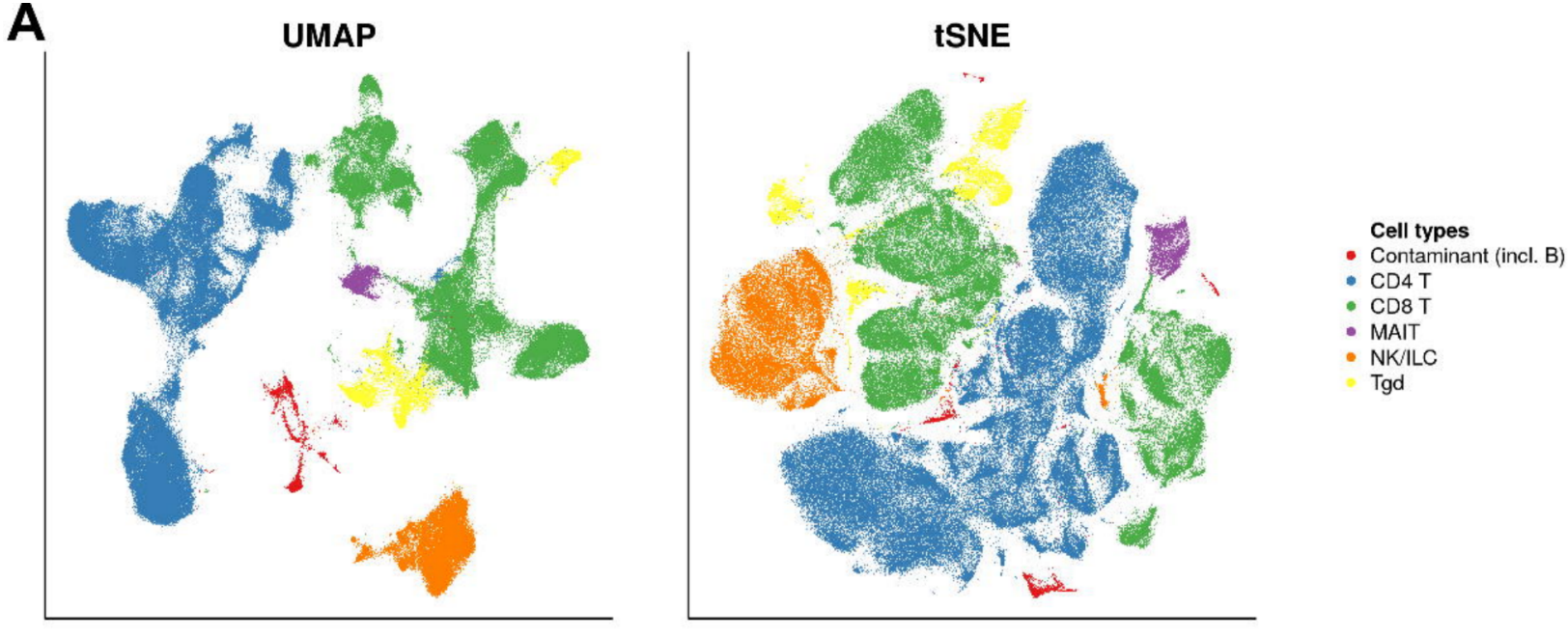


Using high dispersion genes



Second step - manifold embeddings

Unlike PCA where it's up to you to determine how many components are important, t-distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP) project data from the high dimensional space into 2 or 3 dimensions preserving notion of "closeness".



Becht et al. Nat Biotech 2019 (https://www.nature.com/articles/nbt.4314)

Clustering

No gold standard method currently and field is searching for best practice.

Typical options are:

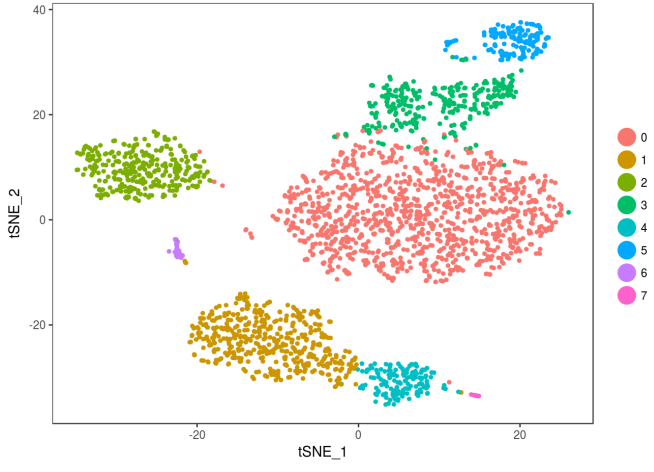
- Spectral clustering or kNN clustering in PCA space.
- Consensus or heirarchical clustering in PCA space.
- Density-based cluster in PCA or t-SNE/UMAP space.
- Constructing a graph/network and using community detection algorithms to assign clusters.

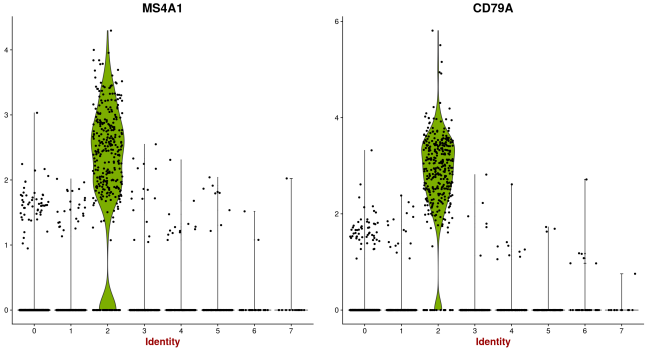
Cluster identification

Once clusters have been assigned, now it's time to return to biology. Usually clusters are assigned a cell type or cell state through the use of biomarker genes. This is done by

- manual curation
  - knowledge of genes expressed cell type markers from bulk RNA-seq or your biology training
  - assign labels by plotting the gene expression of those genes by cluster and iteratively refining your labels
- supervised analysis
  - use many statistical tests to find gene expression that is specific to one cluster (similar to bulk differential expression analysis)
  - still often requires looking up genes in a database or the literature to associate cell types with gene expression

For example, where are the B Cells?





Now what?

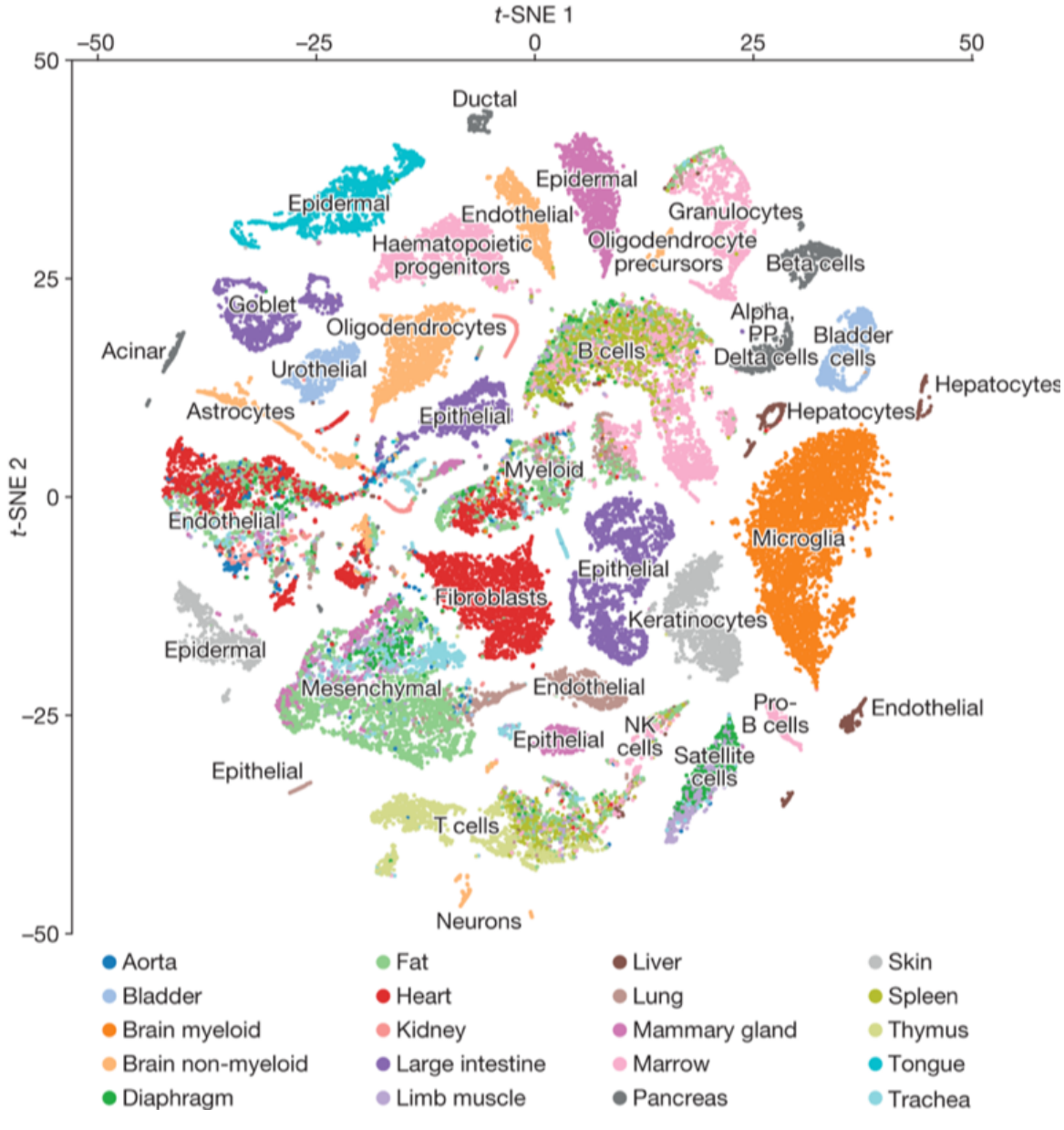
Once you have your cell types labeled, you can:

- Are there new cell types or states that you can't label? What genes define them?
- Explore heterogeneity within clusters: is your cluster oddly shaped? is that shape driven by biological processes (gene expression)?
- Compare cluster proportions among samples

Explore data further with:

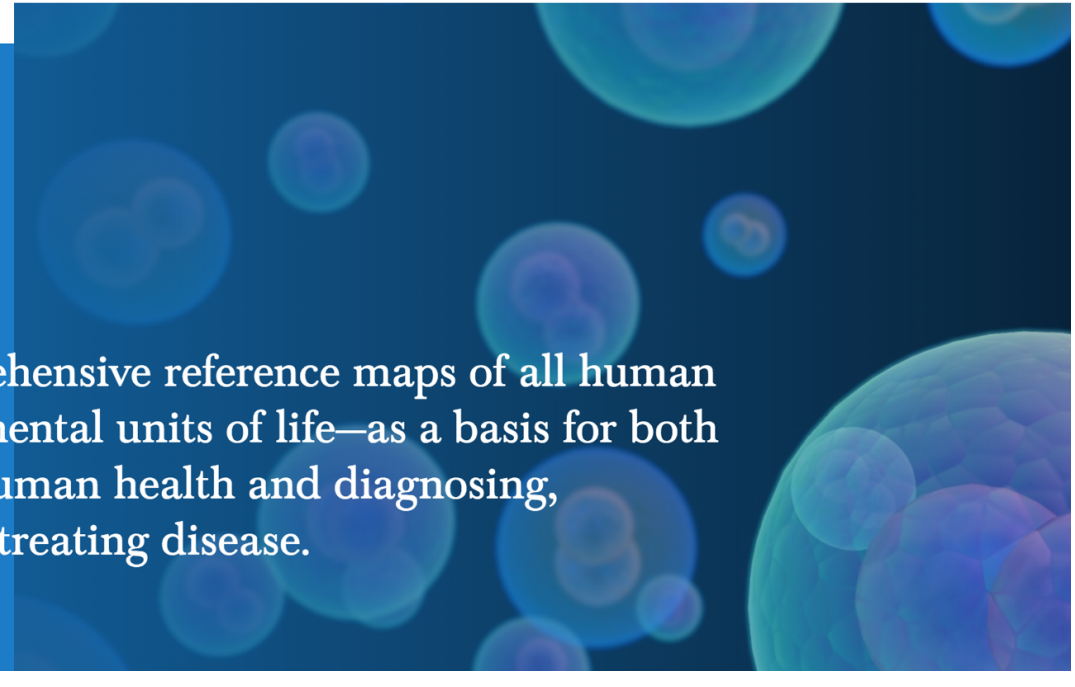
- integrate your data with more samples, either your own or those from the literature
- pseudotime analysis: infer temporal trajectories among your samples
- differential expression analysis: compute differentially expressed genes (DEGs) between similar clusters. With multiple samples, you can analyze both datasets together and compute DEGs among cells in the same cluster that have different sample conditions.

Looking towards the future



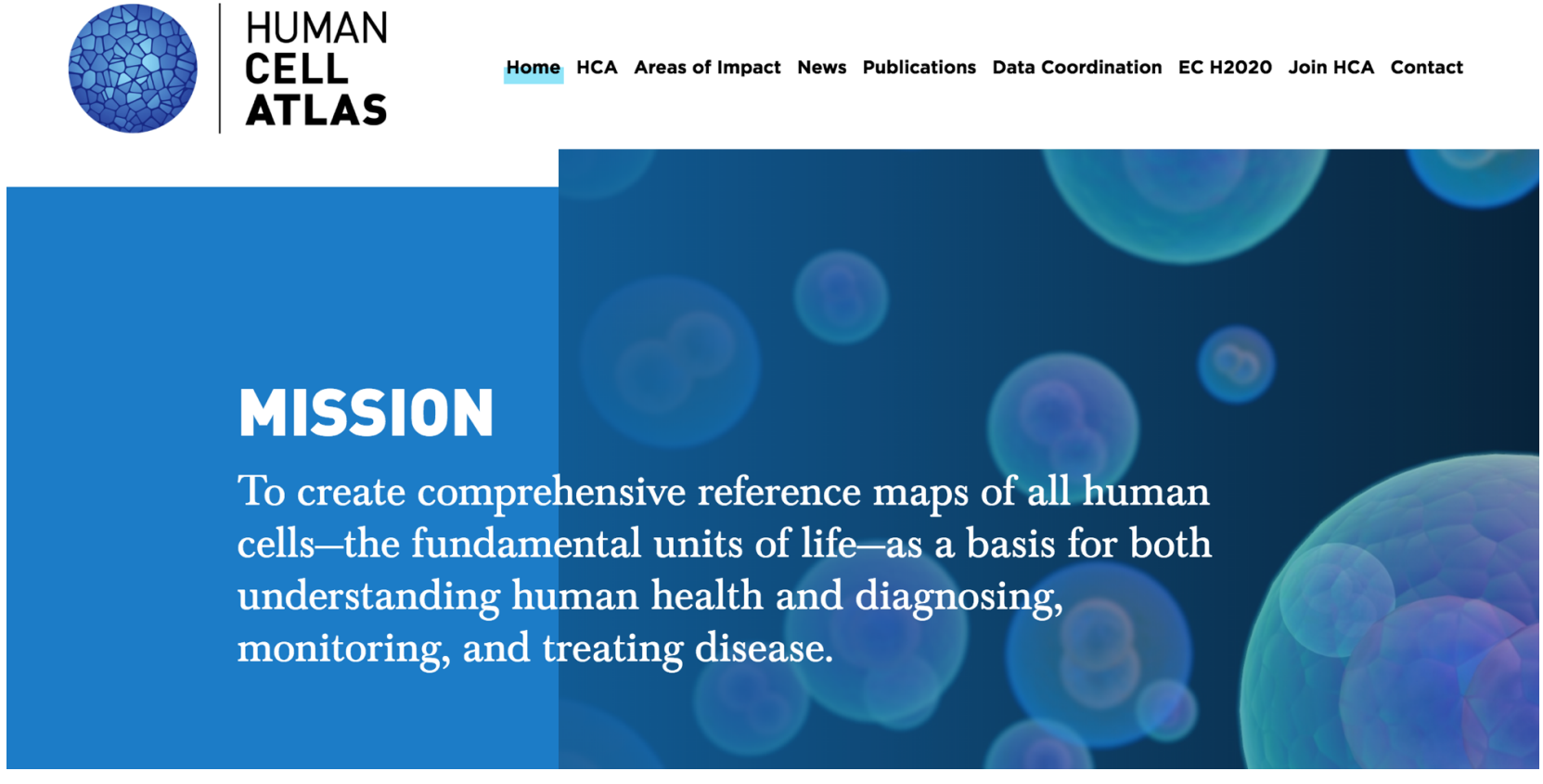
Tabula Muris (https://tabula-muris.ds.czbiohub.org/)

HUMAN CELL ATLAS



MISSION

To create comprehensive reference maps of all human cells—the fundamental units of life—as a basis for both understanding human health and diagnosing, monitoring, and treating disease.



It is paramount that new data be mapped to these reference datasets

This provides interesting challenges

- How to effectively merge datasets together? Across sequencing technology, tissue, even species?
- How to organize and store these data?
- How to effectively analyze data at this scale?
- Does this size of data provide a path to fundamentally change how we analyze/process this data?

Some things we didn't cover:

- Dataset merging/integration techniques: Harmony (https://github.com/immunogenomics/harmony), CCA (https://www.nature.com/articles/nbt.4096), MMN (https://www.nature.com/articles/nbt.4091)
- Pseudotime analysis: inferring temporal ordering to multiple samples. 75(!) methods (https://github.com/dynverse/dynbenchmark)
- Spliced vs unspliced RNA analysis: RNA velocity (https://scvelo.readthedocs.io/en/latest/) and velocyto (http://velocyto.org/velocyto.py/index.html)
- Anything involving mutations, allelic expression, etc.
- Advanced multiplexing

Where did the spatial information go?

Where did the spatial information go?

Pretty much all scRNA-seq methods require you to disassociate your tissue and in the process you lose spatial information.

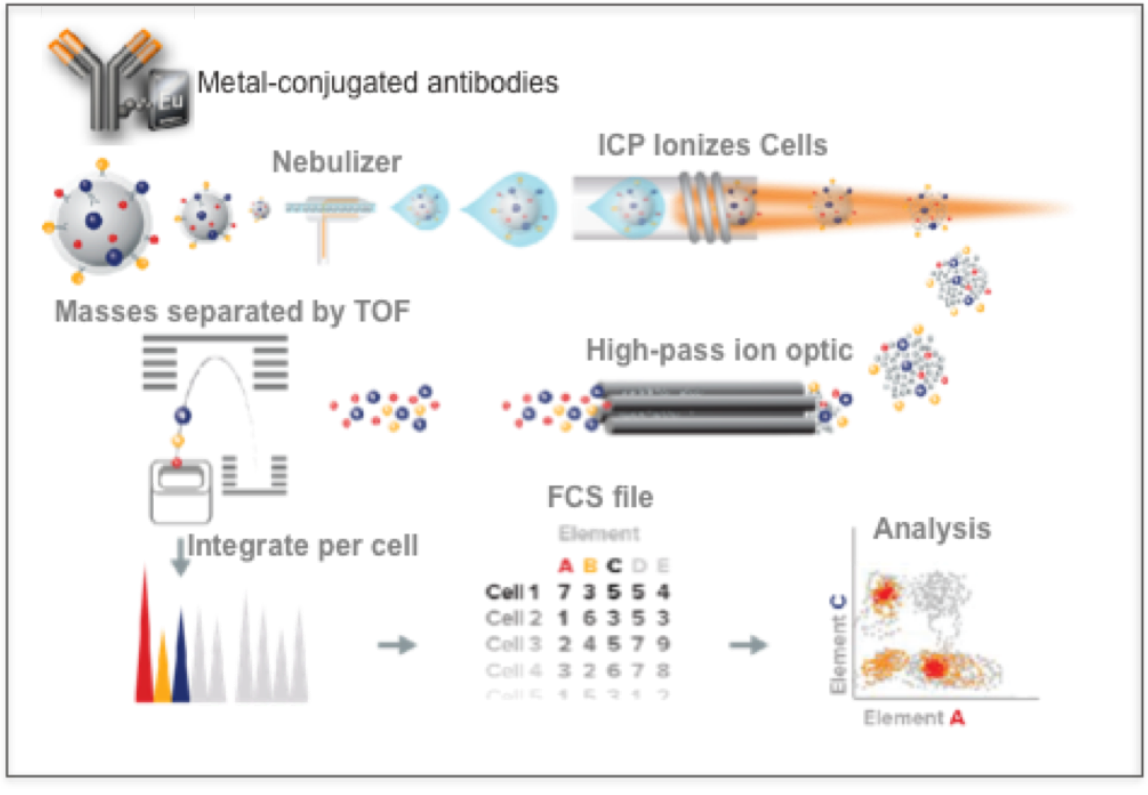
Where did the spatial information go?

Pretty much all scRNA-seq methods require you to disassociate your tissue and in the process you lose spatial information.

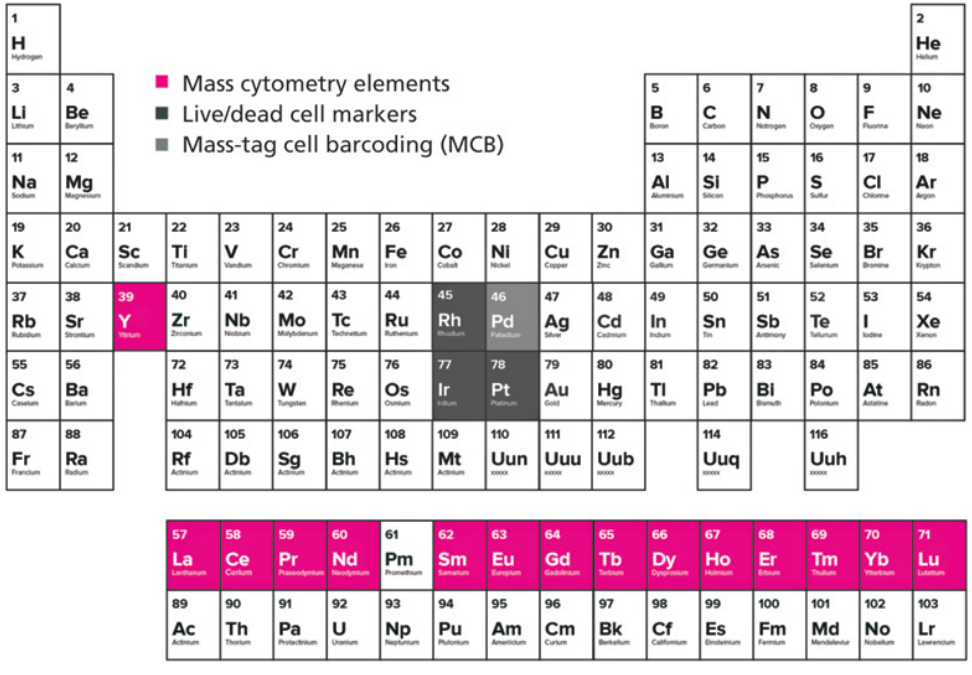
This information is valuable when you want to understand cellular **interactions**.

For example, studying the immune system interacting with different disease states, organism and embryo development

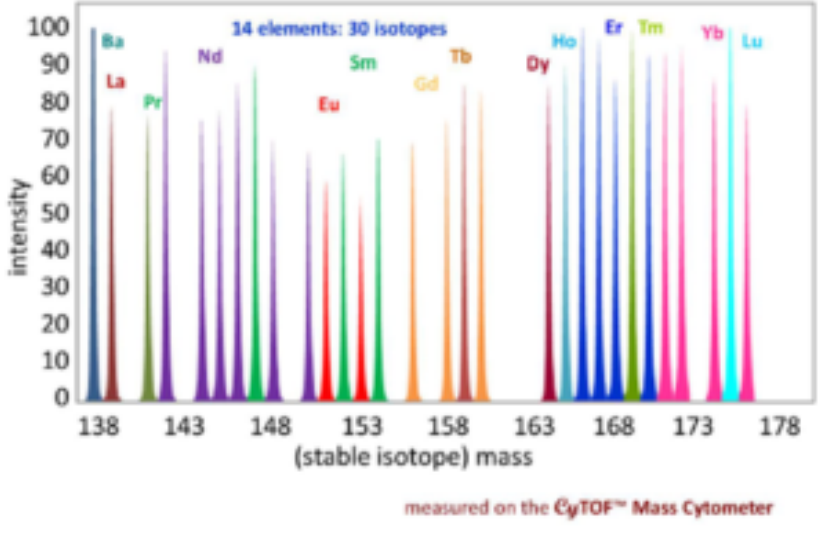
Imaging mass cytometry

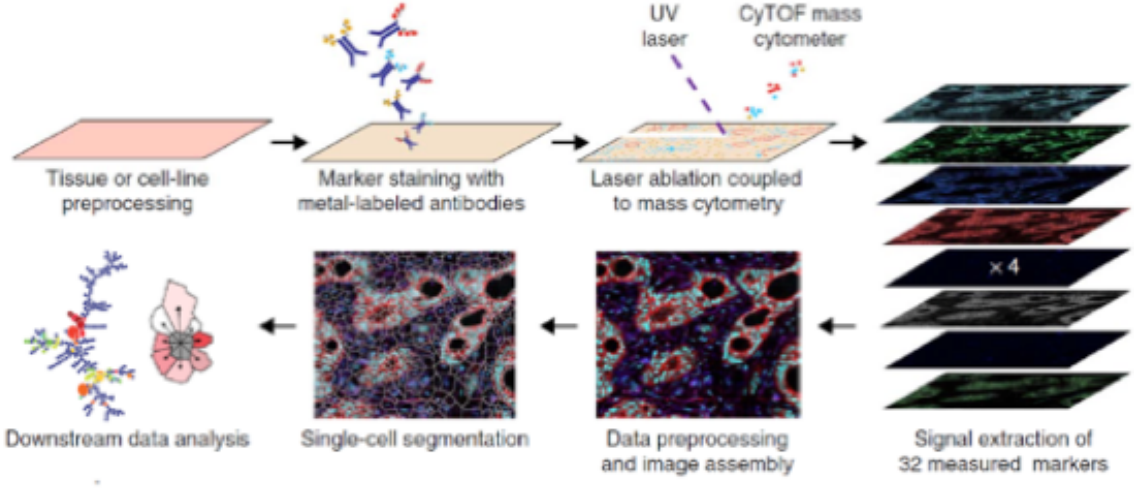


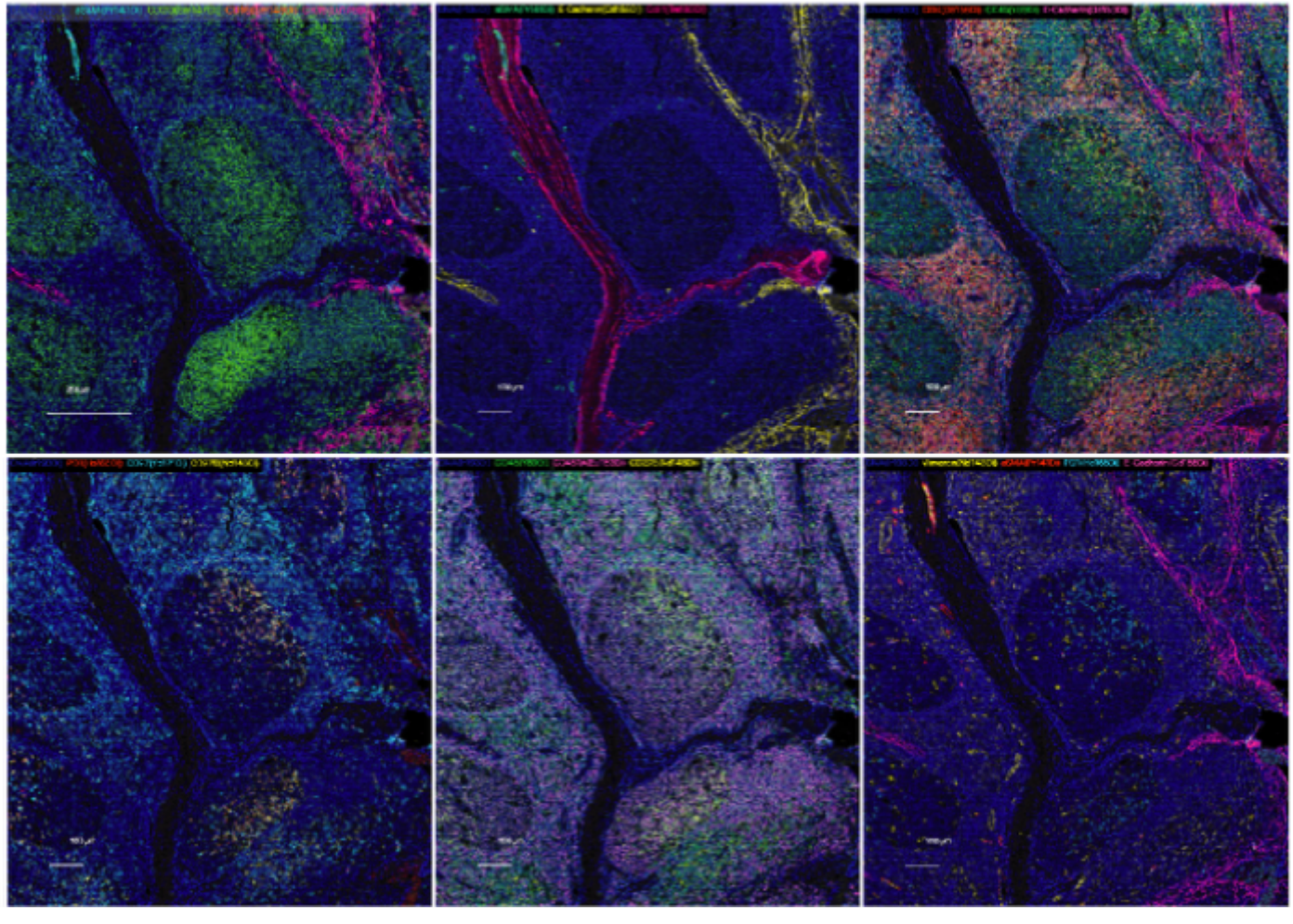
Lanthanide series



Very discrete atomic spectra

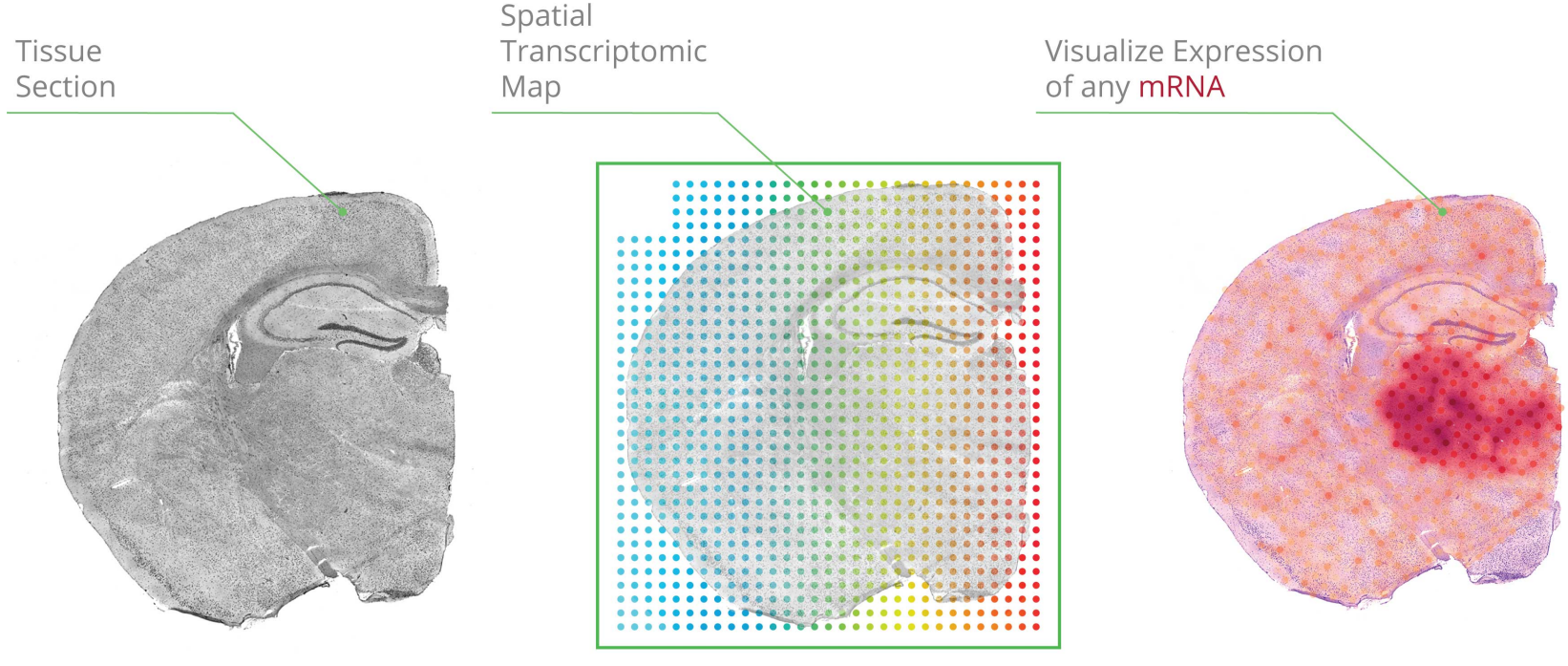



UV
laser
CyTOF mass
cytometer
Tissue or cell-line
preprocessing
Marker staining with
metal-labeled antibodies
Laser ablation coupled
to mass cytometry
×4
Signal extraction of
32 measured markers
Data preprocessing
and image assembly
Single-cell segmentation
Downstream data analysis

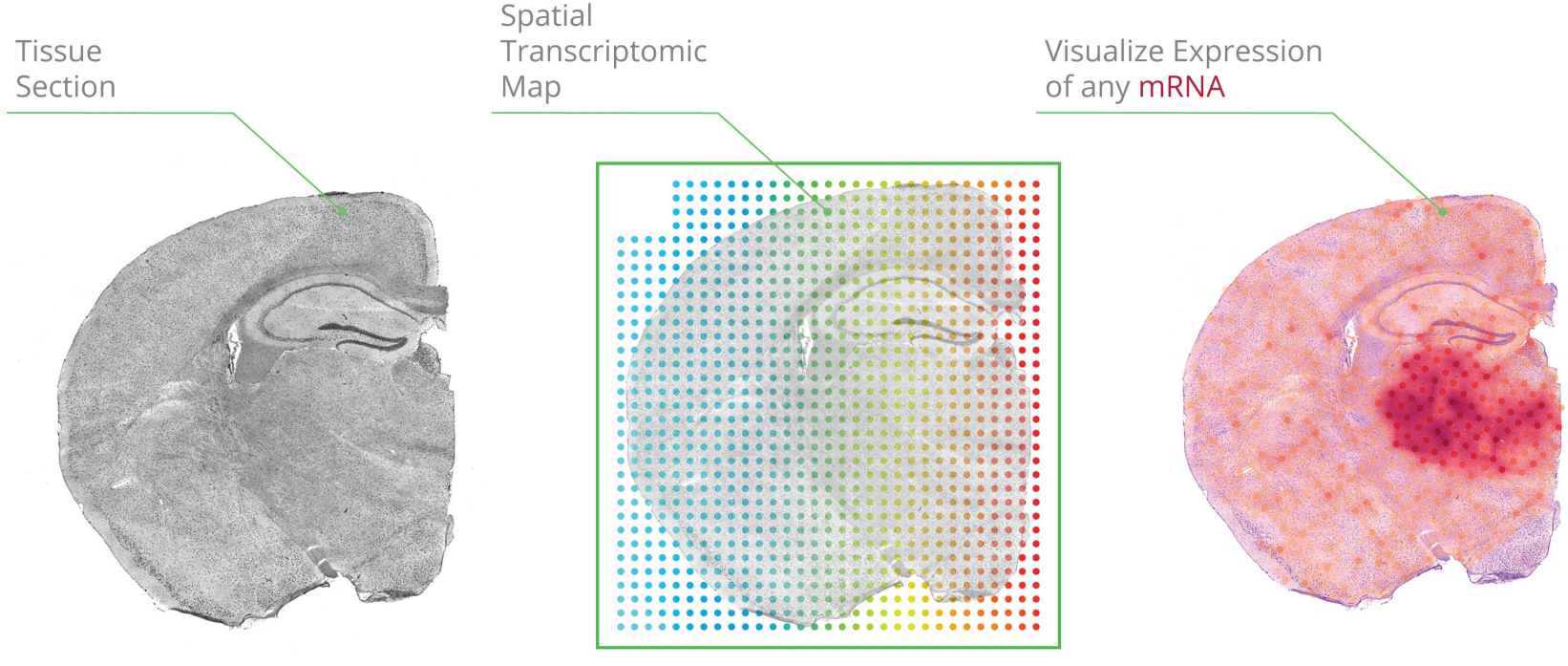


I did say *almost all* methods, except for spatial transcriptomics

I did say *almost all* methods, except for spatial transcriptomics

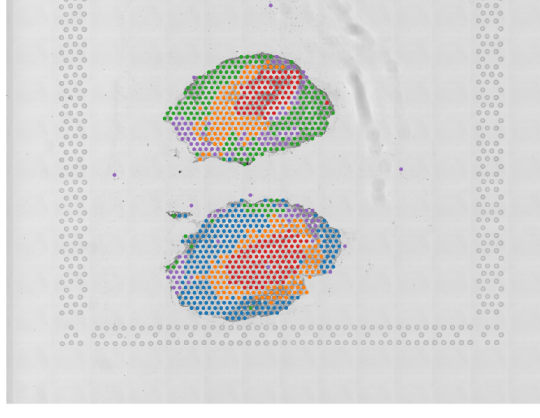


I did say *almost all* methods, except for spatial transcriptomics



This isn't single cell resolution though, only ~45μm.

Allows you to spatially orient your transcriptomic data, albeit at multi-single-cell resolution

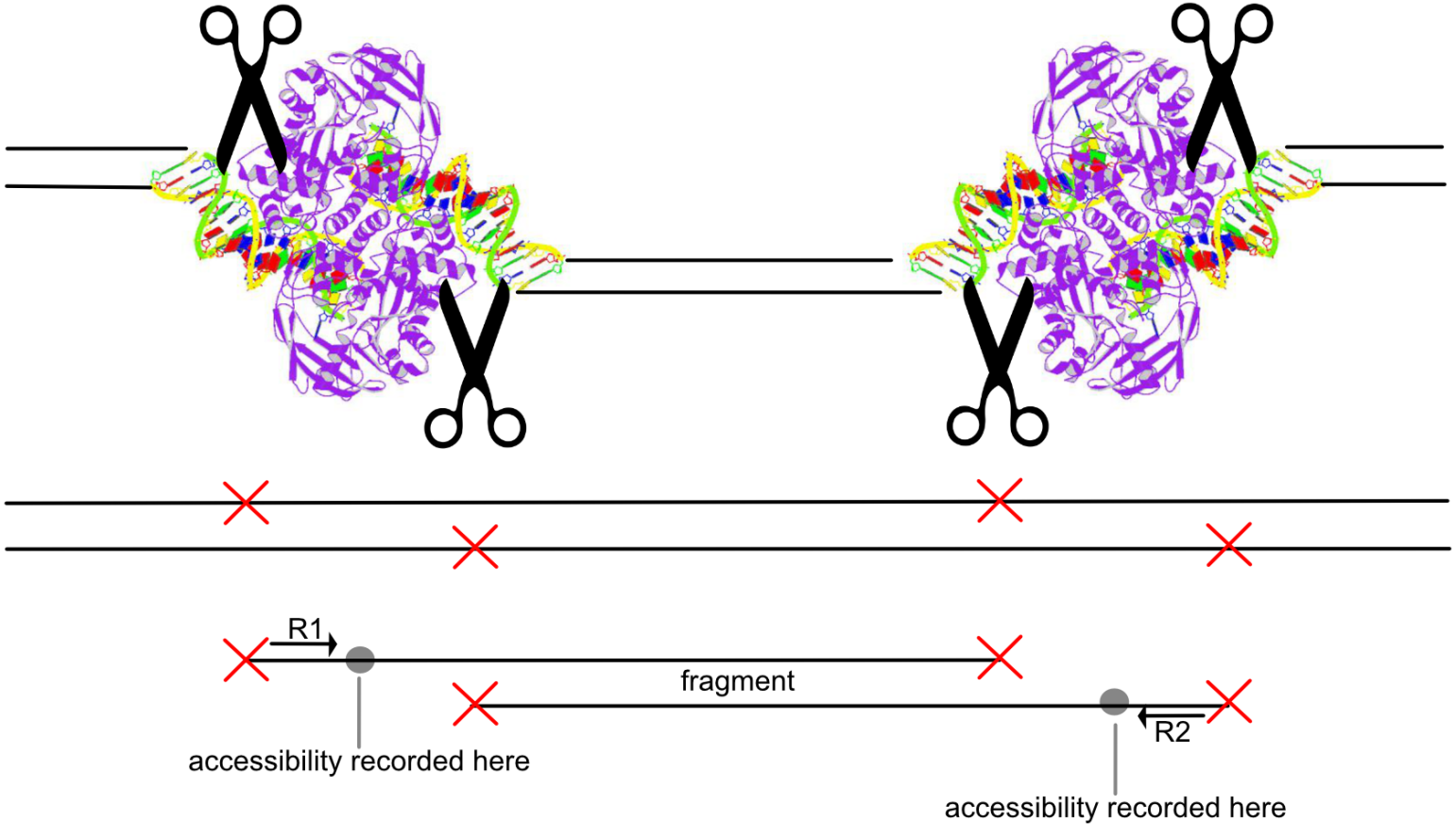


Other genomic readouts

- single nucleus ATAC-seq
- single cell DNA methylation
- single cell copy number analysis

snATAC-seq

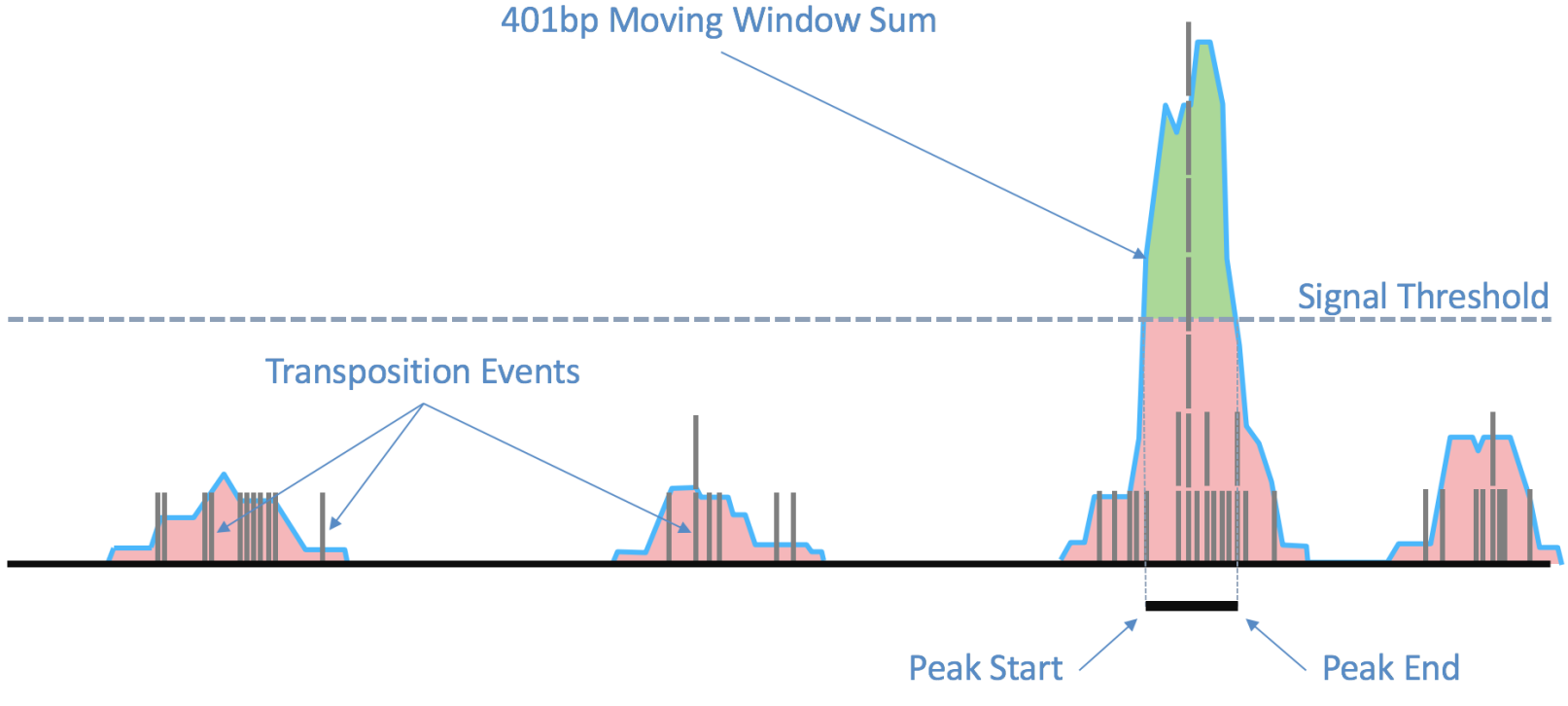
snATAC-seq



Transposase image accessed from the Protein Data Bank, https://rcsb.org/structure/1MUH

Peaks are called from pooling all cell-specific data together

Peaks are called from pooling all cell-specific data together



Challenges

- Peak calling depends on "bulk"ifying the data. What happens to peaks specific to rare populations?
- How do we confidently associate peaks with genes, promoters, and transcription factors?
- Read-out is a percentage, not a count, which leads to tecnical challenges, like proper normalization, dimensionality reduction, etc.

Analysis Walkthrough

Analysis resources

- Sean Davis single-cell-awesome (https://github.com/seandavi/awesome-single-cell)
- Martin Hemberg's Single Cell Book (https://hemberg-lab.github.io/scRNA.seq.course/index.html)
- Aaron Lun's Single Cell Workflow (http://bioconductor.org/help/workflows/simpleSingleCell/)
- Cold Spring Harbor Lab's Single Cell Bioinformatics Workshop (https://github.com/YeoLab/single-cell-bioinformatics/)
- Seurat tutorials (https://satijalab.org/seurat/get_started.html)
- Scanpy tutorials (https://scanpy.readthedocs.io/en/latest/tutorials.html)

Analysis package options

These are the major players

- Seurat (https://satijalab.org/seurat/) (R-based)
- Scanpy (https://scanpy.readthedocs.io/en/latest/) (Python-based)
- SingleCellExperiment (http://bioconductor.org/packages/release/bioc/html/SingleCellExperiment.html) (R-based)

There are dozens of packages that perform one task well and can usually be called from the frameworks above.

Open up Rstudio to use Seurat

Editor's note: These slides use Seurat version 3, released in 2019. Versus version 2, Seurat version 3 makes many of the function calls substantially simpler and the runtimes for many functions have been significantly improved.

Seurat installation instructions

The current way to install Seurat version 3 is simple but requires R 3.4+. Note that it might take a while to install.

Seurat installation instructions

The current way to install Seurat version 3 is simple but requires R 3.4+. Note that it might take a while to install.

In [ ]:
```
install.packages("Seurat")
```

Seurat installation instructions

The current way to install Seurat version 3 is simple but requires R 3.4+. Note that it might take a while to install.

In [ ]:
```
install.packages("Seurat")
```

If for some reason you need the bleeding-edge version, you can install the development branch

Seurat installation instructions

The current way to install Seurat version 3 is simple but requires R 3.4+. Note that it might take a while to install.

In [ ]:
```
install.packages("Seurat")
```

If for some reason you need the bleeding-edge version, you can install the development branch

In [ ]:
```
install.packages('devtools')
devtools::install_github(repo = 'satijalab/seurat', ref = 'develop')
```

Seurat installation instructions

The current way to install Seurat version 3 is simple but requires R 3.4+. Note that it might take a while to install.

In [ ]:
```
install.packages("Seurat")
```

If for some reason you need the bleeding-edge version, you can install the development branch

In [ ]:
```
install.packages('devtools')
devtools::install_github(repo = 'satijalab/seurat', ref = 'develop')
```

If either of these commands fail, you might find this post describing keeping R libraries clean (https://community.rstudio.com/t/help-regarding-package-installation-renviron-rprofile-r-libs-r-libs-site-and-r-libs-user-oh-my/13888/8) helpful. On MacOS, I have in my `~/.Renviron` the following line:

```
R_LIBS_USER=~/Library/R/%v/%p/library
```

If you add something like this to your environment (and make sure that the directory exists using the commands in the post above), restart R, you should be able to install `Seurat` pretty quickly.

If you need help with installing this package, please contact me (mailto:bill.flynn@jax.org).

Possible installation error

Your installation might fail due to lacking the package `multtest` for your version of R (which will then cause `mutoss > metap > Seurat` to fail to install). To remedy this, the best way is probably:

Possible installation error

Your installation might fail due to lacking the package `multtest` for your version of R (which will then cause `mutoss > metap > Seurat` to fail to install). To remedy this, the best way is probably:

In [ ]:
```
install.packages("BiocManager")
BiocManager::install("multtest")
```

Possible installation error

Your installation might fail due to lacking the package `multtest` for your version of R (which will then cause `mutoss > metap > Seurat` to fail to install). To remedy this, the best way is probably:

In [ ]:
```
install.packages("BiocManager")
BiocManager::install("multtest")
```

If that succeeds, then you should be good to reattempt

```
install.packages("Seurat")
```

Download and Load in data

Download and Load in data

In [2]:
```
library(curl)
curl_download("https://github.com/wflynny/MEDS6498-2020/archive/gh-pages.zip", "si
nglecell-lecture.zip")
unzip("singlecell-lecture.zip")
setwd("MEDS6498-2020-gh-pages")
unzip("dataset-inclass.zip")
```

In [ ]:
```
#setwd("/home/ubuntu/singlecell")
library(Seurat)
```

In [ ]:
```
#setwd("/home/ubuntu/singlecell")
library(Seurat)
```

In [5]:
```
list.files("dataset-inclass/filtered_gene_bc_matrices/hg19")
```

'barcodes.tsv' · 'genes.tsv' · 'matrix.mtx'

In [ ]:
```
#setwd("/home/ubuntu/singlecell")
library(Seurat)
```

In [5]:
```
list.files("dataset-inclass/filtered_gene_bc_matrices/hg19")
```

'barcodes.tsv' · 'genes.tsv' · 'matrix.mtx'

In [6]:
```
pbmc.data <- Read10X(data.dir = "dataset-inclass/filtered_gene_bc_matrices/hg19/")
```

Create a Seurat Object

Create a Seurat Object

In [64]:
```
pbmc <- CreateSeuratObject(counts = pbmc.data,
                           min.cells = 3,
                           min.features = 200,
                           project = "10X_PBMC")
```

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

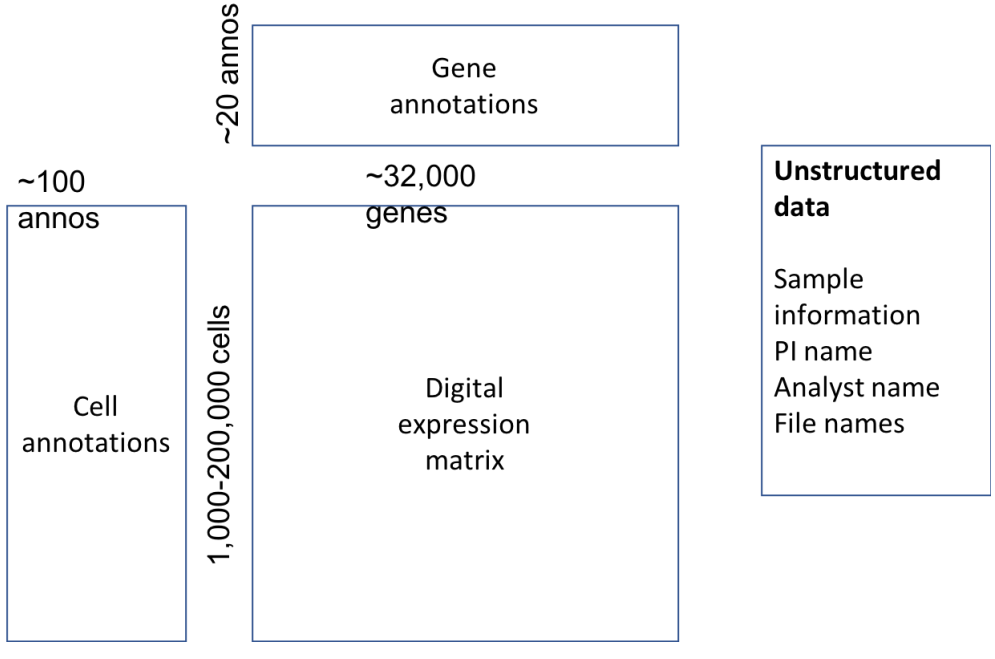
Create a Seurat Object

In [64]:
```
pbmc <- CreateSeuratObject(counts = pbmc.data,
                           min.cells = 3,
                           min.features = 200,
                           project = "10X_PBMC")
```

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

- Preliminarily filtering out cells with fewer than 200 genes
- Preliminarily filtering out genes expressed in fewer than 3 cells

Common organizational structure of a single cell dataset



How else can we filter our data?

Let's examine the amount of mtRNA expressed in each cell.

How else can we filter our data?

Let's examine the amount of mtRNA expressed in each cell.

In [65]:
```
# First find mtRNA genes then compute the fraction
mito.features <- grep(pattern = "^MT-", x = rownames(x = pbmc), value = TRUE)
# Caution! This only works for Human Data.  Mouse data will need a different Rege
x!
```

How else can we filter our data?

Let's examine the amount of mtRNA expressed in each cell.

In [65]:
```
# First find mtRNA genes then compute the fraction
mito.features <- grep(pattern = "^MT-", x = rownames(x = pbmc), value = TRUE)
# Caution! This only works for Human Data.  Mouse data will need a different Rege
x!
```

In [66]:
```
# Compute the percentage of UMIs belonging to mtRNAs
percent.mito <- Matrix::colSums(GetAssayData(pbmc, slot = "counts")[mito.features,
]) /
    Matrix::colSums(GetAssayData(pbmc, slot = "counts")) * 100
```

How else can we filter our data?

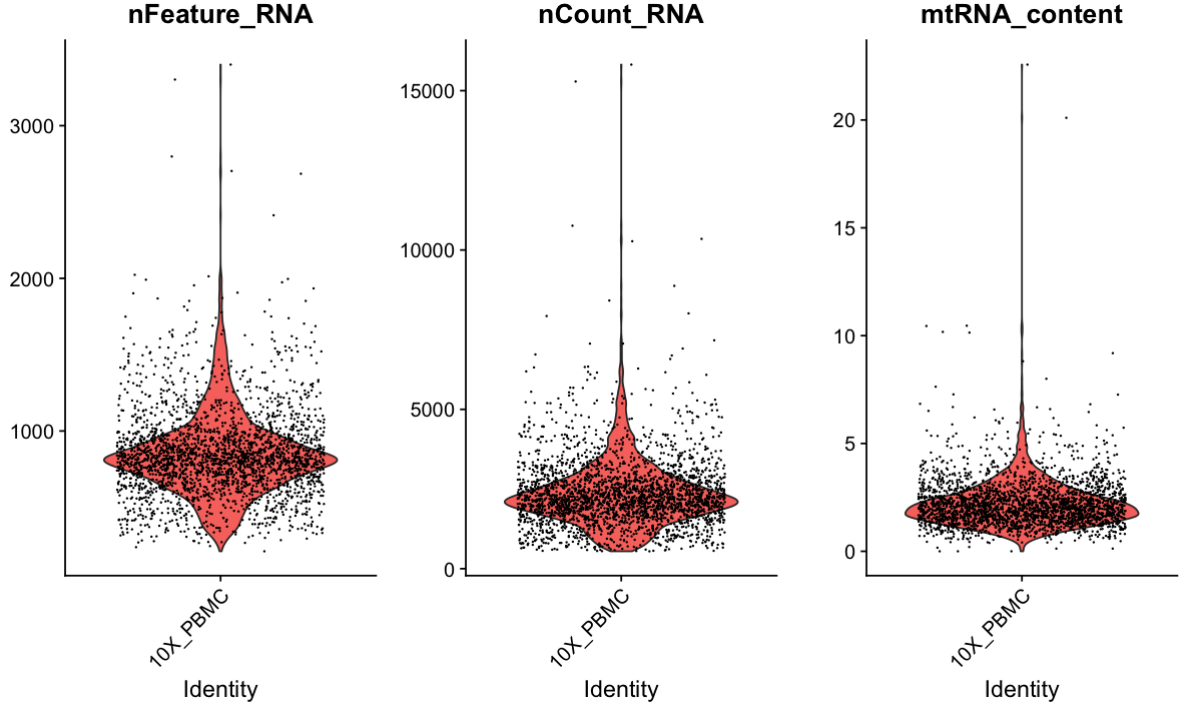
Let's examine the amount of mtRNA expressed in each cell.

In [65]:
```
# First find mtRNA genes then compute the fraction
mito.features <- grep(pattern = "^MT-", x = rownames(x = pbmc), value = TRUE)
# Caution! This only works for Human Data.  Mouse data will need a different Rege
x!
```

In [66]:
```
# Compute the percentage of UMIs belonging to mtRNAs
percent.mito <- Matrix::colSums(GetAssayData(pbmc, slot = "counts")[mito.features,
]) /
    Matrix::colSums(GetAssayData(pbmc, slot = "counts")) * 100
```

In [67]:
```
# Add this as a column to our cells metadata
pbmc[["mtRNA_content"]] <- percent.mito
```

In [69]:
```
VlnPlot(
    object = pbmc,
    features = c("nFeature_RNA", "nCount_RNA", "mtRNA_content"),
    ncol = 3,
    pt.size = 1e-2
)
```



Relationships between metadata

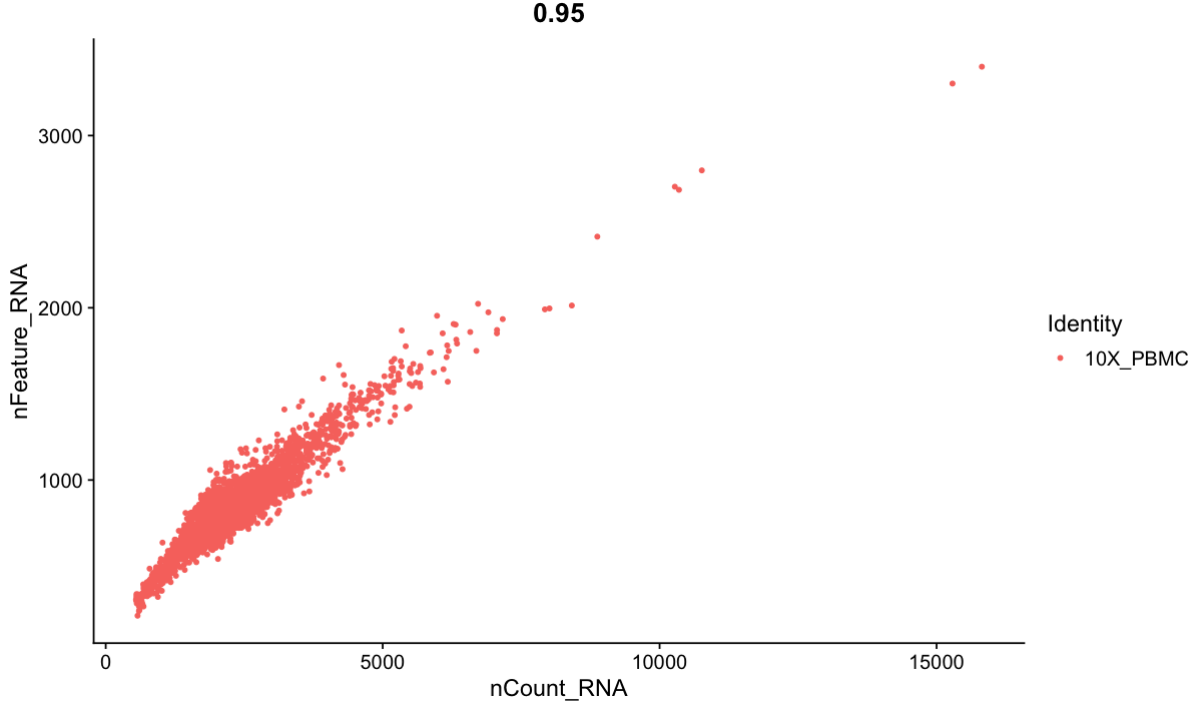
Relationships between metadata

UMIs (`nCount_RNA`) and number of genes (`nFeature_RNA`) are very correlated.

Relationships between metadata

UMIs (`nCount_RNA`) and number of genes (`nFeature_RNA`) are very correlated.

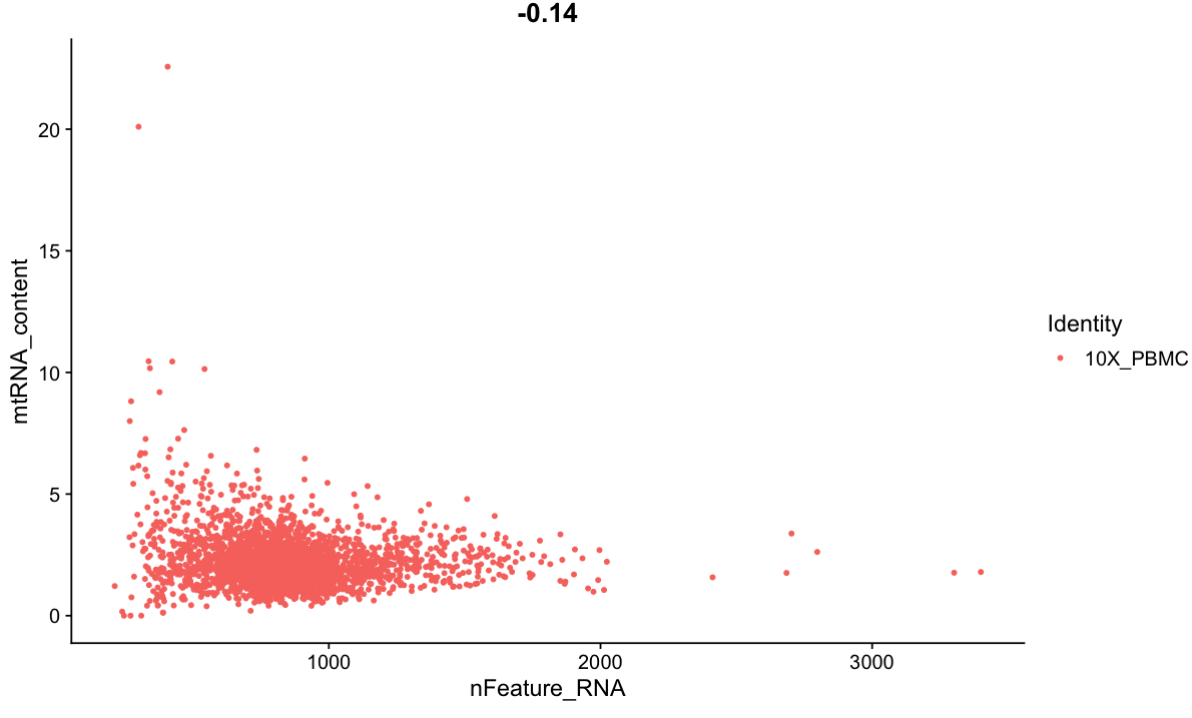
In [70]:
```
FeatureScatter(object = pbmc, feature1 = "nCount_RNA", feature2 = "nFeature_RNA")
```



If mtRNA content is an indicator of low cell quality, then we'd expect that cells with high mtRNA content (`mtRNA_content`) also express few genes. Is this the case?

If mtRNA content is an indicator of low cell quality, then we'd expect that cells with high mtRNA content (`mtRNA_content`) also express few genes. Is this the case?

In [71]:
```
FeatureScatter(object = pbmc, feature1 = "nFeature_RNA", feature2 = "mtRNA_conten
t")
```



Some preliminary filtering

Based on the plots above, let's filter mtRNA content to be `<10%` and the number of genes to be between `200` and `2500`.

Some preliminary filtering

Based on the plots above, let's filter mtRNA content to be `<10%` and the number of genes to be between `200` and `2500`.

In [72]:
```
dim(pbmc)
```

13714 · 2700

Some preliminary filtering

Based on the plots above, let's filter mtRNA content to be `<10%` and the number of genes to be between `200` and `2500`.

In [72]:
```
dim(pbmc)
```

13714 · 2700

In [73]:
```
pbmc <- subset(
    x = pbmc,
    subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mito < 5
)
```

Some preliminary filtering

Based on the plots above, let's filter mtRNA content to be `<10%` and the number of genes to be between `200` and `2500`.

In [72]:
```
dim(pbmc)
```

13714 · 2700

In [73]:
```
pbmc <- subset(
    x = pbmc,
    subset = nFeature_RNA > 200 & nFeature_RNA < 2500 & percent.mito < 5
)
```

In [74]:
```
dim(pbmc)
```

13714 · 2638

After filtering, we normalize

- This library size normalization uses the scaling factor `10,000` and will log-transform the data for us.

After filtering, we normalize

- This library size normalization uses the scaling factor `10,000` and will log-transform the data for us.

In [75]:
```
pbmc.10k <- NormalizeData(
    object = pbmc,
    normalization.method = "LogNormalize"
)
```

After filtering, we normalize

- This library size normalization uses the scaling factor `10,000` and will log-transform the data for us.

In [75]:
```
pbmc.10k <- NormalizeData(
    object = pbmc,
    normalization.method = "LogNormalize"
)
```

Alternatively, we could use the median library size as our scaling factor.

After filtering, we normalize

- This library size normalization uses the scaling factor `10,000` and will log-transform the data for us.

In [75]:
```
pbmc.10k <- NormalizeData(
    object = pbmc,
    normalization.method = "LogNormalize"
)
```

Alternatively, we could use the median library size as our scaling factor.

In [76]:
```
median.lib.size <- median(Matrix::colSums(GetAssayData(pbmc, slot = "counts")))
```

After filtering, we normalize

- This library size normalization uses the scaling factor `10,000` and will log-transform the data for us.

In [75]:
```
pbmc.10k <- NormalizeData(
    object = pbmc,
    normalization.method = "LogNormalize"
)
```

Alternatively, we could use the median library size as our scaling factor.

In [76]:
```
median.lib.size <- median(Matrix::colSums(GetAssayData(pbmc, slot = "counts")))
```

In [77]:
```
pbmc <- NormalizeData(
    object = pbmc,
    normalization.method = "LogNormalize",
    scale.factor = median.lib.size
)
```

Finding genes that are highly varying

There are two ways to select "variable" genes using dispersion. The simplest way is to select the *N* mostly highly dispersed genes. In that case, we can use `selection.method = "dispersion"`:

Finding genes that are highly varying

There are two ways to select "variable" genes using dispersion. The simplest way is to select the N mostly highly dispersed genes. In that case, we can use `selection.method = "dispersion"`:

In [78]:
```
pbmc <- FindVariableFeatures(
    object = pbmc,
    selection.method = "vst",
    nfeatures = 2000
)
```

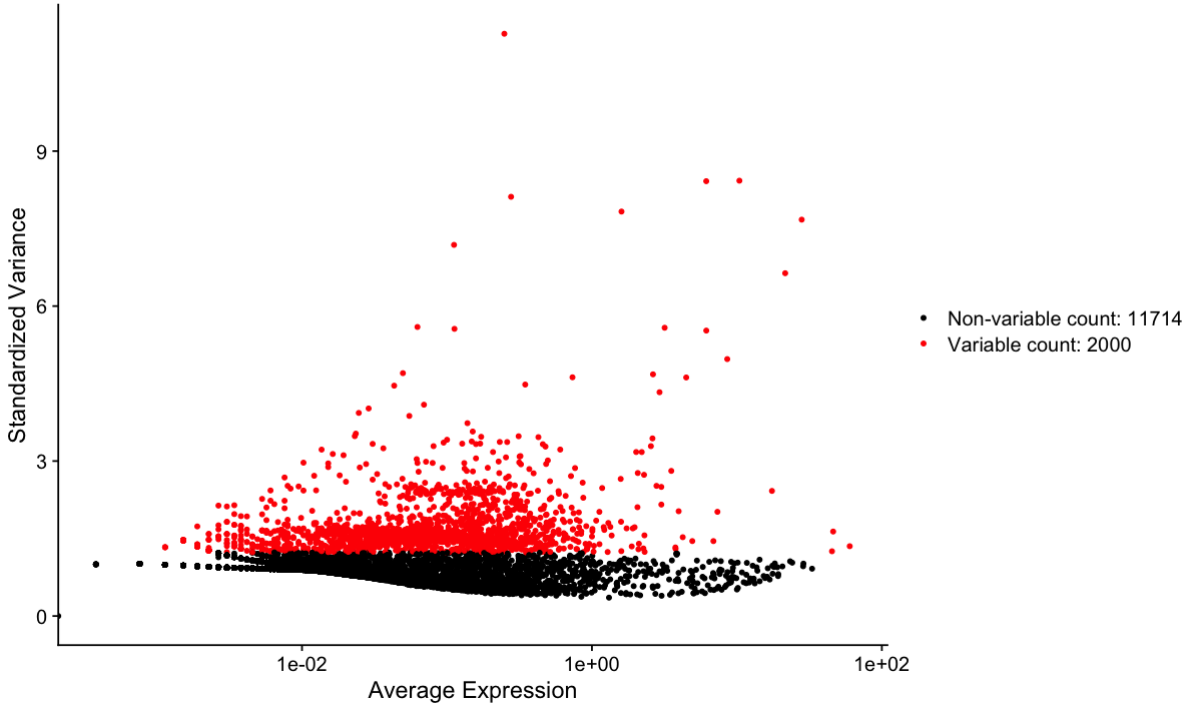
Finding genes that are highly varying

There are two ways to select "variable" genes using dispersion. The simplest way is to select the N mostly highly dispersed genes. In that case, we can use `selection.method = "dispersion"`:

In [78]:
```
pbmc <- FindVariableFeatures(
    object = pbmc,
    selection.method = "vst",
    nfeatures = 2000
)
```

In [79]:
```
VariableFeaturePlot(pbmc)
```

```
Warning message:
“Transformation introduced infinite values in continuous x-axis”
```



Alternatively, can either plot each gene by its mean and variance, then select a region (defined by a rectangle in mean-variance space) in that plot where signature genes live. To do this, we can specify `selection.method = "mean.var.plot"` and the cutoffs that define the rectangle:

- `mean.cutoff` = (minimum, maximum) mean expression to consider
- `dispersion.cutoff` = (minimum, maximum) dispersion to consider

Alternatively, can either plot each gene by its mean and variance, then select a region (defined by a rectangle in mean-variance space) in that plot where signature genes live. To do this, we can specify `selection.method = "mean.var.plot"` and the cutoffs that define the rectangle:

- `mean.cutoff` = (minimum, maximum) mean expression to consider
- `dispersion.cutoff` = (minimum, maximum) dispersion to consider

In [80]:
```
pbmc.mv <- FindVariableFeatures(
    object = pbmc,
    selection.method = "mean.var.plot",
    mean.cutoff = c(0.0125, 3),
    dispersion.cutoff = c(0.5, Inf),
)
```

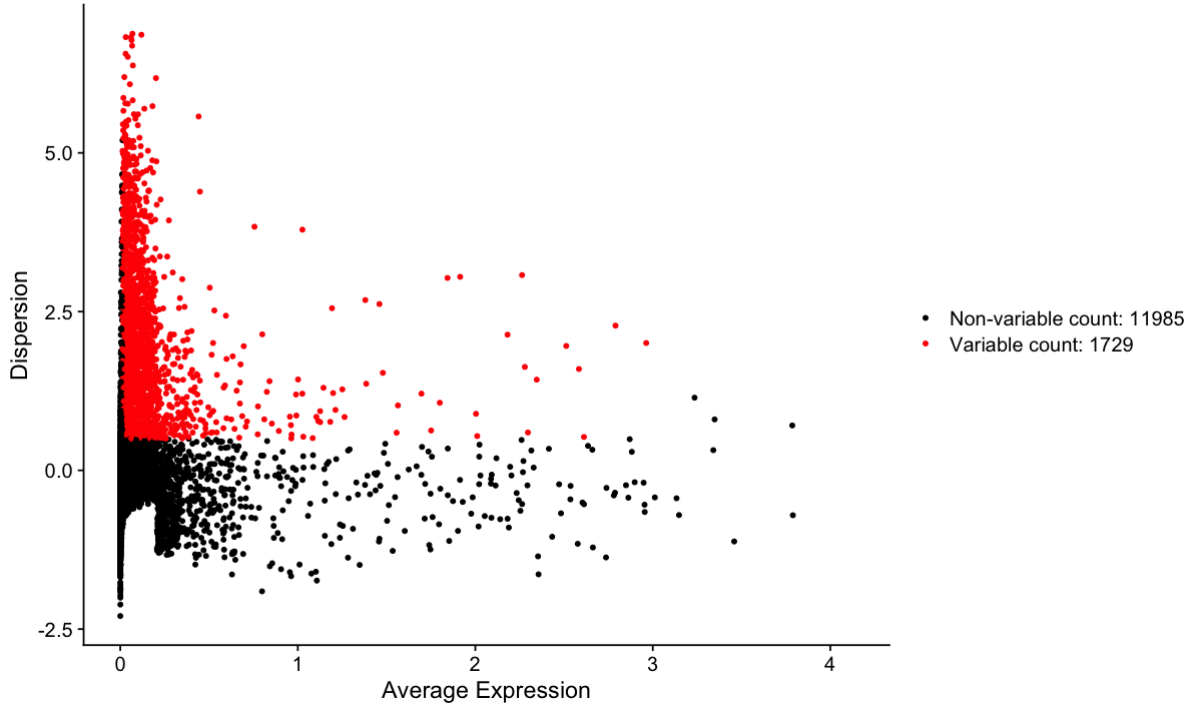
Alternatively, can either plot each gene by its mean and variance, then select a region (defined by a rectangle in mean-variance space) in that plot where signature genes live. To do this, we can specify `selection.method = "mean.var.plot"` and the cutoffs that define the rectangle:

- `mean.cutoff` = (minimum, maximum) mean expression to consider
- `dispersion.cutoff` = (minimum, maximum) dispersion to consider

In [80]:
```
pbmc.mv <- FindVariableFeatures(
    object = pbmc,
    selection.method = "mean.var.plot",
    mean.cutoff = c(0.0125, 3),
    dispersion.cutoff = c(0.5, Inf),
)
```

In [81]:
```
VariableFeaturePlot(pbmc.mv)
```

```
Warning message:
“Removed 1 rows containing missing values (geom_point).”
```



For either method:

- Because the expression has been log transformed, the mean expression of each gene is computed by reversing the log (through `exp`), computing the mean, then log transforming again.
- Dispersion is computed by taking the log of the ratio of variance to the mean for each gene.

Scale the data and run PCA

Scale the data and run PCA

In addition to scaling the data, we will also "regress out" any bias related to the percentage of mtRNA or UMI count per cell.

WARNING: Scaling is pretty quick, but regression takes a few minutes.

Scale the data and run PCA

In addition to scaling the data, we will also "regress out" any bias related to the percentage of mtRNA or UMI count per cell.

WARNING: Scaling is pretty quick, but regression takes a few minutes.

In [82]:
```
pbmc <- ScaleData(
    object = pbmc,
    features = rownames(x = pbmc),
    #vars.to.regress = "percent.mt"
)
```

```
Centering and scaling data matrix
```

Scale the data and run PCA

In addition to scaling the data, we will also "regress out" any bias related to the percentage of mtRNA or UMI count per cell.

WARNING: Scaling is pretty quick, but regression takes a few minutes.

In [82]:
```
pbmc <- ScaleData(
    object = pbmc,
    features = rownames(x = pbmc),
    #vars.to.regress = "percent.mt"
)
```

```
Centering and scaling data matrix
```

In [83]:
```
pbmc <- RunPCA(
    object = pbmc,
    features = VariableFeatures(object = pbmc),
    verbose = FALSE
)
```

What genes are important for our first few PCs?

What genes are important for our first few PCs?

In [84]:
```
print(
    x = pbmc[['pca']],
    dims = 1:5,
    nfeatures = 5,
    projected = FALSE
)
```

```
PC_ 1
Positive:  CST3, TYROBP, FTL, LYZ, AIF1
Negative:  MALAT1, LTB, IL32, IL7R, B2M
PC_ 2
Positive:  CD79A, MS4A1, HLA-DQA1, HLA-DQB1, HLA-DRA
Negative:  NKG7, PRF1, GZMB, CST7, GZMA
PC_ 3
Positive:  CD74, HLA-DPA1, HLA-DPB1, HLA-DQA1, CD79B
Negative:  PPBP, PF4, SDPR, NRGN, GNG11
PC_ 4
Positive:  HLA-DQA1, CD79B, CD74, CD79A, HLA-DQB1
Negative:  VIM, IL7R, IL32, GIMAP7, S100A6
PC_ 5
Positive:  MS4A7, CKB, FCGR3A, RP11-290F20.3, RHOC
Negative:  S100A8, S100A9, LGALS2, CD14, MS4A6A
```

What genes are important for our first few PCs?

In [84]:
```
print(
    x = pbmc[['pca']],
    dims = 1:5,
    nfeatures = 5,
    projected = FALSE
)
```

```
PC_ 1
Positive:  CST3, TYROBP, FTL, LYZ, AIF1
Negative:  MALAT1, LTB, IL32, IL7R, B2M
PC_ 2
Positive:  CD79A, MS4A1, HLA-DQA1, HLA-DQB1, HLA-DRA
Negative:  NKG7, PRF1, GZMB, CST7, GZMA
PC_ 3
Positive:  CD74, HLA-DPA1, HLA-DPB1, HLA-DQA1, CD79B
Negative:  PPBP, PF4, SDPR, NRGN, GNG11
PC_ 4
Positive:  HLA-DQA1, CD79B, CD74, CD79A, HLA-DQB1
Negative:  VIM, IL7R, IL32, GIMAP7, S100A6
PC_ 5
Positive:  MS4A7, CKB, FCGR3A, RP11-290F20.3, RHOC
Negative:  S100A8, S100A9, LGALS2, CD14, MS4A6A
```

Just from this, it looks like PC1 separates dendritic cells from T cells, and PC2 separates Natural Killer cells from B cells.

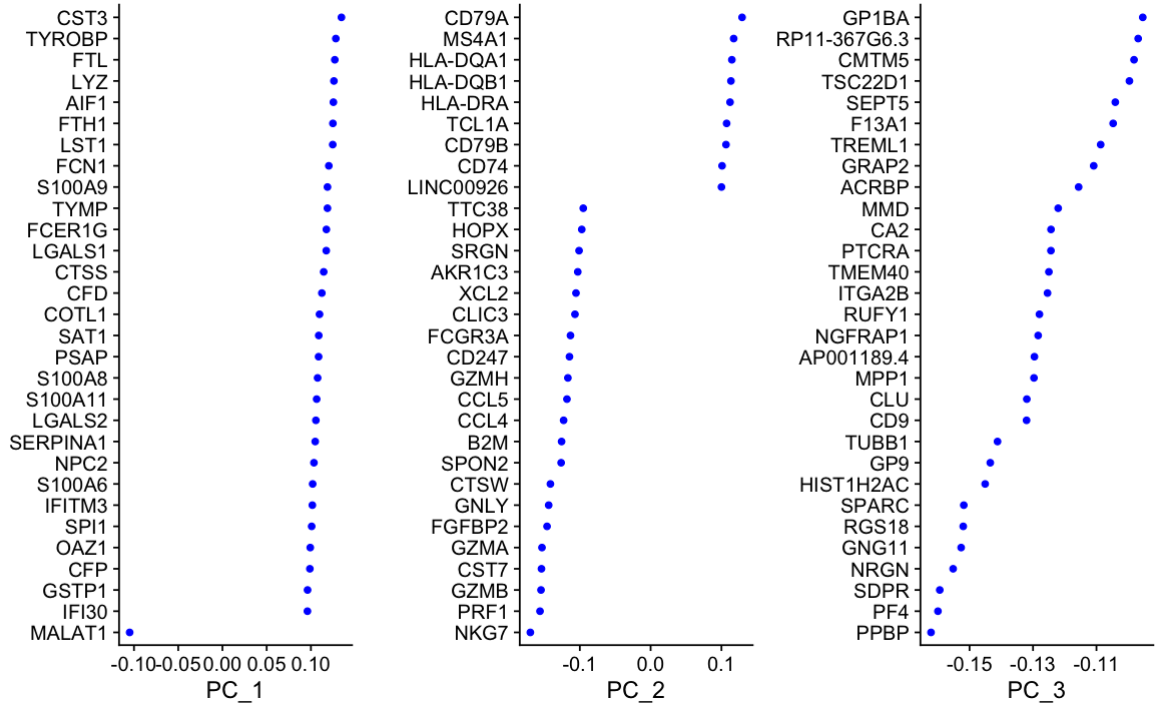
How do we visualize our PCA results?

How do particular genes contribute to each PC?

How do we visualize our PCA results?

How do particular genes contribute to each PC?

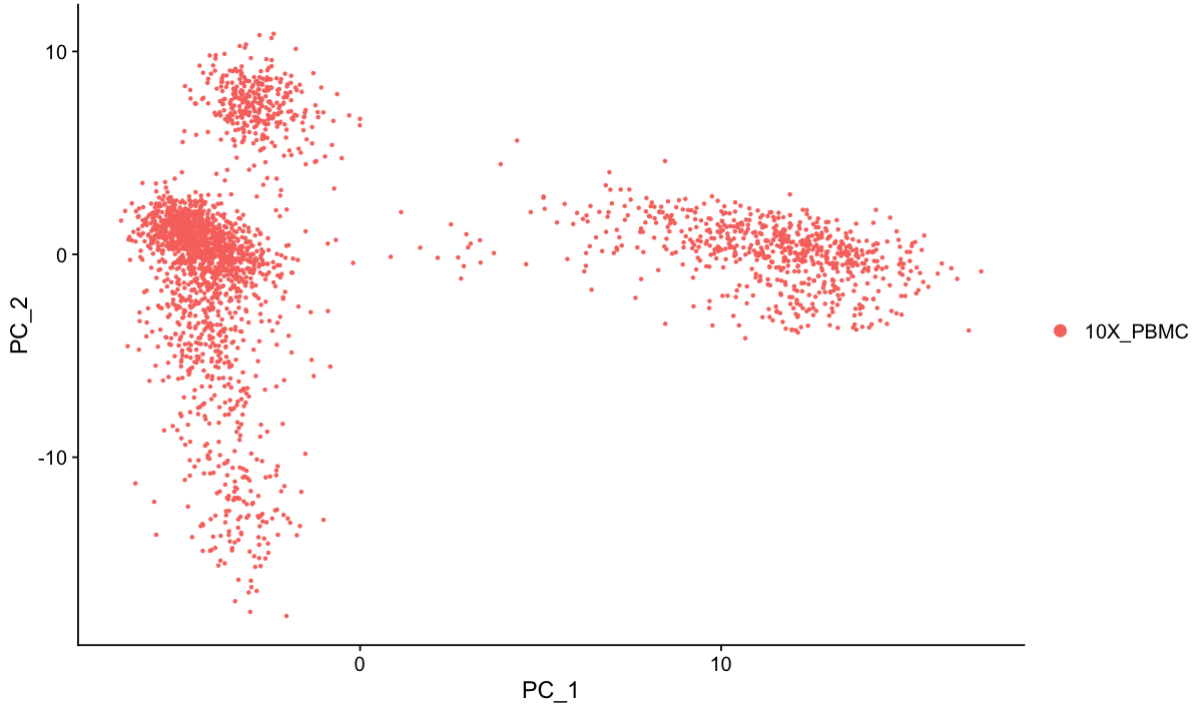
In [85]:
```
VizDimLoadings(object = pbmc, dims = 1:3, ncol = 3)
```



Or we can just plot PC1 vs PC2:

Or we can just plot PC1 vs PC2:

In [86]:
```
DimPlot(object = pbmc, )
```



How do we know how many PCs are important?

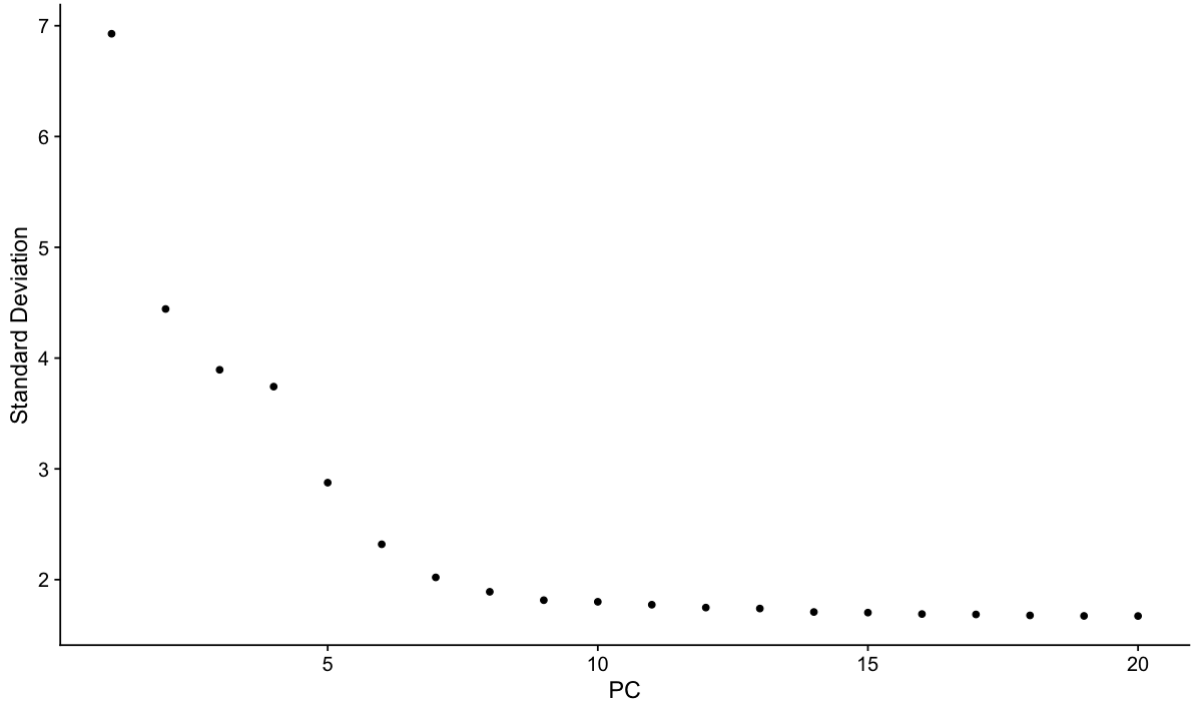
How do we know how many PCs are important?

If we plot the standard deviations of each PC, we can look for a cutoff after which additional PCs don't contribute much.

How do we know how many PCs are important?

If we plot the standard deviations of each PC, we can look for a cutoff after which additional PCs don't contribute much.

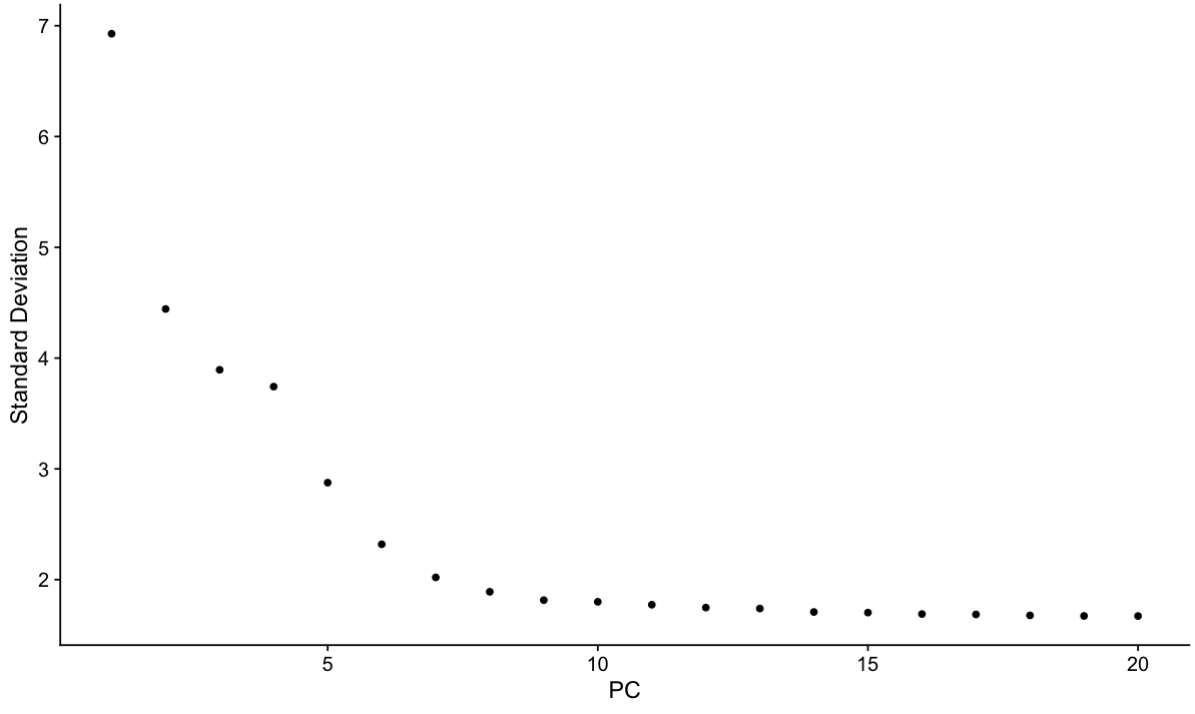
In [87]:
```
ElbowPlot(object = pbmc)
```



How do we know how many PCs are important?

If we plot the standard deviations of each PC, we can look for a cutoff after which additional PCs don't contribute much.

In [87]:
```
ElbowPlot(object = pbmc)
```



Here it looks like the first 4 are really important, and after about 10 PCs contributions start to die off.

More systematically, we can compute the "jackstraws"

This randomly permutes genes and computes PCA on the permuted genes. When compared to the real PCs, we can get a sense for significant our PCs are.

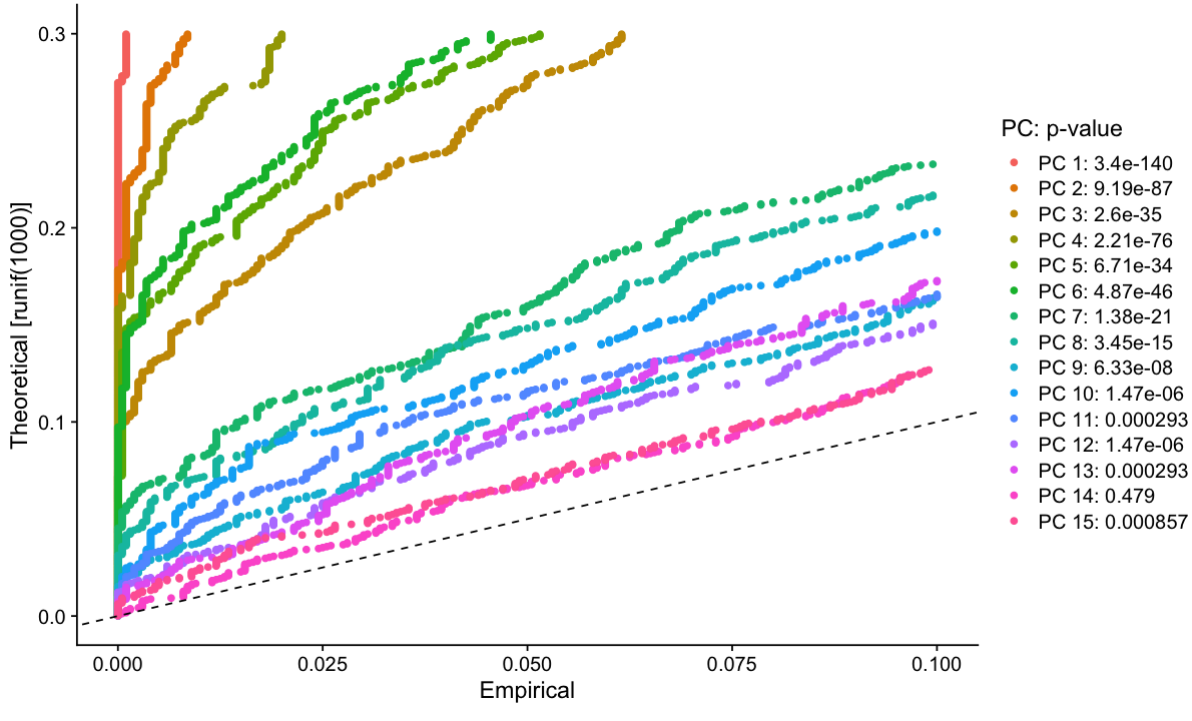
- Below we can see that the first 10-11 PCs (solid-ish lines) have enriched signal over the randomized threshold (dotted lines).
- ***Careful***, this takes a few minutes to compute.

In [88]:
```
# Compute the jackstraws
pbmc <- JackStraw(object = pbmc, num.replicate = 100)
pbmc <- ScoreJackStraw(object = pbmc, dims = 1:15)
```

In [88]:
```
# Compute the jackstraws
pbmc <- JackStraw(object = pbmc, num.replicate = 100)
pbmc <- ScoreJackStraw(object = pbmc, dims = 1:15)
```

In [89]:
```
JackStrawPlot(object = pbmc, dims = 1:15)
```

```
Warning message:
"Removed 23291 rows containing missing values (geom_point)."
```



Now that we've compute PCA, we can build a nearest neighbor graph and find clusters

Two things to note here:

1. we're using the PCs to compute the graph, but we could use any reduction type.
2. `FindClusters` finds "communities" given a certain `resolution` parameter. Higher resolutions lead to more, smaller clusters and lower resolutions lead to fewer but larger clusters. Typical values are `0.32 - 1`.

Now that we've compute PCA, we can build a nearest neighbor graph and find clusters

Two things to note here:

1. we're using the PCs to compute the graph, but we could use any reduction type.
2. `FindClusters` finds "communities" given a certain `resolution` parameter. Higher resolutions lead to more, smaller clusters and lower resolutions lead to fewer but larger clusters. Typical values are `0.32 - 1`.

In [ ]:
```
pbmc <- FindNeighbors(object = pbmc, dims = 1:10)
pbmc <- FindClusters(object = pbmc, resolution = 0.6)
```

Let's visualize these clusters in a low dimensional embedding with t-SNE

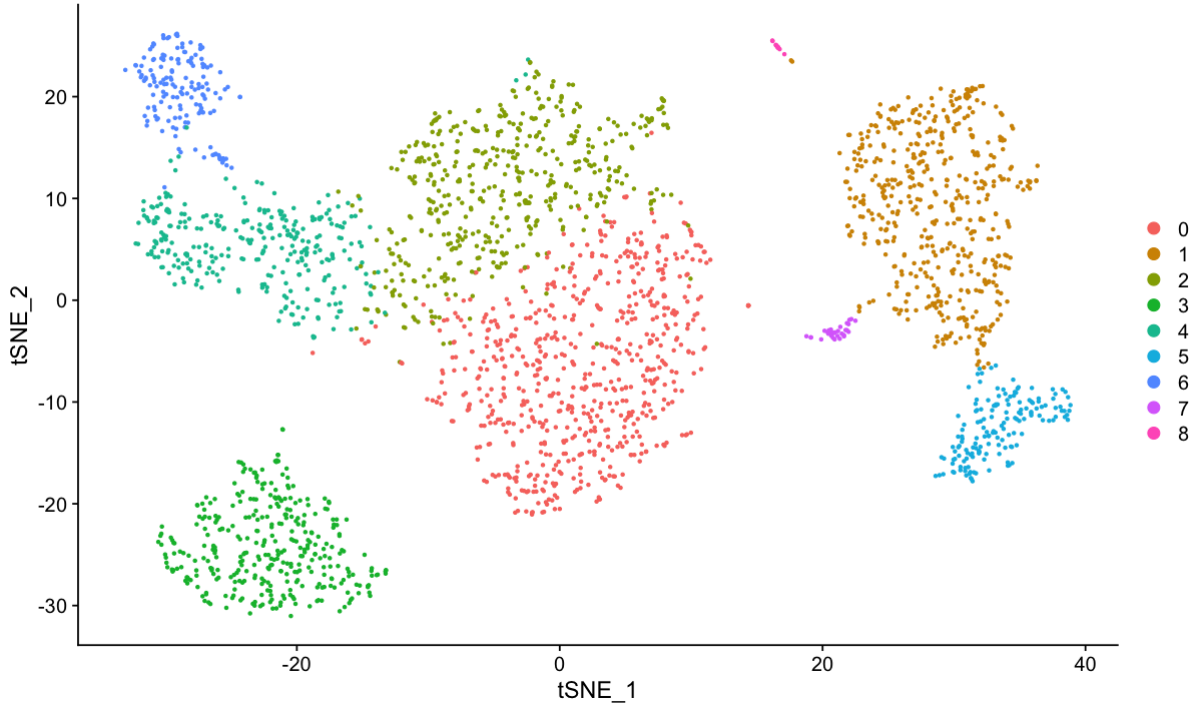
Let's visualize these clusters in a low dimensional embedding with t-SNE

In [91]:
```
pbmc <- RunTSNE(object = pbmc, dims = 1:10)
```

Let's visualize these clusters in a low dimensional embedding with t-SNE

In [91]:
```
pbmc <- RunTSNE(object = pbmc, dims = 1:10)
```

In [92]:
```
DimPlot(object = pbmc, reduction = "tsne")
```

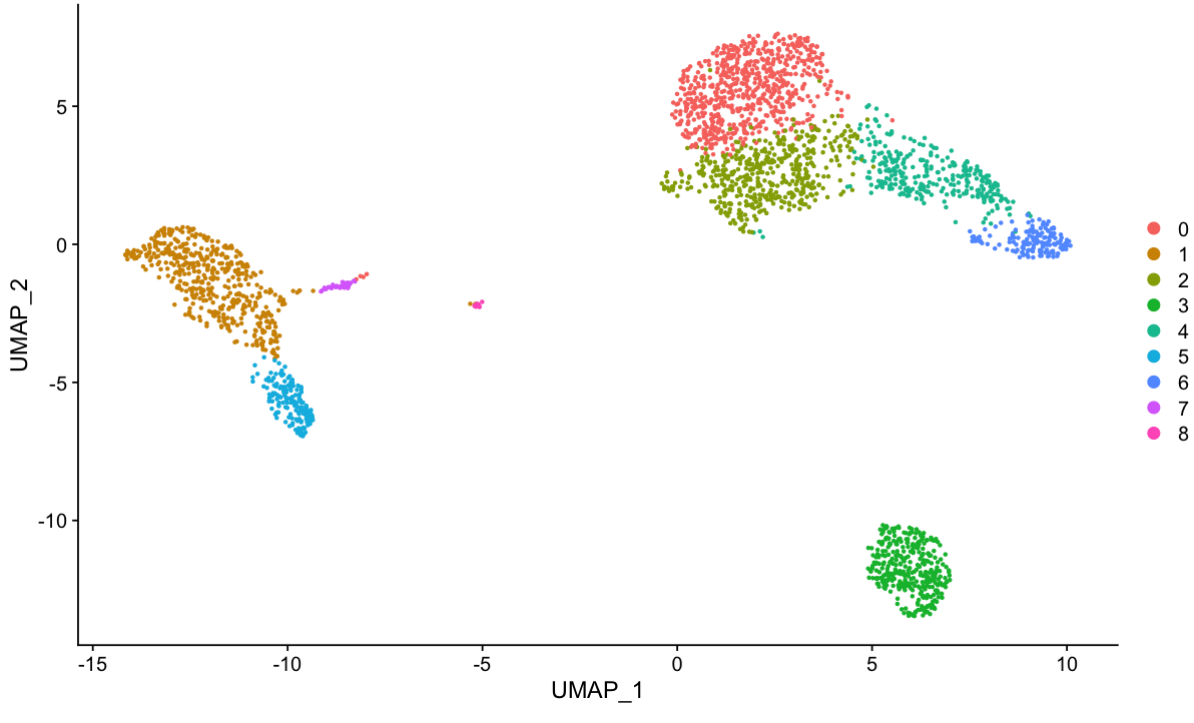


Alternatively, let's use UMAP

Alternatively, let's use UMAP

In [ ]:
```
pbmc <- RunUMAP(object = pbmc, dims=1:10)
```

In [94]:
```
DimPlot(object = pbmc, reduction = "umap")
```



Finding markers

Finding markers

This will take a few minutes.

- `only.pos = TRUE` selects only genes that are enriched in each cluster
- `min.pct = 0.25` limits the testing to genes expressed in at least 25% of cells of each cluster
- `logfc.threshold = 0.25` limits the testing to genes with log fold change (logfc) greater than 0.25.

Finding markers

This will take a few minutes.

- `only.pos = TRUE` selects only genes that are enriched in each cluster
- `min.pct = 0.25` limits the testing to genes expressed in at least 25% of cells of each cluster
- `logfc.threshold = 0.25` limits the testing to genes with log fold change (logfc) greater than 0.25.

In [ ]:
```
pbmc.markers <- FindAllMarkers(
    object = pbmc,
    only.pos = TRUE,
    min.pct = 0.25,
    logfc.threshold = 0.25
)
```

In [96]:
```
suppressPackageStartupMessages(library(dplyr))
# this is some tidyverse magic to select the top 2 genes from each cluster
pbmc.markers %>% group_by(cluster) %>% top_n(n = 2, wt = avg_logFC)
```

A grouped_df: 18 × 7

p_val	avg_logFC	pct.1	pct.2	p_val_adj	cluster	gene
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>	<chr>
1.255929e-118	0.5678495	0.930	0.589	1.722381e-114	0	LDHB
2.291434e-117	0.5333700	0.999	0.975	3.142472e-113	0	RPS3A
0.000000e+00	3.0828090	0.996	0.214	0.000000e+00	1	S100A9
4.102371e-270	2.8133709	1.000	0.516	5.625992e-266	1	LYZ
3.612157e-99	0.7100994	0.950	0.464	4.953712e-95	2	IL32
5.719918e-87	0.7674060	0.979	0.643	7.844296e-83	2	LTB
0.000000e+00	1.7494454	0.936	0.041	0.000000e+00	3	CD79A
5.255736e-185	1.8833636	1.000	0.821	7.207717e-181	3	CD74
6.034271e-213	1.7833536	0.987	0.230	8.275399e-209	4	CCL5
1.128786e-176	1.3456444	0.944	0.215	1.548017e-172	4	NKG7
1.772478e-123	1.7578355	1.000	0.316	2.430777e-119	5	LST1
1.544290e-117	1.5330436	1.000	0.317	2.117839e-113	5	FCER1G
1.807352e-197	2.9447437	0.968	0.130	2.478602e-193	6	GNLY
4.379521e-141	2.4330123	1.000	0.254	6.006075e-137	6	NKG7
1.665286e-21	1.7645240	1.000	0.513	2.283773e-17	7	HLA-DPB1

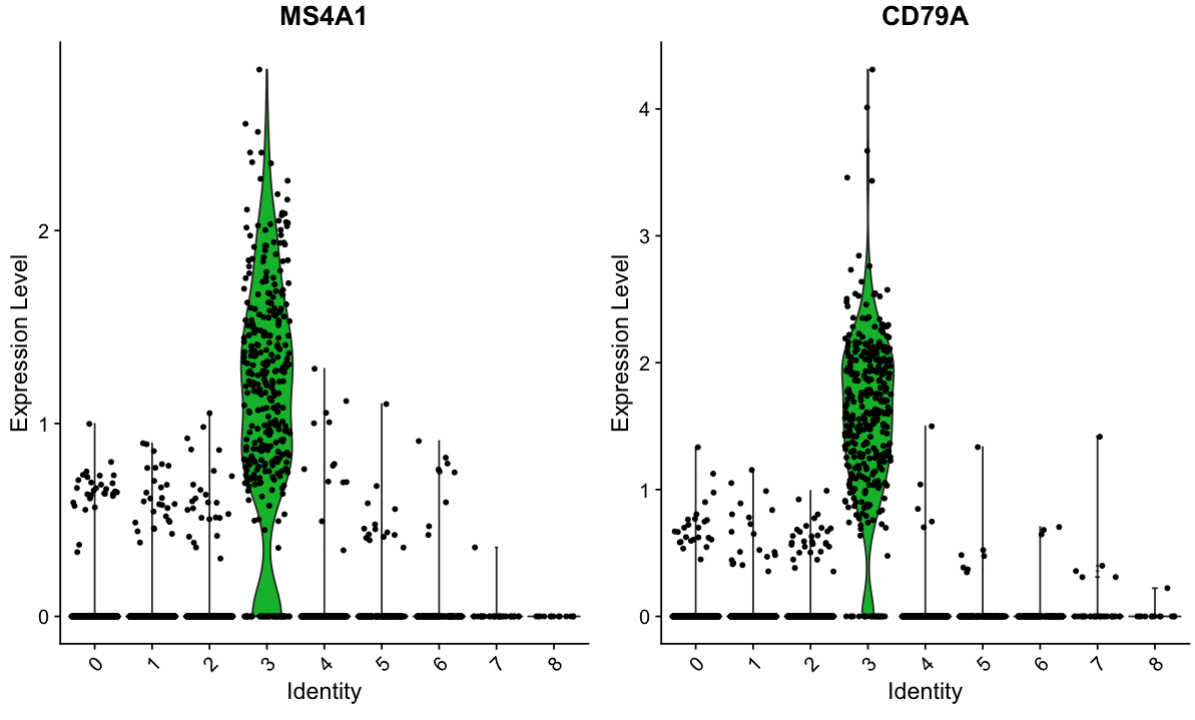
p_val	avg_logFC	pct.1	pct.2	p_val_adj	cluster	gene
8.093693e-20	1.6463954	1.000	0.555	1.109969e-15	7	HLA-DRA
<dbl>	**<dbl>**	**<dbl>**	**<dbl>**	**<dbl>**	**<fct>**	**<chr>**

9.250621e-186	3.6207802	1.000	0.011	1.268630e-181	8	PF4
1.915360e-102	4.5986313	1.000	0.024	2.626724e-98	8	PPBP

Many ways to plot our top genes

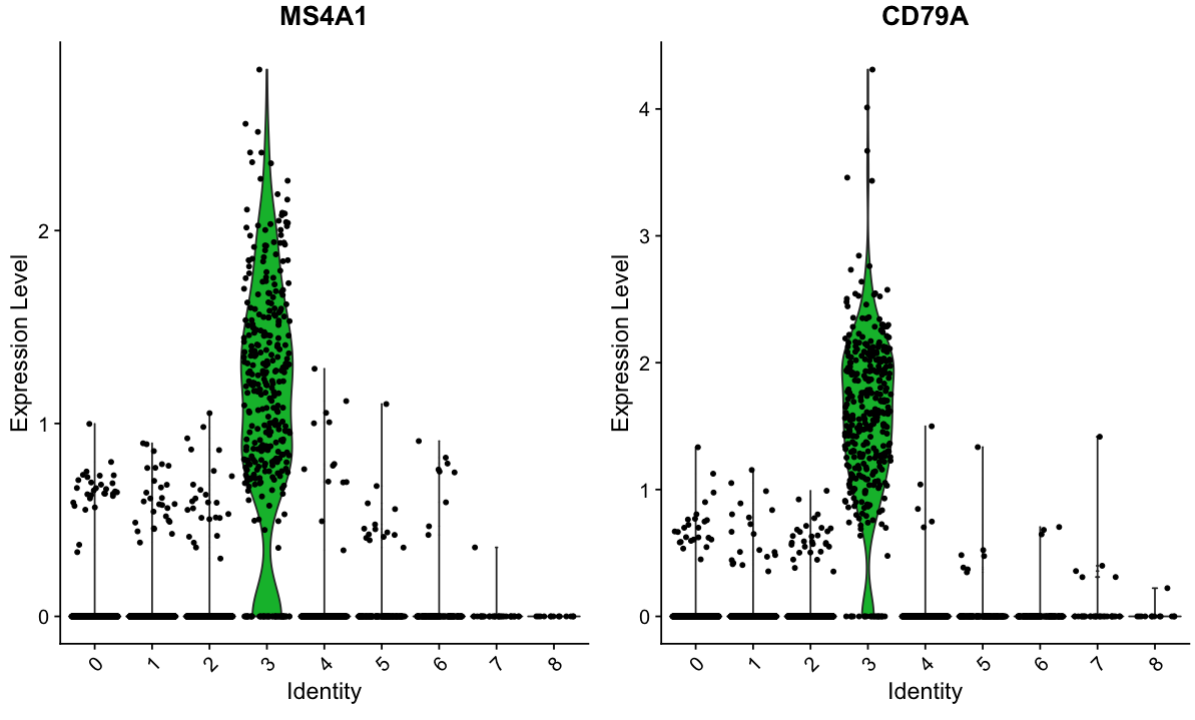
Many ways to plot our top genes

In [97]:
```
VlnPlot(object = pbmc, features = c("MS4A1", "CD79A"))
```



Many ways to plot our top genes

In [97]:
```
VlnPlot(object = pbmc, features = c("MS4A1", "CD79A"))
```

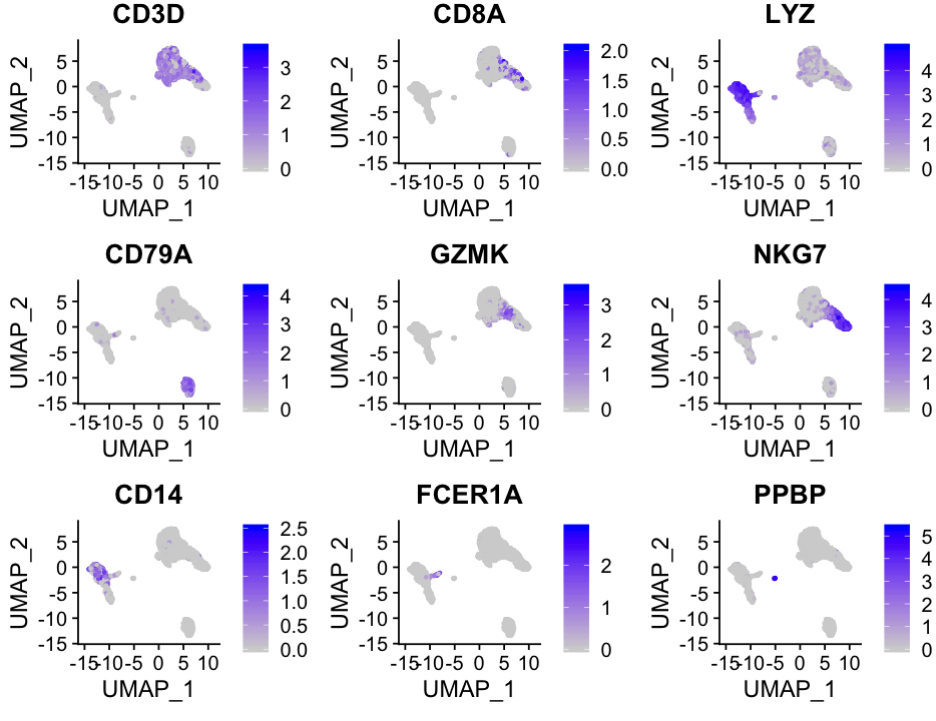


Those are our B cells!

In [99]:

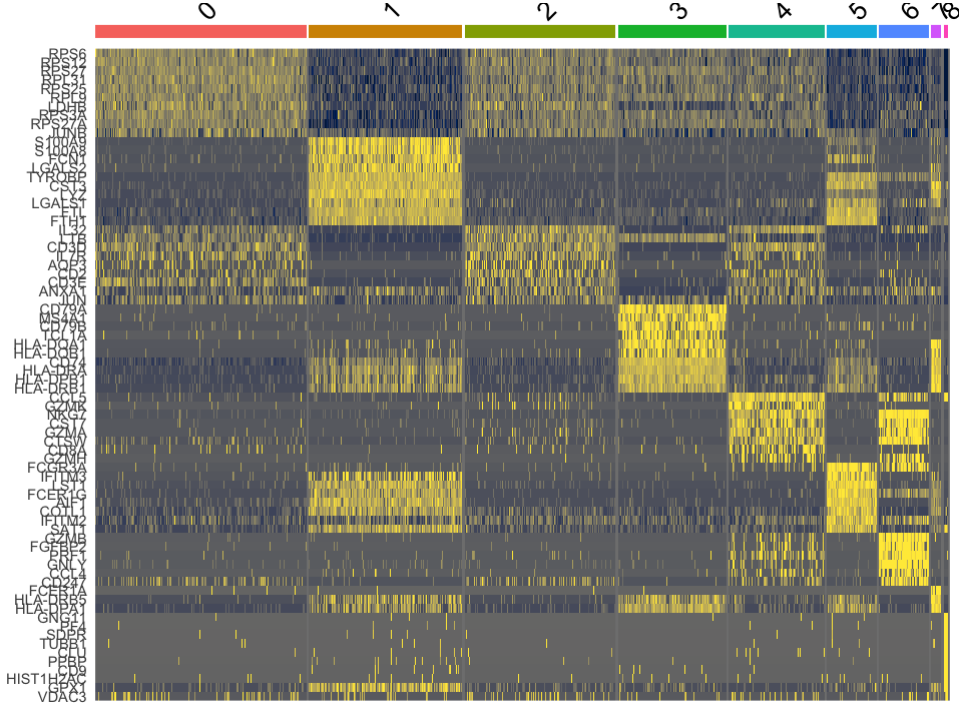
```
top.genes <- c(
  "CD3D", # CD4 T cells
  "CD8A", # CD8 T cells
  "LYZ", # Monocytes
  "CD79A", # B cells
  "GZMK", # Gamma/Delta T cells
  "NKG7", # NK cells
  "CD14", # CD14+ Monocyte
  "FCER1A", # Classical DC
  "PPBP" # Platlets
)
FeaturePlot(object = pbmc, features = top.genes)
```

```
The following functions and any applicable methods accept the dots: plot_grid
```



In [100]:
```
# this is some tidyverse magic to select the top 10 markers from each cluster
pbmc.markers %>% group_by(cluster) %>% top_n(n = 10, wt = avg_logFC) -> top10
DoHeatmap(object = pbmc, features = top10$gene) + NoLegend() +
    ggplot2::scale_fill_gradientn(colors = viridisLite::cividis(256))
```

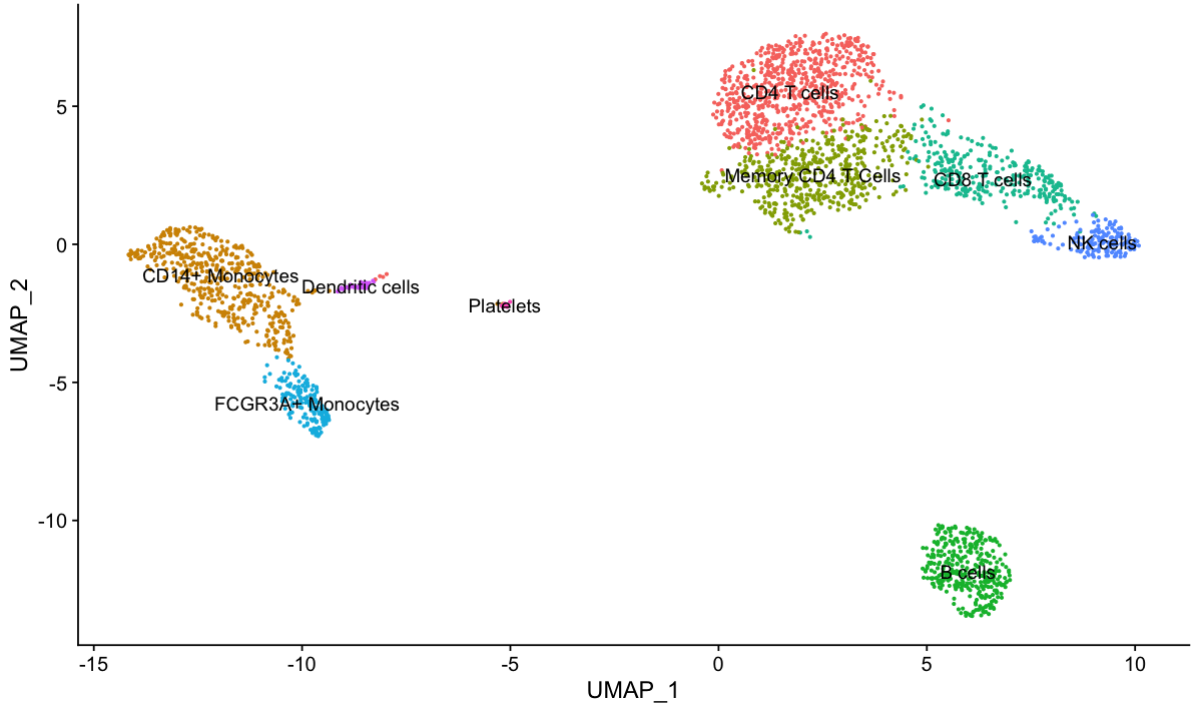
```
Scale for 'fill' is already present. Adding another scale for 'fill', which
will replace the existing scale.
```



Let's make our publication ready plot with cell populations labeled

After visualizing our clusters and attempting to identify them through our biological knowledge, asking colleagues, or looking up genes in a database (http://bio-bigdata.hrbmu.edu.cn/CellMarker/) (or Google!), we can make these labels part of the Seurat object.

In [102]:
```
new.cluster.ids <- c(
    "CD4 T cells", "CD14+ Monocytes", "Memory CD4 T Cells", "B cells", "CD8 T cell
s",
    "FCGR3A+ Monocytes", "NK cells", "Dendritic cells", "Platelets"
)
names(x = new.cluster.ids) <- levels(x = pbmc)
pbmc <- RenameIdents(object = pbmc, new.cluster.ids)
DimPlot(object = pbmc, reduction = 'umap', label = TRUE, pt.size = 0.5) + NoLegend
()
```



Assignment

Experiment with this type of analysis on your own!

Papers for Journal Club

In groups of 2, please read one of these papers and present any main advances or why the described work might be important in the field for ***~10-15*** minutes.

Questions, comments

Email me: bill.flynn@jax.org (mailto:bill.flynn@jax.org)