

BIOINFORMATICS PIPELINE FOR SCRNA-SEQ: FROM RAW DATA TO INSIGHTS

Jerry Li Ph.D.

Research Support Analyst

Digital Research Services, IST

jiarui.li@ualberta.ca

May 8, 2025



**UNIVERSITY
OF ALBERTA**

Outline

1. Day 1: Introduction to scRNA-seq

- The principle
- The history
- Wet Lab pipeline
- Input Data preprocessing

2. Day 2: scRNA-seq Analysis

- QC
- Normalized expression
- Clustering
- Marker genes and cell-type annotation

Objectives

- Understand the principles and workflow of single-cell RNA sequencing (scRNA-seq)
- Learn the importance of quality control in scRNA-seq and the rationale behind it
- Gain hands-on experience running one of the most widely used analysis pipelines

Note

- The slides can be found in Github:
<https://github.com/ualberta-rcg/scRNA-seq>

Login by ssh

- Our workshop cluster

```
ssh user80@spring2025-uofa.c3.ca
```

- The Alliance cluster

```
ssh username@cedar.alliancecan.ca
```

- Your own system

Prep Sample FASTQ

```
cp -r projects/def-sponsor00/scRNA-seq/Sample_FASTQ/ .
```

Install Cellranger if you are using the Alliance or your own system

<https://www.10xgenomics.com/support/software/cell-ranger/downloads#download-links>

```
wget -O cellranger-9.0.1.tar.gz "https://cf.10xgenomics.com/releases/cell-exp/cellranger-9.0.1.tar.gz?Expires=1746585246&Key-Pair-Id=APKAI7S6A5RYOXBWRPDA&Signature=LT7~WaW0pcQeHJx3HB7Wq51i-JPiwN0ee3NBE0vfdKjtxn0DZAe1-RA-2jIj1aBgeDFlictRWD-hJcqCUgaqiyxfzxdS9Pn-~MctYr9oYswolGHZFJR-a9E0pg7RPxHlP50KPhjWPIUV8z~Z3P2REfgaRTHT2RhOa1Q3Vl01bdj5CQF63tK~qRNgfXpDI1r-La2tGCtJP0qlagYkmMOHJcR-fVM1xFs-BIu7lkY4aW1I1BRch2MjruGvddvtJwU7S-kFx-jYkNawVo00xYePrpqUMotrJGeJ-JF-~hacQ4whBy3Hl6re4U7RDn7vYi92Eir~XXHr0h9rfnfns~GTJw__"
```

```
tar -xf cellranger-9.0.1.tar.gz
```

Download Materials if you are using the Alliance or your own system

<https://drive.google.com/drive/folders/18vXOcOPEPUGW85fM6ZbCxKQJQuHnanQD?usp=sharing>

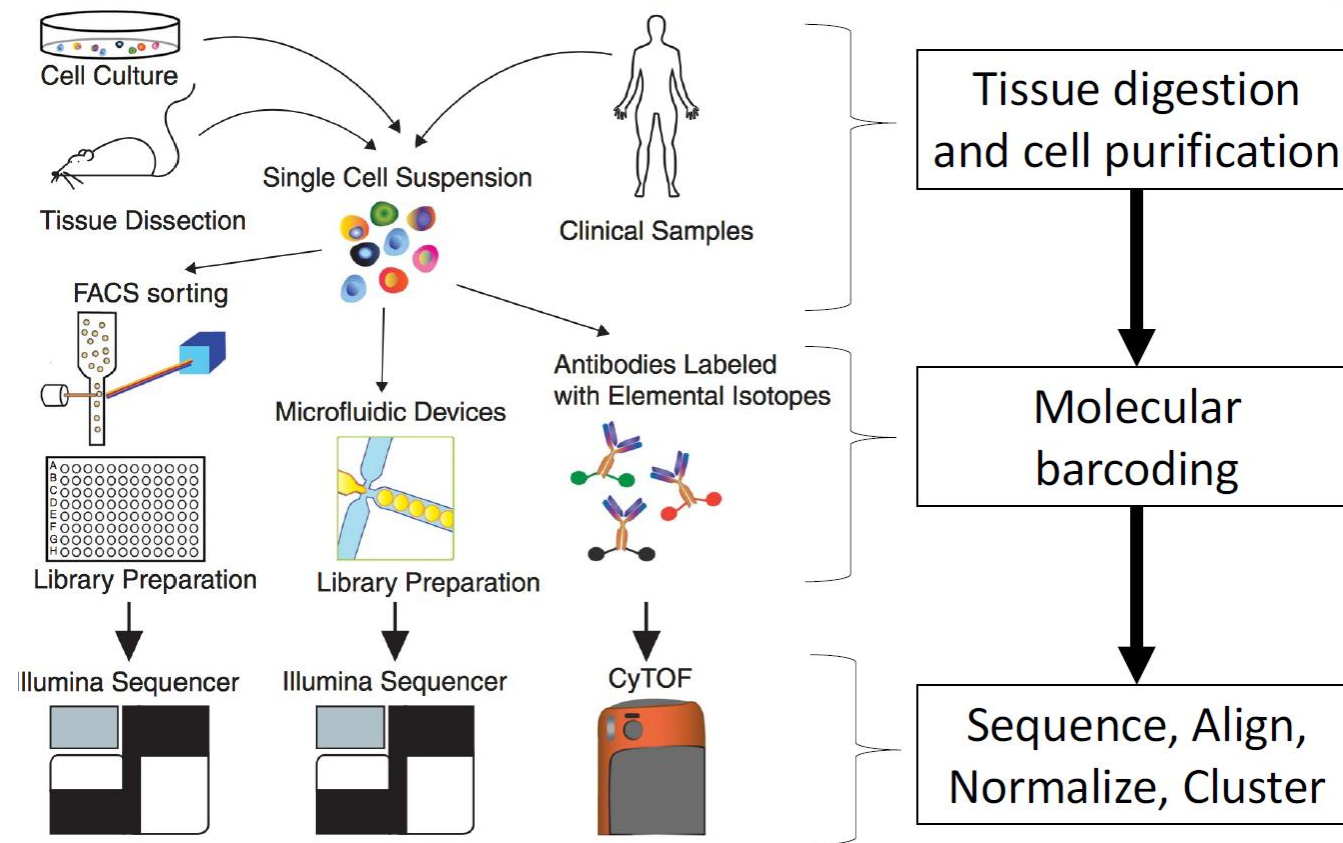
Install Apptainer if you are using your own system

<https://apptainer.org/docs/admin/main/installation.html>

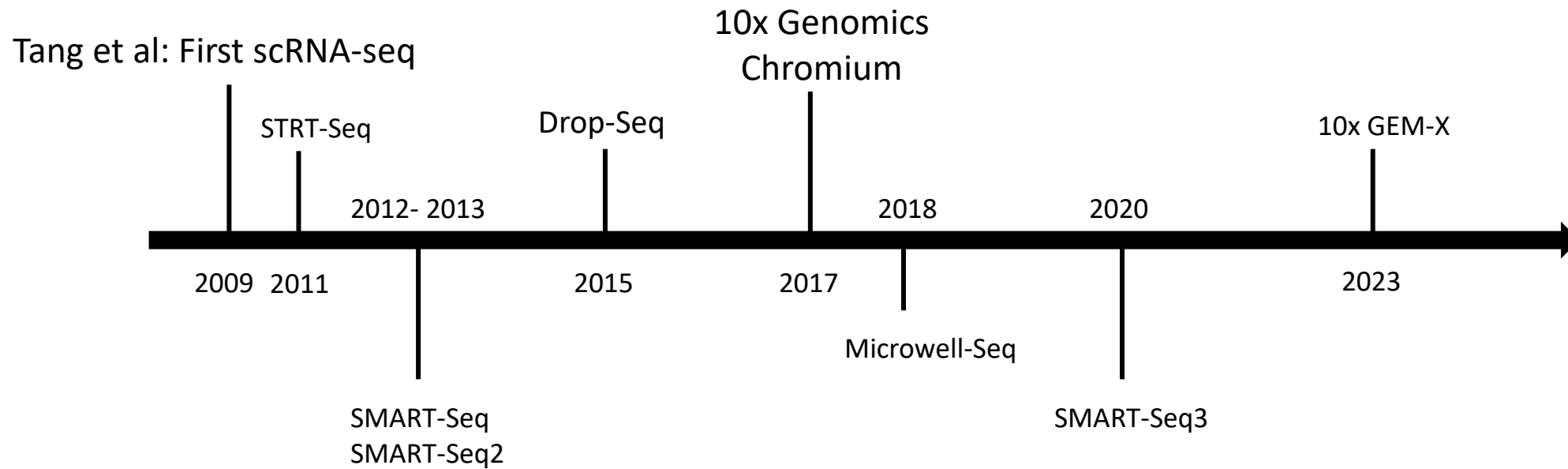
Day 1. Introduction to scRNA-seq



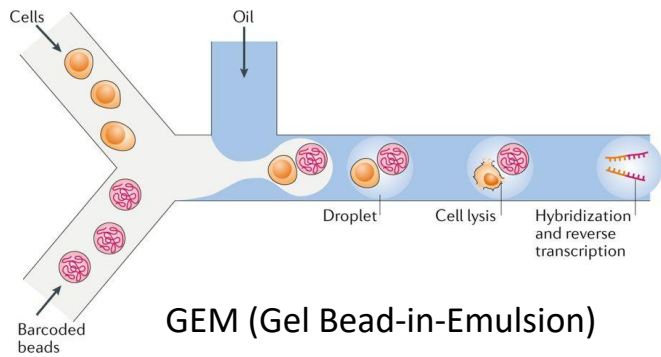
scRNA-seq Wet Lab Pipeline



The Milestone of scRNA-seq



Single Cell Isolation



Droplet-based
S. Steven Potter, Nature Reviews Nephrology
volume 14, pages479–492 (2018)

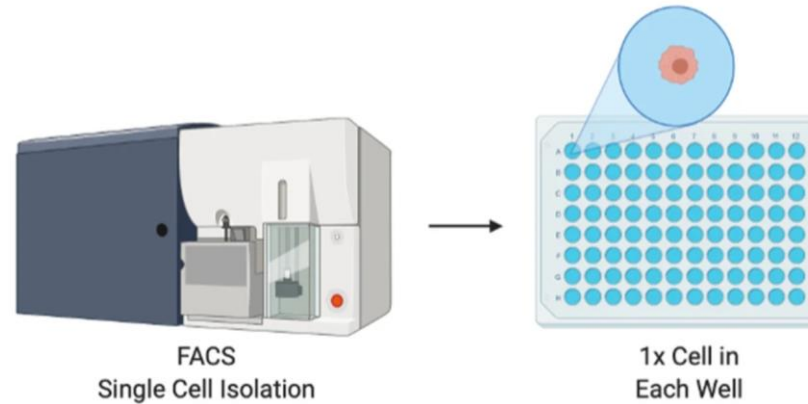
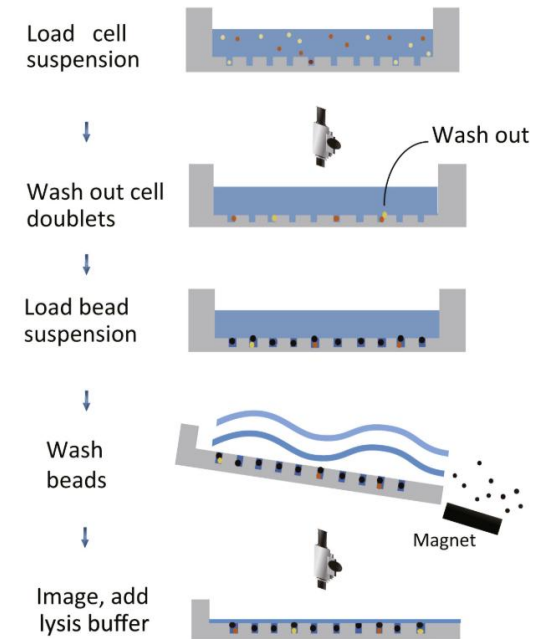


Plate-based
Probst et al., BMC Genomics, 23: 860 (2022)



Microwell-based
Han et al., Cell, 172(5): 1091-
1107.e17 (2018)

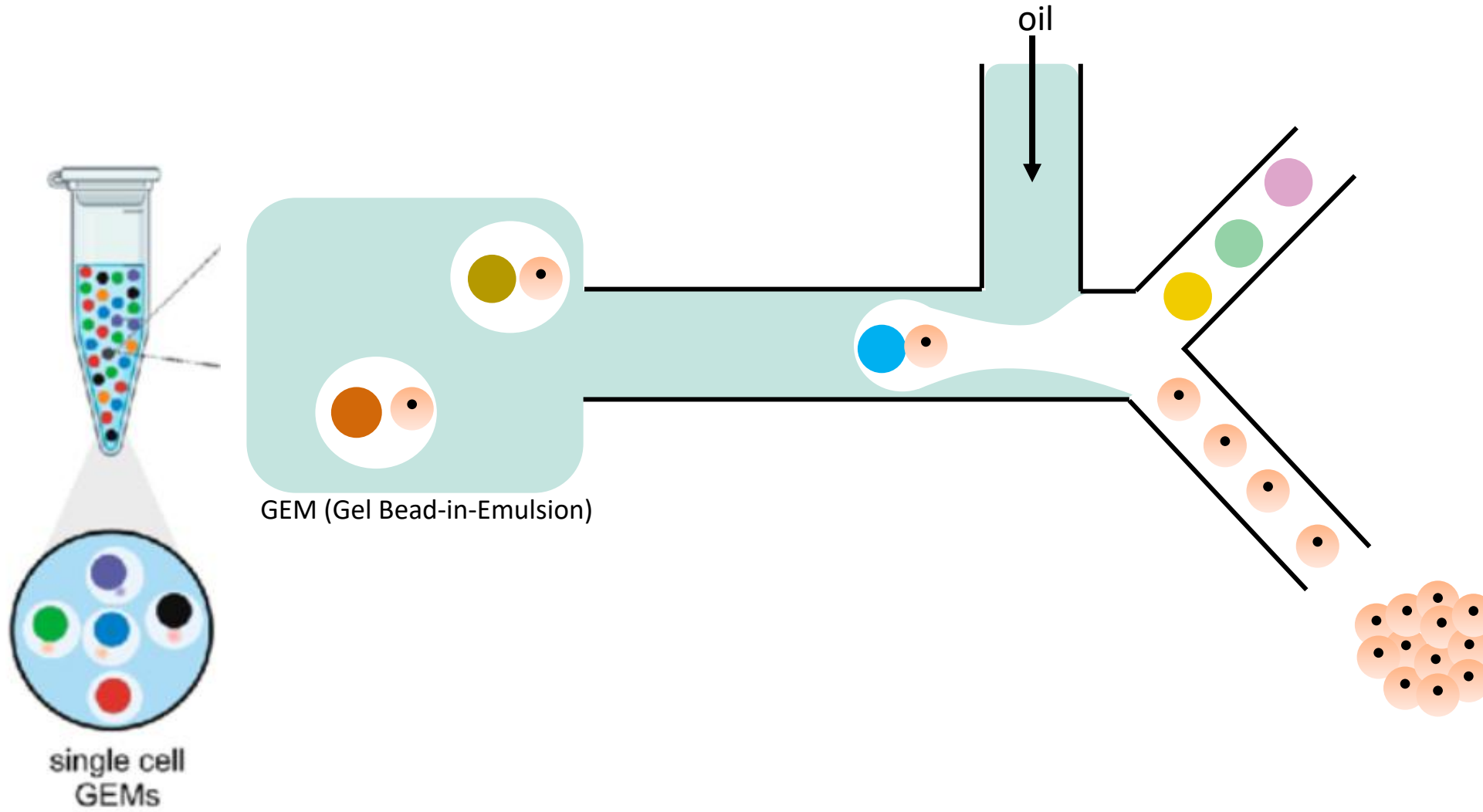
scRNA-seq Wet Lab Pipeline

Protocol	Type	Transcript Coverage	UMI Support	Throughput	Cost per Cell	Special Features / Use Cases
10x Chromium	Droplet-based	3' end or 5' end	Yes	Very High (> 1M cells)	\$0.10–\$0.50	Standard for most scRNA-seq studies; fast and scalable
Smart-seq3	Plate-based	Full-length (with 5' UMI)	Yes (5' only)	Low	\$6–\$10	Full length script, isoform detection
Microwell-seq	Microwell-based	3' end	Yes	Medium	\$0.01-\$0.05	Used in Mouse Cell Atlas; optimized for bulk processing

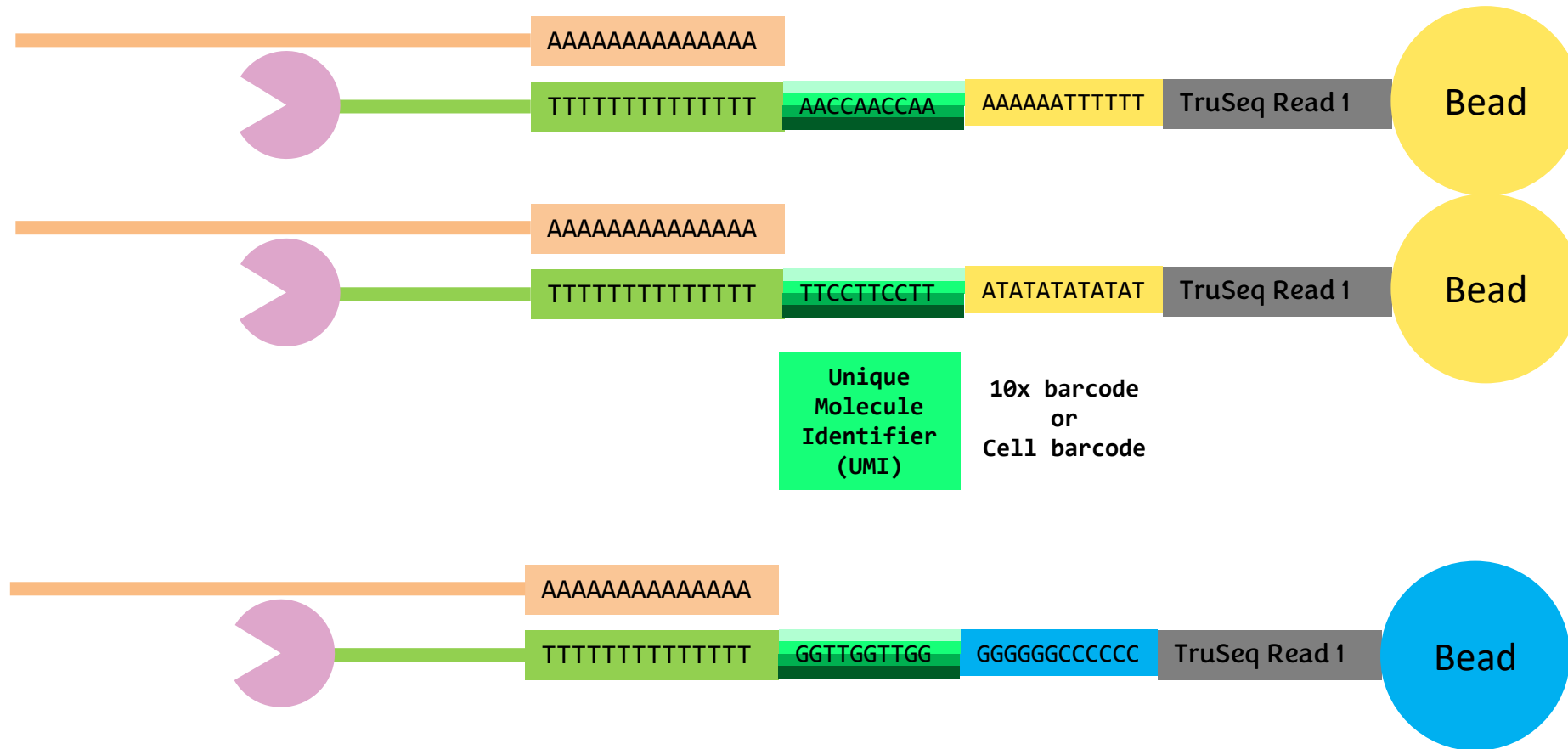
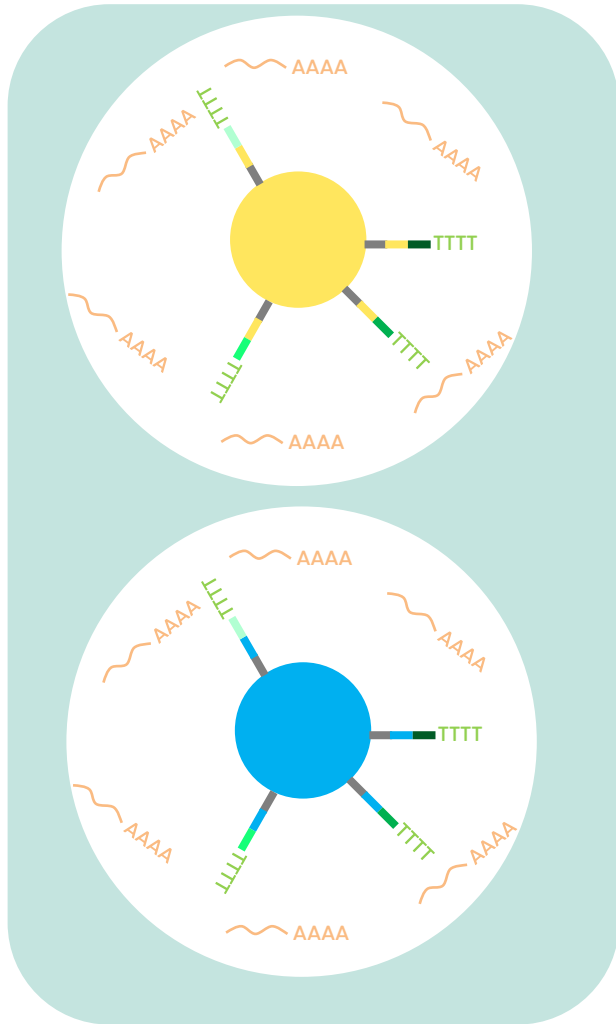
Why 10x Genomics is preferred over others?

- Fully automated
- Consistent and well support
- Can be integrated to multi-modal data such as Assay for Transposase-Accessible Chromatin (ATAC)
- Ease to use and no need custom setup

10x Chromium 3' scRNA-seq – GEM Formation



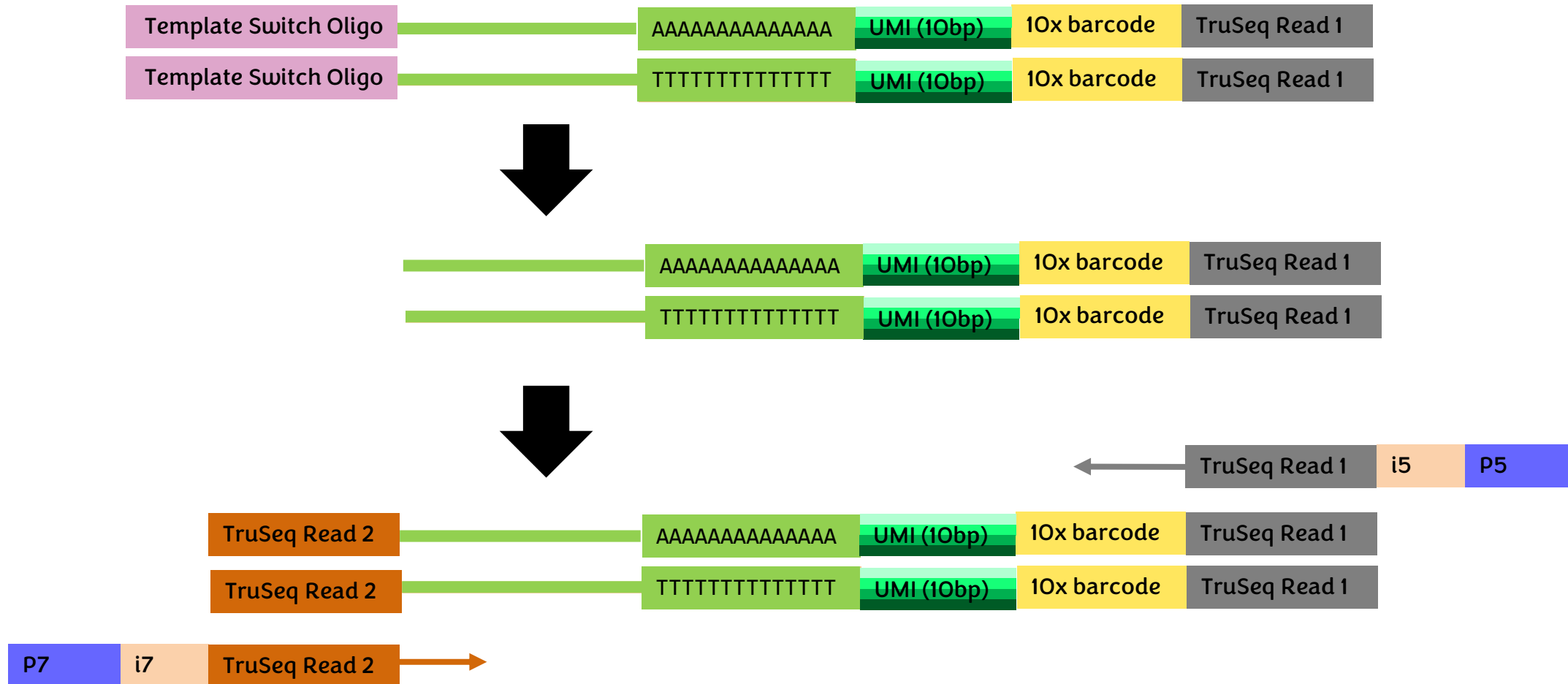
10x Chromium 3' scRNA-seq – Reverse Transcription



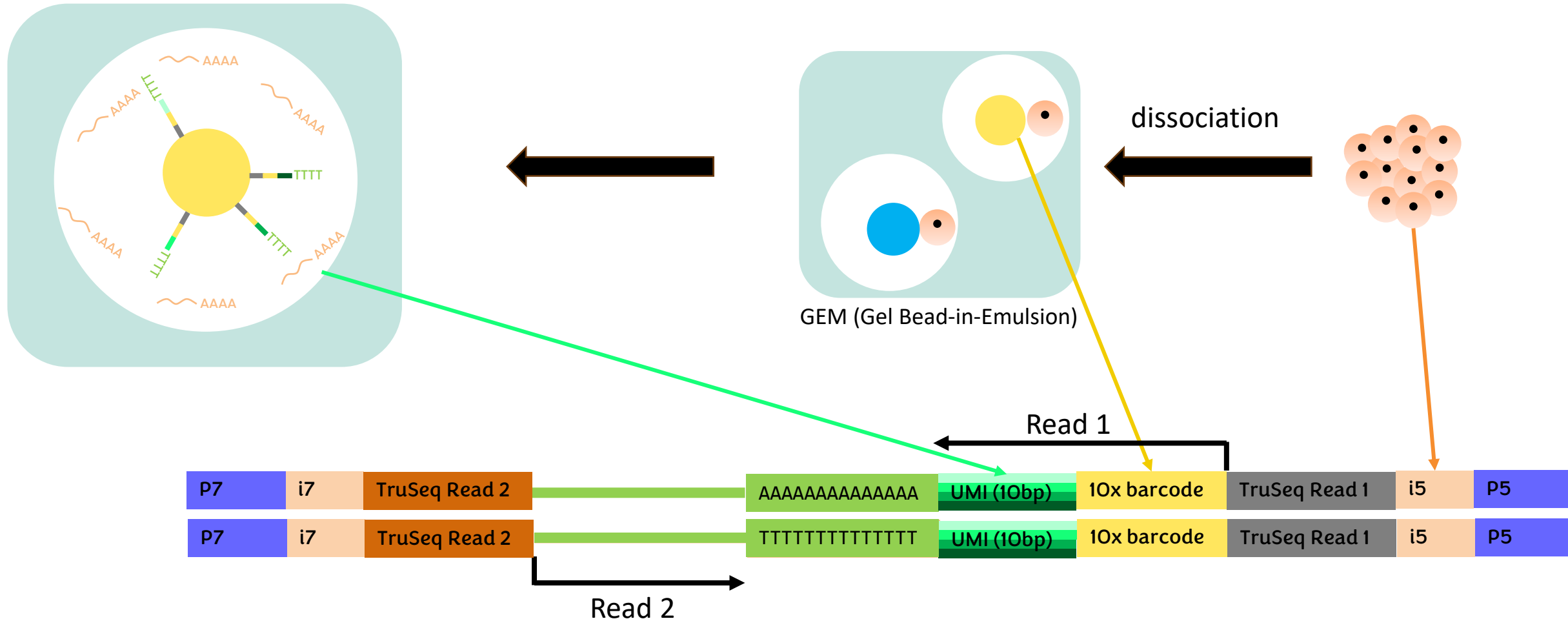
10x Chromium 3' scRNA-seq – Second Strand cDNA



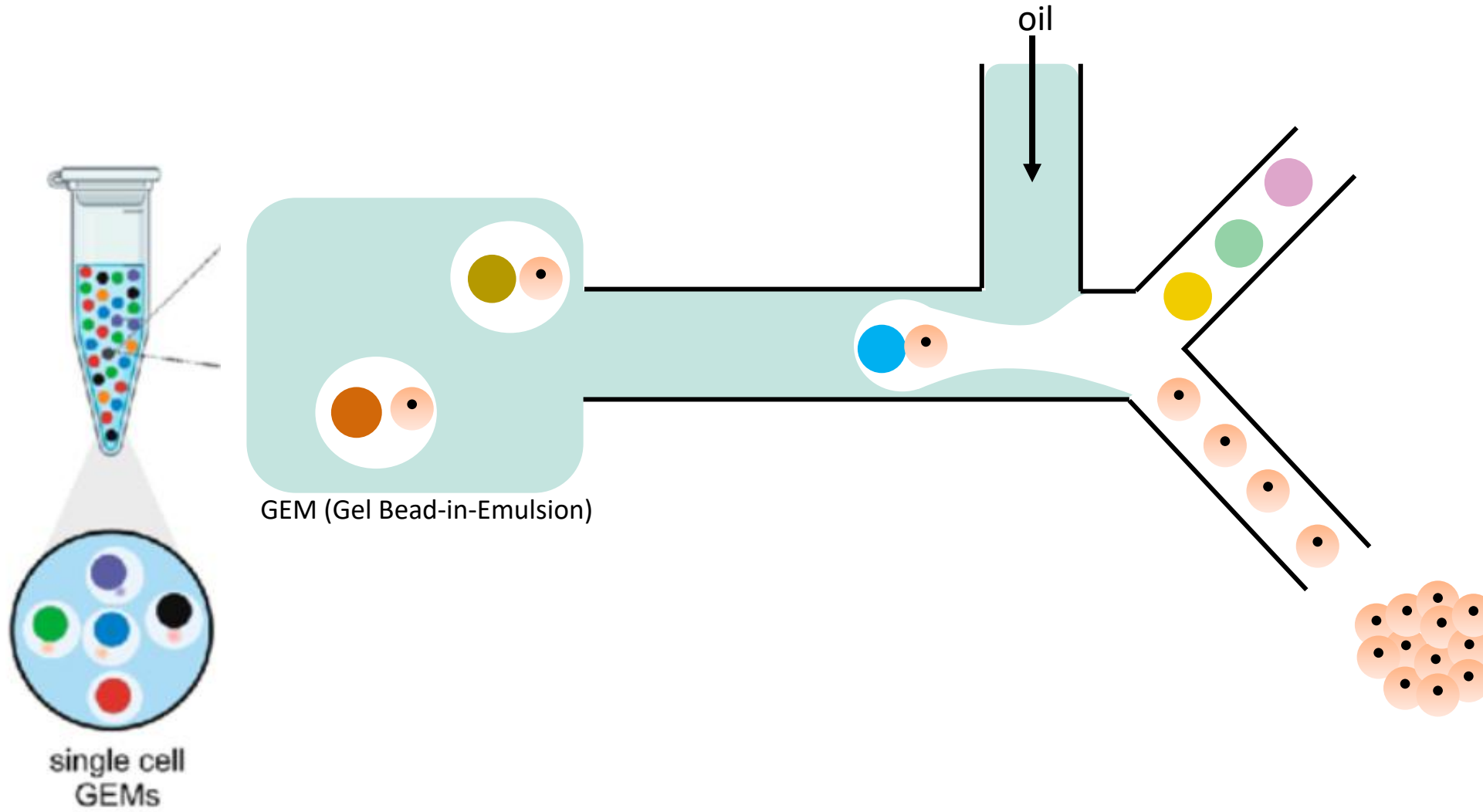
10x Chromium 3' scRNA-seq – Adding sequencing adapter



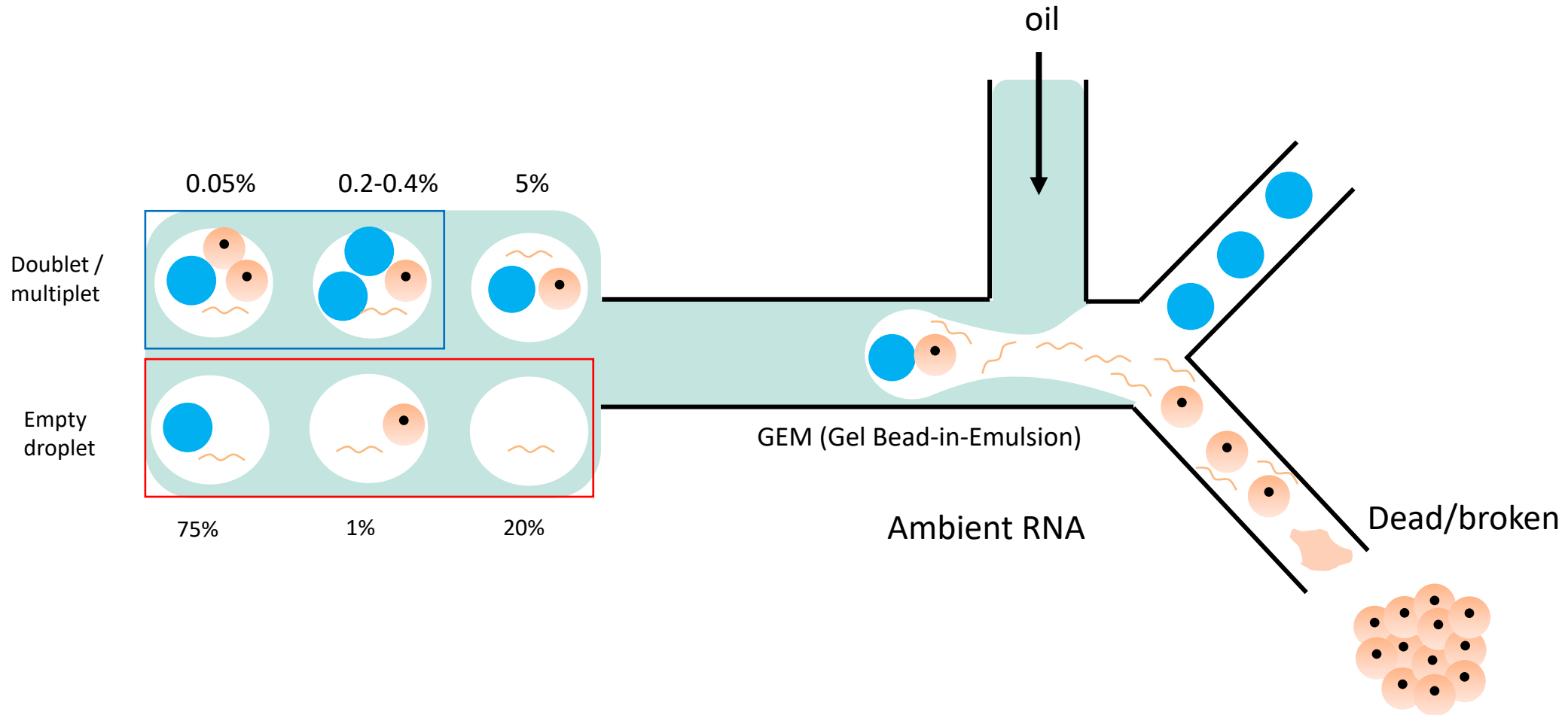
10x Chromium 3' scRNA-seq – Sample, Cell, Molecule



10x Chromium 3' scRNA-seq – GEM Formation

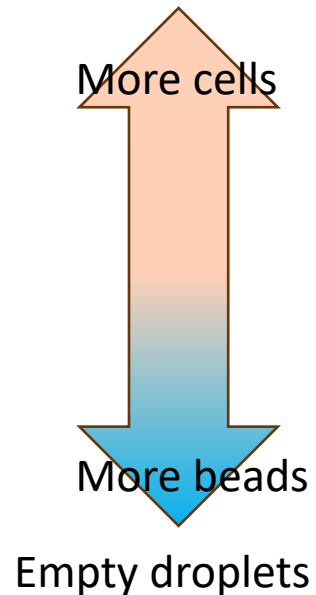


10x Chromium 3' scRNA-seq – In Real World



10x Chromium 3' scRNA-seq – In Real World

Doublets/multiplets



Ambient RNA

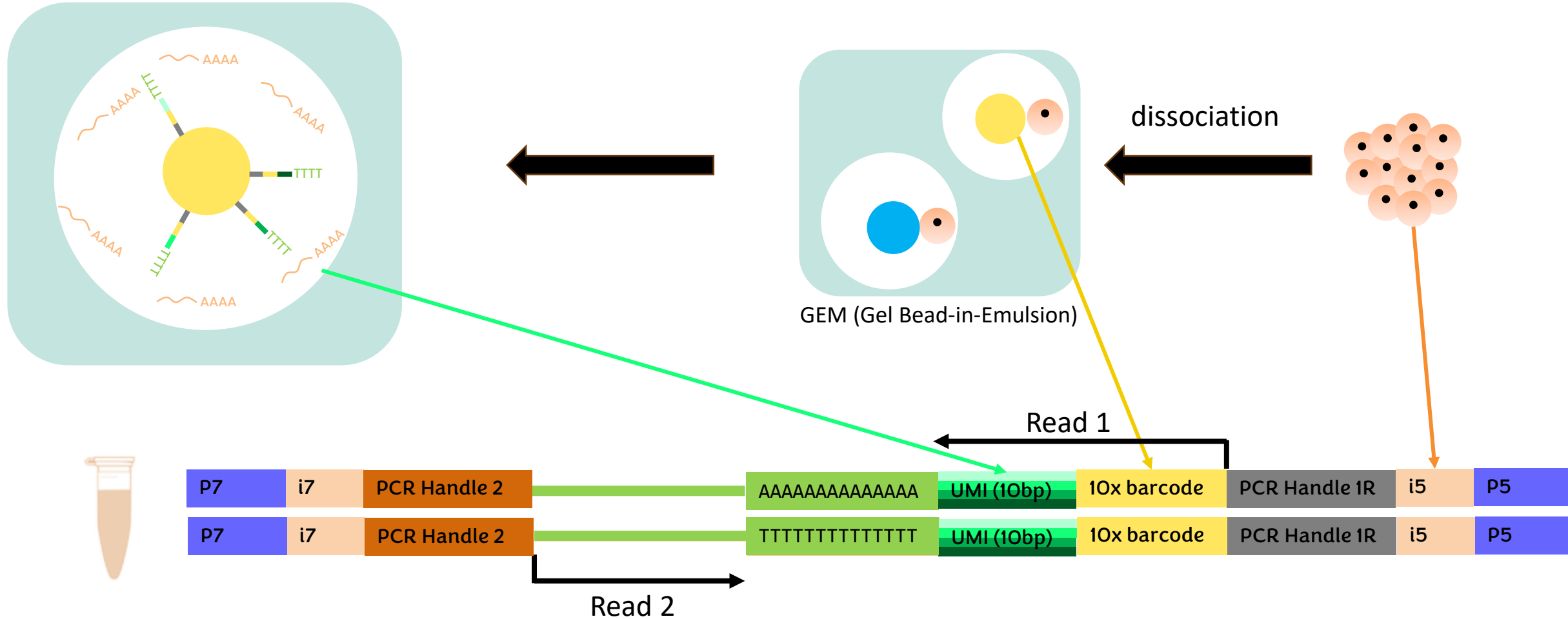
- Sample type
- Tissue dissociation strategy
- Storage and transport conditions
- Wet-lab strategy
- Whether you did a good job

Broken/dead cells

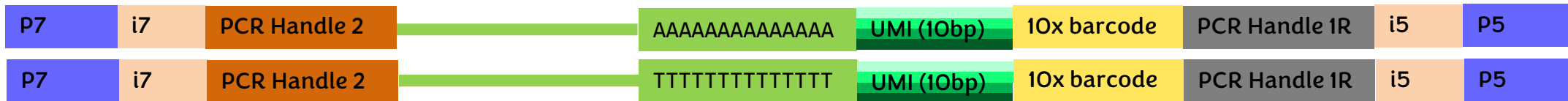
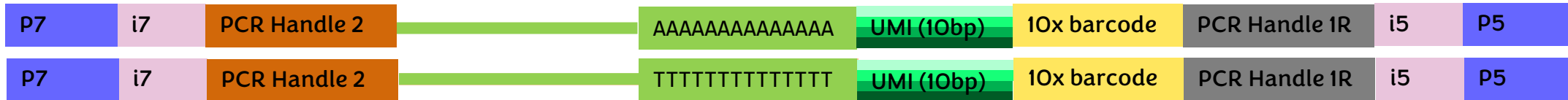
10x Chromium 3' scRNA-seq – Multiplets vs. Throughput

Multiplet Rate (%)	# of Cells Loaded	# of Cells Recovered
~0.4%	~825	~500
~0.8%	~1,650	~1,000
~1.6%	~3,300	~2,000
~2.4%	~4,950	~3,000
~3.2%	~6,600	~4,000
~4.0%	~8,250	~5,000
~4.8%	~9,900	~6,000
~5.6%	~11,550	~7,000
~6.4%	~13,200	~8,000
~7.2%	~14,850	~9,000
~8.0%	~16,500	~10,000

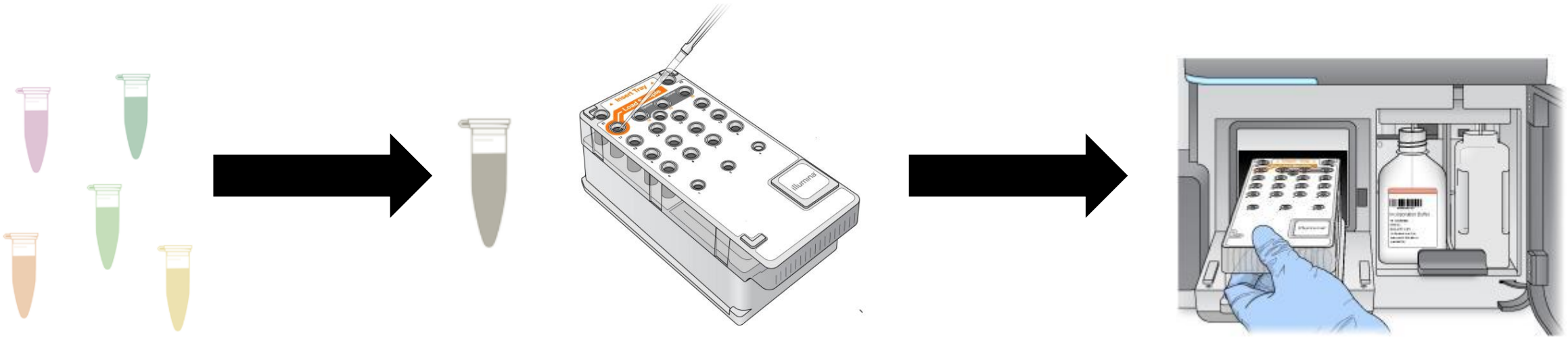
10x Chromium 3' scRNA-seq – One Sample



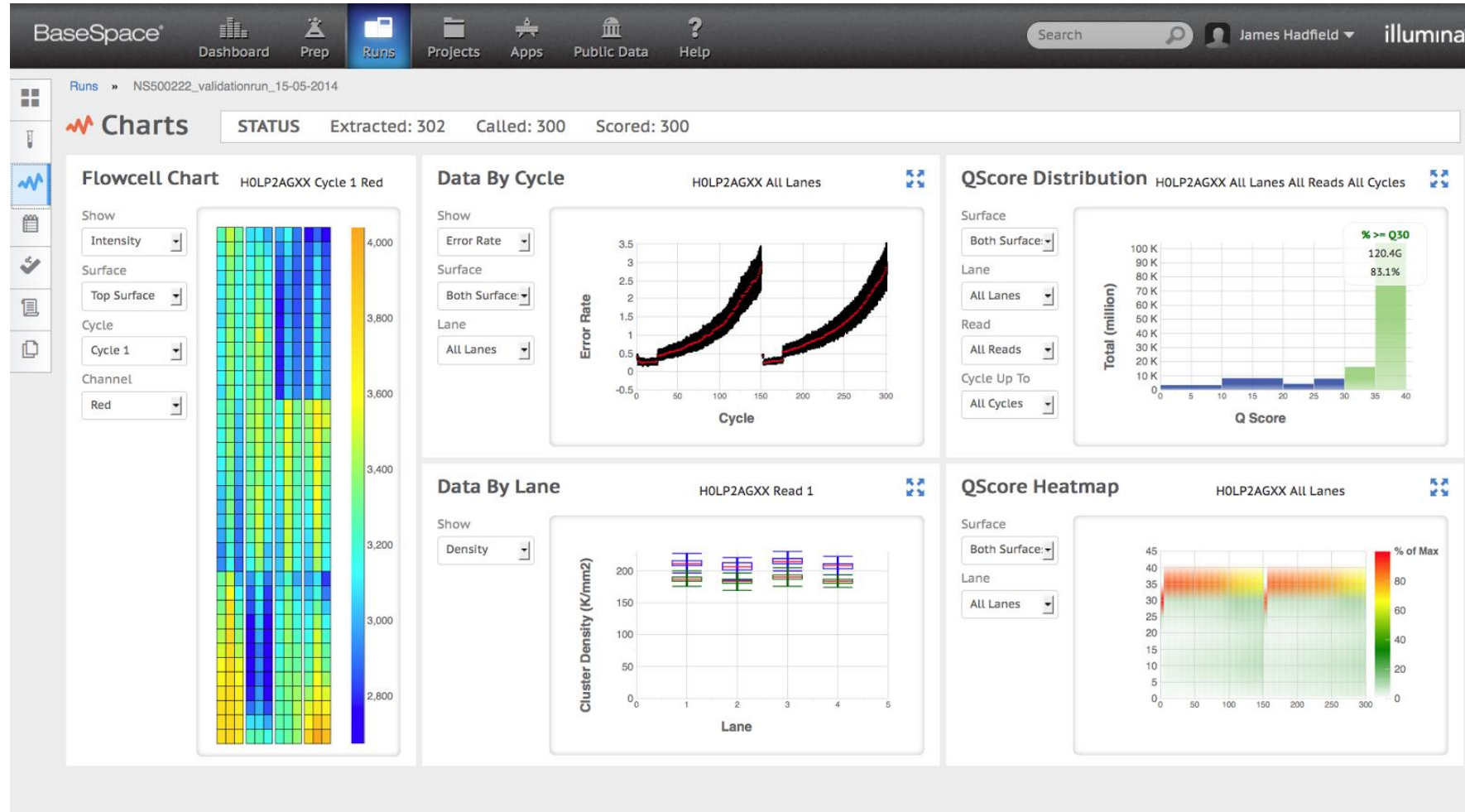
10x Chromium 3' scRNA-seq – Multiple Sample



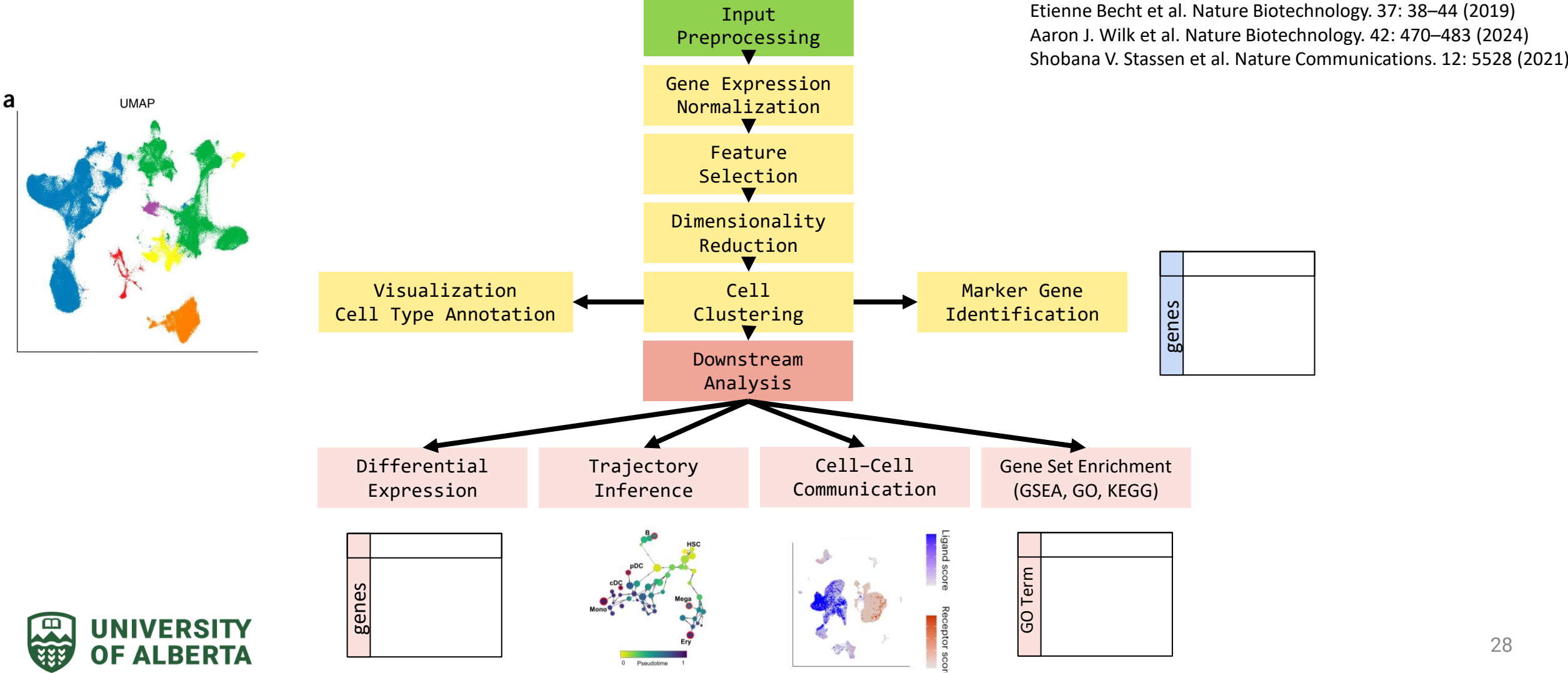
10x Chromium 3' scRNA-seq – Multiplex



10x Chromium 3' scRNA-seq – Sequencing Report



The Pipeline of scRNA-seq Analysis



Input Preprocessing by Cell Ranger



Base calling &
demultiplexing



Alignment &
quantification

Optional Tools:

Cell Ranger

STARsolo

Alevin

Most popular

Open source so you can adjust the parameters, like the tolerance of mismatches

Super fast

Install Cellranger

<https://www.10xgenomics.com/support/software/cell-ranger/downloads#download-links>

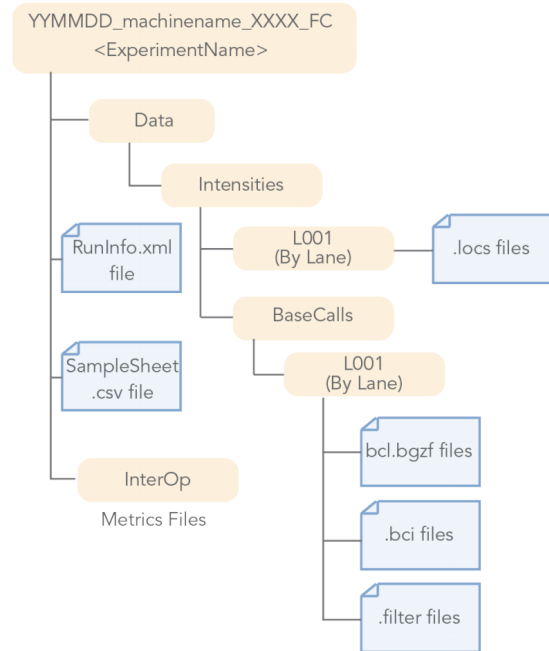
```
wget -O cellranger-9.0.1.tar.gz "https://cf.10xgenomics.com/releases/cell-exp/cellranger-9.0.1.tar.gz?Expires=1746585246&Key-Pair-Id=APKAI7S6A5RYOXBWRPDA&Signature=LT7~WaW0pcQeHJx3HB7Wq51i-JPiwN0ee3NBE0vfdKjtxn0DZAe1-RA-2jIj1aBgeDFlictRWD-hJcqCUgaqiyxfzxdS9Pn-~MctYr9oYswolGHZFJR-a9E0pg7RPxHlP50KPhjWPIUV8z~Z3P2REfgaRTHT2RhOa1Q3Vl01bdj5CQF63tK~qRNgfXpDI1r-La2tGCtJP0qlagYkmMOHJcR-fVM1xFs-BIu7lkY4aW1I1BRch2MjruGvddvtJwU7S-kFx-jYkNawVo00xYePrpqUMotrJGeJ-JF-~hacQ4whBy3Hl6re4U7RDn7vYi92Eir~XXHr0h9rfnfns~GTJw__"
```

```
tar -xf cellranger-9.0.1.tar.gz
```

```
cp -r projects/def-sponsor00/scRNA-seq/Sample_FASTQ/ .
```

Base calling & demultiplexing

Figure 2 BCL Conversion Input Files from the MiniSeq or NextSeq System



SampleName1_S1_L001_R1_001.fastq.gz

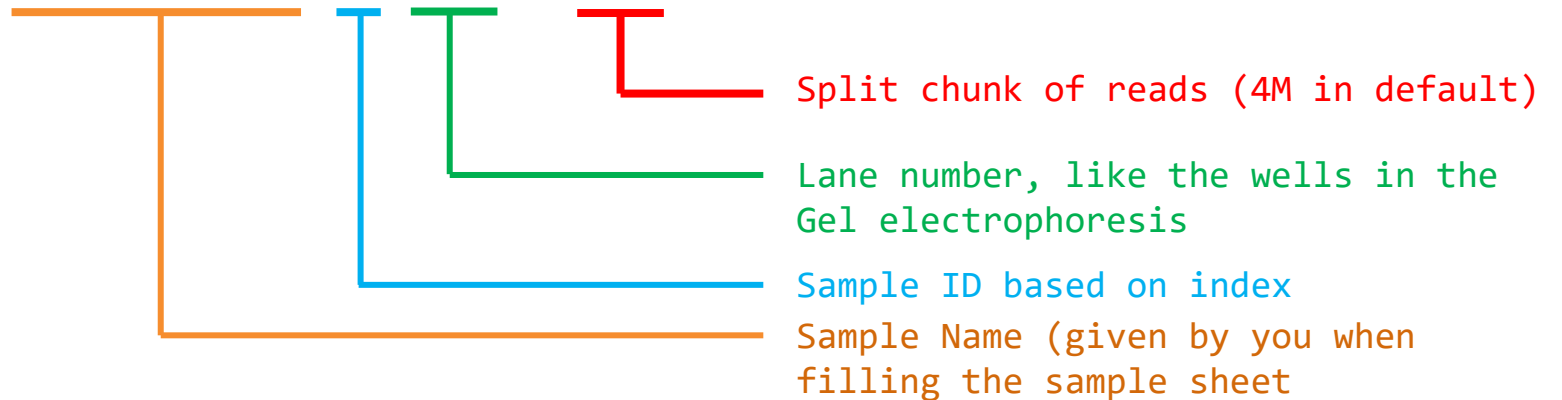
10x barcode+UMI

SampleName1_S1_L001_R2_001.fastq.gz

cDNA sequence

SampleName1_S1_L001_I1_001.fastq.gz

index sequence



```
cellranger mkfastq \
  --id=sample_name_fastq \
  --run=/path/to/bcl_folder \
  --csv=sample_sheet.csv \
  --output-dir=/path/to/fastq/
```

Your Input File: Compressed Fastq

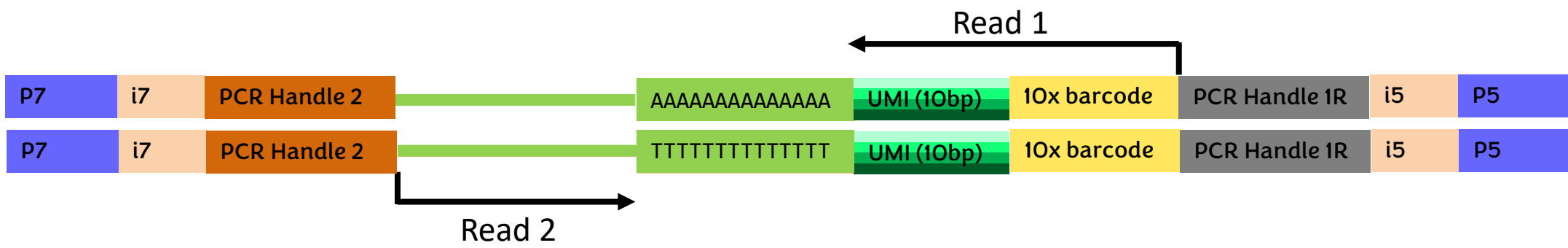
MachineRun IDLaneTileCoordinate

@A00469:87:H5WY2DRX2:1:1101:6247:10324 1:N:0:ATCACG

CCGTATGCGGGGCTCCGATTCCATGTCG

+

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF



@A00469:87:H5WY2DRX2:1:1101:6247:10324 2:N:0:ATCACG

AGACCGGCGGAGGGGCTGGGCGGAGGCTCCGAGAGAGCTGAATGAGGCCTTGGAAGCTCAAGGATGCCAGGAGAGCCGCAGTCAGATCCTAGCGTCGA

+

FF

FASTQ From NCBI SRA

```
SRR9291388_1.fastq.gz      # Read1
SRR9291388_2.fastq.gz      # Read2
SRR9291388_3.fastq.gz      # Index
```

```
@SRR9291388.1 1 length=28
CCGTATGCGGGGCTCCGATTCCATGTCG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
@SRR9291388.2 1 length=28
CCGTATGCGGGGCTCCCTTACCATACGT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Cell Ranger requires FASTQ file names to follow the `bc12fastq` file naming convention.

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

```
SRR9291388_S1_L001_R1_001.fastq.gz  # Read1
SRR9291388_S1_L001_R2_001.fastq.gz  # Read2
SRR9291388_S1_L001_I1_001.fastq.gz  # Index
```



Alignment & quantification – Build The Reference

```
cd  
cp -r projects/def-sponsor00/scRNA-seq/Sample_FASTQ/ .  
export PATH=/project/def-sponsor00/scRNA-seq/cellranger-9.0.1/bin:$PATH
```

```
cd Sample_FASTQ/ref  
cellranger mkref --genome tp53 --fasta tp53.fa --genes tp53.gtf
```

For human reference genome:

<https://support.10xgenomics.com/single-cell-gene-expression/software/downloads/latest>

Alignment & quantification

```
cd $HOME/Sample_FASTQ
vi cellranger.sh
```

```
cellranger count \
  --id output_tp53 \
  --transcriptome ref/tp53 \
  --fastqs fastq \
  --sample tp53test \
  --localcores 1 \
  --localmem 2 \
  --create-bam true \
  --chemistry=SC3Pv4
```

Name of the output folder
Path to reference genome built for Cell Ranger
Folder containing FASTQ files
Sample ID in FASTQ file names
Number of CPU cores
Amount of memory (GB)
whether create a bam file
only for testing in this case, delete it when
 you run your analysis

```
sbatch cellranger.sh
```

Alignment & quantification – check the output

```
cd $HOME/Sample_FASTQ/output_tp53  
du -h | tail -1  
du -h ../fastq
```

The output is 10x larger than the input!

Factors affect the output size:

1. Number of cells
2. Sequencing depth
3. Size of the reference genome
4. Introns included?
5. Bam files generated?

Alignment & quantification – Sequencing Depth

For typical human tissue studies

Cells per sample	5,000 – 20,000
Reads per cell	20,000 – 100,000
Total reads per sample	100M – 1B
Genes detected per cell	A few hundreds to thousands

Factors affecting the sequencing depth:

- Transcriptome size and complexity
- The nature of sample: FFPE, fresh, frozen?
- Genes of interest: highly or lowly expressed? Isoform-specific transcripts?
- Your budget

Alignment & quantification – Computational Resources

Cells × Reads	Cores	RAM	Time
~3k cells, 100M reads	8 cores	32 GB	~1–1.5 hrs
~10k cells, 500M reads	16 cores	64 GB	~2–4 hrs
~50k cells, 1B+ reads	32 cores	128 GB	4–8+ hrs

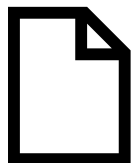
Understand The Output



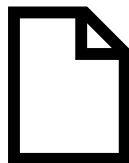
sample_raw_feature_bc_matrix



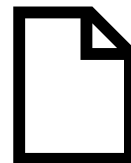
sample_filtered_feature_bc_matrix



web_summary.html



matrix_summary.csv



molecule_info.h5

Understand The Output

- What Cellranger does?

Step	Description	Customization?
Read Alignment	Uses STAR aligner (optimized for spliced transcripts)	No
UMI Filtering	Removes PCR duplicates using UMI + barcode info	No
Barcode Correction	Fixes barcode errors based on whitelist and Levenshtein distance	No
Quality Filtering	Removes low-quality reads, adapters, unaligned reads	No

- Can I just keep on going with the data analysis without checking anything?

NO

- **Two files to check:**
web_summary.html
matrix_summary.csv

File: *web_summary.html*

- Number of cells

Multiplet Rate (%)	# of Cells Loaded	# of Cells Recovered
~0.4%	~825	~500
~0.8%	~1,650	~1,000
~1.6%	~3,300	~2,000
~2.4%	~4,950	~3,000
~3.2%	~6,600	~4,000
~4.0%	~8,250	~5,000
~4.8%	~9,900	~6,000
~5.6%	~11,550	~7,000
~6.4%	~13,200	~8,000
~7.2%	~14,850	~9,000
~8.0%	~16,500	~10,000

File: *web_summary.html*

- Median genes per cell

Cell Type / Sample	Expected Range
Fresh PBMCs	1,000–2,500
Solid tissue	500–1,500
FFPE or nuclei	200–800

File: *web_summary.html*

- Medium UMI counts per cell
Reflects effective sequencing depth per cell.
Fresh PBMCs: ~5,000–15,000
FFPE or nuclei: ~1,000–5,000

File: *matrix_summary.html*

- Sequencing saturation (>30%, ideally >=50%)
= $1 - (\text{Unique UMIs} / \text{Total Reads})$

Check your “matrix_summary.csv” file

Are these two file enough?

- No, when:

Low alignment rate (<70%)

Could mean contamination, degraded RNA, poor annotation, or library issue

Unexpected cell types / bad clusters

Maybe ambient RNA wasn't properly removed or mapping is off

Custom genome / annotation

STAR may fail to align properly; misannotation or poor reference

Exonic vs intronic reads

Important for snRNA-seq (nuclei-based), where you expect more intronic reads

Summary

- The principle, history and pipeline of scRNA-seq
- Understand the data from the wet lab
- Did input preprocessing via CellRanger
- Assessed the quality of CellRanger outputs

Question?

Day 2. scRNA-seq Analysis



Notes

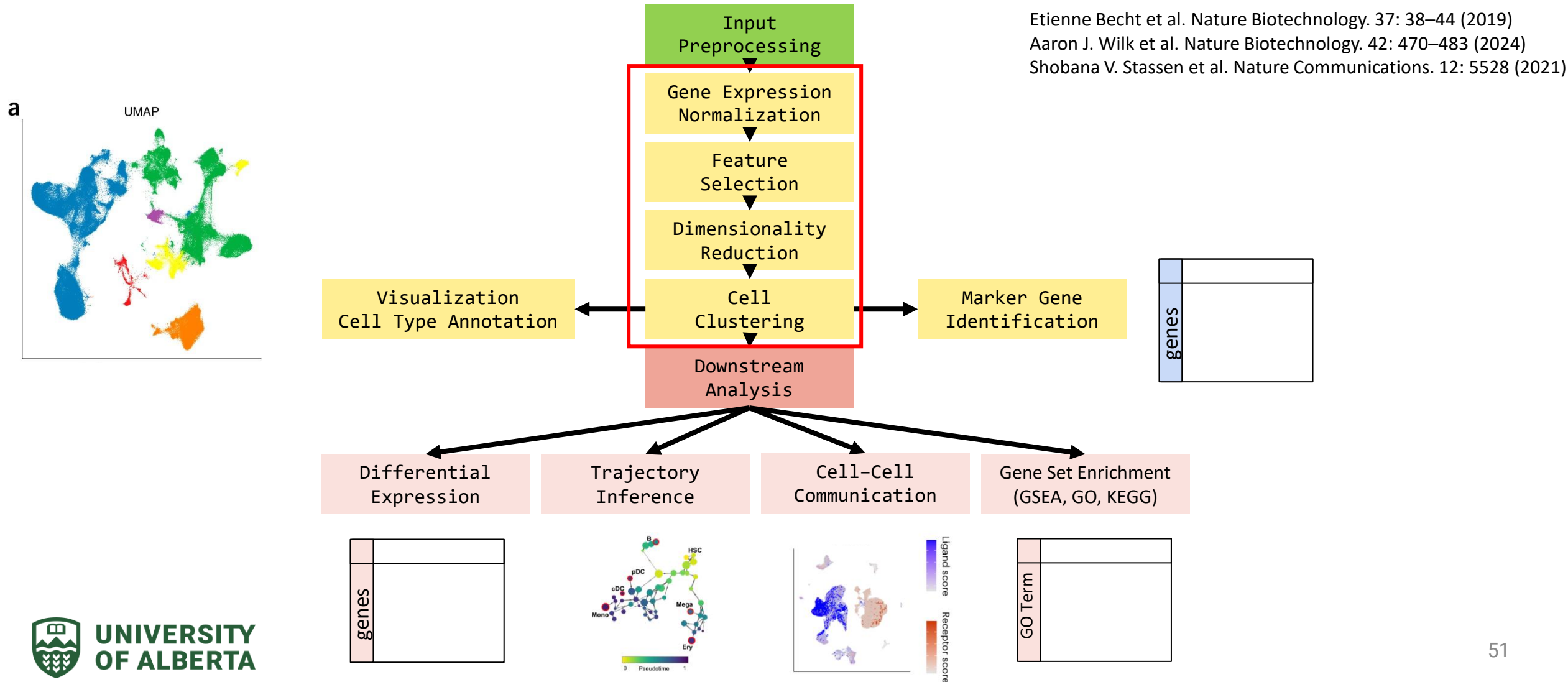
- The full code can be found at:
<https://github.com/ualberta-rcg/scRNA-seq>
- The slides list those codes that you need to make changes when running your own projects. They are highlighted in red, for example:

```
.....  
raw_dat <- Seurat::Read10X(data.dir = "/usr/local/10x_data/sample_raw_feature_bc_matrix")  
.....
```

Start Rstudio

```
cd $HOME  
cp $HOME/projects/def-sponsor00/scRNA-seq/rstudio4.3/scRNA-seq.sif .  
cp $HOME/projects/def-sponsor00/scRNA-seq/Job_scripts/test_cluserter.sh .  
sbatch test_cluserter.sh
```

The Pipeline of scRNA-seq Analysis

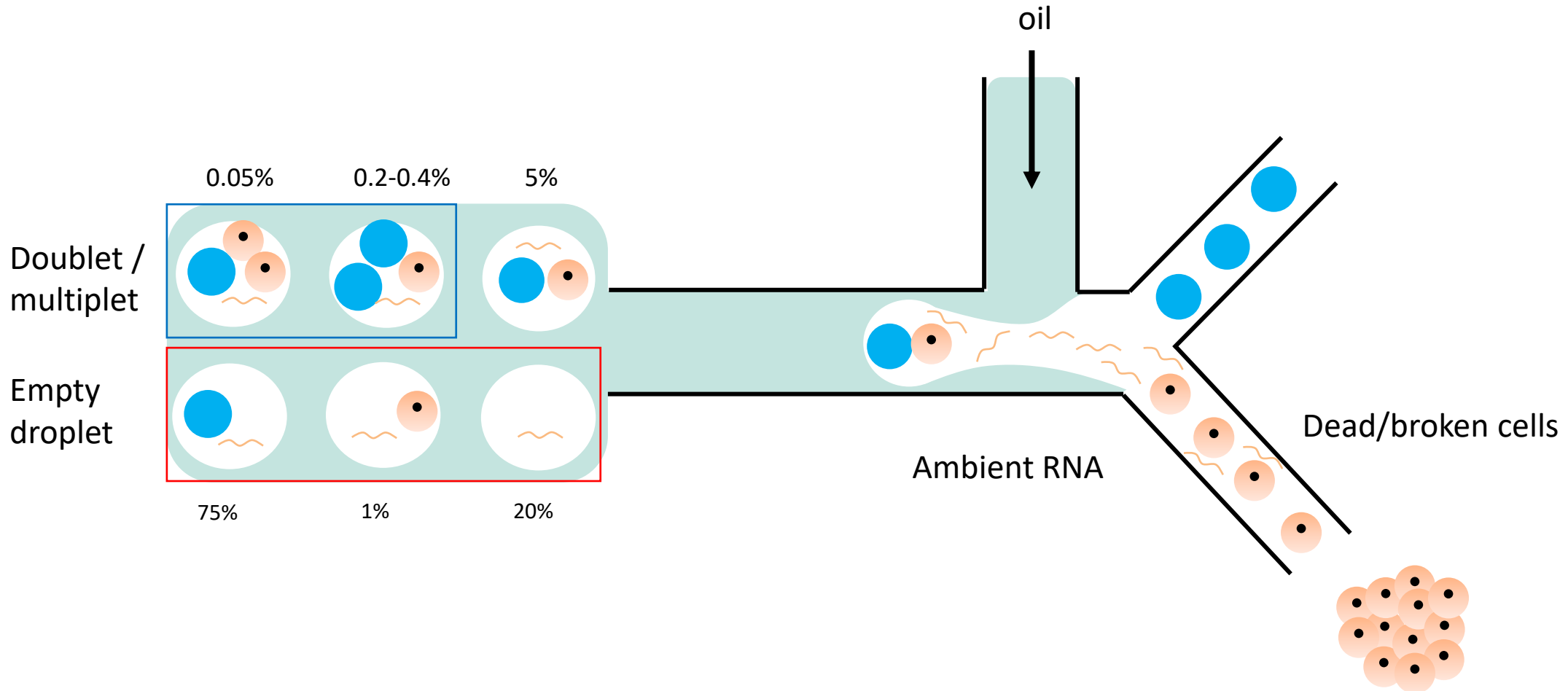


The Pipeline of scRNA-seq Analysis

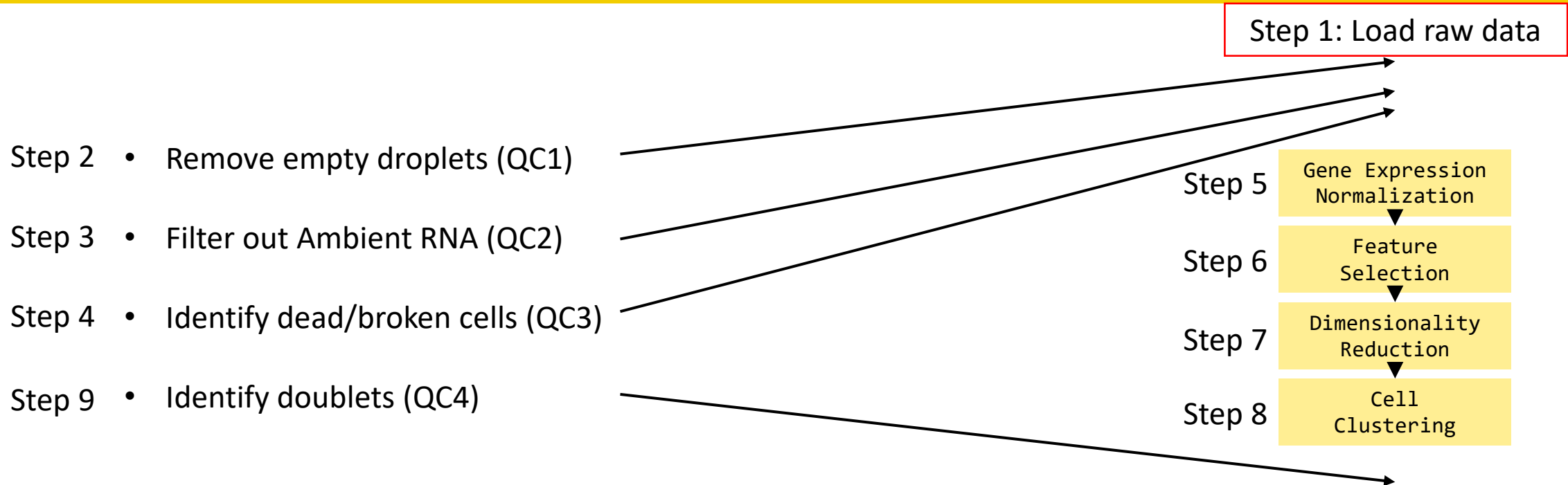
```
filter_dat <- Seurat::Read10X("filtered_feature_bc_matrix/")
seur_obj <- Seurat::CreateSeuratObject(filter_dat, min.cells=5, min.features=100)
seur_obj <- Seurat::NormalizeData(seur_obj)
seur_obj <- Seurat::FindVariableFeatures(seur_obj)
seur_obj <- Seurat::ScaleData(seur_obj)
seur_obj <- Seurat::RunPCA(seur_obj)
seur_obj <- Seurat::FindNeighbors(seur_obj)
seur_obj <- Seurat::FindClusters(seur_obj)
```

- These steps are not overwriting existing data, but add data into “seur_obj”
- Raw “seur_obj” is still available
- “seur_obj” will get bigger and bigger during analysis
- It can store multiple samples, so the size could be MB to GB

Quality Control



Quality Control



There are multiple tools for each step!!!!!!

(Code) Step 1: Load the raw matrix

```
.....  
raw_dat <- Seurat::Read10X(data.dir = "/usr/local/10x_data/sample_raw_feature_bc_matrix")  
.....
```

	AAACCTGAGATAGGAG-1	AAACCTGAGATCCTGT-1	AAACCTGAGATTACAA-1	...
TP53	6	0	2	
KRAS	3	0	7	
...				

Quality Control

Step 2 • Remove empty droplets (QC1)

Step 3 • Filter out Ambient RNA (QC2)

Step 4 • Identify dead/broken cells (QC3)

Step 9 • Identify doublets (QC4)

Step 1: Load raw data

Step 5

Gene Expression
Normalization

Step 6

Feature
Selection

Step 7

Dimensionality
Reduction

Step 8

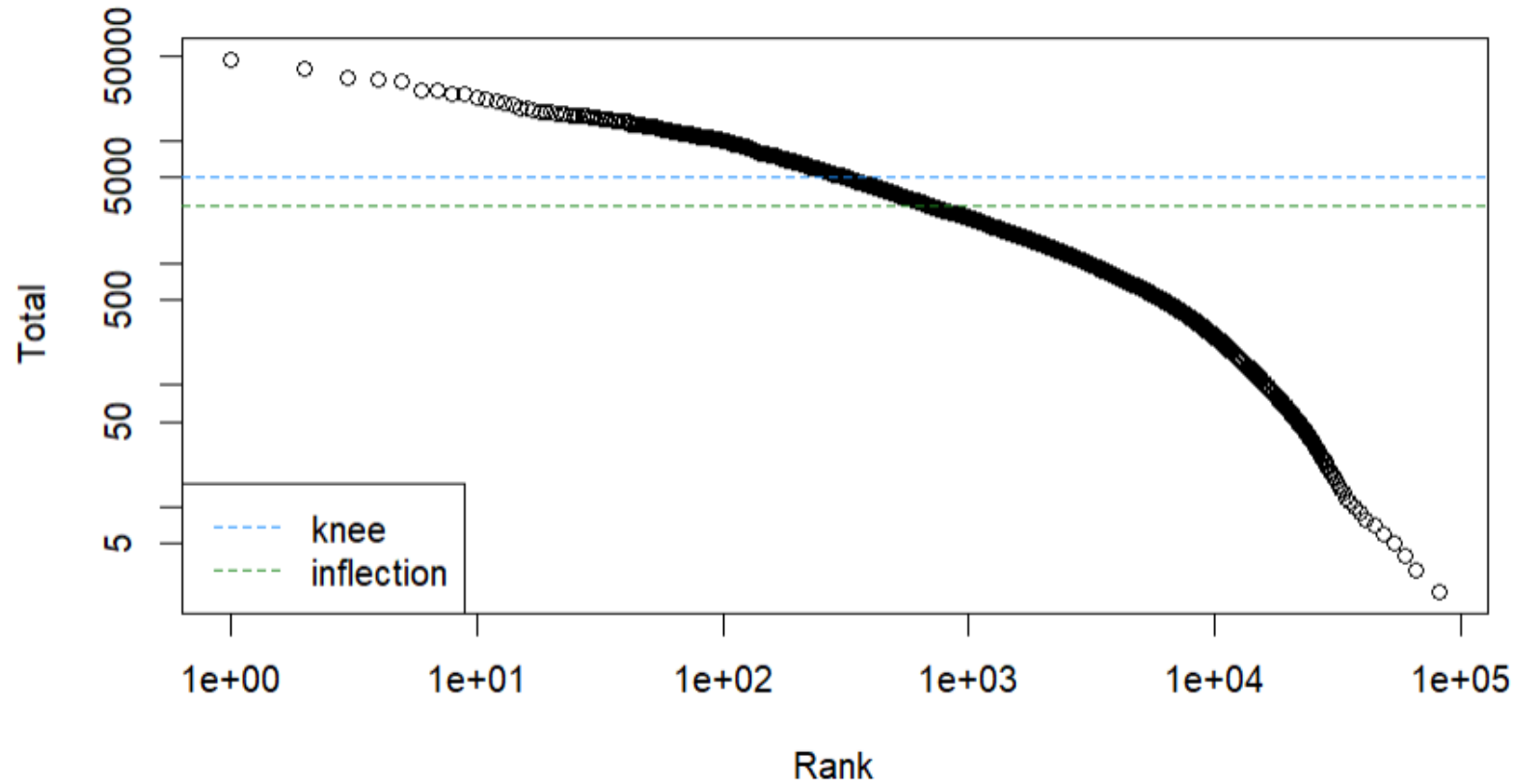
Cell
Clustering

QC1: Remove empty droplets

- Three methods:
 - Cell Ranger Strategy
 - Knee/Inflection
 - Poisson

QC1: Remove empty droplets

- Barcode Rank Plot



Knee: min 2nd derivative (Max curve bending)

Inflection: 2nd derivative = 0

QC1: Remove empty droplets

- Summary of three methods

Method	Based on...	Sensitivity	Specificity	Best for...
Cell Ranger Strategy	Proprietary + knee-like model + internal heuristics	High	Variable	Large cell recovery
Knee/Inflection	Max curve bending (min 2nd derivative) / 2nd derivative = 0	Low-Medium	Medium-High	Explore visually
Poisson	Statistical background noise model	Low	High	Low quality sample (FFPE)

(Code) Step 2: Identify the empty droplets

```
e.out <- emptyDrops(raw_dat, lower=100, niters=10000, ignore=NULL, retain=2*br.out$knee)
```

Step 3: Filter Out Ambient RNA (QC2)

Step 2 • Remove empty droplets (QC1)

Step 3 • Filter out Ambient RNA (QC2)

Step 4 • Identify dead/broken cells (QC3)

Step 9 • Identify doublets (QC4)

Step 1: Load raw data

Step 5

Gene Expression
Normalization

Step 6

Feature
Selection

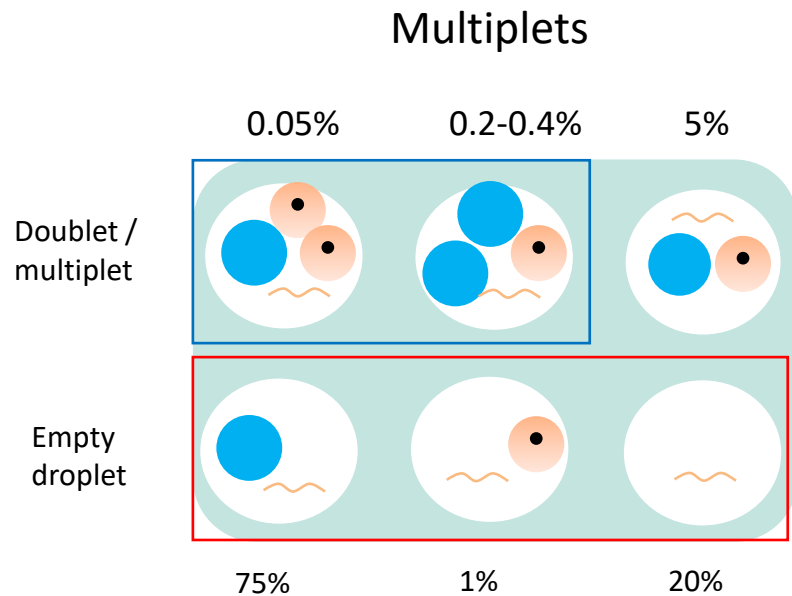
Step 7

Dimensionality
Reduction

Step 8

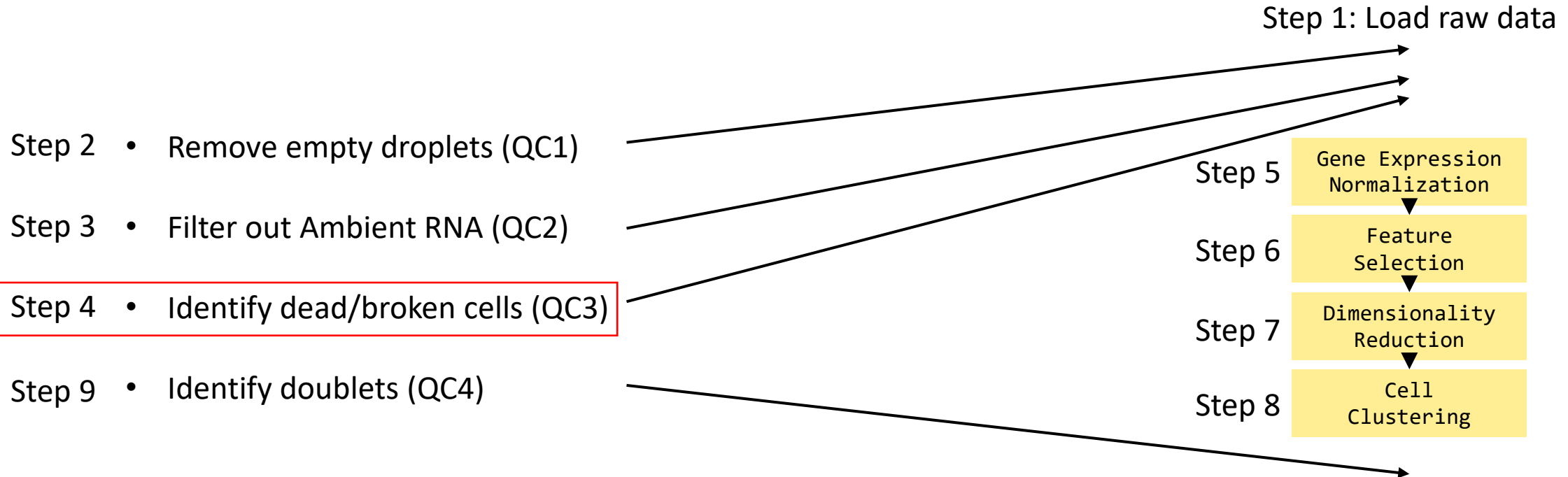
Cell
Clustering

Step 3: Filter out ambient RNA (QC2)



- There are multiple tools/ways.
We are using DecontX today because:
 - It is easy to be integrated into the pipeline.
 - It doesn't need GPU.
 - The syntax is simple and clean.
- Group cells into clusters and estimates cluster-specific expression profiles
- For each cell, tune the cell-specific contamination fraction to make the gene expression profile match the observation and the cluster as much as possible.

Step 4: Identify Dead/Broken Cells (QC3)



Step 4: Identify the dead / broken cells (QC3)

- Through Mitochondrial genes.

Consideration

Human PBMCs / tumors / organs

Low-input / fragile tissues (e.g., FFPE, brain)

Visual inspection

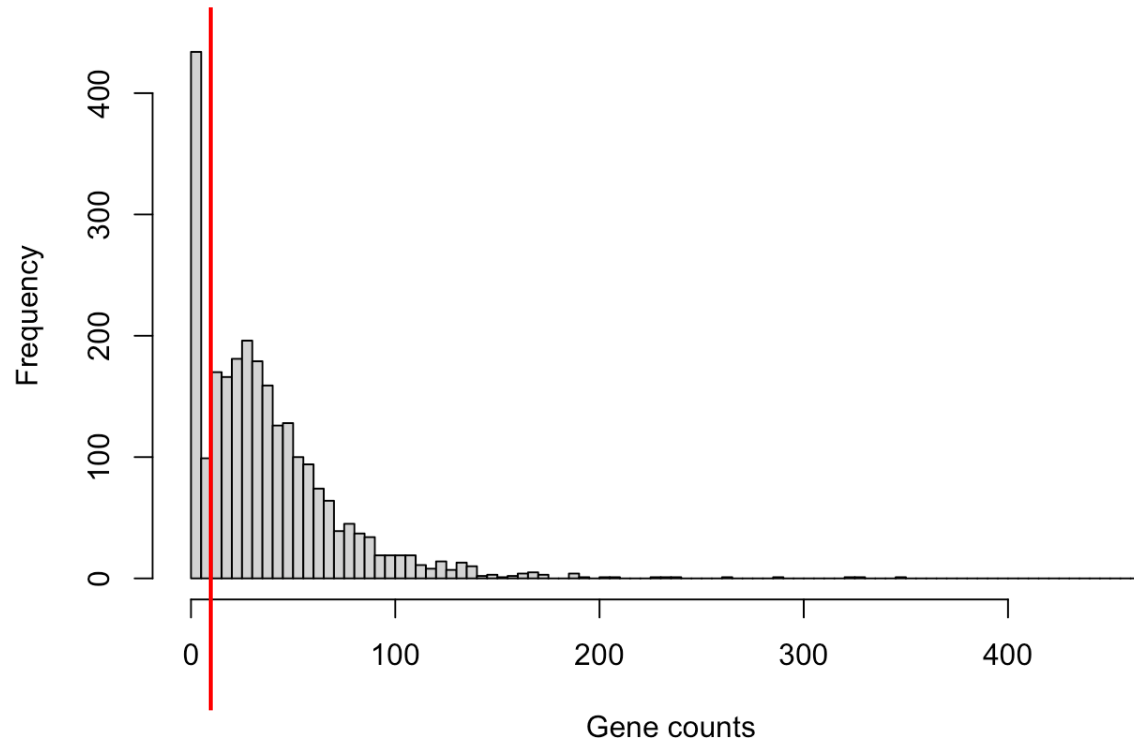
Guidance

200–500 gene threshold is typical. MT% cutoff: 30%

Use a **lower threshold** (100–300). MT% cutoff: 40-60%

Use `hist(sce$detected)` or violin plots to **see natural separation**

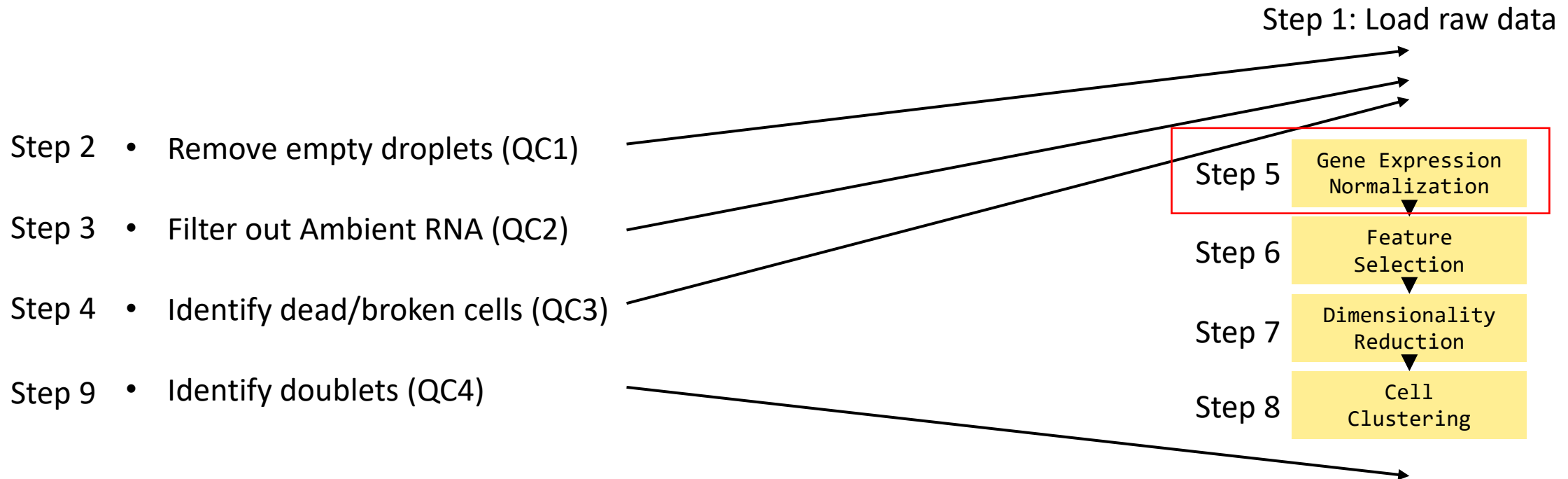
Choose your genes based on your project, for example, for PBMC sample, you may want to remove those red blood cells with high hemoglobin mRNAs (>0.5-1%).



Step 4: Identify dead/broken cells (QC3)

```
.....  
.....  
is.mt <- grepl("^MT-", rowData(sce)$Symbol)  
.....  
.....  
.....  
cell_filter_detect <- sce$detected < 100  
cell_filter_MT <- sce$subsets_Mito_percent > 30
```

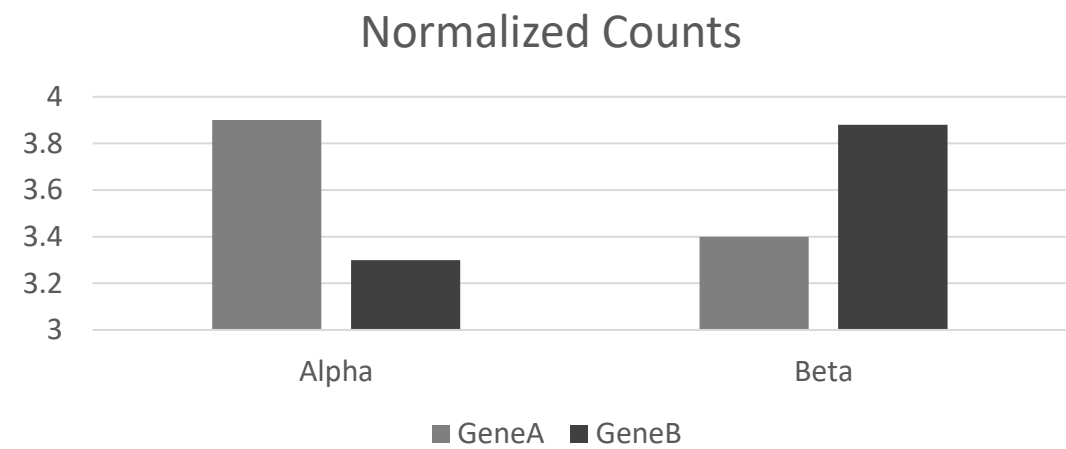
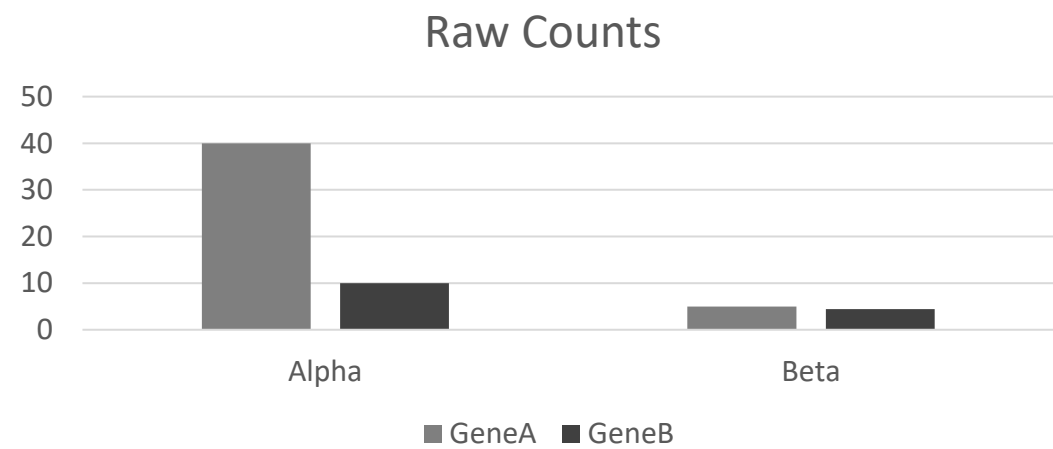
Step 5: Gene Expression Normalization



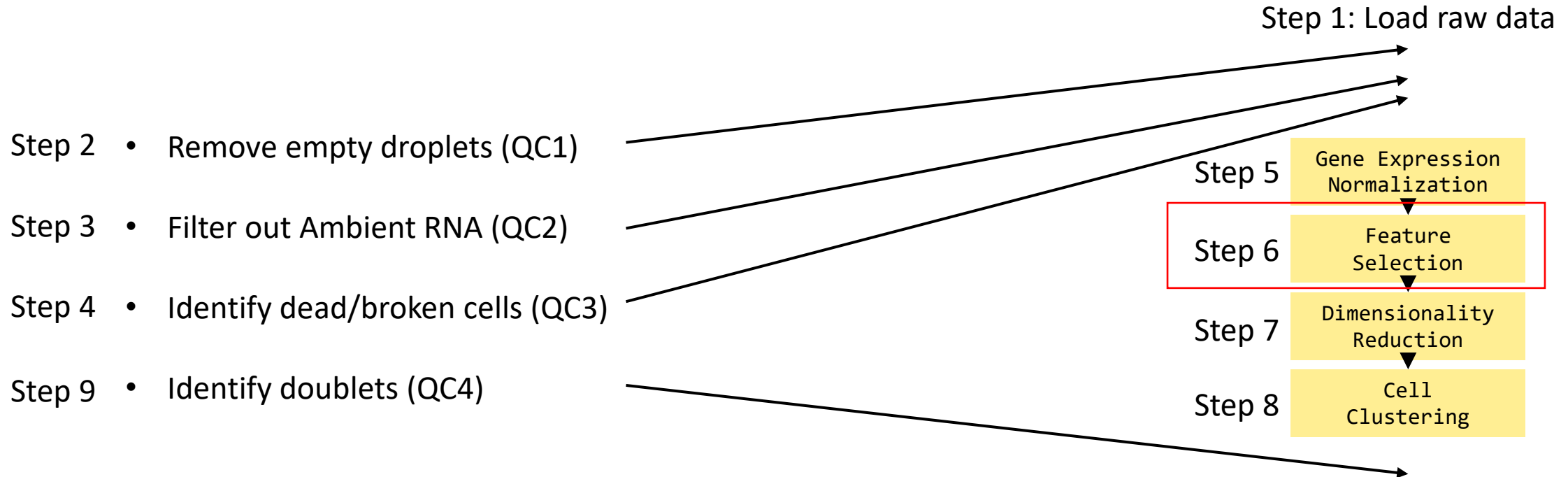
Step 5: Gene Expression Normalization

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

Cell	Gene	UMIs	Scaling	Log Transformation
Alpha	A	40	$40 / (40 + 10) * 10000 = 8000$	$\log(1 + 8000) = 3.90$
	B	10	$10 / (40 + 10) * 10000 = 2000$	$\log(1 + 2000) = 3.30$
Beta	A	5	$5 / (5 + 15) * 10000 = 2500$	$\log(1 + 2500) = 3.40$
	B	15	$15 / (5 + 15) * 10000 = 7500$	$\log(1 + 7500) = 3.88$



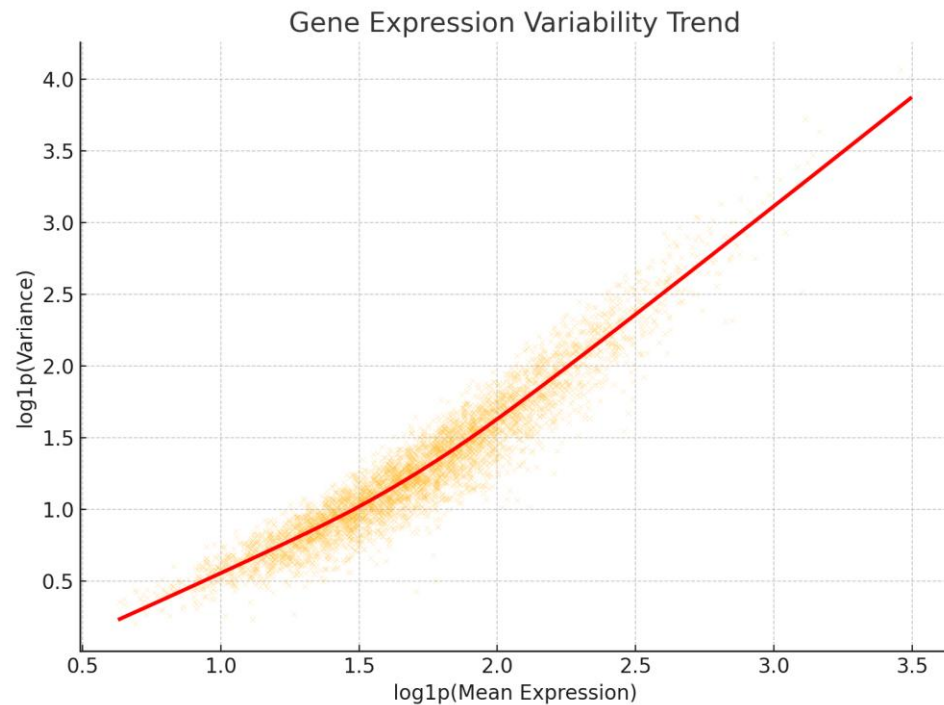
Step 6: Feature Selection



Step 6: Feature Selection - Identify highly variable genes

```
seur_filtered <- FindVariableFeatures(seur_filtered, selection.method = "vst", nfeatures = 500)
```

Identify the most **xxxx** highly variable genes through “Variance Stabilizing Transformation” (VST).



Step 6: Feature Selection - Identify highly variable genes

```
seur_filtered <- FindVariableFeatures(seur_filtered, selection.method = "vst", nfeatures = 500)
```

Identify the most **xxxx** highly variable genes through “Variance Stabilizing Transformation” (VST).



How to determine “nfeatures”?

Total genes detected per cell	Suggested <i>nfeatures</i>
< 500	300–500
~1000	500–1000
>2000	Up to 2000

```
summary(seur_filtered$Feature_originalexp)
```

Step 6: Feature Selection - Identify highly variable genes

```
seur_filtered <- FindVariableFeatures(seur_filtered, selection.method = "vst", nfeatures = 500)
```

Identify the most **xxxx** highly variable genes through “Variance Stabilizing Transformation” (VST).

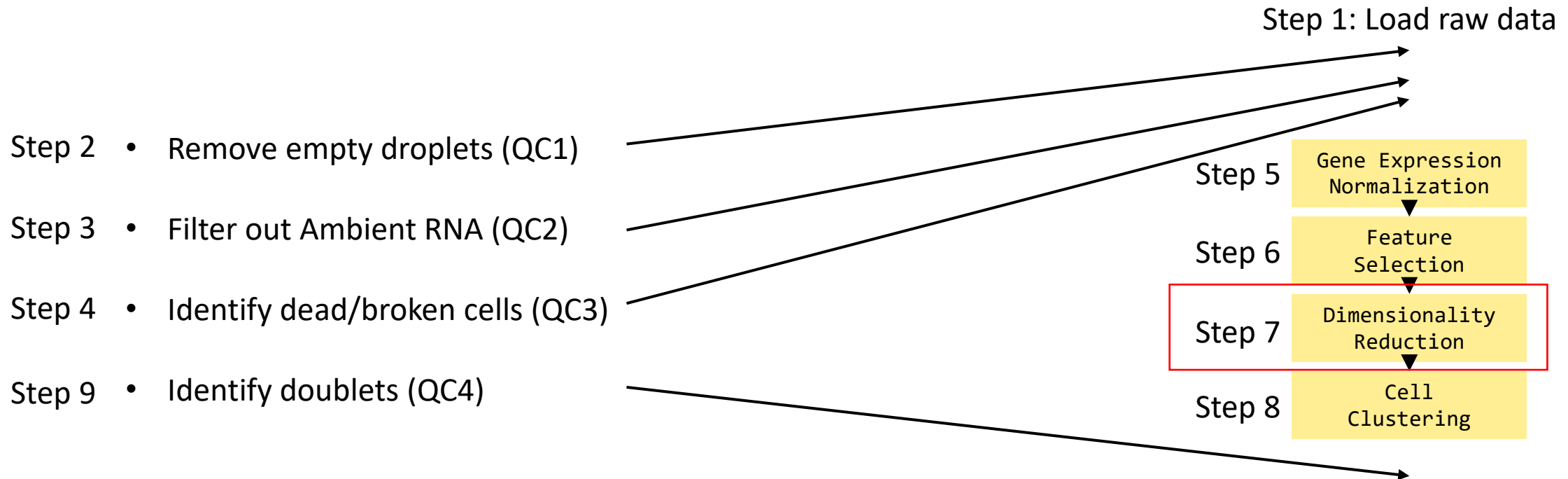
- This is arbitrary.
- You can play with the number and see which makes more sense.
- Usually, it wouldn’t change much from 2000 to 2100, but it usually worth testing 400, 500, 600.

How to determine “nfeatures”?

Total genes detected per cell	Suggested <i>nfeatures</i>
< 500	300–500
~1000	500–1000
>2000	Up to 2000

```
summary(seur_filtered$nFeature_originalexp)
```

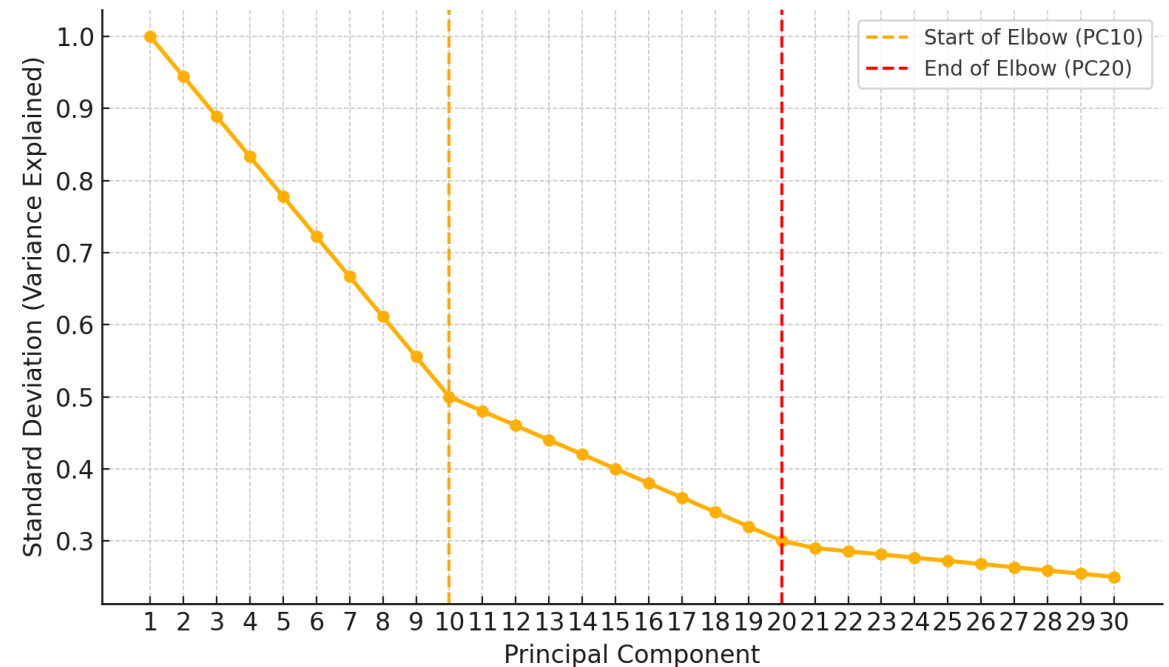
Step 7: Dimensionality Reduction



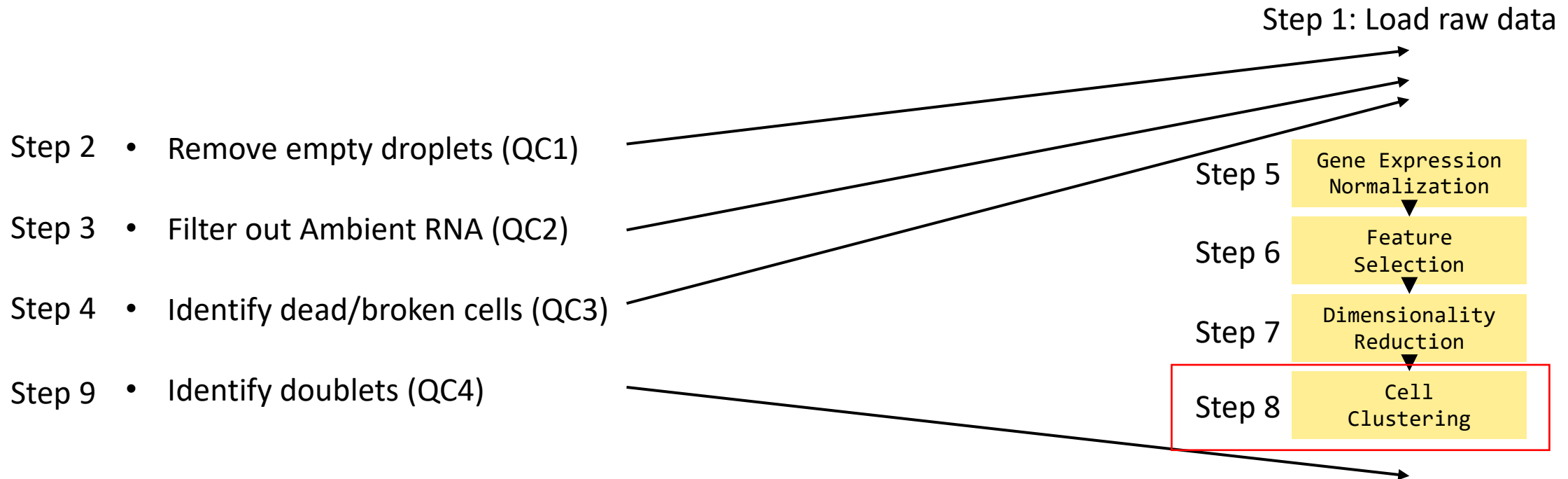
Step 7: Dimensionality Reduction

- Principle Component Analysis (PCA) is the most popular way to reduce dimensionality.
- Why do we want to reduce dimensionality?
 - Remove noise
 - Speed up computation
 - Visualize the data
 - Reveal biological structure
- Determine how many PCs via ElbowPlot

```
ElbowPlot(seur_filtered)  
PCs <- 7
```



Step 8: Cell Clustering



Step 8: Cell Clustering

```
seur_filtered <- FindNeighbors(seur_filtered, dims = 1:PCs)
seur_filtered <- FindClusters(seur_filtered, resolution = 0.3)
```

Resolution value

0.1

0.3

0.8

≥1.0

Effect

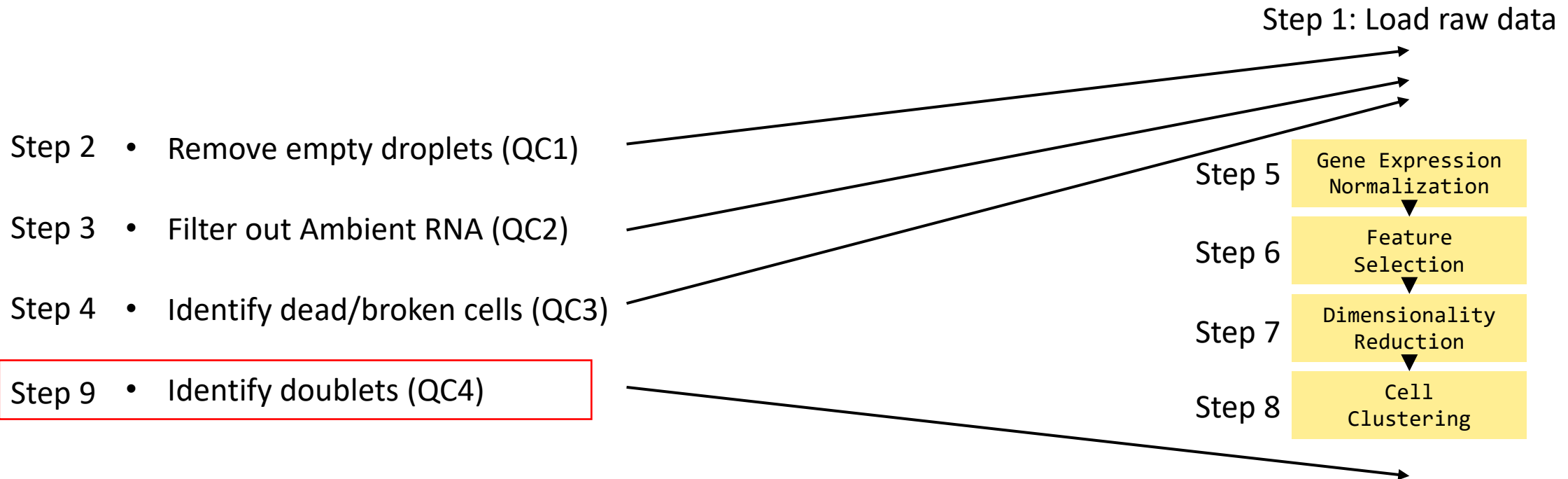
Large, coarse clusters

Moderate clusters (default-ish)

Smaller, finer clusters

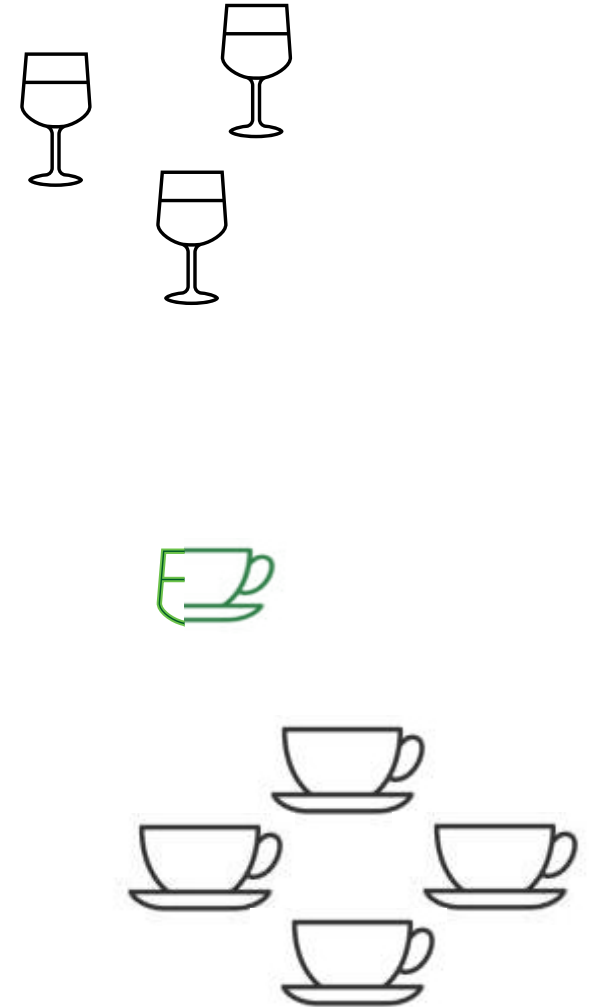
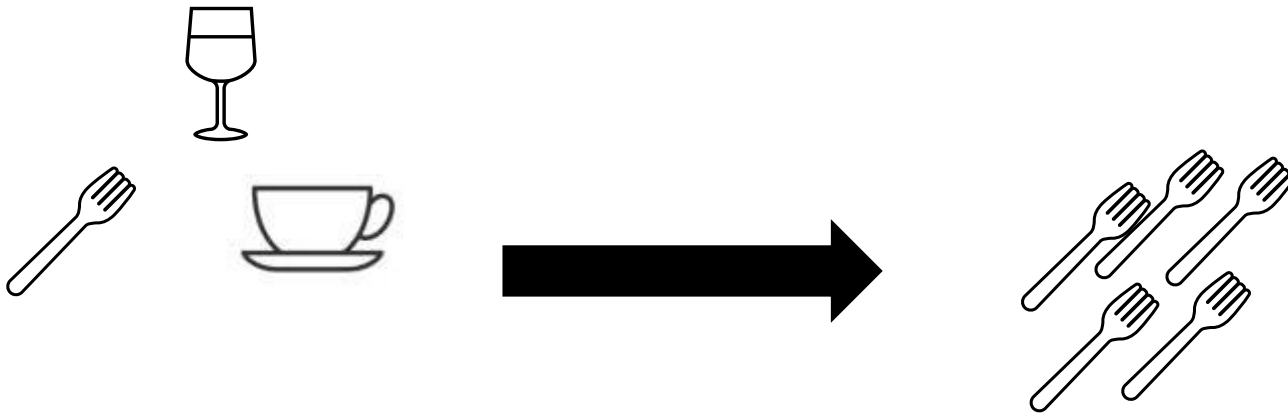
Many small clusters (may over-split)

Step 9: Identify Doublets (QC4)



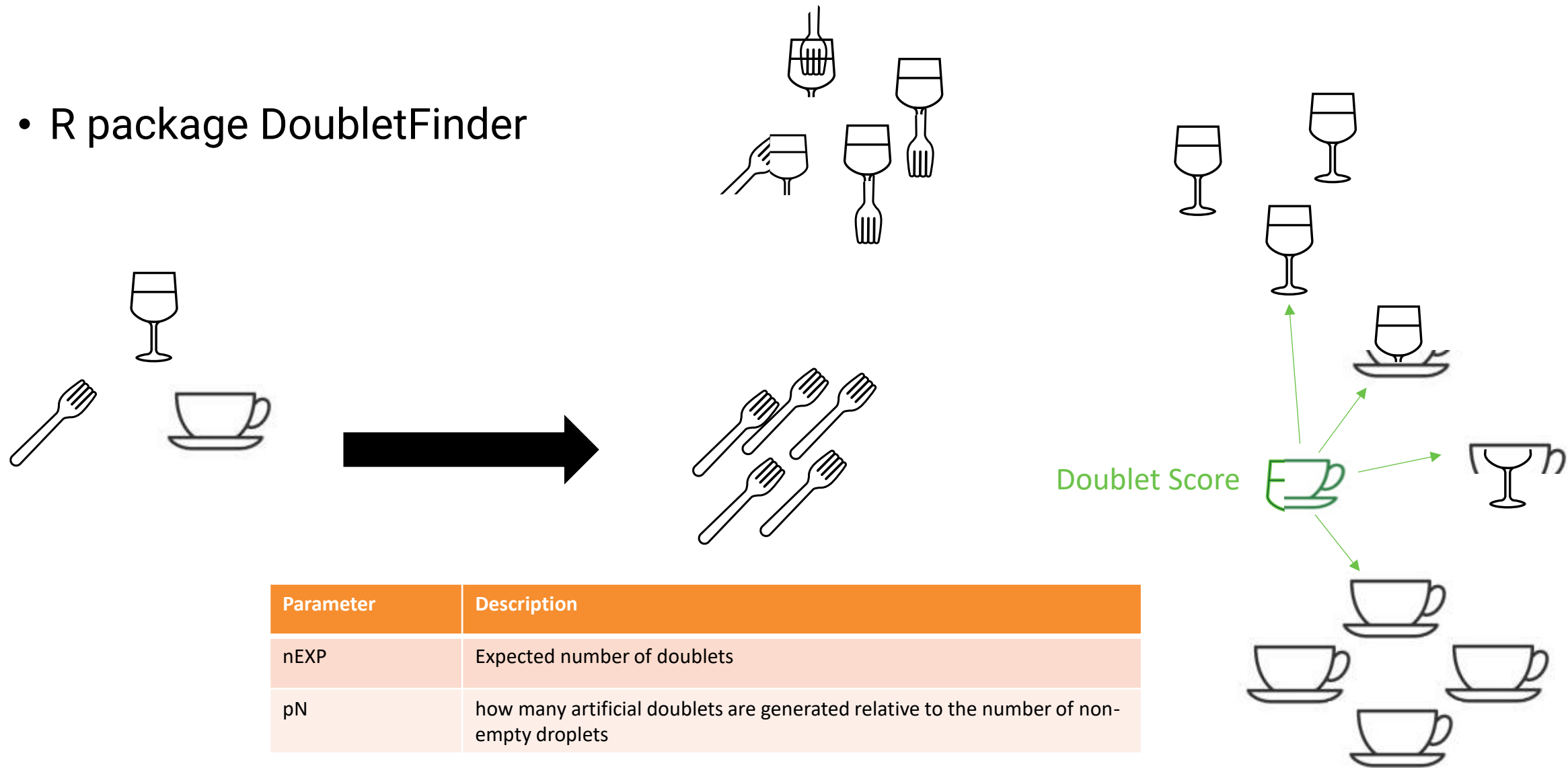
Step 9: Identify the doublets (QC4)

- R package DoubletFinder



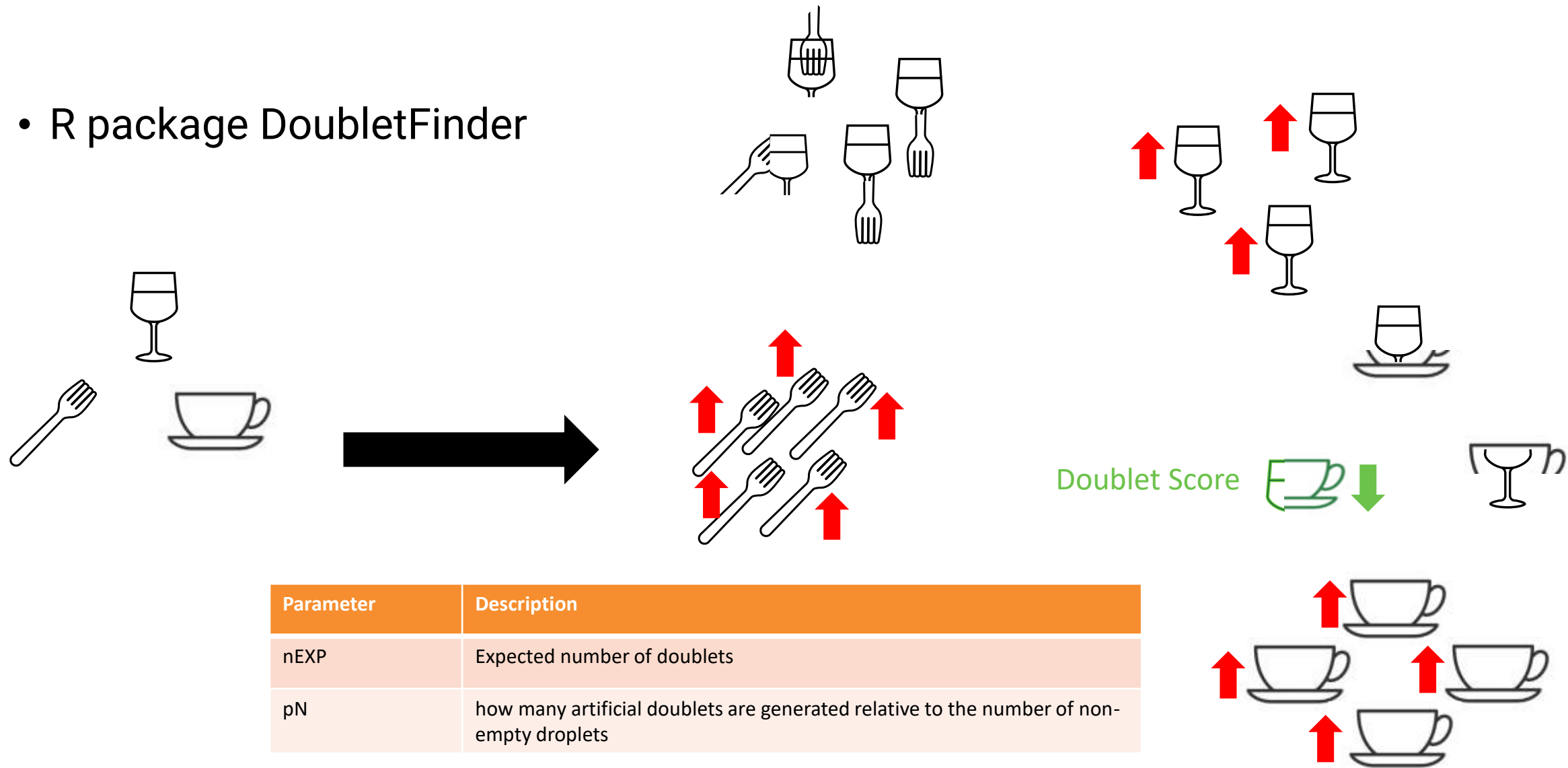
Step 9: Identify the doublets (QC4)

- R package DoubletFinder



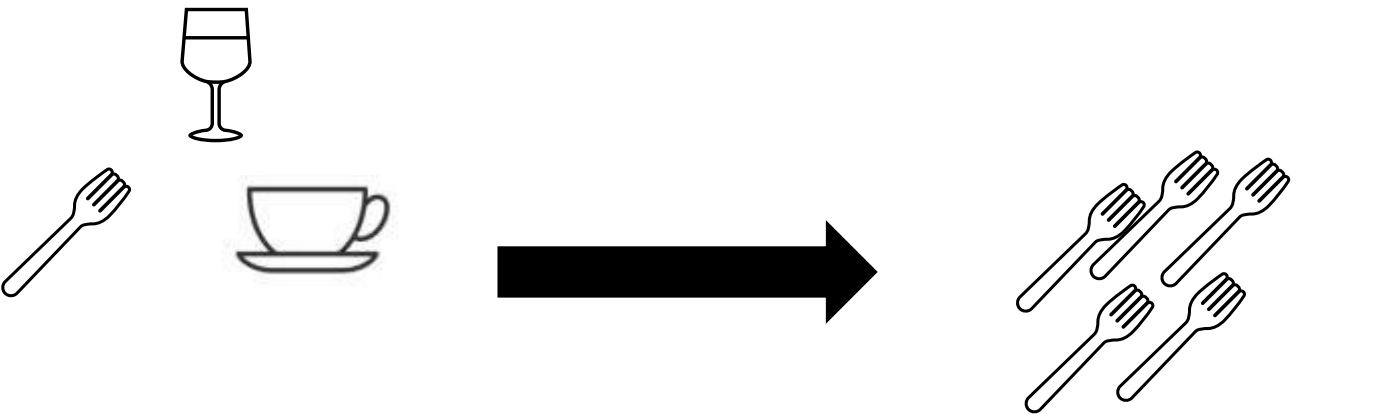
Step 9: Identify the doublets (QC4)

- R package DoubletFinder



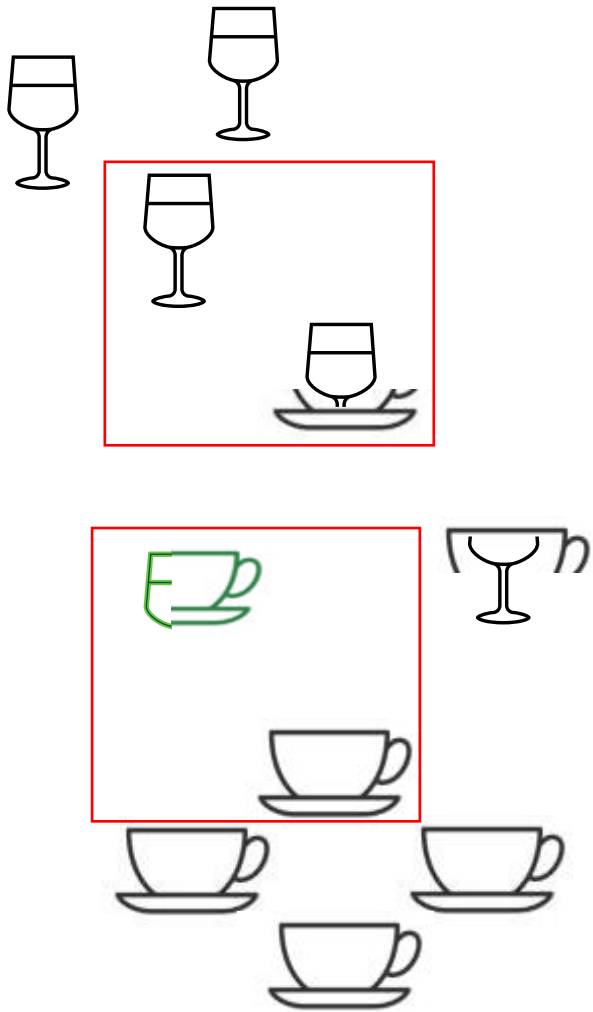
Step 9: Identify the doublets (QC4)

- R package DoubletFinder



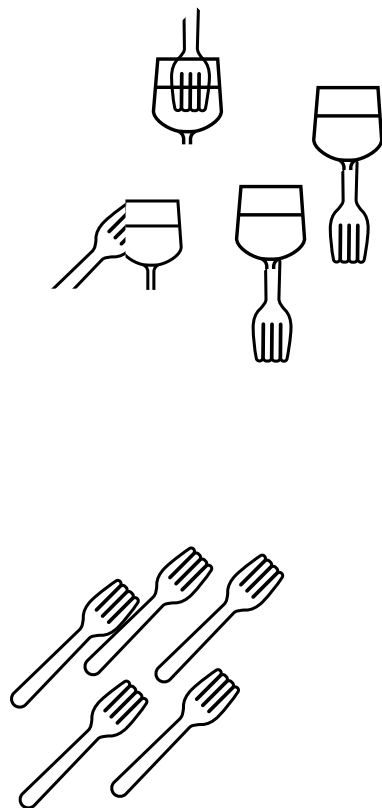
Parameter	Description
nEXP	Expected number of doublets
pN	how many artificial doublets are generated relative to the number of non-empty droplets
pK	the neighborhood size parameter, which determines for a droplet, how many nearby droplets are included to calculate its doublet score

pK is too small



Step 9: Identify the doublets (QC4)

- R package DoubletFinder



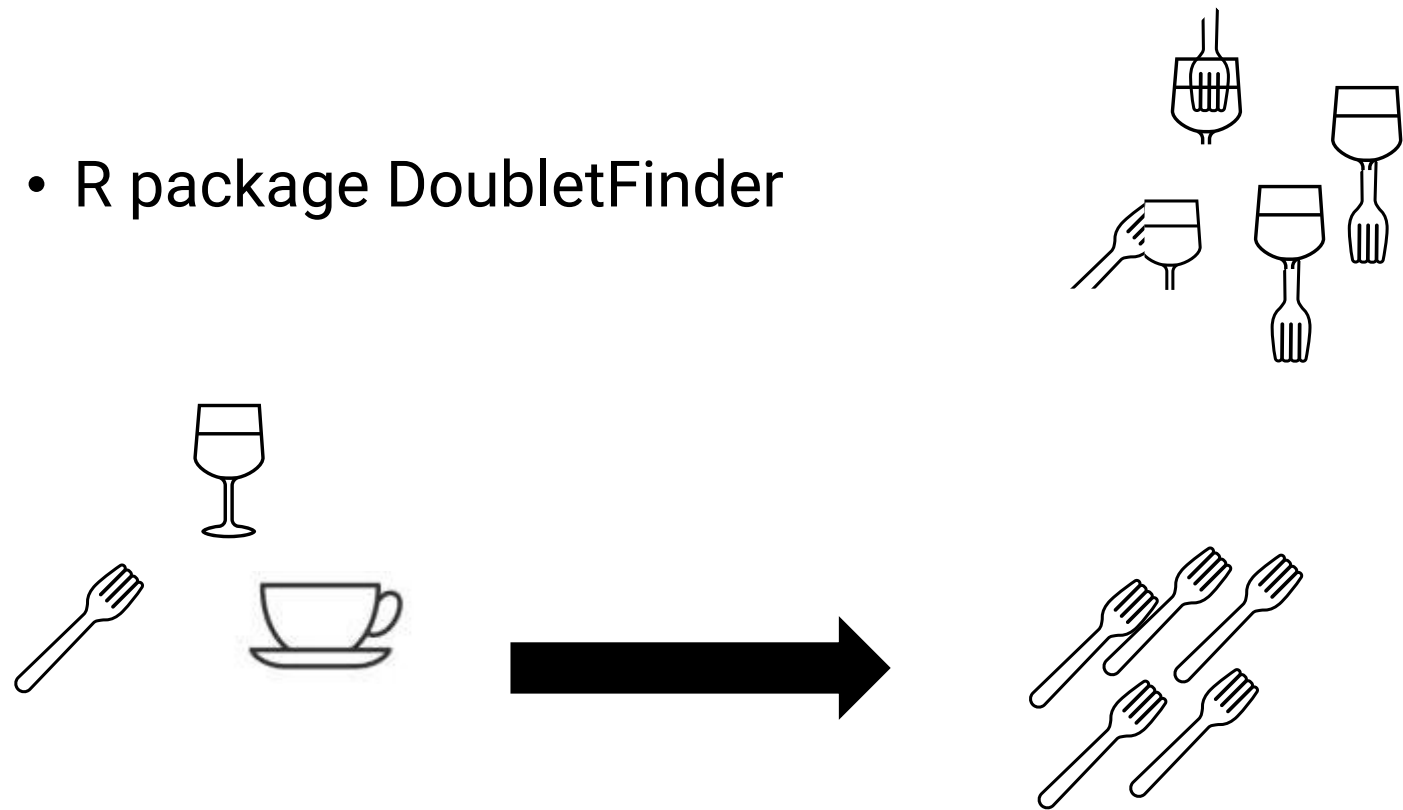
pK is too large



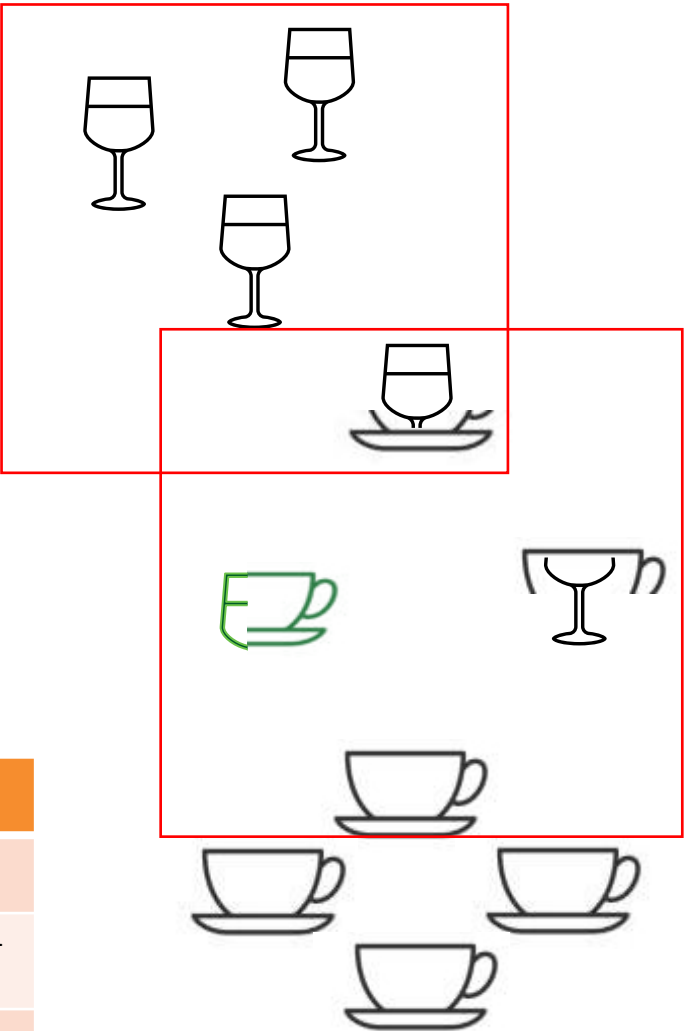
Parameter	Description
nEXP	Expected number of doublets
pN	how many artificial doublets are generated relative to the number of non-empty droplets
pK	the neighborhood size parameter, which determines for a droplet, how many nearby droplets are included to calculate its doublet score

Step 9: Identify the doublets (QC4)

- R package DoubletFinder



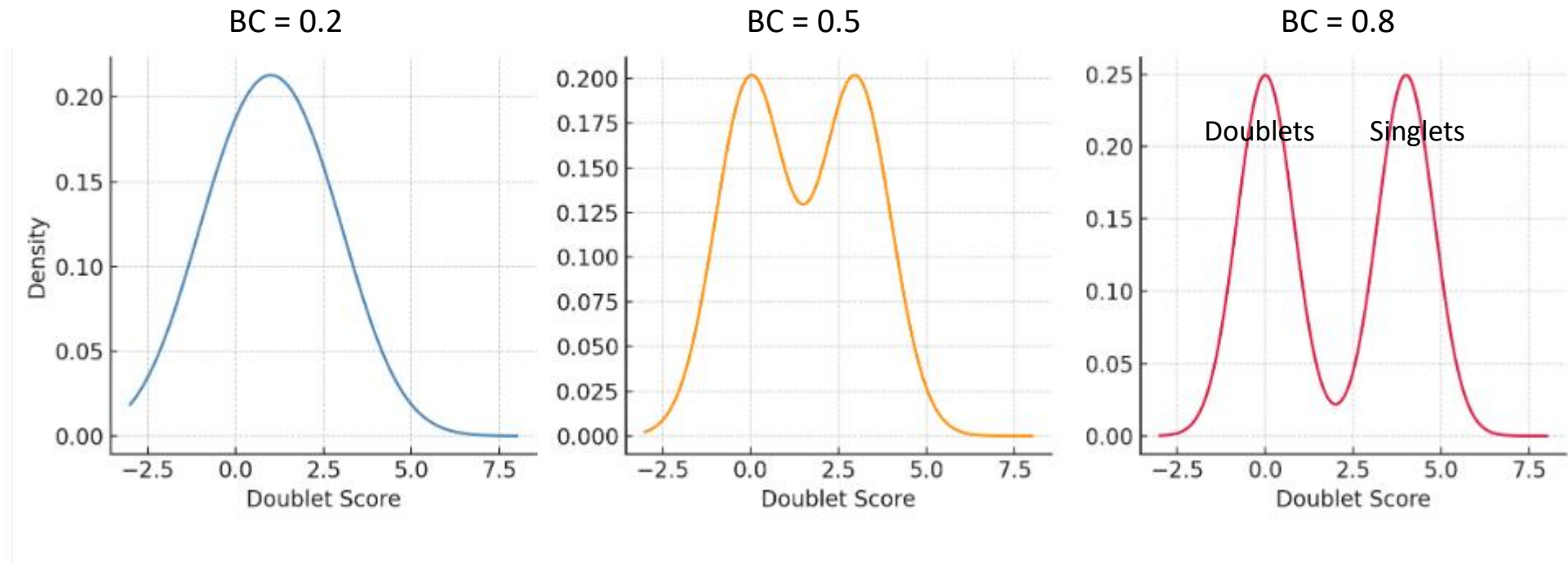
pK is just right



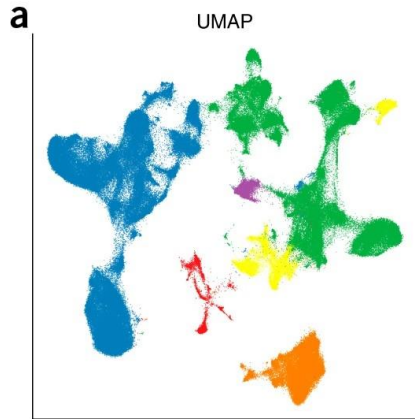
Parameter	Description
nEXP	Expected number of doublets
pN	how many artificial doublets are generated relative to the number of non-empty droplets
pK	the neighborhood size parameter, which determines for a droplet, how many nearby droplets are included to calculate its doublet score
BC	Bimodality Coefficient

Step 9: Identify the doublets (QC4)

- A series of pK are tested. The best pK is the one that generates highest BC.



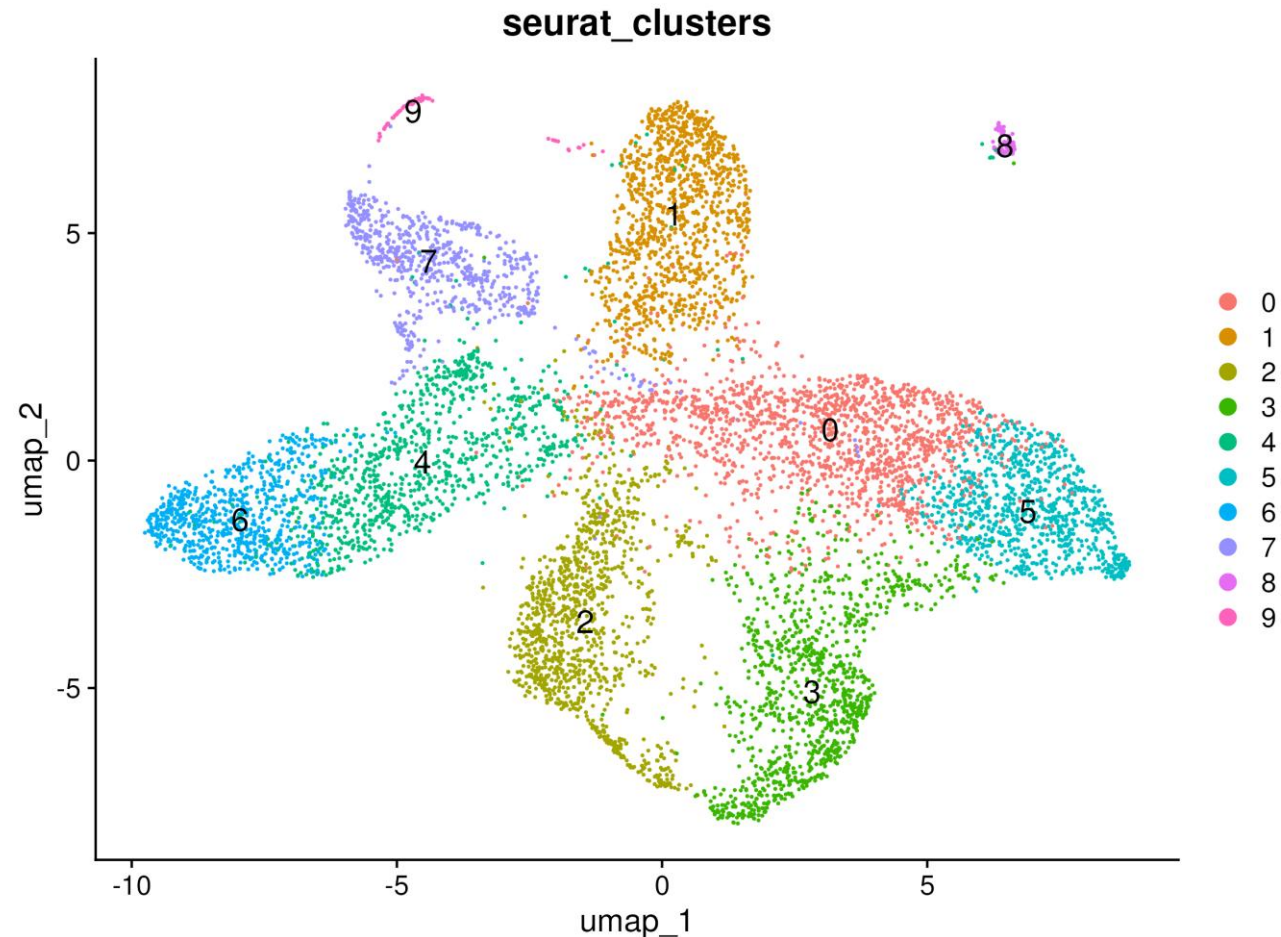
Cell Type Annotation



Etienne Becht et al. Nature Biotechnology. 37: 38–44 (2019)
 Aaron J. Wilk et al. Nature Biotechnology. 42: 470–483 (2024)
 Shobana V. Stassen et al. Nature Communications. 12: 5528 (2021)

Step 11: Visualization of Cell Clusters

```
seur_filtered <- RunUMAP(seur_filtered, dims = 1:PCs)
picture <- DimPlot(seur_filtered, reduction = "umap", group.by = "seurat_clusters", label = TRUE, label.size = 5)
ggsave("umap_cluster_plot.png", plot = p, width = 8, height = 6, dpi = 300)
```



Step 12: Marker Gene Identification

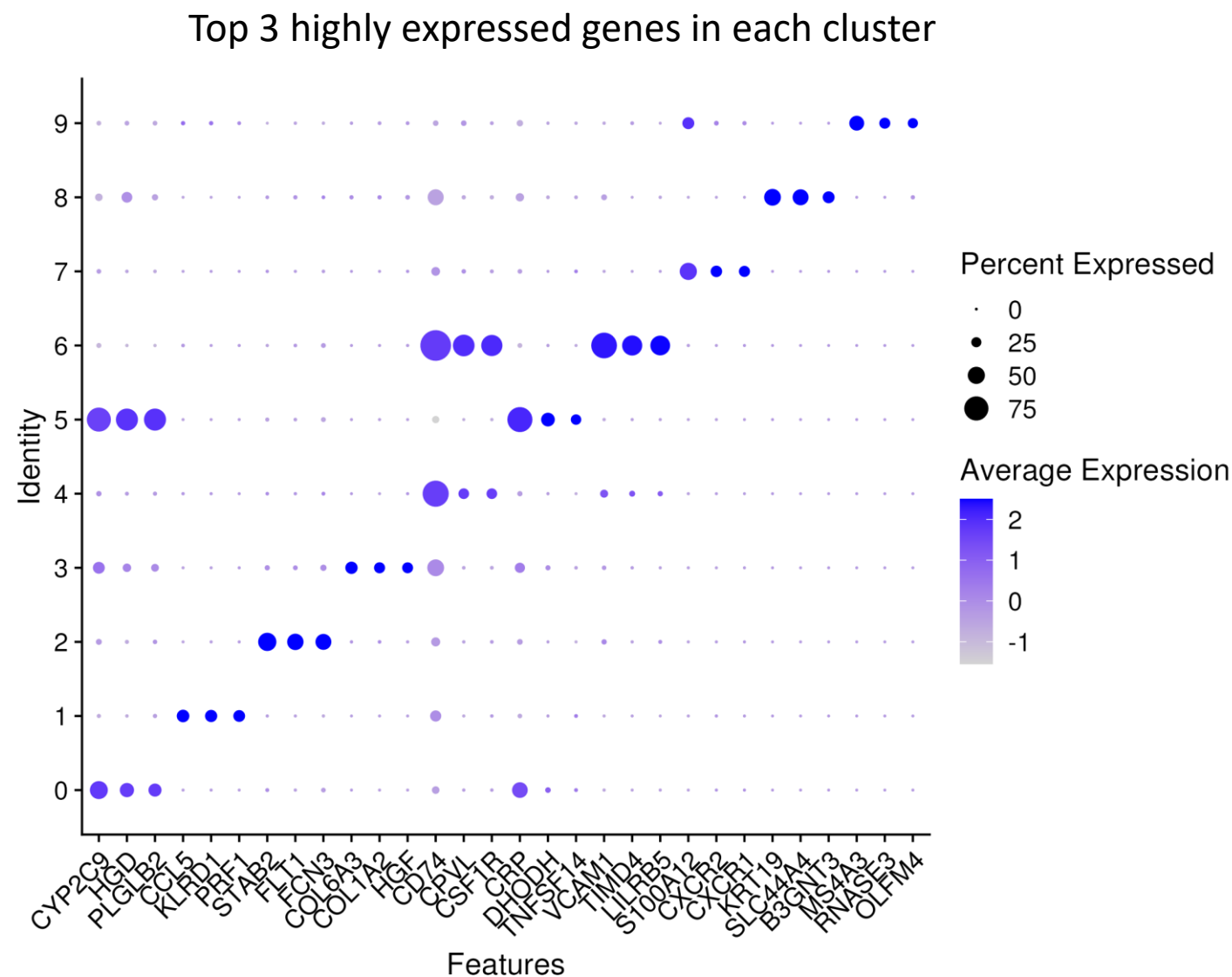
```
markers <- FindAllMarkers(seur_filtered, only.pos = TRUE, min.pct = 0.25, logfc.threshold = 0.25)
```

- min.pct = 0.25 Only test genes expressed in $\geq 25\%$ of cells in either cluster
- Logfc.threshold = 0.25 Only report genes with > 1.2 -fold change in average expression ($\log_2 X \geq 0.25$)

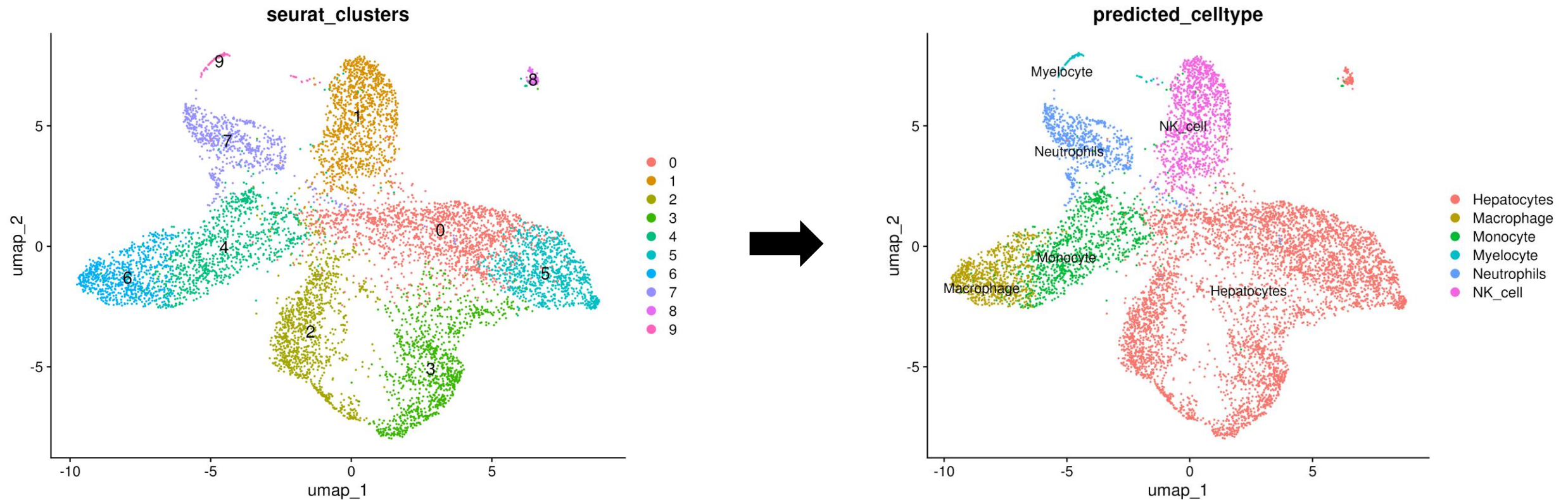
Situation	Suggested Change
You have rare or small clusters	Lower min.pct to 0.1
You're interested in subtle expression differences*	Lower logfc.threshold to 0.1 or 0.15
You're doing exploratory marker discovery	Lower both slightly to get more candidates

* Early development stage, immune cell activation, autoimmune disease, drug responses/resistance etc

Step 12: Marker Gene Identification



Step 13: Automated Cell-Type Annotation by SingleR



Keep In Mind:

- A good understanding of your sample is essential
 - How was it prepared?
 - What are the expected cell types?
 - Are there canonical marker genes?
- A good understanding of the methodology can help you:
 - Optimize the parameters
 - Assess your results
 - Develop new methods
- There are multiple options/tools for each step. Each has pros & cons with different focus and strength. When choosing the tools, you may want to ask:
 - Does it do a good job with your sample and project?
 - Is it easy to integrated into my pipeline?
 - Does it need customization? If so, does it worth my time?

Question?

Thank you!