Introduction to Single-cell RNA-seq Analysis - 1

by Jiajia Li

Research School of Biology

30 July 2025

Australian National University

About the trainer – Jiajia Li

Bioinformatics IT Support Officer at Research School of Biology

- Master of Data Science (AI and Computational Modelling)
  - University of Canberra, Australia
- Bachelor of Engineering (Bioengineering)
  - Henan University, China

Past Workshops:

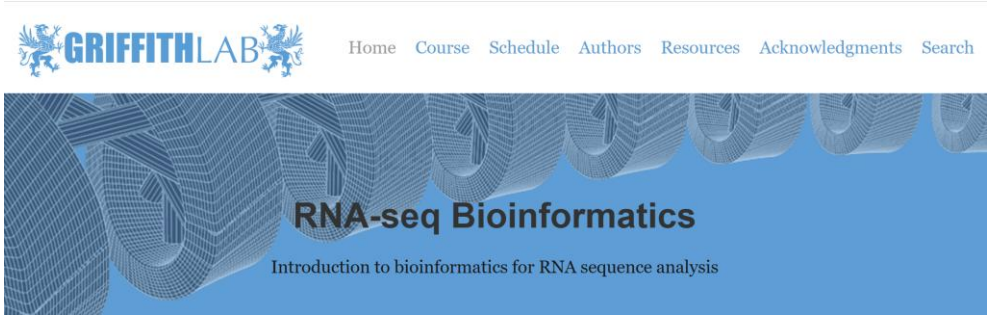
- Introduction to Linux and Variant Calling
- Introduction to Python
- Data Visualisation with Python
- Machine Learning with Python
- Introduction to Git and GitHub
- Introduction to NCI Gadi Supercomputer
- Introduction to Snakemake Workflow Language
- Data Management and Reproducible Research
- Introduction to Bulk RNA-seq Analysis

If you are interested in any of these, I can send you links for recordings.

Learning Objectives of Today

- Learn about single cell technology
- Learn about the experiment and data
- 10x Genomics Chromium Single Cell Platform
- Read in data and create Seurat object
- Quality control
- Filtering

Reference



https://rnabio.org/

This is a great resource from the Griffith Lab at the Washington University. I developed this workshop based on their material. Their course has a lot of useful information from theory to analysis and interpretation. My workshop is a simplified version of it.

It is a good resource to look at if you want a more comprehensive understanding of RNA-sequencing techniques.

https://www.nature.com/articles/s41576-019-0150-2 This is a great paper for introductory level of RNA sequencing.

What is single cell RNA sequencing?

Single cell RNA sequencing (scRNA-seq) measures **whole transcriptome gene expression in individual cells**, offering a detailed view of how cells function and interact within complex tissues.

Tracing individual transcripts back to their cells of origin enables researchers to pinpoint unique gene expression profiles in highly heterogenous samples, ultimately **revealing rare cell types and dynamic cell states** that other methods often miss.

For example, a researcher using bulk RNA-seq, which averages gene expression across many cells, to analyse a deadly tumour could miss a small population of treatment-resistant cells, potentially overlooking a putative drug target.

What can single cell do for my research?

- Identify disease biomarkers and mechanisms.
- Understand cell-specific responses in heterogeneous populations.
- Dynamics of cellular processes.
- Cell-to-cell interactions.
- Identify novel cell types and states.

The experiment

The data are generated from a **murine bladder cancer model system** with the following characteristics:

- Mice were exposed to a carcinogen 4-hydroxybutyl (butyl) nitrosamine (BBN) - this can induce bladder cancer development**,** via drinking water.

- Tumours developed and were isolated to create cell lines. One of these lines known as "**MCB6C**" was the source of the scRNA data used here.

- MCB6C cells were used to create tumours in mice that could be studied for response to immunotherapies.

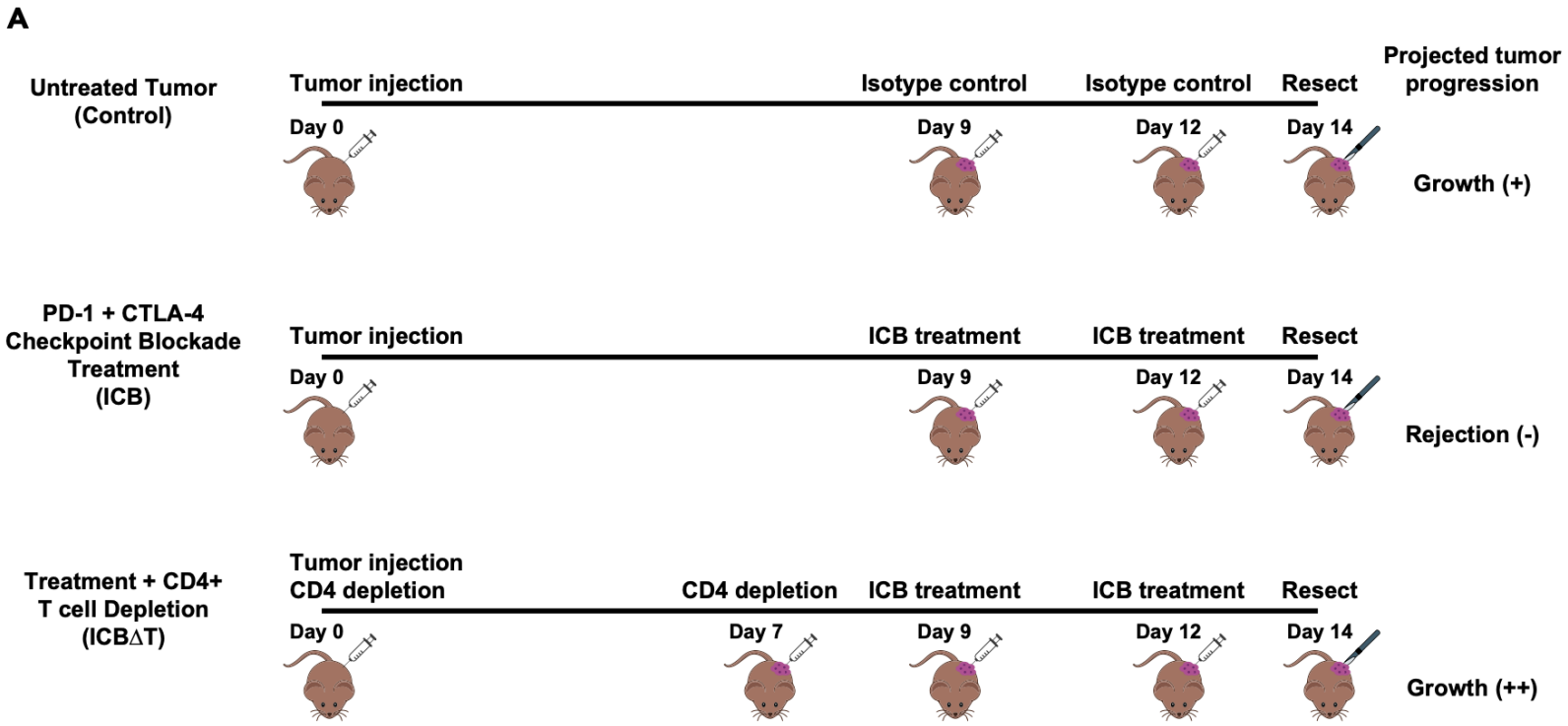
Experimental Details

- Each biological sample consists of bulk tumour samples obtained from three MCB6C tumours that were resected and pooled.

- Mice with MCB6C tumours were subjected to one of three conditions:
  - Control
  - Treatment with **immune checkpoint blockade** (ICB) - a method that can help T cells attack cancer cells
  - Treatment with ICB after **depletion of CD4 cells** (ICB-dT) - a type of T cells

- The tumours grow aggressively without treatment.
- Are rejected with ICB treatment.
- Grow even more aggressively if CD4 cells are depleted.

Experimental Details



Immune checkpoint blockade (ICB)

Immune checkpoints are a normal part of the immune system. Their role is to prevent an immune response from being so strong that it destroys healthy cells in the body.

Immune checkpoints engage when proteins on the surface of T cells recognise and bind to partner proteins on other cells, such as some **tumour cells**. These proteins are called **immune checkpoint proteins**.

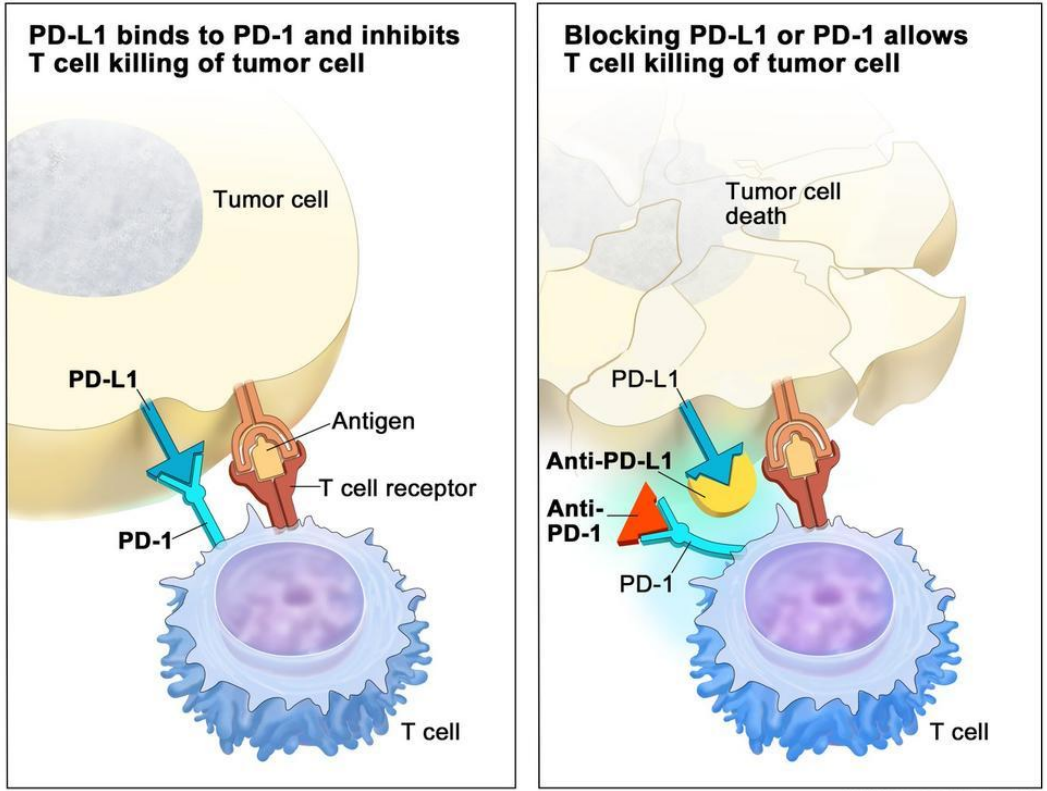
When the checkpoint and partner proteins bind together, they send an **"off"** signal to the T cells. This can prevent the immune system from destroying the cancer.

Immune checkpoint blockade (ICB)

Immunotherapy drugs called **immune checkpoint inhibitors** work by blocking checkpoint proteins from binding with their partner proteins.

This prevents the "off" signal from being sent, allowing the T cells to kill cancer cells.

CTLA-4 and **PD-1** are immune checkpoint proteins.

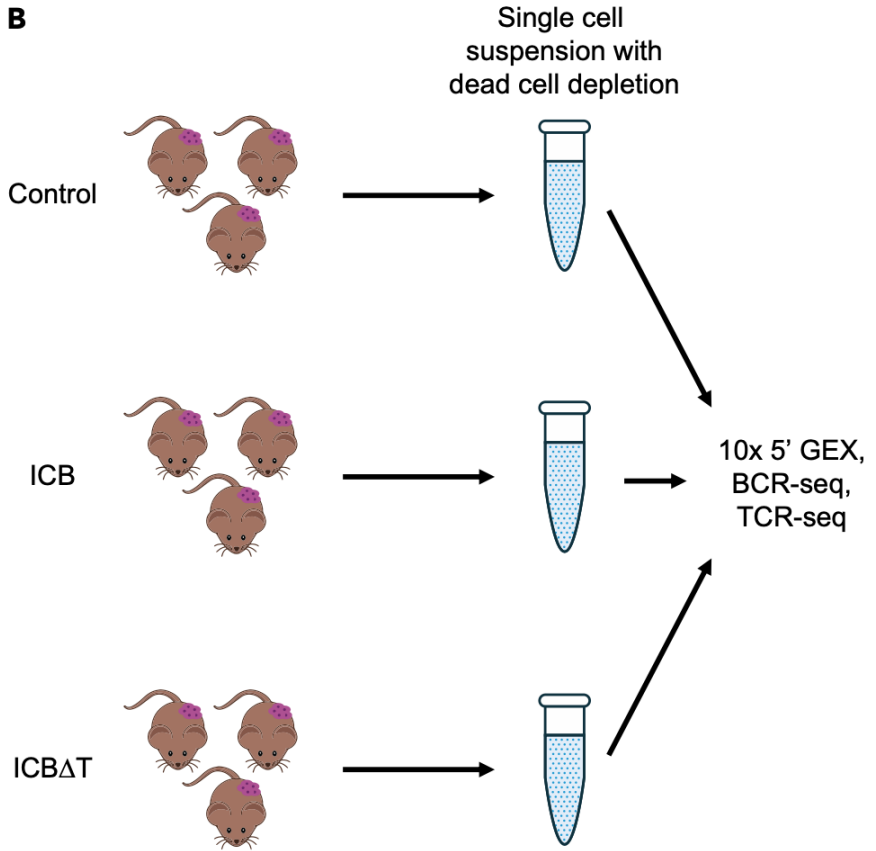


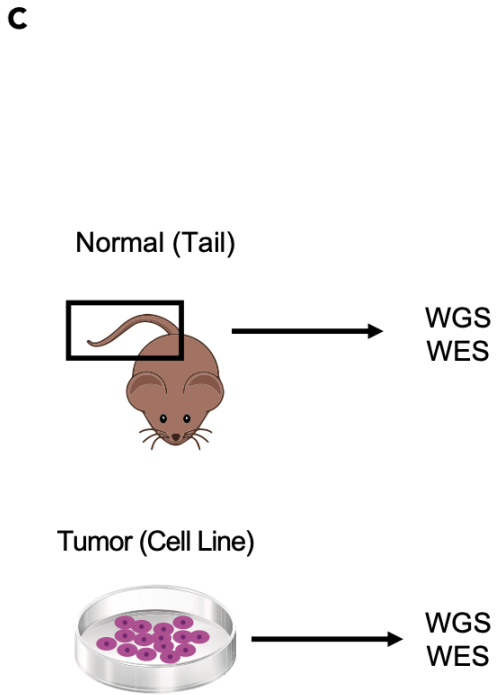
Data Generation

- From pooled NCB6C tumours, single cell suspensions were obtained and subjected to dead cell removal, but no other sorting or filtering.
- A target of 10,000 cells was submitted for 10x library creation using the **5' (V2) Kit**.
- The same libraries were subjected to **10x Genomics V(D)J B cell receptor (BCR) and T cell receptor (TCR)** enrichment and sequencing.
- A total of 8.3 billion sequence reads were generated and 64,049 single cells identified.
  - A subset of these will be analysed here.

- Bulk DNA of the MCB6C system was also subjected to exome and whole genome sequencing. These data were compared to paired data from bulk genomic DNA obtained from the tail of the mouse originally used to establish the MCB6C line.
- Bulk RNA-seq data were also generated from MCB6C tumours and a culture of the MCB6C line.

Data Generation





V(D)J Recombination

V(D)J recombination (variable-diversity-joining rearrangement) is the mechanism of somatic recombination that occurs only in developing lymphocytes during the early stages of T and B cell maturation. It results in the **highly diverse repertoire** of **antibodies/immunoglobulins** and **T cell receptors** found in B cells and T cells, respectively.

V(D)J recombination in mammals occurs in the primary lymphoid organs (bone marrow for B cells and thymus for T cells) and in a **nearly random fashion** rearranges variable (V), joining (J), and in some cases, diversity (D) gene segments.

The process ultimately results in **novel amino acid sequences** in the antigen-binding regions of immunoglobulins and TCRs that allow for the **recognition of antigens** from nearly all pathogens including bacteria, viruses, parasites, and worms as well as "altered self cells" as seen in cancer.

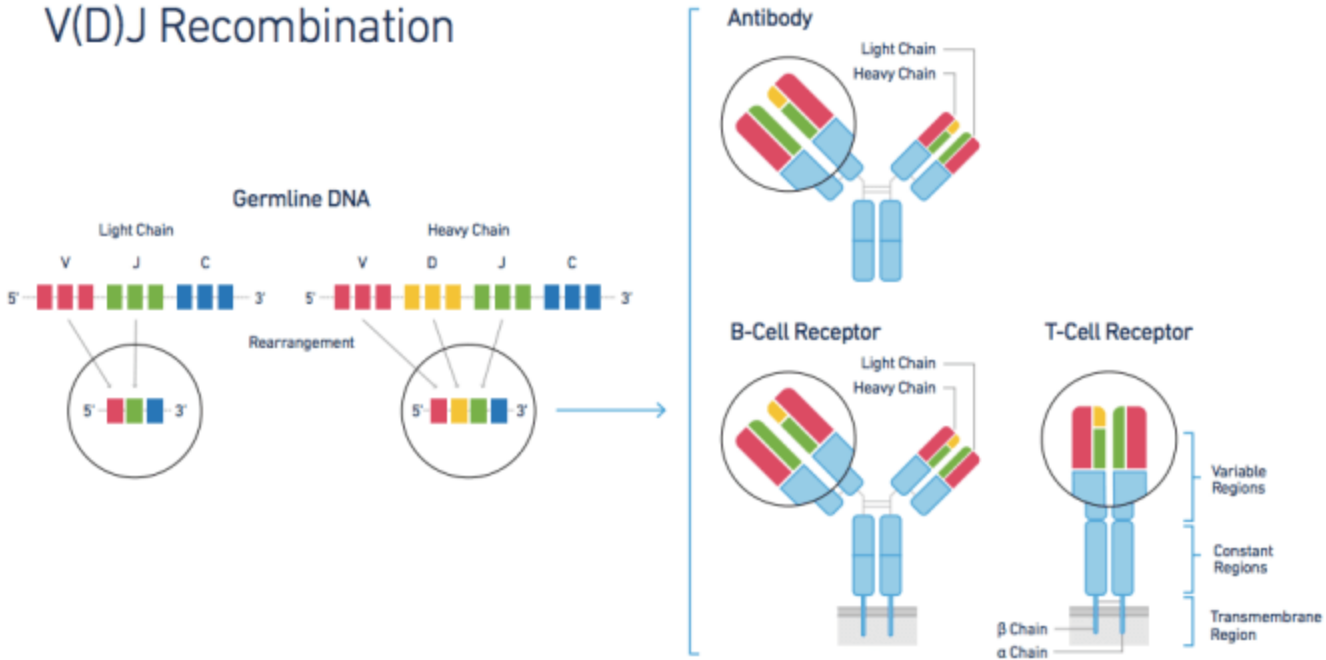
V(D)J Recombination

In this workshop, we will learn the single-cell gene expression analysis pipeline first.

Then, if we have time, we will analyse the VDJ sequencing as well.

We will revisit this topic and explain in detail about VDJ sequencing later.

V(D)J Recombination



Description of samples/replicates and QC reports

In this analysis, we will focus on a simplified comparison of 2 biological conditions:

- treatment with checkpoint ("ICB")
- treatment with ICB after depletion of CD4 cells in the mice ("ICB-dT")

For each of these two conditions, we will have 3 replicates (of 5 total reported in the publication).

- Rep1_ICB: 4,179 cells, 1,759 median genes per cell
- Rep3_ICB: 6,486 cells, 1,645 median genes per cell
- Rep5_ICB: 3,006 cells, 1,163 median genes per cell
- Rep1_ICBdT: 4,024 cells, 2,096 median genes per cell
- Rep3_ICBdT: 5,665 cells, 1,735 median genes per cell
- Rep5_ICBdT: 6,074 cells, 1,336 median genes per cell

Input files for the analysis

We will not be running **Cell Ranger** ourselves and will instead be starting the scRNA analysis in R using matrix files from Cell Ranger and a few other input files briefly described here:

- Various publicly available reference annotation files (e.g., reference cell type signatures, chromosome coordinates, pathway gene sets, etc.).
- Filtered **feature barcode matrix files** (.h5) from Cell Ranger.
  - 6 total, one for each sample listed above
- V(D)J clonotype files for TCR and BCR.
- Somatic variants (SNVs/Indels) identified by analysis of the bulk tumour/normal exome data from MCB6C.
- Somatic copy number variants (CNVs) identified by analysis of the bulk tumour/normal WGS (whole genome sequencing) data for MCB6C.

Chromium Single Cell Platform

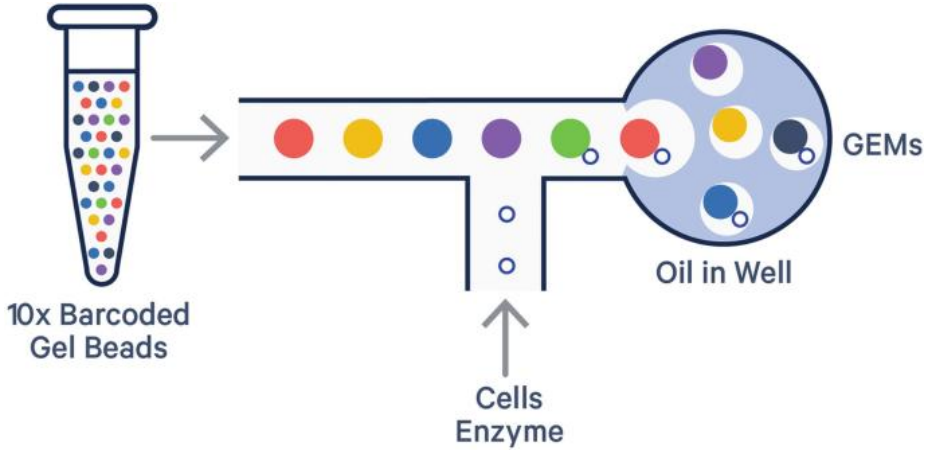
10x Genomics is a leading biotechnology company that develops and manufactures advanced instruments, reagents, and software for analysing biological systems at high resolution, particularly in the fields of **single-cell and spatial transcriptomics**.

Chromium Single Cell Platform: their most well-known technology.

It enables high-throughput single-cell analysis by encapsulating individual cells into nanolitre-sized **Gel Beads-in-emulsion (GEMs)**.

Each GEM acts as a miniature reaction vessel where cells are lysed, and their mRNA is reverse transcribed into cDNA and uniquely barcoded with a cell-specific **barcode** and a **unique molecular identifier (UMI).**

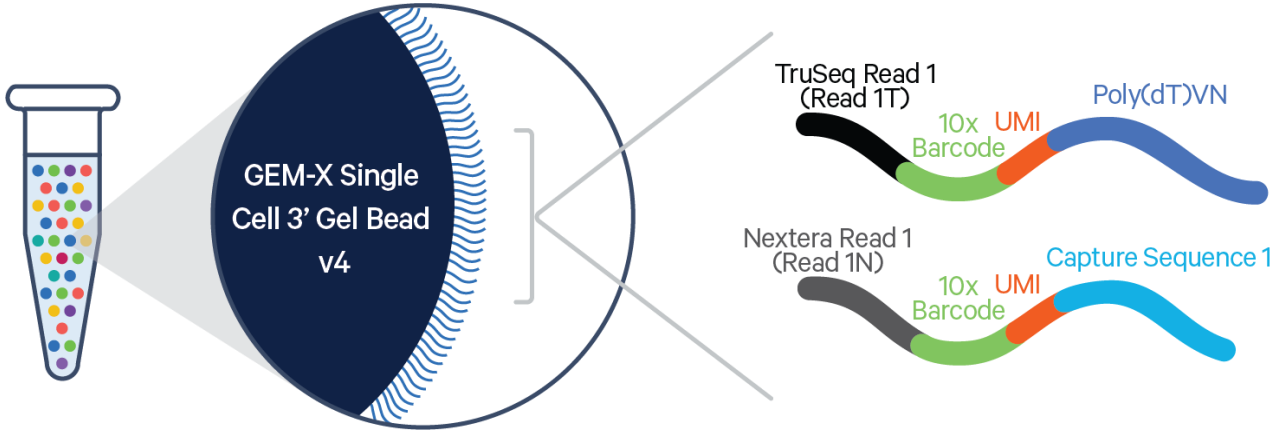
Gel Beads-in-emulsion (GEMs)



This is called the **Drop-Seq** technology, if you are interested you can read the paper.

To achieve single cell resolution, cells are delivered at a limiting dilution, such that the majority (90-99%) of generated GEMs contain no cell, while the remainder largely contain a single cell.

Gel Beads-in-emulsion (GEMs)



Each gel bead has **millions** of these **oligonucleotides** ("hairs").

Each oligo is unique, with the **same barcode** but a **unique molecular identifier (UMI).**

Each oligo will capture a sequence.

Cell Recovery Rate

The **cell recovery rate** is the percentage of cells from your starting sample that are successfully captured and produce high-quality data in a single-cell experiment.

For common droplet-based platforms like 10x Genomics, the expected cell recovery rate is typically between **30% and 70%**.

When you prepare you sample, you can estimate the number of cells by **Manual Hemocytometer Count** or **Cell Sorter**, that's the **targeted number of cells**.

But when running single-cell sequencing, **not all cells will be captured.**

mRNA molecule capture efficiency

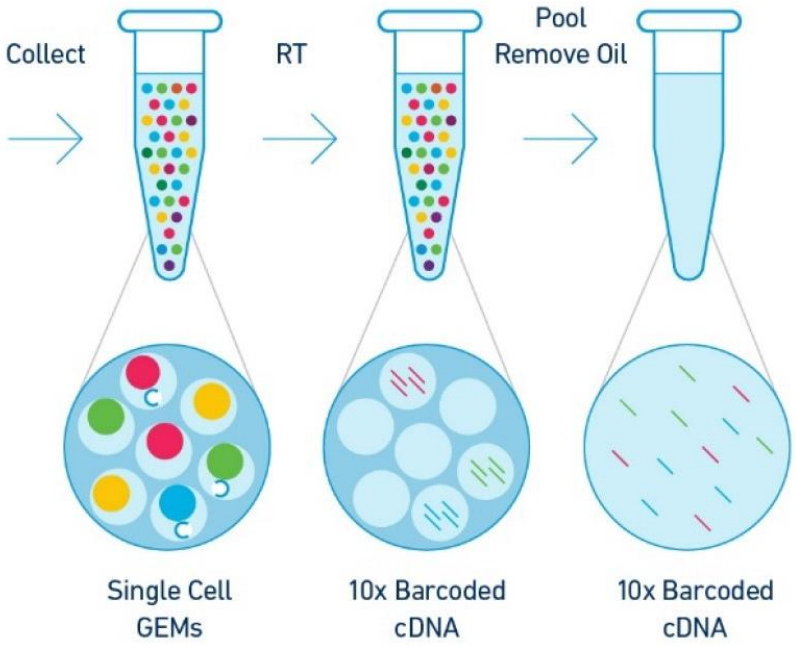
The success rate of capturing mRNA molecules in a single cell, known as **capture efficiency**, is **surprisingly low and highly variable**. For common droplet-based methods like 10x Genomics, it typically ranges from **5% to 25%**.

Might be higher than 25% with their newest technology.

This means that for every 100 mRNA molecules present in a cell, only 5 to 25 are successfully captured, tagged, and converted into data that can be sequenced. The vast majority of the cell's original mRNA is **not detected**.

Because of the **impact** of **recovery rate** and **capture efficiency**, we would always aim for **more cells** so we can have a great coverage of the transcriptome.

Gel Beads-in-emulsion (GEMs)

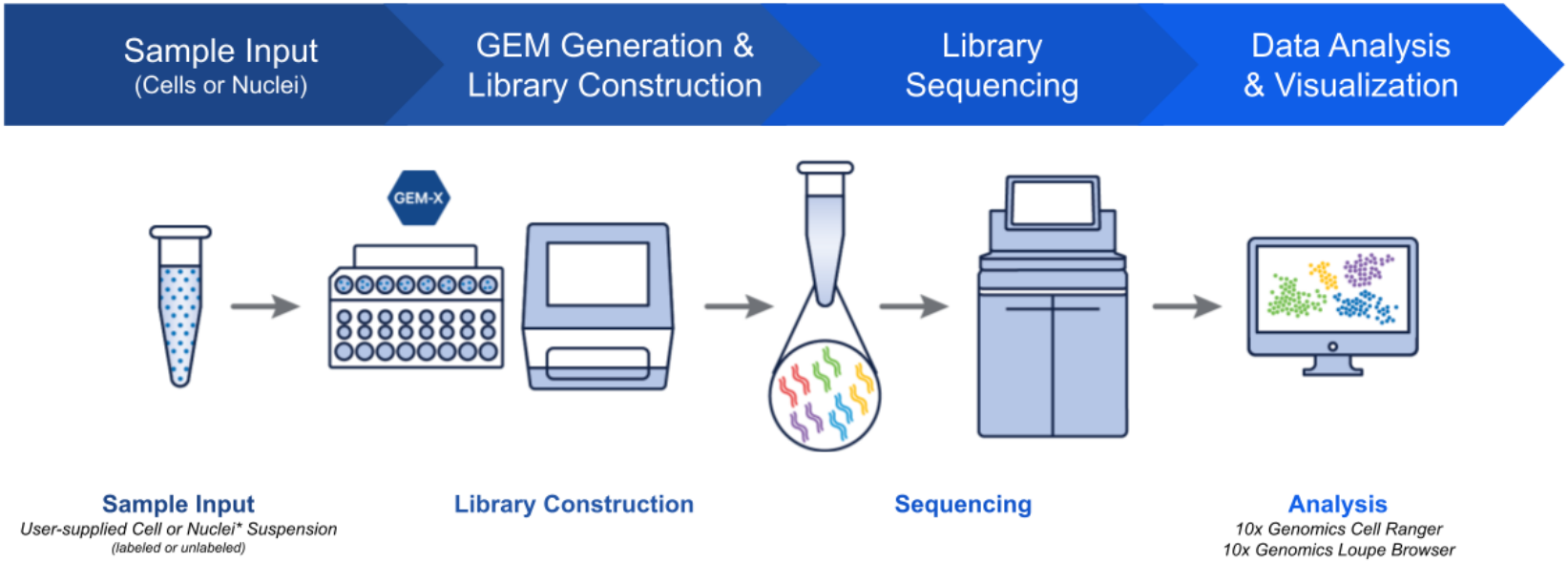


After GEMs are generated and RNA sequences are captured, it will be reverse transcribed to cDNA.

Then the droplet breaks, and cDNA fragments are pooled together for sequencing.

The sequence platform used are commonly **Illumina short-read** sequencing.

Chromium Single Cell Platform



Cell Ranger

If you have experience with Genome or RNA-seq Analysis before you would probably know after we get the raw sequence data from the service provider, we need to do a series of analysis to get the **variant calling results** or the **gene expression matrix**.

Cell Ranger is a free software that 10x Genomics offer to their customers to analyse the raw sequencing data. It is a set of analysis pipelines that process **Chromium Next GEM single cell data** to align reads, generate feature-barcode matrices, perform clustering and other secondary analysis.

In this workshop, we will use the **output matrix** from the pipeline `cellranger multi` as input to R to do the downstream analysis.

Cell Ranger

The reason we are not covering how to use Cell Ranger here is because single-cell data are normally huge, around 50GB per sample or even larger.

It is difficult to run on a personal computer.

Seurat

Seurat is an **R package** designed for the analysis and exploration of **single-cell RNA sequencing** data. It was originally developed by the Satija Lab at the New York Genome Centre.

Seurat provides a comprehensive suite of tools for:

- **Preprocessing & QC**: filtering low-quality cells, normalisation, detection of highly variable genes, scaling.
- **Dimensionality reduction**: PCA, UMAP, t-SNE, etc.
- **Clustering**: identifies groups of transcriptionally similar cells.
- **Differential expression**: finds marker genes that define each cluster.
- **Integration & batch correction:** combines data from multiple experiments or batches.
- **Visualisation**: high-quality plots of clusters, gene expression, etc.

Scanpy is a python equivalent software.

General steps to processing single-cell data

1. Create a Seurat object
2. Filter low-quality cells
3. Merge samples
4. Normalise counts
5. Find variable features
6. Scale data
7. Determine PCs for clustering
8. Clustering -> FindNeighbors, FindClusters, RunUMAP

Load in data & create a Seurat object

If you have followed my set up tutorial, you should have created an R project called "single-cell-workshop" and downloaded the six H5 files.

- Open this project in RStudio.
- Create a folder "data" and move all the H5 files into it.
- Create an R script called "single_cell_analysis.R".

Load in data & create a Seurat object

In the R script, type the following code to load the libraries we need.

```
library("Seurat")
library("ggplot2")
library("cowplot")
library("dplyr")
library("Matrix")
library("viridis")
library("gprofiler2")
```

Load in data & create a Seurat object

First, let's verify our working directory. When we create a project and open this project, our working directory would always be where the project folder is.

But let's double check.

```
getwd()
```

```
> getwd()
[1] "C:/Users/u1133824/Documents/single-cell-workshop"
> |
```

Loading in one sample

Now, we can read in our data matrix generated by Cell Ranger. Cell Ranger provides several output files, the one we are using is "**sample_filtered_feature_bc_matrix.h5**", which is a matrix of the **raw read counts** for all cells in a sample.

```
# Read in matrix file
Rep1_ICBdT_data <- Read10X_h5("data/Rep1_ICBdT-sample_filtered_feature_bc_matrix.h5")
```

The `Read10X_h5` function is only for reading the count matrix from 10X Cell Ranger hdf5 file.

If you have data from other platforms, please refer to the Seurat documentation to find which function to use.

Loading in one sample

You might come across an error.

```
> Rep1_ICBdT_data <- Read10X_h5("data/Rep1_ICBdT-sample_filtered_feature_bc_matrix.h5")

Error in Read10X_h5("data/Rep1_ICBdT-sample_filtered_feature_bc_matrix.h5") :
  Please install hdf5r to read HDF5 files
```

Show Traceback
Rerun with Debug

You can run the following to install `hdf5r`, then run the function again.

```
> install.packages("hdf5r")
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.5/hdf5r_1.3.12.zip'
Content type 'application/zip' length 4008989 bytes (3.8 MB)
downloaded 3.8 MB

package 'hdf5r' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\u1133824\AppData\Local\Temp\Rtmpk32qjX\downloaded_packages
>
```

Feature count matrix - hdf5

The hdf5 file is not human readable but can be visualised as a table where the cell barcodes are the column headers and the genes (or features) are the row names. Something like below.

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

`CreateSeuratObject()`

```r
# Create a Seurat object
Rep1_ICBdT_data_seurat_obj <- CreateSeuratObject(counts = Rep1_ICBdT_data,
                                                 project = "Rep1_ICBdT",
                                                 min.cells = 10,
                                                 min.features = 100)
```

The function is given 4 parameters.

- The count matrix, which is the `**Rep1_ICBdT_data**` variable we created earlier.
- Identification of this Seurat object, normally we use **sample name**.
- `**min.cells = 10**`, this can perform a filtering which only keeps genes are found in a minimum 10 cells.
- `**min.features = 100**`, keeping cells that have a minimum of 100 features.

Exercise

Please do the same for sample "**Rep1_ICB**".

Understanding the Seurat object

We can use the `print()` function to print out the information of our Seurat object.

```
> print(Rep1_ICB_data_seurat_obj)
An object of class Seurat
14423 features across 4169 samples within 1 assay
Active assay: RNA (14423 features, 0 variable features)
 1 layer present: counts
>
```

We have 14423 features which are our genes, 4169 samples which are our cells, and 1 assay which is our counts matrix.

Understanding the Seurat object

For a Seurat object, there may be multiple assays storing different information. To see the list of assays in your Seurat object. Use the `Assays()` function.

```
> Assays(Rep1_ICB_data_seurat_obj)
[1] "RNA"
>
```

For now, we only have one assay, which stores our counts matrix, it is called "RNA".

To access an assay, you can run:

```
> Rep1_ICB_data_seurat_obj[["RNA"]]
Assay (v5) data with 14423 features for 4169 cells
First 10 features:
 Mrpl15, Lypla1, Tcea1, Atp6v1h, Rb1cc1, 4732440D04Rik, Pcmtd1, Gm26901, Sntg1,
Rrs1
Layers:
 counts
```

Understanding the Seurat object

To view our counts matrix, we can run:

```
> Rep1_ICB_data_seurat_obj[["RNA"]]$counts
14423 x 4169 sparse Matrix of class "dgCMatrix"
   [[ suppressing 37 column names ‘AAACCTGAGCGGATCA-1’, ‘AAACCTGCAAGAGGCT-1’, ‘AAACCTGCATACTCTT-1’ ... ]]
   [[ suppressing 37 column names ‘AAACCTGAGCGGATCA-1’, ‘AAACCTGCAAGAGGCT-1’, ‘AAACCTGCATACTCTT-1’ ... ]]


Mrpl15        . . . 1 . . . . . . . 1 1 . 2 . . . . . . . . 1 1 1 . . .  5 . . . 1 1 . . ......
Lypla1        . . . 1 . . . . . . . . . . 1 1 . . . . . . . 2 1 . . 1 .  6 . . 1 . . . . ......
Tcea1         . . . . . . 2 . . . 1 . 1 . 1 . 2 1 1 . . . . . . . . . . 1 20 . . 2 . 2 . 1 ......
Atp6v1h       . 1 . . . . . . . . . . 1 . . 1 . . . . . . . . . 1 . 1 . .  2 . . . 2 1 . . ......
Rb1cc1        . . . . 2 . 1 . . . . . 3 . 1 1 . . . . . 1 . . 2 4 . . .  1 . . . 1 1 . . ......
4732440D04Rik . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  1 . . . . . . . ......
Pcmtd1        . . . . . 1 . . . . . . . . . 1 1 . . . . 1 . . 1 . . 1 .  3 . . . . 1 . . ......
Gm26901       . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  . . . . . . . . ......
Sntg1         . . . . . . . . . . . . . 1 . . . . . . . . . . . . . . . .  . . . . . 1 . . ......
Rrs1          . . . 1 . . . . . . . . . 1 . . . . 1 . . . . . . . . . . 1 1  1 . . . . . . . ......
Adhfe1        . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  . . . . . . . . ......
2610203C22Rik . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  . . . . . . . . ......
Mybl1         . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1 .  . . . . . . . . ......
Vcpip1        . . 1 . . . . . . 1 . 1 . . . . . . . . . . . . . . . . . .  1 . . . . . . . ......
```

Understanding the Seurat object

The Seurat object also has a metadata table `**meta.data**`.

This is a table with each row represents a cell, and each column represents an attribute of the cells.

```
> head(Rep1_ICB_data_seurat_obj@meta.data)
                   orig.ident nCount_RNA nFeature_RNA
AAACCTGAGCGGATCA-1   Rep1_ICB       5679         1758
AAACCTGCAAGAGGCT-1   Rep1_ICB       1397          496
AAACCTGCATACTCTT-1   Rep1_ICB       8463         2444
AAACCTGCATCATCCC-1   Rep1_ICB       4640         1721
AAACCTGGTTCGGGCT-1   Rep1_ICB       3810         1291
AAACCTGTCAGTCCCT-1   Rep1_ICB       7810         2079
```

- `**orig.ident**`: the sample this cell came from.
- `**nCount_RNA**`: the number of reads in that cell.
- `**nFeature_RNA**`: the number of genes in that cell.

Understanding the Seurat object

As we analyse our data, we will add more columns to `meta.data`.

The existing three columns is automatically populated when create the object.

You can also access the meta.data table in this way:

```
> head(Rep1_ICB_data_seurat_obj[[]])
                    orig.ident nCount_RNA nFeature_RNA
AAACCTGAGCGGATCA-1   Rep1_ICB       5679         1758
AAACCTGCAAGAGGCT-1   Rep1_ICB       1397          496
AAACCTGCATACTCTT-1   Rep1_ICB       8463         2444
AAACCTGCATCATCCC-1   Rep1_ICB       4640         1721
AAACCTGGTTCGGGCT-1   Rep1_ICB       3810         1291
AAACCTGTCAGTCCCT-1   Rep1_ICB       7810         2079
```

Understanding the Seurat object

To view a column of the metadata table, you could do:

```
> head(Rep1_ICB_data_seurat_obj@meta.data$orig.ident)
[1] Rep1_ICB Rep1_ICB Rep1_ICB Rep1_ICB Rep1_ICB Rep1_ICB
Levels: Rep1_ICB
>
```

Or

```
> head(Rep1_ICB_data_seurat_obj[['orig.ident']])
                   orig.ident
AAACCTGAGCGGATCA-1   Rep1_ICB
AAACCTGCAAGAGGCT-1   Rep1_ICB
AAACCTGCATACTCTT-1   Rep1_ICB
AAACCTGCATCATCCC-1   Rep1_ICB
AAACCTGGTTCGGGCT-1   Rep1_ICB
AAACCTGTCAGTCCCT-1   Rep1_ICB
```

Adding a column to metadata table

Let's start by adding the **percentage of mitochondrial genes**, this will be a useful number when we begin removing low-quality cells.

Name the new column as "**percent.mt**".

```
Rep1_ICB_data_seurat_obj[["percent.mt"]] <-
  PercentageFeatureSet(Rep1_ICB_data_seurat_obj, pattern = "^mt-", assay = "RNA")
```

The function `**PercentageFeatureSet()**` calculates the percentage of all counts that belong to a given set of features.

- `**pattern = "^mt-"**`: match features start with mt-.
- `**assay = "RNA"**`: calculate this on the assay RNA.

Adding a column to metadata table

Check the newly added column.

```
> head(Rep1_ICB_data_seurat_obj[[]])
                  orig.ident nCount_RNA nFeature_RNA percent.mt
AAACCTGAGCGGATCA-1   Rep1_ICB       5679         1758   2.341962
AAACCTGCAAGAGGCT-1   Rep1_ICB       1397          496  50.894775
AAACCTGCATACTCTT-1   Rep1_ICB       8463         2444   2.197802
AAACCTGCATCATCCC-1   Rep1_ICB       4640         1721   3.750000
AAACCTGGTTCGGGCT-1   Rep1_ICB       3810         1291   2.755906
AAACCTGTCAGTCCCT-1   Rep1_ICB       7810         2079   3.354673
> |
```

The **second cell** seems to have more mitochondrial RNAs than others.

What does it mean if a cell has higher proportion of mitochondrial RNAs?

High-proportion of mitochondrial RNAs

A cell has high mitochondrial RNAs usually means the cell is **stressed**, **damaged**, or **dying**.

In a healthy cell, the cytoplasm is packed with a large volume of messenger RNA (mRNA). Mitochondrial contribute only **a small fraction** of the total mRNA pool.

When a cell becomes stressed or its outer membrane is compromised (a common occurrence during sample preparation), the less protected cytoplasmic mRNA **leaks out**.

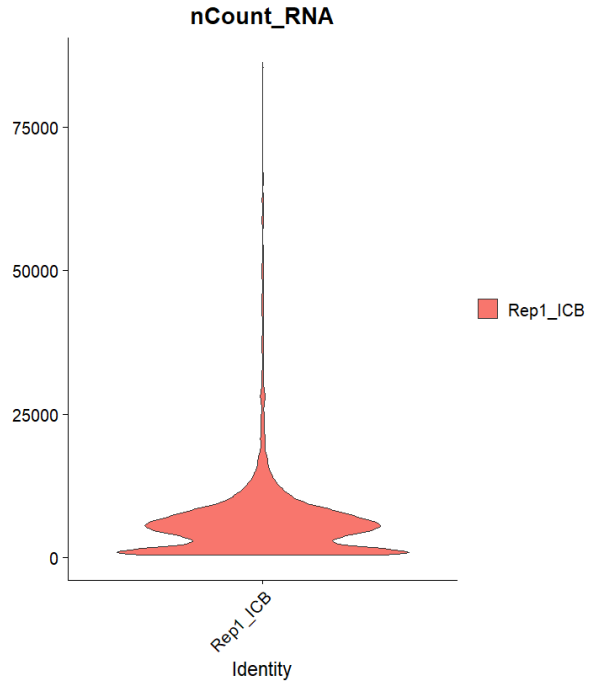
However, the mitochondria and their own RNA are **enclosed within their own membranes** are often retained.

Because of this, the percentage of mitochondrial RNA is one of the most important **quality control metrics** in single-cell RNA-seq analysis.

Quick look of counts, genes, mt-genes

We can use violin plots to look at the **number of reads**, **number of genes**, and the **percentage of mitochondrial genes** in each cell.

```
p1 <- VlnPlot(Rep1_ICB_data_seurat_obj,
              layer = "counts",
              features = c("nCount_RNA"),
              pt.size = 0)
p1
```



For the above code, we only look at the **number of reads**.

We can see there are **2 peaks**.

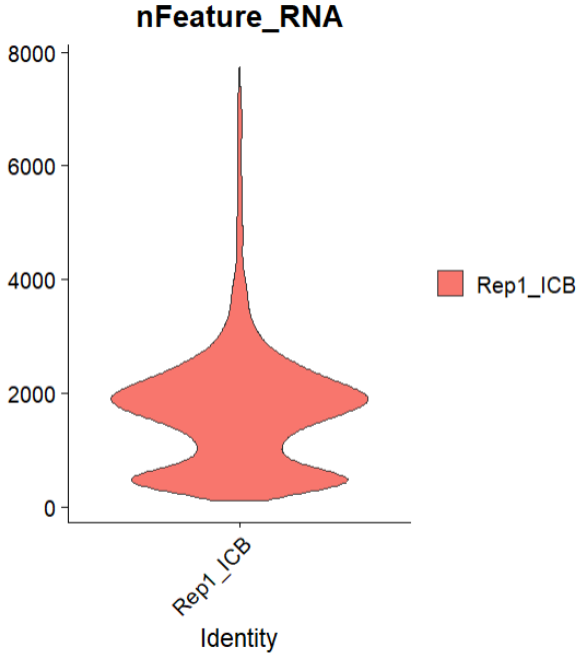
- One is close to 0
- One is around 6000?

Try change the `pt.size`.

Quick look of counts, genes, mt-genes

And we can change the `features =` option to look at other numbers.

```
p2 <- VlnPlot(Rep1_ICB_data_seurat_obj,
              layer = "counts",
              features = c("nFeature_RNA"),
              pt.size = 0)
p2
```



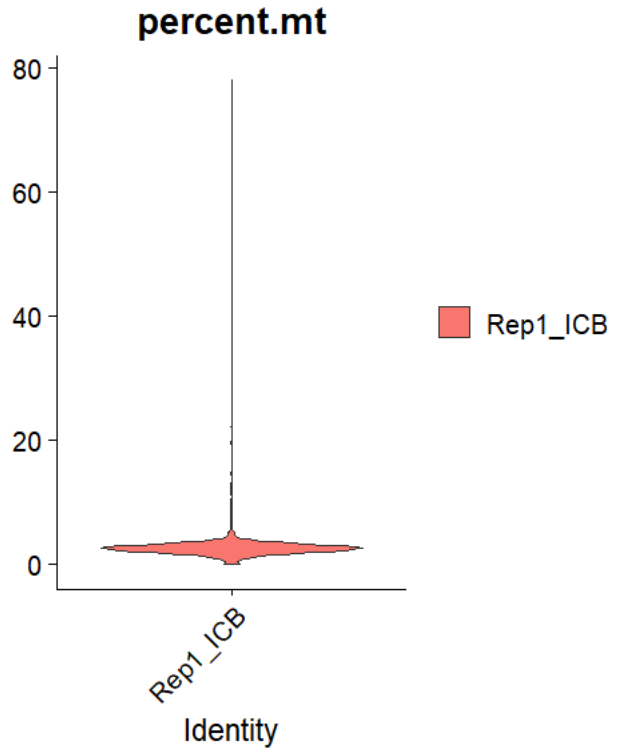
We can see there are **2 peaks**.

- One is around 500
- One is around 2000

Quick look of counts, genes, mt-genes

Exercise: do the same for column "percent.mt".

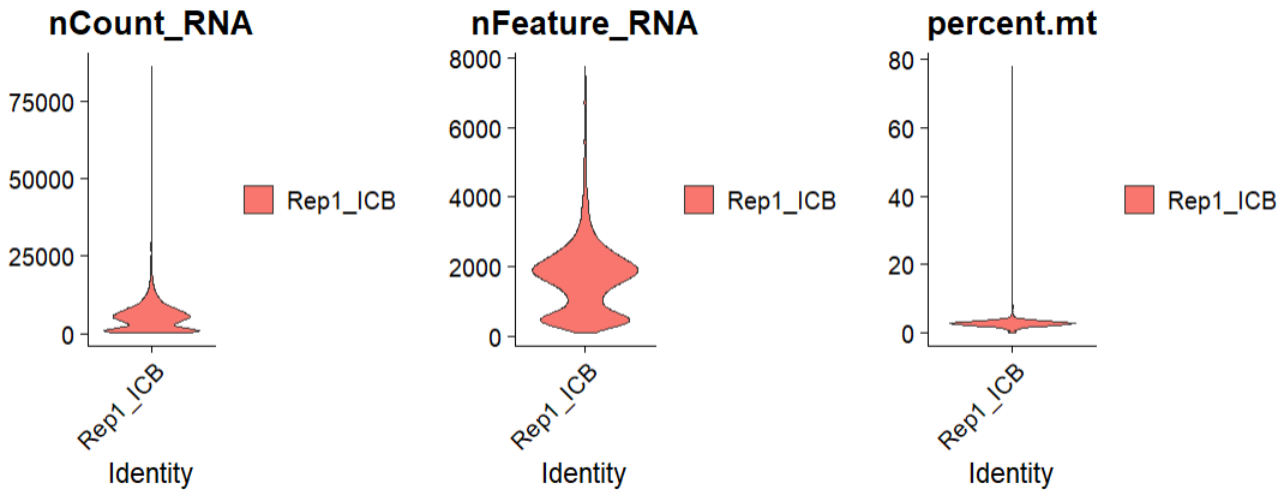
One peak around 3?



Combine 3 plots together

We can put the 3 plots side by side.

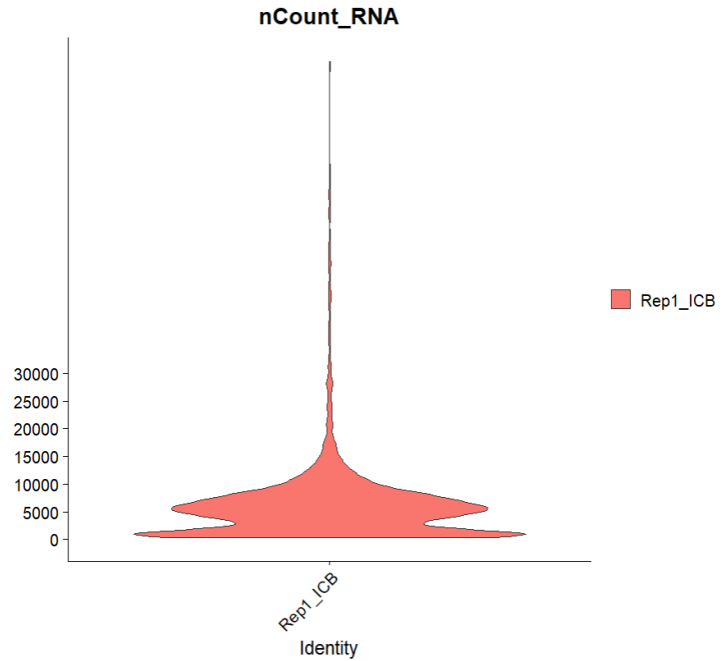
```
p <- plot_grid(p1, p2, p3, ncol = 3)
p
```



Add y-axis ticks

We can add more ticks to our y-axis, so we know the accurate numbers.

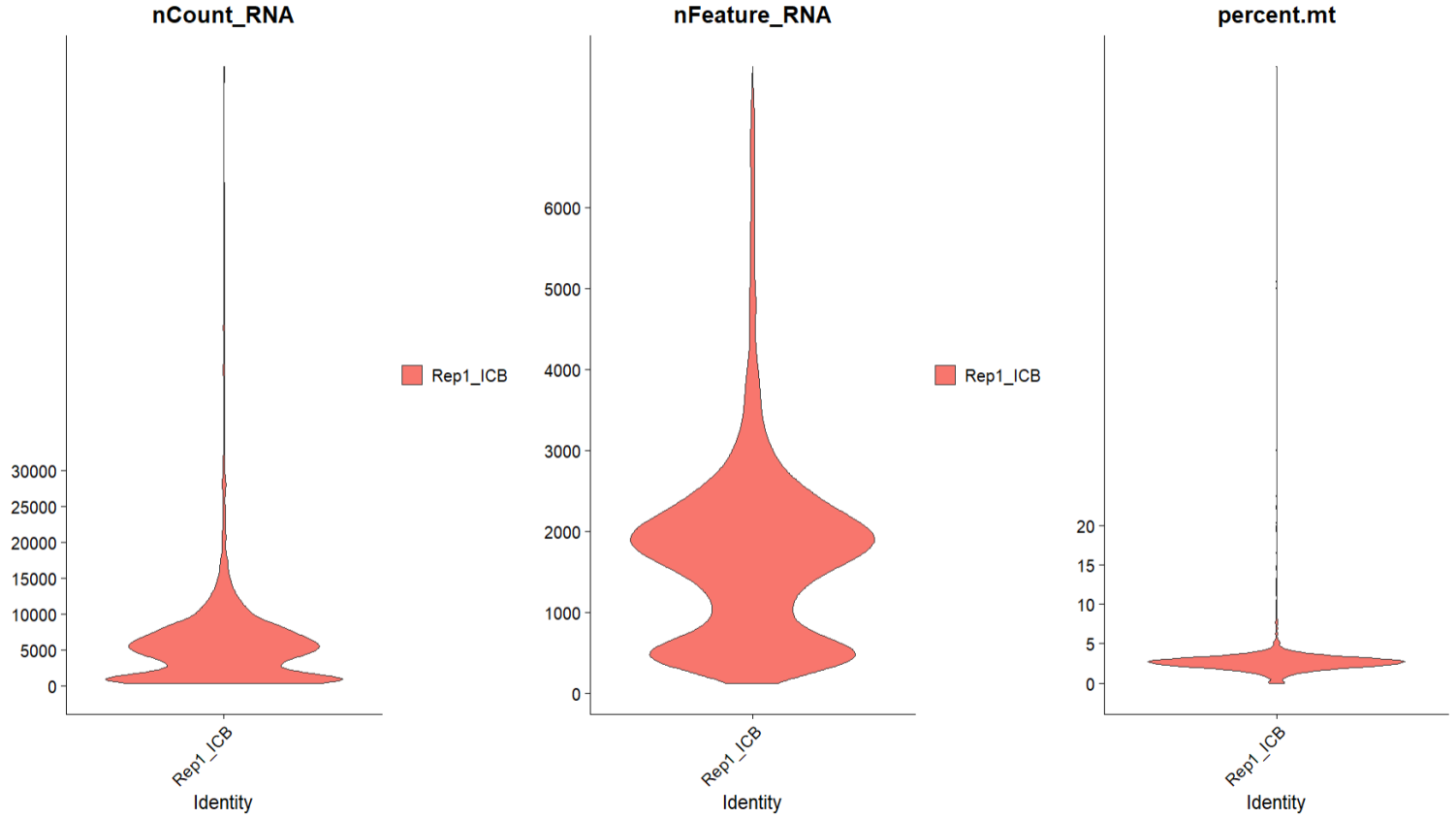
```
p1 <- VlnPlot(Rep1_ICB_data_seurat_obj,
              layer = "counts",
              features = c("nCount_RNA"),
              pt.size = 0) +
  scale_y_continuous(breaks = c(0,5000,10000,15000,20000,25000,30000))
p1
```



We can see most cells have over 2000 reads.

Exercise: do the same for the other two plots.

Add y-axis ticks



Create Seurat object for all 6 samples

We can write a for loop to do this at once.

First, let's create a variable which stores the **names of our samples**.

```
sample_names <- c("Rep1_ICBdT", "Rep3_ICBdT", "Rep5_ICBdT",
                  "Rep1_ICB", "Rep3_ICB", "Rep5_ICB")
```

Then, we need to create an **empty list** for later storing the Seurat objects.

```
sample.data <- list()
```

Create Seurat object for all 6 samples

The for loop can be written like this to input all 6 files in.

```
for (sample in sample_names) {
  cat("\nWorking on sample: ", sample, "\n")
  path <- paste("data/", sample, "-sample_filtered_feature_bc_matrix.h5", sep="")
  cat("Input file path: ", path, "\n")
  data <- Read10X_h5(path)
  seurat_obj <- CreateSeuratObject(counts = data, project = sample,
                                   min.cells = 10, min.features = 100)
  sample.data[[sample]] <- seurat_obj
}
```

The loop will run **6 times**, and each time, it takes one sample name in and run all the code in the box.

For example, the **first loop**, the `sample` would be `Rep1_ICBdT`.

Create Seurat object for all 6 samples

```
> print(sample.data)
$Rep1_ICBdT
An object of class Seurat
17035 features across 4012 samples within 1 assay
Active assay: RNA (17035 features, 0 variable features)
 1 layer present: counts

$Rep3_ICBdT
An object of class Seurat
16376 features across 5656 samples within 1 assay
Active assay: RNA (16376 features, 0 variable features)
 1 layer present: counts
```

If we use print() function to look at `sample.data`, it will print out all six items in it.

As shown above, the first item is called `Rep1_ICBdT`, and it is a Seurat object.

Calculate mitochondrial RNA percentage

To access one item from the list, we can use:

```
> sample.data$Rep1_ICBdT
An object of class Seurat
17035 features across 4012 samples within 1 assay
Active assay: RNA (17035 features, 0 variable features)
 1 layer present: counts
```

Or

```
> sample.data[["Rep1_ICBdT"]]
An object of class Seurat
17035 features across 4012 samples within 1 assay
Active assay: RNA (17035 features, 0 variable features)
 1 layer present: counts
```

Calculate mitochondrial RNA percentage

To calculate the percentage of mitochondrial RNA for all samples, we can write the for loop as:

```
for (sample in sample_names) {
  sample.data[[sample]][["percent.mt"]] <-
    PercentageFeatureSet(sample.data[[sample]], pattern = "^mt-", assay = "RNA")
}
```

Quality Assessment with Violin Plots

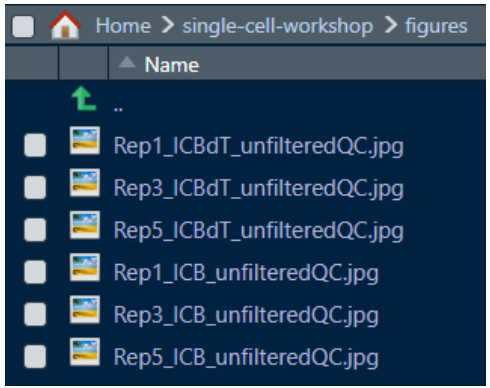
We can create violin plot for all 6 samples, and take time to read each plot, and decide **what threshold to use to filter our data**.

First, let's create a folder called "**figures**" to keep all our plots generated.

```
for (sample in sample_names) {
  jpeg(sprintf("%s/%s_unfilteredQC.jpg", "figures", sample),
       width = 16, height = 5, units = 'in', res = 150)
  p1 <- VlnPlot(sample.data[[sample]], features = c("nCount_RNA"), pt.size = 0)
  p2 <- VlnPlot(sample.data[[sample]], features = c("nFeature_RNA"), pt.size = 0) +
    scale_y_continuous(breaks = c(0, 300, 500, 1000, 2000, 4000))
  p3 <- VlnPlot(sample.data[[sample]], features = c("percent.mt"), pt.size = 0) +
    scale_y_continuous(breaks = c(0, 12.5, 25, 50))
  p <- plot_grid(p1, p2, p3, ncol = 3)
  print(p)
  dev.off()
}
```

Quality Assessment with Violin Plots

After running the code, you should see 6 files generated under "figures".



Then, we should carefully examine these figures to **decide a threshold**.

Cell numbers before filtering

Let's keep a record of the cell numbers of each sample before filtering.

	Number of Feature	Number of Cell
Rep1_ICBdT	17,035	4,012
Rep3_ICBdT	16,376	5,656
Rep5_ICBdT	15,751	6,066
Rep1_ICB	14,423	4,169
Rep3_ICB	16,346	6,474
Rep5_ICB	13,773	2,993

Filtering based on mtRNA percentage

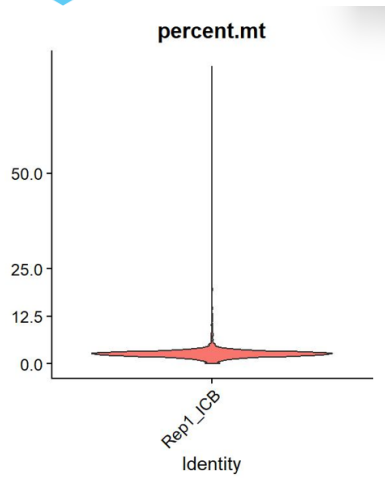
Low mitochondria cells: many common cell types, like immune cells or fibroblasts, have relatively low energy demands and will consistently show a very low mitochondrial RNA percentage, often **<5%**.

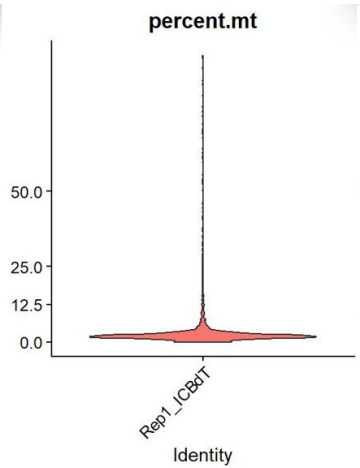
High mitochondria cells: specialised cells with **high energy requirements** naturally have more mitochondria and thus a higher baseline of mitochondrial RNA range from **5% to 15%.**

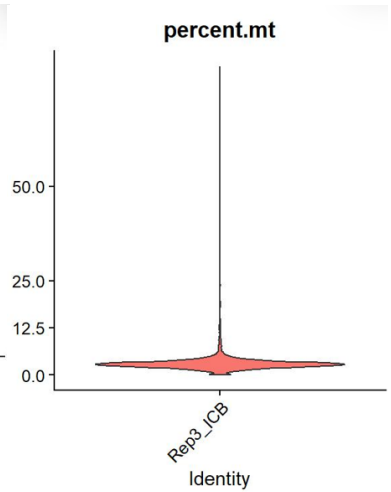
- Heart muscle cells
- Neurons
- Photoreceptors (in the eye)
- Certain kidney cells

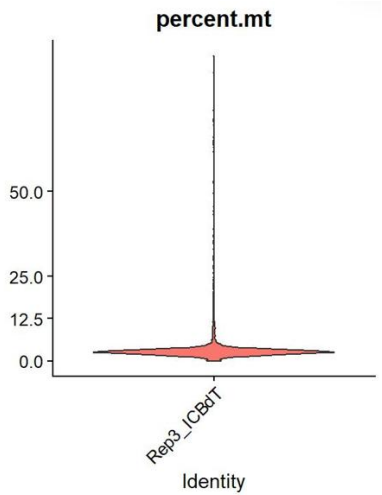
Cancer cells: some malignant cells could reach **10-20%**.

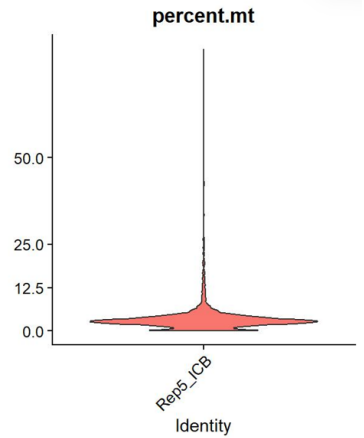
Filtering based on mtRNA percentage

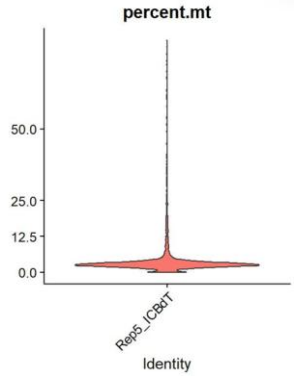












We can set the threshold to **12**.

Remove cells that has a mtRNA percentage above 12.

Filtering based on mtRNA percentage

```
Rep1_ICB_data_seurat_obj[["keep_cell_percent.mt"]] <-
  ifelse(Rep1_ICB_data_seurat_obj[["percent.mt"]] <= 12, TRUE, FALSE)
```

If the `percent.mt` value is less than or equal to 12, we will mark it as **TRUE**.

If not, we will mark it as **FALSE**.

This information will be saved in a new column called `**keep_cell_percent.mt**`.

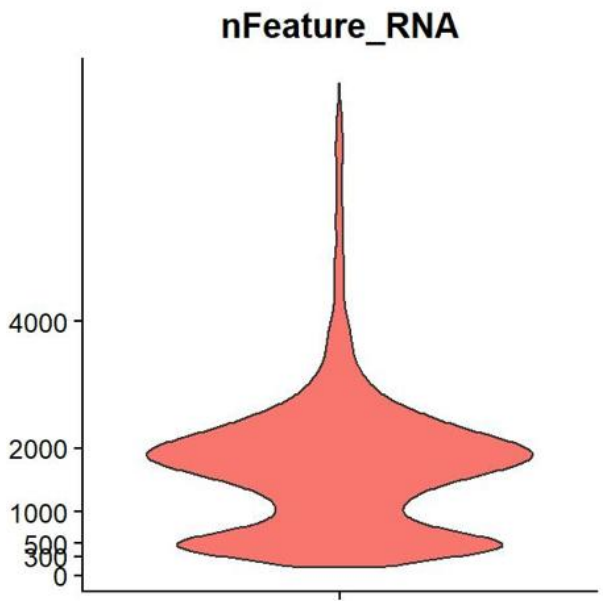
Filtering `nFeature_RNA`

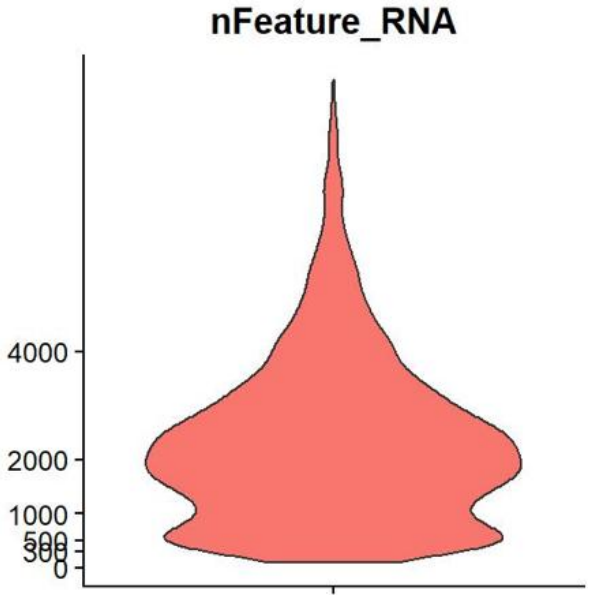
Take a look of this added column.

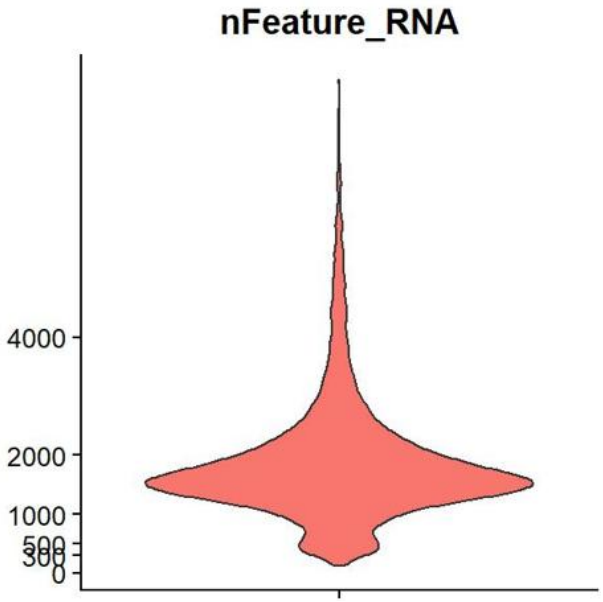
```
> head(Rep1_ICB_data_seurat_obj[[]])
                  orig.ident nCount_RNA nFeature_RNA percent.mt keep_cell_percent.mt
AAACCTGAGCGGATCA-1   Rep1_ICB       5679          1758   2.341962                TRUE
AAACCTGCAAGAGGCT-1   Rep1_ICB       1397           496  50.894775               FALSE
AAACCTGCATACTCTT-1   Rep1_ICB       8463          2444   2.197802                TRUE
AAACCTGCATCATCCC-1   Rep1_ICB       4640          1721   3.750000                TRUE
AAACCTGGTTCGGGCT-1   Rep1_ICB       3810          1291   2.755906                TRUE
AAACCTGTCAGTCCCT-1   Rep1_ICB       7810          2079   3.354673                TRUE
> 
```

Exercise: do the same for `nFeature_RNA`, we will keep cells that has more than 1000 features.

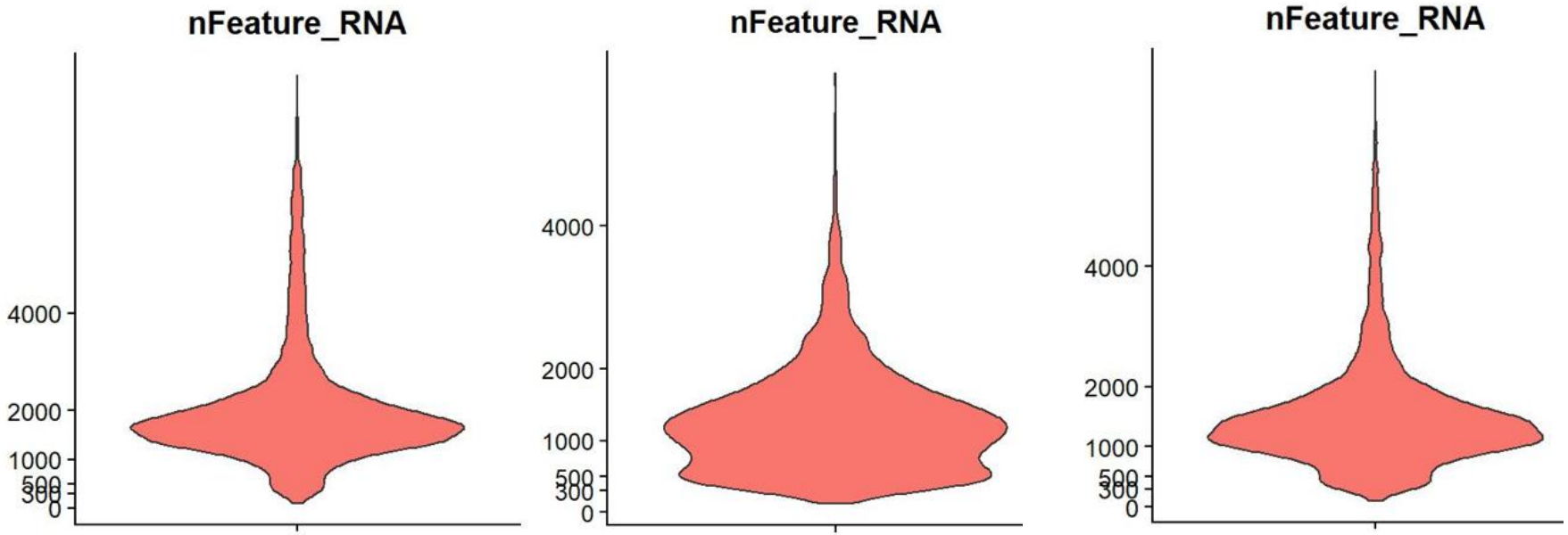
Filtering based on number of genes







Filtering based on number of genes



Filtering based on number of genes

From the violin plots, we can see that most of our samples have two peaks for the number of genes detected in a single cell.

- One is around **300-500**.
- The other one is around **1000-2000**.

The **small initial peak** usually represents **empty droplets** that have captured some free-floating ambient RNA but are not actual cells.

The **main peak** is the distribution of our healthy, viable cells.

Exercise

Add these two columns for all six samples.

Number of cells after filtering

```
for (sample in sample_names) {
  print(sample)
  print(sum(sample.data[[sample]][["keep_cell_nFeature"]] &
      sample.data[[sample]][["keep_cell_percent.mt"]]))
}
```

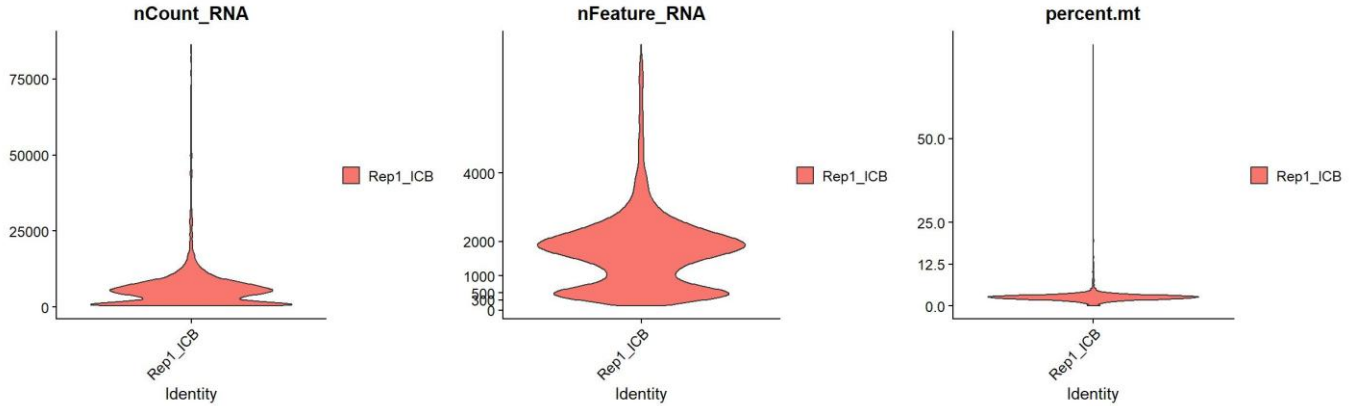
	N_Cell	N_Cell_filtered
Rep1_ICBdT	4,012	3,106
Rep3_ICBdT	5,656	5,072
Rep5_ICBdT	6,066	4,547
Rep1_ICB	4,169	2,986
Rep3_ICB	6,474	5,680
Rep5_ICB	2,993	1,794

Visualisation after filtering

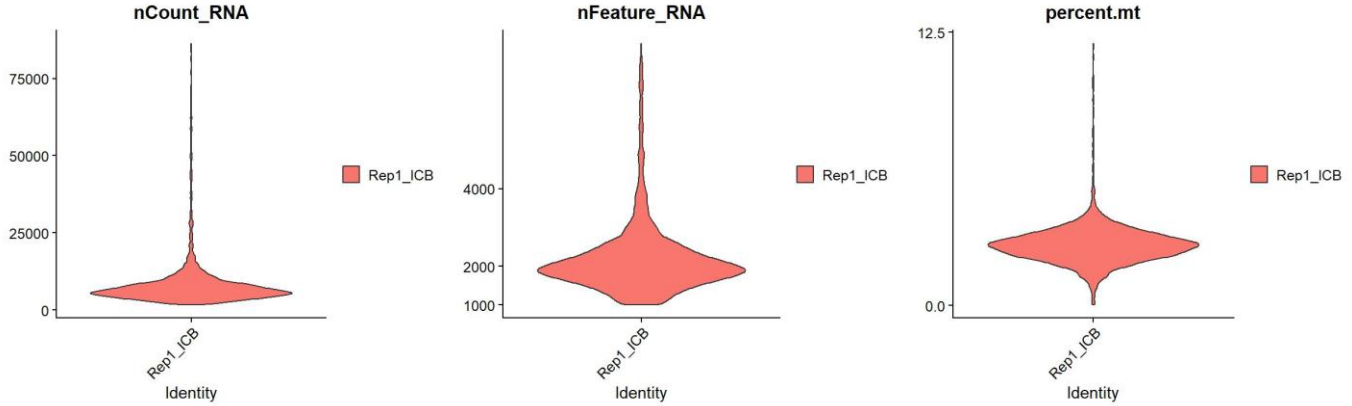
```
for (sample in sample_names) {
  jpeg(sprintf("%s/%s_filteredQC.jpg", "figures", sample), width = 16, height = 5,
       units = 'in', res = 150)
  p1 <- VlnPlot(subset(sample.data[[sample]], nFeature_RNA > 1000 & percent.mt <= 12),
                features = c("nCount_RNA"), pt.size = 0)
  p2 <- VlnPlot(subset(sample.data[[sample]], nFeature_RNA > 1000 & percent.mt <= 12),
                features = c("nFeature_RNA"), pt.size = 0) +
    scale_y_continuous(breaks = c(0, 300, 500, 1000, 2000, 4000))
  p3 <- VlnPlot(subset(sample.data[[sample]], nFeature_RNA > 1000 & percent.mt <= 12),
                features = c("percent.mt"), pt.size = 0) +
    scale_y_continuous(breaks = c(0, 12.5, 25, 50))
  p <- plot_grid(p1, p2, p3, ncol = 3)

  print(p)
  dev.off()
}
```

Compare Rep1_ICB

Before



After



Merge samples

```
# merge samples
unfiltered_merged <- merge(x = sample.data[["Rep1_ICBdT"]],
                           y = c(sample.data[["Rep3_ICBdT"]],
                                 sample.data[["Rep5_ICBdT"]],
                                 sample.data[["Rep1_ICB"]],
                                 sample.data[["Rep3_ICB"]],
                                 sample.data[["Rep5_ICB"]]),
                           add.cell.ids = sample_names)
```

The `**add.cell.ids**` option will add sample names to the end of cell barcode.

```
> print(unfiltered_merged)
An object of class Seurat
18187 features across 29370 samples within 1 assay
Active assay: RNA (18187 features, 0 variable features)
 6 layers present: counts.Rep1_ICBdT, counts.Rep3_ICBdT, counts.Rep5_ICBdT, counts.Rep1_I
CB, counts.Rep3_ICB, counts.Rep5_ICB
>
```

Filtering merged data

```
# filter merged data
merged <- subset(unfiltered_merged, nFeature_RNA > 1000 & percent.mt <= 12) |>
  JoinLayers()
```

When we merge Seurat objects, the count matrix will be stored in different layers.

We can use `JoinLayers()` function to merge them into one layer which is a single matrix for further analysis.

Filtering merged data

```
> merged[['RNA']]$counts
18187 x 23185 sparse Matrix of class "dgCMatrix"

  [[ suppressing 38 column names 'Rep1_ICBdT_AAACCTGAGCCAACAG-1', 'Rep1_ICBdT_AAACCTGAGCCTTGAT-
1', 'Rep1_ICBdT_AAACCTGAGTACCGGA-1' ... ]]
  [[ suppressing 38 column names 'Rep1_ICBdT_AAACCTGAGCCAACAG-1', 'Rep1_ICBdT_AAACCTGAGCCTTGAT-
1', 'Rep1_ICBdT_AAACCTGAGTACCGGA-1' ... ]]


Xkr4        . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
......
Sox17       . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
......
Mrpl15      . . . . . . 1 . 1 1 . 1 . . . 3 . . . . . 1 . 2 . . 3 . 1 . . 3 . . . . . .
......
Lypla1      3 . . . 2 . 2 1 . 2 . 2 . . . 3 . . 1 . . . 2 . 1 . 2 1 . . . 3 . . 1 . . .
......
```

This matrix has **18187 rows** which are gene names, and **23185 columns** which are individual cells from our 6 samples.

Thank you

Contact us

Jiajia Li
Research School of Biology

RN Robertson Building, 46 Sullivan's Creek Rd
The Australian National University
Canberra ACT 2600

E jiajia.li1@anu.edu.au

Australian National University

TEQSA PROVIDER ID: PRV12002 (AUSTRALIAN UNIVERSITY)
CRICOS PROVIDER CODE: 00120C