Hands on: 10X Genomics Data Analysis

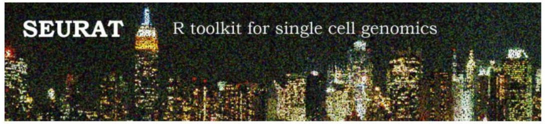
Single Cell RNA-seq Course 2021

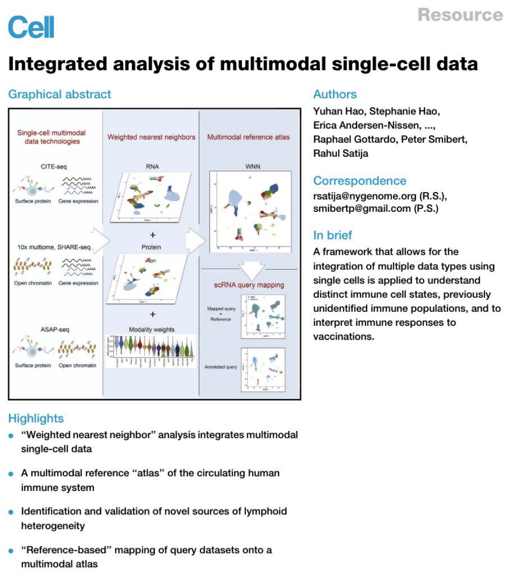
Matthew Madgwick and Anita Scoones

Overview

In this session we will cover:

1. The **theory behind 'downstream' single-cell analysis** with a focus on 10X Genomics data
2. How to **analyse a single-cell dataset** using the **Seurat Package**
3. Address some of the **assumptions/difficulties** you may run into when analysing single-cell data





Cell

Resource

Integrated analysis of multimodal single-cell data

Graphical abstract

Authors

Yuhan Hao, Stephanie Hao, Erica Andersen-Nissen, ..., Raphael Gottardo, Peter Smibert, Rahul Satija

Correspondence

rsatija@nygenome.org (R.S.), smibertp@gmail.com (P.S.)

In brief

A framework that allows for the integration of multiple data types using single cells is applied to understand distinct immune cell states, previously unidentified immune populations, and to interpret immune responses to vaccinations.

Highlights

- "Weighted nearest neighbor" analysis integrates multimodal single-cell data
- A multimodal reference "atlas" of the circulating human immune system
- Identification and validation of novel sources of lymphoid heterogeneity
- "Reference-based" mapping of query datasets onto a multimodal atlas

* **Any questions which unanswered please feel free to post them on slack or contact us directly**

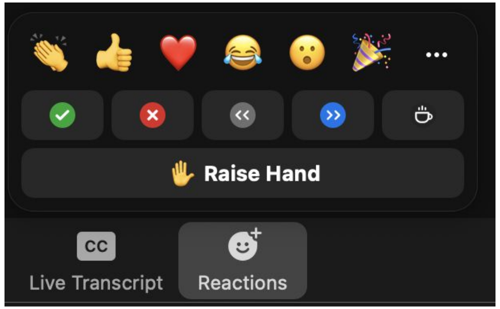
Contents

Breakout Session 1
- Preprocessing
- Starting a new Project
- Loadings your data
- Seurat Object Structure

Breakout Session 2
- QC and Filtering
- Normalisation

Breakout Session 3
- Highly Variable Gene Selection
- Principal Component Analysis

Breakout Session 4
- Clustering
- Differential Expression
- Cell Type Identification

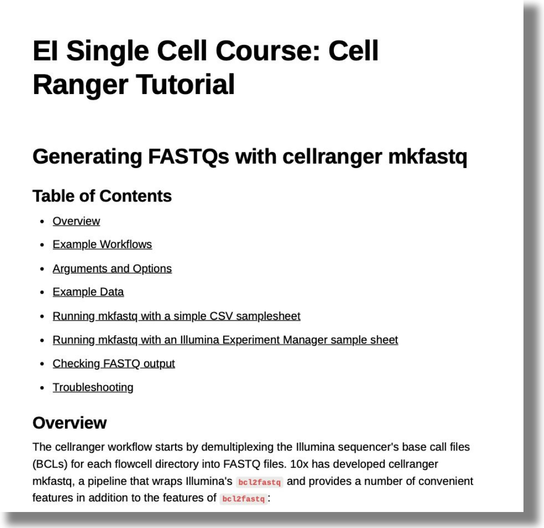


Preprocessing

(already done for you)

Cellranger: 10X Genomics Preprocessing Pipeline (1)

- Made specifically for 10X data
- Very Good documentation
- Relatively slow compared to some of the alternatives however is very frequently updated and improved upon
- The output of from cellranger is a **Feature-Barcode Matrices** (more on this later)



EI Single Cell Course: Cell Ranger Tutorial

Generating FASTQs with cellranger mkfastq

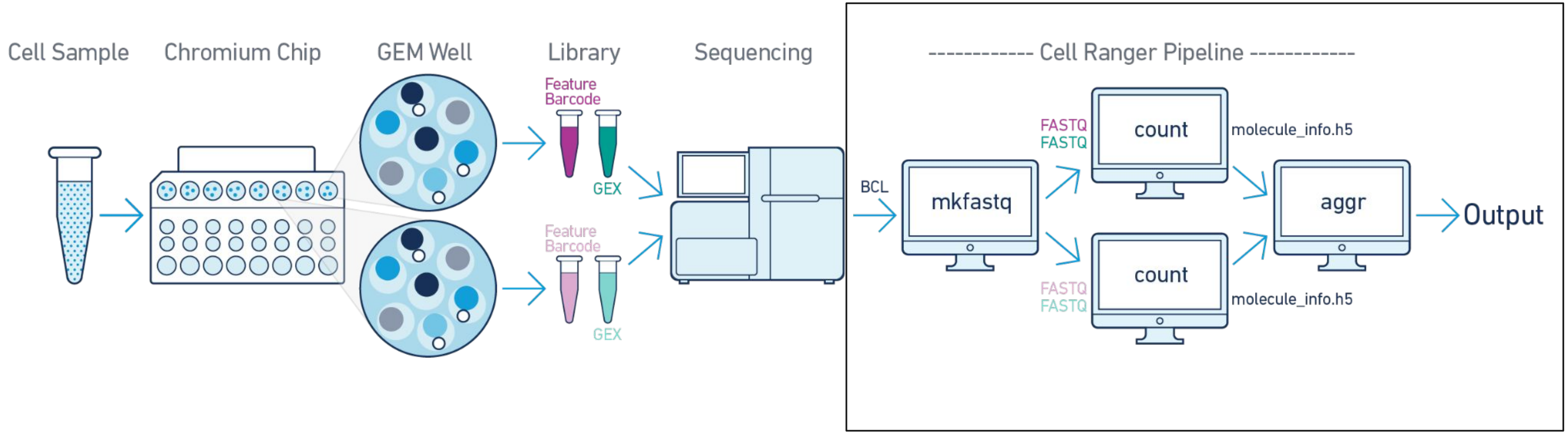
Table of Contents

Overview

The cellranger workflow starts by demultiplexing the Illumina sequencer's base call files (BCLs) for each flowcell directory into FASTQ files. 10x has developed cellranger mkfastq, a pipeline that wraps Illumina's `bcl2fastq` and provides a number of convenient features in addition to the features of `bcl2fastq`:

`ei_sc_cellranger_tutorial.pdf`
(can be found in the downloaded repo)

Cellranger: 10X Genomics Preprocessing Pipeline (2)



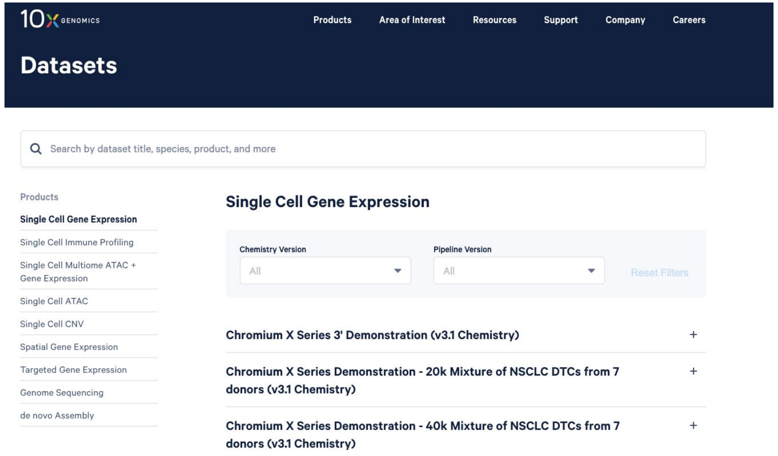
* BCL = base call files (binary format produced by the sequencer)

Loading your data

Data background

The data you will be analysing is Human Peripheral Blood Mononuclear Cells (PBMC).

- 2,700 single cells that were sequenced on the Illumina NextSeq 500
- Data was generated by 10X Genomics and is free available for download via their dataset portal
    - https://www.10xgenomics.com/resources/datasets



Chromium X Series 3' Demonstration (v3.1 Chemistry)

Chromium X Series Demonstration - 20k Mixture of NSCLC DTCs from 7 donors (v3.1 Chemistry)

Chromium X Series Demonstration - 40k Mixture of NSCLC DTCs from 7 donors (v3.1 Chemistry)

Loading data into Seurat

Droplet-based (10X)

Droplet-based methods usually come in the form of **Feature-Barcode Matrices**. These are created of **3 files** (usually compressed seen by the .gz extension) which are held within a **directory/folder**.

```
filtered_feature_bc_matrix
├── barcodes.tsv.gz
├── features.tsv.gz
└── matrix.mtx.gz
```

```
10x_data <- Read10X(data.dir ="filtered_gene_bc_matrices")
seurat_object <- CreateSeuratObject(counts = 10x_data,
                                    project = "name")
```

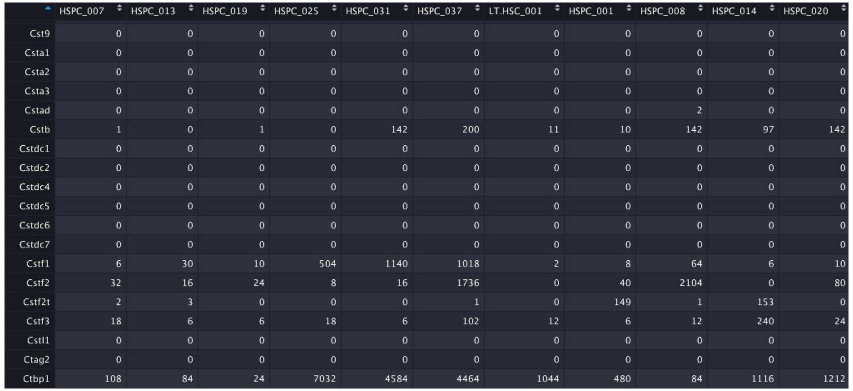
Loading data into Seurat

Plate-based (SS2 or SS3)

Plate-based methods will usually come in the format of a counts matrix. This will be a single file which has the cells in the columns and the gene as the rows*.

`ss2_matrix.tsv`

```
matrix <- read.table(file, header = TRUE,
                           row.names = "gene_name")
seurat_object <- CreateSeuratObject(counts = matrix,
                                         project = "name")
```



	HSPC_007	HSPC_013	HSPC_019	HSPC_025	HSPC_031	HSPC_037	LT.HSC_001	HSPC_001	HSPC_008	HSPC_014	HSPC_020
Cst9	0	0	0	0	0	0	0	0	0	0	0
Csta1	0	0	0	0	0	0	0	0	0	0	0
Csta2	0	0	0	0	0	0	0	0	0	0	0
Csta3	0	0	0	0	0	0	0	0	0	0	0
Cstad	0	0	0	0	0	0	0	0	2	0	0
Cstb	1	0	1	0	142	200	11	10	142	97	142
Cstdc1	0	0	0	0	0	0	0	0	0	0	0
Cstdc2	0	0	0	0	0	0	0	0	0	0	0
Cstdc4	0	0	0	0	0	0	0	0	0	0	0
Cstdc5	0	0	0	0	0	0	0	0	0	0	0
Cstdc6	0	0	0	0	0	0	0	0	0	0	0
Cstdc7	0	0	0	0	0	0	0	0	0	0	0
Cstf1	6	30	10	504	1140	1018	2	8	64	6	10
Cstf2	32	16	24	8	16	1736	0	40	2104	0	80
Cstf2t	2	3	0	0	0	1	0	149	1	153	0
Cstf3	18	6	6	18	6	102	12	6	12	240	24
Cstl1	0	0	0	0	0	0	0	0	0	0	0
Ctag2	0	0	0	0	0	0	0	0	0	0	0
Ctbp1	108	84	24	7032	4584	4464	1044	480	84	1116	1212

** NB: some matrices will have a header for row names (first column) others will not. Check the data before you load it in.*

Earlham Institute

Loading data into Seurat: Things to consider

- Always check your input format before you start. If you are unsure which method to use, then check the Seurat documentation
- Gene names: the gene names may not always be in the form of gene symbols. It could also be ENSEMBL IDs.
    - You can either remap genes during your preprocessing or within your downstream analysis using a package like `biomartr` (https://cran.r-project.org/web/packages/biomartr/index.html)
- You may also want to add meta-data which is associated with your samples. Using your barcode/read name as the ID. You can map another file using this code snippet:

```
meta_data <- read.table(file = “meta_data.txt, header = TRUE,
                        row.names = "gene_name")
seurat_object <- AddMetaData(seurat_obj, metadata = meta_data)
```

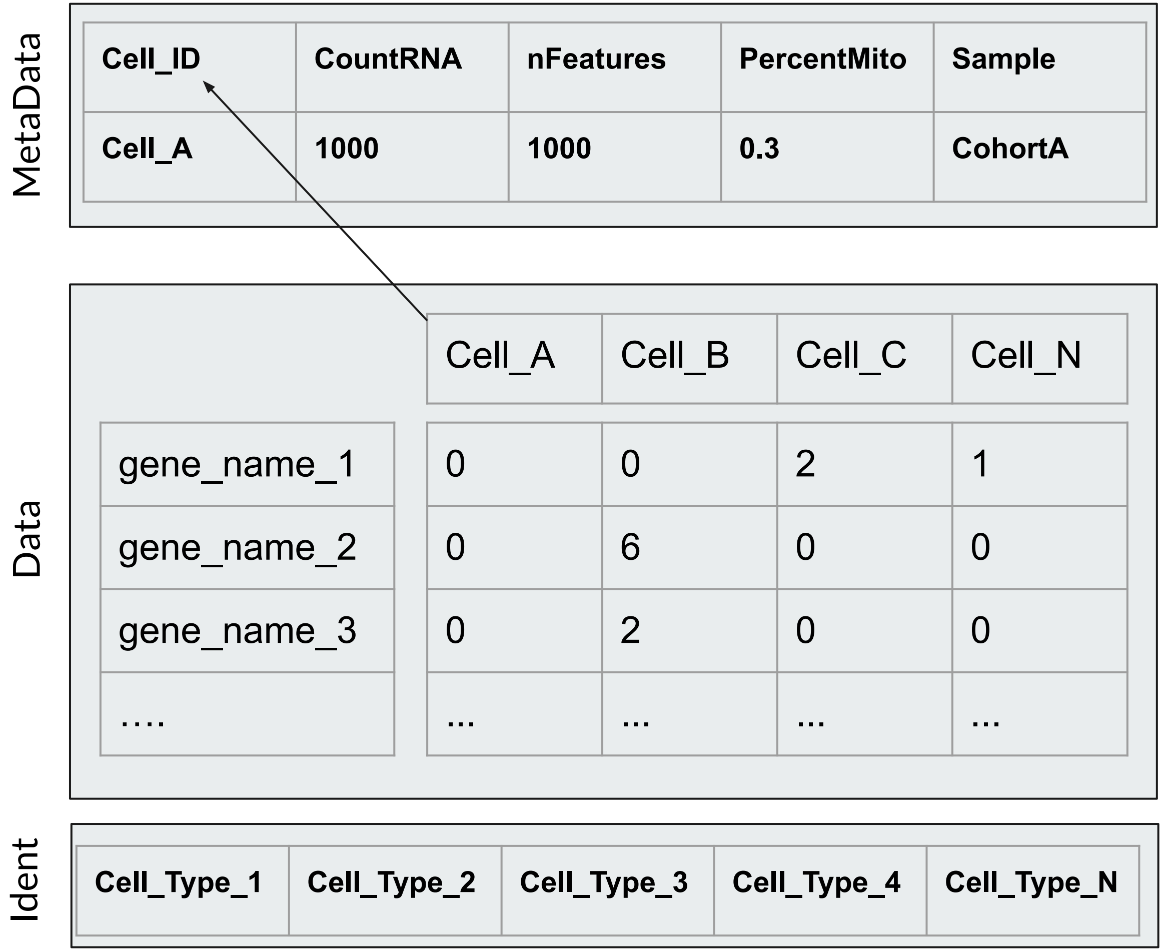
Seurat Object Structure

Seurat Object

A **`SeuratObject`** is a class (data structure) designed to hold all of your single-cell data in one place.

Seurat has these main data blocks:

- Data
- Meta-data
- Idents (names of give to a cell)
    - Before assigning cell-type this will either the barcode or the read name



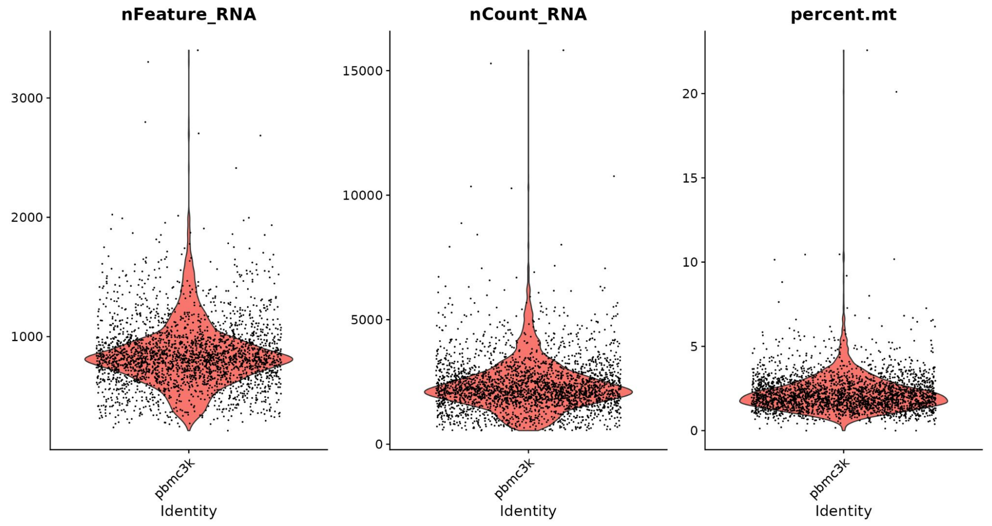
QC Filtering

QC Filtering (1)

Similar to what you have done before during pre-processing, you can explore the quality of your data by assessing:

- Counts (reads per cell)
- Features (number of genes per cell)
- % Mito (mitochondrial content per cell)

**NB: Depending on your biological question, the metrics you plot/are interesting in maybe different.*



QC Filtering (2)

Now that you have decided on your cutoffs (i.e. percent mito, n_counts, n_features). You will need to subset your data.

Seurat has a function called `subset` to do this for you. It uses some logic based operators:

- &: and
- > or <: is greater than and less than
- == : is equal to
- != : not equal to

You can subset on QC values:

```
# Subset on QC values
subset(seurat_object, subset = nFeature_RNA > 100 & mito < 5)
```

Or you can subset on other values in the data or meta data:

```
# Subset on a combination of criteria
subset(x = seurat_object, subset = MS4A1 > 3 & PC1 > 5)
subset(x = seurat_object, subset = MS4A1 > 3, idents = "B
cells")

# Subset on a value in the object meta data
subset(x = seurat_object, subset = orig.ident == "Replicate1")

# Downsample the number of cells per identity class
subset(x = seurat_object, downsample = 100)
```

Breakout Session 1

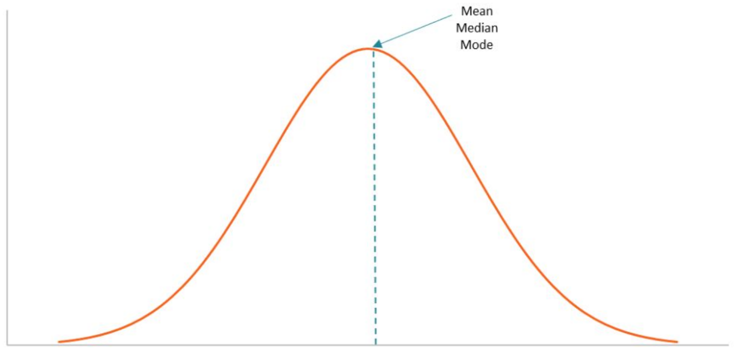
Normalisation

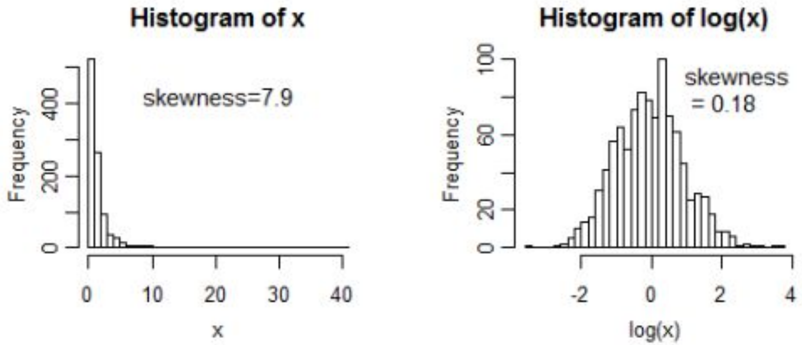
Why normalise data?

- Data normalization is vital to single-cell sequencing, addressing limitations presented by low input material and various forms of bias or noise present in the sequencing process

- The overall goal is to create a more normal (*Gaussian*) **distribution** aka a bell curve

- In seurat you will use:
    - **LogNormalize:** Feature counts for each cell are divided by the total counts for that cell and multiplied by the scale.factor. This is then natural-log transformed using log1p.



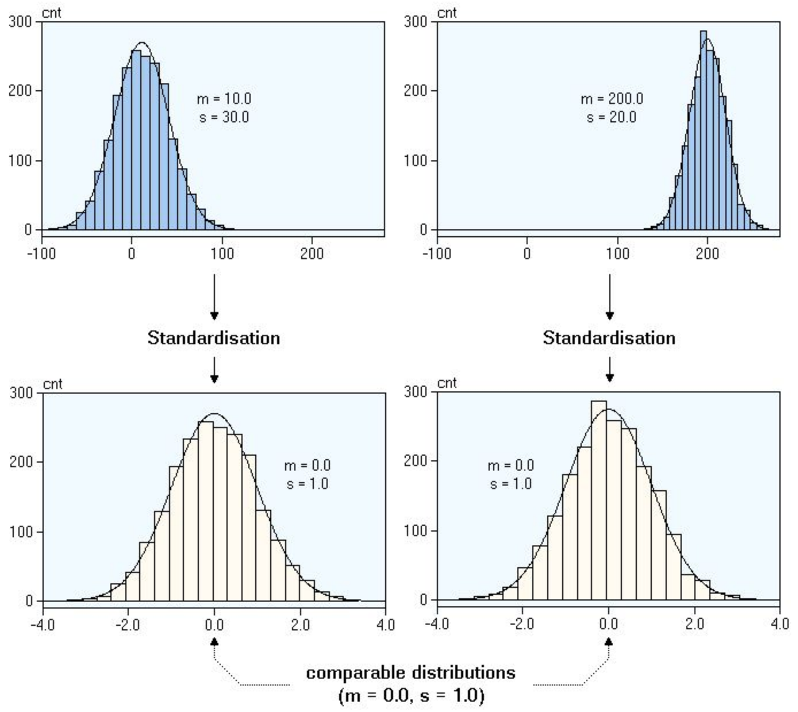


Scaling data

Scaling the data removes unwanted sources of variation

- Shifts the expression of each gene, so that the **mean expression across cells is 0**
- Scales the expression of each gene, so that the **variance across cells is 1**

This step gives **equal weight in downstream analyses**, so that **highly-expressed genes do not take over the analysis**



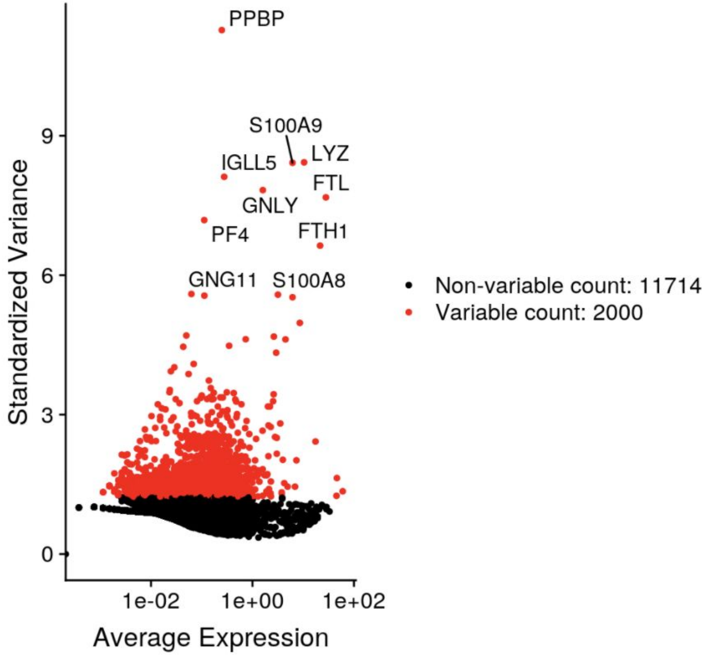
Highly Variable Gene Selection

Highly variable gene selection

Seurat calculates highly variable genes and focuses on these for downstream analysis.

`FindVariableFeatures()` calculates the average expression and dispersion for each gene, places these genes into bins, and then calculates a **z-score** each bin

TLDR; Remove any genes that show little to no variation (minimal differences in gene expression across samples)



Breakout Session 2

Principal Component Analysis (PCA)

What is PCA?

PCA is **dimensionality-reduction** method that is often used to **reduce the dimensionality of large data sets**, by transforming a large set of variables **into a smaller one that still contains most of the information in the large set**.

How does PCA work?

Genes

Cells

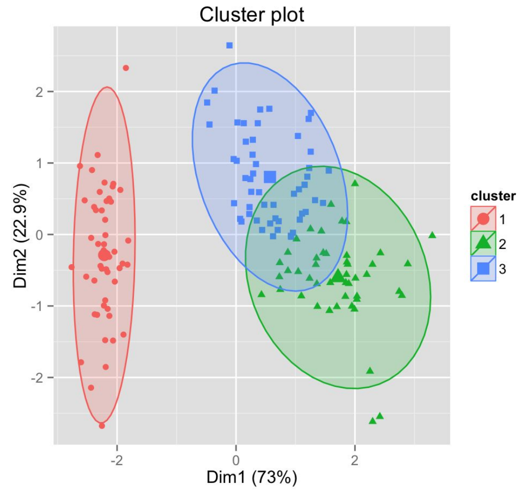
0	2	3	4	0
0	2	1	0	0
2	2	5	0	1
1	0	0	0	0
...	...	...	...	...

PCs

Cells

0.22	0.41
0.10	0.37
0.14	0.53
0.11	0.31
...	...

$$\Sigma = \begin{bmatrix} \mathrm{Var}(X) & \mathrm{Cov}(X,Y) & \mathrm{Cov}(X,Z) \\ \mathrm{Cov}(X,Y) & \mathrm{Var}(Y) & \mathrm{Cov}(Y,Z) \\ \mathrm{Cov}(X,Z) & \mathrm{Cov}(Y,Z) & \mathrm{Var}(Z) \end{bmatrix}$$



PCA: Seurat Object

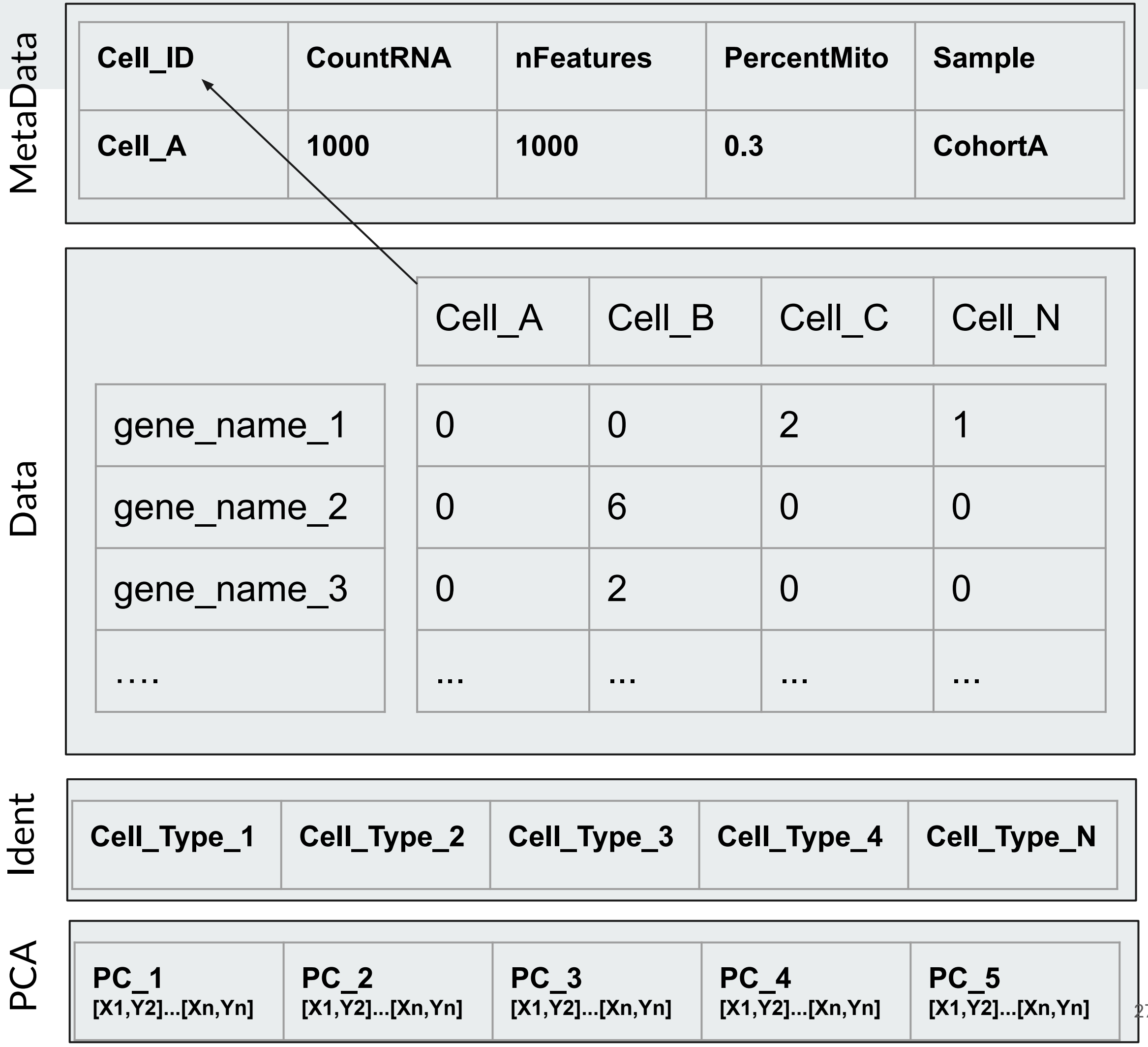
A **`SeuratObject`** is a class (data structure) designed to hold all of your single-cell data in one place.

Seurat has these main data blocks:

- Data
- Meta-data
- Idents (names of give to a cell)

PCA adds on to Seurat Object:

```
seurat_object[["pca"]]
```



MetaData

Cell_ID	CountRNA	nFeatures	PercentMito	Sample
Cell_A	1000	1000	0.3	CohortA

Data

	Cell_A	Cell_B	Cell_C	Cell_N
gene_name_1	0	0	2	1
gene_name_2	0	6	0	0
gene_name_3	0	2	0	0
….	...	...	...	...

Ident

Cell_Type_1	Cell_Type_2	Cell_Type_3	Cell_Type_4	Cell_Type_N

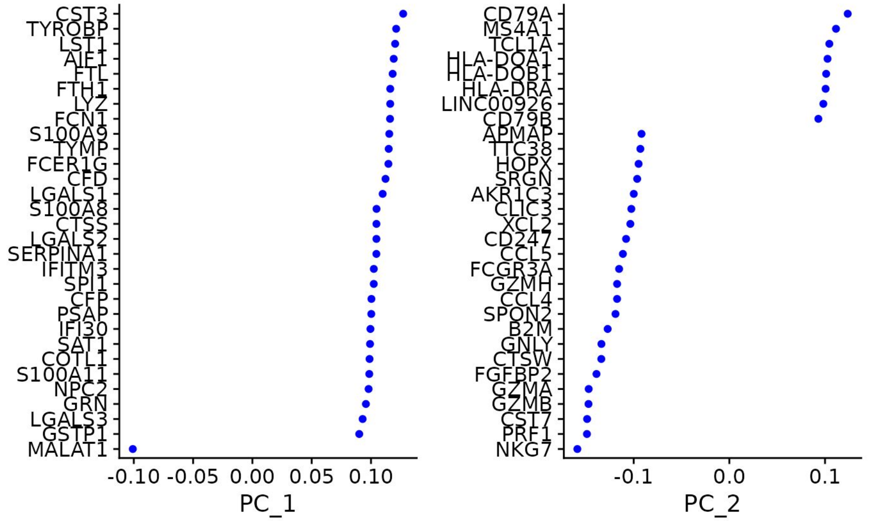
PCA

PC_1 [X1,Y2]...[Xn,Yn]	PC_2 [X1,Y2]...[Xn,Yn]	PC_3 [X1,Y2]...[Xn,Yn]	PC_4 [X1,Y2]...[Xn,Yn]	PC_5 [X1,Y2]...[Xn,Yn]

Investigating the results of a PCA

The loadings of a PCA show the weights (contribution) of a gene to that particular Principle Component (PC).

- The bigger the value the greater the contribution
- You expect to see a sharp cut-off between those with higher weights to those which are less important to that PC.



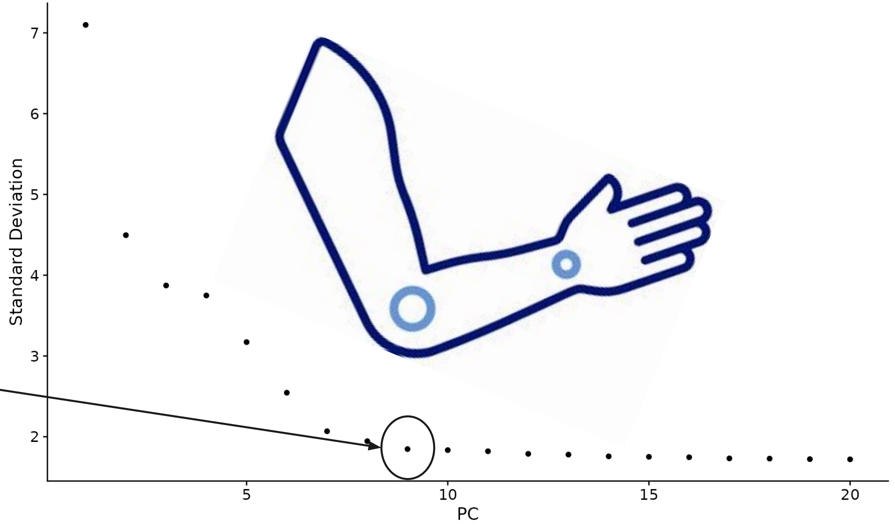
How many Principal Components should I use?

The **elbow plot** is used to determine how many PCs we need include to capture the **majority** of the **variation** in the data.

The elbow plot visualises the **standard deviation of each PC**.

Where the “elbow” appears is usually the threshold for identifying the majority of the variation.

(ie. including additional PCs is unlikely to reveal significant sources of variation)

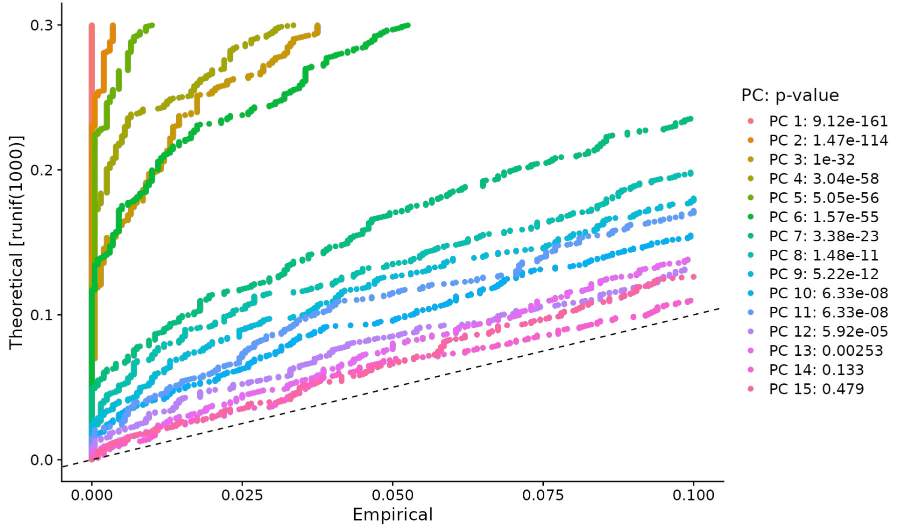


How many Principal Components should I use?

There are other methods which can use. However, these are much more computationally expensive.

Jack Straw Method: this method provides p-values for each PC. The lower the value the higher the significance of the PC.

- Closer the PC (line) is the y-axis the smaller the p-value and therefore the more the component is explaining the variance



Why do we need to use PCA?

- **Reduction**: PCA helps you 'collapse' all the genes into more manageable numbers
- **New information**: PCA creates new variables from the existing genes in different proportions. Lots of genes are compressed into **Principal Components (PCs)**
- **Independent variables**: PCA not only creates new variables but creates them in such a manner that they are not correlated, i.e. avoids lots of correlated variables in a single component.
- **Outliers**: When working with many variables, it is challenging to spot outliers, errors, or other suspicious data points. Reducing a large number of variables and visualizing them help you spot outliers. Spotting outliers is a significant benefit and application of PCA.

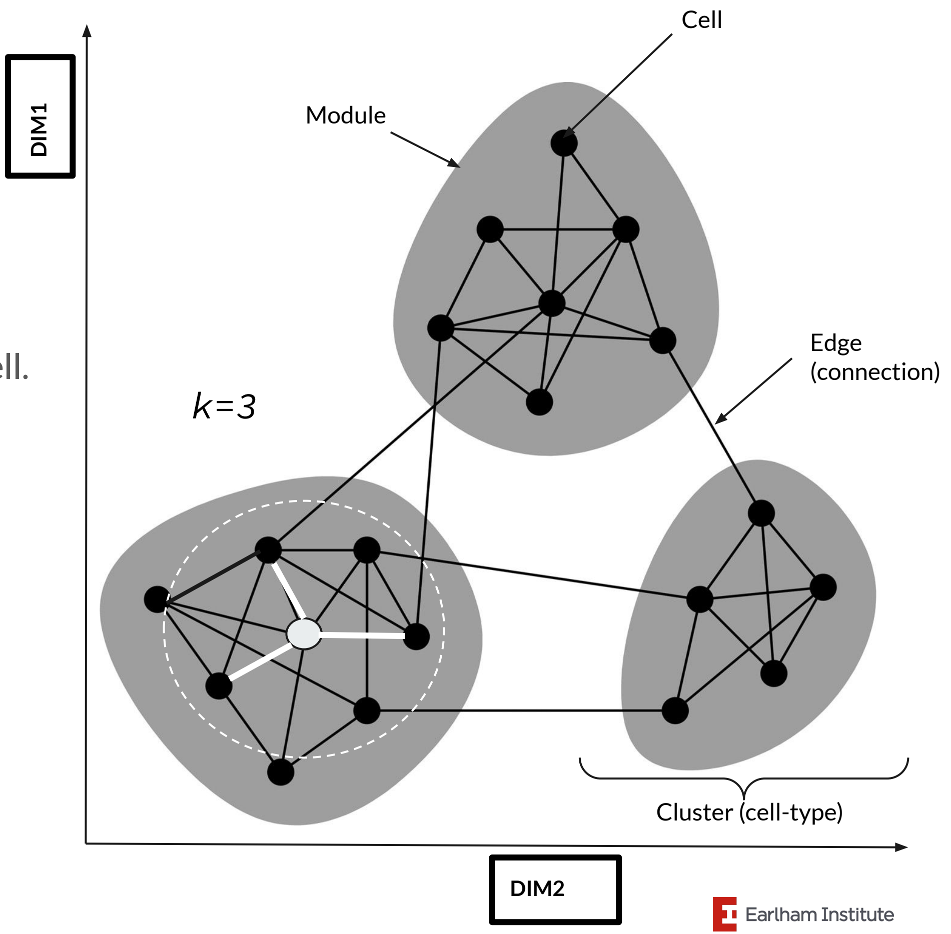
Breakout session 3

Clustering

Clustering Overview (1)

Next wee need to cluster the cells based of the PCs we found. These represent the “transcriptomic profile” of a cell. To do this the algorithm:

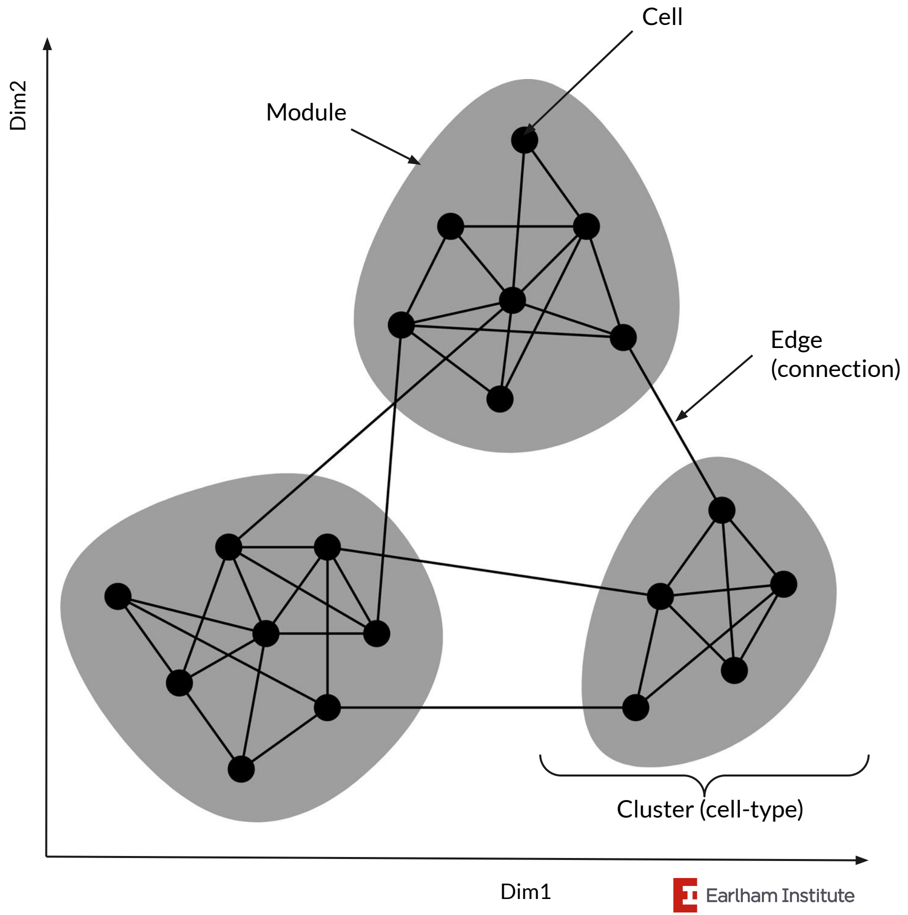
1. **Calculate similarity between cells**
2. **Find overlaps with closest cells**
3. **Put cells into groups**



Earlham Institute

Clustering Overview (2)

1. **Partitioning the cellular distance matrix**: Calculate distance (i.e. similarity) between Principal Components (PCs)
2. **Build a graph based on cells neighbours**: k-nearest neighbours (KNN)-graph based on the distance in PCA space, and optimise the edge weights between any two cells based on the shared overlap with their nearest k neighbours
3. **Group cells together**: based off the “modularity” (or the similarity of the neighbours to make groups of cells) we apply the Louvain algorithm

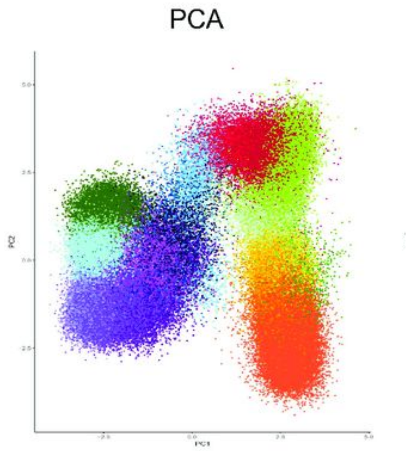


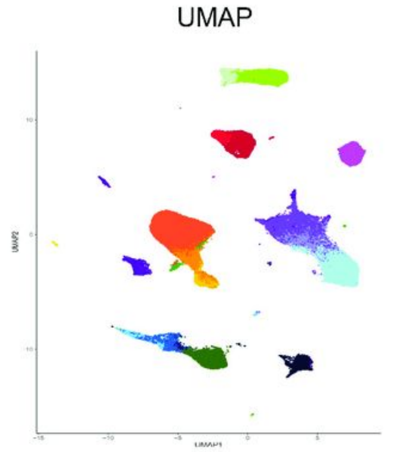
Clustering Overview: Methods (3)

There are many different methods to visualise your clusters*. Some of the most common are:

- **PCA:** Principal Component analysis
- **TSNE:** t-distributed stochastic neighbor embedding
- **UMAP**: Uniform Manifold Approximation is like PCA but is created to separate data the data

* NB: no one technique is better than the other. There is no way to map a high-dimensional data into low dimensions and preserving the whole structure at the same time, there is always a trade-off of qualities one technique will have compared to the other.





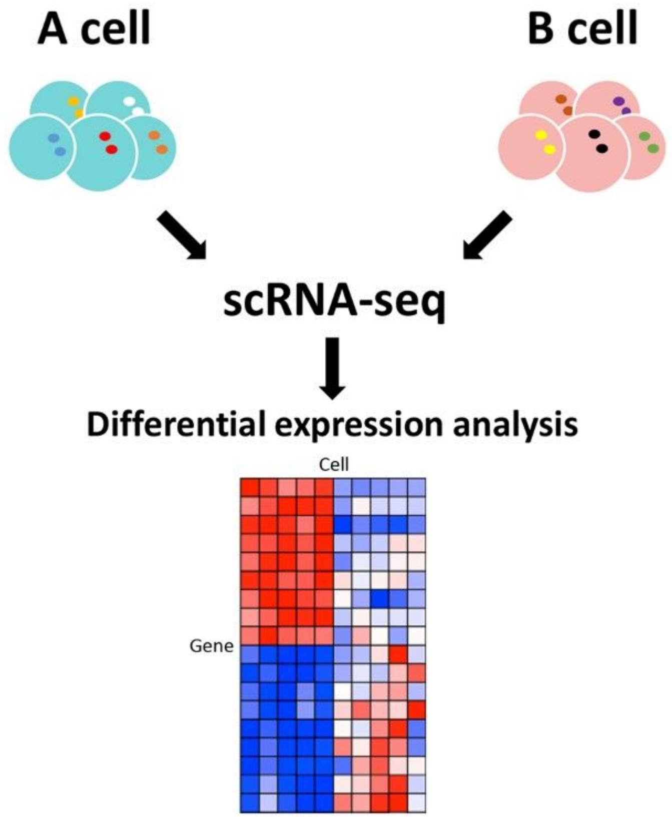
Earlham Institute

Differential Expression

Differential Expression

Seurat uses differential expression to:

1. **Identify marker genes**: i.e. finding markers within cell clusters to be able to assign a cell type (`FindAllMarkers`)
2. **Compare between experiment groups**: this could be conditions, particular cell-types, populations, etc (`FindMarkers`)



Differential Expression between conditions

We can use this same function `FindMarkers` to find differential expressed genes (DEGs) between conditions. To do this we just need to add/change a few parameters in FindMarkers:

```
FindMarkers(pbmc, ident.1 = "ConditionA", ident.2 = "ConditionB", only.pos = TRUE, min.pct = 0.5)
```

You can set these to the Idents() from the meta-data

NB: depending on the size of the dataset, this can take A LOT OF TIME!

Differential Expression in Seurat

	p_val	avg_logFC	pct.1	pct.2	p_val_adj
S100A9	0	3.860873	0.996	0.215	0
S100A8	0	3.796640	0.975	0.121	0
LGALS2	0	2.634295	0.908	0.059	0
FCN1	0	2.352693	0.952	0.151	0
CD14	0	1.951644	0.667	0.028	
TYROBP	0	2.111879	0.994	0.265	0

First 6 values from the differential expression results

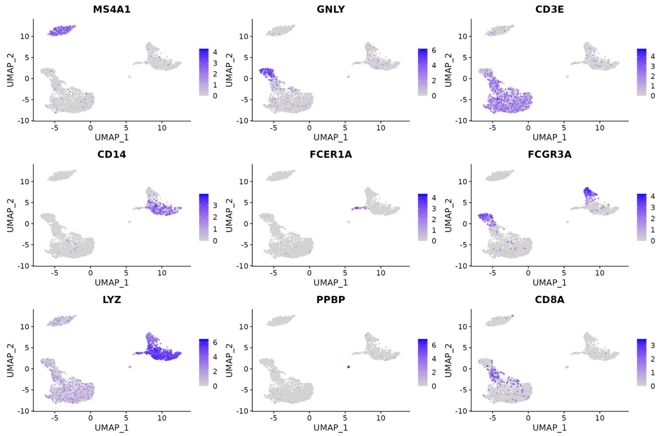
- **p_val**: p_val (unadjusted)
- **avg_log2FC** : log fold-change of the average expression between the two groups. Positive values indicate that the feature is more highly expressed in the first group.
- **pct.1**: The percentage of cells where the feature is detected in the first group
- **pct.2**: The percentage of cells where the feature is detected in the second group
- **p_val_adj**: Adjusted p-value, based on bonferroni correction using all features in the dataset.

Cell Type Identification

Cell Type Identification

To determine what cell-type a cluster is there are several different tools/resources you can use:

- Search top marker genes in databases (i.e. uniprot or human cell atlas)
- Literature search for marker genes
- Use reference single-cell data to label your cells
- Use machine learning-based tools to predict the cell type from known transcriptomic profiles



UniProt

HUMAN CELL ATLAS

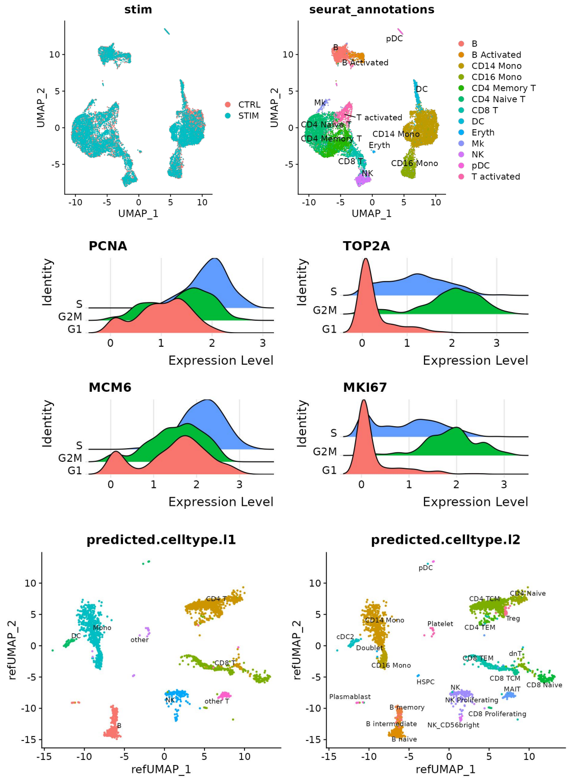
Breakout Session 4

Additional features

Seurat can also:

- **Integrate multi-modal data**: i.e. different single-cell datasets
- **Regress out confounding effects**: this can be done for ribosomal or cell cycle genes.
- **Assign Cell Types from reference**: can label your cells from a reference datasets. Useful if you have already annotated a dataset

You can find more example and features here: https://satijalab.org/seurat/index.html



Conclusion

To wrap up

The takeaways from the session are:

- You have a better understanding of how a ‘downstream’ single-cell analysis works
- You have a better understanding of the importance of exploring and investigating your data
- You can use the same workbook you have, by changing the input data you can perform the same analysis of your 10X/plate-based data or any public data in the same format

Any questions?