Practical Example: NGS – data handling and single cell differentiation

Ivan G. Costa, Martin Manolov, James Nagai, Zhijian Li, Mingbo Cheng, Mina Shaigan

Institute for Computational Genomics

Joint Research Centre for Computational Biomedicine

RWTH Aachen University, Germany

Institute for Computational Genomics
01011011010
10100100101

RWTH AACHEN UNIVERSITY

Contact Information

Martin Manolov, Zhijian Li, James Nagai, Mingbo Cheng, MINA Shaigan

martin.manolov@rwth-aachen.de

zhijian.li@rwth-aachen.de

james.nagai@gmail.com

mcheng@ukaachen.de

mina.shaigan@gmail.com

Useful URL

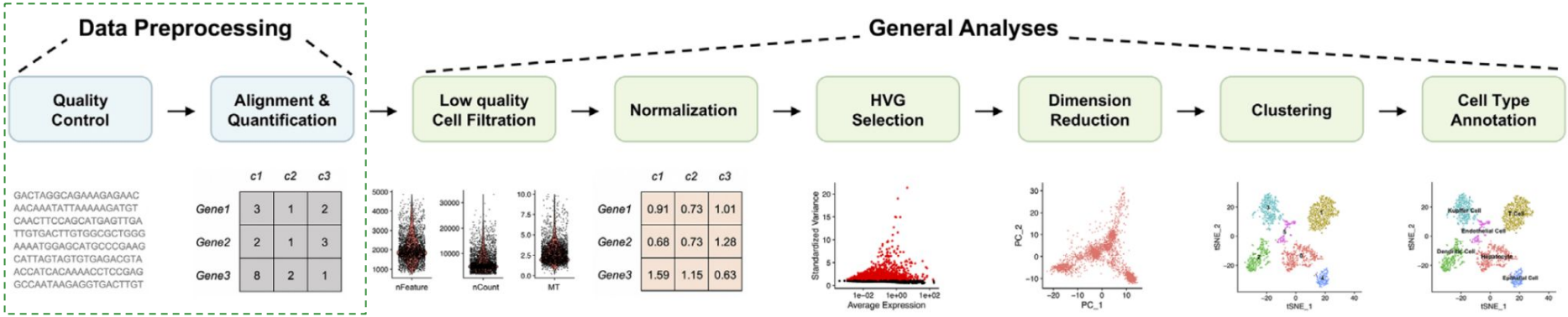
- Course material: https://www.costalab.org/software-lab-in-bioinformatics-2021/
- Single Cell Broad Institute Workshop: https://broadinstitute.github.io/2020_scWorkshop/
- Best Practices in Single Cell Analysis: https://www.embopress.org/doi/10.15252/msb.20188746

Overview

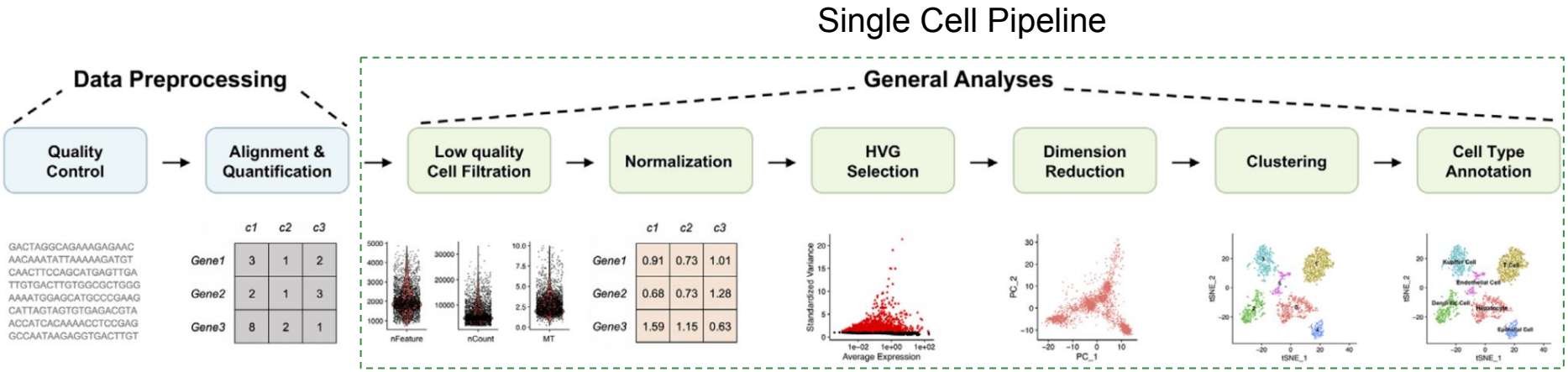
- DNA Sequencing:
  - New Generation Sequencing Methods
  - Sequencing Quality Check
  - Alignment of Sequenced Reads
- Single Cell Sequencing
  - Demultiplexing Reads
  - Single Cell Expression Matrix
  - From the Expression Matrix to Information Retrieval

Overview

DNA Sequencing

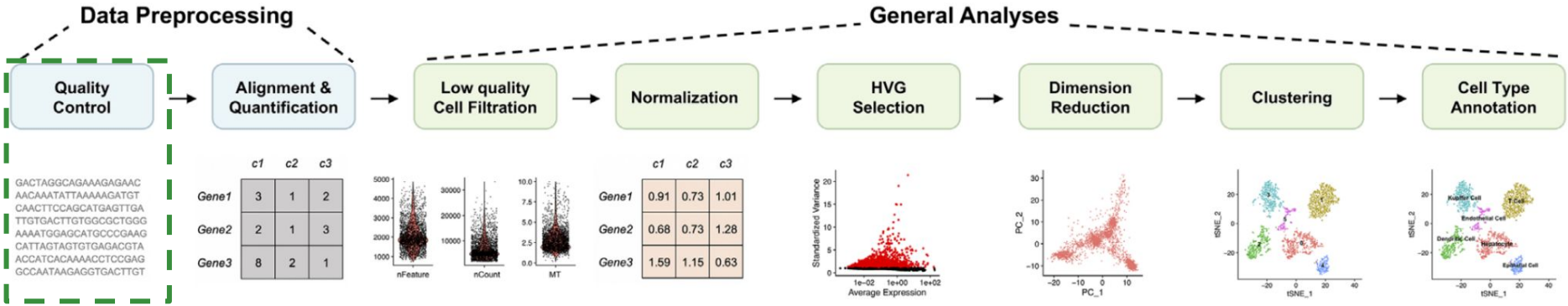


Overview


Single Cell Pipeline
Data Preprocessing
General Analyses
Quality Control
Alignment & Quantification
Low quality Cell Filtration
Normalization
HVG Selection
Dimension Reduction
Clustering
Cell Type Annotation
GACTAGGCAGAAAGAGAAC
AACAAATATTAAAAAGATGT
CAACTTCCAGCATGAGTTGA
TTGTGACTTGTGGCGCTGGG
AAAATGGAGCATGCCCGAAG
CATTAGTAGTGTGAGACGTA
ACCATCACAAAACCTCCGAG
GCCAATAAGAGGTGACTTGT
c1 c2 c3
Gene1 3 1 2
Gene2 2 1 3
Gene3 8 2 1
nFeature
nCount
MT
c1 c2 c3
Gene1 0.91 0.73 1.01
Gene2 0.68 0.73 1.28
Gene3 1.59 1.15 0.63
Standardized Variance
Average Expression
PC_2
PC_1
tSNE_2
tSNE_1
Kupffer Cell
T Cell
Endothelial Cell
Dendritic Cell
Neutrophil
Epithelial Cell

Overview

DNA Sequencing



DNA Sequencing

- Problem: Converting a DNA molecule to a string
  - string: sequence of bases (A, T, C, G)

- Many possible sequencing techniques exist:

illumina®

PACBIO®

Oxford NANOPORE™ Technologies

FASTA file

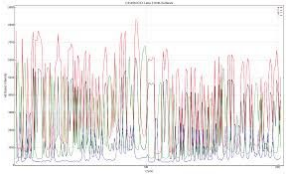
- Can be used to store DNA/protein sequence in a text-based file
- Mainly used to store large genomic sequences
- Header (lines that start with '>') + DNA sequence
- Alphabet: A, C, G, T, N

```
>NC 005248.1:3950-4810 Escherichia coli plasmid pIGAL1, complete sequence
ATGAGTATTCAACATTTCCGTGTCGCCCTTATTCCCTTTTTTGCGGCATTTTGCCTTCCTGTTTTTGCTC
ACCCAGAAACGCTGGTGAAAGTAAAAGATGCTGAAGATCAGTTGGGTGCACGAGTGGGTTACATCGAACT
GGATCTCAACAGCGGTAAGATCCTTGAGAGTTTTCGCCCCGAAGAACGTTTTCCAATGATGAGCACTTTT
AAAGTTCTGCTATGTGGCGCGGTATTATCCCGTGTTGACGCCGGGCAAGAGCAACTCGGTCGCCGCATAC
ACTATTCTCAGAATGACTTGGTTGAGTACTCACCAGTCACAGAAAAGCATCTTACGGATGGCATGACAGT
AAGAGAATTATGCAGTGCTGCCATAACCATGAGTGATAACACTGCGGCCAACTTACTTCTGACAACGATC
GGAGGACCGAAGGAGCTAACCGCTTTTTTGCACAACATGGGGGATCATGTAACTCGCCTTGATCGTTGGG
AACCGGAGCTGAATGAAGCCATACCAAACGACGAGCGTGACACCACGATGCCTGCAGCAATGGCAACAAC
GTTGCGCAAACTATTAACTGGCGAACTACTTACTCTAGCTTCCCGGCAACAATTAATAGACTGGATGGAG
```

From signal to string data







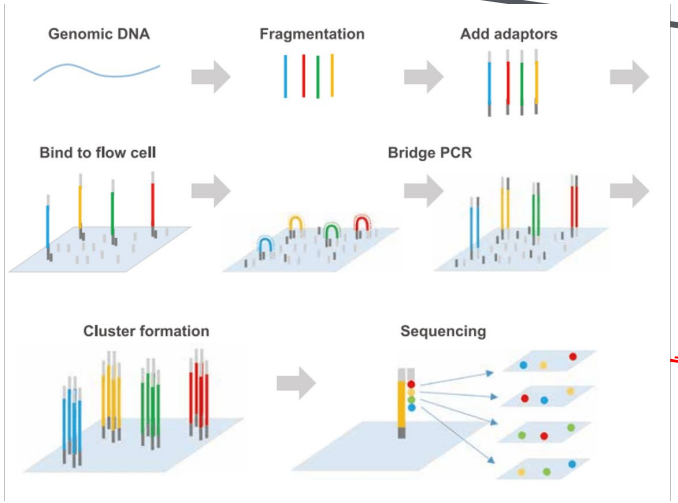
Signal: Ilumina HiSeq

How we can measure the quality of these algorithms?

Base detection algorithms

ACCTTCGAACGGCGGGGGTTACAA

From signal to string data



ACCTTCGAACGGCGGGGGTTACAA

FASTQ

Phred quality scores Q are defined as a property which is logarithmically related to the base-calling error probabilities P.[2]

$$Q = -10 \log_{10} P$$

or

$$P = 10^{\frac{-Q}{10}}$$

For example, if Phred assigns a quality score of 30 to a base, the chances that this base is called incorrectly are 1 in 1000.

Phred quality scores are logarithmically linked to error probabilities

Phred Quality Score	Probability of incorrect base call	Base call accuracy
10	1 in 10	90%
20	1 in 100	99%
30	1 in 1000	99.9%
40	1 in 10,000	99.99%
50	1 in 100,000	99.999%
60	1 in 1,000,000	99.9999%

The phred quality score is the negative ratio of the error probability to the reference level of $P = 1$ expressed in Decibel (dB).

https://en.wikipedia.org/wiki/Phred_quality_score

FASTQ

- Also text-based. Mainly used to store short DNA sequences (reads) from NGS-based experiments.
- Line 1: Begins with '@' and is followed by a an identifier.
- Line 2: DNA sequence.
- Line 3: Begins with '+' and is optionally followed by the same sequence identifier (and any description) again.
- Line 4: Quality values for the sequence in Line 2, and must contain the same number of symbols as the sequence.

```
@NS500746:56:HLY2MAFXX:1:11101:10096:1016 1:N:0:1
GCCTGNCGCATTGCATTCATCAAACGCTGAATAGCAAAGCCTCTACGCGATTTCATAGTGGAGGCCTCCAGCAATCTTGAACACTC
ATCCTTAATACCTTTCTTTTTGGGGTAATTATACTCATCGCGAATATCCTTAAGAGGGCGTTCAGCAGCCAGCTTGCGG
+
AAAAA#EEEEEEEEEAEE/AEEEAEEEEEEEEEEAEE/EEEEEEEEEEA/EE<AEEEEEEEEEEEEAEAEAE/EEEEEE/E<AEE
<EE/EEE//E/AEEEEE<AAEEEEEEEEEEAEE/EAAA/AEAEEEEEE/EEEAA<AAEA<EEAEAEEE</AE<A/A<<<
```

FASTQ

- Also text-based. Mainly used to store short DNA sequences (reads) from NGS-based experiments.
- Line 1: Begins with '@' and is followed by a an identifier.
- Line 2: DNA sequence.
- Line 3: Begins with '+' and is optionally followed by the same sequence identifier (and any description) again.
- Line 4: Quality values for the sequence in Line 2, and must contain the same number of symbols as the sequence.

```
@NS500746:56:HLY2MAFXX:1:11101:10096:1016 1:N:0:1
GCCTGNCGCATTGCATTCATCAAACGCTGAATAGCAAAGCCTCTACGCGATTTCATAGTGGAGGCCTCCAGCAATCTTGAACACTC
ATCCTTAATACCTTTCTTTTTGGGGTAATTATACTCATCGCGAATATCCTTAAGAGGGCGTTCAGCAGCCAGCTTGCGG
+
AAAAA#EEEEEEEEEAEE/AEEEAEEEEEEEEEEAEE/EEEEEEEEEEA/EE<AEEEEEEEEEEEEAEAEAE/EEEEEE/E<AEE
<EE/EEE//E/AEEEEE<AAEEEEEEEEEEAEE/EAAA/AEAEEEEEE/EEEAA<AAEA<EEAEAEEE</AE<A/A<<<
```

FASTQ

- Also text-based. Mainly used to store short DNA sequences (reads) from NGS-based experiments.
- Line 1: Begins with '@' and is followed by a an identifier.
- Line 2: DNA sequence.
- Line 3: Begins with '+' and is optionally followed by the same sequence identifier (and any description) again.
- Line 4: Quality values for the sequence in Line 2, and must contain the same number of symbols as the sequence.

```
@NS500746:56:HLY2MAFXX:1:11101:10096:1016 1:N:0:1
GCCTGNCGCATTGCATTCATCAAACGCTGAATAGCAAAGCCTCTACGCGATTTCATAGTGGAGGCCTCCAGCAATCTTGAACACTC
ATCCTTAATACCTTTCTTTTTGGGGTAATTATACTCATCGCGAATATCCTTAAGAGGGCGTTCAGCAGCCAGCTTGCGG
+
AAAAA#EEEEEEEEEAEE/AEEEAEEEEEEEEEEAEE/EEEEEEEEEEA/EE<AEEEEEEEEEEEEAEAEAE/EEEEEE/E<AEE
<EE/EEE//E/AEEEEE<AAEEEEEEEEEEAEE/EAAA/AEAEEEEEE/EEEAA<AAEA<EEAEAEEE</AE<A/A<<<
```

FASTQ

- Also text-based. Mainly used to store short DNA sequences (reads) from NGS-based experiments.
- Line 1: Begins with '@' and is followed by a an identifier.
- Line 2: DNA sequence.
- Line 3: Begins with '+' and is optionally followed by the same sequence identifier (and any description) again.
- Line 4: Quality values for the sequence in Line 2, and must contain the same number of symbols as the sequence.

```
@NS500746:56:HLY2MAFXX:1:11101:10096:1016 1:N:0:1
GCCTGNCGCATTGCATTCATCAAACGCTGAATAGCAAAGCCTCTACGCGATTTCATAGTGGAGGCCTCCAGCAATCTTGAACACTC
ATCCTTAATACCTTTCTTTTTGGGGTAATTATACTCATCGCGAATATCCTTAAGAGGGCGTTCAGCAGCCAGCTTGCGG
+
AAAAA#EEEEEEEEEAEE/AEEEAEEEEEEEEEEAEE/EEEEEEEEEEA/EE<AEEEEEEEEEEEEAEAEAE/EEEEEE/E<AEE
<EE/EEE//E/AEEEEE<AAEEEEEEEEEEAEE/EAAA/AEAEEEEEE/EEEAA<AAEA<EEAEAEEE</AE<A/A<<<
```

FASTQ Evaluation – FastQC

Fastq files can be very big with millions of (long) reads. Infeasible to investigate.

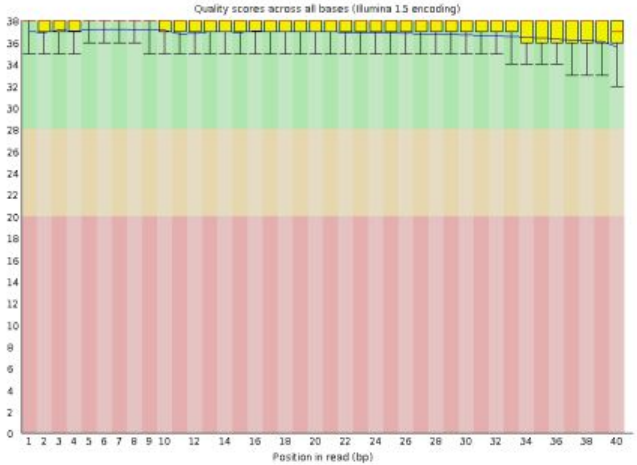
- Phred-Score hard to read in ASCII form.
- FastQC (usually provided by NGS core facilities)
- Tool to analyse quality of reads from sequencing.
- Indicate problems in library preparation or sequencing steps.

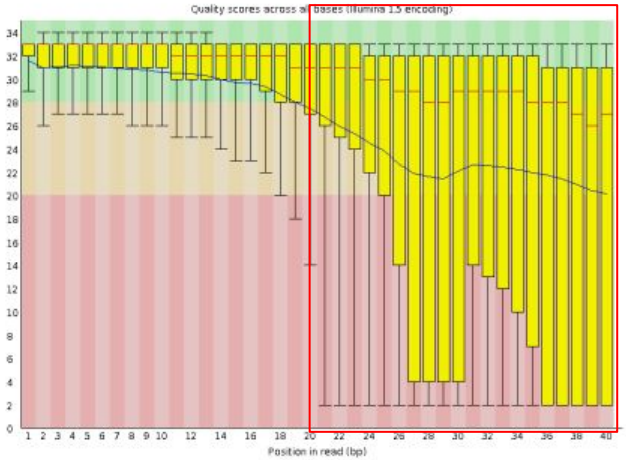
Example – **good quality sequences**
http://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html
Example – **bad quality sequences**
http://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html

Help Documents:
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/

FASTQ Evaluation – FastQC

Sequencing quality decreases with size.

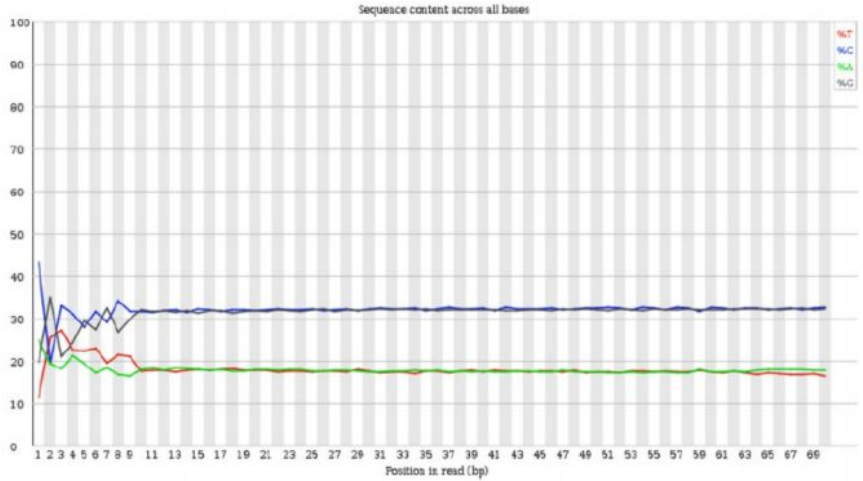




Solution: trim ends of reads, if quality is low.

FASTQ Evaluation – FastQC

Read position sequence bias.



Solution: Trim starts of reads.

Hands-on time

- Download **data1.zip** from the lecture website.
- Use FastQC to analyze the data:
    - **create** new directory "**fastqc_results**"
    - read the documentation of FastQC to understand how to export the files to the new directory:
    - fastqc -h
    - What do you see?
      What is the overall quality?
      Do we have any adapters?
- Trim the reads from the identified adapter using **trim_galore** (trim_galore –help) in a new folder "**trimmed_results**". Again analyze the fastq. What do you see? Are the adapters gone?

trim_galore guide:
TrimGalore/Trim_Galore_User_Guide.md at master

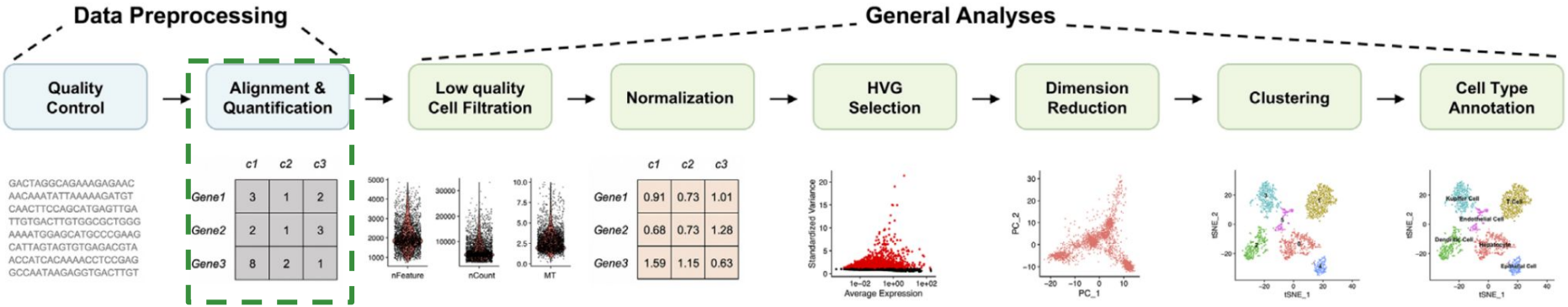
Hands-on time

1. ● fastqc -o fastqc_results/ ERR522959_1.fastq.gz ERR522959_2.fastq.gz

2. ● trim_galore –nextera -o trimmed_results/ ERR522959_1.fastq.gz ERR522959_2.fastq.gz
3. ● fastqc -o trimmed_results/ trimmed_results/ERR522959_1_trimmed.fq.gztrimmed_results/ERR522959_2_trimmed.fq.gz

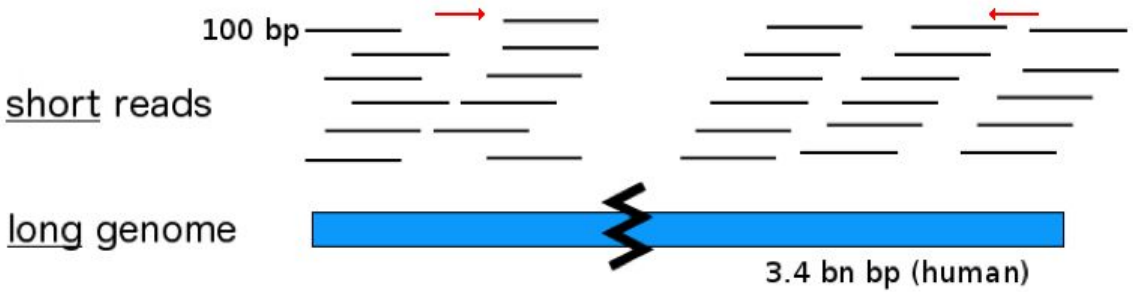
Overview

DNA Sequencing



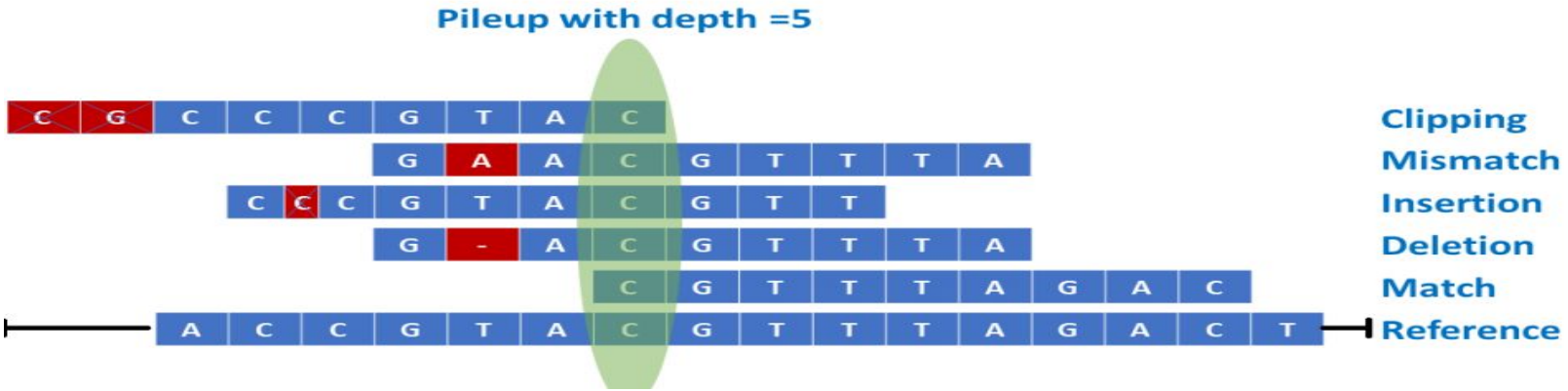
Alignment

Usually very large genomes (with repetitive regions) and very small reads.



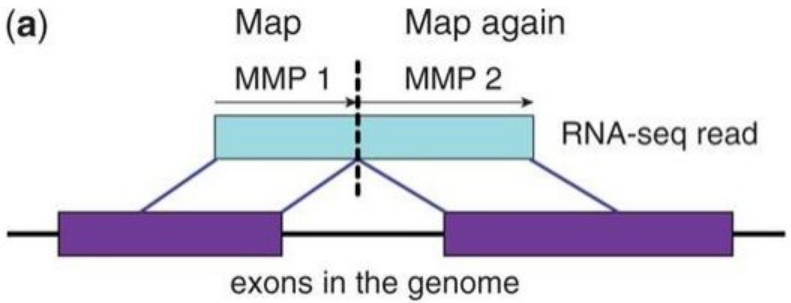
Alignment

Problem: aligning DNA sequence to a reference genome.



STAR

STAR allows a sequence to be split and aligned to different exons



Source: Dobin et al. (2013), Bioinformatics.

SAM File

- Sequence Alignment/Map format.
- Text-based tab-delimited file.
- Header + records (aligned reads)
  - Information: https://samtools.github.io/hts-specs/SAMv1.pdf

HEADER RECORDS

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

SAM File

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	Query template NAME
2	FLAG	Int	[0,2^{16}-1]	bitwise FLAG
3	RNAME	String	*|[!-()+-<>-~][!-~]*	Reference sequence NAME
4	POS	Int	[0,2^{31}-1]	1-based leftmost mapping POSition
5	MAPQ	Int	[0,2^{8}-1]	MAPping Quality
6	CIGAR	String	*|([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	*|=|[!-()+-<>-~][!-~]*	Ref. name of the mate/next read
8	PNEXT	Int	[0,2^{31}-1]	Position of the mate/next read
9	TLEN	Int	[-2^{31}+1,2^{31}-1]	observed Template LENgth
10	SEQ	String	*|[A-Za-z=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

SAM File

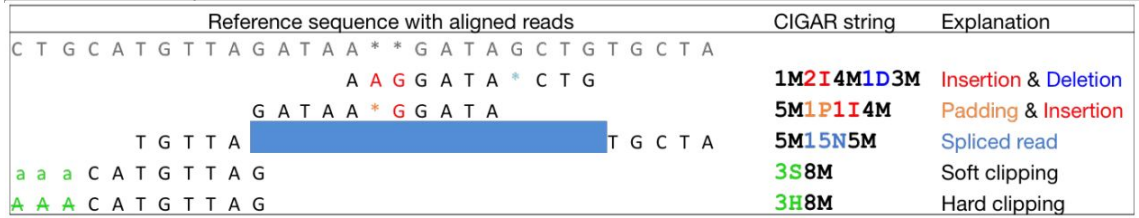
Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	Query template NAME
2	FLAG	Int	[0,2^{16}-1]	bitwise FLAG
3	RNAME	String	*|[!-()+-<>-~][!-~]*	Reference sequence NAME
4	POS	Int	[0,2^{31}-1]	1-based leftmost mapping POSition
5	MAPQ	Int	[0,2^{8}-1]	MAPping Quality
6	CIGAR	String	*|([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	*|=|[!-()+-<>-~][!-~]*	Ref. name of the mate/next read
8	PNEXT	Int	[0,2^{31}-1]	Position of the mate/next read
9	TLEN	Int	[-2^{31}+1,2^{31}-1]	observed Template LENgth
10	SEQ	String	*|[A-Za-z=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001   99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002    0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
r003    0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA       * SA:Z:ref,29,-,6H5M,17,0;
r004    0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
r003 2064 ref 29 17 6H5M       *  0   0 TAGGC             * SA:Z:ref,9,+,5S6M,30,1;
r001  147 ref 37 30 9M         =  7 -39 CAGCGGCAT         * NM:i:1
```

SAM File

CIGAR (Concise idiosyncratic gapped alignment report string)

Operator	Description
M	alignment match(can be a sequence match or mismatch
I	inserting to the reference
D	deletion from the reference
N	kkip region from the reference
S	soft clipping(clipped sequence present in SEQ)
H	hard clipping(clipped sequence NOT present in SEQ)
P	padding(silence from the reference)
=	sequence match
X	sequence mismatch



BAM File

- Binary Alignment/Map format – compressed version of SAM.
- Compression: BGZF block compression.
- Efficient random access: UCSC bin/chunk scheme.
- BAI index files.
- More Information:
  http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2723002/
  http://www.ncbi.nlm.nih.gov/pmc/articles/PMC186604/

In summary

Reference sequence (FASTA)

```
>chr1
TACCTCCAGGGGGCATCCTCCCCCCCAATTCG
AAACACAATCGTAGCCCCTGGCACTACCTATG
TGTGTCAATTCGGAGAGAGAGAGATTCACGAA
AAAAAAGTCTGGACTCAACTAGGATACACACA
TTCGGCTACAGATACCAAAAAAAAAAAAAAAA
AAATTTTCACCATTGAGGCACCACCTTCTCGT
CGCTGCGTCGCTCTGCTCGCTTCGGCTAAAAA
TTCGCGCAATACATTCGGCTACAGATACCAAA
```

Unaligned reads

```
@seq1
ATTCGAAACA...
+
DDED88(999...
@seq2
CCCCGTTTCA...
+
AAC887BBAC...
```

Alignment/ Mapping

SAM/BAM format aligned reads

```
seq1   99     1      3666901       60
149M   =      3666935       185
ATTCGAAACA...DDED88(999    MC:Z:151M
MD:Z:149      RG:Z:15-0017315_1   NM:i:0
MQ:i:60       AS:i:149      XS:i:44
seq2   147    1      3666935       60
151M   =      3666901       -185
CCCCGTTTCA...AAC887BBAC...MC:Z:149M
MD:Z:151      RG:Z:15-0017315_1   NM:i:0
MQ:i:60       AS:i:151      XS:i:59
```

Samtools

- Provides various utilities for manipulating alignments in the SAM format.

- Tools useful for quality check and bias correction.

- More Information:
  Paper: http://www.ncbi.nlm.nih.gov/pubmed/19505943
  Website: http://samtools.sourceforge.net/

Hands-on time

- Download data2.zip from the lecture website.

- Use STAR to align the reads to the supplied small reference genome (smaller_reference.fa) and output sam file

- FIRST! Index the genome: STAR --runThreadN 4 --runMode genomeGenerate --genomeDir output_dir/ --genomeFastaFiles smaller_reference.fa

  - STAR –help # for manuals

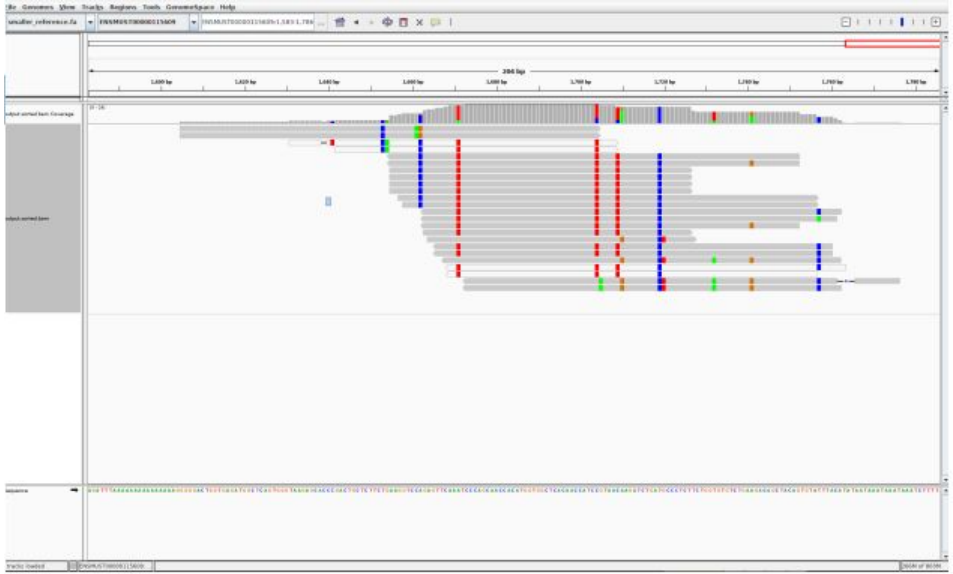
- Convert the SAM file to BAM (samtools view –help)

- Sort and index (samtools sort; samtools index)

IGV

- Tool for visualising sequences, reads and/or variants at genomic locations

- Open IGV. From menu: Genomes → Load genomes from file. → Navigate to genome fasta file
  - File →
  - Load from File →
  - Navigate to indexed Bam file.



Single Cell Sequencing

Single Cell Analysis

- Extract sequences from a specific cell for the purpose of discovering differences in gene expression level

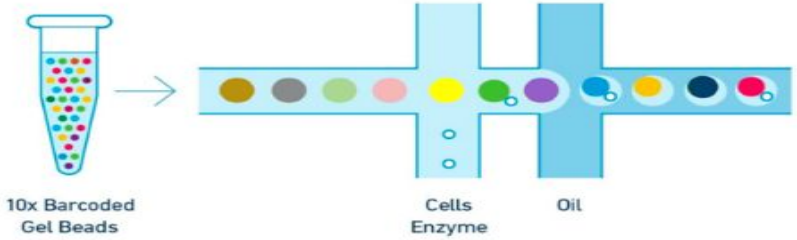
- Every sample is prepared by artificially adding a barcode and (preferably) Unique Molecule Identifier (UMI)

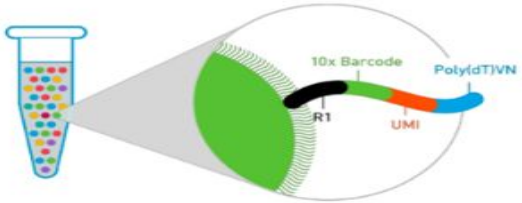
- All molecules from the same batch have the same barcode

- Every individual molecule has a separate UMI

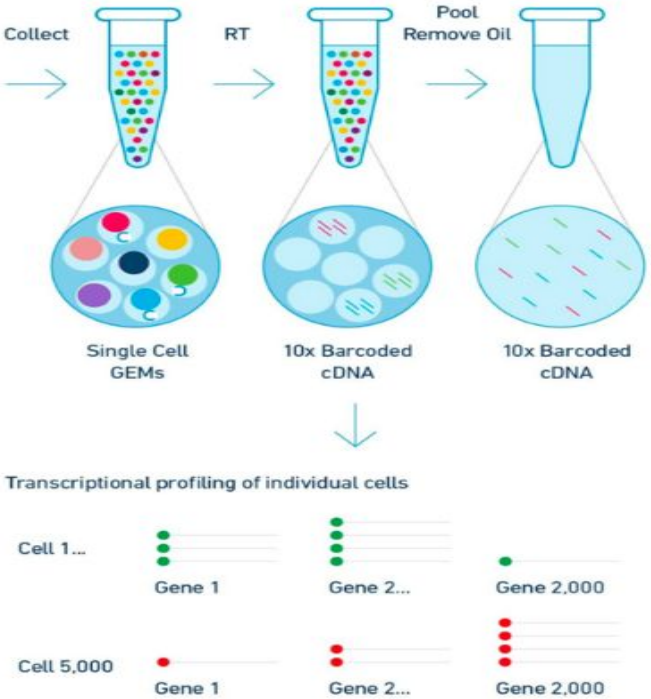
- Because of sequencing errors, we need to make sure that we can correct small amount of bases (1-2) and still have the same barcode – by maximizing the Hamming distance

Single Cell Analysis

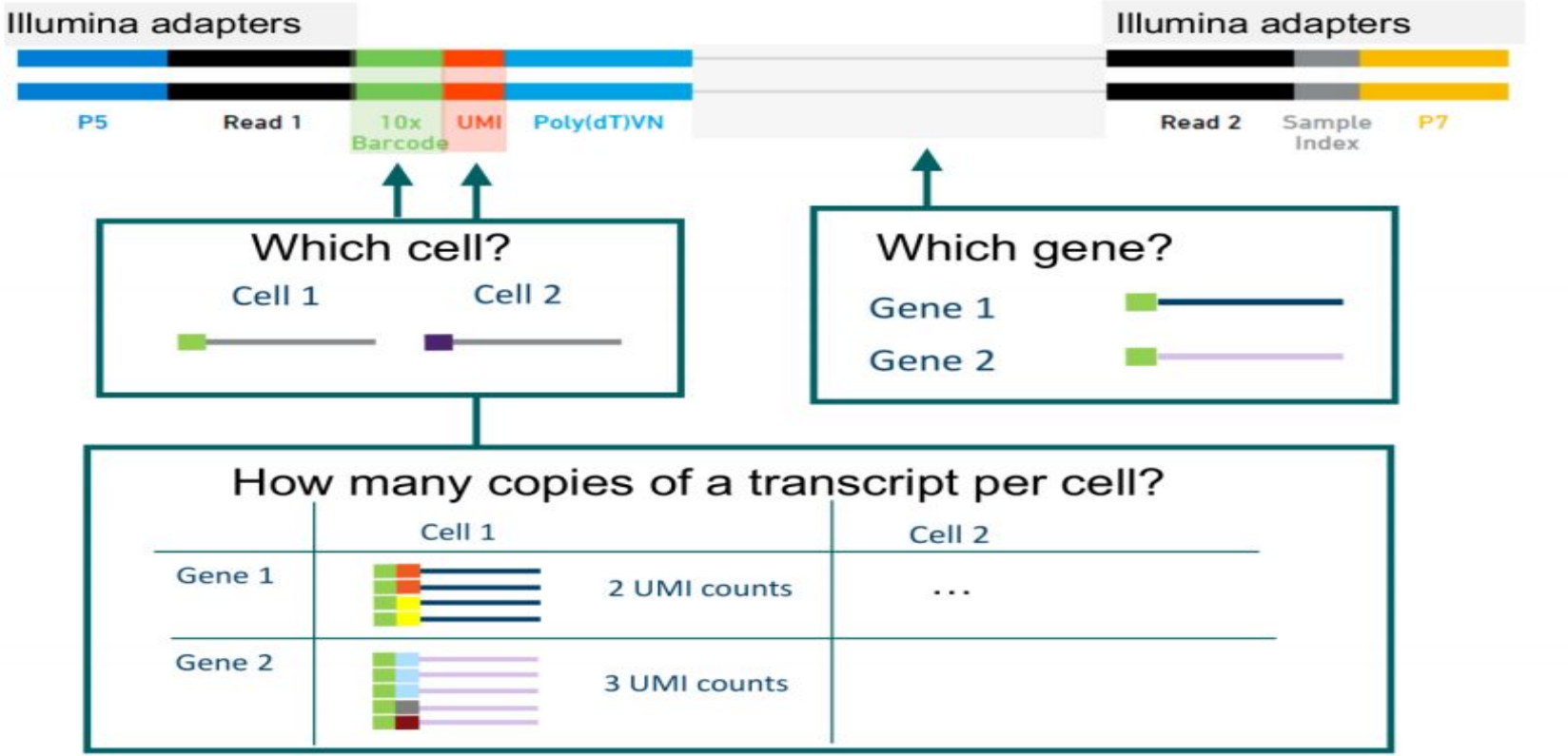




- Input: Single cells in suspension + 10x Gel Beads and Reagents
- Output: Digital gene expression profiles from every partitioned cell



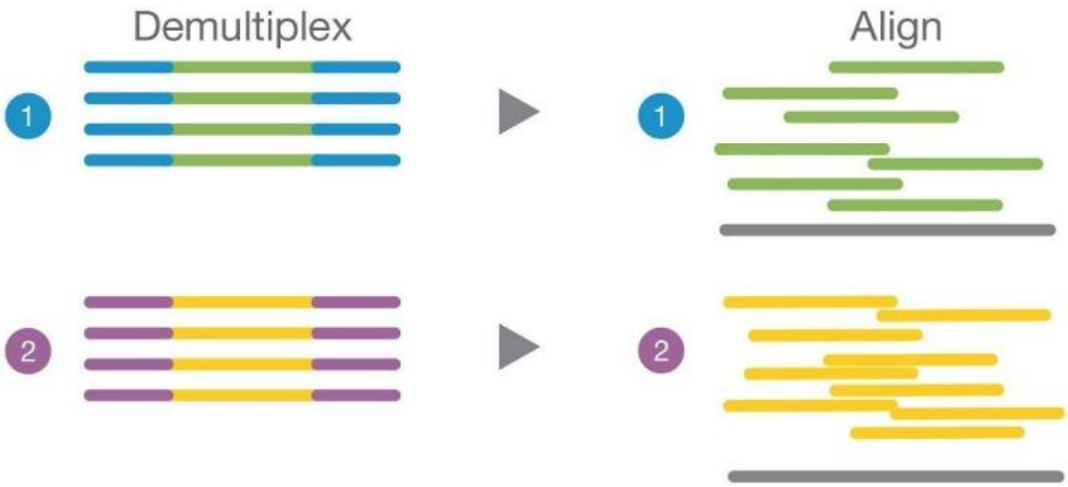
Demultiplexing



Source: 10x genomics

Demultiplexing

Distinguishing different DNA samples based on added barcode



Source: Illumina webinar

Hamming Distance

- A measure of similarity between two strings of equal length

- Measured by the amount substitutions needed to derive the second string from the first

B	I	O	I	N	F	O	R	M	A	T	I	C	S
B	I	O	I	N	F	O	R	M	A	T	I	K	K
F	O	R	M	S	B	I	O	L	O	G	I	E	S

H = 2

H = 12

Hamming Distance - Example

A	C	T	G	G	G	A	C	G	T
G	A	C	T	T	A	C	G	G	A
A	C	T	G	G	G	A	C	G	A
T	A	T	C	A	G	C	C	G	A
T	A	C	T	T	G	C	G	G	A

Barcode 1

Barcode 2

Read 1 – H(1) = 1;

Read 2 – H(1) = 6;

Read 3 – H(1) = 7;

Barcode 1

Barcode 2

Read 1 - H(1) = 1; H(2) = 9

Read 2 - H(1) = 6; H(2) = 6

Read 3 - H(1) = 7; H(2) = 2

Designing a set of equidistant barcodes for optimal error correction is NP-complete problem

Demultiplexing

- Demultiplexing both:
  - Barcode
  - UMI (Unique Molecule Identifier)

- Usually UMI is added to read of the paired read.

- This results in one Fastq File per barcode

Demultiplexing - Example

- For simplicity a demultiplexing script is provided as well as sample data - data3.zip.

Use it to extract demultiplexed reads and get familiar with the inputs and outputs.

mkdir data3/results

python3 ../script/demultiplexing.py -b data3/10cells_barcodes.txt -f data3/10cells_read1.fastq -r data3/10cells_read2.fastq -o data3/results/

Expression Matrix

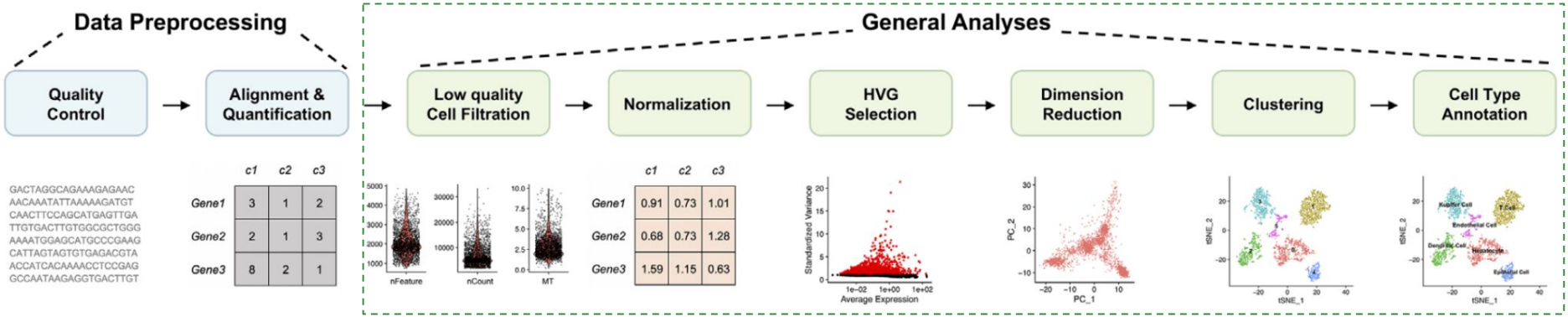
- After performing QC we align the reads and count UMIs for specific barcodes and positions to create an Expression Matrix (mxn).
  n can vary to ~100 to ~ 1M

- Columns represent a cell
- Rows represent a gene (transpose used by some authors)

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

https://hbctraining.github.io/scRNA-seq/lessons/02_SC_generation_of_count_matrix.html

Data preprocessing



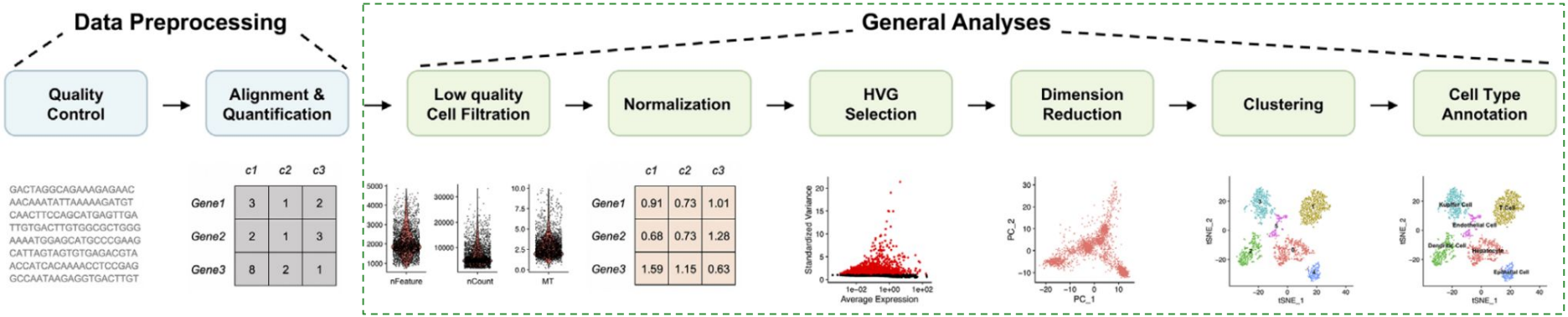
Scanpy

- A python package designed for higher level analysis and exploration of single-cell RNA-seq data.

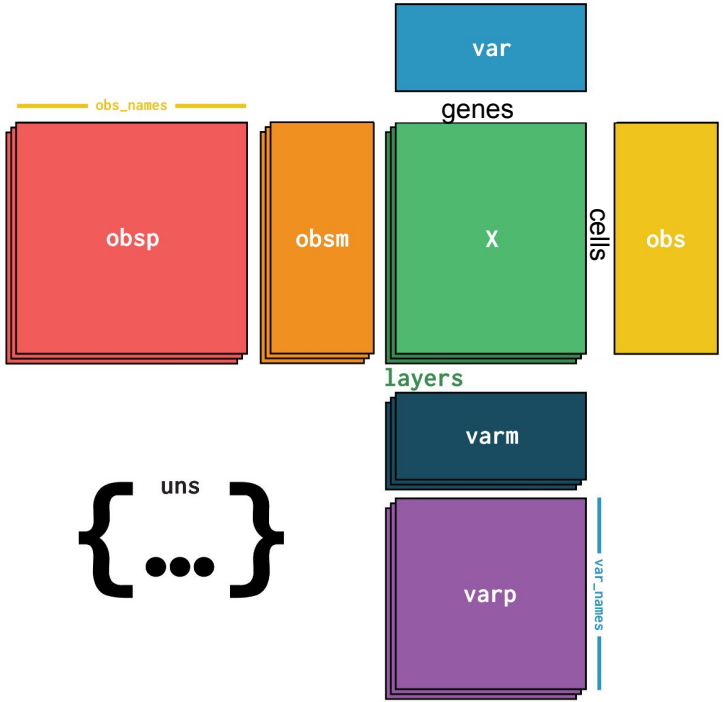
- Current version: 1.9.1

- Allows various functions like PCA and clustering and supports an array of different plotting capabilities.

Scanpy-pipeline

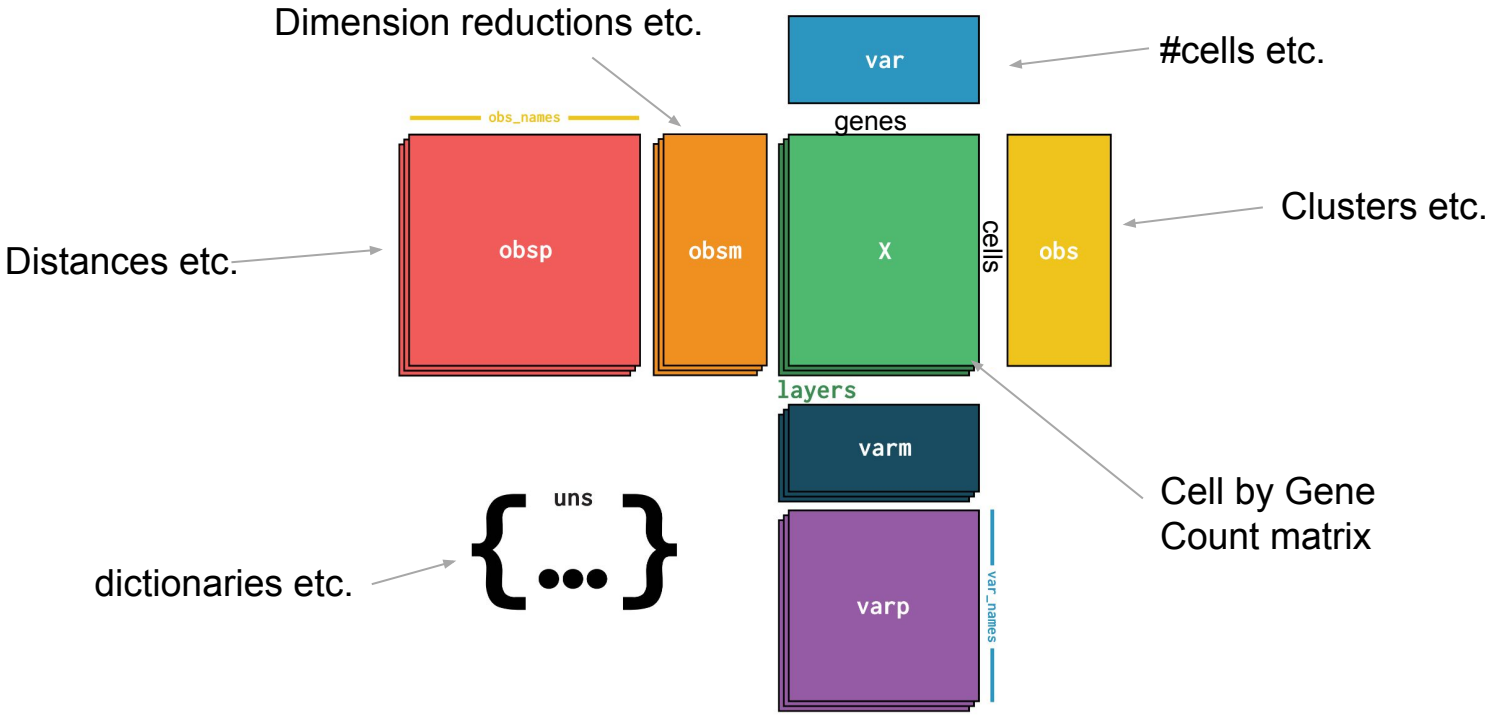


Scanpy-anndata



https://anndata.readthedocs.io/en/latest/

Scanpy-anndata



https://anndata.readthedocs.io/en/latest/

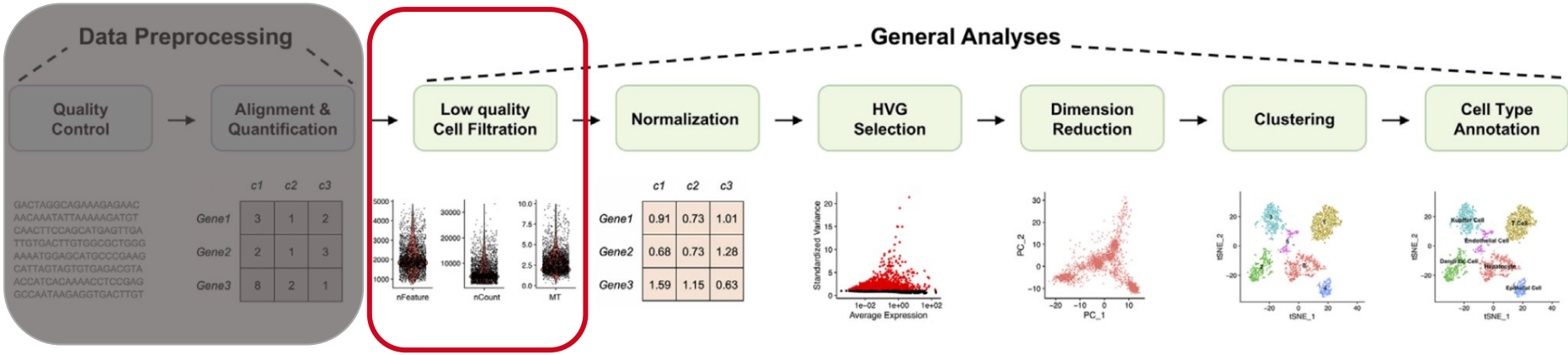
Scanpy-load data

```
import numpy as np
import pandas as pd
import scanpy as sc
import scanpy.external as sce
sc.settings.verbosity = 3             # verbosity: errors (0), warnings (1), info (2), hints (3)
sc.logging.print_header()
sc.settings.set_figure_params(dpi=100, facecolor='white')
```

Scanpy-load data

```
import numpy as np
import pandas as pd
import scanpy as sc
import scanpy.external as sce
sc.settings.verbosity = 3             # verbosity: errors (0), warnings (1), info (2), hints (3)
sc.logging.print_header()
sc.settings.set_figure_params(dpi=100, facecolor='white')


## load anndata from h5ad file
adata =  sc.read_h5ad("ifnb.h5ad")  # Reading the h5ad containing the data
adata.var_names_make_unique() # Making the data unique
```

Scanpy-pipeline



D Jovic, X Liang, H Zeng, L Lin, F Xu, Y Luo, 2022

Institute for Computational Genomics
01011011010
10100100101

RWTH AACHEN UNIVERSITY

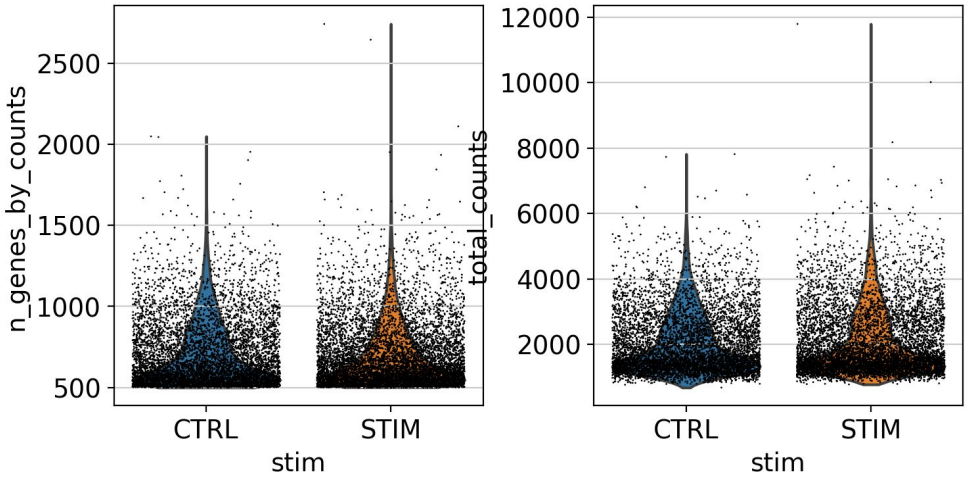
Scanpy-preprocessing

```
sc.pp.filter_cells(adata,
                   min_genes=200) ## min_genes: Minimal feature per cell
sc.pp.filter_genes(adata,
                  min_cells=3)   ## min_cells: Minimal cells per genes


sc.pp.calculate_qc_metrics(adata,
                           percent_top=None,
                           log1p=False,
                           inplace=True)
sc.pl.violin(adata, ['n_genes_by_counts', 'total_counts'],
                    groupby="stim", jitter=0.4, multi_panel=True)
```

Cells with too high (doublets) or low counts (not viable cells) can be artifacts!

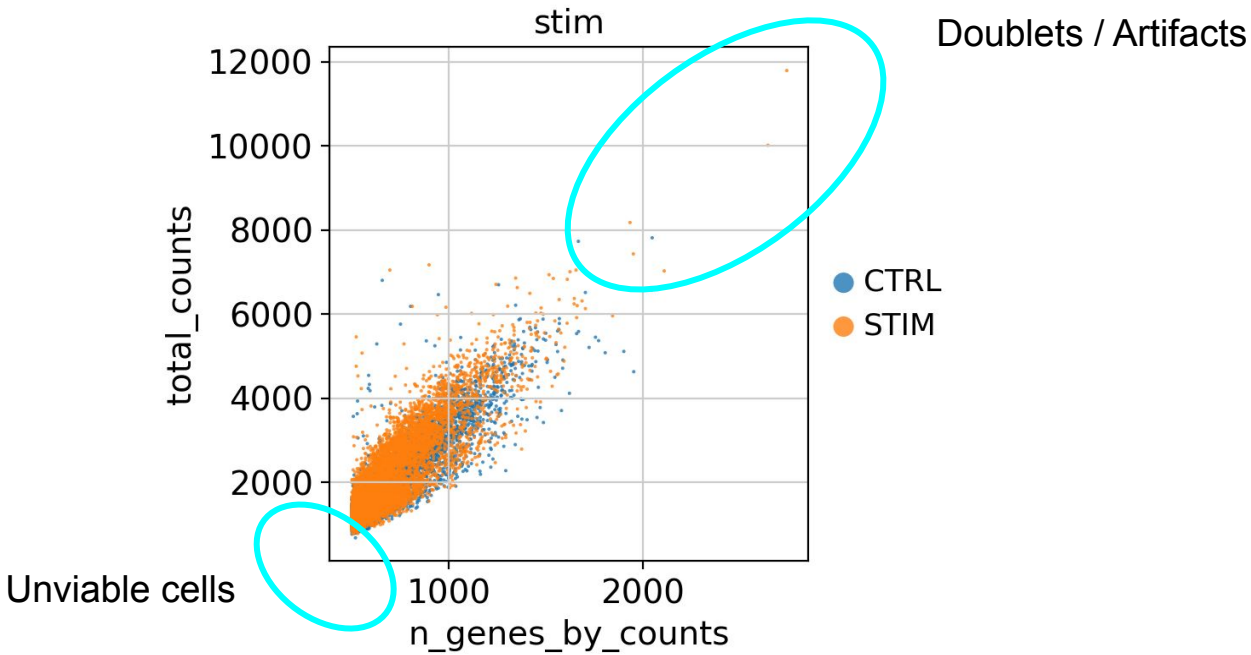
Scanpy-preprocessing

```
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)


sc.pp.calculate_qc_metrics(adata, percent_top=None, log1p=False, inplace=True)
sc.pl.violin(adata, ['n_genes_by_counts', 'total_counts'],
                    groupby="stim", jitter=0.4, multi_panel=True)
```



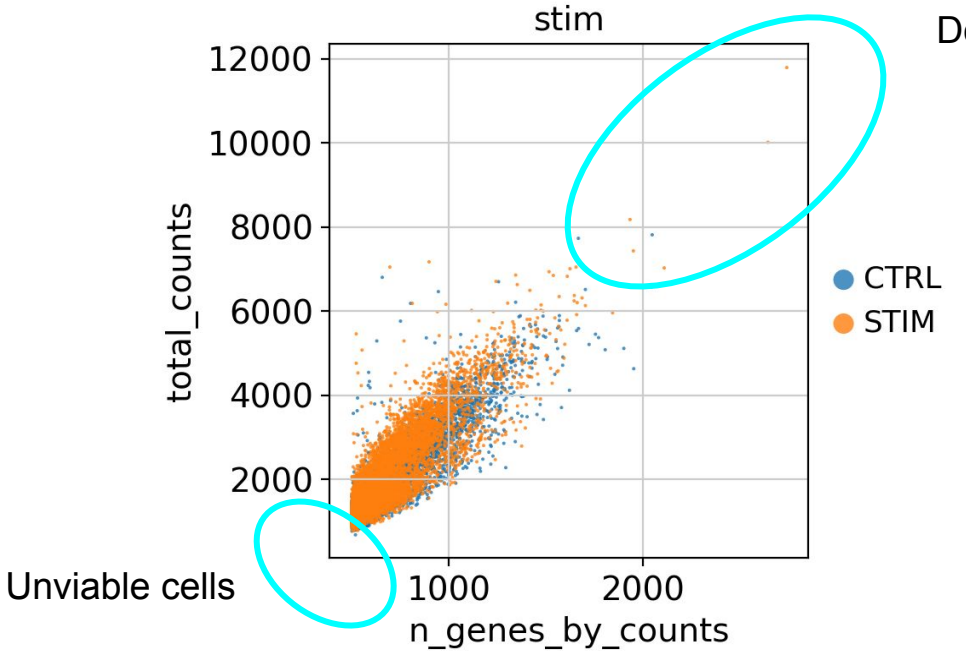
Scanpy-preprocessing

```
sc.pl.scatter(adata, x="n_genes_by_counts", y="total_counts", color="stim",alpha=0.8)
```

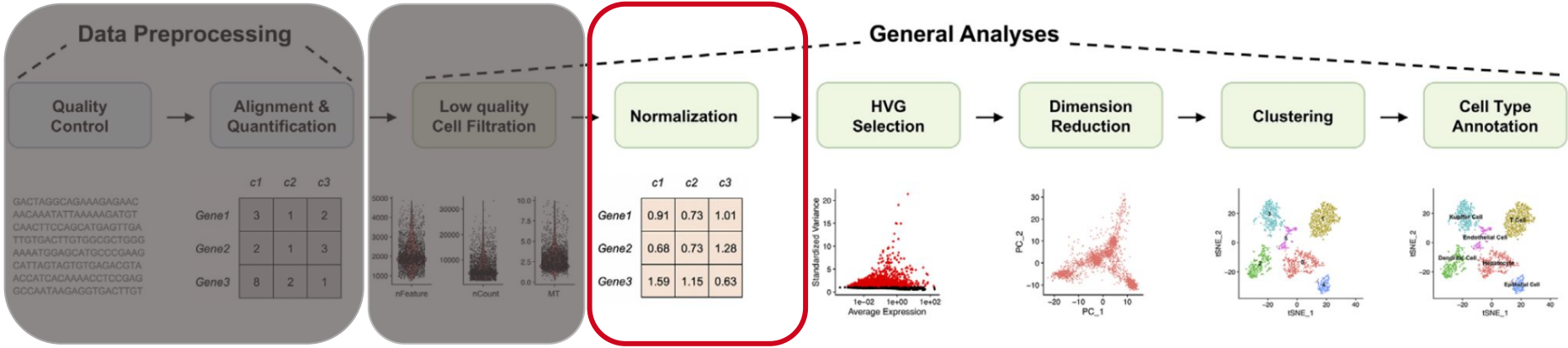


Scanpy-preprocessing

```
sc.pl.scatter(adata, x="n_genes_by_counts", y="total_counts", color="stim",alpha=0.8)
adata = adata[adata.obs.n_genes_by_counts < 2500, :]
```



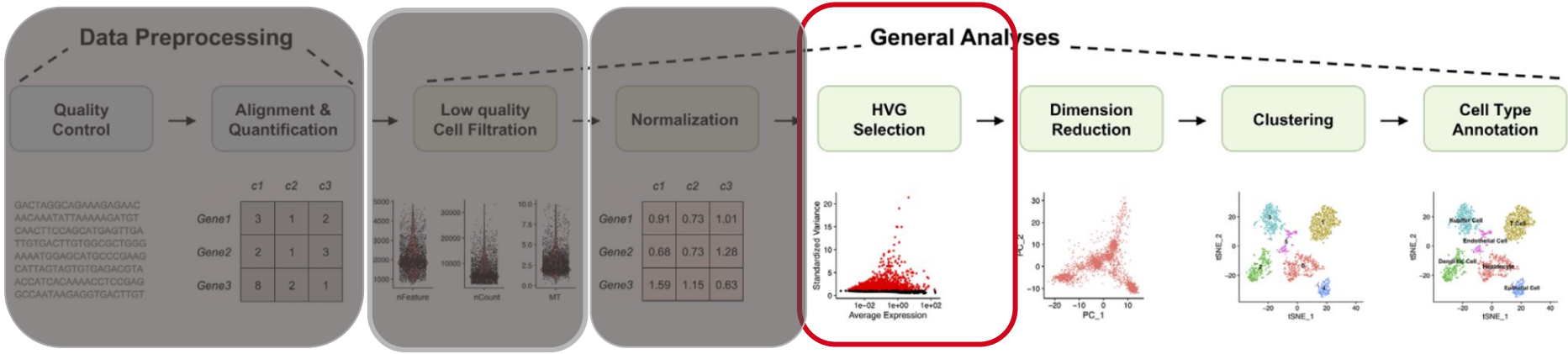
Scanpy-pipeline



Scanpy-Cell Normalization

```
# Normalization
sc.pp.normalize_total(adata,
                      target_sum=1e4)
sc.pp.log1p(adata)
```

Scanpy-pipeline

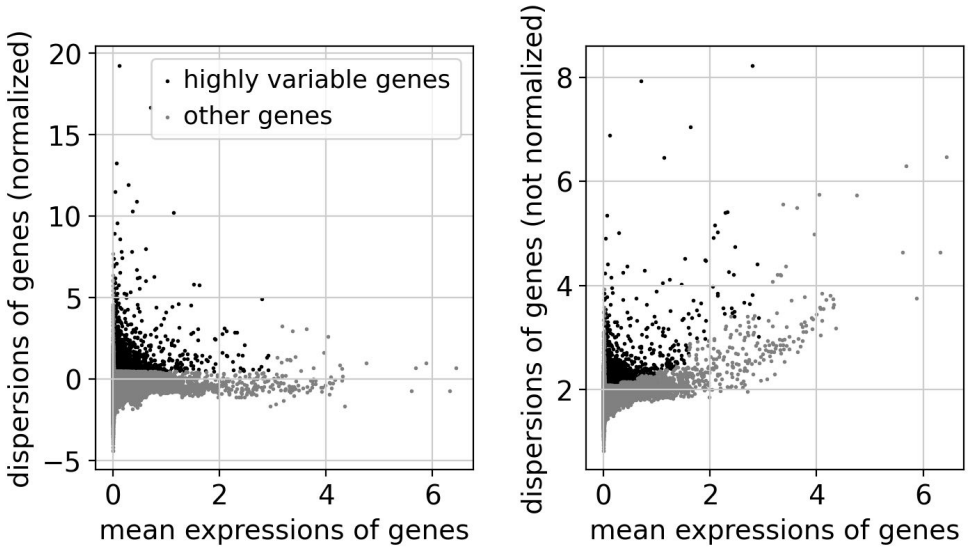


Scanpy-Features

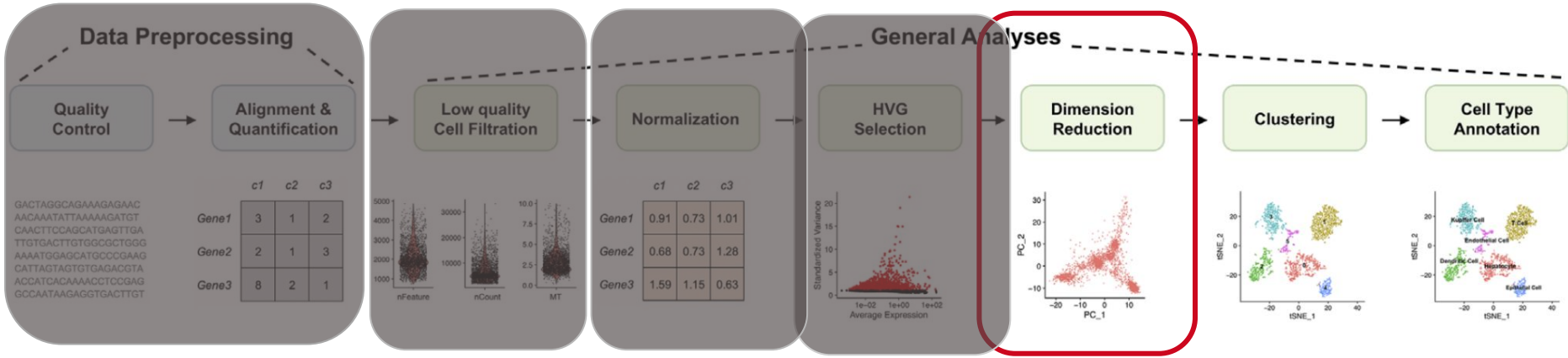
```
sc.pp.highly_variable_genes(adata,
                            min_mean=0.0125,
                            max_mean=3,
                            min_disp=0.5)
```

Scanpy-Features

```
sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_disp=0.5)
sc.pl.highly_variable_genes(adata)
```



Scanpy-pipeline

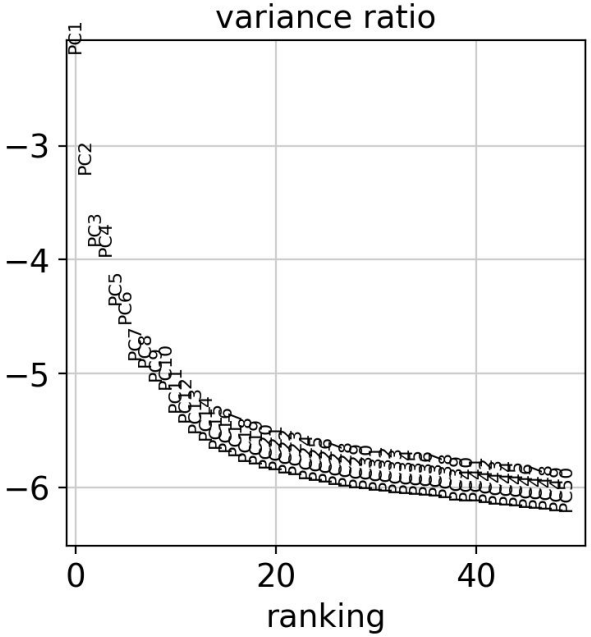


Scanpy-Dimesion reduction

```
sc.tl.pca(adata,
          svd_solver='arpack')
```

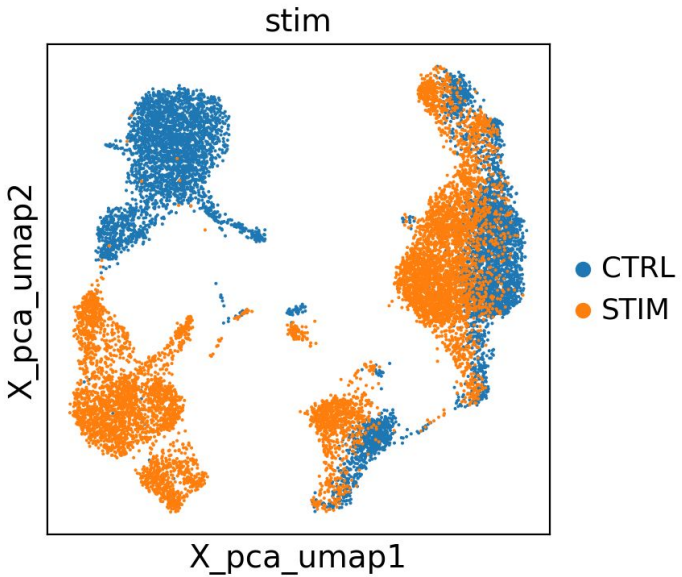
Scanpy-Dimension reduction

```
sc.tl.pca(adata, svd_solver='arpack')
sc.pl.pca_variance_ratio(adata, log=True, n_pcs=50)
```

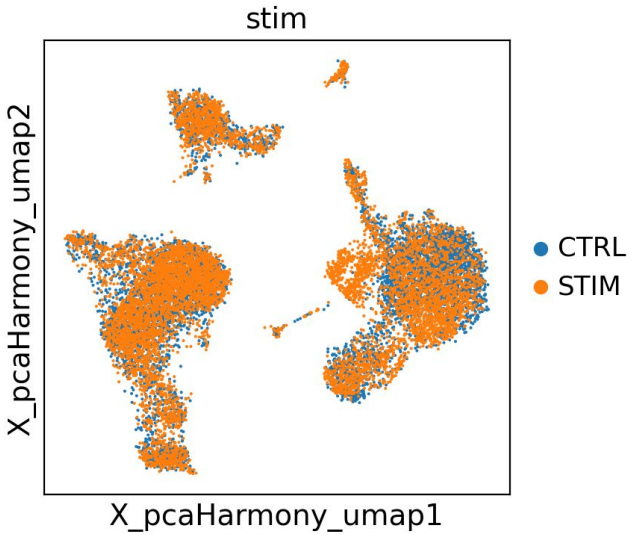


Scanpy-Data integration

Before data integration



After data integration

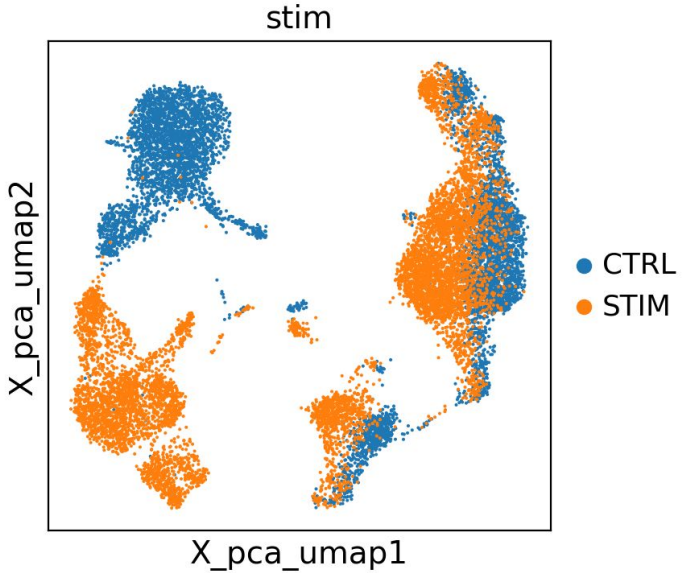


Scanpy-Uncorrected

```
sc.pp.neighbors(adata,
                n_neighbors=10,
                n_pcs=40,
                use_rep="X_pca")
sc.tl.umap(adata)
adata.obsm['X_pca_umap'] = adata.obsm['X_umap']
```

Scanpy-Uncorrected

```
sc.pp.neighbors(adata, n_neighbors=10, n_pcs=40, use_rep="X_pca")
sc.tl.umap(adata)
adata.obsm['X_pca_umap'] = adata.obsm['X_umap']
sc.pl.embedding(adata, color='stim', basis="X_pca_umap")
```



Scanpy-Data integration

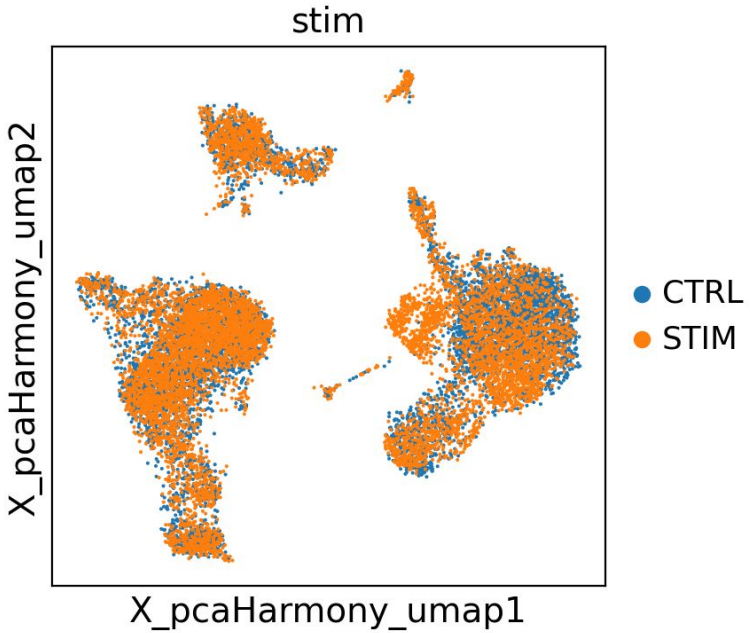
```
import harmonypy as hm
harmony_out = hm.run_harmony(data_mat=adata.obsm['X_pca'][:, 0:40],
                             meta_data=adata.obs,
                             vars_use="stim" )
adata.obsm['X_pca_harmony'] = harmony_out.Z_corr.T
```

Scanpy-Data integration

```
import harmonypy as hm
harmony_out = hm.run_harmony(data_mat=adata.obsm['X_pca'][:, 0:40],
                             meta_data=adata.obs,
                             vars_use="stim" )
adata.obsm['X_pca_harmony'] = harmony_out.Z_corr.T


sc.pp.neighbors(adata, n_neighbors=10, n_pcs=40, use_rep="X_pca_harmony")
sc.tl.umap(adata)
adata.obsm['X_pcaHarmony_umap'] = adata.obsm['X_umap']
```

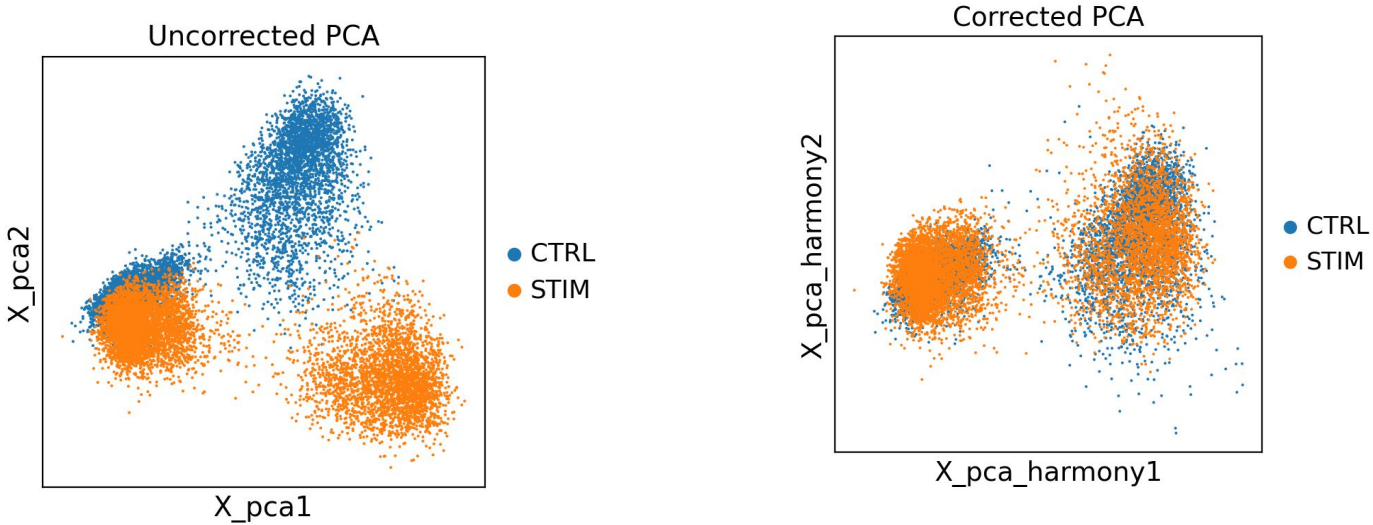
Scanpy-Data integration

```
sc.pl.embedding(adata, color='stim', basis="X_pcaHarmony_umap")
```

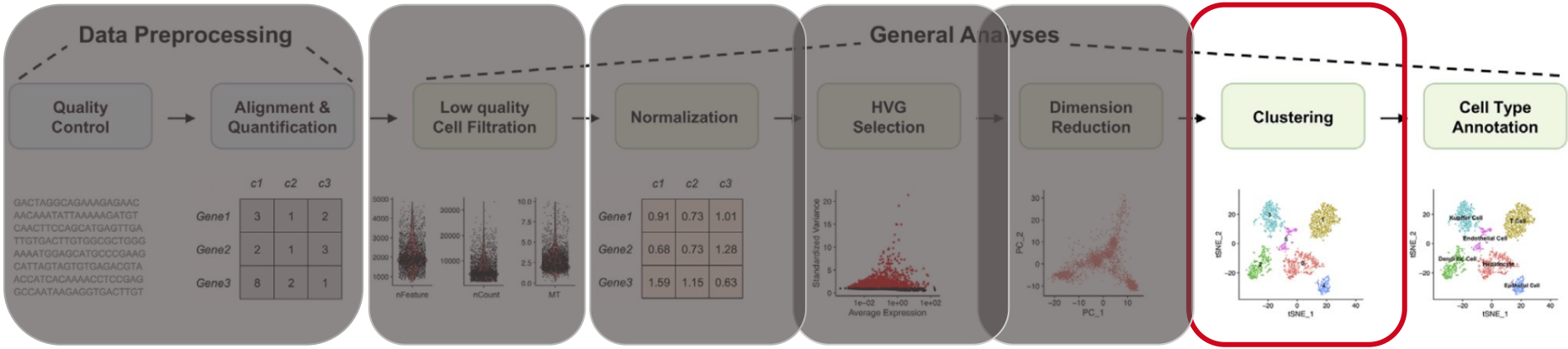


Scanpy-Data integration

```
sc.pl.embedding(adata, color='stim', basis='X_pca',title='Uncorrected PCA')
sc.pl.embedding(adata, color='stim', basis='X_pca_harmony',title='Corrected PCA')
```



Scanpy-pipeline



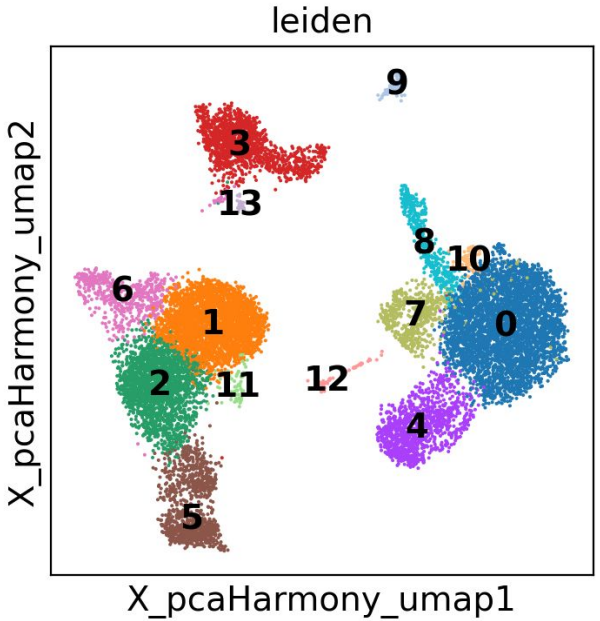
D Jovic, X Liang, H Zeng, L Lin, F Xu, Y Luo, 2022

Institute for Computational Genomics

RWTH AACHEN UNIVERSITY

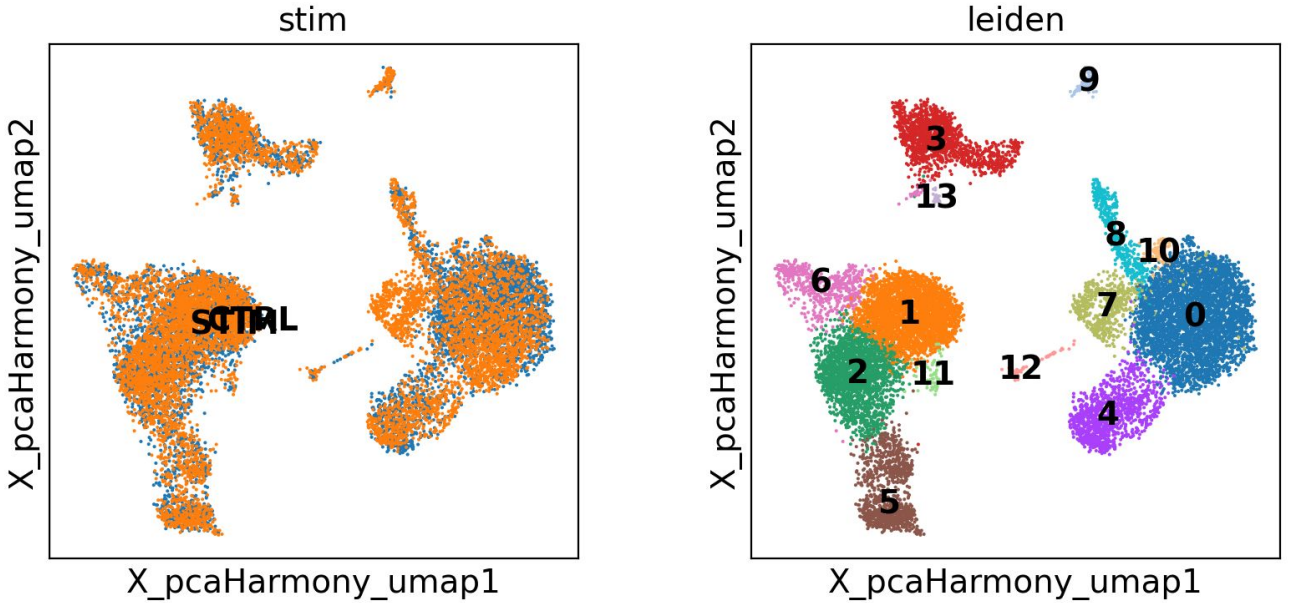
Scanpy-Cluster cells

```
sc.tl.leiden(adata, resolution=0.7)
sc.pl.embedding(adata, color='leiden', basis='X_pcaHarmony_umap', legend_loc="on data")
```



Scanpy-Cluster cells

```
sc.pl.embedding(adata, color=('stim','leiden'),
                basis='X_pcaHarmony_umap',legend_loc="on data")
```

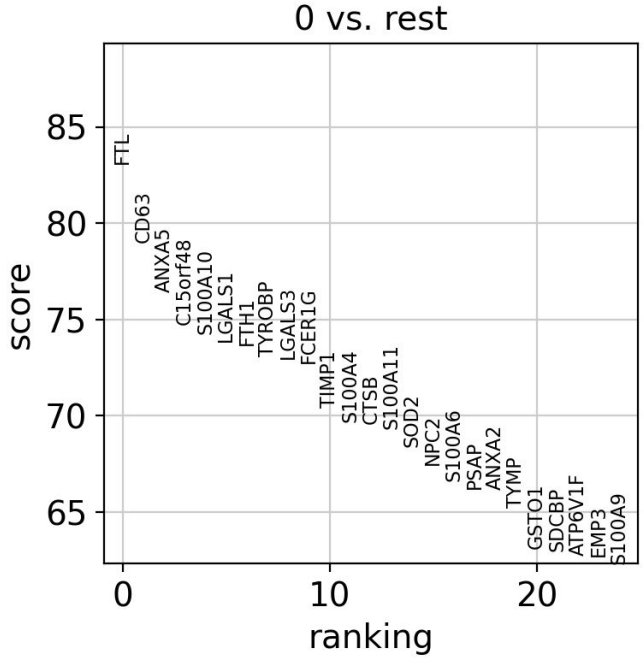


Scanpy-Identify markers for cells

```
sc.tl.rank_genes_groups(adata,
                        'leiden',
                         method='wilcoxon',
                         corr_method='bonferroni')
```

Scanpy-Identify markers for cells

```
sc.tl.rank_genes_groups(adata, 'leiden', method='wilcoxon',corr_method='bonferroni')
sc.pl.rank_genes_groups(adata, n_genes=25, sharey=False, groups='0')
```



Scanpy-Identify markers for cells

```
pd.DataFrame(adata.uns['rank_genes_groups']['names']).head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	FTL	RPS14	PABPC1	CD74	FCGR3A	NKG7	EIF1	APOBEC3B	HLA-DPB1	SEC61B	NKG7	PPBP	HBB	RPS4X
1	CD63	RPL32	TMSB4X	CD79A	MS4A7	GNLY	BTG1	TYMP	HLA-DRA	TSPAN13	GNLY	PF4	HBA1	SERPINB1
2	ANXA5	RPS6	RPS4X	RPL18A	CXCL16	GZMB	BIRC3	ISG15	HLA-DPA1	CD74	GZMB	GNG11	HBA2	GNB2L1
3	C15orf48	RPL13	CREM	HLA-DRA	LST1	CCL5	CACYBP	IDO1	HLA-DRB1	GZMB	CCL5	SDPR	ALAS2	RPS9
4	S100A10	RPL21	RPL3	RPL8	VMO1	APOBEC3G	UBC	CCL2	CD74	TXN	FTL	CCL5	SNCA	RPL15

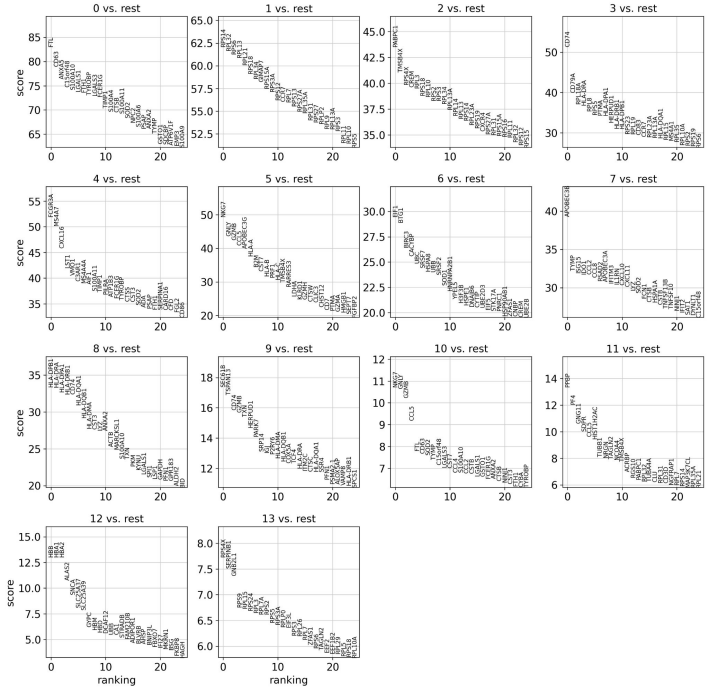
Scanpy-Identify markers for cells

```
result = adata.uns['rank_genes_groups']
groups = result['names'].dtype.names
pd.DataFrame(
    {group + '_' + key[:1]: result[key][group]
    for group in groups for key in ['names', 'pvals']}).head(5)
```

	0_n	0_p	1_n	1_p	2_n	2_p	3_n	3_p	4_n	4_p	...	9_n	9_p	10_n	10_p	11_n	11_p	12_n	12_p	13_n	13_p
0	FTL	0.0	RPS14	0.0	PABPC1	0.0	CD74	0.000000e+00	FCGR3A	0.0	...	SEC61B	9.059854e-68	NKG7	6.865625e-27	PPBP	8.005835e-41	HBB	6.392843e-39	RPS4X	1.030871e-14
1	CD63	0.0	RPL32	0.0	TMSB4X	0.0	CD79A	0.000000e+00	MS4A7	0.0	...	TSPAN13	7.916287e-64	GNLY	1.616282e-26	PF4	1.783033e-33	HBA1	6.412164e-39	SERPINB1	5.735683e-14
2	ANXA5	0.0	RPS6	0.0	RPS4X	0.0	RPL18A	6.051406e-303	CXCL16	0.0	...	CD74	9.317010e-57	GZMB	1.269984e-24	GNG11	2.046196e-26	HBA2	6.549051e-39	GNB2L1	1.684350e-13
3	C15orf48	0.0	RPL13	0.0	CREM	0.0	HLA-DRA	4.404436e-298	LST1	0.0	...	GZMB	1.227945e-55	CCL5	3.531466e-20	SDPR	1.003018e-23	ALAS2	5.516963e-27	RPS9	1.339722e-11
4	S100A10	0.0	RPL21	0.0	RPL3	0.0	RPL8	5.780738e-275	VMO1	0.0	...	TXN	1.183696e-53	FTL	5.706444e-15	CCL5	5.049139e-22	SNCA	6.391794e-21	RPL15	1.606794e-11

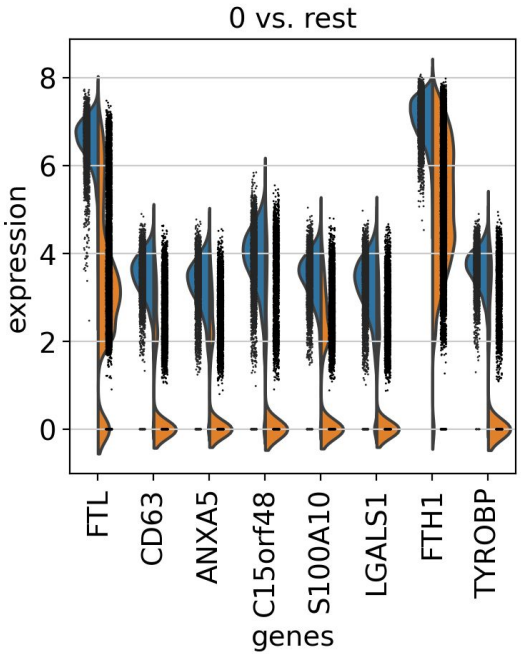
Scanpy-Identify markers for cells

```
sc.pl.rank_genes_groups(adata, n_genes=25, sharey=False)
```



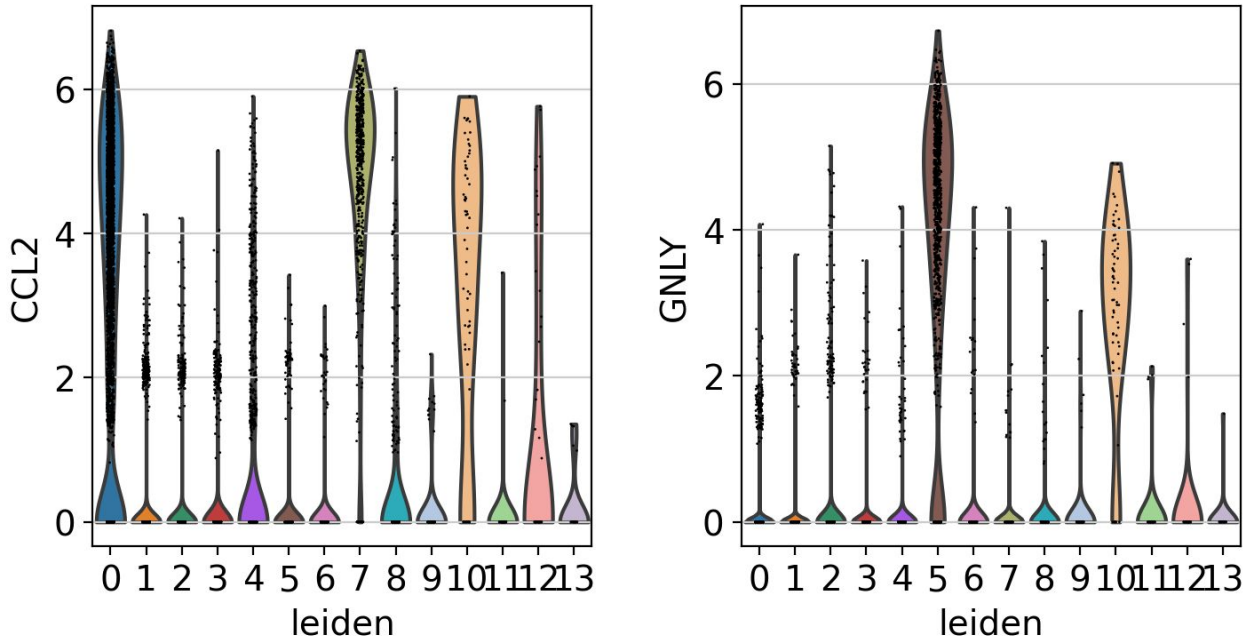
Scanpy-Identify markers for cells

```
sc.pl.rank_genes_groups_violin(adata, groups='0', n_genes=8)
```



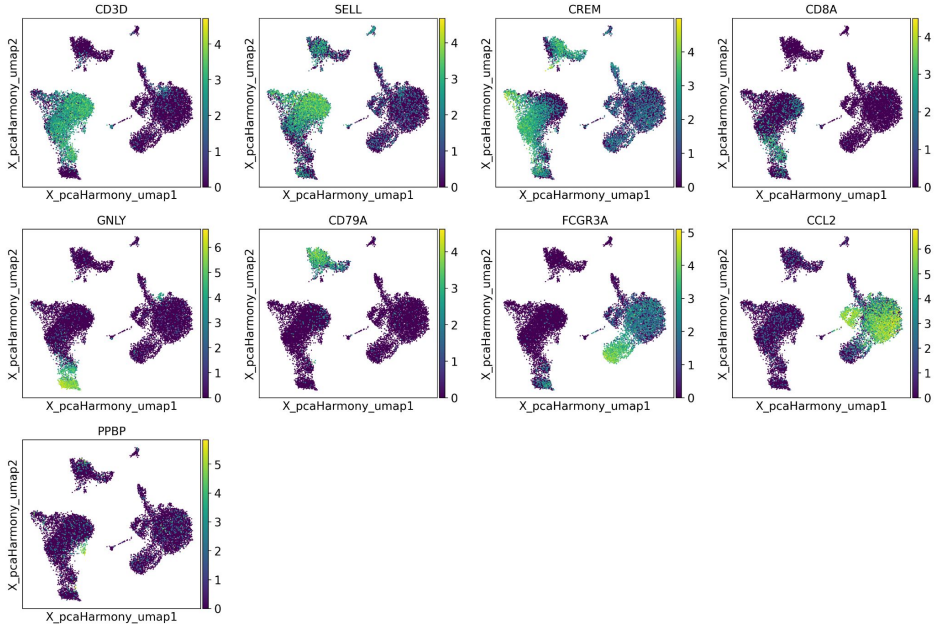
Scanpy-Identify markers for cells

```
sc.pl.violin(adata, ["CCL2", "GNLY"], groupby='leiden')
```

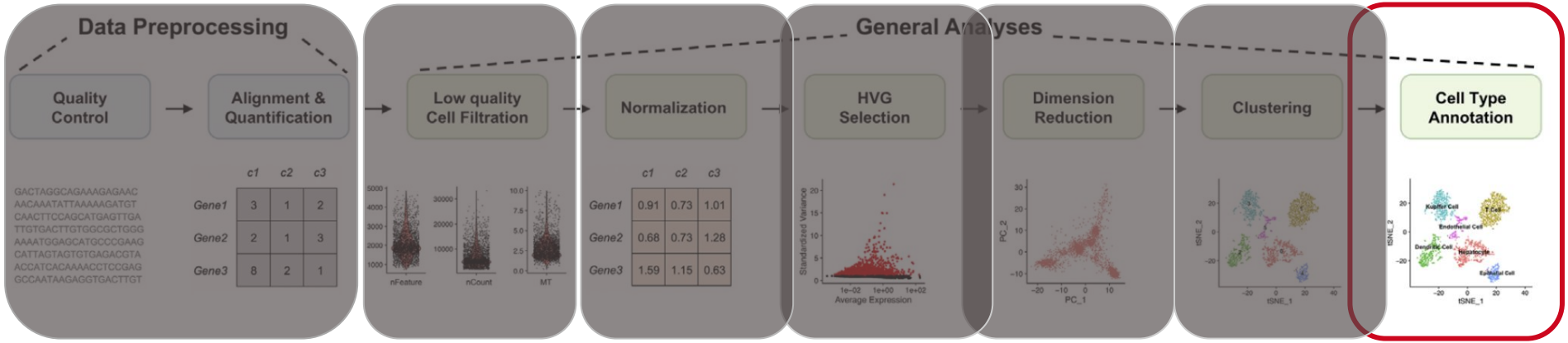


Scanpy-Identify markers for cells

```
genes = ["CD3D", "SELL", "CREM", "CD8A", "GNLY","CD79A", "FCGR3A","CCL2", "PPBP"]
sc.pl.embedding(adata, color=genes, basis='X_pcaHarmony_umap')
```



Scanpy-pipeline

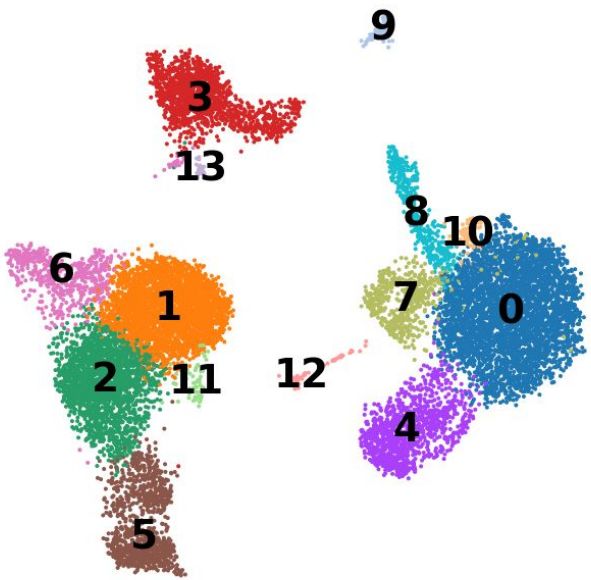


Scanpy-Identify markers for cells

```
new_cluster_names ={"0" :  "CD14 Mono", "1" : "CD4 Naive", "2" : "CD4 Memory T",
                    "3" : "B", "4" : "CD16 Mono", "5" : "CD8 T",
                    "6" : "NK", "7" : "T activated", "8" : "DC",
                    "9" : "CD14 Mono", "10" : "Mk", "11" : "pDC",
                    "12" : "Mk", "13" : "Eryth" }
adata.obs['annotation'] = (adata.obs['leiden'].map(new_cluster_names).astype('category'))
```

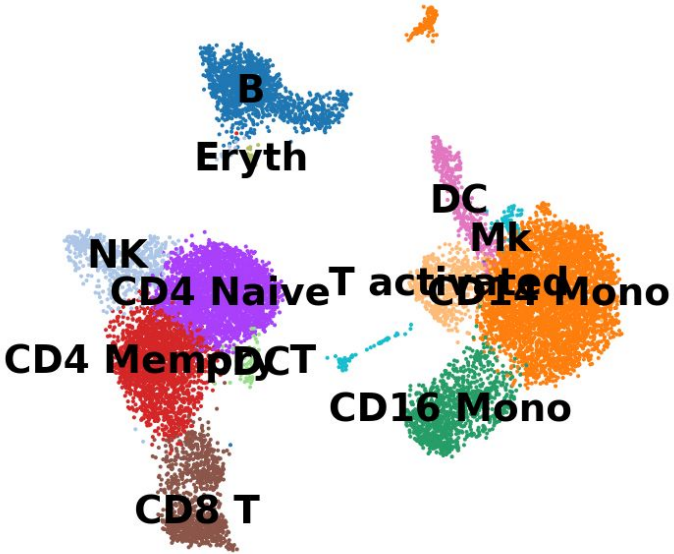
Scanpy-Identify markers for cells

```
sc.pl.umap(adata, color='leiden',
           legend_loc='on data', title='', frameon=False, save='.pdf')
```

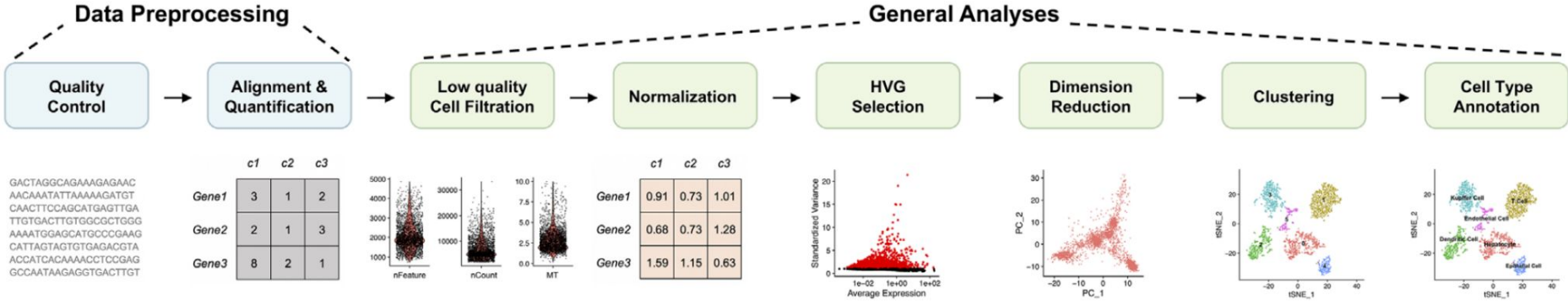


Scanpy-Expert-based cluster annotation

```
sc.pl.umap(adata, color='annotation',
           legend_loc='on data', title='', frameon=False, save='.pdf')
```



Scanpy-pipeline



Alternative in Python - seurat

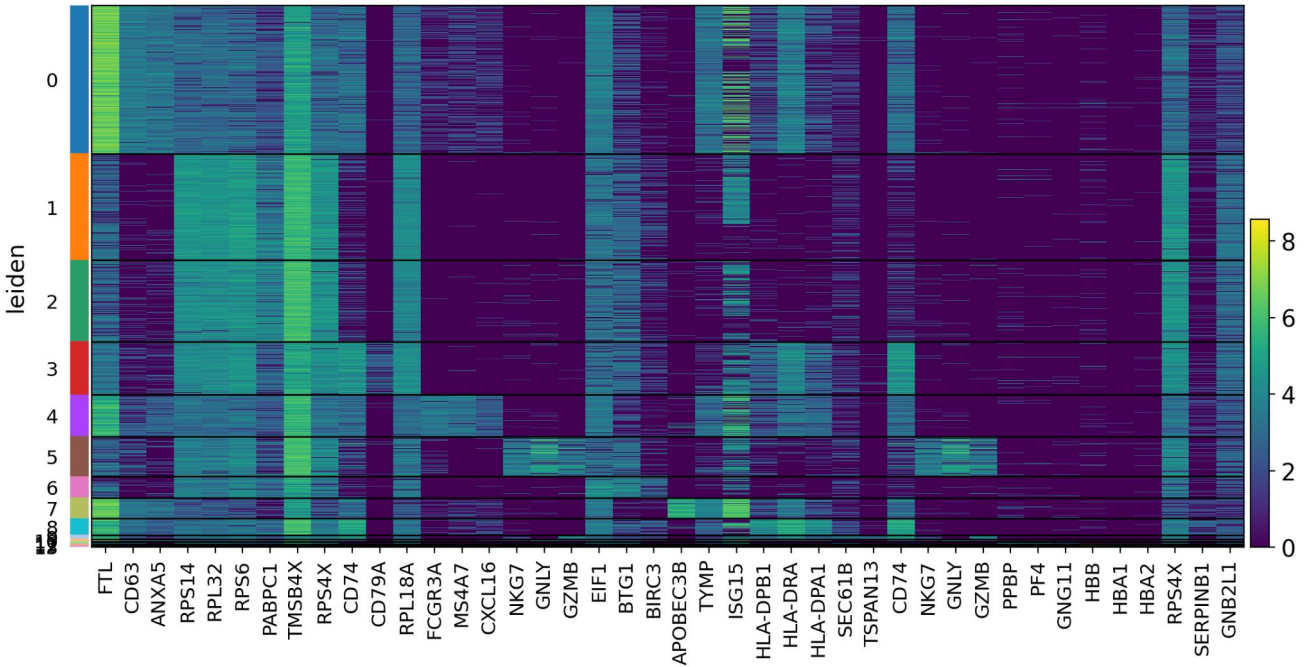
Very similar analysis can be made in R using Seurat:

https://costalab.ukaachen.de/open_data/SOSE_2021/lecture_2_singlecell_practice.pdf

Thanks for your attention

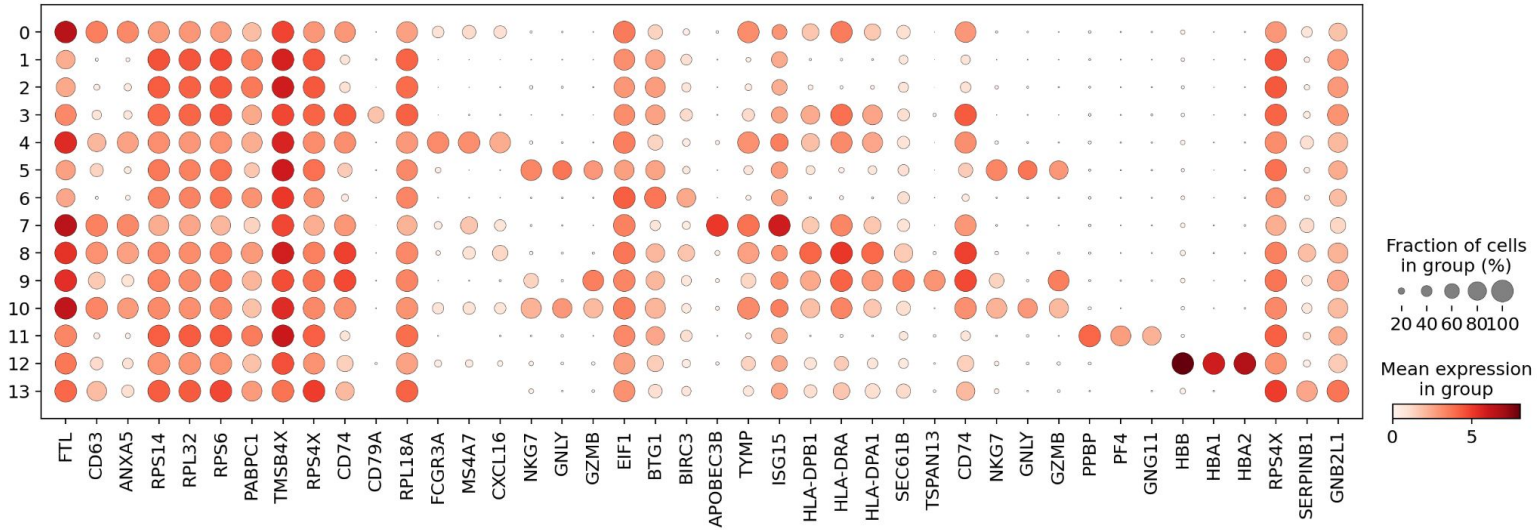
Scanpy-Identify markers for cells

```
sc.pl.heatmap(adata, genes, groupby='leiden')
```



Scanpy-Identify markers for cells

```
sc.pl.dotplot(adata, genes, groupby='leiden')
```



Scanpy-Identify markers for cells

```
sc.pl.stacked_violin(adata, genes, groupby='leiden', rotation=90);
```