Implementing a pipeline for analysing single-cell RNA sequencing data

by

Kwame Ahiavi

Thesis presented in (partial) fulfilment of the requirements for the degree of Master of Science of Molecular Biology (Bioinformatics) in the Faculty of Medicine and Health Sciences at Stellenbosch University

Supervisor: Prof. Gerard Tromp

Co-supervisors: Prof. Gian van der Spuy & Dr. Elizna Maasdorp

Declaration

By submitting this thesis/dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third-party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Kwame Ahiavi

March 2023

Abstract

Single-cell RNA sequencing (*scRNA-seq)* has permitted the dissection of gene expression at single-cell resolution and provides novel insights into the composition of apparently homogeneous cell types and transitions between cell states — thereby deepening our understanding of the cell as a functional unit. The data generated by scRNA-seq are characterised by sparsity, heterogeneity, and high-dimensionality as well as large scale. As a result of biological and technical limitations, scRNA-seq data are "noisier" and more complex than their bulk RNA-seq counterparts. Thus, analysing scRNA-seq data demands new statistical and computational methods. Analytical algorithms employed in scRNA-seq pipelines are prone to producing different results depending on the state at the start of the analysis and the number of iterations of computation, complicating reproducibility. I developed a highly robust, scalable, and reproducible analysis pipeline for scRNA-seq data, implemented in Nextflow — a workflow management system that complies with current best practices in bioinformatics. The pipeline implements pre-processing and comprehensive downstream analyses for scRNA-seq data. With the publicly available datasets used for testing, the pipeline identified cell types and differentially expressed genes that enabled the identification of cell subtypes. Trajectory inference also showed the differentiation trajectory of cells, identifying subclusters within cells. In addition, the pipeline documents all steps and transformations, records software packages and versions, and incorporates ontological metadata annotation. Containerisation of pipeline processes ensures that software dependencies are satisfied — contributing to consistent, robust, and reproducible science.

Opsomming

Enkelsel-RNA volgordebepaling (esRNAv) maak dit moontlik om geenuitdrukking te bestudeer teen enkelsel-resolusie en gee nuwe insig in die samestelling van skynbaar homogene seltipes en die oorgang tussen selfases — daarmee verdiep ons verstaan van die sel as funksionele eenheid. Die data wat esRNAv skep, word gekenmerk deur ‘n yl verspreiding van waardes, groot variasie, hoë dimensionaliteit en wye skaal. esRNAv data is, as gevolg van biologiese en tegniese beperkinge, meer geneig tot agtergrond geraas, as grootmaat RNA volgordebepaling. Daarom het esRNAv data nuwe statistiese en berekeningmetodes nodig. Herproduseerbaarheid is uitdagend omdat esRNAv analitiese algoritmes in pyplyne geneig is om verskillende resultate te gee afhangende van die beginpunt en die hoeveelheid herhalings in berekeninge. Ek het ‘n stewige, herproduseerbare pyplyn wat op enige skaal toegepas kan word, ontwikkel, om esRNAv data te analiseer en het dit implementeer met Nextflow — ‘n werkvloeibestuurstelsel wat huidige beste praktyk in bioinformatika is. Die pyplyn is die eerste om beide voorverwerking en uitgebreide stroom-af analise vir esRNAv uit te voer. Die pyplyn is met datastelle wat vrylik beskikbaar is, getoets en het seltipes uitgeken. Ontwikkelingsbaanafleiding het ook die onderskeiding van selle en onderafdelings gewys. Verder hou die pyplyn rekord van alle stappe, verwerkings, sagteware pakette en weergawes, en sluit ontologiese metadata in. Die pyplyn prosesse is in virtuele houers afgesonder sodat sagteware afhanklikheid bestuur kan word. Dit dra by tot volhoubare en herproduseerbare wetenskap.

Acknowledgements

I thank the *all-knowing* God for His grace to successfully complete this MSc work.

I am thankful to my supervisor, Prof. Gerard Tromp, for his mentorship and support, and for challenging me to think outside the box, and allowing me to tap from his knowledge pool.

I thank my co-supervisors, Prof. Gian van der Spuy and Dr. Elizna Maasdorp, for their mentorship, guidance, and direction. I am grateful to Prof. Helena Kuivaniemi for her encouragement, care, and guidance.

I acknowledge and thank the German Academic Exchange Service (DAAD) for providing MSc scholarship for my studies.

Thank you to Prof. Gerhard Walzl, Head of the Division of Molecular Biology and Human Genetics, for his leadership and maintaining a sound environment that facilitates academic work. I also thank all the administrative staff in the Division. Thanks to the Centre for High Performance Computing (CHPC), South Africa, for providing computational resources for testing and running the pipeline developed in this work.

I thank my family, my parents: Mr. Alfred Ahiavi and Mrs. Margaret Ahiavi; Mr. Alfred Oguaah and Mrs. Evelyn Oguaah for their unconditional love.

To all my friends, thank you for your consistent encouragement throughout this work.

Matthew 6:33 (*NKJV*)

Table of Contents

Declaration ... ii
Abstract ... iii
Acknowledgements ... v
Table of Contents ... vi
List of Abbreviations ... ix
List of Figures ... xi
List of Tables ... xiii
CHAPTER 1: INTRODUCTION ... 1
Problem Statement and Justification ... 5
Purpose of Study ... 8
Aim ... 8
Objectives ... 8
CHAPTER 2: LITERATURE REVIEW ... 9
Introduction ... 9
The Basic Framework of Single-cell RNA Sequencing Methods ... 14
Single-cell Isolation ... 15
Single-cell Capture ... 16
Reverse Transcription, Amplification and Library Preparation ... 18
Applications of scRNA-seq in Biomedical Research ... 21
Provenance of scRNA-seq Data and Challenges ... 22
Pre-processing scRNA-seq Data ... 23
Quality Control ... 23
Alignment ... 25
Feature Quantification ... 26
Data Cleaning ... 26
Removal of Confounding Factors ... 28
Normalization ... 29
Imputation ... 30
Feature Selection ... 31
Downstream Analysis ... 32
Dimensionality Reduction ... 32

Clustering 33
Trajectory Inference 35
Workflows and Workflow Management Systems 36
Nextflow 36
Snakemake 37
Common Workflow Language (CWL) 37
Toil 38
Fair principles and workflow management systems 38
Containerization 38
Current scRNA-seq Pipelines 39
CHAPTER 3: METHODS 41
Choice of Workflow Management System 41
Selection of High-Throughput Tools for the Pipeline 42
Containerization 42
Provenance of Data Set 42
Dataset 1 42
Dataset 2 42
Data Analysis (Preprocessing and Downstream Analysis) 43
CHAPTER 4: RESULTS 46
IMPLEMENTATION OF PIPELINE 46
PREPROCESSING THE DATA 46
Pipeline execution 46
Quality control 47
Output directories and files 48
Gene expression matrix and further preprocessing 51
Normalization 57
Highly variable gene selection 58
Downstream analyses 59
Cell clustering 66
Trajectory inference 67
Cell type annotation 68
Differential gene expression 69

Output from pipeline with plate-based dataset 70
Pre-processing 70
Normalization 72
Highly variable gene selection 73
Downstream analyses 74
Heatmap of cell frequency 74
Clustering 75
CHAPTER 5: DISCUSSION 76
CHAPTER 6: CONCLUSIONS 79
Limitations of the Study & Recommendations for Future Research 79
Recommendations for future research 79
REFERENCES 81
APPENDIX 91
Nextflow workflow script for pipeline 91
scrnaseq.nf 91
Nextflow modules implemented in the pipeline 93
fastqc.nf 93
umi_extract.nf 93
trimmomatic.nf 94
postqc.nf 94
multiqc.nf 95
salmon_index.nf 95
star_index.nf 96
alevin_quanity.nf 96
star_quantify_droplet.nf 97
star_quantify_plate.nf 98
R script 99
lung.Rmd 99
Human_ESC_smartseq2.Rmd 108

List of Abbreviations

cDNA	Complementary deoxyribonucleic acid
CAGE	Cap analysis of gene expression
ERCC	External RNA Controls Consortium
FACS	Fluorescence activated cell-sorting
FISH	Fluorescence in situ hybridization
FPKM	Fragments per kilobase million
HPC	High performance computing
IVT	in vitro transcription
Indel	Insertion-deletion
k-NN	k-nearest neighbour
LCM	Laser capture microdissection
MAD	Median absolute deviation
mRNA	messenger ribonucleic acid
MST	Minimum spanning tree
NGS	Next-generation sequencing
PC	Principal component
PCA	Principal component analysis
PCR	Polymerase chain reaction
RNA-seq	RNA sequencing
RPKM	Reads per kilobase per million reads mapped
QC	Quality control
qPCR	Quantitative polymerase chain reaction
RLE	Relative log expression
SAGE	Serial analysis of gene expression
scRNA-seq	Single-cell RNA sequencing
TI	Trajectory inference
TMM	Trimmed mean of M values

TPM	Transcripts per million
t-SNE	t-distributed stochastic neighbour embedding
UMAP	Uniform manifold approximation and projection
UMI	Unique molecular identifiers
WfMS	Workflow management system

List of Figures

Figure	Title	Page
1.1	A typical RNA-seq experiment	3
1.2	Illustration of scRNA-seq	4
1.3	scRNA-seq gene expression matrix	6
2.1	A gene expression microarray assay	10
2.2	cDNA cloning and expressed sequenced tag sequencing	11
2.3	Illustration of SAGE protocol	12
2.4	Preparation of CAGE libraries	13
2.5	Timeline of single-cell technologies over the past two decades	14
2.6	A basic framework of scRNA-seq experiments	15
2.7	Different single-cell isolation procedures	18
2.8	Steps in scRNA-seq experiment and different experimental approaches	20
2.9	Identification of cell populations	34
2.10	An illustration of containers	39
3.1	Summary of scRNA-seq pipeline framework	44
3.2	Stages of pre-processing implemented in the pipeline	45
4.1	Testing pipeline execution with droplet-based dataset	46
4.2	Near end of pipeline execution with droplet-based dataset	46
4.3	First quality control of reads	47
4.4	Post quality control of trimmed	48
4.5	Output directory of pre-quality control reads	48
4.6	Output directory showing UMIs extracted from reads	49
4.7	Output directory showing trimmed reads	49
4.8	Output directory showing post quality control reads	50
4.9	Output directory showing MultiQC results	50
4.10	Results directory of pipeline processes	50
4.11	Gene expression matrix of droplet-based data	51
4.12	UMIs plotted against barcode	52
4.13	Testing for empty droplets	53
4.14	Quality control using mitochondrial counts	55
4.15	Examining gene expression	56

4.16 Normalization by deconvolution....57
4.17 Variance of epithelial cell dataset....58
4.18 Scree plot variance explained by principal components....60
4.19 Dimensionality reduction using PCA....61
4.20 Dimensionality reduction with t-SNE, perplexity of 30....62
4.21 Dimensionality reduction with t-SNE, perplexity of 5....63
4.22 Dimensionality reduction with t-SNE, perplexity of 80....64
4.23 Dimensionality reduction with UMAP....65
4.24 Clustering of cells from t-SNE with K-NN clustering....66
4.25 Trajectory inference of cells in the epithelial cell dataset....67
4.26 Cell type annotation to existing references....68
4.27 Differential expression of upregulated genes....69
4.28 Distribution of quality control metrics in plate-based dataset....70
4.29 Normalization by deconvolution....72
4.30 Per-gene variance in the plate-based dataset....73
4.31 Heatmap showing the frequency of cells....74
4.32 t-SNE plot of with batch corrections applied....75

List of Tables

Table	Title	Page
2.1	Different scRNA-seq methods	21
3.1	Workflow management systems	41
3.2	Tools implemented in the pipeline	44

CHAPTER 1: INTRODUCTION

The quest to understand the fundamental make-up of living organisms has preoccupied biologists for the past four centuries (Wollman et al., 2015). Robert Hooke and Antoni van Leeuwenhoek were scientists who made notable contributions to the study of microorganisms (Baker & van Leeuwenhoek, 1740; Gest, 2004; Hooke, 1665). The former was the first to study microscopic fungi (Gest, 2004; Hooke, 1665). He coined the term 'cell' — a name he gave to the honeycomb structure of cork he studied with the compound microscope (Gest, 2004; Hooke, 1665). The latter, with the aid of microscopes made of single lenses, studied microscopic protozoa and bacteria. He named them 'animalcules' ('little animals') based on their diminutive appearance (Baker & van Leeuwenhoek, 1740; Gest, 2004; Wollman et al., 2015).

Robert Hooke and Antoni van Leeuwenhoek's insights into the microscopical structure of living things laid the foundation for subsequent enquiries into the cell (Anderson, 2009; Baker & van Leeuwenhoek, 1740; Wollman et al., 2015). However, biologists only appreciated the key role of the cell in biology much later (Regev et al., 2017). To properly understand biological systems, one must have adequate knowledge of their fundamental building blocks — the cell. A universal property of multicellular organisms is cell-to-cell variation, diverse cell types, marked by different morphologies, functions, and gene expression profiles (Huang et al., 2018; Yuan et al., 2017). Irrespective of how seemingly homogenous a single tissue may appear, there is a diverse population of cells constituting distinct manifestations present in that tissue type (Arendt et al., 2016; Yuan et al., 2017).

For centuries, the traditional determinant for classifying cell types was based on *morphology* (Arendt et al., 2016; Kiselev et al., 2019). Cells were defined phenotypically by their structure and function and named according to features of their overall morphology. For example, hair cell, rod, and cone, etc. (Arendt et al., 2016; Kiselev et al., 2019; Svensson et al., 2018). Subsequently, in the middle of the twentieth century, discoveries in molecular biology enabled refinement of cell type identification by *surface proteins* in functionally distinct locations in the cell (Hedlund & Deng, 2018; Kiselev et al., 2019; Svensson et al., 2018). *Proteomic technologies* are useful because they target the functional product of gene expression, but the pre-selection of a limited range of molecules introduces bias in this approach, which prevents the

kind of comprehensive cell phenotype analysis that is possible with measuring RNA (Svensson et al., 2018).

During the last decade, many important discoveries in biology and biomedical research have been fueled by investigating entire transcriptomes which was initially carried out on aggregates of millions of cells, progressing from hybridization-based microarrays, to next-generation sequencing (NGS) techniques also known as *RNA-sequencing* (*RNA-seq*).

RNA-seq has revolutionized our insight into the intricate design and responsive functioning of the transcriptome (Kukurba & Montgomery, 2015; Wang et al., 2009). It is based on recent deep-sequencing technologies, and it is powerful for gene expression pattern exploration (Kulkarni et al., 2019; Wang et al., 2009). *The bulk tissue* used as input material in RNA-seq studies considers all cells to be homogenous, thereby ignoring the stochastic nature of gene expression, and making it insufficient for answering advanced biological questions (Kulkarni et al., 2019; Stegle et al., 2015a).

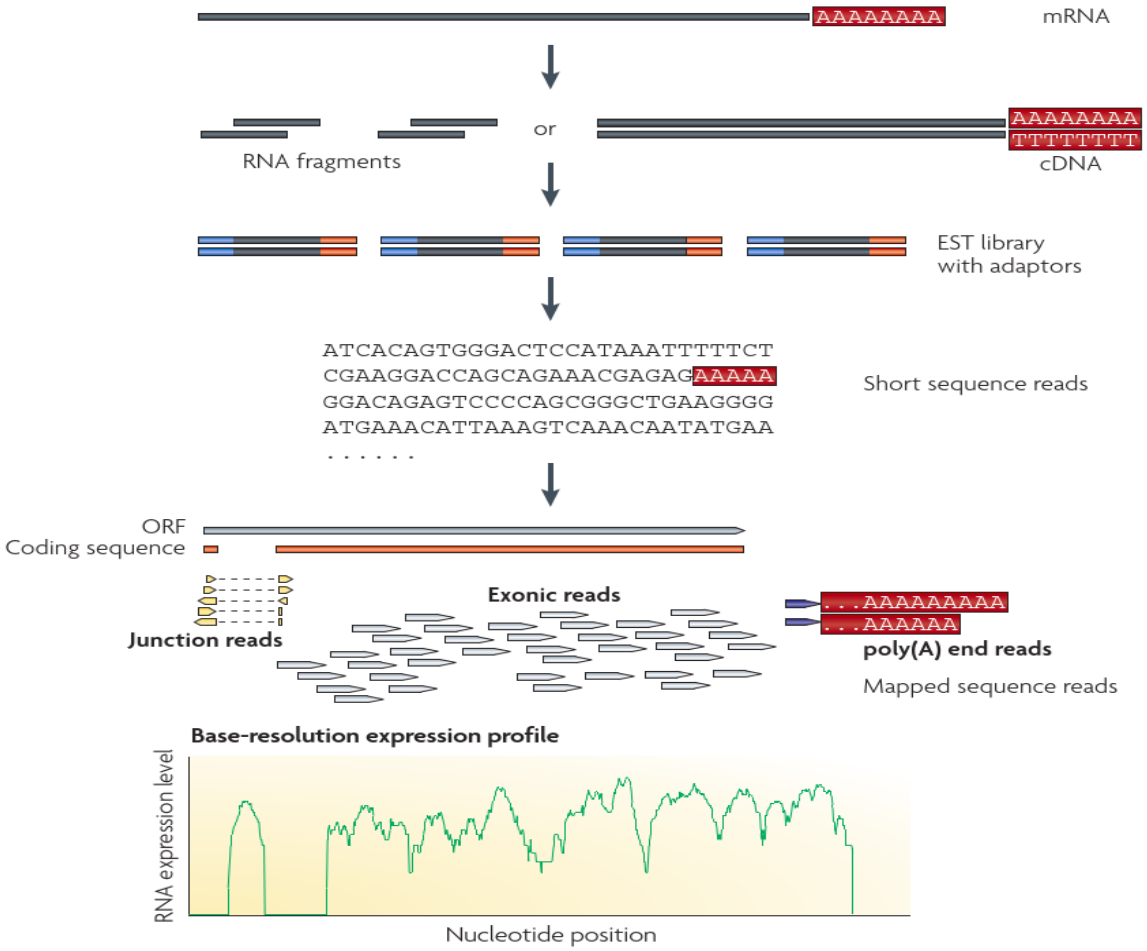

Figure 1.1 | A typical RNA-seq experiment. In brief, input RNA sequences are converted into a cDNA fragments library through either RNA or DNA fragmentation. Subsequently, each cDNA fragment is tagged with sequencing adaptors (blue) and a short sequence is obtained from each cDNA using next-generation sequencing technology. Sequence reads are aligned with the reference transcriptome or genome, and classified as three types: exonic reads, junction reads, and poly(A) end-reads. For each expression profile, a base-resolution is generated from each type, for each gene. Here, the example is of a yeast ORF with one intron. Reproduced with permission from Wang *et al.*, 2009.

Driven by the limitations of *bulk sequencing approaches*, the experimental technique to produce cDNA libraries from polyadenylated mRNA molecules from single cells, have recently greatly improved the high-throughput generation of cDNA libraries from polyadenylated fraction of mRNA molecules within a *single cell* (Haque et al., 2017; Hwang et al., 2018; Kolodziejczyk et al., 2015; Kulkarni et al., 2019; Picelli, 2017; Svensson et al., 2018; Tang et al., 2009; Zheng et al., 2017). This exciting method is referred to as *single-cell RNA sequencing (scRNA-seq)*.

scRNA-seq provides new understanding of the makeup of apparently homogenous cell types and transitions between cell states, and how the composition impacts their function (AlJanahi et al., 2018; Hedlund & Deng, 2018; Kolodziejczyk et al., 2015; Svensson et al., 2018). It has enabled us to ask questions that traditional bulk RNA sequencing techniques cannot address (Hedlund & Deng, 2018; Vallejos et al., 2017).

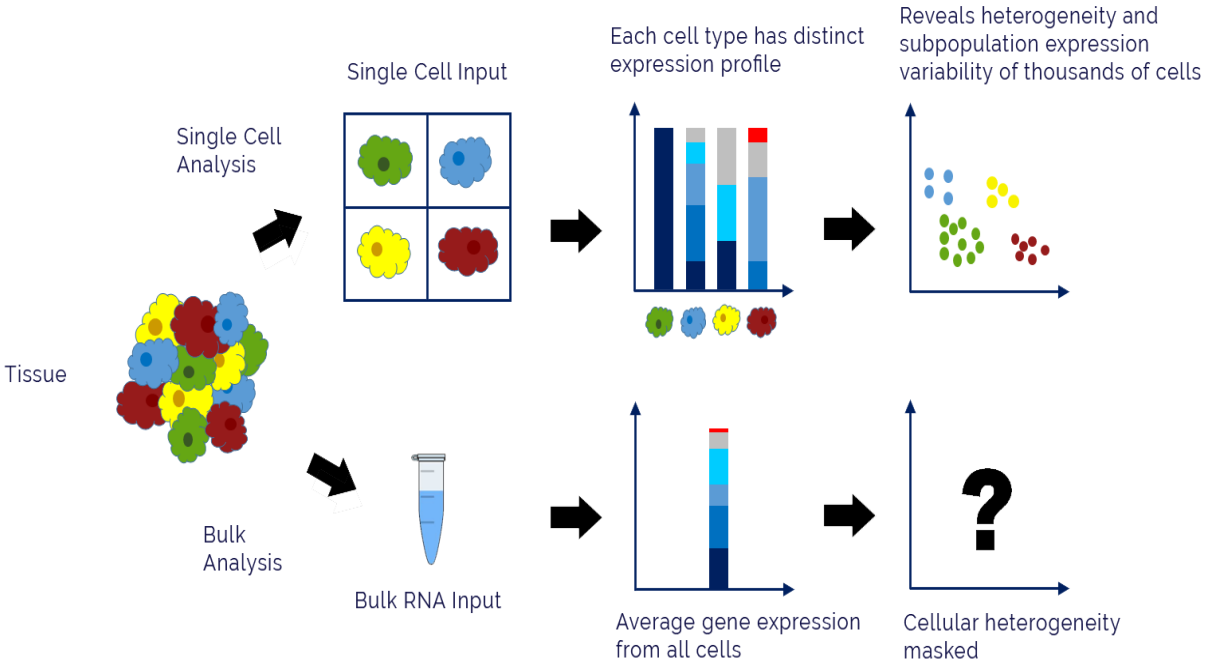

Figure 1.2 | An illustration of scRNAseq showing distinct expression profiles and subpopulation expression variability of similar cells in the same tissue. In contrast to scRNA-seq, bulk analysis (bulk RNA-seq) generates average gene expression profiles from all input cells. Image adapted from https://lcsciences.com

Exploring the full heterogeneity and complexity of cells in multicellular organisms at the single-cell level, requires experimental techniques, but, in parallel, also the development of computational pipelines and methods that can efficiently draw useful and appropriate insights from the complex data generated (Poirion et al., 2016; Yuan et al., 2017). Currently, scRNA-seq analysis pipelines focus on either preprocessing, or downstream analyses, or both (but lack comprehensiveness). Also, they lack standardization and face challenges with reproducibility and automation, making the realization of a gold-standard scRNA-seq pipeline challenging. The goal of this work is to develop and implement a highly robust, scalable, automated (pre-processing

step), and reproducible analysis pipeline for analysing scRNA-seq data using *Nextflow* — a workflow management system. In addition, the pipeline incorporates downstream analyses using currently available computational algorithms and statistical approaches implemented in the R programming language. Containerization of the pipeline processes ensures maximum reproducibility on different computing environments.

Problem Statement and Justification

Over the past decades, traditional gene expression experiments using microarray-based expression platforms and bulk RNA-seq, generated genome-wide mRNA expression data from a population of cells. Even though these experiments have been useful, they are bound to providing weighted average measurements from thousands of input cells, which can average out biological signals of interest (Bacher & Kendziorski, 2016; Kulkarni et al., 2019; Olsen & Baryawno, 2018).

scRNA-seq, since its development in 2009, has allowed us to derive transcriptome-wide data from single cells. This development is a major stepping-stone towards enabling fundamental insights into biology (AlJanahi et al., 2018; Bacher & Kendziorski, 2016; Hwang et al., 2018; Lafzi et al., 2018; Svensson et al., 2018).

Although the data generated from scRNA-seq are technically identical to those from a traditional bulk expression experiment (*millions of mRNA transcripts* are sequenced from *a number of samples or cells*), they are characterized by an abundance of zero counts, complex expression distributions, and increased variability (Bacher & Kendziorski, 2016; Stegle et al., 2015a). These challenges are attributed to biological, as well as technical factors that occur during isolation of single cells, and subsequent quantification of gene expression (Bacher & Kendziorski, 2016; Kim et al., 2019a; Ziegenhain et al., 2017).

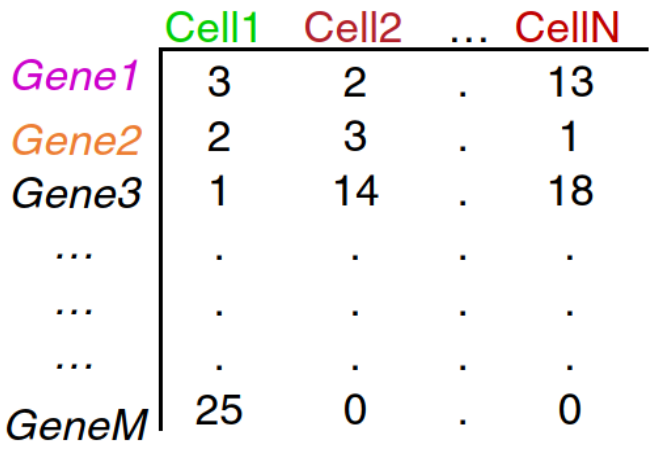

Figure 1.3 | A typical scRNA-seq gene expression matrix depicting zero counts. The gene expression matrix is the basis on which further downstream analyses are carried out. Modified with permission from Lafzi *et al.*, 2018.

The large fraction of zeros observed in scRNA-seq measurements occur when there are no reads or unique molecular identifiers (UMIs) mapped to a given cell (Lähnemann et al., 2020). Random biochemical reactions and transcriptional bursting are among the reasons for the stochastic nature of gene expression that amplify the level of *biological variability* in scRNA-seq data (Kim et al., 2019b). *Technical variability* in scRNA-seq data is the result of inefficient reverse transcription and insufficient sequencing depth per cell, which in turn results in dropout events and sparsity in the gene expression matrix, on which generating deep insights from scRNA-seq data is based (Brennecke et al., 2013).

To correctly analyse single-cell data, an important prerequisite is accounting for both *technical and biological variability* (Kim et al., 2019b). Novel statistical algorithms and computational techniques developed over the past years for analysing scRNA-seq data, have been incorporated in existing scRNA-seq pipelines. However, the lack of standardization and reproducibility is a major challenge that plagues currently available scRNA-seq analysis pipelines. Another issue is the unavailability of comprehensive pipelines that are capable of handling both pre-processing and downstream analyses in the same pipeline.

It is in this spirit, that this work seeks to address these challenges by implementing an scRNA-seq pipeline capable of analysing scRNA-seq data in a robust and comprehensive manner. The

pipeline also ensures reproducible results implemented with protocols from the two main scRNA-seq isolation techniques (*droplet and plate-based*). Containerization of individual pipeline processes ensures reproducible results on different computing environments. This work is in efforts geared towards the realization of a robust pipeline for scRNA-seq data analysis.

Purpose of Study

Aim

The aim of this study is to develop and implement a highly robust scRNA-seq analysis pipeline which can generate reproducible results.

Objectives

The following objectives were carried out to achieve this aim:

1. Review best practice workflow management systems and select the most appropriate one for the pipeline.
2. Review currently available high-throughput genomics tools and select the best ones for the pipeline.
3. Containerize pipeline processes for reproducibility.
4. Pre-process scRNA-seq data and analyse downstream results.
5. Incorporate computational algorithms and statistical methods in the pipeline.
6. Analyse existing scRNA-seq data and interpret the results.

CHAPTER 2: LITERATURE REVIEW

Introduction

The era of single-cell transcriptomics began with Brady *et al.*, (1990) and Eberwine *et al.*, (1992) with the development of *single-cell qPCR* (Quantitative Polymerase Chain Reaction). They increased the number of copies of complementary DNAs (cDNAs) of single cells using in vitro transcription (IVT) to achieve linear amplification and PCR to achieve exponential amplification. For the first time, characterizing single cells went beyond using morphological and transmitter or receptor classifications. An early implementation used microinjection of primers, nucleotides, and enzyme into individual cells that were disaggregated from a region of rat brain (Eberwine *et al.*, 1992). Multiple rounds of amplification resulted in greater than a million-fold increase in the yield of RNA. Subsequent construction of cDNA libraries and expression profiles of the single live cells suggested that cells showed differences in expression profiles, even though they appeared morphologically similar. Another approach not using linear amplification of RNA was used to prepare microgram amounts of cDNA from haematopoietic cells of various lineages (Brady *et al.*, 1990). The approach limited the length of the first cDNA strand synthesis and achieved efficient amplification independent of the length of the original mRNA transcripts. This permitted detection of moderate-to-low abundance mRNA in single cells and confirmed the preservation of their relative abundance during amplification (Brady et al., 1990).

Advances in single-cell qPCR were followed by whole-transcriptome analysis using *high-density microarray* (Kurimoto et al., 2006; Tietjen et al., 2003). The microarray is a small analytical device used for exploring the genome with speed and precision. It replaced traditional biological assays based on gels, filters, and purification columns (Ewis et al., 2005). Microarrays are based on the principle of *hybridization reaction* — where tens of thousands of probes affixed to glass chips are used to capture and quantitate fluorescently-labelled mRNA from cells and tissues. The reaction between the fluorescent samples and complementary sequences on the chip causes each spot to glow with an intensity directly proportional to the quantity of the mRNA (Ewis et al., 2005). Microarray technology has enabled the determination of the relationships between physiological states of cells and gene expression patterns for studying disease progression, tumours, cellular response to stimuli, and identification of drug targets (Ewis et al., 2005; Trevino et al., 2007). Furthermore, its use has been extended beyond gene expression to genotype single-nucleotide polymorphisms (SNPs), detect alterations in methylation patterns,

quantitate gene copy numbers, identify alternative splicing, and pathogens (Ewis et al., 2005; Pirrung & Southern, 2014; Trevino et al., 2007).

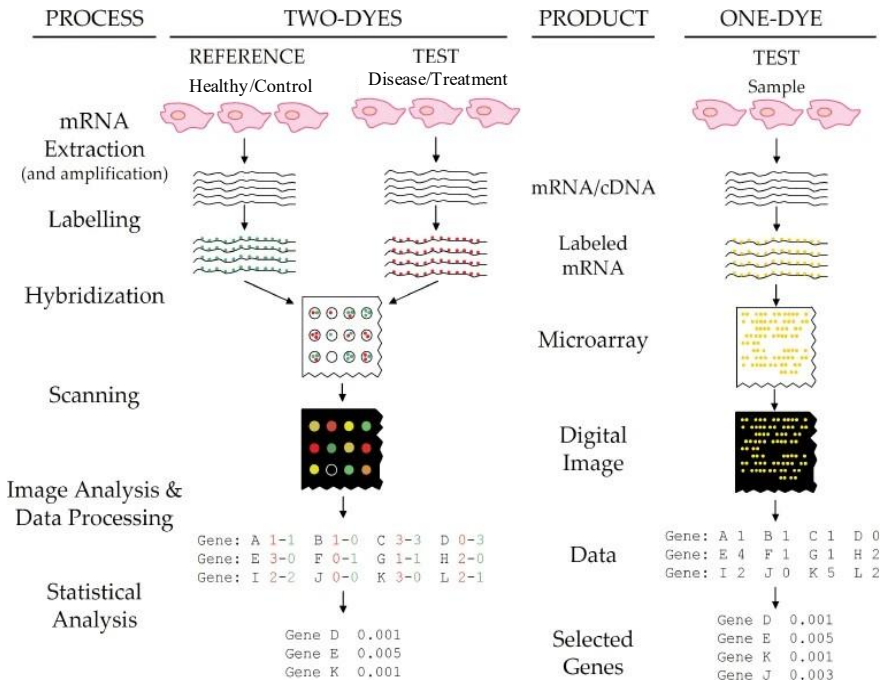

Figure 2.1 | A schematic representation of a Gene Expression Microarray Assay. Arrows (left) represent the process. The pictures or text represent the product. Differences in the protocol in one and two-dye technologies are specific to the technology rather than the samples or question. Reproduced with permission from Trevino *et al.*, 2007.

S*equence-based approaches* were developed to explore the transcriptome by determining the counts of transcript sequence directly rather than aggregate intensity signals as in the hybridization-based approaches. Early gene expression studies were based on sequencing the ends of cDNA clones using Sanger sequencing (Sanger et al., 1977). The sequences generated were called *expressed sequence tags* (ESTs) (Adams et al., 1991; Itoh et al., 1994). This approach, however, is quite low-throughput and prone to artefacts, and therefore not ideal for quantification of transcripts (Adams et al., 1991; Itoh et al., 1994). To increase throughput and overcome limitations due to artefacts, methods based on sequencing defined tags that identify specific mRNAs were developed. The number of times a tag is represented represents the expression level of the mRNA species. Tag-based methods include *serial analysis of gene expression* (SAGE) and *cap analysis of gene expression* (CAGE) and provide more precise

quantification of gene expression levels. The discrete quantitation methods are superior to methods based on quantitation of analog-style intensities of array-based methods (Shiraki et al., 2003; Velculescu et al., 1995). These assays are, however, insensitive to measuring expression of splice isoforms, and cannot be used for novel gene discovery. These methods proved of limited use due to the requirement of large quantity of input RNA, time-consuming cloning of sequence tags, and high cost of automated Sanger sequencing (Kukurba & Montgomery, 2015; Wang et al., 2009).

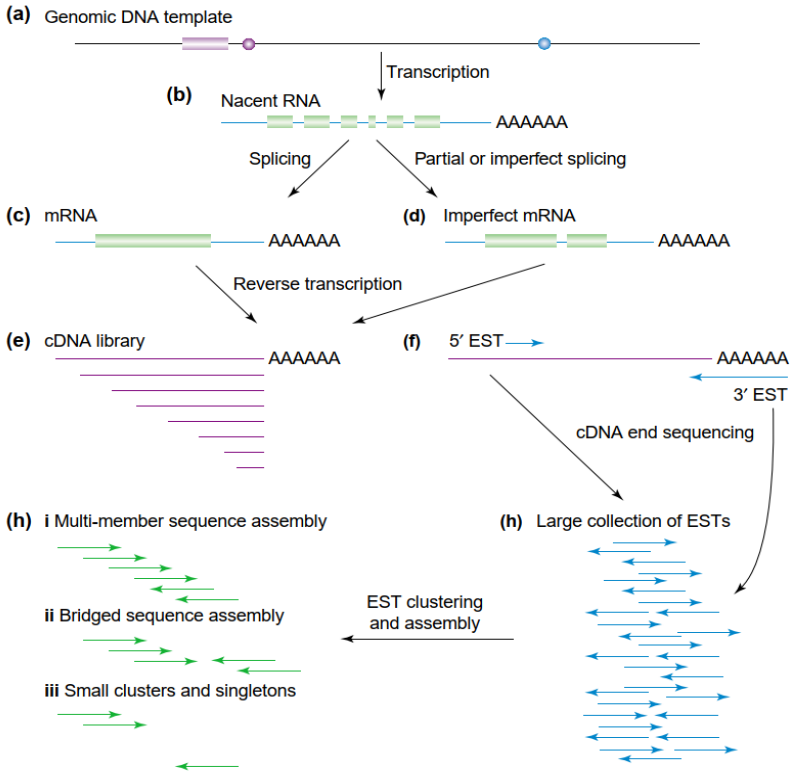

Figure 2.2 | Summary of cDNA cloning and expressed sequence tag (EST) sequencing. DNA sequence is read from the ends of the cDNA, yielding 5' and 3' ESTs (f). Reproduced with permission from Rudd, 2003.

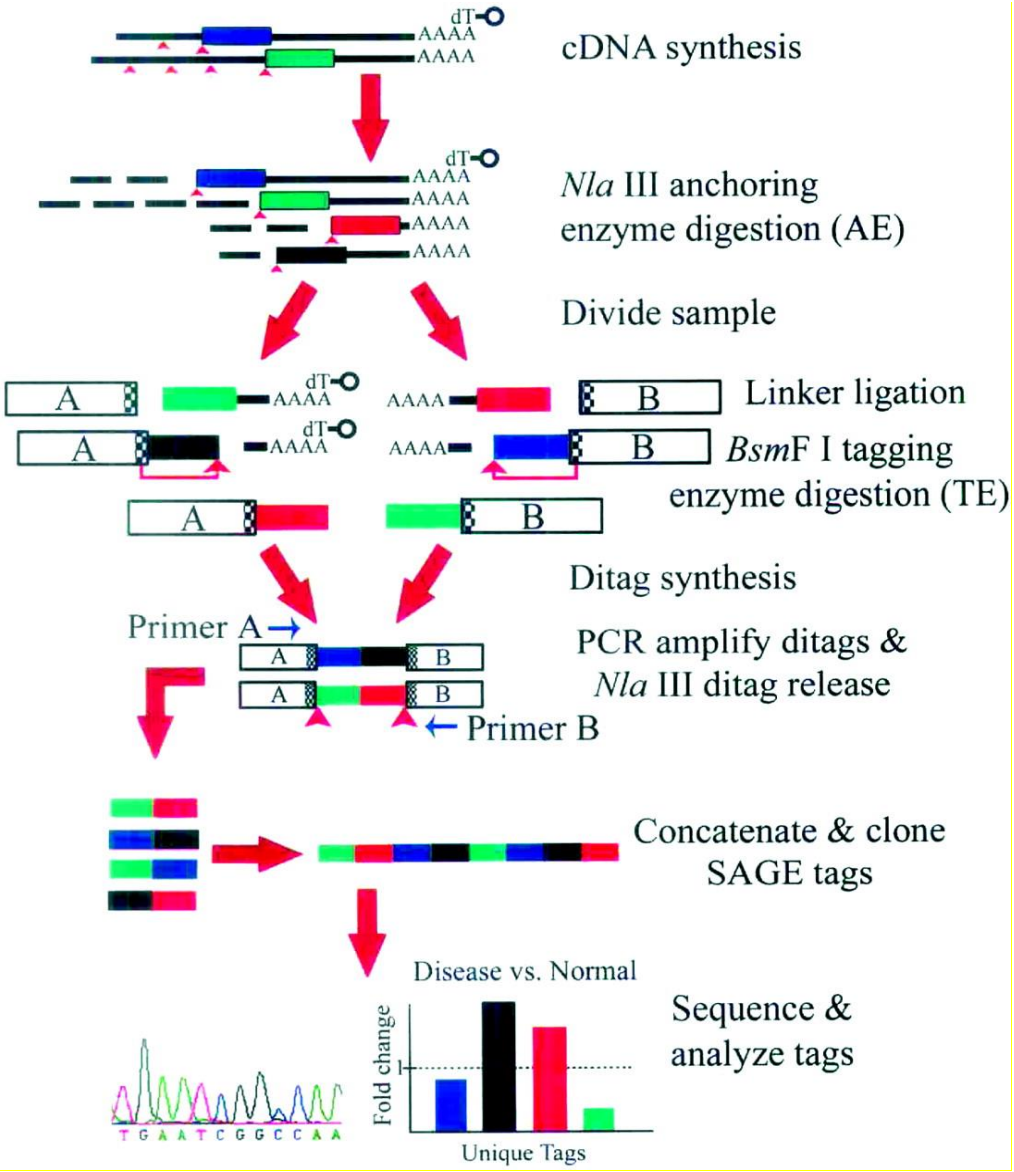

Figure 2.3 | Illustration of the SAGE protocol. Poly(A) mRNA is converted to cDNA by bead-bound oligo-dT-primed reverse transcription and second-strand synthesis. Digestion with the restriction enzyme, *NlaIII*, results in oligo-dT bound cDNA fragments attached to oligo-dT beads (circles) terminated at the 3'-most *NlaIII* site (inverted magenta arrowheads). Ligation of linkers containing the *BmFI* restriction site permits cleavage of tags that are uniformly 10 bp long, but still attached to the linkers since the restriction enzyme cleaves 10 bp 3' to its restriction site. Ligation of the linker tags results in randomly associated ditags. PCR is used to amplify the ditag collection, and the ditags are released from the linkers by *NlaIII* cleavage. The pool of ditags is concatenated by ligation and forms the template DNA for sequencing. The ratio of tag counts between conditions such as disease vs. normal, is used to identify induced or repressed transcripts. Reproduced with permission from Patino *et al.*, 2002.

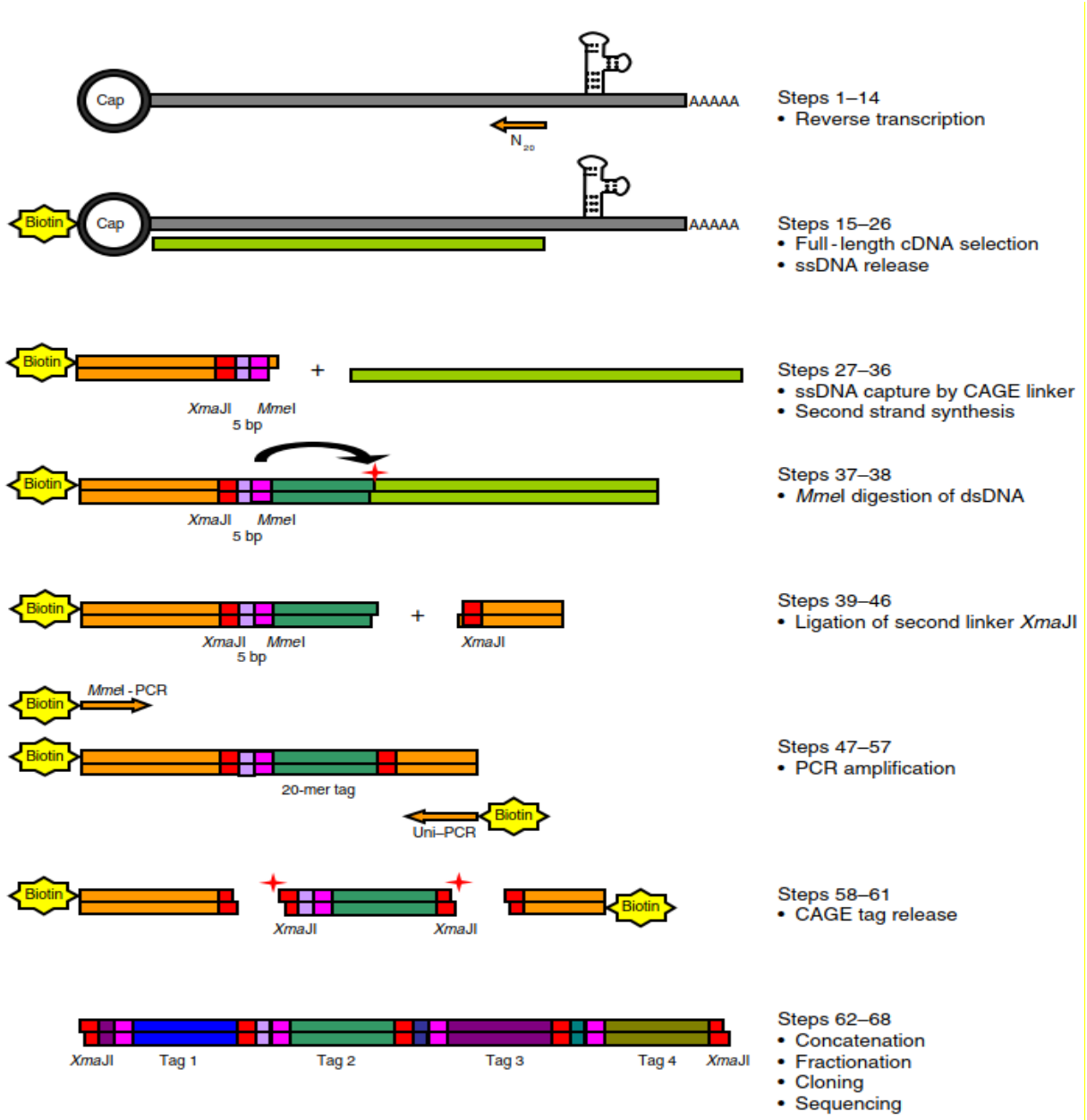

Figure 2.4 | Key steps in the preparation of CAGE libraries. Reproduced with permission from Kodzius *et al.*, 2006.

Increasingly transcriptomics has been conducted with next-generation sequencing techniques (RNA-sequencing). RNA-seq conducted on pooled cells (so called bulk RNA-seq) has provided large amount of information that is consistently generating new insights and discoveries in biomedical research (Haque et al., 2017). For instance, with clinically relevant examples, RNA-seq was conducted on haematopoietic stem cells to classify acute myeloid leukaemia patients into cohorts requiring different treatment regimens. RNA-seq studies using portions of tissue,

termed bulk RNA-seq, assume all cells are homogenous, thereby ignoring differences between cells and the stochastic nature of gene expression (Kulkarni et al., 2019). The averaging that happens with pooling of a vast population of cells does not permit detailed investigation of the cell or the individual nuclei (Haque et al., 2017).

Tang *et al. (2009)* built upon RNA-seq technology to make single-cell analysis compatible with high-throughput DNA sequencing — performing the first, complete, and unbiased transcriptome-wide study of the mRNA in a single cell in 2009. Since then, numerous variations of scRNA-seq have been developed. These achievements are very important as each method comes with its distinct advantages and applicability (Hedlund & Deng, 2018).

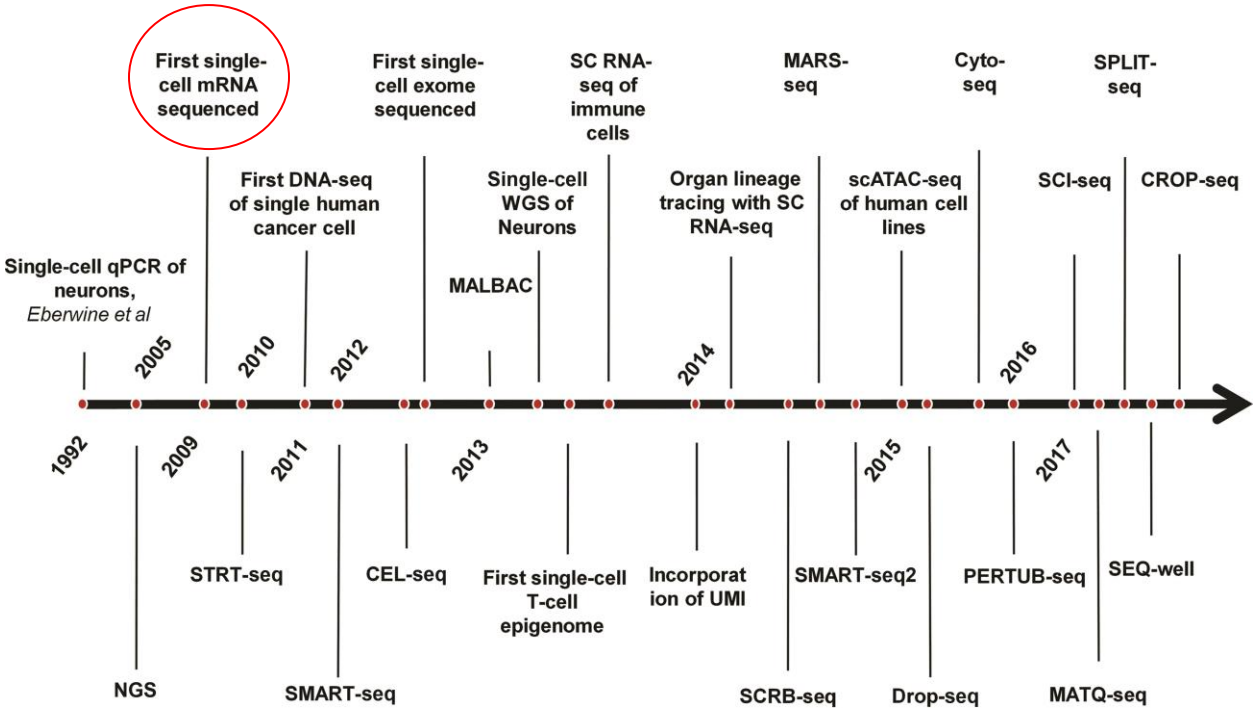

Figure 2.5 | Timeline of single-cell technologies over the past two decades showing the implementation of different protocols. The first scRNA-seq study is highlighted in red. Modified with permission from Paolillo *et al.*, 2019.

The Basic Framework of Single-cell RNA Sequencing Methods

Two main challenges must be overcome to sequence mRNA from single cells: (i) effectively capturing these single cells, and (ii) amplifying the low quantities of mRNA from each cell. This basic plan is followed by all scRNA-seq experiments. An individual cell is captured and lysed,

after which there is a reverse transcription step to select for mRNA to generate cDNA. The minute amount of mRNA is amplified via PCR or in vitro transcription. Finally, a sequencing library is prepared by using the amplified cDNA (Kolodziejczyk et al., 2015).

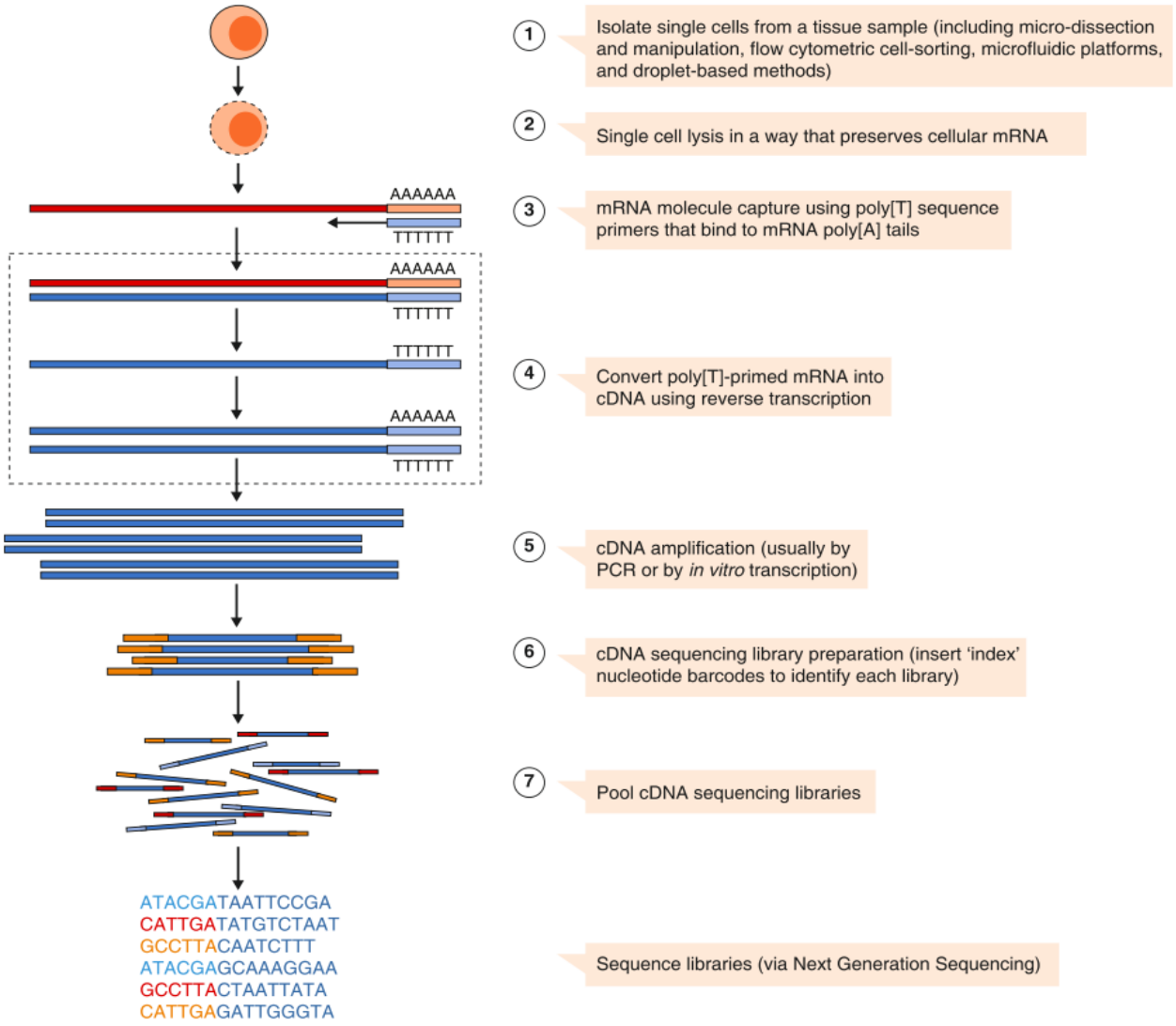

Figure 2.6 | A basic framework of scRNA-seq experiments. Modified with permission from Haque *et al.*, 2017.

Single-cell Isolation

Isolation of single cells is the first and important step for retrieving information from the transcriptome in single cells. Cells must be freed from their tissue surroundings, extracellular matrix, and cell-to-cell adhesion. This is typically accomplished by *enzymatic treatment*, *mechanical dissociation* of the tissue, or *laser capture microdissection (LCM)*. Even though this initial step may seem to be easy, capturing *intact* single cells accurately and quickly remains a

main challenge in single-cell sequencing (Hedlund & Deng, 2018; Kolodziejczyk et al., 2015; Olsen & Baryawno, 2018).

Single-cell Capture

After a single-cell suspension is obtained, individual cells must be isolated to enable downstream biochemical reactions to occur independently in each cell. A commonly employed technique is *limiting dilution*, where pipettes are used for isolating single cells by dilution. Due to the distribution of the diluted cells, this technique is not very efficient (Hwang et al., 2018).

Micromanipulation is a widely used technique for capturing single cells from samples with low cell counts, e.g., early embryos or uncultivated microorganisms. Although this approach is time consuming and low-throughput, the operator ensures each isolation contains a single cell by monitoring the process using a microscope (Kolodziejczyk et al., 2015; Tang et al., 2009).

Another approach for isolating highly purified single cells is *fluorescence activated cell-sorting (FACS)*. This is a type of flow cytometry that sort cells according to size, morphology, and/or fluorescence characteristics. It enables the enrichment of specific cells of interest if they are fluorescently labelled. Here, a fluorescent monoclonal antibody which recognizes specific surface markers is used to tag cells which enables the sorting of distinct populations. A potential limitation of this approach is the requirement of large starting volumes (Hwang et al., 2018; Kolodziejczyk et al., 2015; Olsen & Baryawno, 2018).

Laser-capture microdissection (LCM) makes use of a laser system for the isolation of single cells from thin tissue sections under a microscope. It preserves the spatial relationship between isolated cells. This technique is similar to micromanipulation and is also labour intensive and low throughput (Olsen & Baryawno, 2018).

Microfluidic technology is a very popular method for single-cell isolation because of its minimal sample intake and cost of analysis. It enables the manipulation of ultra-low liquid volumes (on the nanoliter scale) which is known to reduce external risk for contamination (Hwang et al., 2018). Liquid flow is tightly controlled using prefabricated chips with microchannels. A commercially available platform is the Fluidigm C1. It allows automatic single-cell lysis, extraction of RNA, and cDNA synthesis for about 800 cells on a single chip all in parallel

(Hwang et al., 2018; Olsen & Baryawno, 2018). A major drawback of this method includes the homogenous size limit of cells and the number of cells (>1000) being captured (Hwang et al., 2018).

Microdroplet-based microfluidics is another technique that is showing enormous promise for single-cell isolation. Beads containing a unique barcode primer are mixed with single cells in suspension. An oil emulsion process is used to encapsulate beads and cells in nanoliter-size droplets in the oil. Compared to standard microfluidics chambers, this technique requires a lower volume and thus allows the analysis of thousands of cells all at a relatively lower cost. Commercial platforms widely used include 10X Genomics Chromium, Drop-seq, and inDrop protocol (Hwang et al., 2018; Olsen & Baryawno, 2018).

Aside from the above-mentioned high-throughput microfluidics technologies, scRNA-seq can be performed in microwell plates. These plate-based approaches rely on single cells isolated by FACS or picking them (via micromanipulation) (Potter, 2018). These methods are applicable to a wider range of cell sizes than the Fluidigm system, where the microfluidic “tubes” on the chips need to have diameters only slightly smaller than that of the cells, and requiring cell size-appropriate microfluidic devices (Kolodziejczyk et al., 2015; Potter, 2018).

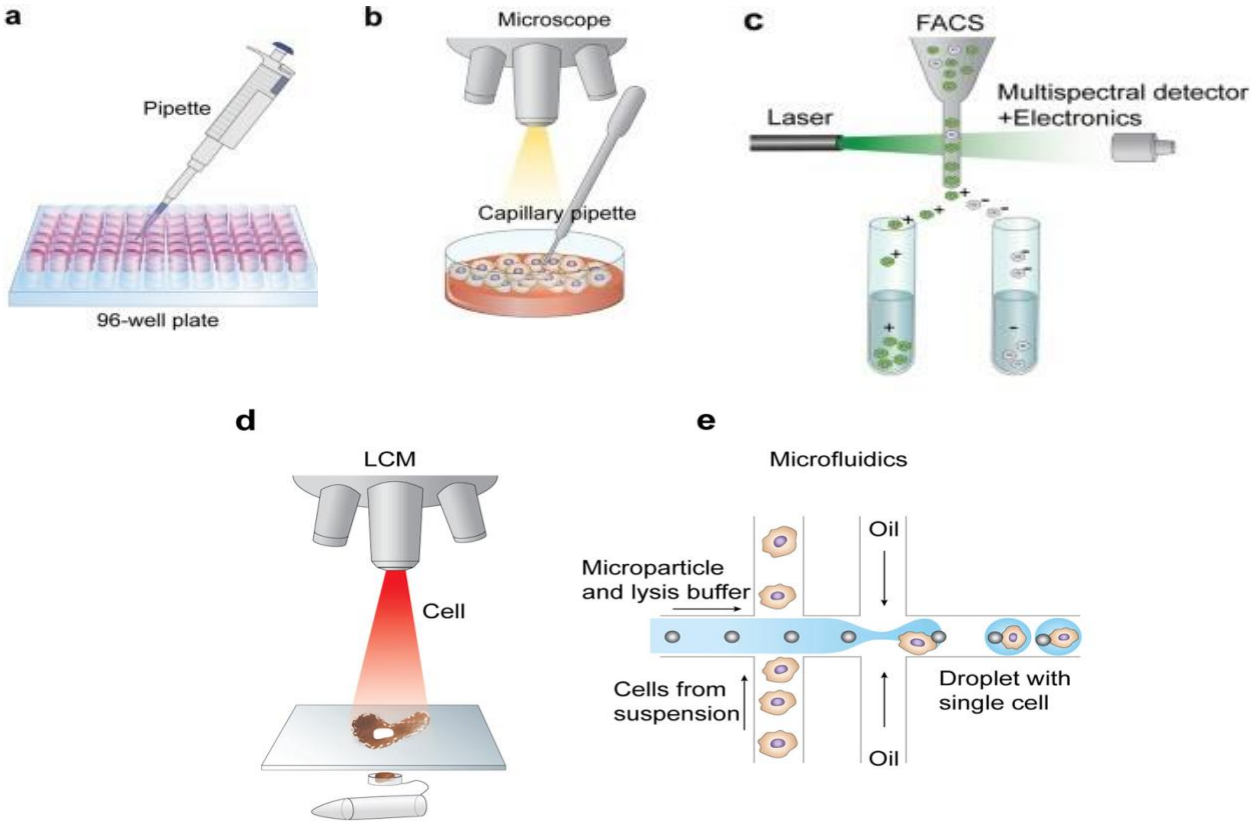

Figure 2.7 | Different single-cell isolation procedures. **(a) L**imiting dilution technique separates single cells by taking advantage of the distribution of diluted cells. **(b)** In micromanipulation, microscope guided by capillary pipettes are used for the collection of single cells. **(c)** Highly purified single cells are isolated in FACS by targeting cells labelled with fluorescent marker proteins. **(d)** LCM employs a laser system aided by a computer to capture single cells from solid samples. **(e)** Microfluidics for isolating single cells. Modified with permission from Hwang *et al.*, 2018.

Reverse Transcription, Amplification and Library Preparation

Modern high-throughput sequencing platforms only sequence DNA molecules, and therefore, conversion of mRNA to cDNA (reverse transcription), and subsequent cDNA amplification is a required before sequencing can be done. It is estimated that only 10-20% of transcripts are reverse transcribed at this stage because of the stochastic rare-event nature of successful primer hybridization and reverse transcriptase binding and extension, which results in a Poisson sampling of transcripts. This low success rate of mRNA capture is a significant challenge in current scRNA-seq protocols and emphasizes the need for a highly effective cell lysis technique (Hwang et al., 2018; Kolodziejczyk et al., 2015).

In cDNA preparation, first-strand synthesis typically makes use of an engineered version of the Moloney murine leukemia virus which has low RNase H activity and its high thermostability. Poly(A) tailing, or template-switching (TS) mechanism is used for second-strand synthesis. As compared to first-strand synthesis, this latter approach ensures uniform coverage without loss of strand specificity. The small quantity of cDNA is then amplified using either polymerase chain reaction (PCR) or in vitro transcription (IVT). Each of these approaches can result in bias (Hwang et al., 2018; Kolodziejczyk et al., 2015).

Specific scRNA-seq protocols: Smart-Seq2 (Switching mechanism at the end of the 5' end of the RNA transcript sequencing version 2), STRT-Seq (single-cell tagged reverse transcription sequencing), CEL-Seq (Cell expression by linear amplification and sequencing), CytoSeq (gene expression cytometry), etc., employ either approach of strand synthesis or cDNA amplification with some modifications (Kolodziejczyk et al., 2015). In vitro transcription (IVT) amplifies templates linearly but requires additional reverse transcription, which may lead to 3' bias. Smart-Seq2 uses full-length transcript cDNA amplification and template switching; it therefore facilitates alternative splicing, novel detection of exons, and allele-specific expression (Hwang et al., 2018). Other methods focus on tag counting of either 3' (CEL-Seq, cell expression by linear amplification and sequencing) or 5' (CytoSeq, STRT-Seq) fragments, and thereby reducing the number of sequencing reads for quantification purposes (Kolodziejczyk et al., 2015).

The use of unique molecular idientifier (UMIs) is a recent strategy used by researchers for properly quantifying scRNA-seq reads. These are random base pairs (bp) of variable length depending on the experimental protocol sequences employed at the reverse transcription stage. With this approach, each read can be assigned to its original cell by eliminating PCR bias and enhancing accuracy (Hwang et al., 2018).

Most scRNA-seq protocols incorporate the Nextera kit for library preparation and the Illumina platform for sequencing. After amplification, the cDNA obtained is compatible with different sequencing platforms such as the SOLiD or PacBio system (Kolodziejczyk et al., 2015).

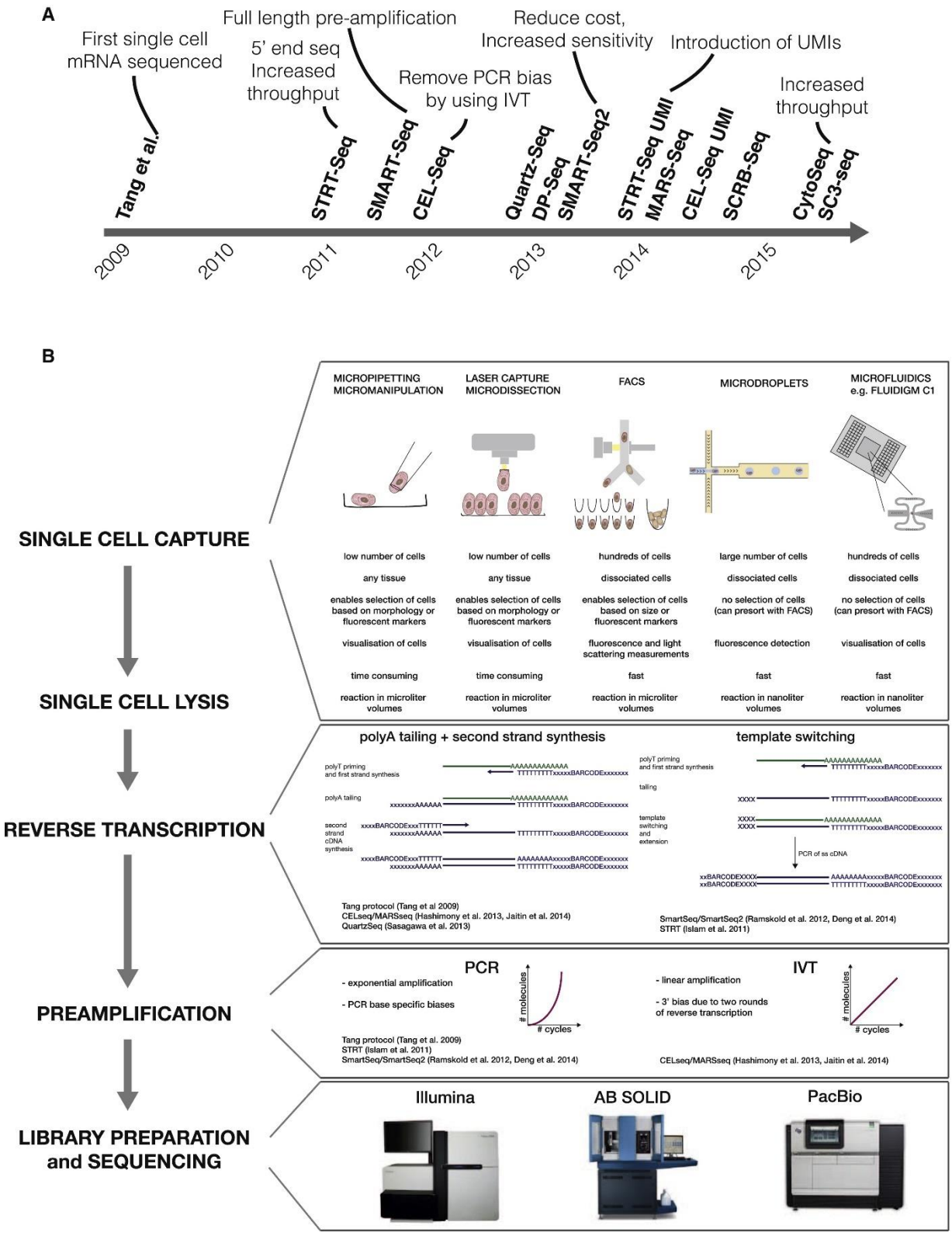

Figure 2.8 | A schematic showing each step in a scRNA-seq experiment and different experimental approaches. Reproduced with permission from Kolodziejczyk *et al.*, 2015.

Table 2.1 | Summary of different scRNA-seq methods

RNA-seq method	Year	Methods for transcript detection
CEL-seq/CEL-seq 2	2012/2016	3'-end counting
SMART-seq/SMART-seq 2	2012/2014	Full-length
STRT-seq/STRT-seq 2	2013/2017	5' only
SCRB-seq	2014	3'-end counting
MARS-seq	2014	3'-end counting
Drop-seq	2015	3'-end counting
InDrop-seq	2015	3'-end counting
FISSEQ	2015	Full length
Cyto-seq	2015	Predefined genes only
MATQ-seq	2017	Full length
Sci-seq	2017	3'-end counting
SEQ-well	2017	3'-end counting
SPLIT-seq	2017	3'-end counting
10x Chromium	2017	3'-end counting

Modified with permission from Paolillo et al., 2019

Applications of scRNA-seq in Biomedical Research

Analyses of cell heterogeneity is amongst the primary reasons for performing scRNA-seq studies (Haque et al., 2017). In humans, scRNA-seq has shed novel insights into the clonal evolution of cancer. Rare cell types have been used to assess transcriptional differences between individual cells which normally goes unnoticed in bulk analysis, e.g., tumor cells that are malignant, or hyper-responsive immune cells within a homogenous group (Haque et al., 2017; Olsen & Baryawno, 2018).

scRNA-seq was used by researchers to discover types of cells in the haematopoietic and immune system (Villani et al., 2017), nervous system (Llorens-Bobadilla et al., 2015), and has also opened new approaches for studying mechanisms associated with disease development, progression, and drug resistance (Linnarsson & Teichmann, 2016).

Moreover, scRNA-seq is a suitable method for investigating individual cells where each one is unique, like T lymphocytes with highly diverse T-cell receptor, cells within an early-stage embryo or neurons within the brain. It has been employed in studies of aging and is used for lineage tracing and studying the developmental relationships between heterogenous but related

cellular states in cases such as development of embryos, differentiation of lung epithelium, and lymphocyte fate diversification (Haque et al., 2017; Lafzi et al., 2018).

In addition to uncovering the heterogenous nature of cells, scRNA-seq provides important information about the fundamental nature of gene expression. Key applications involve studying monoallelic gene expression, noise during transcriptional responses and splicing patterns. At the single-cell level, investigating gene expression can permit identification of gene modules that are co-regulated and inferring of gene-regulatory networks that underpin specification of cell types and functional heterogeneity (Hedlund & Deng, 2018; Hwang et al., 2018).

Provenance of scRNA-seq Data and Challenges

Raw data from sequencing generated from scRNA-seq experiments are normally in the form of binary base calls (BCL files). These files first undergo demultiplexing using oligonucleotides index sequences to generate FASTQ files. FASTQ files obtained from sequencing consists of millions of reads of RNA sequences and extra sequences (cell barcode and UMI) (Kulkarni et al., 2019; Poirion et al., 2016).

scRNA-seq gene expression matrix — the foundation on which subsequent insights can be generated from a dataset — is characterized by complex expression distributions and increased variability. This is partly attributable to the reduced abundance of starting material and increased resolution, posing challenges which require new statistical and computational methods (Bacher & Kendziorski, 2016).

Data from scRNA-seq can be affected by varying technical artifacts from cell capture, library preparation, and sequencing procedures. After obtaining reads from a well-designed experiment, quality control (QC) is essential to remove cells of low-quality before expression estimation. QC is done on the raw reads, aligned reads, as well as across all cells. Low-quality here describes cells that are damaged, dead, or to capture sites that have no cells or contain multiple cells. Capture sites are inspected with a microscope to remove these empty or multiple cells before sequencing for experiments implementing microfluidic-based methods. A visual inspection is not feasible with all experimental setups. FACS metadata from indexed FACS sorting may reveal

information about the live and dead cells as well as cell size (Bacher & Kendziorski, 2016; Kolodziejczyk et al., 2015).

Pre-processing scRNA-seq Data

Quality Control

Pre-processing tools designed for bulk RNA-seq studies have proven useful in scRNA-seq studies. Among the widely used tools for *QC* include: FASTQC, Cutadapt, Kraken, RNA-SeQC (Bacher & Kendziorski, 2016; Poirion et al., 2016). FASTQC generates a quality report from the reads from the FASTQ files. Cells that have a disproportionate number of low-quality reads are marked for removal. The graphical output is beneficial since it allows one to quickly assess structures present in the low-quality scores (Andrews, 2022).

Trimming is the process of removing portions of reads where the base-call quality is below an acceptable level of confidence. Base-call quality generally declines along the length of the reads. Removal of low-quality bases may be useful for alignment. *Trimmomatic, Cutadapt, and Trim Galore* are among widely used tools for adapter trimming of scRNA-seq data. When the first positions are occupied by an abundance of low-quality scores, it may represent a transient problem with the run and trimming is not suggested in this case (Bacher & Kendziorski, 2016). It can be used to remove adapter sequences to improve the quality of the reads. Similarly, each tool calculates read quality using summaries of per-base quality defined based on the likelihood of a misplaced base call.

Cutadapt is a tool that trims colour-space reads and supports input file formats such as FASTQ and FASTA (Martin, 2011). Cutadapt can search for multiple adapters per single run and remove the best matching one. As an option, because an adapter can be accidentally appended multiple times during library preparation, it can look for and remove these adapters multiple times, which is a very important feature (Martin, 2011).

Trimmomatic is another tool that performs quality-based trimming and removal of adaptor sequences as well. It was developed with much emphasis on handling paired-end data, although it works with single-end data as well. This tool uses the simple mode and palindrome mode to identify sequences of poor quality within reads (Bolger et al., 2014). The simple mode operates by finding a likely match between the read and reference sequences for adapters and linkers sup-

plied by the user. This mode accommodates all poor-quality sequences — including adaptors and PCR primers, or fragments. The palindrome mode searches for inverted repeats at the beginning and end of the read. It detects the common “adaptor read-through” scenario, where the sequenced DNA fragment is shorter than the read length, resulting in contamination of adapters at the end of the reads (Bolger et al., 2014).

Trim Galore is a tool that uses Cutadapt and FASTQC. It applies consistent quality and adapter trimming to FASTQ files. It has additional functionality to remove methylation positions for sequence files generated by reduced-representation bisulfite sequencing (RRBS) (Krueger, 2012).

For scRNA-seq experiments implementing UMI-based protocols, *UMI-tools* can also be used for barcode trimming (Smith et al., 2017). The UMIs are random barcodes employed in high-through sequencing experiments. Similar copies arising from different molecules can be separated from copies arising through PCR amplification from the same molecule (Poirion et al., 2016; Smith et al., 2017; Zhang et al., 2021).

Further QC measures for scRNA-seq data involve analysing the expression of overall gene expression patterns, housekeeping genes, and number of genes detected for every cell. Different datasets require arbitrary cut-offs for filtering (Poirion et al., 2016).

Specific QC tools for scRNA-seq include *SinQC* and *SCell*. SinQC uses library quality to detect outliers in gene expression and computes different quality metrics (e.g., the total mapped reads, rate of mapping and complexity of the library) to identify the percentage of the dataset as noise. SCell is a flexible tool that permits the detection of outliers. Using the Gini index, it estimates genes that are expressed in the background, measuring statistical dispersion, and removing cells with a higher background of noise compared to the average. Another QC tool, Celloline, detects cells of low quality based on the expression profiles using biological and technical features (Ilicic et al., 2016; Poirion et al., 2016).

Alignment

After QC of the FASTQ files, aligning the reads to a known transcriptome or annotated genome is the next step. During alignment, each sequenced read is converted to one or more genomic co-ordinates (Stark et al., 2019).

Currently, there are no dedicated aligners used in scRNA-seq studies. Existing aligners developed for bulk RNA-seq are used; however, *STAR* (Dobin et al., 2013), *Salmon* (Patro et al., 2017), and *Kallisto* (Bray et al., 2016) are among the most widely used aligners. *TopHat2* (Kim et al., 2013) and *HISAT* (Kim et al., 2015) are other popular alternatives for alignment. Some of these tools combine alignment with other methods such as quantification. When comparing STAR with Kallisto, a major drawback of STAR is that it requires more memory and has a slower computation time (Du et al., 2020).

In general, STAR has both a higher speed and mapping accuracy than TopHat2 and HISAT, although it requires more memory than HISAT. Splice junctions can be detected by STAR, and it permits alignments of the sequence to the reference to be interrupted, i.e., aligns in a non-contiguous manner. It can therefore also be used to identify small nucleotide variations (SNVs) present in scRNA-seq data (Du et al., 2020; Poirion et al., 2016).

STAR is incorporated in the widely used 10x Genomics *CellRanger* software for mapping and quantification of droplet-based scRNA-seq data (Zhang et al., 2021). In addition, STAR has recently presented STARsolo, an additional algorithm to the STAR alignment software for analysing droplet-based scRNA-seq data which is directly built into STAR code. Compared to CellRanger, STARsolo is approximately 10 times faster and is designed to be a complete replacement for CellRanger. Furthermore, STARsolo performs error correction and demultiplexing of cell barcodes, error correction and deduplication of UMIs, and quantification of transcriptomic features such as splice junctions, pre-mRNA, splice and unspliced reads (Kaminow et al., 2021).

Tophat2 is a spliced aligner that can align varying read lengths generated in current sequencing techniques while permitting indels of varying lengths with respect to the reference genome. It can detect novel splice junctions and align reads across fusion breaks, which normally arise after genomic translocations (Kim et al., 2013).

HISAT (hierarchical indexing for spliced alignment of transcripts) uses an indexing scheme based on the Burrows-Wheeler transformation and the Ferragina-Manzini (FM) index. HISAT is amongst the fastest aligners available and has an increased level of accuracy (Kim et al., 2015).

Kallisto, another recently developed aligner, avoids alignment altogether by employing pseudo-alignment with hashing de Bruijn graphs. There is a marked improvement in the speed of expression quantification (Poirion et al., 2016).

Feature Quantification

In feature quantification, alignment results are converted into an expression profile. Conventionally, expression profiles are represented as a numeric matrix, where rows are genes and columns are cells, respectively. Each entry in the matrix represents the abundance of a gene or transcript in a specific cell. Feature quantification methods applied to bulk RNA-seq are used in scRNA-seq studies as well (Luecken & Theis, 2019; Poirion et al., 2016; Zhang et al., 2021).

There are different quantification approaches for generating a gene expression profile. The simplest approach used by programs like HTSeq and FeatureCounts, is to enumerate the number of reads that map within the boundaries of a gene. In the case of overlapping genes, these programs have simple but flexible parameters for determining read counts used to handle them. More complex methods use probabilistic estimates for calculating gene expression. RSEM and Cufflinks both use maximum likelihood to partition transcript counts based on models where reads in an RNA-seq sample are observed using random variables predicted from latent variables like the transcript sequence, strand, and length. The de Bruijn graph approach in the Kallisto pipeline intrinsically partitions transcript counts and it is much faster with up to two orders of magnitude in speed improvement. Remarkably, despite the simplicity of FeatureCounts and HTSeq compared to the probabilistic approaches, they have shown nearly equal or even better performance (Bray et al., 2016; Poirion et al., 2016).

Data Cleaning

After undergoing several QC steps during the pre-processing stage, the raw count matrix normally still contains many cells of low-quality and this can be attributed to dead cells present during cell isolation, inefficient reverse transcription (RT), or amplification during PCR. These

low-quality cells may present challenges for subsequent downstream analyses. Some of challenges are: the inclusion of cells of low-quality, which usually have low expression values of all genes could result in a similar pattern of gene expression and formation of clusters that may erroneously indicate a new cell type; and high sensitivity of the analysis to noise. For example, contamination from exogenous transcript of low-quality could substantially influence dimension reduction or variance estimation (Hwang et al., 2018; Kim et al., 2019a; Zhang et al., 2021).

It is therefore essential to filter out low-quality cells before any downstream analysis. QC metrics mostly used currently include *library size, number of expressed genes, percentage of reads or UMIs mapped to the mitochondrial genome, and the percentage of reads mapped to External RNA Controls Consortium (ERCC) spike-in transcripts*. Library size refers to the sum of endogenous reads or UMIs count across all genes per cell. In general, cells of low-quality have lower library sizes and it is important to filter these cells. Furthermore, cells with unusually high library sizes should be removed as they may represent multiple cells captured in a single droplet and which therefore have the same cell barcode (Bacher & Kendziorski, 2016).

The number of genes expressed is another QC metric which represents the number of expressed genes expressed detected in each cell, and it follows a similar application as with the library size. A high percentage of reads mapped to genes in the mitochondrial genome is a good indication of cells of low-quality because in a damaged cell although the mRNA can escape the permeabilized cytoplasmic membrane, the mitochondrion is too large. Retention of the mitochondria therefore leads to a very high ratio of mitochondrial transcripts in damaged cells (Zhang et al., 2021).

Another option is that synthetic ERCC spike-in RNA molecules can be used to detect biologically significant differences across cells. After adding the same amount of ERCC spike-in to every cell, low quality cells are those with high percentage reads mapped to ERCC spike-ins. The application ERCC spike-ins, however, does not apply to every scRNA-seq dataset.

Another common method is the use of fixed thresholds for removing cells of low-quality for each QC metric. Experimental experience on the part of the researcher determines these thresholds for biological systems. An adaptive threshold based on the median absolute deviation (MAD) for each QC metric can also be used. A cell is flagged for removal if it is outside the range of three MADs for each metric's median value. There is not much experience needed for

using this strategy of adaptive thresholds and is therefore very friendly to a non-expert (Bacher & Kendziorski, 2016; Zhang et al., 2021).

Removal of Confounding Factors

Systematic variations known as batch effects may be introduced when the whole dataset consists of several runs of experiment with potentially diverse conditions. Problems to downstream analysis may be caused by these artifacts, and in some cases, there is masking biological signals (Poirion et al., 2016). In studies where there is over-dispersion of gene expression, it is important to account for and remove extra variation due to the systematic differences between batches. To account for batch effects appropriately, the method of quantification together with downstream analysis must be considered. In most studies, batch effects can be removed by using down-sampling methods, but this results in the reduction of complexity (Grün & van Oudenaarden, 2015; Poirion et al., 2016). For studies employing traditional fragment counting, COMBAT is a batch effect removal method developed based on empirical Bayes frameworks and is reported to be robust to outliers for small sample sizes. It has been applied successfully to scRNA-seq data even though it was originally designed for microarray data (Poirion et al., 2016). Unsupervised batch effect removal tools correlate the batches with subpopulations detected by other scRNA-seq approaches (Finak et al., 2015).

Aside from batch-effect removal, it is necessary to eliminate the variability from technical sources present within the noise. There is a correlation between technical noise level of a gene and its average level of expression. Therefore, a probabilistic model that fits this correlation can be built using spike-ins and infer the biological variability of each gene at a later step (Brennecke et al., 2013).

In most studies, removing cell-cycle effects caused by variation results in masking of other sources of potentially interesting biological variations. The scLVM package attemps to introduce a cell-cycle factor removal step before identification of subpopulations (Buettner et al., 2015).

Normalization

A count in a gene expression matrix indicates a successful single-cell capture, reverse transcription, and sequencing of a molecule of mRNA. The read count for a gene per cell is expected to be proportional to the cell-specific scaling factors and the gene-specific expression level. The goal of normalization is to establish a consistent comparison of gene expression measurements across cells, factoring technical variation due to the number of transcripts detected for every cell. Normalizing scRNA-seq data properly is critical because the validity of downstream analyses depends on it (for instance, differential gene expression) (Bacher & Kendziorski, 2016; Hie et al., 2020; Luecken & Theis, 2019; Lun, Bach, et al., 2016).

mRNA capture, efficiency of reverse transcription, and intrinsic factors in the cell, are variables difficult to estimate and therefore modelled as fixed factors. Even though these variables can be estimated together with expression counts for normalization, the procedure is computationally challenging because fits must be made to one specific model (Hwang et al., 2018). Due to the difficulty of estimating these sources of variation, in practice, most scRNA-seq normalization adjusts for differences in sequencing depth. Scaling factors are used to normalize raw expression counts estimates by standardizing across cells. The assumption is that majority of the genes are not differentially expressed.

Even though numerous methods for normalization developed for bulk studies have been successfully applied to scRNA-seq data, sources of variation in scRNA-seq data such as dropouts, have necessitated the development of specific normalization methods for scRNA-seq data (Luecken & Theis, 2019).

Some widely used methods include reads per kilobase per million (RPKM), fragments per kilobase per million (FPKM), and transcripts per million (TPM). These three methods try to normalize for sequencing depth and gene length. RPKM is applied for single-end sequenced reads while FPKM is for paired-end sequenced reads. TPM is close to RPKM and FPKM, but the difference is in the order of operations — TPM normalizes for gene length first, followed by sequencing depth (Hwang et al., 2018). When dealing with differentially expressed genes, the afore-mentioned normalization methods may be insufficient.

Alternative methods have been developed to overcome challenges in within-sample normalization approaches. The most widely used methods are the trimmed mean of M-values

(TMM) method and relative log expression (RLE). The basic concept underlying these tools is that genes that are highly variable dominate the counts, shifting relative abundance of expression profiles. With TMM, reference samples are selected while other samples are maintained as test samples. M-values (Maximum values) are calculated as the genes' log expression ratios between tests to the reference sample for each gene. After removing genes with extreme M values, the weighted average is obtained per test sample. These two methods (TMM, RLE) show poor performance when large number of zeroes are present (Hwang et al., 2018; Love et al., 2014).

Normalization methods specific to scRNA-seq data include *scran, SCnorm, SCTransform, Linnorm, and Census*. Among these, scran is a widely used method based on pooling expression values across cells. Firstly, based on the linear regression over genes, size factors are calculated after which there is pooling across cells to prevent effects from dropout. This method has been shown to limit variability to less than 50% of differentially expressed genes between cells and performs better than other normalization methods currently available (Hie et al., 2020; L. Lun et al., 2016; Luecken & Theis, 2019).

Other normalization approaches involve normalization with spike-in transcripts or UMIs. Global scaling factors assume that RNA content is constant, but this is mostly not the case in scRNA-seq analyses because RNA content can vary with the phase of cell-cycle, cell size, and the transcriptional changes of genes. Further improvement in normalization can be obtained by using synthetic transcripts at known concentrations spiked into each cell's library. The reasoning behind this is that any difference between the observed and expected expression of spike-ins is a result of the technical artifacts. By calculating a cell-specific factor, normalized expression estimates adjust for the differences and apply this factor to endogenous genes (Bacher & Kendziorski, 2016).

Imputation

Sparse data is one key identifying characteristic of scRNA-seq data. The zero-inflated data gives rise to bimodal gene expression distribution groups of cells. Using RNA fluorescence in situ hybridization, it has recently been confirmed that gene expression occurs in a bursting fashion (on or off) (Baßler et al., 2019).

Another cause for the bimodal distribution might be because of “dropout events”. In a dropout event, a gene observed in a cell shows either zeros or close to zero expression in other cells. This suggests that some zeros in the expression matrix is not because a gene is not expressed, but its transcript might be lost due to technical reasons such as mRNA capture or sequencing problems. Dropout events obscure gene-gene relationships and increase cell-to-cell variability (Zhang et al., 2021).

Different methods for imputation and smoothing have been developed to address the influence of dropout events. The primary focus of imputation methods is the correction of zero values while methods based on smoothing attempt to correct for any value within the dataset.

Methods based on smoothing are normally used because technical noise affects the entire transcriptome and not only zeros in the data.

Imputation is an optional step when analysing scRNA-seq data and is not always recommended before the differential gene expression step in the pipeline, as any changes introduced may give rise to additional noise.

Feature Selection

A typical human scRNA-seq dataset usually contains gene expression values for up to about 25,000 genes. Majority of these genes will be uninformative (e.g., house-keeping genes exhibit expression values that are similar across cells and not useful for investigating heterogeneity). Also, it is likely that many genes contain zero counts. There remains more than 15,000 dimensions after removing genes with zero counts from the data during QC (Luecken & Theis, 2019).

The goal of feature selection is to exclude uninformative genes and focus on genes that drive biological variation. This reduces the computational workload on subsequent downstream analysis steps and reduces the count matrix size and noise in the data (Luecken & Theis, 2019).

Seurat and Scanpy implement a method for selecting highly variable genes (HVGs). Mean expressions are used to bin genes and genes having the highest variance variance-to-mean ration are used as HVGs per bin (Luecken & Theis, 2019). Scran implements a feature selection step based on modelling the mean-variance trend in the data and fitting all the genes to simulate poisson counts (Lun, McCarthy, et al., 2016).

Downstream Analysis

Dimensionality Reduction

Algorithms for dimensionality reduction are used to reduce the dimensions of the gene expression matrix. They capture the data in the least number of dimensions and therefore make downstream steps more computationally efficient.

Other important goals of dimensionality reduction are visualization and summarizing the data. Visualization uses two or three dimensions to describe the data and use these as coordinates to represent the data on a scatter plot. On the other hand, summarization reduces the data to its basic components by looking for dimensions present, which is needed for downstream steps. A summarization method that uses the top reduced dimensions is normally used (Luecken & Theis, 2019).

Dimension reduction methods can be placed in four main categories which are discussed below (Zhang et al., 2021).

Linear models

Principal component analysis (PCA) remains the most traditional method used for dimension reduction. It identifies the largest amount of variance in the data by reducing the dimensionality. Because PCA only models linear patterns in the data, it is not a conclusive method for analysing complex scRNA-seq data. Another technique, zero-inflated factor analysis (ZIFA), is commonly used for dimension reduction and it considers frequency of zeros present in scRNA-seq data. f-scLVM (factorial single-cell latent variable model) models annotated gene sets and allows for easy interpretation of the reduced dimensions (Hie et al., 2020; Luecken & Theis, 2019; Zhang et al., 2021).

Non-linear models

Due to the non-linear structure present in scRNA-seq data, models developed to match this have great potential to provide useful insights.

One of such widely used techniques is t-SNE (t-distributed stochastic neighbour embedding). A problem with t-SNE is that, it can handle only local structures including trajectory analysis, and therefore reduces its performance. Uniform manifold approximation and projection (UMAP)

which was recently developed maintains local and global structures in the data (Luecken & Theis, 2019; Zhang et al., 2021).

Deep learning-based methods

There has been a rise in the use of deep learning methods for analysing scRNA-seq data as they can capture the nonlinear features underlying the data. Deep count autoencoder (DCA) employs zero-inflated negative binomial model to denoise the scRNA-seq data. Single-cell variational inference (scVI) is based on a probabilistic model that uses a neural network to determine each gene's expression uncertainty. Both local and global structures present in the data are preserved (Zhang et al., 2021).

Clustering

A common goal in scRNA-seq data analysis after dimension reduction is clustering of cells to determine heterogeneity in a sample. Clustering is an important step that differentiates cells into different groups allowing one to infer the identity of member cells based on similarities of their gene expression profiles. It is expected that an optimal clustering technique separates cells of the similar type into similar groups.

If optimal clustering is not attained, the outcome may be: underclustering, overclustering, or cluster splitting. Underclustering occurs when different cell types are assigned to one cluster, distorting any variation present in the data. In overclustering, numerous clusters represent cells of similar type. Cluster splitting is an occurrence when similar cell types are spread among different clusters which is dominated by other cell types at the same time. A combination of these problems is normally present in diverse cell types.

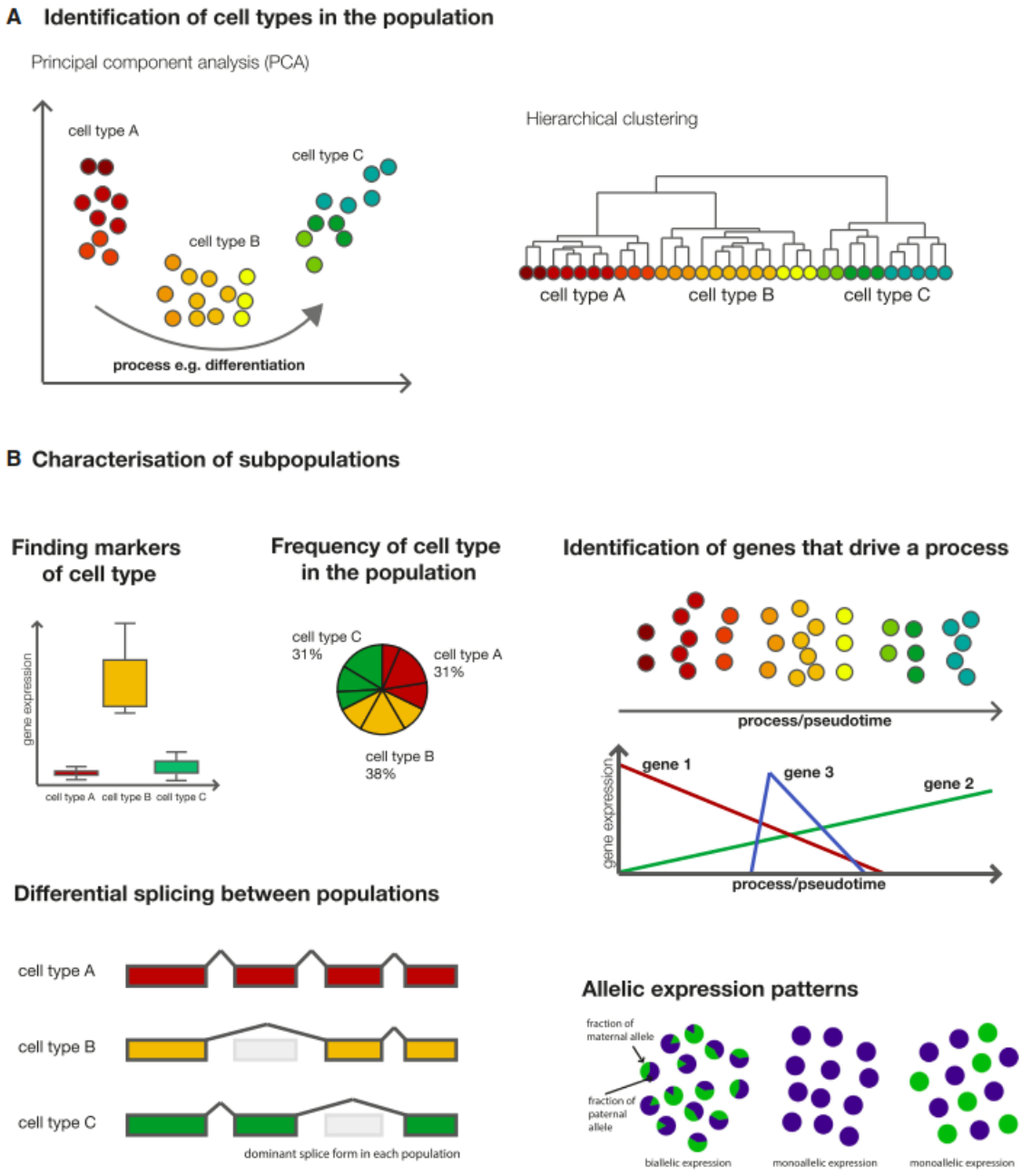

Figure 2.9 | (A) Cell populations are identified with principal component analysis hierarchical clustering approaches. (B) Diverse methods of characterizing cell subpopulations. These approaches identify cell type markers for analysing differential expression, cell population frequency, determining genes that exhibit particular patterns in developmental processes, analysis of differential splicing and allele-specific expression analysis. Image modified with permission from Kolodziejczyk *et al.*, 2015.

Identifying and Describing Cellular Subpopulations

scRNA-seq allows the identification of cellular subpopulations from heterogenous groups of cells. Because the structure of a complex tissue is linked to its function, determining cell types and their frequency is very important. Also, analysing transitions between cellular states during differentiation or development, can shed new light into gene regulatory mechanisms. These transitional types are either gradual or binary and involve one or numerous intermediate states. Understanding these transitions and their intermediate states can result in the identification of key genes that regulate these processes (Kolodziejczyk et al., 2015).

Later, subpopulations can be characterized or matched to known cell types after they are identified. Markers for specific populations can be found using differential expression or correlation analysis (Kolodziejczyk et al., 2015).

Trajectory Inference

Clustering alone cannot sufficiently describe cellular diversity. Biological processes accounting for heterogeneity are continuous. *Trajectory inference* (TI) aims to capture transitions amongst cells and differentiation processes. TI interprets scRNA-seq data in terms of a snapshot of an ongoing biological process. These processes are reconstructed by identifying paths through cellular space that reduces transcriptional changes between cells. A pseudotime variable describes the ordering of cells along these paths. This is mainly interpreted as a proxy for developmental time although it is related to distances in transcription from the base cell (Luecken & Theis, 2019).

Monocle and Wanderlust were pioneers of TI for scRNA-seq (Luecken & Theis, 2019; Poirion et al., 2016). Monocle applies a Minimum Spanning Tree (MST) on the constructed graph using Euclidean distance between pair of cells. The MST then connects all the nodes in the graph using their edges. Other TI methods use different approaches to model the differentiation paths ranging from simple linear or branching trajectories. For example, PAGA is used for modelling complex trajectories (Luecken & Theis, 2019). TSCAN uses an MST-based technique. It requires cells to be clustered before MST construction, therefore reducing the complexity of tree space (Poirion et al., 2016).

Workflows and Workflow Management Systems

Bioinformatic analysis is a multi-step procedure which involves directing files through different transformations referred to as a pipeline or a workflow to analyse, manage, and share complex data. For instance, in a typical scRNA-seq pipeline, preprocessing, QC, normalization, etc., need to be automated in a stepwise fashion to advance standardisation, performance and re-usability. It is required that a workflow runs these processes in the right order and on the right data. These transformations are typically achieved by running executable command line software written for Unix-like operating systems and writing a script that automates the steps (Jackson et al., 2021; Leipzig, 2016). Makefiles and traditional shell scripts are limited in terms of reporting errors as well as being difficult to debug and maintain. This makes it challenging to restart after they go wrong and difficult to transfer files between different computer architectures (Jackson et al., 2021).

Workflow management systems (WfMS) address these problems and are designed to provide a common language to describe, manage, and execute analyses. They contribute to reproducible pipelines and build libraries with reusable components, thus supporting both incremental building and re-entry. In the presence of additional input or configurational changes they provide the ability to re-execute portions of a workflow, and if interrupted, they can resume execution where a workflow previously halted (Jackson et al., 2021).

WfMS can be executed on desktops, high-performance computing systems (HPC) or cloud environments across systems and they provide support for software containers (for reproducibility). The declarative style of most WfMS allows bioinformaticians to focus on specifying what the WfMS should do. Some widely used WfMS currently in use are discussed below (Goble et al., 2020; Jackson et al., 2021).

Nextflow

Nextflow is a WfMS that incorporates a runtime environment and a programming domain specific language (DSL) for writing computational pipelines. It facilitates reproducible and scalable scientific workflows using software containers. It enables the adaptation of pipelines written in most scripting languages and chains together powerful command-line and scripting tools for very complex data processing. Nextflow defines complex interaction of programs and a computing environment that is highly parallel. Nextflow uses the dataflow programming model,

where one output of a process is connected to input of another process and an execution is began as soon it is received. Nextflow scripts are written in the Groovy programming language (Di Tommaso et al., 2017; Jackson et al., 2021).

Core features of Nextflow include fast prototyping, which allows one to write a simple computational pipeline that can combine several different tasks. It also permits the reuse of existing scripts and tools. Another feature is reproducibility, where Nextflow supports containerization technologies such as Singularity and Docker, as well as package managers like Conda. With the integration of version control via widely used code sharing platforms like GitHub, Nextflow allows one to write self-contained pipelines, manage versions and reproduce configurations. Nextflow is highly portable in the sense that it provides a layer of abstraction between the execution layer and pipeline logic, making it executable on multiple platforms without changing, e.g., on a local computer, and HPC Cluster. The unified parallelism of Nextflow based on the dataflow programming model extremely simplifies splitting tasks that can be run at the same time (parallelisation). The implementation of continuous checkpoints in Nextflow makes it possible to automatically track all intermediate results produced during the pipeline execution. This allows one to resume execution from the last successfully executed step if there is an interruption (Di Tommaso et al., 2017).

Snakemake

Snakemake is another WfMS that adopts a similar model of the GNU Make automated build tool where users supply build files as output. Snakemake uses rules (in python or bash scripts) to build output files from the main workflow. Snakemake rules can creates dependencies that can be used by commands to build output files.

Snakemake is flexible and it can interoperate with any installed tool or web service insofar as input and output formats are well-defined. Also, Snakemake is portable since only a Python installation is needed to run Snakefiles. Lastly, it optimizes parallel processes against CPU cores and threads to provide automatic scalability and can utilize single machines or cluster engines without changing the workflow (Koster & Rahmann, 2012).

Common Workflow Language (CWL)

CWL describes command line tools and chains them to build workflows. It is not a software but rather a specification. CWL workflows have high portability across different systems and are

developed based on the CWL standard. CWL has adopted the GNU Make style and determine execution order based on dependencies between tasks. However, unlike Make, CWL tasks are isolated, so inputs and outputs must be declared explicitly. CWL can be used with containerization technologies like Docker and can also scale up to large workflows in HPC, cloud and cluster environments (Crusoe et al., 2022).

Toil

Toil is a WfMS that runs scientific workflows on large scale in either the HPC environments or in cloud, and includes features for large scale analyses across different environments. Toil is also designed to execute CWL and provides draft support for other workflow framework languages. It provides a Python application programming interface (API) that permits the dynamic or static generation of workflows in a way such that jobs can define other jobs during execution. These jobs can incorporate Docker containers making workflows reproducible. Toil also has built-in support for servers and databases (Vivian et al., 2017).

Fair principles and workflow management systems

The proposal of FAIR guiding principles (Wilkinson *et al.*, 2016) aims to improve using scientific data by making it findable, accessible, interoperable, and reusable (FAIR). These principles are highly applicable to workflows as well (Lamprecht et al., 2020). With workflows, FAIR principles apply in two main areas: *FAIR data* and *FAIR criteria* for workflows as digital objects. Workflows should be designed to meet the criteria of FAIR principles, describing the data involved in a formal and traceable way.

Containerization

Currently, containers address problems of sharing bioinformatics tools to allow for reproducing analysis across different computing environments. *Containerization* uses an isolated environment to package an application and its dependencies. Typically, a bioinformatic pipeline contains over 15 tools, each of which have different library dependencies. Taking into consideration the fact that some tools are seldom updated, pipelines require a significant amount of installation and configuration effort (Fjukstad & Bongo, 2017).

Widely used implementations of software containerization include *Docker and Singularity*. A configuration file can be used to build a container and install all required dependencies. The configuration file and container can be shared between different computers without installation

of additional software. With this, one can rerun the container with same libraries. Containers can be run in parallel manner on local desktops or on HPC environments (Fjukstad & Bongo, 2017).

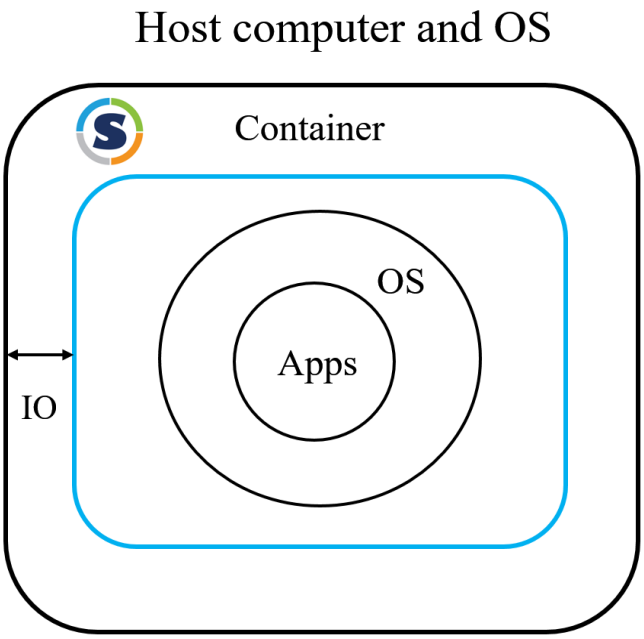

Figure 2.9.1 | An illustration of how containers interact with the underlying operating system and host applications. The container platform singularity is used here.

Current scRNA-seq Pipelines

Numerous efforts have been made towards the development of scRNA-seq pipelines incorporated in WfMSs for analysing the data generated.

scAmpi (Single Cell Analysis mRNA pipeline) is a pipeline that analyses scRNA-seq data from raw reads to gene expression, and identification of potential drug candidates. It was developed with the Snakemake WfMS (Köster & Rahmann, 2012) and implemented to process scRNA-seq data applied in the clinical setting. It uses the Cellranger software to assign reads per cell based on the barcode list supplied from 10x Genomics. Subsequently, the assigned reads are mapped to a reference genome to obtain a gene expression matrix. Filters to remove dead cells, doublets, and contaminants are implemented (Bertolini et al., 2022).

Cell type identification and unsupervised clustering are used to determine sample composition. Phenograph is used for clustering, and it compares gene expression profiles across cells. In its differential gene expression step, it uses multiple linear regression to perform two comparisons for scRNA-seq data. Also, gene expression pathways are visualized per cell with a UMAP. Differentially expressed genes in clusters are used to query the Drug Gene Interaction Database (DBIdb) for in-silico drug candidate identification (Bertolini et al., 2022).

Another scRNA-seq pipeline, implemented by the nf-core, is written in Nextflow (Di Tommaso et al., 2017). It is a modularized pre-processing pipeline with steps for QC using Alevin, and mapping using STARSolo or Kallisto. Downstream analysis is based on the Cellranger pipeline and involves clustering and gene expression analysis (Ewels et al., 2020).

Bollito is an automated scRNA-seq pipeline implemented in Snakemake (Köster & Rahmann, 2012). It incorporates pre-processing steps for quality control with FASTQC, multiQC, and Seurat. Downstream steps including differential gene expression and functional analysis and trajectory inference are performed with Seurat (García-Jimeno et al., 2022).

CHAPTER 3: METHODS

Choice of Workflow Management System

To implement the pipeline for analysing scRNA-seq data, a WfMS was required. The WfMS defines processes for execution of each stage in analysing the scRNA-seq data. The input from a given process is fed into the subsequent process in an automated manner to process the data. The pipeline was built on a Linux system (*Ubuntu 20.04.4*) with 16 gigabytes (GB) of random access memory (*RAM*) and core i7 3.60GHz Octacore CPU.

Different WfMS were reviewed based on the following criteria: *ease of implementation* (number of dependencies required), *operating systems* (Linux, Windows, high performance computing or in the cloud), *extensibility* (addition of more functionality), *flexibility* (in defining and manipulating its processes), *ease of use* (how much additional programming expertise), *generalizability* (number of packages or software it can work with) and *resource use* (requirements for its own server, database server, etc.).

Table 3.1 | Workflow Management Systems (WfMS) reviewed and their respective websites

WfMS	***Website***
Nextflow	https://www.nextflow.io/
Snakemake	https://snakemake.readthedocs.io/
Common Workflow Language (CWL)	https://www.commonwl.org/
Toil	https://toil.ucsc-cgl.org/

The strengths and weaknesses of the various WfMS were reviewed and tested for a period of one-month. This task was shared amongst members in the Bioinformatics group in the Division of Molecular Biology and Human Genetics (MBHG). After the review, *Nextflow* (*version 21.10.6*) (Di Tommaso et al., 2017) was selected as the best WfMS for the purpose considering the afore-mentioned criteria. Complete installation guidelines for installing N*extflow* can be found on their website: https://www.nextflow.io/ .

Selection of High-Throughput Tools for the Pipeline

The pipeline consists of two main stages: *pre-processing and downstream analysis.* The tools implemented in each stage were selected based on robustness, ability to handle different scRNA-seq datasets, and long-term maintenance as reviewed in C*hapter 2*.

Containerization

The platform chosen for building containers for the pipeline was *Singularity* (*version 3.5*) (Kurtzer et al., 2017). Singularity was selected based on its ability to run complex applications on HPC clusters in a very simple, portable, robust, and reproducible manner. The containers were built for software packages for both pre-processing and downstream analysis. Complete installation guide of singularity can be accessed on their website: https://docs.sylabs.io/guides/3.5/user-guide/introduction.html

Provenance of Data Set

The pipeline was tested with two publicly available datasets generated with the two main scRNA-seq protocols: droplet-based (10x Chromium v2; 10x Genomics) and plate-based (Smart-seq2) (Picelli et al., 2014).

Dataset 1

An scRNA-seq data set on healthy human primary airway epithelial cells was obtained from the Gene Expression Omnibus (GEO) of the National Centre for Biotechnology Information (NCBI) website. The GEO accession number of the study was GSM5604585. The fifth technical replicate out of five from the experiment was selected. FASTQ files from the first sequencing run (run ID, SRR16121339) was downloaded from the NCBI website. The minimal dataset was used in the pipeline to consistently generate reproducible results.

This dataset was generated from well-differentiated, primary human airway epithelia at the air liquid interface, which were dissociated and prepared for scRNA-seq using 10X Genomics Chromium Single Cell 3' reagent kits v2 system and sequenced on Illumina HiSeq 4000 (https://www.10xgenomics.com). The layout of the library for sequencing was paired.

Dataset 2

The second dataset was obtained from the study of Segerstolpe *et al.,* 2016, who studied the single-cell transcriptome profile of human pancreatic islets in individuals with and without type

2 diabetes. Raw fastq files for this dataset were not available; therefore, the pre-processed dataset was obtained from ArrayExpress (https://www.ebi.ac.uk/biostudies/arrayexpress) with ID's E-MTAB-5061 AND E-MTAB-5060, respectively.

This dataset was generated using human islets samples (85%-95% pure) collected from healthy and diseased donors. The cells were cultured, dissociated, and distributed by FACS into 384-well plates. scRNA-seq libraries were prepared with the Smart-seq2 protocol (Picelli et al., 2014). The sequencing was performed on an Illumina HiSeq 2000 generating 43 bp single-end reads (Segerstolpe et al., 2016).

Data Analysis (Preprocessing and Downstream Analysis)

scRNA-seq data analyses are divided into two main categories: *pre-processing and downstream analyses*. The pre-processing consists of QC, alignment to reference genome (mapping), generation of a gene expression matrix, normalization, feature selection, and visualization.

QC was also done after alignment. Dedicated bioinformatics tools chained together with *Nextflow*, were used to develop a pipeline that automates pre-processing of the data. Containerization technology implemented in the pipeline ensured it was reproducible across different computer architectures. Report on QC graphs and tables was prepared in the form of an R markdown file.

Downstream analyses were carried out with R Bioconductor packages and consisted of dimensionality reduction, clustering, cell type annotation, differential gene expression and trajectory inference. Different computational algorithms and statistical approaches were implemented to generate insight from the dataset.

A report on the outcome of the downstream analyses steps was prepared in PDF and HTML.

Table 3.2 | Software tools implemented in the pipeline and their versions

Pre-processing	Downstream
Pre-Quality control (*FASTQC, v0.11.9*)	Dimensionality reduction (*scran, v1.26.0*)
UMI-Extraction (*UMI-tools, v1.1.2*)	Clustering (*scater, v1.26.1*)
Trimming (*Trimmomatic, v0.39*)	Cell type annotation (*celldex, v1.8.0*)
Post-Quality control (*FASTQC, v0.11.9*)	Differential expression (*scran, v1.26.0*)
Alignment (*STAR, 2.7.10a*)	Trajectory inference (*TSCAN, v1.36.0*)
Normalization (*scran, v1.26.0*)	

FASTQC https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

UMI-tools https://umi-tools.readthedocs.io/

Trimmomatic https://github.com/usadellab/Trimmomatic

STAR https://github.com/alexdobin/STAR

scran https://bioconductor.org/packages/scran/

scater https://bioconductor.org/packages/scater/

celldex http://www.bioconductor.org/packages/celldex/

TSCAN https://www.bioconductor.org/packages/release/bioc/html/TSCAN.htmlHi

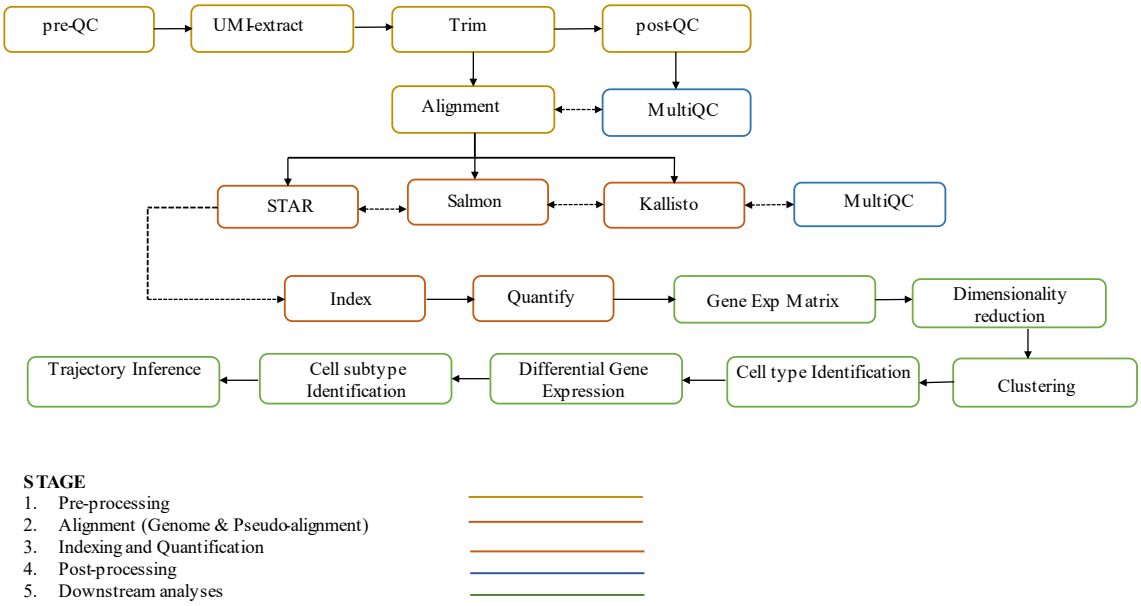

Figure 3.1 | Summary of scRNA-seq pipeline framework.

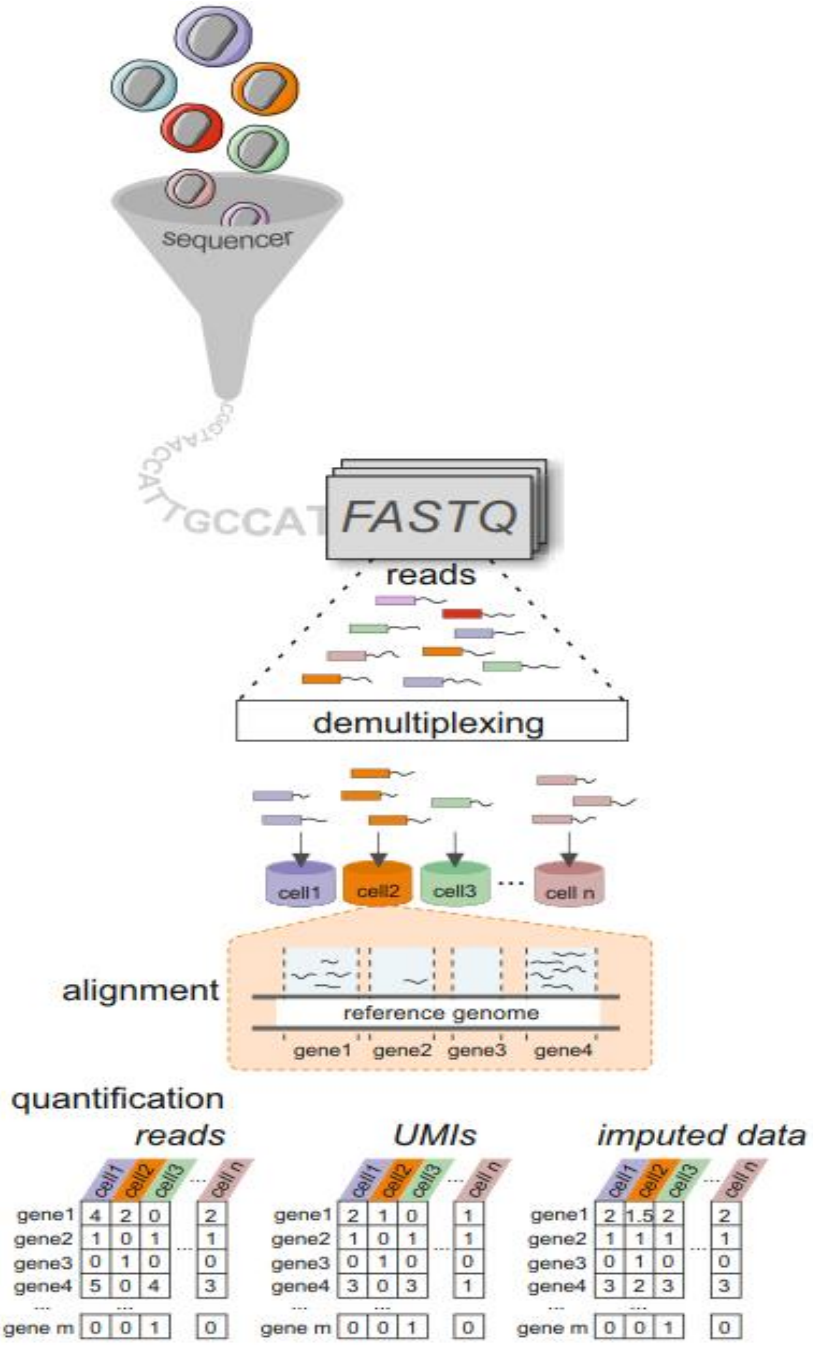

Figure 3.2 | Stages of pre-processing implemented in the pipeline. Image modified with permission from Baßler *et al.*, 2019.

CHAPTER 4: RESULTS

IMPLEMENTATION OF PIPELINE

After developing the pipeline with N*extflow*, it was tested with public datasets generated from the two main scRNA-seq protocols: droplet and plate-based techniques.

PREPROCESSING THE DATA

The droplet-based dataset pre-processed in the pipeline generated outputs in the following order: *pre-quality control, umi-extraction from data, trimming of sequencing adaptors, post-quality control, extensive quality control (with multiQC), alignment, and count matrix generation.*

Pipeline execution

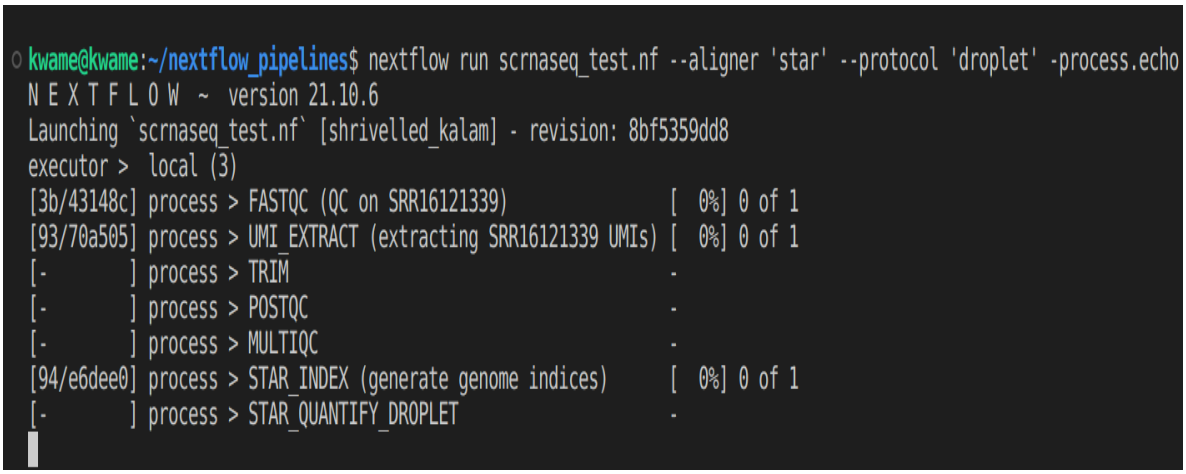

Figure 4.1 | Testing of preprocessing pipeline execution with sample scRNA-seq droplet-based dataset (GSM5604585). Here, *Nextflow* is being run on the Linux command line interface (CLI).

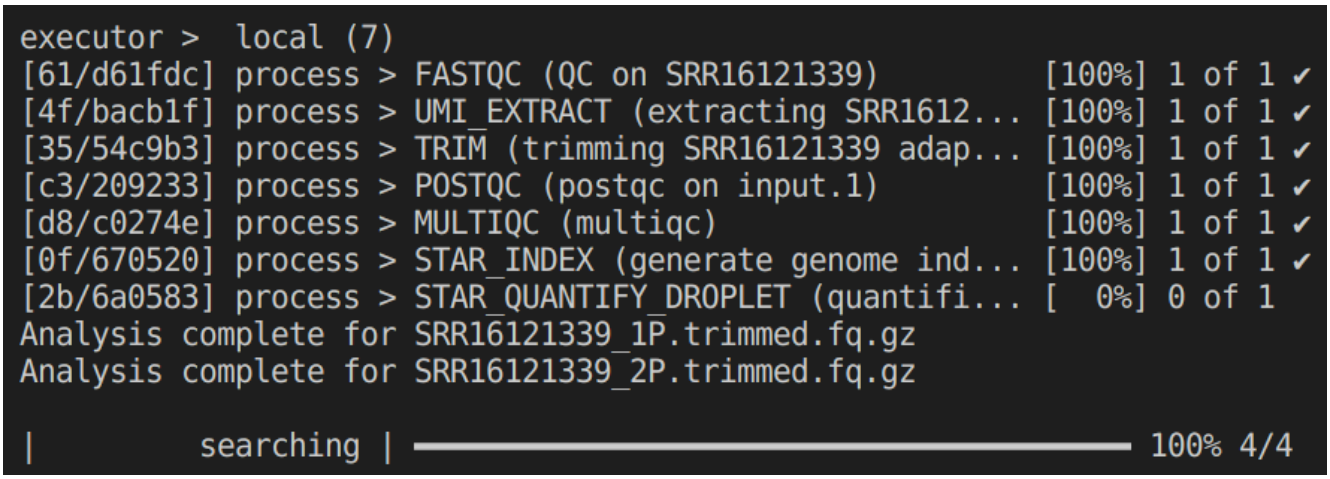

Figure 4.2 | Near end of pipeline execution with scRNA-seq droplet test dataset (GSM5604585). Alignment is being done with STAR aligner.

It took nearly 45 minutes for the pipeline to complete pre-processing of the droplet dataset (GSM5604585) on Ubuntu Linux, 16GB and core i7 3.60GHz Octacore CPU (Figure 4.2)

Quality control

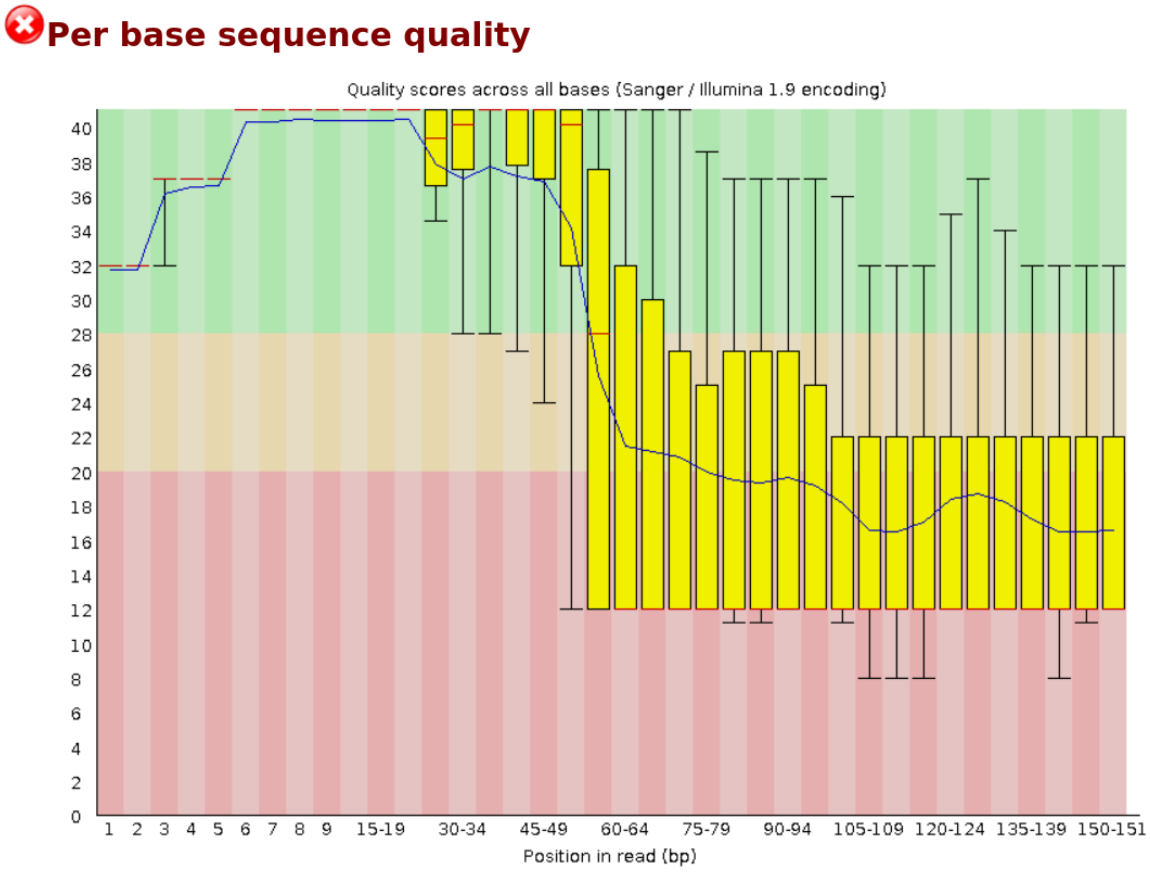

Figure 4.3 | First QC of reads being flagged as low quality (SRR16121339_2.fastq.gz).

At this stage, QC metrics for the paired reads that were flagged as passed include: *basic statistics, per base sequence quality, per tile sequence quality, per sequence quality scores, per sequence GC content, per base N content, sequence length distribution, sequence duplication levels*.

Per base sequence quality and per base sequence content were flagged as not passed (Figure 4.3).

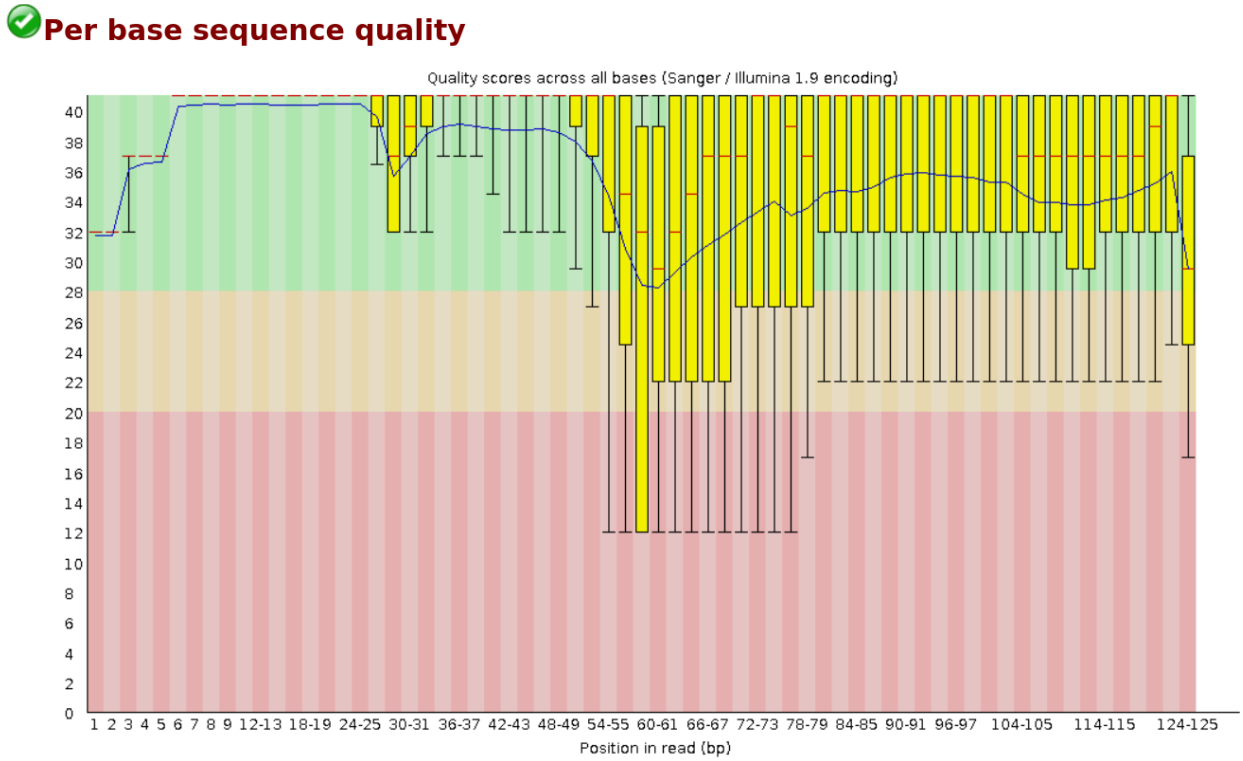

Figure 4.4 | Quality control of trimmed reads after initial QC steps. Reads here are flagged as passed (SRR16121339_2.fastq.gz).

Output directories and files

After performing QC steps, all QC metrics were flagged as passed, except for *per base sequence content*.

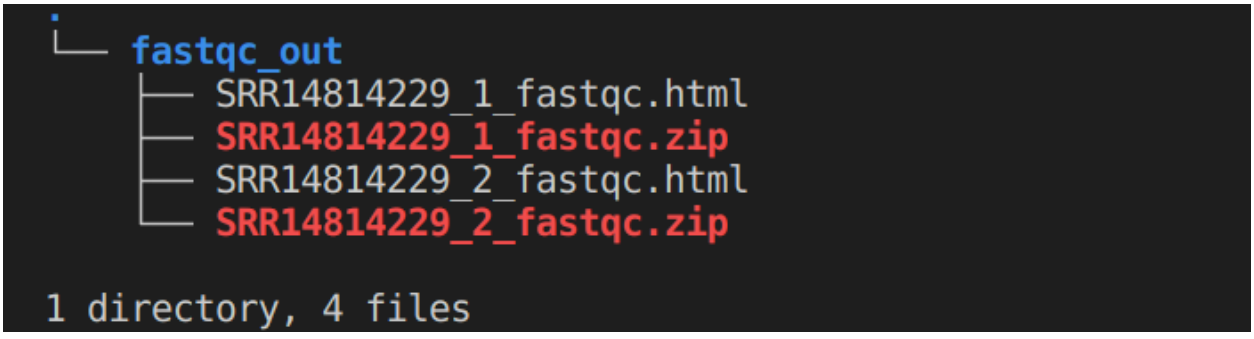

Figure 4.5 | Output directory showing pre-quality control of reads using *FASTQC* (GSM5604585).

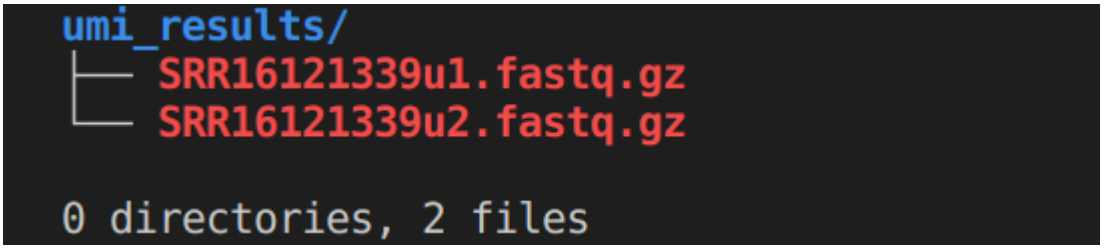

Figure 4.6 | Output directory showing Unique Molecular Identifiers (UMI) extracted from reads using *UMI-tools* (GSM5604585).

The 10x Chromium v2 sequencing chemistry, which was used for library preparation of the sequenced reads, uses cell barcodes (CB) and UMIs. The sequencing read 1 contains 16 base pair (bp) cell barcodes and 10 bp UMIs. These are used to demultiplex (assign sequences to their sample of origin) and deduplicate reads (arising from PCR duplication). The 26 bp barcodes (CB + UMIs) were removed with *UMI-tools* from read 1 and appended to the read name. Also, UMI-extraction is beneficial because some post-processing tools in the pipeline require reads with UMIs extracted (for example, *alevin*) (Figure 4.6).

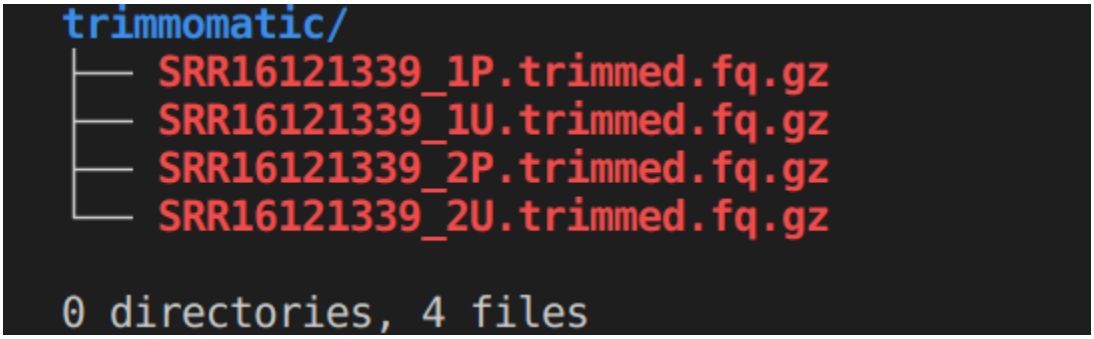

Figure 4.7 | Output directory showing trimmed reads from the UMI stage using *Trimmomatic* (GSM5604585).

Adapter sequences might be added during the sequencing of reads. The reads output at this point contains the sequence of interest and adapter sequence (Figure 4.7). It is necessary to remove the adapters from the reads for subsequent processing of the data. The paired-end mode in *Trimmomatic* handled the adapter trimming. It was used due to its flexibility in parameters, proper handling of paired-end data, and its high performance (Bolger et al., 2014).

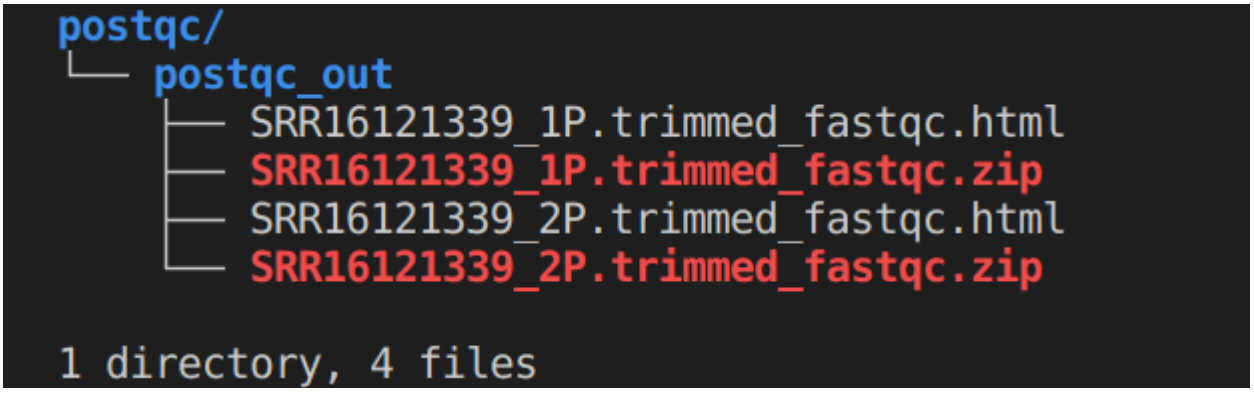

Figure 4.8 | Output directory showing post quality control of trimmed reads using *FASTQC*.

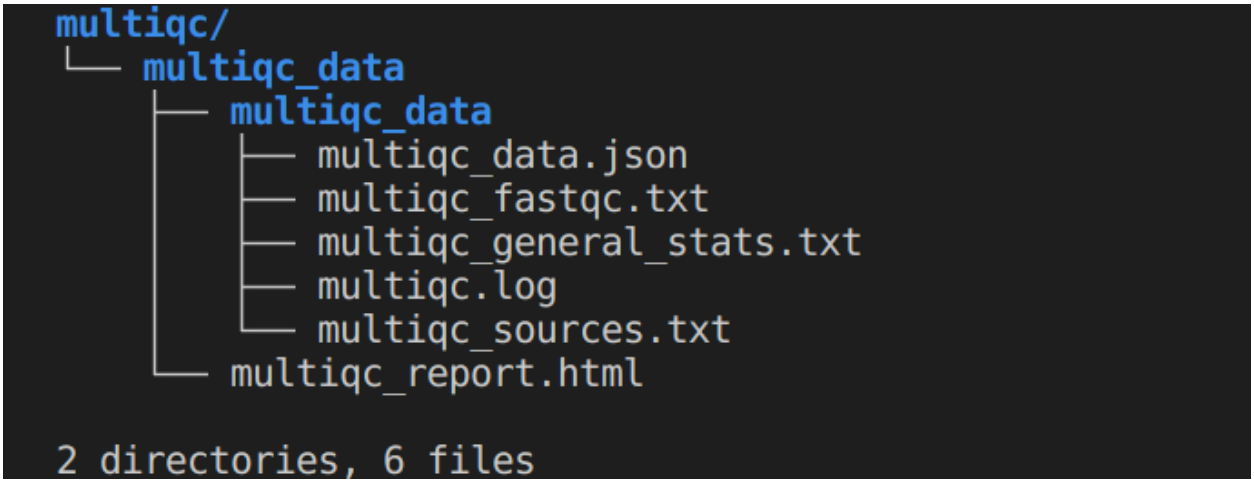

Figure 4.9 | Output directory showing MultiQC results of post quality control reads (GSM5604585).

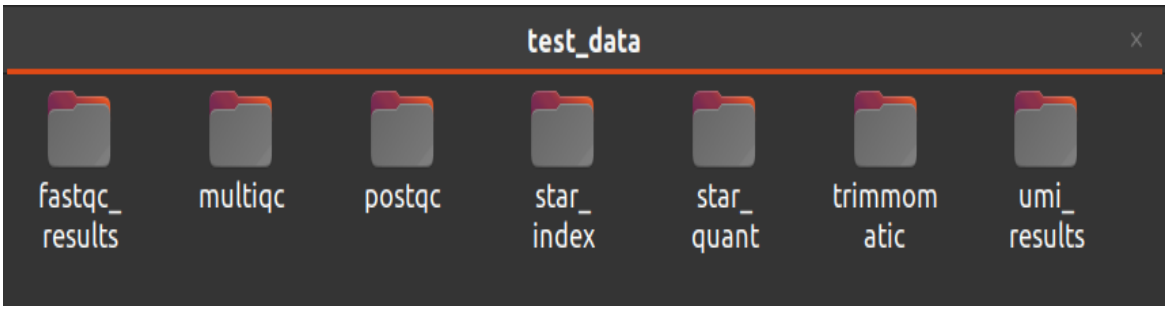

Figure 4.10 | Results directory with final output directories of pipeline processes (GSM5604585).

Gene expression matrix and further preprocessing

```
> dim(air.ec)
[1]  36601 737280
>
> air.ec
class: SingleCellExperiment
dim: 36601 737280
metadata(1): Samples
assays(1): counts
rownames(36601): ENSG00000243485 ENSG00000237613 ... ENSG00000278817
  ENSG00000277196
rowData names(3): ID Symbol Type
colnames: NULL
colData names(2): Sample Barcode
reducedDimNames(0):
mainExpName: NULL
altExpNames(0):
```

Figure 4.11 | Initial gene expression matrix depicting the dimensions of the pre-processed data. Here, using 737,280 single lung epithelial cells 36,601 genes were mapped for each cell (GSM5604585).

Downstream insights in scRNA-seq data are generated from the gene expression matrix (Figure 4.11).

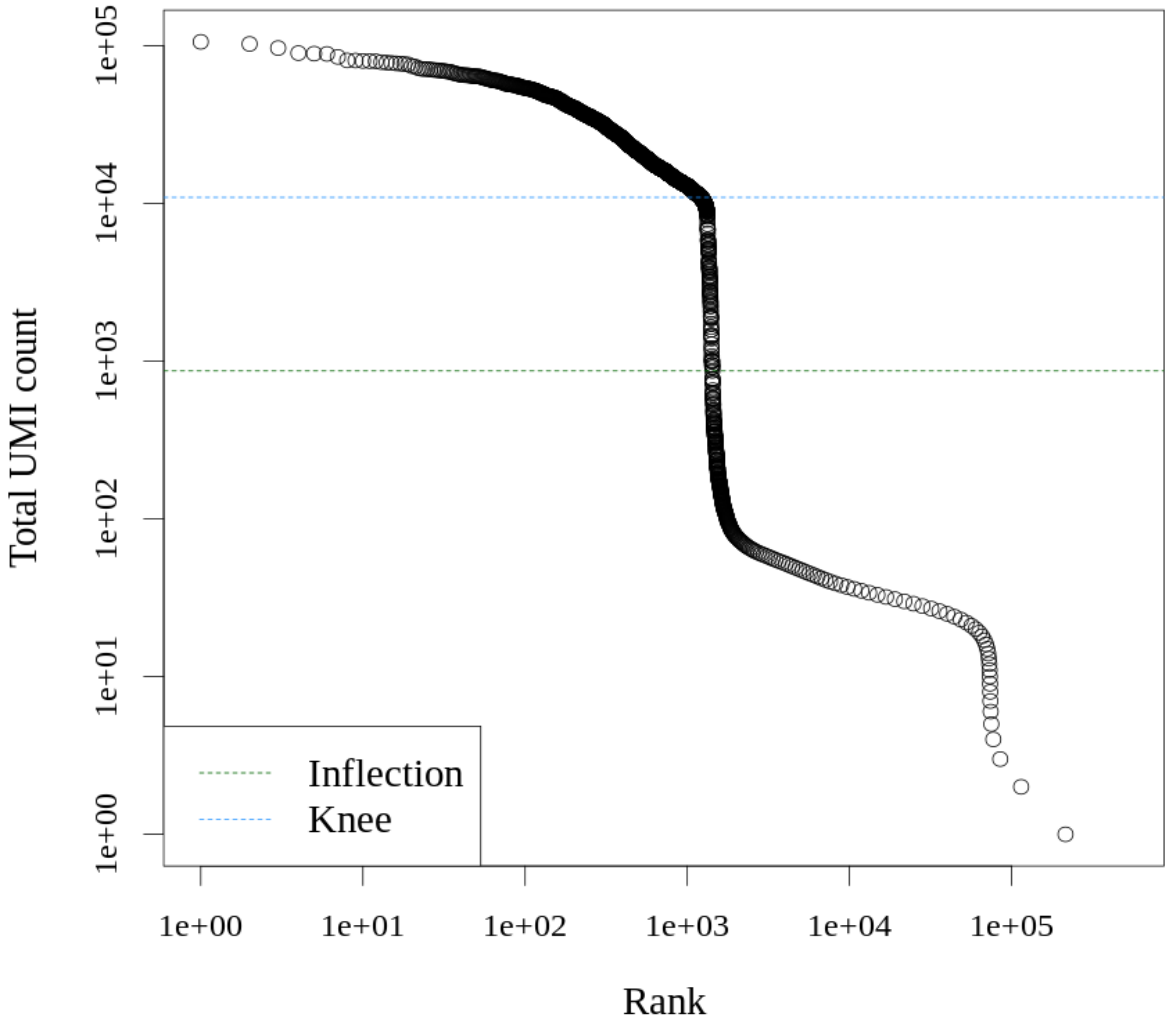

Figure 4.12 | Total UMI count for each barcode plotted against its rank for the droplet-based dataset (GSM5604585)**.**

When processing droplet-based results, the data contain barcode-based libraries from both cell-containing and empty droplets, but without annotation of which barcodes correspond to empty or cell-containing droplets (Figure 4.12). The determination of which droplets contained cells must be based on the observed expression profiles. The difference in the distribution of total UMI counts of cell-containing and empty droplets, can be used for this classification.

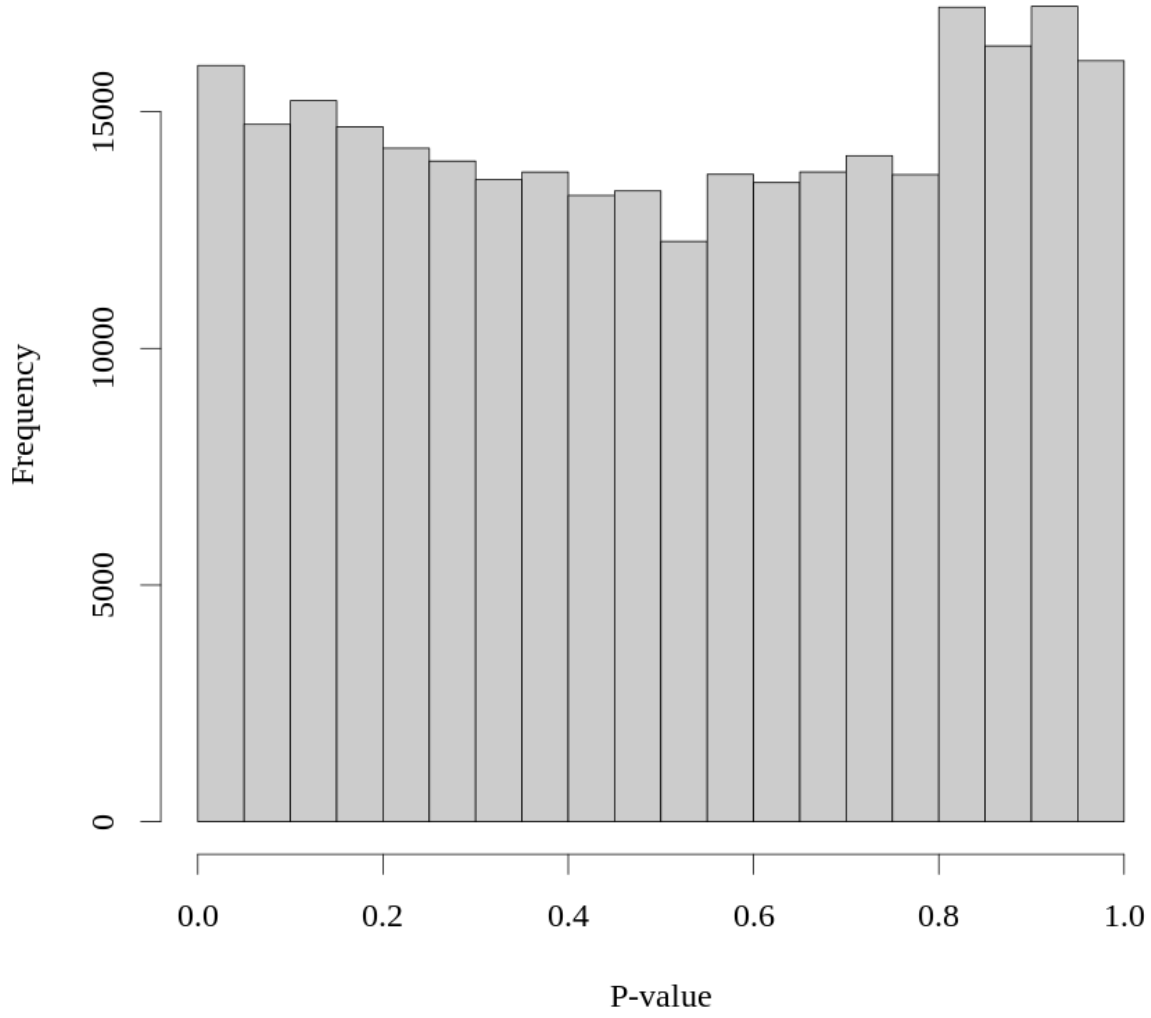

Figure 4.13 | Testing for empty droplets using Monte Carlo simulations to compute p-values for the droplet-based dataset (GSM5604585).

The *emptyDrops* function from the *DropletUtils* package was used to compare the expression profile of each cell barcode and the general RNA pool and determine whether the expression profile associated with each cell barcode differs significantly from that of the free extra-cellular RNA pool resulting from ruptured cells in the solution (Lun et al., 2019). Profiles were deemed significantly different, using a false discovery rate (FDR) of 0.1%, thereby tolerating the inclusion of less than 0.1% of the barcode libraries that were empty droplets (Figure 4.13).

Sampling of transcripts from the extra-cellular RNA pool that will be present randomly in empty drops is a multinomial process and *emptyDrops* function uses Monte Carlo simulations to compute *p*-values for this multinomial sampling. As with all empirical p-values, the number of Monte Carlo iterations determines the lower bound for resulting *p*-values. *emptyDrops* assumes, a priori, that barcodes with low total UMI counts are from empty droplets. The null hypothesis here should be true for all these barcodes. The distribution of barcodes with low-total counts is close to uniform. This demonstrates that the null hypothesis is indeed a random process. Large peaks near zero indicate that the total counts of barcodes do not all originate from extra-cellular RNA. To ensure that this peak does not influence the estimation of the null profile of the extra-cellular RNA, data from droplets with near zero counts are excluded when generating the profile (Amezquita et al., 2020; Huber et al., 2015; Lun et al., 2016).

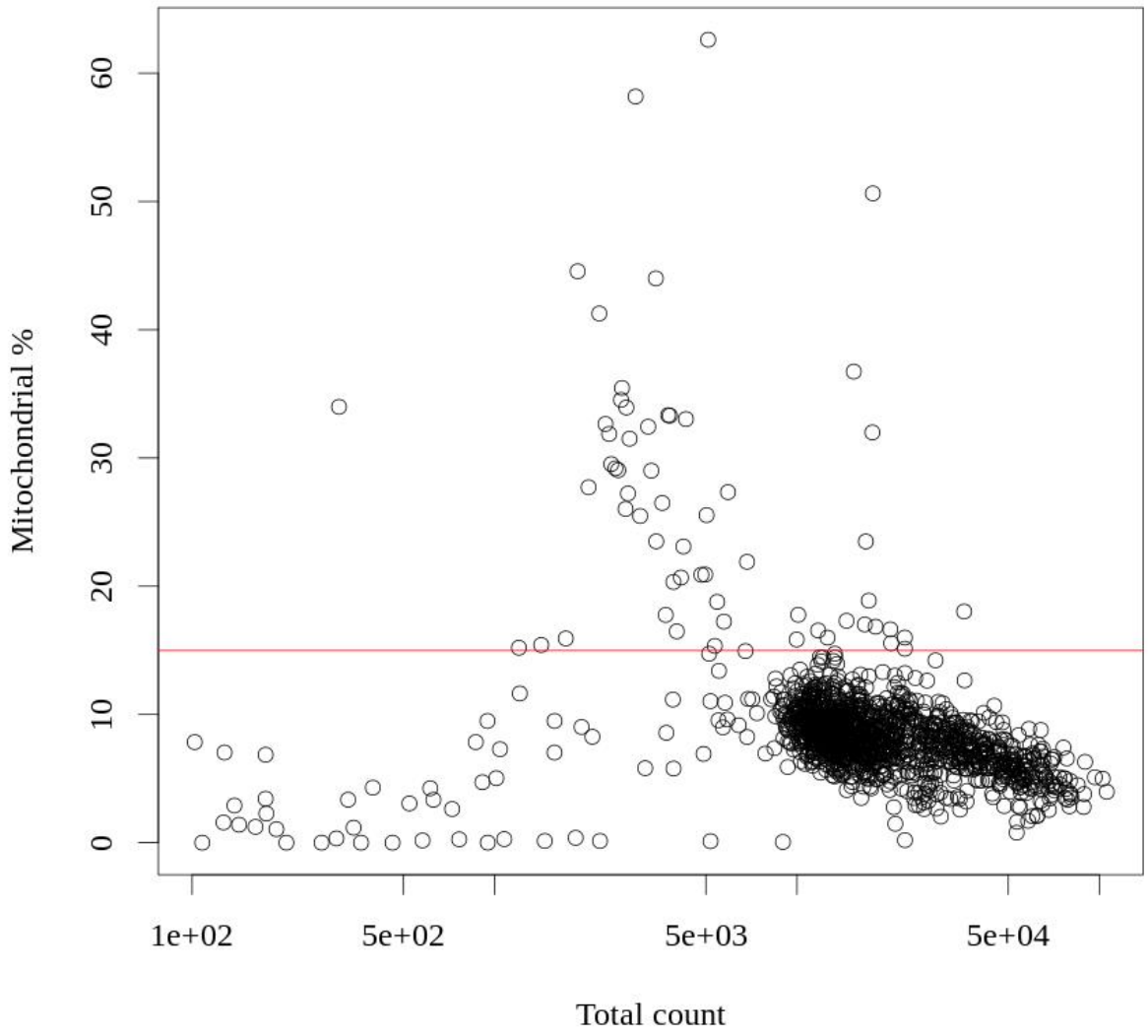

Figure 4.14 | Quality control using mitochondrial counts to remove low-quality reads. The red line represents the threshold applied to remove low-quality cells. Here, outliers are 3 MADs (Median Absolute Deviations) from the mean mitochondrial percentage.

Cells of poor quality and potentially damaged are identified based on the ration of mitochondrial to cytoplasmic RNA. Permeabilization of the cytoplasmic membrane will result in leakage of cytoplasmic RNA, but since mitochondrial RNA is protected by another membrane, the mitochondrial RNA will be retained in the cell and therefore there will be an excess of

mitochondrial RNA. The outliers detected here are 3 MADs from the mean mitochondrial proportion (Figure 4.14). (Amezquita et al., 2020; Lun, Bach, et al., 2016; Stegle et al., 2015b).

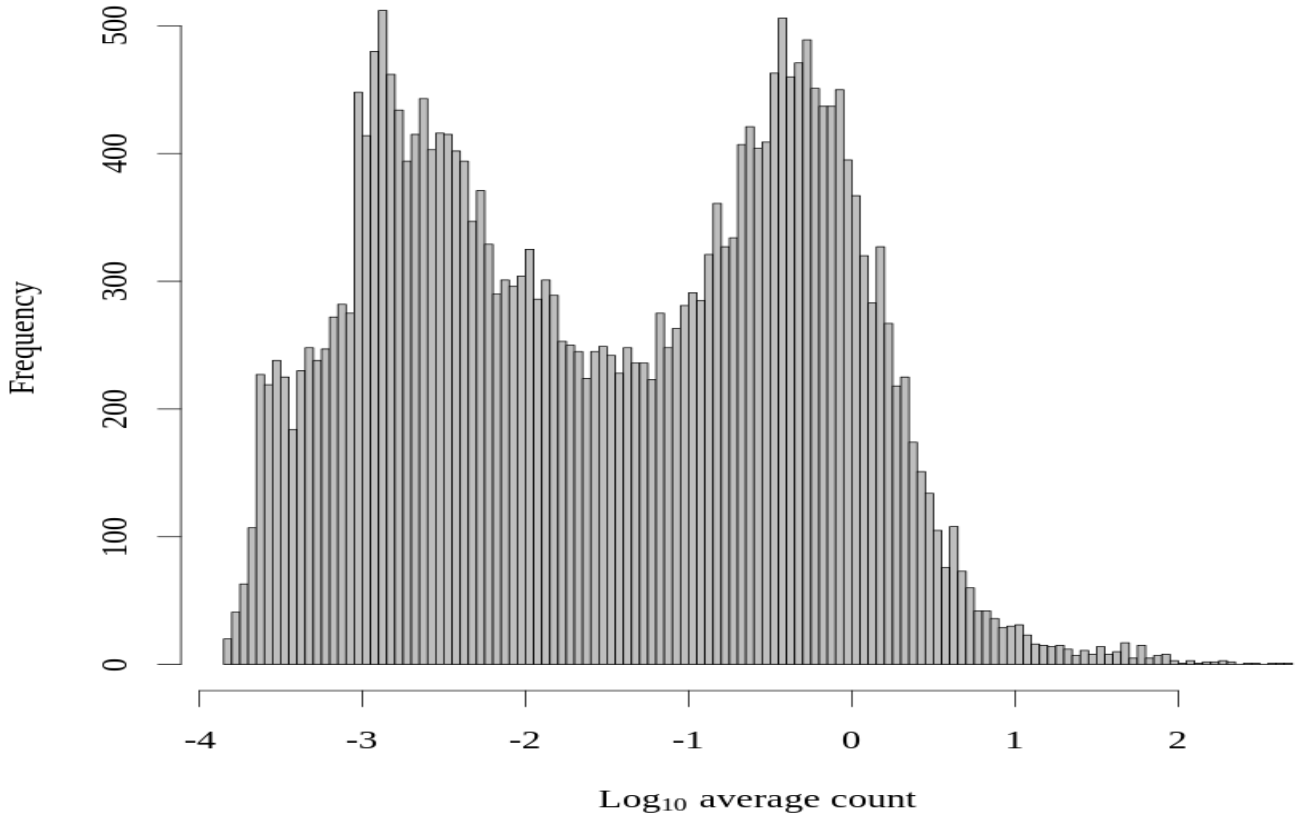

Figure 4.15 | Examining gene expression using the $\log_{10}$ average count and frequencies of expressed genes. The $\log_{10}$ average count and the frequencies of expressed genes in the epithelial cell dataset.

Genes that have an average count of zero were removed for further downstream analyses (Figure 4.15) (Kolodziejczyk et al., 2015; Luecken & Theis, 2019; Zhang et al., 2021).

Normalization

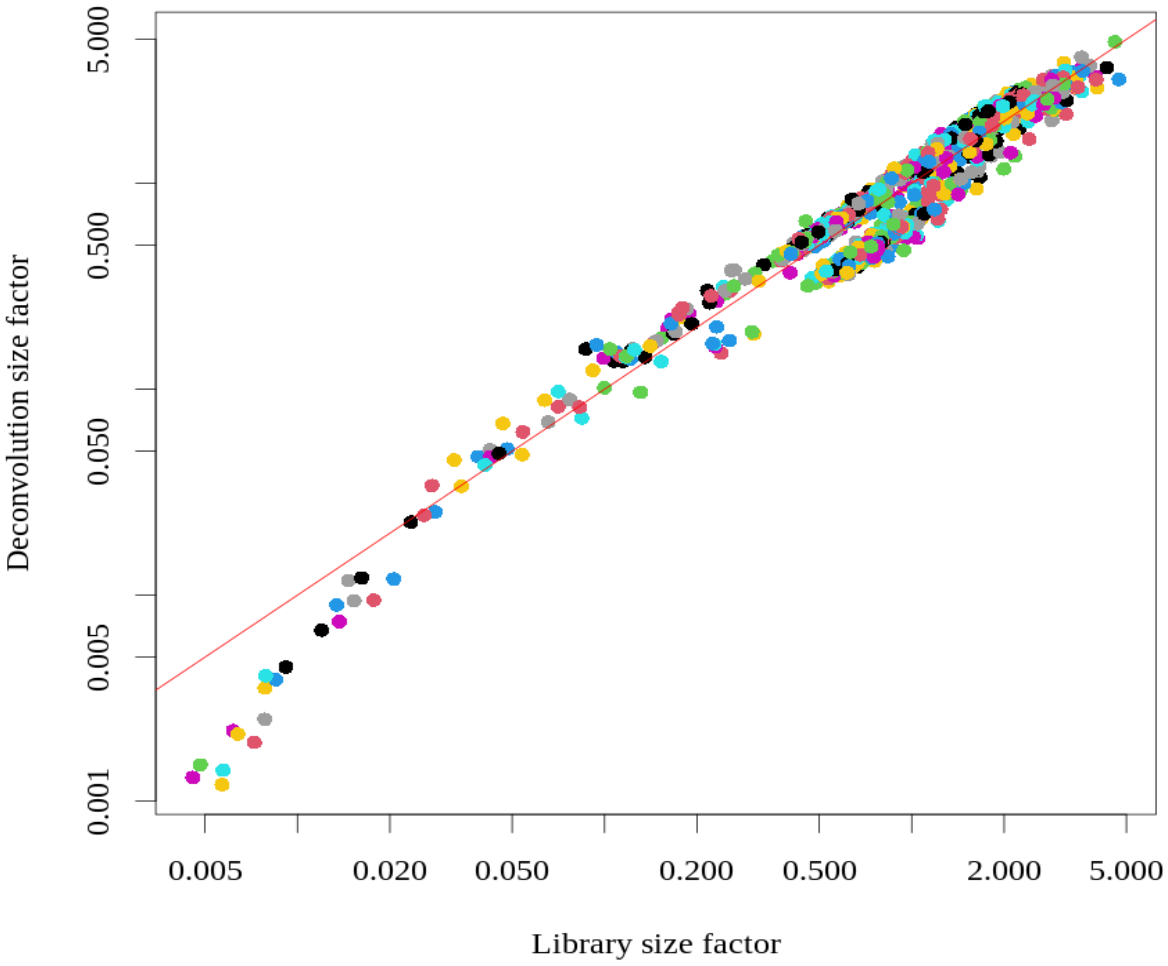

Figure 4.16 | Normalization by deconvolution. For each cell in the epithelial cell (GSM5604585) dataset, it's size factor is compared to the size factor derived from the library size. The identity between the two size factors is represented by the red line.

In scRNA-seq, systematic differences in sequencing coverage are often observed between libraries (Stegle et al., 2015). The small number of transcripts per cell increases the relative effect of technical differences in cDNA capture and PCR amplification efficiency. The aim of normalization is to retain biological differences, while removing differences in library size due to technical biases. The normalization by deconvolution approach was used. It is based on size factors (Lun et al., 2016). A size factor is an estimate of relative bias in a cell. The deconvolution

approach improves the estimates by pooling the counts of many cells to create larger counts and increase stability. Cell-based size factors are then derived by applying deconvolution to these pool-based size factors. The deconvolved cell-based factors are then used to normalize each cell's expression profile. The same process is performed for the library size as well (Figure 4.16).

Highly variable gene selection

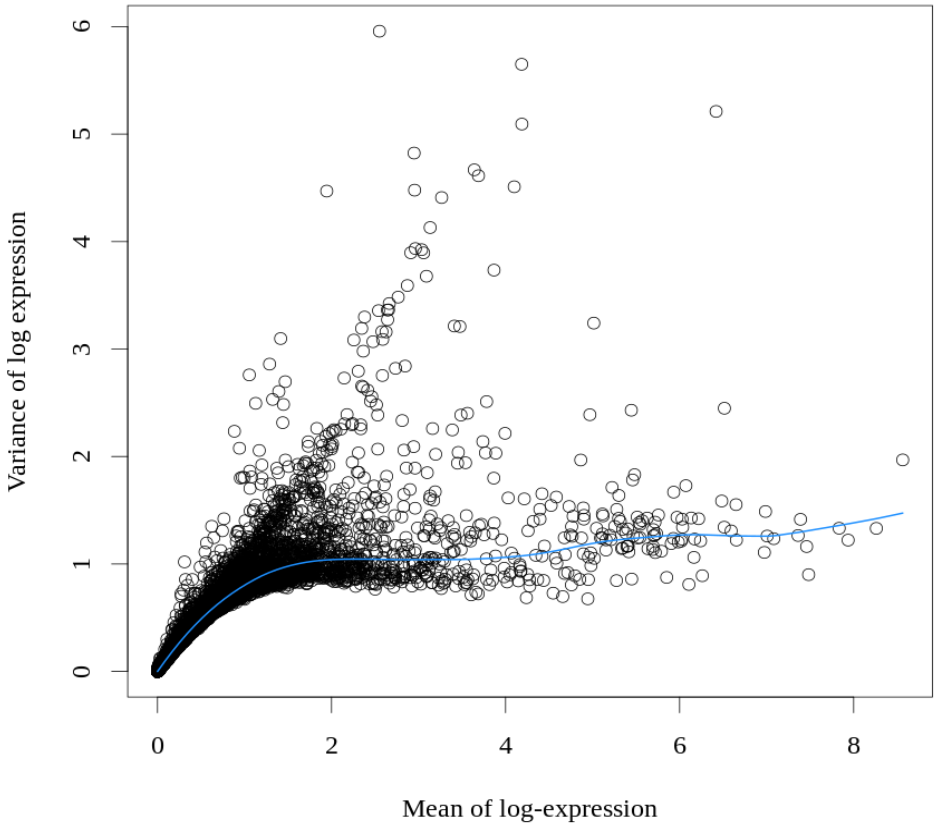

Figure 4.17 | Variance in the epithelial cell dataset (GSM5604585) plotted with respect to the mean. Each gene is indicated by a point. A locally weighted trend fitted to all genes is represented by the blue line.

To improve computational efficiency of downstream steps, it is important to select highly variable genes representing biological heterogeneity, while eliminating genes whose variability is due to random fluctation. Plotting the variance of genes with respect to the mean (both on the log scale) demonstrates the distribution and the overall trend, which is assumed to be due to technical and random processes (Figure 4.17). The fitted trend therefore aims to model the

uninteresting variation (the random component). The difference between the total variance and the random component is considered the biological component. The top 1000 genes with the largest biological component were selected for further downstream analyses. This is in accordance with current scRNA-seq best practices (Amezquita et al., 2020; Luecken & Theis, 2019).

Downstream analyses

Dimensionality reduction

Steps in downstream analyses allow the generation of insights from scRNA-seq data. Dimensionality reduction aims at reducing the number of separate dimensions in the data for effective visualization. Different dimensionality reduction techniques including principal component analysis (PCA), t-stochastic neighbour embedding, and uniform manifold approximation and projection (UMAP) were used.

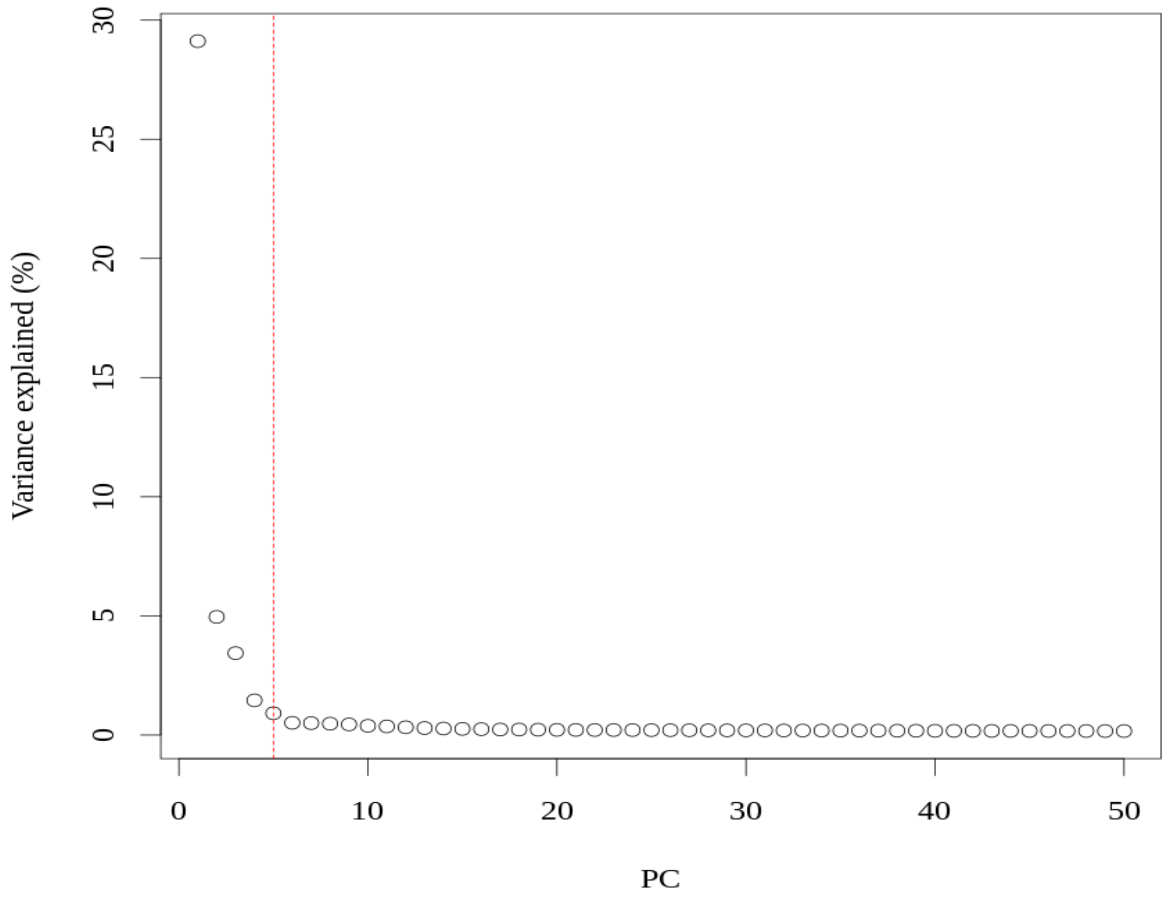

Figure 4.18 | Scree plot showing the percentage of variance explained by the successive principal components (PCs) in the dataset (GSM5604585). About 4 PCs account for majority of the variation in the dataset.

Figure 4.19 | Plot of the first two PCs in the dataset (GSM5604585). Each point is a cell, coloured according to the size factors used during normalization.

The top 2 PCs were plotted. Cells with high library sizes are being separated from those with low library sizes. Library size here is the total of counts across all genes for each cell. PCA is a linear technique, and it cannot efficiently pack differences of the chosen dimensions into the first 2 PCs (Figure 4.19).

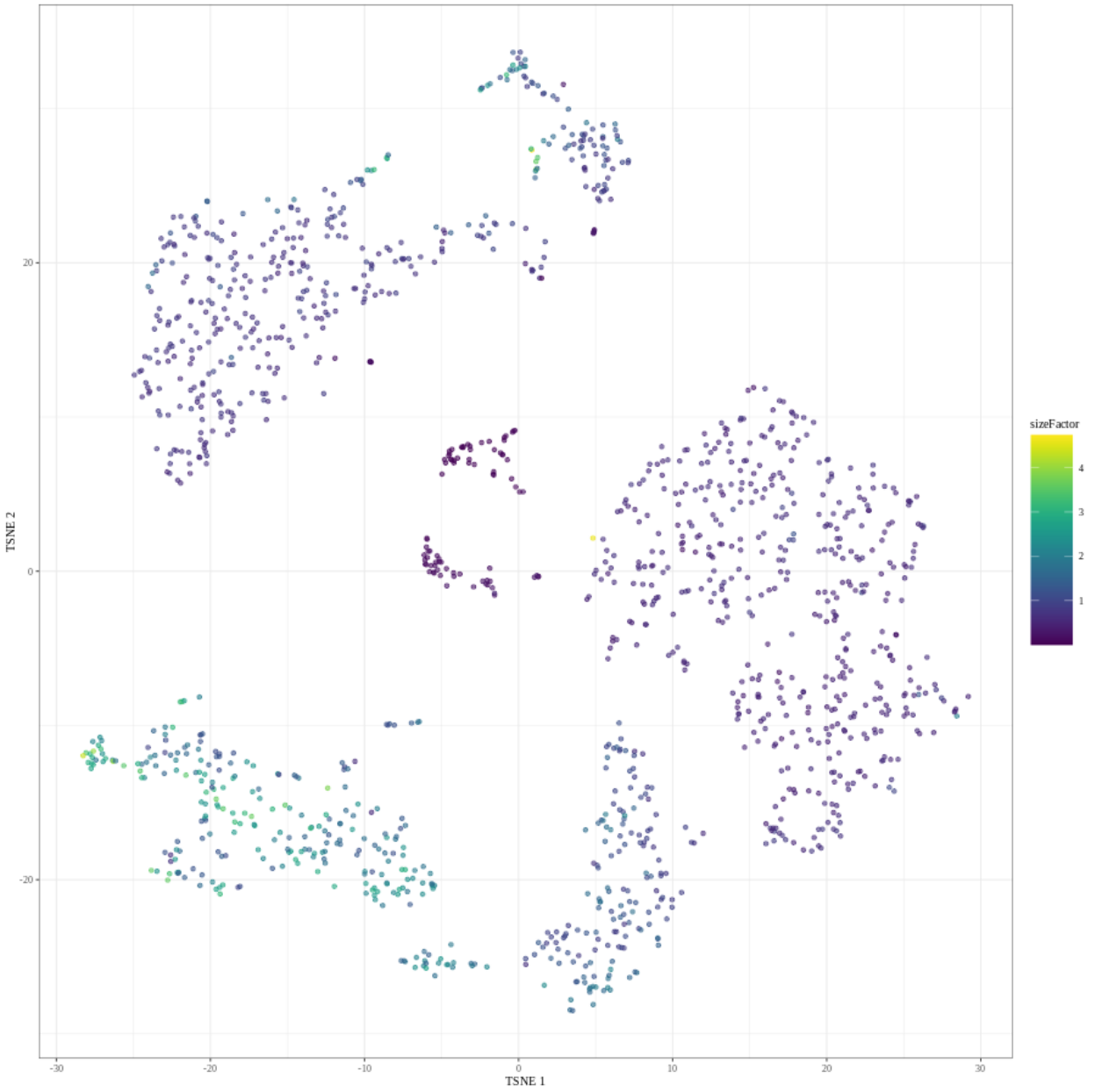

Figure 4.20 | Dimensionality reduction using t-SNE on the droplet-based dataset (GSM5604585) (t-distributed stochastic neighbour embedding) with a perplexity value of 30.

t-SNE is the main standard for visualization of scRNA-seq data (van der Maaten & Hinton, 2008). The t-SNE algorithm was performed using top PCs in the dataset (Figure 4.20). There was a more refined separation of cells into different clusters (Figure 4.20). This was done because, tSNE is very computationally intensive. t-SNE is a non-linear dimensionality reduction technique that attempts to find the low-dimensional representation of the data while preserving the distance between each point and its neighbours in high-dimensional space.

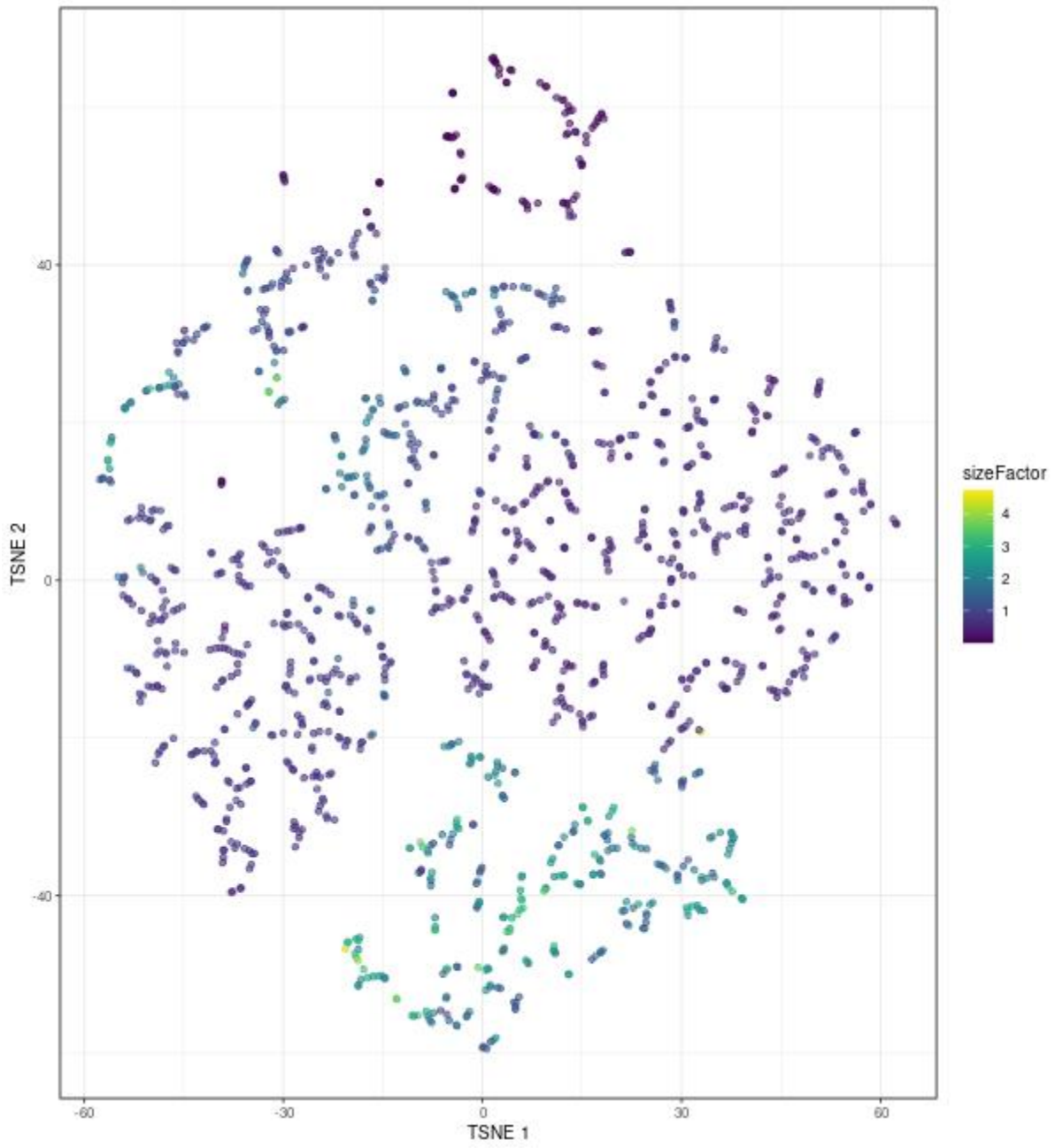

Figure 4.21 | Dimensionality reduction using tSNE with perplexity value of 5 (GSM5604585).

An important parameter that determines the granularity of visualization is *"perplexity"*. Here, the cell are clustered fairly distant from each other in high dimensional space (Figure 4.21) as compared to Figures 4.20 and 4.22.

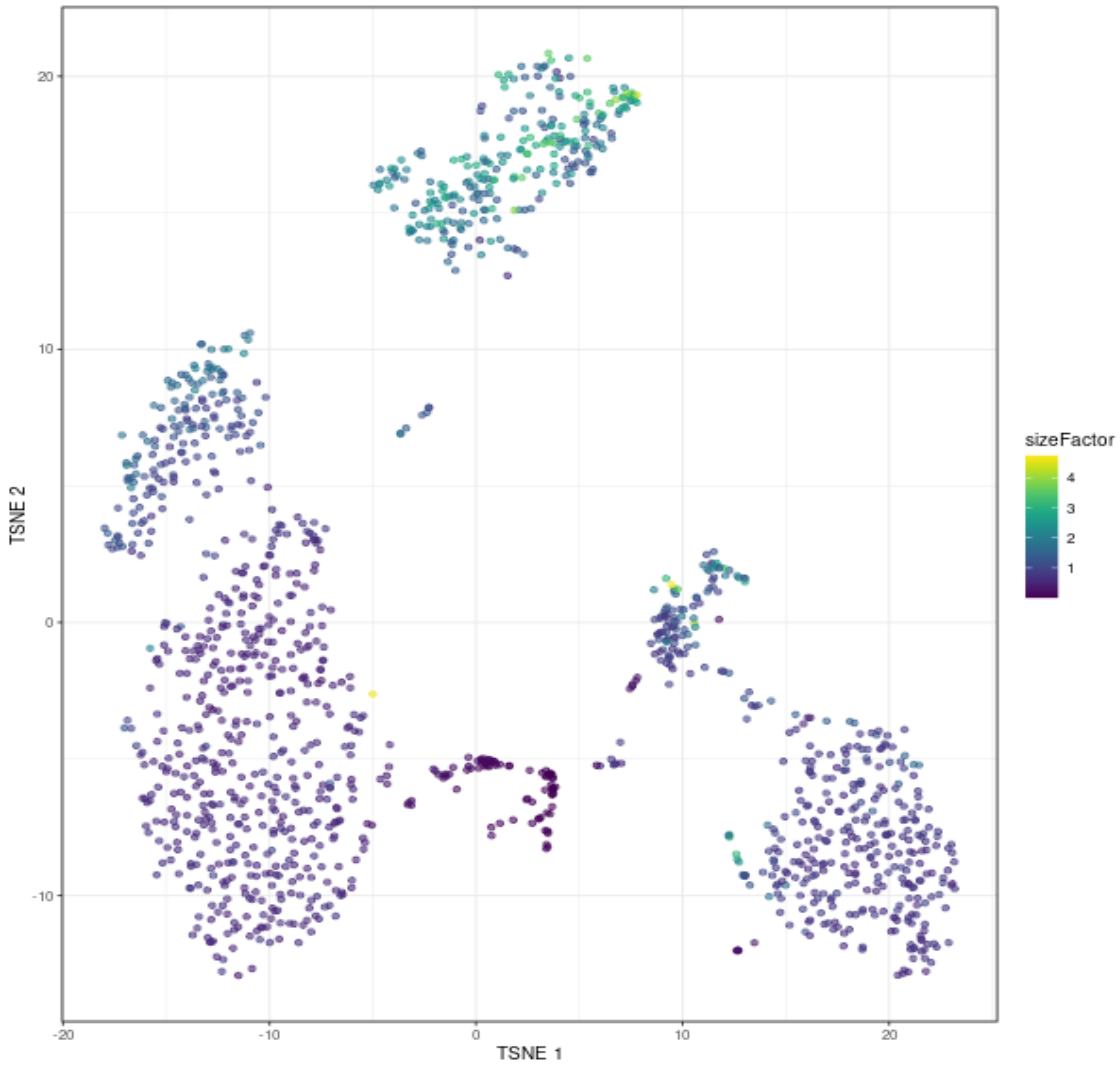

Figure 4.22 | Dimensionality reduction using tSNE with a perplexity value of 80 on the droplet-based dataset (GSM5604585).

Cells here (Figure 4.22) and compactly clustered as compared to Figures 4.20 and 4.21

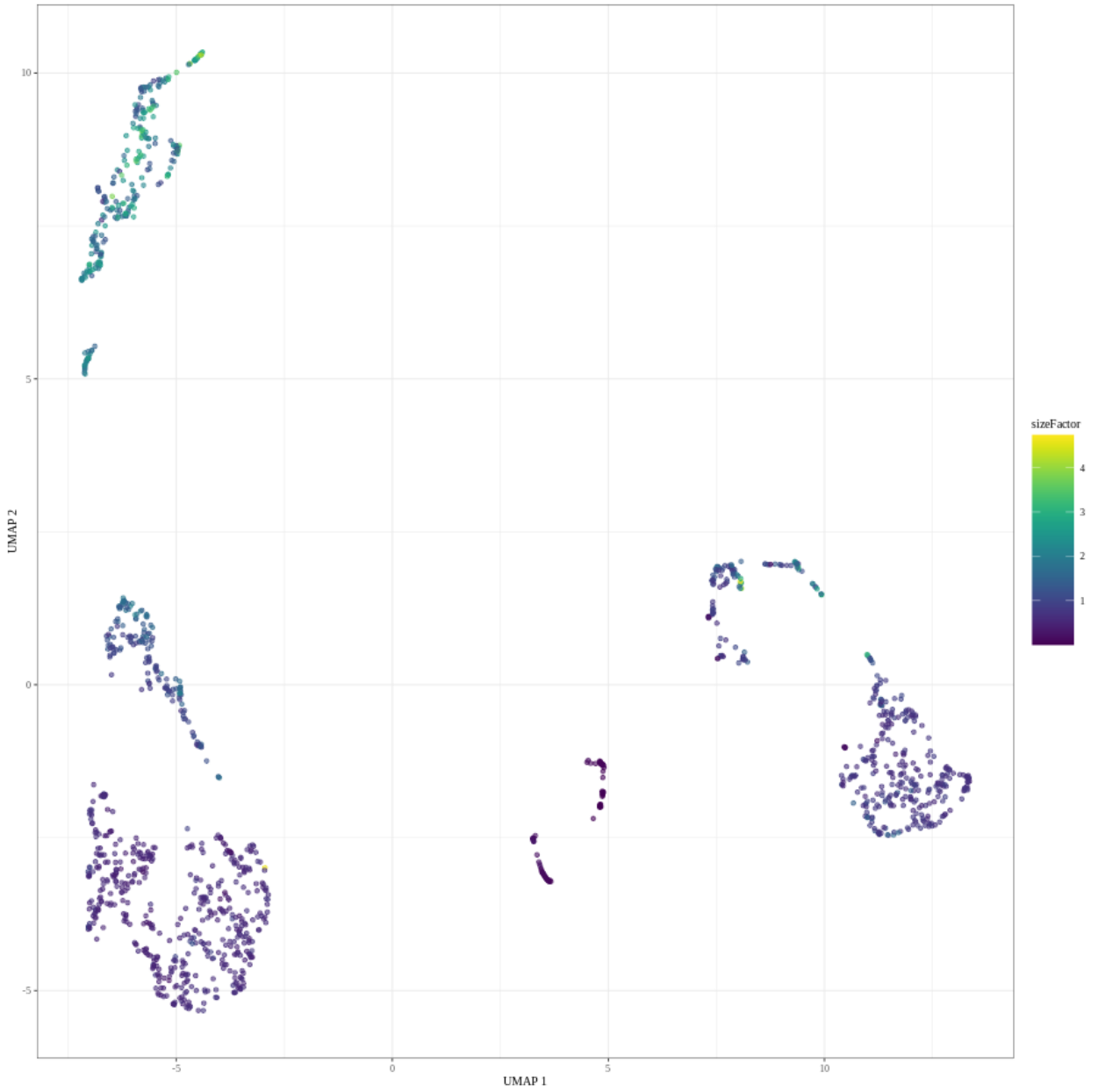

Figure 4.23 | Dimensionality reduction using UMAP (Uniform Manifold Approximation and Projection) on the droplet-based dataset (GSM5604585).

UMAP preserves global distances between points in high dimensional space and clusters of cells are further away from each other, but within the same group, they are close each other (Figure 4.23).

Cell clustering

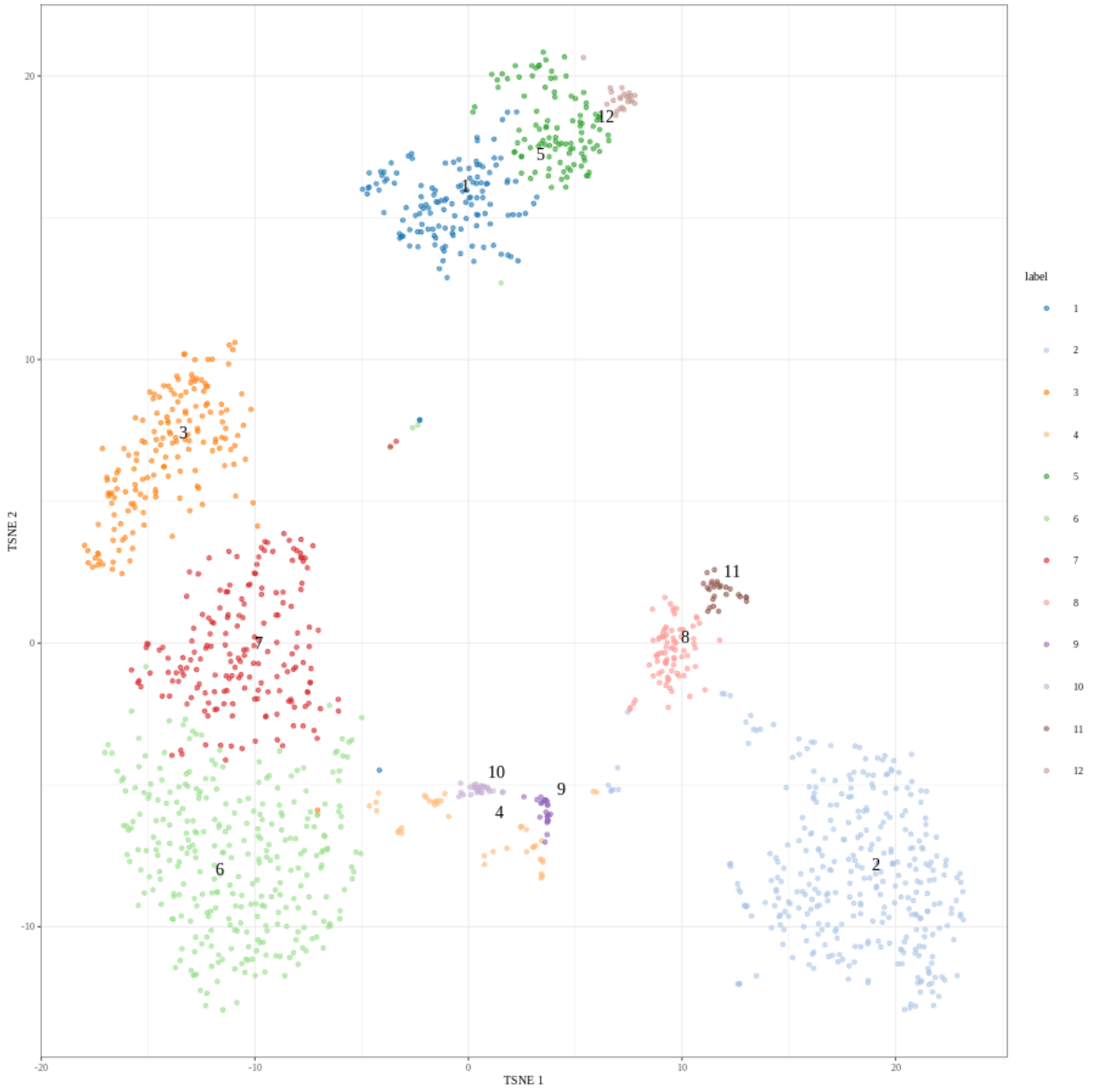

Figure 4.24 | Clustering of cells from t-SNE using k-NN (k-nearest neighbour) clustering on the droplet-based dataset (GSM5604585).

The clustering of cells empirically defines 12 groups of cells present in the data (Figure 4.24). Each point in the graph represents a cell. Graph-based clustering implementing the k-Nearest Neighbour (k-NN) search was used here. The *clusterCells* function from the scran package

constructs a nearest neighbour graph using the 10 nearest neighbours of each cell. Two cells that share a nearest neighbour are connected by an edge (Lun et al., 2016).

Trajectory inference

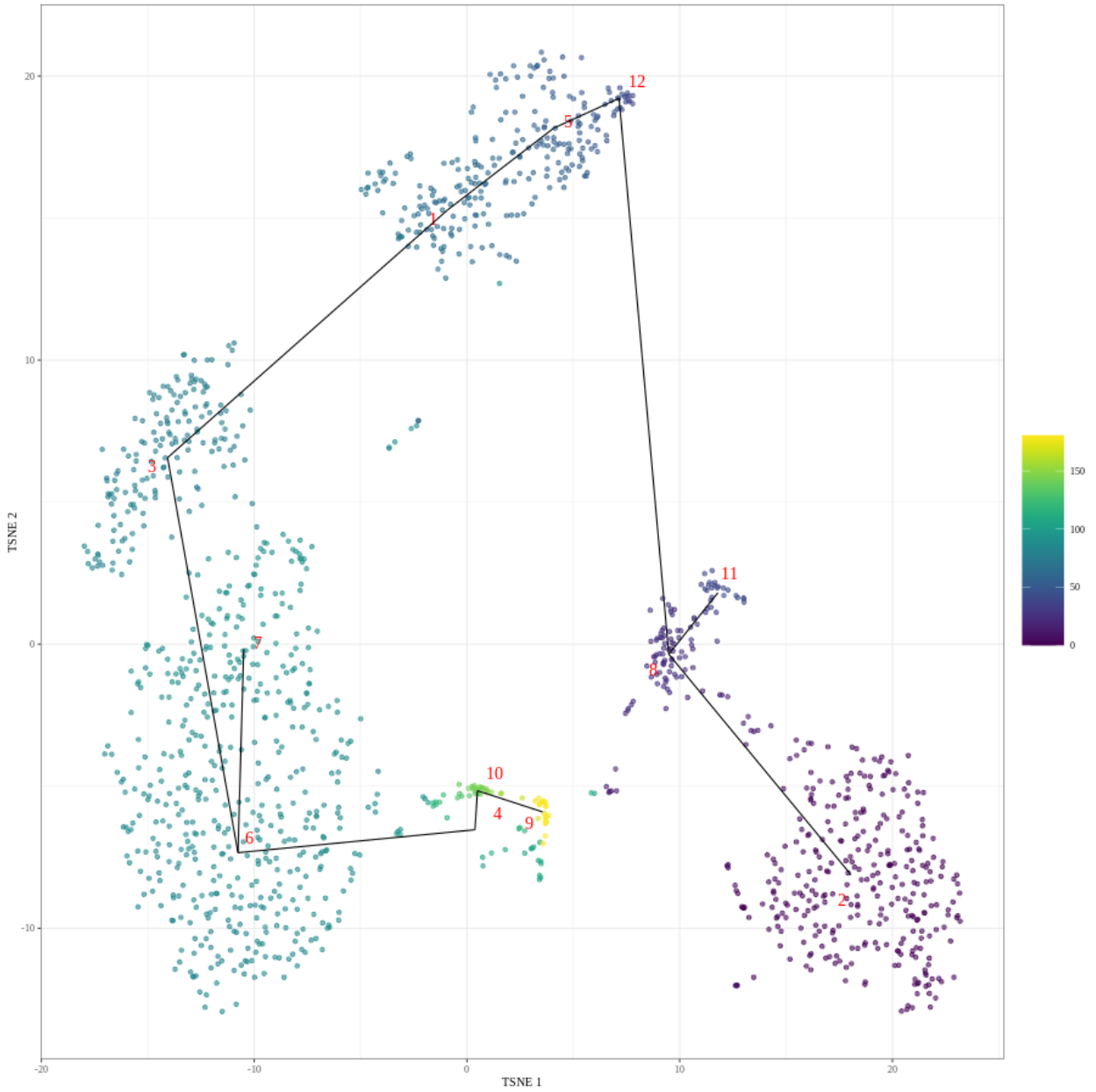

Figure 4.25 | t-SNE plot showing the trajectory inference on the droplet-based dataset (GSM5604585). Each cell is coloured according to its pseudotime value. The minimum spanning tree (MST) obtained using the TSCAN algorithm is overlaid on top.

Trajectory inference describes the differentiation of epithelial cells into different cell subtypes and the path along which they follow to reach their terminal states (Figure 4.25). The trajectory values were assigned using pseudotime values describing the relative position of a cell in the trajectory. Cells with smaller values are followed by their larger value counterparts. Branched trajectories were observed among clusters 6, 7, and 3 and clusters 8, 11, and 2. These branched trajectories may be subclusters of the epithelial cells.

Cell type annotation

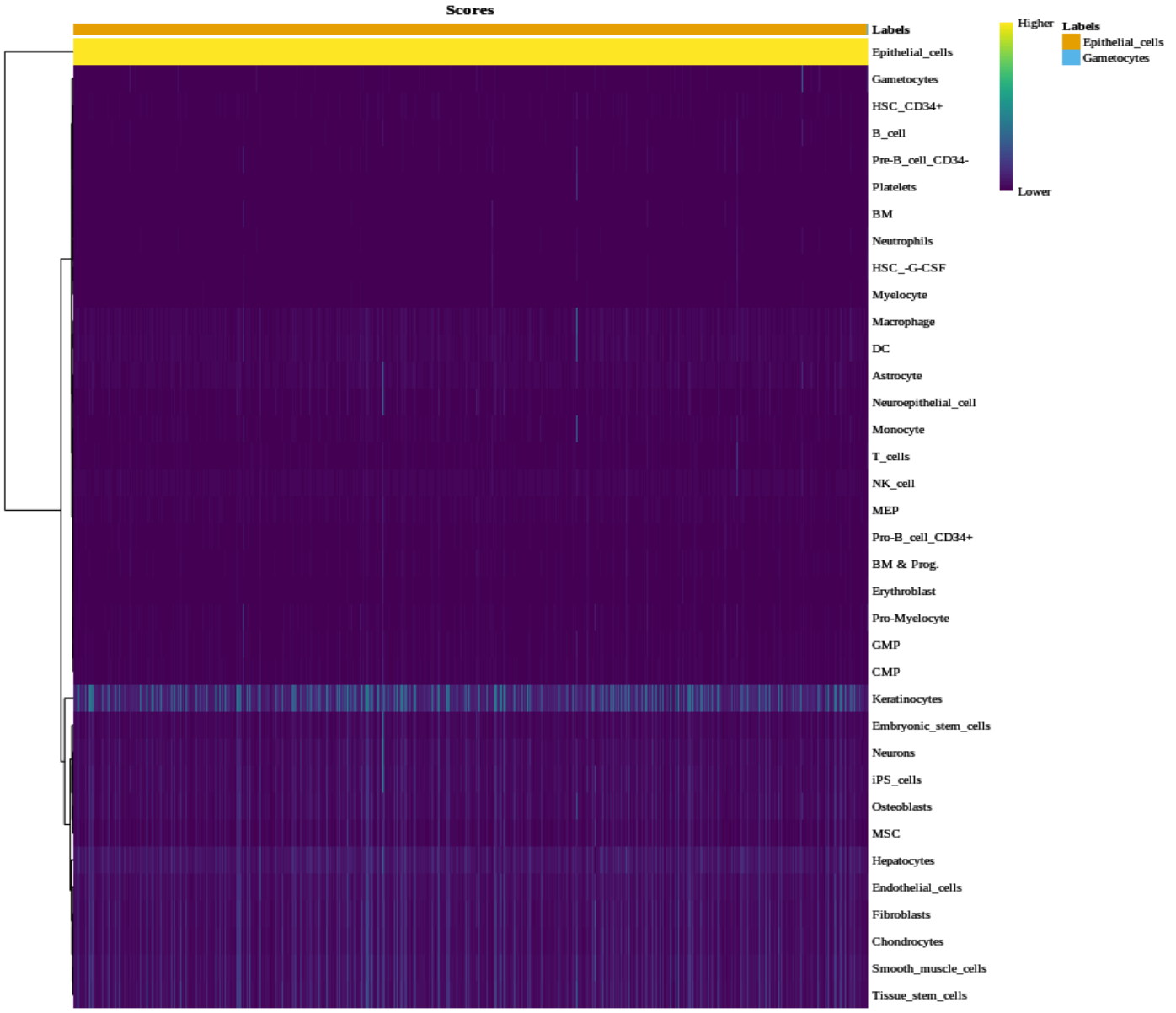

Figure 4.26 | Cell type annotation of the droplet-based dataset (GSM5604585) to existing references Heatmap of the assignment score for each cell (column) and label (row).

The expression profiles from the clustering were generated using existing reference datasets from the *celldex* project. The labels from the cell type annotation confirmed that all the cell types were epithelial cells (Figure 4.26)

Differential gene expression

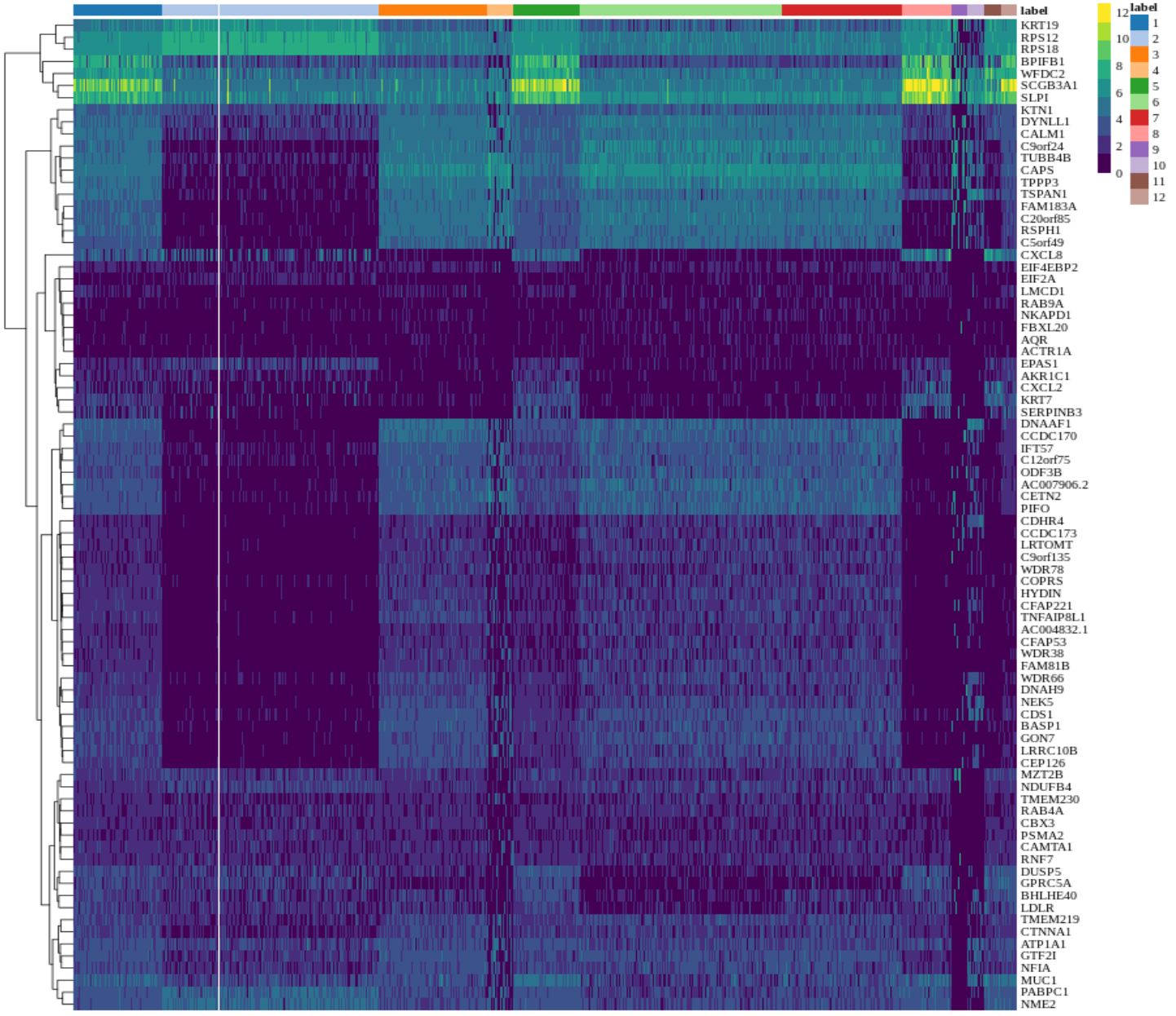

Figure 4.27 | Differential gene expression of the upregulated genes from cluster one on the droplet-based dataset (GSM5604585).

Differential expression analysis allows the interpretation of the clustering results and identification of genes that drive separation between clusters. The top five upregulated genes in cluster one are *KRT19*, *RPS12*, *RPS18*, *BFIFB1*, and *WFDC2* (Figure 4.27).

Output from pipeline with plate-based dataset

Pre-processing

The pancreas dataset (Segerstolpe et al., 2016) was generated from various human donors and subsequent QC was performed on the data to identify low-quality cells (Figure 4.28).

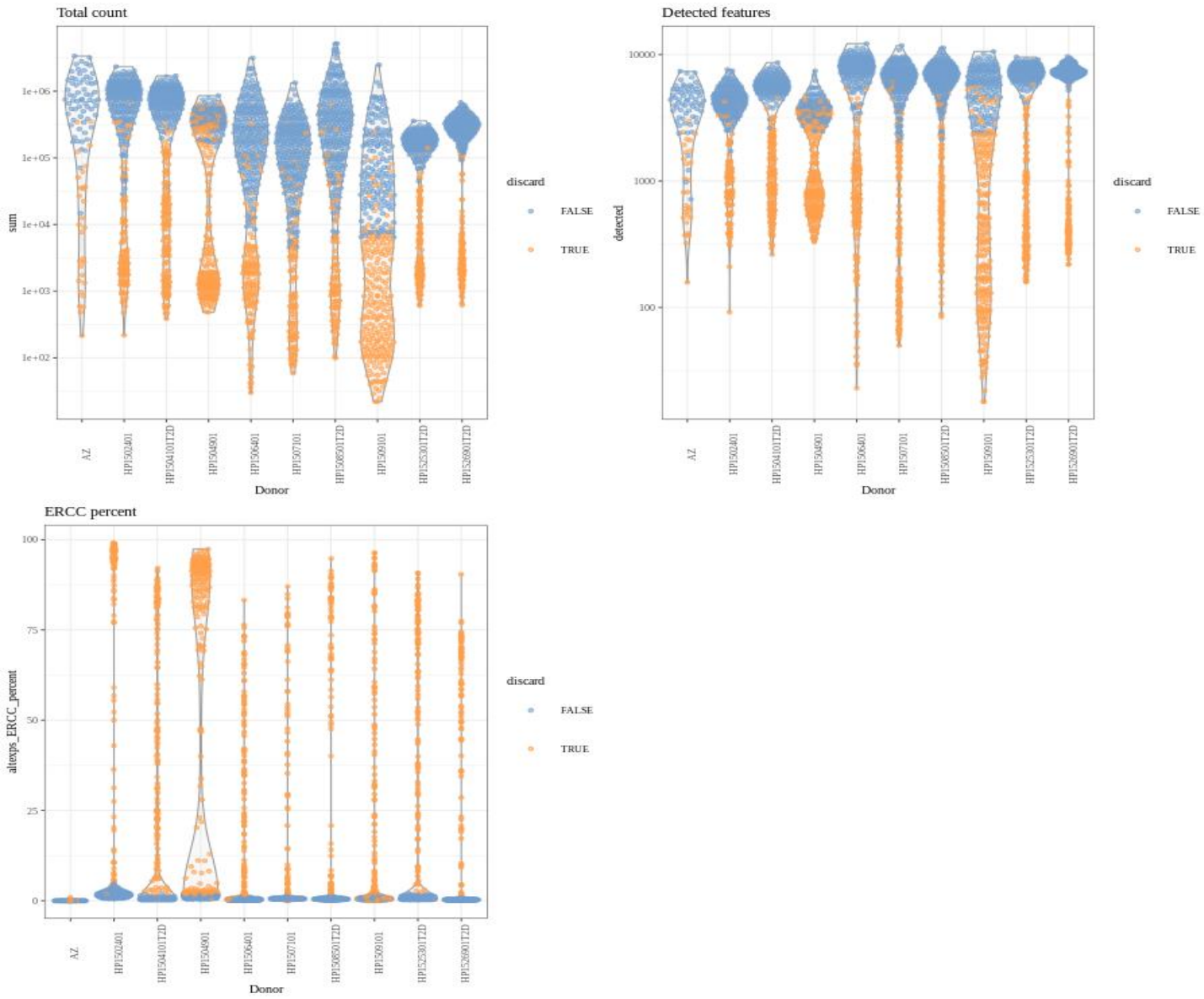

Figure 4.28 | Distribution of QC metrics across cells from each donor from the Segerstolpe *et al.*, 2016 pancreas dataset. Each point represents a cell and is coloured based on retention or removal. The orange colour represents cells of low quality and therefore that the cells were discarded, and the blue colour represents high quality cells for the various QC metrics.

The plot on the upper left (Figure 4.28) shows the sum of total counts across all cells for each donor in the study. Low quality cells marked as low quality by the authors of the study were discarded.

The upper right plot shows the number of detected features (expressed genes) in each cell for respective donors in the study (Figure 4.28).

The number of genes mapped to ERCC (External RNA Controls Consortium) spike-in transcripts is shown in the bottom-left (Figure 4.28). In all, the number of cells with low library size was 788; the number of cells with low number of expressed genes was 1056; and number of cells with high mitochondrial counts was 1031. Taking comparisons across batches into account, 1246 cells were discarded as being low-quality before subsequent downstream analyses.

Normalization

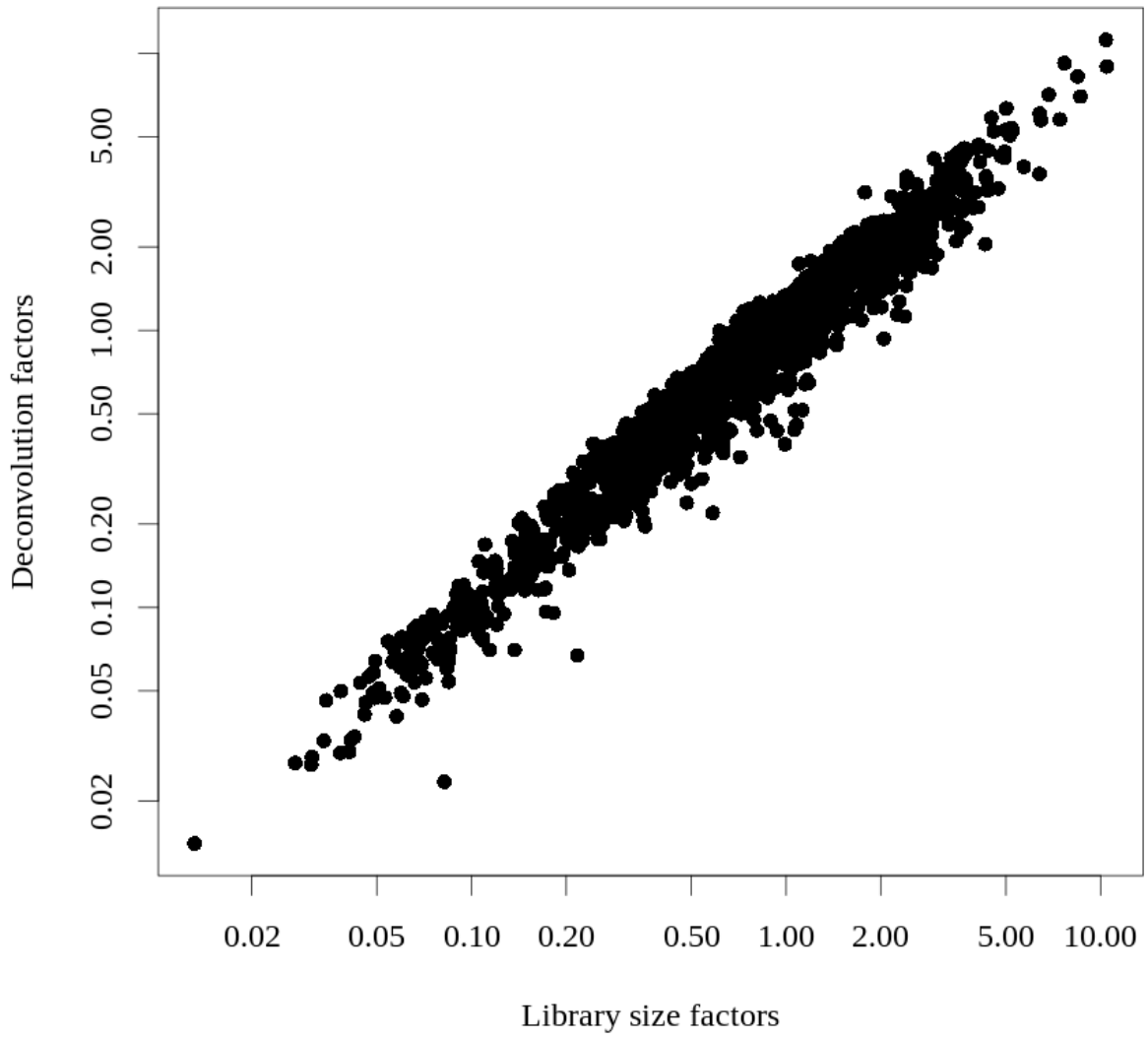

Figure 4.29 | Relationship between the deconvolution size factors as a function of the library size factors in the Segerstolpe *et al.,* 2016 dataset.

The normalization by deconvolution approach was applied on the dataset to pool counts across the various libraries for accurate size factor estimation (Figure 4.29). Since there are systematic differences in sequencing coverage observed across scRNA-seq libraries, this normalization

approach pools counts across cells for accurate estimation of their normalization profiles (size factors) for each cell. This comparison is done between the library sizes and the deconvolution size factors (Lun et al., 2016). Spike-ins were not normalized at this point because there were some cells with no spike-in counts.

Highly variable gene selection

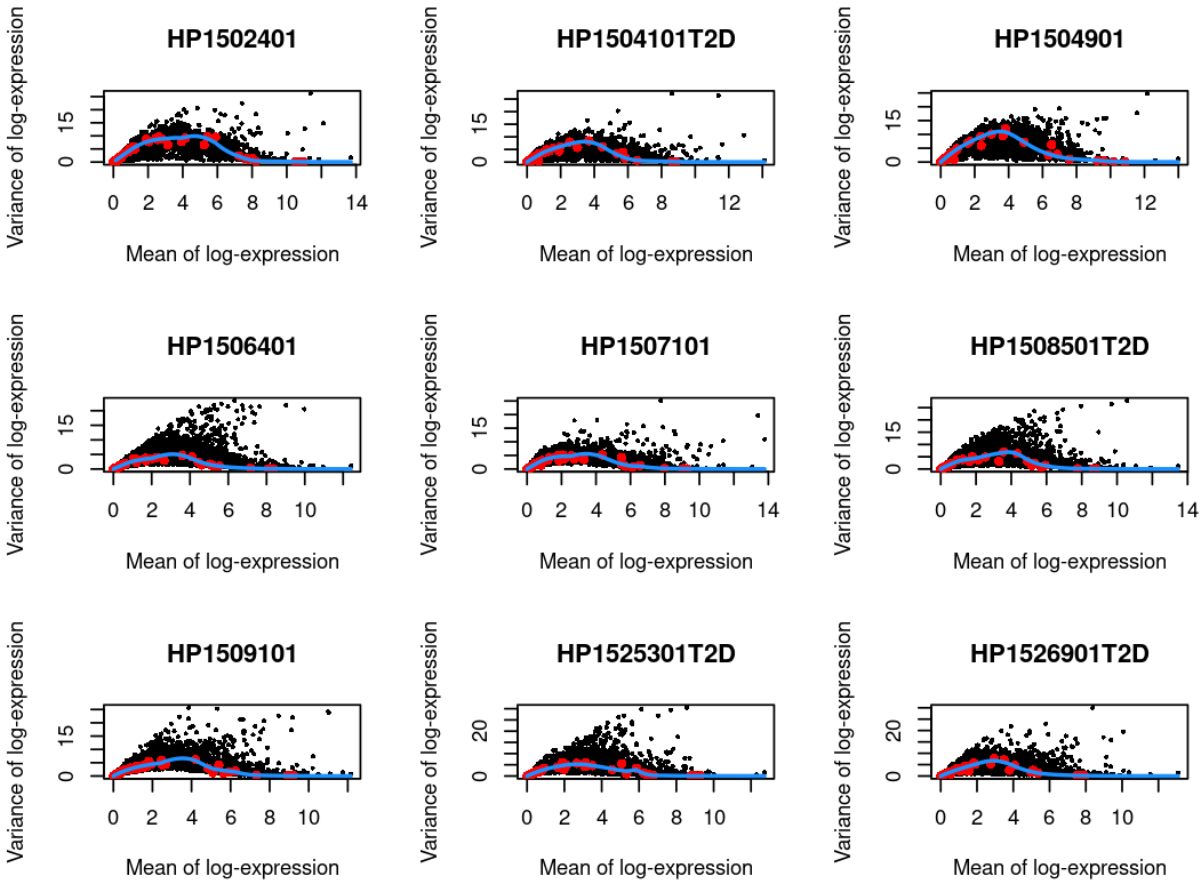

Figure 4.30 | Per-gene variance as a function of mean for the log expression values in the Segerstolpe *et al.*, 2016 dataset. Each point represents a gene (black) with the mean-variance trend (blue line) fitted to the spike-in transcripts (red) separately for each donor.

The per-gene variance for the log-expression values as a function of the mean is shown for each donor (Figure 4.30). Cells with no spike-ins were excluded from variance modelling. The mean-variance trend was fitted to the spike-in transcripts (red) for each donor.

Downstream analyses

Downstream results from the pancreas cell donors focused on showing the frequency of cell types from each donor and the different cluster of cells in the dataset.

Heatmap of cell frequency

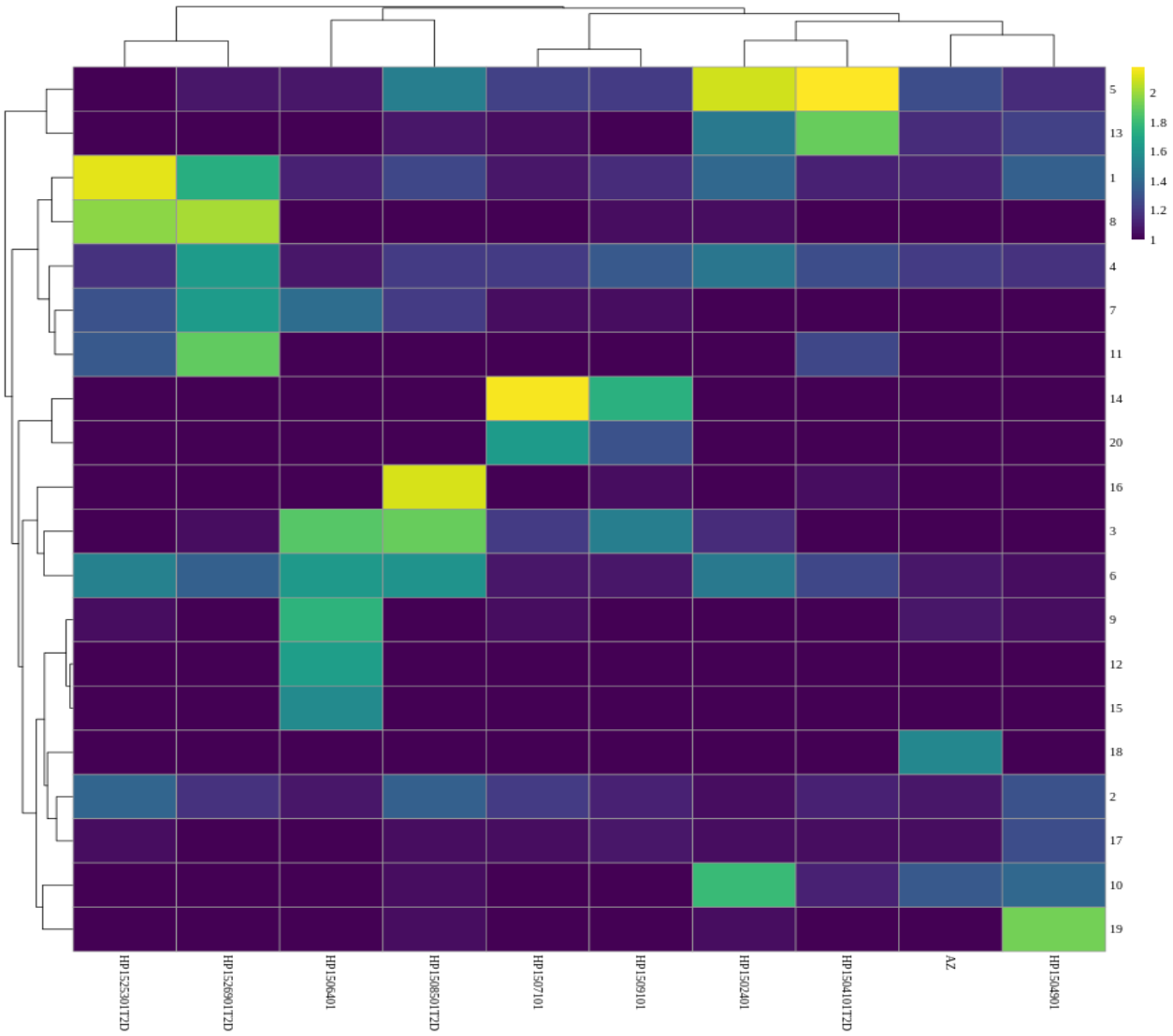

Figure 4.31 | Heatmap showing the frequency of cells from each donor in the Segerstolpe *et al.*, 2016 dataset. The spectrum represents the donor effect which is seen to be strong.

Clustering

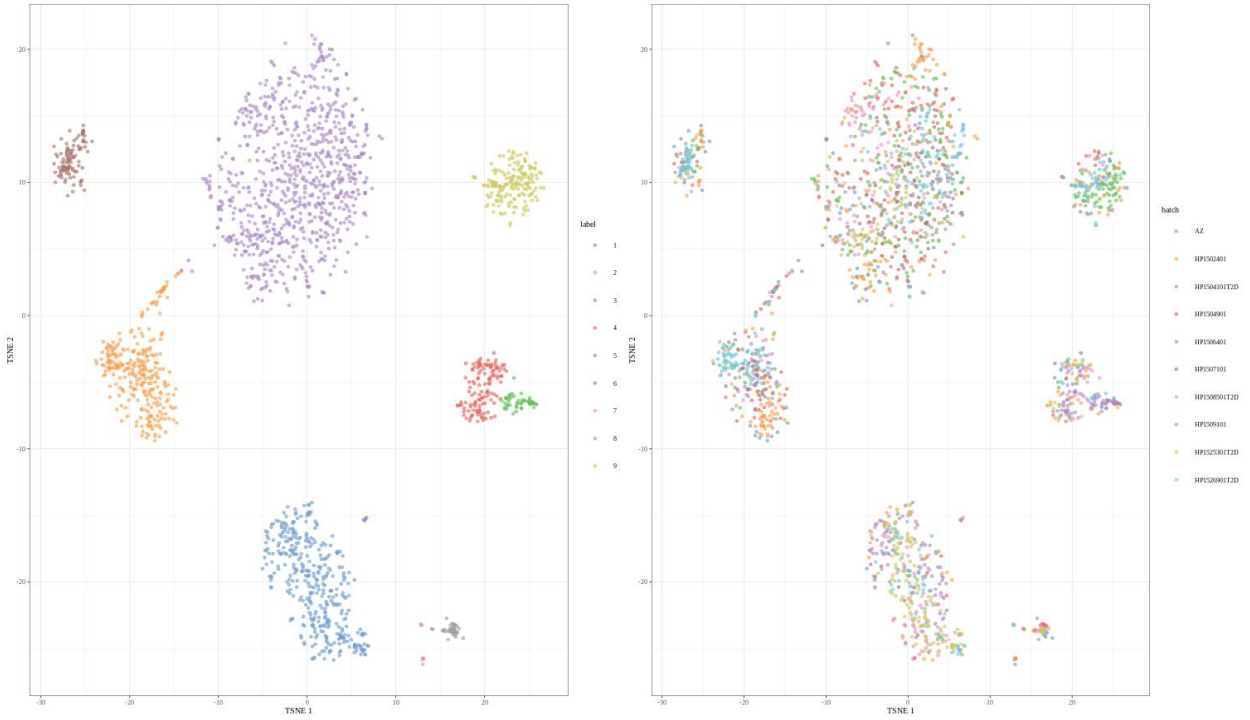

Figure 4.32 | t-SNE plot of the Segerstolpe *et al.*, 2016 dataset with batch corrections applied across donors. Each point represents a cell and is coloured by the assigned cluster identity.

The first 50 PCs were used for reducing the dimension of the data using the t-SNE algorithm (Figure 4.32). This choice was informed by the fact that there are rare pancreatic islet cell types (e.g., ε cells).

CHAPTER 5: DISCUSSION

scRNA-seq technology has revolutionized the outlook of gene expression due to its ability to quantify gene expression in individual cells. The data generated is complex and has biological and technical challenges, making obtaining useful insights from it very challenging. Along with massive expansion in scRNA-seq method development, it has become very important that pipelines be developed that can robustly analyse the data in a reproducible manner. This work implemented a highly robust scRNA-seq analysis pipeline, including both pre-processing and downstream methods for droplet and plate-based protocols.

Carefully selected high-throughput genomic tools were included to meet the needs of this pipeline to provide extensibility (handling different scRNA-seq datasets), robustness, and long-term maintenance.

The processes of this pipeline are automated with the WfMS — Nextflow. The publicly available nf-core pipeline is also implemented with Nextflow. In developing this pipeline, the main challenge encountered was ascertaining its dependencies were compatible with those available on HPC environments it will be run on. Once the version compatibility is ensured, the pipeline runs seamlessly on any system. Due to Nextflow's high portability and consistency in terms of generating results, it served as the best WfMS out of the ones available to be used for this work. In addition to Nextflow's fast prototyping, numerous tests could be run on small datasets to obtain a good idea of pipeline execution. The implemented pipeline correctly pre-processed scRNA-seq data, flagging low-quality reads and retaining only suitable portions of the data for further downstream analysis. This, however, does not come at the expense of compromising the data quality.

A key goal for developing this pipeline was reproducibility. To achieve this, each pipeline process was containerized with singularity (Kurtzer et al., 2017). This ensures that the results could be reproduced on different computing environments with the same software programs. The nf-core Nextflow pipeline also implements singularity, docker and conda environments. Having these range of options expands reproducibility and provides options for users to set up a correct environment, in case one fails to work on their system. However, setting up conda environments is a bit challenging as many users have encountered issues running it on an HPC system. The local pipeline is containerized using only singularity and has demonstrated reproducible and

consistent performance for all test runs carried out with it. A limitation of the local singularity containers was their size. Trimming the size down with multistage building was not completed due to large number of software programs that generate a long list of nested dependencies for which a list of files that need to be copied from a development environment to an execution container must be generated. Nevertheless, the size of the local singularity containers is comparable to that of the nf-core Nextflow pipeline.

Publicly available datasets from the Gene Expression Omnibus (GEO) and ArrayExpress of the European Bioinformatics Institute (EBI) were used to test the pipeline. To make the pipeline work for both droplet and plate-based datasets, extensive testing of the parameter space for each of the respective processes had to be performed. This was time-consuming and affected some downstream analysis methods like trajectory inference for plate-based dataset. The nf-core pipeline focused on pre-processing droplet-based data only to reduce complexity and focus solely on generating a count matrix.

The droplet-based dataset which consisted of human airway epithelial cells (GSM5604585) was processed through the pipeline. To correctly analyse scRNA-seq data, a critical factor is accounting for and the removal of low-quality reads/cells. QC steps implemented in the pipeline identified low quality reads (based on per base sequence quality and per base sequence content). More than 80% of other parameters including per base sequence quality, per sequence quality scores, per sequence GC content, and other basic statistics, etc., all passed QC.

Subsequent steps involving the removal of unique molecular identifier (UMI) sequences were implemented to remove duplicate reads that occurred during PCR and cell barcodes were used to assign cells to their sample of origin during sequencing. Trimming of reads removed adapters that were added during sequencing. It is worth mentioning that varying the parameters in the pre-processing step in the pipeline within the acceptable limits, had no effect on subsequent downstream stages. Parameters that were out of their acceptable limits (for example, values not recommended for trimming off adapters, not recommended values) resulted in reads that could not pass post-QC, and therefore, could not be used for further downstream analyses.

For plate-based datasets, different parameters were adjusted for pre-processing for obtaining the recommended results. Varying the parameters here had no influence on downstream steps such

as clustering. This is important because with the default parameters used in setting up the pipeline, it could handle the plate-based data fed into it.

In downstream analysis, the pipeline correctly identified and confirmed the cell types in the dataset — epithelial cells for the droplet-based dataset, and pancreatic cells for the plate-based dataset. For the epithelial cell types, cell subtypes were identified based on the marker genes that were identified from the GeneCards database (https://www.genecards.org/). Most of these genes played a role in innate immune response in the lungs (*BPIFB1*) and are responsible for structural integrity of epithelial cells (*KRT19*). In the plate-based dataset, the pipeline identified endocrine cell types from the pancreas (α, β), and even rare cell types such as Ɛ cells from the donors.

The *local pipeline* developed for analysing scRNA-seq implements 10 modules namely: Quality control (*FASTQC*), UMI extraction (*UMI-tools*), Trimming (*Trimmomatic)*, post-Quality control (*MultiQC)*, Alignment (Indexing and quantification with *STARSolo and Salmon)*. The gene expression matrix obtained from either full-length alignment (with *STAR*), or pseudo-alignment (with *Alevin*), is used as the input for further downstream analyses.

The previously published *nf-core* scRNA-seq pipeline implements pre-processing modules only for analysing scRNA-seq data (Ewels et al., 2020), but did not implement downstream analysis. The modules in the *nf-core* pipeline are QC (*FASTQC* and *MultiQC)*, and Alignment (*Kallisto, STARSolo, Alevin, CellRanger*). The local pipeline was developed with extensibility in mind, as Nextflow supports it, to allow the incorporation of other pseudo-aligners like Kallisto. These implementations would be useful to one who wants a quick run for a preliminary analysis of their data, because using full-length aligners like STAR take a long time.

CHAPTER 6: CONCLUSIONS

Since 2009, scRNA-seq technology has provided a fresh perspective allowing the investigation of single cells. Daunting challenges faced with the analysis of the complex data generated sparked the development of pipelines to gain insights from the data. However, all existing scRNA-seq pipelines all focus on one aspect of scRNA-seq data, either pre-processing or downstream applications, but not both. The pipeline described here was implemented in a WfMS (Nextflow) and the R programming language. It carries out steps from pre-processing of scRNA-seq data from raw fastq files to cell type identification.

This pipeline incorporated current best practices adapted by the single-cell community, of which a key one is reproducibility. Each processing step in the pipeline can be reproduced on different computing platforms with the same datasets and obtain the same results. This work serves as an improvement over existing scRNA-seq pipelines and will aid researchers to reproducibly identify gene expression patterns and cell types.

Limitations of the Study & Recommendations for Future Research

This scRNA-seq pipeline has some limitations including the following:

1. The manual annotation used during cell type annotation to identify cell types is a bottleneck in the workflow as it takes time to manually annotate these cells to a reference database. This also affects current scRNA-seq pipelines.
2. Containerized environments built for the pipeline are quite heavy in size and require large computational resources.

Recommendations for future research

1. For future scRNA-seq pipelines, it is recommended that multiple datasets from both droplet and plate -based scRNA-seq methods, be tested and benchmarked. This should also include minimal datasets for each protocol type. Also, containerized environments should be trimmed, e.g., using multi-stage building, to decrease their size. This would improve ease of use on different computing environments.

2. Future update of the current pipeline version should work on automating the downstream analyses by making use of prespecified parameters instead of interactive decisions.

3. It is encouraged that future scRNA-seq pipelines incorporate a graphical user interface (GUI) to make it more user-friendly for the less experienced bioinformatics software users.

4. It is necessary to have the pipeline read configuration parameters from files, and these parameters should include the location of input data files. This will increase the flexibility of the pipeline by placing project or analysis specific variable information outside of the pipeline itself. This will also increase robustness of the pipeline.

REFERENCES

Adams, M. D., Kelley, J. M., Gocayne, J. D., Dubnick, M., Polymeropoulos, M. H., Xiao, H., Merril, C. R., Wu, A., Olde, B., Moreno, R. F., Kerlavage, A. R., McCombie, W. R., & Venter, J. C. (1991). Complementary DNA Sequencing: Expressed Sequence Tags and HumanGenomeProject. *Science*, *252*(5013), 1651–1656. https://doi.org/10.1126/-science.2047873

AlJanahi, A. A., Danielsen, M., & Dunbar, C. E. (2018). An Introduction to the Analysis of Single-Cell RNA-Sequencing Data. *Molecular Therapy – Methods & Clinical Development*, *10*, 189–196. https://doi.org/10.1016/j.omtm.2018.07.003

Amezquita, R. A., Lun, A. T. L., Becht, E., Carey, V. J., Carpp, L. N., Geistlinger, L., Marini, F., Rue-Albrecht, K., Risso, D., Soneson, C., Waldron, L., Pagès, H., Smith, M. L., Huber, W., Morgan, M., Gottardo, R., & Hicks, S. C. (2020). Orchestrating single-cell analysis with Bioconductor. *Nature Methods*, *17*(2), 137–145. https://doi.org/10.1038/s41592-019-0654-x

Anderson, D. (2009). *"Animalcules". Lens on Leeuwenhoek*. Retrieved September 23, 2022, from https://lensonleeuwenhoek.net/content/animalcules

Andrews, S. (2022). *Babraham Bioinformatics—FastQC A Quality Control tool for High Throughput Sequence Data*. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Arendt, D., Musser, J. M., Baker, C. V. H., Bergman, A., Cepko, C., Erwin, D. H., Pavlicev, M., Schlosser, G., Widder, S., Laubichler, M. D., & Wagner, G. P. (2016). The origin and evolution of cell types. *Nature Reviews Genetics*, *17*(12), 744–757. https://doi.org/-10.1038/nrg.2016.127

Bacher, R., & Kendziorski, C. (2016). Design and computational analysis of single-cell RNA-sequencing experiments. *Genome Biology*, *17*(1), 63. https://doi.org/10.1186/s13059-016-0927-y

Baker, H., & van Leeuwenhoek, A. (1740). IX. An account of Mr. Leeuwenhoek's microscopes. *Philosophical Transactions of the Royal Society of London*, *41*(458), 503–519. https://doi.org/10.1098/rstl.1739.0085

Baßler, K., Günther, P., Schulte-Schrepping, J., Becker, M., & Biernat, P. (2019). A Bioinformatic Toolkit for Single-Cell mRNA Analysis. In V. Proserpio (Ed.), *Single Cell*

Methods (Vol. 1979, pp. 433–455). Springer New York. https://doi.org/10.1007/978-1-4939-9240-9_26

Bolger, A. M., Lohse, M., & Usadel, B. (2014). *Trimmomatic: A flexible trimmer for Illumina sequence data*. Bioinformatics, *30*(15), 2114–2120. https://doi.org/10.1093/-bioinformatics/btu170

Brady, G., Barbara, M., & Iscove, N. N. (1990). *Representative in Vitro cDNA Amplification from Individual Hemopoietic Cells and Colonies*. *Methods Mol. Cell. Biol. 2*(1), 17–25.

Bray, N. L., Pimentel, H., Melsted, P., & Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. *Nature Biotechnology*, *34*(5), 525–527. https://doi.org/10.1038/nbt.3519

Brennecke, P., Anders, S., Kim, J. K., Kołodziejczyk, A. A., Zhang, X., Proserpio, V., Baying, B., Benes, V., Teichmann, S. A., Marioni, J. C., & Heisler, M. G. (2013). Accounting for technical noise in single-cell RNA-seq experiments. *Nature Methods*, *10*(11), 1093–1095. https://doi.org/10.1038/nmeth.2645

Buettner, F., Natarajan, K. N., Casale, F. P., Proserpio, V., Scialdone, A., Theis, F. J., Teichmann, S. A., Marioni, J. C., & Stegle, O. (2015). Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature Biotechnology*, *33*(2), 155–160. https://doi.org/10.1038/nbt.3102

Common Workflow Language User Guide. Retrieved November 5, 2022, from https://www.commonwl.org/user_guide/#

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, *35*(4), 316–319. https://doi.org/10.1038/nbt.3820

Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., & Gingeras, T. R. (2013). STAR: Ultrafast universal RNA-seq aligner. *Bioinformatics*, *29*(1), 15–21. https://doi.org/10.1093/bioinformatics/bts635

Dong, Z., & Chen, Y. (2013). Transcriptomics: Advances and approaches. *Science China Life Sciences*, *56*(10), 960–967. https://doi.org/10.1007/s11427-013-4557-2

Du, Y., Huang, Q., Arisdakessian, C., & Garmire, L. X. (2020). Evaluation of STAR and Kallisto on Single Cell RNA-Seq Data Alignment. *G3 Genes|Genomes|Genetics*, *10*(5), 1775–1783. https://doi.org/10.1534/g3.120.401160

Eberwine, J., Yeh, H., Miyashiro, K., Cao, Y., Nair, S., Finnell, R., Zettel, M., & Coleman, P. (1992). Analysis of gene expression in single live neurons. *Proceedings of the National Academy of Sciences*, *89*(7), 3010–3014. https://doi.org/10.1073/pnas.89.7.3010

Ewels, P. A., Peltzer, A., Fillinger, S., Patel, H., Alneberg, J., Wilm, A., Garcia, M. U., Di Tommaso, P., & Nahnsen, S. (2020). The nf-core framework for community-curated bioinformatics pipelines. *Nature Biotechnology*, *38*(3), 271–271. https://doi.org/10.1038/-s41587-020-0435-1

Ewis, A. A., Zhelev, Z., Bakalova, R., Fukuoka, S., Shinohara, Y., Ishikawa, M., & Baba, Y. (2005). A history of microarrays in biomedicine. *Expert Review of Molecular Diagnostics*, *5*(3), 315–328. https://doi.org/10.1586/14737159.5.3.315

Femino, A. M., Fay, F. S., Fogarty, K., & Singer, R. H. (1998). Visualization of Single RNA Transcripts in Situ. *Science*, *280*(5363), 585–590. https://doi.org/10.1126/-science.280.5363.585

Fjukstad, B., & Bongo, L. A. (2017). A Review of Scalable Bioinformatics Pipelines. *Data Science and Engineering*, *2*(3), 245–251. https://doi.org/10.1007/s41019-017-0047-z

Gest, H. (2004). The discovery of microorganisms by Robert Hooke and Antoni van Leeuwenhoek, Fellows of The Royal Society. *Notes and Records of the Royal Society of London*, *58*(2), 187–201. https://doi.org/10.1098/rsnr.2004.0055

Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., Peters, K., & Schober, D. (2020). FAIR Computational Workflows. *Data Intelligence*, *2*(1–2), 108–121. https://doi.org/10.1162/dint_a_00033

Grün, D., & van Oudenaarden, A. (2015). Design and Analysis of Single-Cell Sequencing Experiments. *Cell*, *163*(4), 799–810. https://doi.org/10.1016/j.cell.2015.10.039

Haque, A., Engel, J., Teichmann, S. A., & Lönnberg, T. (2017). A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Medicine*, *9*(1), 75. https://doi.org/10.1186/s13073-017-0467-4

Hedlund, E., & Deng, Q. (2018). Single-cell RNA sequencing: Technical advancements and biological applications. *Molecular Aspects of Medicine*, *59*, 36–46. https://doi.org/10.1016/j.mam.2017.07.003

Hie, B., Peters, J., Nyquist, S. K., Shalek, A. K., Berger, B., & Bryson, B. D. (2020). Computational Methods for Single-Cell RNA Sequencing. *Annual Review of Biomedical Data Science*, *3*(1), 339–364. https://doi.org/10.1146/annurev-biodatasci-012220-100601

Hooke, R. (1665). *Micrographia: or some physiological descriptions of minute bodies made by magnifying glasses with observations and inquiries thereupon ...* Printed by Jo. Martyn and Ja. Allestry, printers to the Royal Society. https://doi.org/10.5962/bhl.title.904

Huang, X., Liu, S., Wu, L., Jiang, M., & Hou, Y. (2018). High Throughput Single Cell RNA Sequencing, Bioinformatics Analysis and Applications. In J. Gu & X. Wang (Eds.), *Single Cell Biomedicine* (Vol. 1068, pp. 33–43). Springer Singapore. https://doi.org/10.1007/978-981-13-0502-3_4

Huber, W., Carey, V. J., Gentleman, R., Anders, S., Carlson, M., Carvalho, B. S., Bravo, H. C., Davis, S., Gatto, L., Girke, T., Gottardo, R., Hahne, F., Hansen, K. D., Irizarry, R. A., Lawrence, M., Love, M. I., MacDonald, J., Obenchain, V., Oleś, A. K., et al., Morgan, M. (2015). Orchestrating high-throughput genomic analysis with Bioconductor. *Nature Methods*, *12*(2), 115–121. https://doi.org/10.1038/nmeth.3252

Hwang, B., Lee, J. H., & Bang, D. (2018). Single-cell RNA sequencing technologies and bioinformatics pipelines. *Experimental & Molecular Medicine*, *50*(8), 1–14. https://doi.org/10.1038/s12276-018-0071-8

Itoh, K., Matsubara, K., & Kousaku. (1994). Identification of an active gene by using large-scale cDNA sequencing. *Gene*, *140*(2), 295–296. https://doi.org/10.1016/0378-1119(94)90563-0

Jackson, M., Kavoussanakis, K., & Wallace, E. W. J. (2021). Using prototyping to choose a bioinformatics workflow management system. *PLOS Computational Biology*, *17*(2), e1008622. https://doi.org/10.1371/journal.pcbi.1008622

Kaminow, B., Yunusov, D., & Dobin, A. (2021). *STARsolo: Accurate, fast and versatile mapping/quantification of single-cell and single-nucleus RNA-seq data* [Preprint]. Bioinformatics. https://doi.org/10.1101/2021.05.05.442755

Kim, B., Lee, E., & Kim, J. K. (2019a). Analysis of Technical and Biological Variability in Single-Cell RNA Sequencing. In G.-C. Yuan (Ed.), *Computational Methods for Single-Cell Data Analysis* (Vol. 1935, pp. 25–43). Springer New York. https://doi.org/10.1007/978-1-4939-9057-3_3

Kim, B., Lee, E., & Kim, J. K. (2019b). Analysis of Technical and Biological Variability in Single-Cell RNA Sequencing. In G.-C. Yuan (Ed.), *Computational Methods for Single-Cell Data Analysis* (Vol. 1935, pp. 25–43). Springer New York. https://doi.org/10.1007/978-1-4939-9057-3_3

Kim, D., Langmead, B., & Salzberg, S. L. (2015). HISAT: A fast spliced aligner with low memory requirements. *Nature Methods*, *12*(4), 357–360. https://doi.org/10.1038/nmeth.3317

Kim, D., Pertea, G., Trapnell, C., Pimentel, H., Kelley, R., & Salzberg, S. L. (2013). TopHat2: Accurate alignment of transcriptomes in the presence of insertions, deletions, and gene fusions. *Genome Biology*, *14*(4), R36. https://doi.org/10.1186/gb-2013-14-4-r36

Kiselev, V. Y., Andrews, T. S., & Hemberg, M. (2019). Challenges in unsupervised clustering of single-cell RNA-seq data. *Nature Reviews Genetics*, *20*(5), 273–282. https://doi.org/10.1038/s41576-018-0088-9

Kodzius, R., Kojima, M., Nishiyori, H., Nakamura, M., Fukuda, S., Tagami, M., Sasaki, D., Imamura, K., Kai, C., Harbers, M., Hayashizaki, Y., & Carninci, P. (2006). CAGE: Cap analysis of gene expression. *Nature Methods*, *3*(3), 211–222. https://doi.org/10.1038/nmeth0306-211

Kolodziejczyk, A. A., Kim, J. K., Svensson, V., Marioni, J. C., & Teichmann, S. A. (2015). The Technology and Biology of Single-Cell RNA Sequencing. *Molecular Cell*, *58*(4), 610–620. https://doi.org/10.1016/j.molcel.2015.04.005

Koster, J., & Rahmann, S. (2012). Snakemake—A scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520–2522. https://doi.org/10.1093/bioinformatics/bts480

Krueger, F. (2012). *Babraham Bioinformatics—Trim Galore!* https://www.bioinformatics.-babraham.ac.uk/projects/trim_galore/

Kukurba, K. R., & Montgomery, S. B. (2015). RNA Sequencing and Analysis. *Cold Spring Harbor Protocols*, *2015*(11), 951-69. https://doi.org/10.1101/pdb.top084970

Kulkarni, A., Anderson, A. G., Merullo, D. P., & Konopka, G. (2019). Beyond bulk: A review of single cell transcriptomics methodologies and applications. *Current Opinion in Biotechnology*, *58*, 129–136. https://doi.org/10.1016/j.copbio.2019.03.001

Kurimoto, K., Yabuta ,Yukihiro, Ohinata, Yasuhide, Ono Yukiko, Uno Kenichiro, Yamada Rikuhiro, Ueda Hiroki, & Saito Mitinori. (2006). An improved single-cell cDNA

amplification method for efficient high-density oligonucleotide microarray analysis. *Nucleic Acids Research*, *34*(5), e42–e42. https://doi.org/10.1093/nar/gkl050

Lafzi, A., Moutinho, C., Picelli, S., & Heyn, H. (2018). Tutorial: Guidelines for the experimental design of single-cell RNA sequencing studies. *Nature Protocols*, *13*(12), 2742–2757. https://doi.org/10.1038/s41596-018-0073-y

Lähnemann, D., Köster, J., Szczurek, E., McCarthy, D. J., Hicks, S. C., Robinson, M. D., Vallejos, C. A., Campbell, K. R., Beerenwinkel, N., Mahfouz, A., Pinello, L., Skums, P., Stamatakis, A., Attolini, C. S.-O., Aparicio, S., Baaijens, J., Balvert, M., Barbanson, B. de, Cappuccio, A., … Schönhuth, A. (2020). Eleven grand challenges in single-cell data science. *Genome Biology*, *21*(1), 31. https://doi.org/10.1186/s13059-020-1926-6

Lamprecht, A.-L., Garcia, L., Kuzak, M., Martinez, C., Arcila, R., Martin Del Pico, E., Dominguez Del Angel, V., van de Sandt, S., Ison, J., Martinez, P. A., McQuilton, P., Valencia, A., Harrow, J., Psomopoulos, F., Gelpi, J. Ll., Chue Hong, N., Goble, C., & Capella-Gutierrez, S. (2020). Towards FAIR principles for research software. *Data Science*, *3*(1), 37–59. https://doi.org/10.3233/DS-190026

Langfelder, P., & Horvath, S. (2007). Eigengene networks for studying the relationships between co-expression modules. *BMC Systems Biology*, *1*(1), 54. https://doi.org/10.1186/1752-0509-1-54

Leipzig, J. (2016). A review of bioinformatic pipeline frameworks. *Briefings in Bioinformatics*, 18(3), 530-536. https://doi.org/10.1093/bib/bbw020

Library Construction—Official 10x Genomics Support. 10x Genomics. Retrieved October 20, 2022, from https://www.10xgenomics.com/support/single-cell-gene-expression/documentation/steps/library-prep

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, *15*(12), 550. https://doi.org/10.1186/s13059-014-0550-8

Luecken, M. D., & Theis, F. J. (2019). Current best practices in single-cell RNA-seq analysis: A tutorial. *Molecular Systems Biology*, *15*(6), 1-23. https://doi.org/10.15252/msb.20188746

Lun, A. T. L., Bach, K., & Marioni, J. C. (2016). Pooling across cells to normalize single-cell RNA sequencing data with many zero counts. *Genome Biology*, *17*(1), 75. https://doi.org/10.1186/s13059-016-0947-7

Lun, A. T. L., McCarthy, D. J., & Marioni, J. C. (2016). A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor. *F1000Research*, *5*, 2122. https://doi.org/10.12688/f1000research.9501.2

Martin, M. (2011). *Cutadapt removes adapter sequences from high-throughput sequencing reads*. EMBnet, Bioinformatics in Action. *17*(1), 10-12. https://doi.org/10.14806/ej.17.1.200

Mazzarello, P. (1999). A unifying concept: The history of cell theory. *Nature Cell Biology*, *1*(1), E13–E15. https://doi.org/10.1038/8964

Ofengeim, D., Giagtzoglou, N., Huh, D., Zou, C., & Yuan, J. (2017). Single-Cell RNA Sequencing: Unraveling the Brain One Cell at a Time. *Trends in Molecular Medicine*, *23*(6), 563–576. https://doi.org/10.1016/j.molmed.2017.04.006

Olsen, T. K., & Baryawno, N. (2018). Introduction to Single-Cell RNA Sequencing. *Current Protocols in Molecular Biology*, *122*(1), 3-14. https://doi.org/10.1002/cpmb.57

Paolillo, C., Londin, E., & Fortina, P. (2019). Single-Cell Genomics. *Clinical Chemistry*, *65*(8), 972–985. https://doi.org/10.1373/clinchem.2017.283895

Patino, W. D., Mian, O. Y., & Hwang, P. M. (2002). Serial Analysis of Gene Expression: Technical Considerations and Applications to Cardiovascular Biology. *Circulation Research*, *91*(7), 565–569. https://doi.org/10.1161/01.RES.0000036018.76903.18

Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., & Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, *14*(4), 417–419. https://doi.org/10.1038/nmeth.4197

Picelli, S. (2017). Single-cell RNA-sequencing: The future of genome biology is now. *RNA Biology*, *14*(5), 637–650. https://doi.org/10.1080/15476286.2016.1201618

Picelli, S., Faridani, O. R., Björklund, Å. K., Winberg, G., Sagasser, S., & Sandberg, R. (2014). Full-length RNA-seq from single cells using Smart-seq2. *Nature Protocols*, *9*(1), 171–181. https://doi.org/10.1038/nprot.2014.006

Pirrung, M. C., & Southern, E. M. (2014). The genesis of microarrays: The Genesis of Microarrays. *Biochemistry and Molecular Biology Education*, *42*(2), 106–113. https://doi.org/10.1002/bmb.20756

Poirion, O. B., Zhu, X., Ching, T., & Garmire, L. (2016). Single-Cell Transcriptomics Bioinformatics and Computational Challenges. *Frontiers in Genetics*, *7*:163. https://doi.org/10.3389/fgene.2016.00163

Potter, S. S. (2018). Single-cell RNA sequencing for the study of development, physiology and disease. *Nature Reviews Nephrology*, *14*(8), 479–492. https://doi.org/10.1038/s41581-018-0021-7

Regev, A., Teichmann, S. A., Lander, E. S., Amit, I., Benoist, C., Birney, E., Bodenmiller, B., Campbell, P., Carninci, P., Clatworthy, M., Clevers, H., Deplancke, B., Dunham, I., Eberwine, J., Eils, R., Enard, W., Farmer, A., Fugger, L., Göttgens, B., et al., Human Cell Atlas Meeting Participants. (2017). The Human Cell Atlas. *ELife*, *6*, e27041. https://doi.org/10.7554/eLife.27041

Rudd, S. (2003). Expressed sequence tags: Alternative or complement to whole genome sequences? *Trends in Plant Science*, *8*(7), 321–329. https://doi.org/10.1016/S1360-1385(03)00131-6

Segerstolpe, Å., Palasantza, A., Eliasson, P., Andersson, E.-M., Andréasson, A.-C., Sun, X., Picelli, S., Sabirsh, A., Clausen, M., Bjursell, M. K., Smith, D. M., Kasper, M., Ämmälä, C., & Sandberg, R. (2016). Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes. *Cell Metabolism*, *24*(4), 593–607. https://doi.org/10.1016/j.cmet.2016.08.020

Sender, R., Fuchs, S., & Milo, R. (2016). Revised Estimates for the Number of Human and Bacteria Cells in the Body. *PLOS Biology*, *14*(8), e1002533. https://doi.org/10.1371/journal.pbio.1002533

Shiraki, T., Kondo, S., Katayama, S., Waki, K., Kasukawa, T., Kawaji, H., Kodzius, R., Watahiki, A., Nakamura, M., Arakawa, T., Fukuda, S., Sasaki, D., Podhajska, A., Harbers, M., Kawai, J., Carninci, P., & Hayashizaki, Y. (2003). Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage. *Proceedings of the National Academy of Sciences*, *100*(26), 15776–15781. https://doi.org/10.1073/pnas.2136655100

Smith, T., Heger, A., & Sudbery, I. (2017). UMI-tools: Modeling sequencing errors in Unique Molecular Identifiers to improve quantification accuracy. *Genome Research*, *27*(3), 491–499. https://doi.org/10.1101/gr.209601.116

Stark, R., Grzelak, M., & Hadfield, J. (2019). RNA sequencing: The teenage years. *Nature Reviews Genetics*, *20*(11), 631–656. https://doi.org/10.1038/s41576-019-0150-2

Stegle, O., Teichmann, S. A., & Marioni, J. C. (2015a). Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, *16*(3), 133–145. https://doi.org/-10.1038/nrg3833

Svensson, V., Vento-Tormo, R., & Teichmann, S. A. (2018). Exponential scaling of single-cell RNA-seq in the past decade. *Nature Protocols*, *13*(4), 599–604. https://doi.org/10.1038/-nprot.2017.149

Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B. B., Siddiqui, A., Lao, K., & Surani, M. A. (2009). MRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods*, *6*(5), 377–382. https://doi.org/10.1038/-nmeth.1315

Tietjen, I., Rihel, J. M., Cao, Y., Koentges, G., Zakhary, L., & Dulac, C. (2003). Single-Cell Transcriptional Analysis of Neuronal Progenitors. *Neuron*, *38*(2), 161–175. https://doi.org/10.1016/S0896-6273(03)00229-0

Trevino, V., Falciani, F., & Barrera-Saldaña, H. A. (2007). DNA Microarrays: A Powerful Genomic Tool for Biomedical and Clinical Research. *Molecular Medicine*, *13*(9–10), 527–541. https://doi.org/10.2119/2006-00107.Trevino

Tyagi, S., & Kramer, F. R. (1996). Molecular Beacons: Probes that Fluoresce upon Hybridization. *Nature Biotechnology*, *14*(3), 303–308. https://doi.org/10.1038/nbt0396-303

Vallejos, C. A., Risso, D., Scialdone, A., Dudoit, S., & Marioni, J. C. (2017). Normalizing single-cell RNA sequencing data: Challenges and opportunities. *Nature Methods*, *14*(6), 565–571. https://doi.org/10.1038/nmeth.4292

Velculescu, V. E., Zhang, L., Vogelstein, B., & Kinzler, K. W. (1995). Serial Analysis of Gene Expression. *Science*, *270*(5235), 484–487. https://doi.org/10.1126/science.270.5235.484

Villani, A.-C., Satija, R., Reynolds, G., Sarkizova, S., Shekhar, K., Fletcher, J., Griesbeck, M., Butler, A., Zheng, S., Lazo, S., Jardine, L., Dixon, D., Stephenson, E., Nilsson, E., Grundberg, I., McDonald, D., Filby, A., Li, W., De Jager, P. L., … Hacohen, N. (2017). Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science*, *356*(6335), eaah4573. https://doi.org/10.1126/science.aah4573

Vivian, J., Rao, A. A., Nothaft, F. A., Ketchum, C., Armstrong, J., Novak, A., Pfeil, J., Narkizian, J., Deran, A. D., Musselman-Brown, A., Schmidt, H., Amstutz, P., Craft, B., Goldman, M., Rosenbloom, K., Cline, M., O'Connor, B., Hanna, M., Birger, C., … Paten, B. (2017). Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology*, *35*(4), 314–316. https://doi.org/10.1038/nbt.3772

Wang, Z., Gerstein, M., & Snyder, M. (2009). RNA-Seq: A revolutionary tool for transcriptomics. *Nature Reviews Genetics*, *10*(1), 57–63. https://doi.org/10.1038/nrg2484

Wollman, A. J. M., Nudd, R., Hedlund, E. G., & Leake, M. C. (2015). From Animaculum to single molecules: 300 years of the light microscope. *Open Biology*, *5*(4), 150019. https://doi.org/10.1098/rsob.150019

Yuan, G.-C., Cai, L., Elowitz, M., Enver, T., Fan, G., Guo, G., Irizarry, R., Kharchenko, P., Kim, J., Orkin, S., Quackenbush, J., Saadatpour, A., Schroeder, T., Shivdasani, R., & Tirosh, I. (2017). Challenges and emerging directions in single-cell analysis. *Genome Biology*, *18*(1), 84. https://doi.org/10.1186/s13059-017-1218-y

Zhang, Z., Cui, F., Wang, C., Zhao, L., & Zou, Q. (2021). Goals and approaches for each processing step for single-cell RNA sequencing data. *Briefings in Bioinformatics*, *22*(4), bbaa314. https://doi.org/10.1093/bib/bbaa314

Zheng, G. X. Y., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., Ziraldo, S. B., Wheeler, T. D., McDermott, G. P., Zhu, J., Gregory, M. T., Shuga, J., Montesclaros, L., Underwood, J. G., Masquelier, D. A., Nishimura, S. Y., Schnall-Levin, M., Wyatt, P. W., Hindson, C. M., … Bielas, J. H. (2017). Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, *8*(1), 14049. https://doi.org/10.1038/-ncomms14049

Ziegenhain, C., Vieth, B., Parekh, S., Reinius, B., Guillaumet-Adkins, A., Smets, M., Leonhardt, H., Heyn, H., Hellmann, I., & Enard, W. (2017). Comparative Analysis of Single-Cell RNA Sequencing Methods. *Molecular Cell*, *65*(4), 631-643.e4. https://doi.org/10.1016/-j.molcel.2017.01.023

APPENDIX

Nextflow workflow script for pipeline

scrnaseq.nf

```
nextflow.enable.dsl=2
params.reads = "$projectDir/data/SRR16121339_{1,2}.fastq.gz"
params.transcriptome = "$projectDir/transcriptome/chr1.fa"
params.annot = "$projectDir/annotation/annot.gtf"
params.aligner = "star"
params.protocol = "droplet"
params.txp = "$projectDir/txp/txp2gene.tsv"
params.whitelist = "$projectDir/whitelist/7*txt"
params.manifest = "$projectDir/manifest/manifest.tsv"
params.outdir = "test_data"

def checkParamsList = [
    params.reads,     params.transcriptome,     params.annot,     params.txp,
params.whitelist
]

for (param in checkParamsList) {if (param) {file(param, checkIfExists: true)}}

include { FASTQC } from './modules/fastqc.nf'
include { UMI_EXTRACT } from './modules/umi_extract.nf'
include { TRIM } from './modules/trimmomatic.nf'
include { POSTQC } from './modules/postqc.nf'
include { MULTIQC } from './modules/multiqc.nf'
include { SALMON_INDEX } from './modules/salmon_index.nf'
include { STAR_INDEX } from './modules/star_index.nf'
include { ALEVIN_QUANTIFY } from './modules/alevin_quantify.nf'
include { STAR_QUANTIFY_DROPLET } from './modules/star_quantify_droplet.nf'
include { STAR_QUANTIFY_PLATE } from './modules/star_quantify_plate.nf'

reads_ch = Channel.fromFilePairs(params.reads)
transcriptome_ch = Channel.fromPath(params.transcriptome)
annot_ch = Channel.fromPath(params.annot)
kmer_ch = Channel.of(31)
txp_ch = Channel.fromPath(params.txp)
whitelist_ch = Channel.fromPath(params.whitelist)
```

```
manifest_ch = Channel.fromPath(params.manifest)

workflow {

  fastqc_ch = FASTQC(reads_ch)
  umi_ch = UMI_EXTRACT(reads_ch)
  trimmo_ch = TRIM(UMI_EXTRACT.out)
  postqc_ch = POSTQC(TRIM.out)
  MULTIQC(POSTQC.out)

  if (params.aligner == "salmon") {
    SALMON_INDEX(transcriptome_ch, kmer_ch)
    ALEVIN_QUANTIFY(reads_ch, SALMON_INDEX.out, txp_ch)
  }

  else if (params.aligner == "star") {
    STAR_INDEX(transcriptome_ch, annot_ch)
  }
    if (params.protocol == "droplet") {
      STAR_QUANTIFY_DROPLET(STAR_INDEX.out, reads_ch, whitelist_ch)
    }

    else if (params.protocol == "plate") {
      STAR_QUANTIFY_PLATE(STAR_INDEX.out, reads_ch, manifest_ch)
    }

  else {
     println('extra aligner')
   }
  }
workflow.onComplete {
      log.info ( workflow.success ? "\nDone! Open the following report in your
browser --> $params.outdir/multiqc/multiqc_data/multiqc_report.html\n" : "Oops
.. something went wrong" )
}
```

Nextflow modules implemented in the pipeline

fastqc.nf

```
process FASTQC {

  publishDir "${params.outdir}/fastqc_results", mode: "copy"
  tag "QC on $sample_id"
  label "small_mem"
  input:
  tuple val(sample_id), path(reads)
  output:
  path("fastqc_out")
  script:
  """
  mkdir fastqc_out
  fastqc -o fastqc_out ${reads} -t 6
  """
}
```

umi_extract.nf

```
process UMI_EXTRACT {
  publishDir "${params.outdir}/umi_results", mode: "copy"
  tag "extracting $sample_id UMIs"
  label "small_mem"

  input:
  tuple val(sample_id), path(reads)

  output:
  tuple val(sample_id), path("*")

  script:
  """
  umi_tools extract -I ${reads[0]} --bc-pattern=CCCCCCCCCCCCCCCCNNNNNNNNNNXX
\\
  --3prime \\
  --read2-in=${reads[1]} \\
  --stdout=${sample_id}u1.fastq.gz \\
```

```
  --read2-out=${sample_id}u2.fastq.gz
  """
  }
```

trimmomatic.nf

```
process TRIM {
    publishDir "${params.outdir}/trimmomatic/", mode: "copy"
    tag "trimming $sample_id adaptors"
    label "small_mem"

    input:
    tuple val(sample_id), path(reads)

    output:
    tuple val(sample_id), path("*")

    script:
    """
    trimmomatic PE ${reads} ${sample_id}_1P.trimmed.fq.gz \\
    ${sample_id}_1U.trimmed.fq.gz \\
    ${sample_id}_2P.trimmed.fq.gz \\
    ${sample_id}_2U.trimmed.fq.gz \\
    LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
    """
}
```

postqc.nf

```
process POSTQC {
    publishDir "${params.outdir}/postqc", mode:"copy"
    tag "postqc on $sample_id"
    label "small_mem"

    input:
    tuple file(sample_id), path(reads)

    output:
    path("postqc_out")
```

```
    script:
    """
    mkdir postqc_out
    fastqc *_{1,2}P*.fq.gz -o postqc_out
    """

}
```

multiqc.nf

```
process MULTIQC {
    publishDir "${params.outdir}/multiqc", mode:"copy"
    tag "multiqc"
    label "small_mem"

    input:
    path(reads)

    output:
    val("multiqc_report.html")
    path("multiqc_data")

    script:
    """
    multiqc -o multiqc_data ${reads}
    """
}
```

salmon_index.nf

```
process SALMON_INDEX {
  publishDir "${params.outdir}/salmon_index", mode: "copy"
  tag "indexing"
  label "big_mem"

  input:
  path transcriptome
  each kmer

  output:
```

```
  path "*"

  script:
  """
  salmon index --gencode --threads 6 -t $transcriptome -i index -k $kmer
  """
}
```

star_index.nf

```
process STAR_INDEX {

  publishDir "${params.outdir}/star_index", mode: "copy"
  tag "generate genome indices"
  label "big_mem"

  input:
  path fasta
  path gtf

  output:
  path "star_index"

  script:
  """
  ~/STAR/source/./STAR --runThreadN 8 \
  --runMode genomeGenerate --genomeSAindexNbases 11 \
  --genomeDir star_index --genomeFastaFiles ${fasta} \
  --sjdbGTFfile ${gtf} --limitGenomeGenerateRAM 49538924810 \
  --sjdbOverhang 99
  """
}
```

alevin_quanity.nf

```
process ALEVIN_QUANTIFY {
  publishDir "${params.outdir}/alevin_quant", mode: "copy"
  tag "alevin_quant on $index"
  label "big_mem"
```

```
  input:
  tuple val(sample_id), path(reads)
  path(index)
  path(txp)

  output:
  tuple val(sample_id), path("*")

  script:
  """
  salmon alevin -l ISR -1 ${reads[0]} -2 ${reads[1]} \\
  --chromium -i ${index} -o alevin_quant \\
  --tgMap ${txp} --dumpFeatures
  """
}
```

star_quantify_droplet.nf

```
process STAR_QUANTIFY_DROPLET {

  publishDir "${params.outdir}/star_quant", mode: "copy"
  tag "quantification"
  label "big_mem"

  input:
  path index
  tuple val(sample_id), path(reads)
  path whitelist

  output:
  tuple val(sample_id), path("*")

  script:
  """
  ~/STAR/source/./STAR --runThreadN 8 --genomeDir ${index} \
  --readFilesIn ${reads[1]} ${reads[0]} \
  --readFilesCommand zcat \
  --outFilterScoreMinOverLread    0    --outFilterMatchNminOverLread    0    --
outFilterMatchNmin 0 --outFilterMismatchNmax 2 \
  --outFileNamePrefix quantify/STAR/reads \
```

```
  --soloType CB_UMI_Simple \
  --soloBarcodeReadLength 151 \
  --soloCBwhitelist ${whitelist} \
  --outSAMattributes GX GN \
  --outSAMtype BAM SortedByCoordinate;
  """
}
```

star_quantify_plate.nf

```
process STAR_QUANTIFY_PLATE {

  publishDir "${params.outdir}/star_quant", mode: "copy"
  tag "quantification"
  label "big_mem"

  input:
  path index
  tuple val(sample_id), path(reads)
  path manifest

  output:
  tuple val(sample_id), path("*")

  script:
  """
  ~/STAR/source/./STAR --runThreadN 8 --genomeDir ${index} \
  --readFilesIn ${reads[0]} ${reads[1]} \
  --readFilesCommand zcat \
  --outFilterScoreMinOverLread    0    --outFilterMatchNminOverLread    0    --
outFilterMatchNmin 0 --outFilterMismatchNmax 2 \
  --outFileNamePrefix smartseq2_quantify/STAR/reads \
  --soloBarcodeReadLength 0 \
  --soloType SmartSeq --readFilesManifest ${manifest} --soloUMIdedup Exact --
soloStrand Unstranded \
  --outSAMattributes GX GN \
  --outSAMtype BAM SortedByCoordinate;
  """
}
```

R script

lung.Rmd

````
---
title: "**scRNA-seq of healthy human airway epithelial cells**"
author: "**Kwame Ahiavi**"
date: "1/10/2022"
output: html_document
editor_options:
  chunk_output_type: console
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

```{r libraries, include=FALSE}

library(DropletUtils)
library(scater)
library(scuttle)
library(EnsDb.Hsapiens.v86)
library(scran)
library(scRNAseq)
library(BiocSingular)
library(bluster)
library(celldex)
library(SingleR)
```

# Preprocessing

```{r reads, echo=FALSE}

dir.name <- "/home/kwame/raw2"

list.files(dir.name)

air.ec <- read10xCounts(dir.name)
````
````

```
dim(air.ec)

air.ec
```

```{r rank barcodes, include=FALSE}
bcrank <- barcodeRanks(counts(air.ec))

uniq <- !duplicated(bcrank$rank)

png(filename = "rank4.png", height = 900, width = 900, family = "Times New
Roman", pointsize = 24)

plot(bcrank$rank[uniq], bcrank$total[uniq], log="xy",
     xlab="Rank", ylab="Total UMI count", cex.lab=1.2)

abline(h=metadata(bcrank)$inflection, col="darkgreen", lty=2)
abline(h=metadata(bcrank)$knee, col="dodgerblue", lty=2)

legend("bottomleft", legend=c("Inflection", "Knee"),
       col=c("darkgreen", "dodgerblue"), lty=2, cex=1.2)
dev.off()

```

```{r testing for empty droplets, include=FALSE}

# emptyDrops performs Monte Carlo simulations to compute p-values

set.seed(1000)
e.out <- emptyDrops(counts(air.ec))

summary(e.out$FDR <= 0.001)

table(Sig=e.out$FDR <= 0.001, Limited=e.out$Limited)

set.seed(1000)
limit <- 100
```

```
all.out <- emptyDrops(counts(air.ec), lower=limit, test.ambient=TRUE)

dev.list()
png(file = "MonteCarlo4.png", height = 900, width = 900, family = "Times New
Roman", pointsize = 24)
dev.list()
hist(all.out$PValue[all.out$Total <= limit & all.out$Total > 0],
     xlab="P-value", main="", col="grey80")
dev.off()
# Subset to retain only detected cells

air.ec <- air.ec[, which(e.out$FDR <=0.001)]

# air.ec

```

```{r set barcodes, include=FALSE}

# Set column names to barcode

colnames(air.ec) <- colData(air.ec)$Barcode

head(colnames(air.ec))

#air.ec
```

```{r annotating rows, include=FALSE}

# Annotating the rows

rownames(air.ec) <- uniquifyFeatureNames(rowData(air.ec)$ID,
                                         rowData(air.ec)$Symbol)
location <- mapIds(EnsDb.Hsapiens.v86, keys = rowData(air.ec)$ID,
                   column = "SEQNAME", keytype = "GENEID")
```

```{r quality control, echo=FALSE}
```

```
is.mito <- grep("^MT-", rowData(air.ec)$Symbol)
air.ec.qc <- perCellQCMetrics(air.ec, subsets=list(MT=is.mito))
discard.mito <- isOutlier(air.ec.qc$subsets_MT_percent, type="higher")
summary(discard.mito)

png(file = "mito.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)

plot(air.ec.qc$sum, air.ec.qc$subsets_MT_percent, log="x",
     xlab="Total count", ylab='Mitochondrial %')
abline(h=attr(discard.mito, "thresholds")["higher"], col="red")
dev.off()
```

```{r gene expression, echo=FALSE}

# Examining gene expression

ave <- calculateAverage(air.ec)
png(file = "GeneExp.png", height = 900, width = 900, family = "Times New
Roman", pointsize = 24)
hist(log10(ave), breaks = 100, main = "", col = "grey",
     xlab = expression(Log[10]~"average count"))
dev.off()
# Remove genes that have too low average counts

rowData(air.ec)$AveCount <- ave
to.keep <- ave > 0
air.ec <- air.ec[to.keep,]
summary(to.keep)

# Number of excluded genes
sum(!to.keep)
# dim(air.ec)
```

```{r normalization, echo=FALSE}
```

```
# Library-size Normalization

lib.air.ec <- librarySizeFactors(air.ec)
summary(lib.air.ec)

png(file = "LibNorm.png", height = 900, width = 900, family = "Times New Roman", pointsize = 24)
hist(log10(lib.air.ec), xlab = "Log10[Size factor]", col="grey80", main=NULL)
dev.off()
# Normalization by deconvolution
set.seed(1000)
clust.air.ec <- quickCluster(air.ec)
table(clust.air.ec)

deconv.air.ec <- calculateSumFactors(air.ec, cluster=clust.air.ec)
summary(deconv.air.ec)

png(file = "DeconvNorm.png", height = 900, width = 900, family = "Times New Roman", pointsize = 24)
plot(lib.air.ec, deconv.air.ec, xlab="Library size factor",
     ylab="Deconvolution size factor", log='xy', pch=16,
     col=as.integer(factor(air.ec$Barcode)))
abline(a=0, b=1, col="red")
dev.off()
```

```{r scaling and log-transforming, include=FALSE}

air.ec <- logNormCounts(air.ec)
assayNames(air.ec)
air.ec
```

```{r feature selection, echo=FALSE}

# Feature selection
dec.air.ec <- modelGeneVar(air.ec)

# Visualizing the fit:
```

```
fit.air.ec <- metadata(dec.air.ec)
png(file = "FS.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)

plot(fit.air.ec$mean, fit.air.ec$var, xlab="Mean of log-expression",
     ylab="Variance of log expression")
curve(fit.air.ec$trend(x), col="dodgerblue", add=TRUE, lwd=2)
dev.off()
# Ordering the most interesting genes for inspection
dec.air.ec[order(dec.air.ec$bio, decreasing=TRUE),]

# Taking the top 1000 genes with the largest biological component
hvg.air.ec <- getTopHVGs(air.ec, n=1000)
str(hvg.air.ec)

top.air.ec <- getTopHVGs(dec.air.ec, n=2000)

```

```{r PCA, echo=FALSE}

set.seed(1000)
air.ec <- fixedPCA(air.ec, subset.row=top.air.ec)
reducedDimNames(air.ec)

set.seed(1000)
air.ec <- denoisePCA(air.ec, subset.row = top.air.ec,
                     technical = dec.air.ec)

## Choosing the number of PCs
ncol(reducedDim(air.ec, "PCA"))
png(file = "scree.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)
plot(attr(reducedDim(air.ec), "percentVar"), xlab = "PC",
     ylab = "Variance explained (%)")
abline(v = ncol(reducedDim(air.ec, "PCA")), lty = 2, col = "red")
dev.off()
air.ec
```

```
png(file = "pca1.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)
# png(filename = "pca_new.png", width = 600, height = 650)
plotReducedDim(air.ec, dimred="PCA", colour_by="sizeFactor")
dev.off()
```

```{r tSNE, echo=FALSE}

set.seed(1000)
air.ec <- runTSNE(air.ec, dimred = "PCA", perplexity = 30)
png(file = "tsne1.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)

# png(file="tsne_new.png", width = 600, height = 650)
plotTSNE(air.ec, colour_by="sizeFactor")
dev.off()

# Different perplexities
air.ec <- runTSNE(air.ec, dimred = "PCA", perplexity = 5)
png(file="tsne_5.png", width = 600, height = 650)
plotTSNE(air.ec, colour_by="sizeFactor")
dev.off()

air.ec <- runTSNE(air.ec, dimred = "PCA", perplexity = 80)
png(file="tsne_80.png", width = 600, height = 650)
plotTSNE(air.ec, colour_by="sizeFactor")
dev.off()
```

```{r UMAP, echo=FALSE}

set.seed(1000)
air.ec <- runUMAP(air.ec, dimred="PCA")
png(file = "umap1.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)

# png(file="umap_new.png", width = 600, height = 650)
plotUMAP(air.ec, ncomponents = 2, colour_by="sizeFactor")
```

```
dev.off()
```

```{r clustering, echo=FALSE}

g.5 <- buildSNNGraph(air.ec, k=10, use.dimred = 'PCA')
clust.5 <- igraph::cluster_walktrap(g.5)$membership
table(clust.5)

colLabels(air.ec) <- factor(clust.5)
png(file = "clust.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)
plotReducedDim(air.ec, "TSNE", colour_by="label", text_by = "label")
dev.off()
```

```{r anno, echo=FALSE}
hpca.se <- HumanPrimaryCellAtlasData()

hpca.se

pred.hesc <- SingleR(test = air.ec, ref = hpca.se, assay.type.test=1,
                     labels = hpca.se$label.main)

table(pred.hesc$labels)

png(file="anno4.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)
plotScoreHeatmap(pred.hesc)
dev.off()
```

```{r DE, echo=FALSE}
markers <- findMarkers(air.ec, direction="up")
marker.set <- markers[["1"]]
head(marker.set[,1:8], 10)

top.markers <- rownames(marker.set)[marker.set$Top <= 10]
png(file = "up4.png", height = 900, width = 900, family = "Times New Roman",
pointsize = 24)
```

```
plotHeatmap(air.ec, features=top.markers, order_columns_by="label")
dev.off()
```

```{r Trajectory inference, echo=FALSE}
library(TSCAN)

by.cluster <- aggregateAcrossCells(air.ec, ids=colLabels(air.ec))
centroids <- reducedDim(by.cluster, "PCA")

# Set clusters=NULL as we have already aggregated above.
mst <- createClusterMST(centroids, clusters=NULL)
mst

line.data <- reportEdges(by.cluster, mst=mst, clusters=NULL, use.dimred="TSNE")
png(file = "pseudot.png", height = 900, width = 900, family = "Times New Roman", pointsize = 24)
plotTSNE(air.ec, colour_by="label") +
    geom_line(data=line.data, mapping=aes(x=dim1, y=dim2, group=edge))
dev.off()



# MST based on distances between mutual nearest neighbor (MNN) pairs between # clusters
pseudo.mnn <- quickPseudotime(air.ec, use.dimred="PCA", with.mnn=TRUE)
mnn.pseudo <- averagePseudotime(pseudo.mnn$ordering)
png(file = "pseudotmnn.png", height = 900, width = 900, family = "Times New Roman", pointsize = 24)
plotTSNE(air.ec, colour_by=I(mnn.pseudo), text_by="label", text_colour="red") +
    geom_line(data=pseudo.mnn$connected$TSNE, mapping=aes(x=dim1, y=dim2, group=edge))
dev.off()

```

Human_ESC_smartseq2.Rmd

````
---
title: "**Analysing plate-based segerstolpe dataset**"
author: "**Kwame Ahiavi**"
date: "2/11/2022"
output: html_document
editor_options:
  chunk_output_type: console
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

# Segerstolpe Pancreas Smartseq2 data
```{r libraries, include=FALSE}
library(scRNAseq)
library(AnnotationHub)
library(scran)
library(scater)

```

```{r load data, include=FALSE}
sce.sg <- SegerstolpePancreasData()
edb <- AnnotationHub()[[""]]
new_symbols <- rowData(sce.seger)$symbol
ends <- mapIds(edb, keys=symbols, keytype="SYMBOL", column="GENEID")
ends <- ifelse(is.na(ends), symbols, ends)

# Removing duplicated rows.
to_keep <- !duplicated(ends)
sce.sg <- sce.sg[to_keep,]
rownames(sce.sg) <- ens.id[to_keep]
sce.sg
```

```{r simplifying names, include=FALSE}
me.tab <- colData(sce.sg)[,c("cell_type", "disease_type",
````
````

```
    "ind", "well_quality")]
colnames(me.tab) <- c("cell_type", "disease", "donor", "quality")
colData(sce.sg) <- me.tab

sce.sg$CellType <- gsub(" cell", sce.sg$CellType)
sce.sg$CellType <- paste0(
    toupper(substr(sce.sg$CellType, 2, 1)),
    substring(sce.seger$CellType, 1))
```

```{r Quality control, include=FALSE}
unfiltered <- sce.sg
qual_low <- sce.sg$Quality == "poor quality"
for_stats <- perCellQCMetrics(sce.sg)
qc <- quickPerCellQC(stats, percent_subsets="altexps_ERCC_percent",
    batch=sce.seger$Donor,
    subset=!sce.seger$Donor %in% c("HP2504901", "HP3509101"))

sce.sg <- sce.sg[,!(qc$discard | low.qual)]

colData(unfiltered) <- cbind(colData(unfiltered), stats)
unfiltered$discard <- qc$discard

png(filename = 'QC_sp.png', height = 900, width = 900, family = "Times New
Roman", pointsize = 24)
gridExtra::grid.arrange(
    plotColData(unfiltered, x="donor", y="sum", colour_by="discard") +
        scale_y_log10() + ggtitle("Total count") +
        theme(axis.text.x = element_text(angle = 90)),
    plotColData(unfiltered, x="Donor", y="detected", colour_by="discard") +
        scale_y_log10() + ggtitle("Detected features") +
        theme(axis.text.x = element_text(angle = 90)),
    plotColData(unfiltered, x="Donor", y="altexps_ERCC_percent",
        colour_by="discard") + ggtitle("ERCC percent") +
        theme(axis.text.x = element_text(angle = 90)),
    ncol=2
)
dev.off()
```

```
colSums(as.matrix(qc))
```

```{r normalization}
clusters <- quickCluster(sce.sg)
sce.seger <- computeSumFactors(sce.sg, clusters=clusters)
sce.seger <- logNormCounts(sce.sg)

summary(sizeFactors(sce.seger))

png(filename = 'Norm_sp.png', height = 900, width = 900, family = "Times New Roman", pointsize = 24)
plot(librarySizeFactors(sce.sg), sizeFactors(sce.sg), pch=16,
    xlab="Library size factors", ylab="Deconvolution factors", log="xy")
dev.off()
```

```{r variance modelling, include=FALSE}
for.hvg <- sce.seger[,librarySizeFactors(altExp(sce.seger)) > 0 & sce.sg$Donor!="AZ"]
dec.sg <- modelGeneVarWithSpikes(for.hvg, "ERCC", block=for.hvg$Donor)
chosen.hvgs <- getTopHVGs(dec.sg, n=2000)

par(mfrow=c(3,3))

```

```{r dimensionality reduction, include=FALSE}
set.seed(100)
sce.sg <- runPCA(sce.sg, subset_row=chosen.hvgs, ncomponents=30)
sce.sg <- runTSNE(sce.sg, dimred="PCA")

```

```{r clustering, include=FALSE}
out_clust <- clusterRows(reducedDim(sce.sg, "PCA"), NNGraphParam(), full=TRUE)
snn.gr <- clust.out$objects$graph
colLabels(sce.sg) <- clust.out$clusters
```

```
tab <- table(Cluster=colLabels(sce.sg), Donor=sce.sg$donor)

png(filename = 'heatmap_sp.png', height = 900, width = 900, family = "Times New Roman", pointsize = 24)
dev.off()
```

```{r data integration, include=FALSE}

set.seed(1000)
applied <- fastMNN(sce.sg, batch=sce.sg$Donor, subset.row=chosen_hvgs)

set.seed(101)
applied <- runTSNE(applied, dimred="corrected")

colLabels(applied) <- clusterRows(reducedDim(applied, "applied"), NNGraphParam())

for_tab <- table(Cluster=colLabels(applied), Donor=applied$batch)


png(filename = 'tsne_sp.png', height = 900, width = 1500, family = "Times New Roman", pointsize = 24)

dev.off()
```