DIPLOMARBEIT

A Workflow for Single Cell RNA Sequencing Analysis with the Goal of a Robust and Comprehensible Clustering by Cell Populations

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Biomedical Engineering

eingereicht von

Dipl.-Ing. Stephan Reichl, BSc

Matrikelnummer 01029023

ausgeführt am Institut für Stochastik und Wirtschaftsmathematik
der Fakultät für Mathematik und Geoinformation der Technischen Universität Wien

Betreuung
Betreuer: Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Karl Grill

Wien, 28.11.2018

(Unterschrift Verfasser) (Unterschrift Betreuer)

Abstracts

Abstract

Next generation sequencing, transcribing parts of the DNA, underwent drastic developments in the last years. One of the newest developments enables researchers to extract gene expression data from single cells. One of these techniques, micro-droplet sequencing, managed to catapult single cell RNA (transcriptome) sequencing to the top of the field, because it provides high throughput and high accuracy sequencing information for a fraction of the costs. This new technology introduces researchers to a new era of single cell sequencing and thereby understanding mechanisms in biology on a single cell level. Different applications are the discovery of new cell types, identification of targets for drug development or the observation of biological phenomena on a cellular level, just to name a few.

With this new technology a new kind of data is generated. Therefore, new challenges emerged in the field of computational biology, for example very high dimensionality within large data sets, susceptibility to confounding factors, limitations in visualizing the data, high levels of noise and lacking confidence in clustering results. The literature recommends to address these challenges with procedures such as quality control, normalization, confounding factor analysis, dimensionality reduction and clustering. The major challenge is to determine the right methods and developing a best practice workflow for the rigorous analysis of such data.

This work focuses on the development, verification and validation of such a workflow, which addresses all the newly arisen challenges. The workflow is based on theoretical considerations of our previous work [Rei18] in this field. The main goal is to achieve a robust clustering by cell populations through the application of mostly automated and comprehensible methods. Furthermore, the discovery of potential sub populations inside of clusters or further downstream analyses are promoted by the information provided within the results of the workflow.

The workflow is verified with the help of simulated datasets, which were specifically designed to resemble single cell RNA sequencing datasets. Every dataset was parameterized to test for certain supported features of the workflow and its limits concerning artificially added challenges as noise or confounding factors.

In the end we validate the workflow through the analysis of a real publicly available dataset from the literature and by comparing the results and biological interpretation with discoveries made by the community.

Zusammenfassung

Die Sequenzierungsmethoden der zweiten Generation (engl. next generation sequencing), deren Ziel es ist Teile der DNA zu entschlüsseln, durchliefen in den letzten Jahren rapide Entwicklungen. Eine der neuesten Entwicklungen ermöglicht es Wissenschaftlern, die Genexpression von einzelnen Zellen zu quantifizieren. Eine dieser Technologien, "micro-droplet sequencing", schaffte es, die Einzelzell-RNA (= Transkriptom) Sequenzierung (engl. single cell RNA sequencing) an die Spitze dieses Forschungsgebietes zu katapultieren, da sie einen hohen Durchsatz und präzise Sequenzierungsinformationen für einen Bruchteil der Kosten ermöglicht.

Diese neue Technologie ermöglicht es Forschern eine neue Ära der Einzelzell-RNA Sequenzierung und damit die Erforschung von biologischen Mechanismen auf zellulärer Ebene einzuläuten. Verschiedene Anwendungen sind die Entdeckung neuer Zelltypen, die Identifikation von Angriffspunkten für die Medikamentenentwicklung oder die Beobachtung biologischer Phänomene auf zellulärer Ebene, um nur einige wenige zu nennen.

Durch die Anwendung dieser neuartigen Technologie wird eine noch nie da gewesene Art von Daten generiert. Daraus resultieren neue Herausforderungen auf dem Gebiet der Bioinformatik, zum Beispiel eine sehr hohe Dimensionalität in großen Datensätzen, Anfälligkeit für Störfaktoren, Einschränkungen bei Visualisierungen, hohe Rauschanteile und unzulängliche Ergebnisse in der Clusteranalyse. Die Literatur empfiehlt, diese Herausforderungen mit Verfahren wie Qualitätskontrolle, Normalisierung, Störfaktoranalyse, Dimensionsreduktion und Clusteranalyse zu bewältigen. Jedoch besteht die größte Herausforderung darin, die richtigen Methoden zu finden und einen Workflow für die rigorose Analyse solcher Daten zu entwickeln.

Diese Arbeit konzentriert sich auf die Entwicklung, Verifizierung und Validierung eines solchen Workflows, der alle neu entstandenen Herausforderungen adressiert. Der Workflow basiert auf theoretischen Überlegungen unserer vorherigen Arbeit [Rei18] in diesem Bereich. Hauptziel ist eine robuste und nachvollziehbare Clusterfindung anhand von Zellpopulationen durch den Einsatz von hauptsächlich automatisierten Methoden. Darüber hinaus wird die Entdeckung potenzieller Subpopulationen innerhalb von identifizierten Clustern oder nachfolgende Analysen durch die in den Ergebnissen des Workflows bereitgestellten Informationen gefördert.

Der Workflow wird mit Hilfe von simulierten Datensätzen verifiziert, die gezielt generiert wurden, um Einzelzell-RNA-Sequenzierungsdaten zu entsprechen. Jeder Datensatz wurde entsprechend parametrisiert, um bestimmte unterstützte Funktionen des Workflows und seine Grenzen hinsichtlich künstlich hinzugefügter Herausforderungen in der Form von Rauschen oder Störfaktoren zu testen.

Am Ende validieren wir den Workflow durch die Analyse eines realen öffentlich verfügbaren Datensatzes aus der Literatur und durch den Vergleich der Ergebnisse und biologischen Interpretation mit bereits vorhandenen Erkenntnissen.

Acknowledgments

First of all, I would like to thank Professor Karl Grill who again accepted the challenge to supervise me and help my endeavors by guiding me throughout the creation of this thesis. I am very grateful for the trust and the freedom, which was given to me concerning the topic, working schedules and our colloquia.

I thank as well Professor Felix Breitenecker, for his support during my bachelor studies and beyond. Especially for showing me, through his research group and summer schools, that mathematics can be found in every scientific area, how to interpret the real world in a mathematical way and the connection between the two.

Of course, I want to thank my alma mater, the TU Wien, for this master program in Biomedical Engineering with major in Mathematical and Computational Biology, because it was a totally different experience compared to my previous studies in Technical Mathematics. It combined a lot of different academic fields, involved hands on exercises and taught state of the art concepts. Never before was I engaged in such an interdisciplinary way. Even the little work experience I already gathered in research has shown me how valuable this solid and broad academic basis is.

I want to thank my former Basel colleagues Fabian Birzele, Javier Gayan, Nils Grabole, Tony Kam-Thong, Benjamin Loos and Andreas Roller, who showed me how much can be achieved during a nine-month internship through collaboration across a wide range of scientific fields such as bioinformatics, biostatistics, genomics, biochemistry and biology.

Having good friends among your fellow students who are taking the same way to graduation as oneself is a factor, which should not be underestimated. Therefore, I want to thank Mariella Gregorich, Eric Mörth and Niklas Lackner for making studying an exciting and fun activity.

A very warm thank you is dedicated to my parents Monika and Christoph and my aunt Irene for their understanding concerning my desire to keep on studying after having already graduated and their unwavering support throughout the years.

Special thanks go to my girlfriend Stefanie Schinnerl, who is one of the reasons I always try to do my best and for keeping me in check, especially when I dive deep into a new topic and forget the world around me.

Without all of you this would not have been possible, Thank You!

Yours truly,

Stephan

Contents

1. Motivation

> "Welcome those big, sticky, complicated problems. In them are your most powerful opportunities."
>
> Ralph Marston

Next generation sequencing (NGS), is the name of a group of deep, high-throughput, in-parallel DNA (deoxyribonucleic acid) sequencing technologies. The goal of (DNA) sequencing, is to determine the order of nucleobases (for DNA or RNA: Adenine, Cytosine, Guanine, Thymine or Uracil) within a DNA or RNA molecule of a given organism. For the determination of the exact order of nucleotides different techniques were developed.

Single cell RNA sequencing (scRNAseq) concerns itself with the quantification of gene expression through RNA sequencing on a cellular level. Due to novel technological developments, which make this approach more accurate, affordable and faster, scRNAseq experiences a new advent.

Applications for this kind of novel technologies are research in medicine, biology and chemistry or the life sciences in general and a lot of developments in the respective fields can be expected with the help of scRNAseq. Especially in the understanding of biological mechanisms, differentiation of disease characteristics on a cellular level, discovery of new cell types or drug development.

However, conducting experiments and generating data are only the first steps in the process of applying this technology on the just stated challenges and fields. Without robust and comprehensible analysis of the data, no further research can be done. It is critical to have high quality data and rigorous analysis in research for the development of new pharmaceutical components or discovering mechanisms in biology.

The problem at hand can be formulated as the absence of a comprehensible and holistic best practice approach for this kind of data, because firstly it comes with novel characteristics and secondly ground truths or reference data, which are needed to evaluate results, are scarce. Furthermore, there are not yet any widely spread or accepted standards concerning formats or methods. As a matter of fact, there are a lot of different methods out there for every aspect of analyzing scRNAseq data and every week a new "best practice" algorithm is published. This especially holds true for clustering algorithms, needed to detect populations within a sample for further analysis. Sadly, most of the novel approaches only work on certain datasets and not in general.

Therefore, we identified the need for a robust, comprehensible, semi-automated, modular and last but not least easy to use best practice for the analysis of scRNAseq data. The results should represent a solid basis for further exploratory analysis and research.

To have such a frame work, adhering to the stated goals, which deals with scRNAseq data, in a semi-automated fashion is essential for the work of computational biologists and exploratory data analysts. Therefore, a well documented and modular workflow is an ideal way to address this immanent need. Both attributes enable further development or modifications without rendering the current workflow useless in the future.

This work aims at presenting such a workflow for the rigorous analysis of scRNAseq data, which adheres to the goals and challenges described above. The methods, approaches and techniques applied within the workflow are mostly based on our theoretical considerations in a previous work [Rei18] on that matter.

To position this work and to give the reader a proper understanding for the current situation in this field, we start out with an introduction into the rather broad topic of scRNAseq. This will also entail a quick review on the immediate past in the field of next generation sequencing and also how scRNAseq differs from other technologies. After that, one particular technique of micro-droplet sequencing will be presented and we discuss the data, which is generated by that process. Naturally, this new kind of data comes with a lot of challenges, which are tackled by proper scRNAseq analysis.

The next part is dedicated to the construction and presentation of a semi-automated workflow for scRNAseq analysis, which is the main subject of this work. There, we will present concrete objectives concerning the functionalities, the general architecture and the components, called modules, of the workflow. This will be done by explaining a modules main objective, concrete implementation and results. Every module is accompanied by the respective part of an analysis of a simulated dataset, to make the explanations more comprehensible. At the end of this part we will discuss some areas of application and how the workflow is intended to be used.

The last part concerns itself with the verification and validation of the workflow. This is done with the help of eight simulated datasets to check for the designed and required functionalities and a real dataset for testing the workflow's real life capabilities and applicability.

Every part is supplied with illustrations, visualizations and tables to promote better understanding.

2. Introduction to Single Cell RNA Sequencing

> "It's only those who are persistent, and willing to study things deeply, who achieve the Master Work."
>
> Paulo Coelho, The Alchemist

In this introductory chapter, we start out by defining the term single cell RNA sequencing (scRNAseq) and what differentiates this method from previous RNA sequencing approaches. After that, we briefly touch upon biomedical applications and why this technology is very promising for scientists, patients and society in general. To make the approach more tangible, we describe one specific technology, which enables scRNAseq on a never seen before scale concerning throughput and accuracy. We finish the chapter by describing the kind of data generated by scRNAseq, the challenges resulting from it and how to deal with these novel challenges according to the literature.

2.1. Single Cell RNA Sequencing (scRNAseq)

Single cell RNA sequencing (in the following scRNAseq) is a next generation sequencing (NGS) technology, which enables scientist to determine the gene expression levels of a single cell by quantifying the RNA molecules present within the cell at a certain point in time. This can not only be done for one cell at a time, but for thousands of cells simultaneously.

To make the novelty more understandable, we can have a look at the previous state of the art technology for the determination of gene expression levels within a sample, namely bulk RNA sequencing (bulk-RNAseq), using a work on NGS by Kulski [Kul16]. The goal of this approach is to quantify the gene expression levels of a biological sample consisting of cells. For that, the cells within the sample are lysed and the RNA molecules are extracted with the help of some purification steps. To use standardized NGS methods, which are designed for sequencing strands of DNA, the RNA molecules are synthesized to complementary DNA (cDNA) by reverse transcription. This ensures that a more robust structure (double stranded vs. single stranded) carries the information and more importantly that standard sequencing approaches can be applied. The result of bulk-RNAseq then is the quantification of the number of RNA molecules relating to each gene within a sample or in other

words we get one value for each gene representing in total the gene expression levels of the whole sample.

This leads us directly to the main difference between bulk-RNAseq and scRNAseq. In scRNAseq we determine the number of RNA molecules relating to one gene for every gene within each cell of the sample. Therefore, the result is not one value per gene, but one value per gene and cell within a sample.

This is achieved through new technological developments in the sample preparation of the sequencing experiments. The cells need to be separated and "tagged" beforehand to make the quantification of gene expression on a cellular level even possible. As we will see, even the RNA molecules themselves have to be marked in a specific way to ensure that they can be traced back to their cell of origin. Therefore, the sample preparation in scRNAseq is a lot more sophisticated, compared to standard DNA or even bulk-RNAseq. For this process of "cell-capturing" different methods have been developed and they can be assigned to one of three categories: cell-sorting based, micro-well based and micro-droplet based. To illustrate the whole process of scRNAseq we will dedicate Chapter 2.3 to describe in detail a micro-droplet based approach, which yields high throughput and accurate results.

Due to the fact that these two RNAseq approaches quantify the gene expression on different levels (sample vs. cellular level) the data they generate is also different in nature. Instead of having only one value per gene, as we have seen in bulk-RNAseq, we encounter in scRNAseq a value for every gene in every cell. Therefore, data originating from scRNAseq experiments is different in structure, has an increased volume, higher complexity and is a lot sparser. The exact characterization of scRNAseq data, the resulting new challenges and how to deal with them by scRNAseq analysis are addressed in Chapter 2.4.

The most important thing concerning this novel technology are the new possibilities and potential applications in biomedical research. The next section of this part will deal with this aspect by reflecting on the current applications of NGS and the potential of scRNAseq in this context.

2.2. Biomedical Applications

The probably most famous project in DNA sequencing was the Human Genome Project, which was initiated in 1990 and finalized in 2013, taking 13 years to sequence and map all of the genes (= genome) of members of the human species, Homo sapiens, costing approximately $2.7 billion [Nat15]. Today, thanks to NGS it only takes one day to sequence up to 45 human genomes for one thousand dollars per genome [Ill17].

Due to the advances in NGS, increasing the throughput and quality and at the same time lowering the costs, it is already widely used not only in research but also in

clinical practice on patients. Three present applications of NGS can be found in cancer research, infectious disease analysis and drug discovery. In cancer research it is used to identify the best treatment for a patient by screening the tumor for certain genetic mutations. This can also be used to identify the cause of cancerogenesis and helps with the discovery of new mechanisms, which can be targeted by new treatment options. Infectious disease outbreaks of viral or bacterial nature can be stopped or contained by sequencing the respective vector and thereby finding a way to destroy it with a valid therapeutic agent or discover its origin. Drug discovery uses sequencing to understand the differences between healthy and diseased cells. This aids in the development of therapeutic agents, by identifying potential targets which inhibit the underlying mechanisms of the disease. Furthermore, NGS can be used in experiments to study the effect of potential treatments on diseased cells.

With the advent of new approaches in scRNAseq concerning the cell capturing techniques, advances in microfluidics and sequencing in general, it is now cheaper and more accurate than ever to conduct scRNAseq experiments. Therefore, this technology can be used in a wider field of applications. In general it can be said that one of the main goals is to detect masked or hidden cellular variation, which could help or lead to new discoveries and applications.

Future applications of scRNAseq or projects concerning the human health or genome, of which some are already in discussion or being implemented, are:

- The human cell atlas, a global effort to build a comprehensive atlas of all human cells with the help of scRNAseq [RTL+17].
- Newborn screening by sequencing the genome as soon as possible after birth to anticipate potential health risk factors [Nat16].
- Regular check ups at the doctor could be complemented by scRNAseq of certain tissues to asses potential health risks and intervene immediately.
- By analyzing samples on a cellular level new cell types are discovered and new biological mechanisms are found.
- Modeling of biological systems can be enhanced with deeper knowledge about the roles of single cells. Simulations on a cellular level could lead to more detailed and accurate insights.
- Patients in clinical trials could be stratified in better fitting groups to benefit from a particular drug or avoid adverse events.
- Diseases could be described more detailed on a cellular level and therefore more precise therapeutic agents or other more meaningful strategies can be applied.

This list could be even longer, but should only demonstrate the huge inherent potential of the novel advances in scRNAseq technology. In the next part we will describe one of various ways from the sample to the data generation.

2.3. scRNAseq with the 10x Genomics Solution

In this part we want to describe the whole process from sample preparation until the generation of the count matrix, which represents the starting point of our scRNAseq analysis workflow. For the crucial part, the preparation in the beginning, we decided to present one of the newest technologies in "micro-droplet" sequencing, which uses microfluidics for the capturing of single cells, provided by 10x Genomics [10x16b] and used by Zheng et al [ZTB+17]. The subsequent NGS sequencing protocol is presented with the help of materials from NGS technology providers ABM [ABM] and Illumina [Ill17].

From Cells to cDNA

We start out with the preparation of the sample of interest, consisting of thousands of cells, for the subsequent sequencing process. For that, we need the *10x Barcoded Gel Beads* and the *10x Chromium* machine, both provided by 10x Genomics, for the process of capturing and tagging the individual cells.

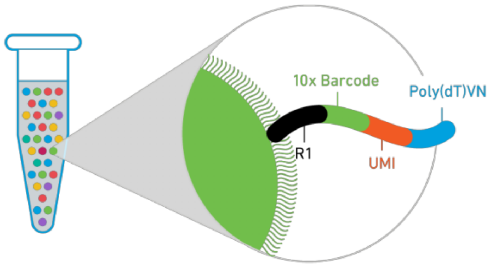

Figure 2.1.: A *10x Gel Bead*. Figure adopted from 10x Genomics.

A *10x Gel Bead* can be seen in Figure 2.1 and they host a multitude of oligonucleotides (= short DNA or RNA strands with different applications), which consist of 4 parts with different purposes. These parts are called R1, *10x Barcode*, *UMI* (Unique Molecular Identifier) and Poly(dT)VN. The first and the last part are used to catch the RNA molecules and in the process of reverse transcription (RT) as connectors or primers. The *10x Barcode* is a unique sequence within every *10x Gel Bead*. The *UMI* is a randomly generate sequence, which is different for every RNA molecule. Together, the *10x Barcode* and the *UMI*, are used to precisely trace back the RNA molecule to its cell of origin. Furthermore, it circumvents a common issue in NGS called amplification bias, which we will touch upon later in this chapter.

Having now established the makeup of the *10x Barcoded Gel Beads*, we will describe the outlined 10x Genomics *Chromium* workflow from Figure 2.2. First, the sample, oil and the *10x Barcoded Gel Beads* are loaded on a microfluidics chip, which is

especially designed for that purpose. Then, the 10x Genomics *Chromium* machine pumps the cells into microfluidic channels, which are only a little broader than the cells to avoid parallel transport. As we can see in the second step in Figure 2.2 at the first intersection of the barcode- and the cell/enzyme channel the barcodes and cells form (ideally) pairs and together they are put into micro droplets or GEMs (= Gel Bead in Emulsion) at the following intersection with the oil channel. Inside of each GEM lysis (= breaking down of the membrane of the cell) and barcoded reverse transcription of polyadenylated RNA from the single cells are performed. After removing the oil and some purification steps, this procedure yields barcoded cDNA molecules for the next step in the sequencing workflow.

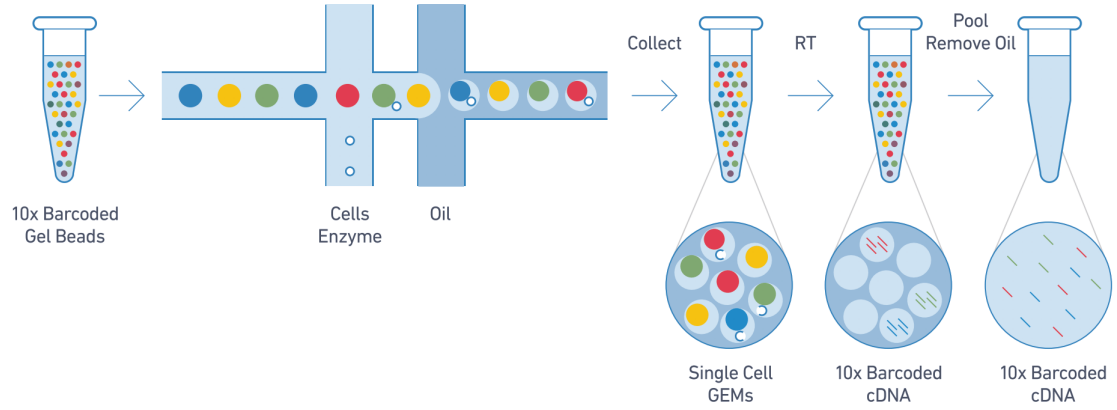

Figure 2.2.: Formation of GEMs, RT takes place inside each GEM, which is then pooled for cDNA amplification and library preparation. [10x16b]

Library Preparation & Cluster Generation

With the cDNA molecules from the last step we can use the standard procedure of NGS workflows as described by Illumina [Ill17].

The library preparation step consists of random fragmentation of the cDNA molecules and adapter ligation (= process of joining two nucleic acid fragments through the action of an enzyme) on both ends of the fragment as can be seen in Figure 2.3. Sometimes, these two steps are combined and called "tagmentation". The adapter-ligated fragments are then amplified with the help of polymerase chain reactions (PCR) and purified. This collection of cDNA fragments with adapters ligated to each end is called library.

Cluster Generation is crucial to ensure the detection of every nucleotide within each fragment. For that, the library is loaded into a flow cell where the adapter-ligated ends of the fragments are captured or hybridized by or to a lawn of surface-bound oligonucleotides complementary to the library adapters. To generate clusters of fragments, meaning a group of copies of each fragment close to the original one,

bridge amplification (a special form of amplification conducted with the help of a flow cell, which we will not explain here) is performed as illustrated in Figure 2.3. This results in distinct clonal clusters of each fragment on the flow cell, which are ready for the actual sequencing step.

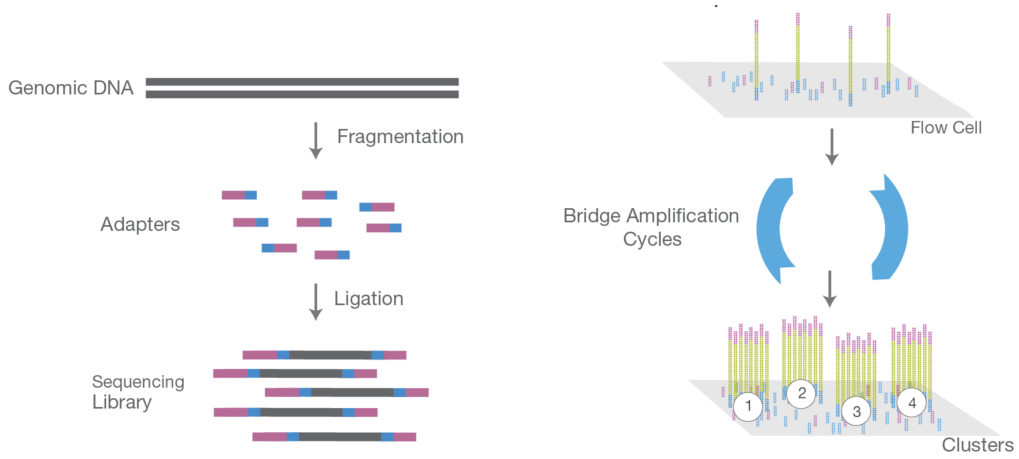

Figure 2.3.: NGS library preparation by fragmentation of DNA and ligation of adapters on both ends (left) and cluster generation for four fragments on a flow cell by bridge amplification (right). [Ill17]

Sequencing by Synthesis

Finally, we have reached the point where the actual sequencing takes place. There are a lot of different approaches on how to determine the exact sequence of nucleobases, but we will only describe sequencing by synthesis (SBS) as it is implemented by Illumina [Ill17].

As Figure 2.4 tries to illustrate, sequencing by synthesis, as most sequencing techniques, is a cyclic endeavor, which is repeated until the sequencing reaction is completed or the desired read length is reached. In every cycle another nucleobase per cluster is detected and this base-by-base sequencing yields highly accurate results.

In the following we will describe one cycle of the SBS process, as illustrated by the right part of Figure 2.4. Sequencing reagents, including fluorescently labeled nucleotides with a missing hydroxy group at the 3' end, are added. Only one complementary base per fragment is incorporated. Through the missing hydroxy group it is ensured that no further base can be incorporated during this sequencing cycle. During the incorporation the fluorescently labeled nucleotides emit light. This process is recorded for each cluster. By analyzing the wavelength and intensity of the emissions, the incorporated base per cluster can be determined. At last the fluorophores are detached and washed away and a hydroxy group at the 3' end is recreated to enable the next cycle.

The recorded and identified bases per cluster are then exported to an output file for the next step, the alignment to a reference genome.

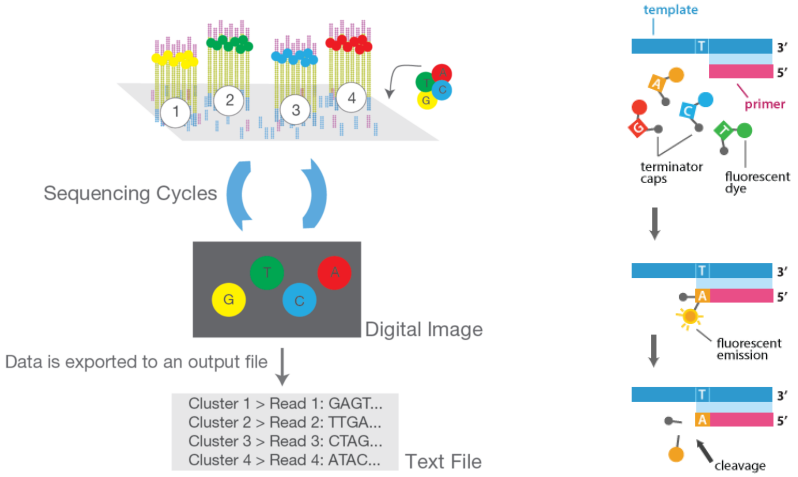

Figure 2.4.: NGS SBS overview from the flow cell to the data output [Ill17] (left) and a detailed view of one cycle performed on a single fragment [ABM] (right).

Data Alignment

The last step in the process of sequencing is mapping the sequenced reads to a known reference genome, which is a fully sequenced and mapped genome used exactly for that purpose. This is done with bioinformatics software. It is important to mention that the alignment process in the case of scRNAseq needs more sophisticated methods, due to the fact that we do not only want to count the number of RNA molecules present per gene, but also want to know from which cell the molecule originated from and avoid amplification biases to be introduced into the final results. To do all that 10x Genomics provides a set of analysis pipelines called *Cell Ranger* to process *Chromium* scRNAseq output to align reads, aggregate sequences and generate a gene by cell matrix, when supplied with a reference genome (for example hg19 for homo sapiens). In the end we have a gene by cell matrix, which values are the *UMI* counts relating to the respective gene and cell.

Differences to standard DNA or bulk-RNAseq can be found mostly in the beginning and the end of the sequencing process. DNA samples can directly start with the library preparation step, whereas bulk-RNAseq also needs to generate cDNA molecules for the standard NGS process. The data alignment step and especially the subsequent data analysis are very different for DNA, bulk-RNA and scRNA sequencing.

Amplification biases can be introduced through the different amplification steps within the standard NGS workflow. Some molecules amplify faster than others, therefore a bias can be introduced concerning the real number of RNA molecules. To circumvent this issue the unique tag of each RNA molecule, the *10x Barcode* combined with the *UMI*, ensures, additionally to the information on the cell of origin, that a single RNA molecule is only counted once. Thereby, it is guaranteed that one *UMI* count in the final gene by cell matrix represents exactly one detected RNA molecule within the respective cell and of the respective gene. This is done during the aggregation step of the data alignment part. Other methods are not able to account for that and have to deal with the effects on a computational level during the data analysis.

ERCC RNA spike-ins are a common set of external RNA controls, basically artificial genes in different concentrations. They have been developed by the External RNA Controls Consortium (ERCC) to control for sources of variability such as for example sample quality, level of cellularity and RNA yield, used platform or experiment operator. They can be added to the sample beforehand and used in the analysis to apply certain quality control measures.

We described the whole process of scRNAseq starting out with a sample until the point of the data generation and thereby conveyed background information on the biological and biochemical side of scRNAseq. We think it is crucial to understand how the data in question is generated, to process and analyze it in a rigorous way. The next part is dedicated to the novel characteristics of the generated data and the resulting challenges and goals in the analysis of it.

2.4. Data, Challenges & Analysis

After having explained how the data is generated through scRNAseq, this chapter will start out by describing the general features of the data. After that, we will address the challenges, resulting from this novel kind of data. At last, we will discuss how to master these challenges and adhere to stated goals with the help of scRNAseq analysis. This will give us a good foundation for the main part of this work, where we describe a workflow for scRNAseq analysis.

Data

The last chapter ended with the generation of a gene by cell matrix, which represents the basic structure of scRNAseq data. In this matrix every row represents a gene and every column a cell. The values are the quantified gene expression levels of the respective cell and gene. In the case of the 10x Genomics technique every value is a *UMI* count and relates directly to the number of RNA molecules expressing the

respective gene within the respective cell. This matrix is often referred to as expression matrix or count matrix and represents the starting point of every scRNAseq analysis.

Before we describe the data from a technical perspective, we wanted to mention the biological meaning of the data. The values of the matrix state which genes are expressed within the cell. Furthermore, they even represent the level of expression. This leads to the idea that we can also infer the number of proteins being synthesized with the help of the detected RNA molecules, which in turn can be used to determine the cell type, function or other protein related matters. Of course, we always have to keep in mind that the data represents the state of a cell at a certain point in time.

In the following list we will point out a few characteristics of the data.

- The major difference to DNA or bulk-RNAseq data lies in the structure. DNA sequencing data consists of the detected genes and bulk-RNAseq data describes the expression levels within the whole sample. However, scRNAseq data informs about the gene expression levels of each cell within the sample.
- The volume of the data represented by the two dimensions of the matrix depend on the used reference genome and the number of sequenced cells. The human reference genome hg19 consists of 32738 genes, relating to the same number of rows. Sample sizes vary according to the experimental setup, but reach from a few hundred to tens of thousands of cells.
- After the sequencing process, the matrix consists of zeros and positive values, because there is no such thing as negative expression. Through the course of the analysis and multiple manipulations, this can change.
- Zeros represent the majority of values, up to 95% of all values, within the expression matrix. That is why we talk about sparse matrices, when dealing with scRNAseq data. The zeros either represent the absence of RNA relating to the respective gene or measurement errors due to insufficient sequencing accuracy, sometimes called dropouts.

There is one additional kind of data, which is very important when dealing with scRNAseq data, namely annotation or meta data. That is additional information, most of the time cell specific, concerning the experimental setup. Common examples are the processing batch of the sample, the donor from which the sample was taken or the operator who conducted the sequencing. Annotation data is crucial in the analysis of scRNAseq data to unveil and remove unwanted sources of influence (batch effect) or explain observed behavior within the data (treatment effect or donor). If the annotation data describes unwanted or wanted variations always depends on the experiments objective.

According to these new characteristics of scRNAseq data and the technical background of the data generation, novel challenges emerge in the field of computational biology, when dealing with this data. At the same time new goals are formulated

due to the never seen before capabilities this technology provides. Both of these aspects will be discussed in the next part.

Challenges

We can recognize two different kinds of challenges in the context of scRNAseq, namely challenges occurring during the sequencing process and in the analysis of the data. The consequences of these challenges are inseparably connected. Most of them are inherent to the unprecedented level of sensitivity, which is present in the process of scRNAseq. Also the novel characteristics of scRNAseq data lead to challenges of their own.

We will briefly point out the most prominent challenges encountered, when confronted with scRNAseq in general and the data in particular.

- A few problems can occur during GEM formation, as for example the GEM only contains a *Gel Bead* or a cell. Another GEM related scenario entails so called doublets or multiplets, in this case a GEM contains two or even more cells. The problem at hand is that from this point on these two cells are not distinguishable from each other, due to the nature of the process. Multiplets are nearly impossible to detect during the analysis of the data, but quality control measures can be applied. Last but not least, it can happen that a GEM is formed with a Gel Bead and free floating RNA originating from already lysed or damaged cells. This leads to poor quality in the data, but usually can be easily removed. Therefore, dying or damaged cells should be rigorously removed beforehand.
- The previously described sparsity of the data represents another major challenge, because it artificially inflates the data, leading to a greater volume. Furthermore, a lot of mathematical and computational methods can not deal with this shear mass of zeros.
- The increasing volume (number of cells per sample) and dimensions (number of genes) of scRNAseq data require state of the art high performance computation clusters for storage and analysis. Additionally, new and fast methods for large high dimensional data sets have to be developed.
- The technical sensitivity inherent to the process of scRNAseq has also its downsides, namely the increased susceptibility to noise and confounding factors due to biological (contamination, storage, handling) and technical (anything else) influences.
- Cells of different cell types contain varying amounts of RNA and a lot of samples are very inhomogeneous when it comes to their cell type distribution. This makes the process of normalization, making different cells comparable to each other, very difficult. Additionally, cells of the same cell type can be

in different stages of their differentiation, cell cycle or apoptosis, leading to varying expression profiles although they are all of the same cell type.

- Data sets in scRNAseq vary quite a lot even when the same scRNAseq process was used to generate the data. Therefore, every experiment has to be processed and analyzed individually.

Most of these challenges directly translate to responsibilities, which have to be addressed by the process of scRNAseq analysis. The next part discusses potential goals of scRNAseq and how they can be achieved through proper analysis.

Goals & Analysis

In the last part of this introduction to scRNAseq, we want to present some goals and how scRNAseq analysis should be designed to adhere to them and the previously stated challenges. That means the analysis of scRNAseq data has to take care of the above outlined challenges and goals formulated by researchers and medicals alike.

We now present three examples of potential applications with concrete goals in scRNAseq.

- Identification and characterization of known or new cell types by differential gene expression analysis to better understand biological makeup of healthy or diseased tissue or identify biomarkers to describe certain pathologies. Furthermore, target and binding sites for pharmaceutical agents can be identified.
- Investigation of reactions and effects on a cellular level over the course of time, caused by therapeutic interventions, with the help of pseudotime analysis.
- Comparing cell type distributions among different species to identify similarities on a cellular level.

A good basis to reach these goals and answer all of the immanent questions is to determine a robust and comprehensible clustering by cell types through rigorous scRNAseq analysis. This of course raises the question: How does a best practice scRNAseq analysis workflow look like? This work deals exactly with answering this nontrivial question. Following the literature, as for example Stegle [STM15], we can learn from the analysis of previous sequencing techniques, such as bulk-RNAseq. They present general steps, which have to be taken to ensure rigorous analysis of scRNAseq data, without specifying best practice methods. With the goal of determining clustering by cell types, the following steps are recommended in the presented order.

1. Alignment and generation of counts
2. Quality control
3. Normalization
4. Confounding factor analysis

5. Cell type identification
6. Cell type characterization

Due to the fact that a best practice workflow should solve the above outline challenges and adhere to the defined goals, the content of these individual steps has to be developed and clearly defined.

In the following chapter, the main part of this work, we use these recommended steps and describe the development of a best practice scRNAseq analysis workflow with the goal of a robust and comprehensible clustering by cell populations.

3. The Semi-Automated Workflow for scRNAseq Analysis

> "Start where you are. Use what you have. Do what you can."
>
> Arthur Robert Ashe Jr.

In the main part of this work we describe in detail the development and implementation of a best practice scRNAseq analysis workflow with automated components. We will start out by presenting an overview of the whole workflow followed by an introduction to the modular architecture. After that, we will discuss every module of the workflow in the order in which the data is processed most of the time or at least initially. In every chapter we will describe the concrete implementation, used algorithms, given decision making support, automation aspects, visualizations and results.

The workflow is based on the theoretical work of Reichl [Rei18] and we use the recommendations given to select the concrete tools, methods and approaches, which adhere to the stated requirements. Furthermore, we applied the new approach on the validation of clustering results, presented in [Rei18], and describe its implementation in Chapter 3.8.

The goal of this workflow is to have an implementation in place for the rigorous analysis of scRNAseq data generated with the help of the 10x Genomics solution, which we have already described in the introduction in Chapter 2.3, or found in the literature or public databases. Therefore, the workflow should work for any kind of scRNAseq data.

Although this goal is very broad in nature we tried to build a workflow, which detects subtle differences within the data, accounts for confounding factors and yields robust and comprehensible clustering results by cell populations. The main purpose is to help the researcher (data scientist, data analyst, bioinformatician,...) in the analysis and decision making process, when trying to answer a scientific question.

To make it more concrete the following requirements for the workflow should be met

- robust - small changes in the data or parameters lead to similar results
- transparent - every action performed by the workflow is comprehensible
- semi-automated - after initial configuration no further interaction is necessary
- modular - skipping, replacing or optimizing certain steps is enabled

- the results represent a solid basis for further exploratory analysis
- best practice - the latest and validated methods are used
- easy to use - straight forward interaction with the workflow by the scientist

The whole workflow, therefore every module in this work, was implemented in *R* version 3.4 [R C17]. Every other *R* package and function mentioned in the following is written in *italic* and the packages are referenced in a glossary at the end of this work. Additionally, a short description and the used version is provided. We will not include or mention the respective dependencies. The backbone of this workflow is built on the three packages *scater*, *clusterExperiment* and *Seurat*. The main object which will be generated right in the beginning and manipulated along the way is from the *SingleCellExperiment* class.

On the infrastructural side, we wanted to mention that the computational effort was quite a challenge, which resulted in the requirement of a High Performance Computing (HPC) cluster and a lot of parallelization, which we will not discuss any further.

3.1. Workflow Overview & Module Architecture

A scRNAseq analysis workflow has to consist of certain steps or components, which can be realized in various ways and depths. As we have briefly shown in Chapter 2.4, where we presented the recommended steps of a general scRNAseq analysis, the sequence of these steps is by design and it is important that the data is processed in the given order. The following components, formulated as modules, constitute the backbone of the presented workflow and try to adhere to all the steps, which are required for a best practice scRNAseq analysis workflow according to the literature.

- **Quality Control** ensures that only relevant data, concerning the analytical question at hand, is used in the process of analyzing the data. Messy, wrong or incomplete data can distort the results in a significant way. To ensure a robust result, we have to get rid of such datapoints within the dataset.
- **Normalization** is required to ensure comparability between different cells. Without this step, certain datapoints could influence the analysis to a greater degree than others, although it was simply a technical artifact that lead to the imbalance or difference between them.
- **Confounding Factor Analysis** is one of the most important and novel challenges within the realm of scRNAseq analysis, because here every cell can be influenced by certain factors and we are not using the overall expression levels (as for example in bulk-RNAseq) in the analysis to derive insights. Furthermore, the used technologies are far more sensitive than before, after all they have to detect every single RNA molecule within each cell in the sample.

- **Clustering** by cell populations is one possible end point and for sure a very important milestone within the analysis of scRNAseq data. Determining the membership of one cell to a certain population has proven to be, at least, the basis for further investigations. Therefore, it is incredibly important that the user knows how the clustering results came to be and which methods were used. A lot of challenges are met in scRNAseq analysis due to the fact that the data landscape is quite heterogeneous and the algorithms in the literature usually work perfectly on some datasets, but do not reach reasonable results in other cases.
- **Analysis of Clustering Results** is formulated as a separate step (and module), because we want to adhere to the goal of robustness and transparency. As we just mentioned a lot of algorithms only deliver on certain types of data reasonable results. Therefore, we presented in [Rei18] an approach for the validation of clustering results in scRNAseq analysis and an aid in the decision making process, which is implemented in the following workflow.
- **Visualization** is an integral part of every data driven analysis, therefore we tried to integrate it whenever possible or sensible. This aids our comprehensibility aspirations as well as in the decision making processes.

After this short introduction to our approach to scRNAseq analysis or to be more precise the components of the scRNAseq analysis workflow, we will see that a modular architecture, which we have chosen for the implementation of the workflow, makes intuitively sense.

The workflow operates in a standardized manner and is split in six different modules, following the above outlined core components. Those modules are called and coordinated by a central main function. Every module consists of a computational and visualization part. The basic idea of this architecture is, that every module follows the same two main steps in a consecutive order, which are computation followed by the according visualization. In more detail, the steps for module M_i would be

1. **Load** the post-module object O_{i-1}, which was previously created and saved by module M_{i-1}, for the computational part.
2. Perform the module M_i specific **computations, manipulations and transformations** on the data.
3. **Save** the transformed or newly generated data in the post-module object O_i.
4. **Load** the just created post-module object O_i, for the visualization part.
5. Create and save relevant **plots and diagrams**, that explain which steps were taken, why and the effect it had on the data, without changing the data in any way.

All of these steps are controlled with the help of configuration files, which were specifically designed for that purpose and have to be created or filled by the user. Further explanations will be given in Chapter 3.2. Thereby, we ensure by design

that a lot of requirements we stated in the goal proposition before, as for example modularity, transparency, usability and the automation of certain steps within the analysis, are met. Furthermore, it enables the user to be immensely flexible and agile by skipping modules or pausing in between for parameter adjustments during the analysis process, as we will explain and discuss in Chapter 3.10.

Each module within the workflow will be described by one of the following chapters. Again, the order is by design and represents how the data should be processed, assuming one complete run (= all modules are needed) is intended to be performed. That means a dataset is passed from one module to the next. The workflow always starts where it left off concerning the module architecture, this enables a more interactive utilization. For better understanding we walk the reader in the last section of every chapter through the analysis of a simulated dataset.

Visualizations will always be given and explained in the corresponding module as every module has plots and diagrams as part of its output. These visualizations additionally represent the basis for the decision making process on how to proceed in the analysis. We will use one of the simulated datasets for the purpose of illustrating and demonstrating the concepts and functionality of the workflow. Further information on the datasets mentioned in this work can be found in Appendix A.

For the sake of readability and structure, every module chapter will be constructed in the same way, consisting of the following parts

- **Main Objective** of the module and significance in scRNAseq analysis
- **Implementation**
- **Results & Output** including visualizations, based on the simulated dataset *sim_2_2*. Details concerning the dataset can be found in Appendix A.

We will not concern ourselves with technical details in this work such as hardware requirements, naming conventions of files and command line implementations, because such infrastructural challenges are out of the scope of this work and would not generate any value in different systems. Nevertheless, we wanted to mention that all the files generated follow a strict nomenclature for better understanding and/or automatic steps within the workflow.

3.2. Configuration & Control

Before we dive into the detailed explanation of the separate modules, which make up the workflow, we wanted to mention how the workflow is controlled and how the user can interact with it. As we stated before, all the interactions are performed via configuration files, which have to be created by the user. These files consist of parameters, flags, file paths and filter criteria.

We will describe in the following the information, which can be conveyed via the configuration files for the purpose of understanding how the user can interact with

the workflow, which border conditions have to be given and for later reference, when the parameters are used in the example and result analyses. We grouped the parameters together according to their application type and module dependencies.

Workflow control parameters:

- Flags for every (computational) module and its visualization part, which indicate if the module should be executed or skipped. It can happen that data is already preprocessed in some way, for example normalized, and repeating this step is not advised. Turning off the visualization part of certain modules can be convenient if only the end result is of interest or the analysis is time critical. At last, there is a flag to indicate after which module to stop the workflow, which is intended to be used in an step-wise analysis, where decisions concerning module dependent parameters are made in-between modules.
- Paths to the source and annotation files of the data, configuration files and output directories.

Module-dependent configuration parameters:

- Import Module
  - input.type - three different common types of data are supported (options: 10x for 10x Genomics derived data, matrix for a plain matrix or mm for the sparse matrix format MatrixMarket)
  - MT.pattern - used to look for the given text pattern in gene names to mark them as mitochondrial
  - spikein.pattern - used to look for the given text pattern in gene names to mark them as spike-in “genes”
  - read10X_min_total_cell_counts - total UMI/count minimum for cells in case of 10x Genomics originated data (used during import module, because the whole list of used barcodes is saved by 10x Genomics, which does not necessarily resemble the number of cells in the sample)
- Quality Control Module (threshold parameters for filtering)
  - filter.cells.by.umi.min - total UMI/count minimum for cells
  - filter.cells.by.umi.max - total UMI/count maximum for cells
  - filter.cells.by.geneexpression - total gene count minimum for cells
  - filter.cells.by.MTpct - maximum percentage of mitochondrial genes allowed per cell
  - filter.cells.by.ERCCpct - maximum percentage of spike-in genes allowed per cell
  - filter.gene.by.cellexpression - number of cells, which have to "express" a gene (what "express" means is defined by the next parameter) to avoid the gene being filtered out

  - filter.gene.by.no.expression.per.cell - number of counts in a cell to call a gene "expressed" in that cell

- Normalization Module
  - norm.method - preferred normalization method (options: count, log2, log2_cpm, cpm, SF, LSF, UQ, DS, TMM)

- Confounding Factor Analysis & Cluster Module
  - pcs.use* - number of principal components, which should be used in the confounding factor analysis and clustering
  - cf.number.variable.features - number of variable genes used for dimension reduction or certain clustering algorithms
  - pc.threshold.pct - threshold for the automatic detection of the number of principal components to use (unit: percentage of additional variation explained per additional principal component) (default: 0.25)
  - rsq.threshold.pct - R^2 threshold for the influence of a confounding factor on a principal component or each other (default: 20)
  - cf.blacklist* - metadata name, which effect is not allowed to be removed (for example: treatment or diseased)
  - no.of.clusters* - number of assumed or anticipated clusters (often denoted by k)

- Cluster Analysis Module
  - biggest.cluster.max.pct - threshold for filtering clustering results. The maximum percentage of cells assigned to the biggest cluster allowed, in regard to all cells (default: 90)
  - min.no.of.clusters - minimum number of clusters to be accepted in a clustering result (default: 2)
  - not.clustered.pct - threshold for filtering clustering results. The maximum percentage of not clustered cells (default: 20)
  - minClusterSize.pct - minimum cluster size allowed in percentage, depending on the number of cells after the quality control module, which are used in the combination step of the cluster analysis module
  - n.top.clusterings - number of top clustering results to be used for the combination step of the cluster analysis module and to be presented in the end
  - combine.proportion.use - proportion parameter used for combining the computed top clustering results in the cluster analysis module (default: 0.51)

  - comparison* - metadata for which the ARI & NMI metrics will be calculated (for example a classification result from another publication or a previous clustering result)
- Visualization
  - ValueOfInterest* - metadata for which plots are generated (colored)
  - GenesToPlot* - real gene names of interest for plots, not necessarily the symbols, same names as used in the data source

In some cases we provided the default values, which were used in the later presented analyses if not differently stated. All the other parameters depend highly on the dataset at hand and have to be filled after initial inspection of the data or with the help of prior knowledge about the data. In the following the parameters and configurations are always stated in square brackets and italic, e.g. *[exampleConfig=TRUE]*.

Every parameter marked with an asterisk (*) is allowed to be skipped (by using NA), without impacting the functionality. Of course, more information leads to better results.

3.3. Import

Although we did not mention this module in the beginning of the chapter, we want to quickly state which formats are common and therefore supported by the workflow. Also, the process of importing the data and generating the appropriate objects will be explained. Furthermore, we show some visualizations, which are generated automatically after the import module, to inform the user and aid in the decision making process concerning the configuration of the workflow.

Main Objective

Load the data and its metadata (aka annotation) and prepare it for the processing with the workflow. Ideally, get already rid of unnecessary components and calculate metrics for an initial investigation of the data, on which upon decisions concerning the further analysis can be made.

Implementation

The import module locates and loads the following two basic data sources, with the help of the according path configurations.

- The sequencing data as a count/expression matrix. The three supported and most common types are

  - 10x Genomics originated data, which consists of three files: genes, barcodes/cells and the UMI/count matrix *[input.type=10X]*. Only in this case we apply a filter during the import process, due to the nature of the data. 10x Genomics supplies all the used barcodes (e.g. droplet with a barcode) as cells in the count matrix, although not every droplet necessarily captured a cell. Therefore, we already apply a pre-filter to not even import the "empty droplets" *[read10X_min_total_cell_counts]*.
  - a plain *gene* × *cell* count matrix in a text file *[input.type=matrix]*.
  - a count matrix in the MatrixMarket format (.mtx) for sparse matrices *[input.type=mm]*.
- Annotation or metadata file as *cell* × *metadata* matrix in a text file.

Due to the fact that the data can be very large but sparse, appropriate matrix formats, such as the class *dgCMatrix* from the package *Matrix* and functions from the package *DelayedMatrixStats*, are recommended. Otherwise the computational effort is too high for most systems, even HPC clusters.

After loading the expression and annotation data, mitochondrial genes and ERCC spike-ins get marked as "feature control" genes to be used in the quality control module, later on. This is only possible if the parameters *[MT.pattern]* and *[spikein.pattern]* are provided by the user.

The module creates a *SingleCellExperiment* object (a popular standard for scRNAseq data) with the following information:

- the original expression matrix
- metadata aka annotation information
- detected mitochondrial genes, by gene name
- detected ERCCs as spike-ins, by gene name

With the function *calculateQCMetrics* from the package *scater* certain metrics, based and depending on the supplied information, are calculated for the next step, namely the quality control module.

Results & Output

Although, not a lot of computations were performed so far, it is critical to inspect the data as soon as possible in the course of an analysis. Thereby, it is ensured that the following manipulations take place in a controlled manner and the procedures are comprehensible to the analyst. Furthermore, it is sometimes the case that the user does not possess enough information on the data at hand to define the appropriate parameters.

For that purpose and general good practice, the import module delivers the following visualizations as output, apart from the newly generated *SingleCellExperiment* object, which is ready to be analyzed by the subsequent modules in a semi-automated manner.

- Histograms to investigate the frequency of counts per cell, genes per cell and counts per gene.
- Density plots to look at the distribution of counts per cell, genes per cell and counts per gene.
- Kneeplots, which are practical plots for the investigation of trends, within the data. The y-axis describes a condition (for example UMI counts) and the x-axis the number of elements (for example cells), which fulfill that condition. By looking for the "knee" (= a steep gradient in the curve) an initial cut off value can be derived for example for the import parameter of 10x Genomics originated data. Furthermore, these plots convey a good sense for the range in which most elements can be found and complement histograms and density plots nicely. The workflow provides counts vs. cells, counts vs. genes and cells vs. genes kneeplots.

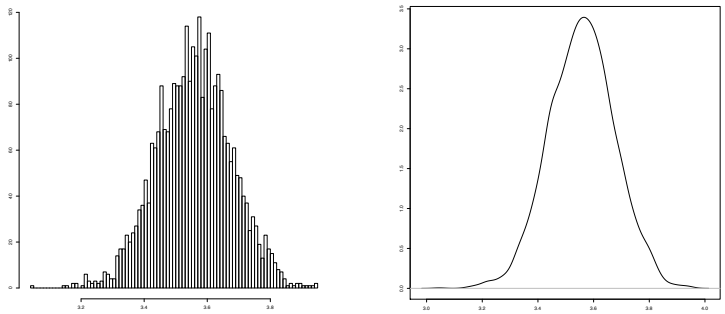

Figure 3.1.: Histogram of log_{10} UMI counts per cell (left) and density plot of log_{10} UMI counts per cell (right).

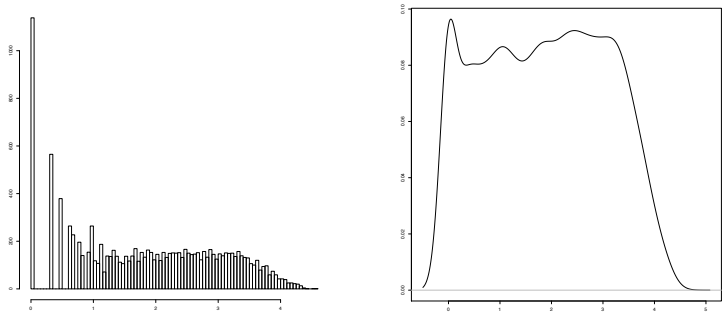

Figure 3.2.: Histogram of log_{10} UMI counts per gene (left) and density plot of log_{10} UMI counts per gene (right).

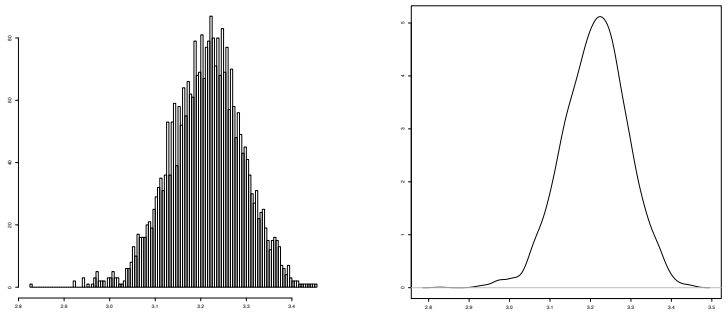

Figure 3.3.: Histogram of log_{10} genes per cell (left) and density plot of log_{10} genes per cell (right).

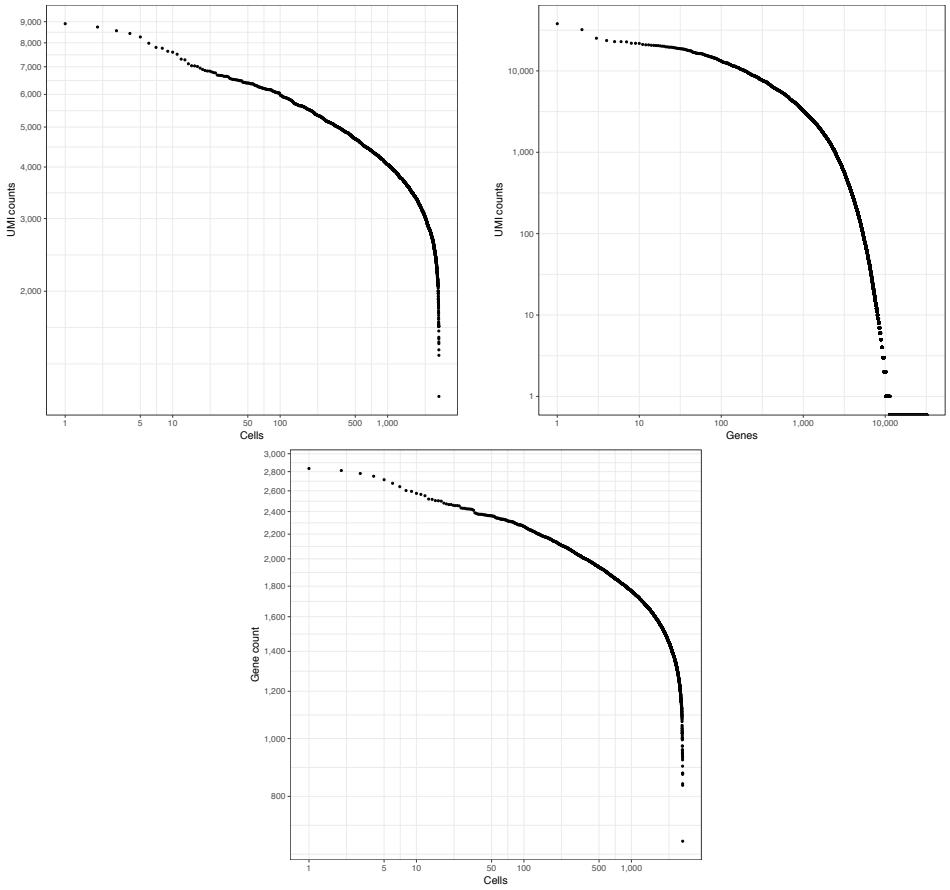

Figure 3.4.: Kneeplots of UMI counts vs. cells (top left), UMI counts vs. genes (top right) and genes vs. cells (bottom).

Looking at Figure 3.1, 3.2 and 3.3, we already see that the data is very neat, which of course is a consequence of the fact that the dataset is simulated. Especially in Figure 3.1 and 3.3 we do not detect any large group of outliers or obvious low-quality cells. In the search for irrelevant genes in Figure 3.2, we already detect a

lot of candidates, which should be filtered out by the next module. Investigating the kneeplots in Figure 3.4, we conclude that the initial import cut offs were either chosen wisely or the data simply does not contain any cells with less than 500 total UMI counts and 600 expressed genes (which is the case in this simulated dataset). Nevertheless, some outliers can be spotted.

3.4. Quality Control

Coming from the import module, nothing much happened so far with the dataset. This part tries to get rid of not relevant or messy datapoints and dimensions within the dataset.

Main Objective

In the first part we try to loose as many useless or damaged datapoints, e.g. cells, to ensure a more meaningful analysis. The second part gets rid of not-expressed and "not-detectable" (we will define this term shortly) dimensions, e.g. genes.

Implementation

Here, we use all the filter configuration parameters from Chapter 3.2, to get rid of not meaningful cells and genes concerning the analysis. These thresholds are difficult to define and depend on the sample at hand. It is recommended to inspect the output of the import module for the decision making process on these filter criteria. It does not suffice to simply filter out cells at both ends of the count spectrum, because considering for example highly proliferating tumor cells, which are expected to have a lot more RNA molecules compared to healthy cells, we would remove potentially interesting cells. This scenario represents legitimate very high total UMI count values in cells we definitely do not want to remove from the data. Therefore, we see how important the prior knowledge about the sample and experimental setup is.

We start out with the cells and filter each by

- a minimum number of total counts (= sum of all UMI counts of the cell) detected, to ensure that the datapoint really consists of a cell and not only an aggregation of free floating RNA or a damaged or dying cell with only a few functional RNA molecules left *[filter.cells.by.umi.min].*
- a maximum number of total counts, due to the possibility of doublets or multiplets, which would distort the data immensely *[filter.cells.by.umi.max].*
- a minimum number of expressed genes, determined by at least one UMI count per gene *[filter.cells.by.geneexpression].*

- the percentage of mitochondrial genes expressed by the cell, because very high mitochondrial activity indicates apoptosis or other cell cycle specific processes, which render the cell irrelevant for the analysis *[filter.cells.by.MTpct]*. This phenomenon was experimentally investigated by 10x Genomics in a technical note [10X17], which was based on older publications from Wang [Wan01] and Newmeyer and Ferguson-Miller [NFM03].
- the percentage of ERCC spike-in "genes" present, because high percentages indicate that it is probably an empty droplet, which captured only free floating RNA and the homogeneously distributed ERCC spike-ins *[filter.cells.by.ERCCpct]*.

After getting rid of low quality, damaged and therefore irrelevant cells, we look at the dimensions, e.g. genes, and try to determine which convey no information concerning the current dataset. Reducing dimensions in a rigorous manner simplifies the following analysis drastically and increases the influence of the remaining ones at the same time.

A gene is filtered out if

- it is generally not expressed in the whole dataset.
- its expression is found to be "not-detectable". We define the term "not-detectable" in this context by two parameters as follows: a gene is detectable if at least *[filter.gene.by.cellexpression]* cells contain at least *[filter.gene.by.no.expression.per.cell]* counts of that gene, otherwise it is marked as "not-detectable" and filtered out.
- it is an ERCC spike-in.

A few remarks on the quality control or filtering process we just described.

- The filters are strict, which means all filtered cells and genes are removed from the following analysis permanently.
- The order of first filtering cells and then genes is important, due to the fact that some genes can sometimes only be rendered meaningless, because of the previous removal of cells, which expressed them.
- We did not use any kind of outlier detection algorithms, although they are often used in the realm of computational biology. The reason for that, as we already discussed in [Rei18], lies in the fact that with scRNAseq data the usual approaches do not work well, mostly because of the high dimensionality. Additionally low-quality cells are often not recognized as outliers, because they form some kind of clusters in the feature space and are consequently treated as high-quality datapoints by most approaches. Therefore, we concluded that the use of cut offs, e.g. filters, instead of sophisticated outlier detection methods is recommended.

Results & Output

The result of this module is a clean dataset, which is ready for further manipulations. As we stated above the filters are strict and therefore all the cells or genes which did not pass the criteria are removed from the *SingleCellExperiment* object and not available any more from this point on.

To inform the analyst on the quality control measures, which were taken by the module, the following visualizations and metrics, in form of a report as a plain text file, summarize the whole process.

- Histograms which show UMI counts per cell and genes per cell before and after being processed by the quality control module, with vertical red lines indicating the cut offs.
- Scatter plots (each cell is a dot) to visualize the cell filtering process concerning the mitochondrial and ERCC spike-in thresholds (percentage vs. total number of expressed genes), again with the help of red lines to represent the threshold parameters.
- Scatter plots of the data, after being processed by the quality control module and log_2 transformed presented in a space reduced to two dimensions (for the purpose of visualization). This is achieved either by Principal Component Analysis (PCA) [Shl03], where the first two principal components are taken, or by t-distributed Stochastic Neighbor Embedding (t-SNE) [VH08]. The size of the dots indicate the number of expressed genes by the respective cell. The plots are generated by the according plot functions (*plotPCA* and *plotTSNE*) within the *scater* package.
- Filter metrics described in the report are expressed genes (number of genes expressed at least once in one cell), relevant genes and relevant cells, concerning the above described filtering criteria.
- Data metrics before and after the quality control module described in the report are cells, genes and the sparsity. We define sparsity as the proportion of zero values within the data.
- The metrics complement each other (for example the relevant cells correspond to the number of cells after being processed by the quality control module), but describe different matters, therefore the occasional redundancy is intentional.

	true	false
expressed genes	11343	20657
relevant cells	2995	5
relevant genes	6428	4915

Table 3.1.: Metrics summarizing the filtering process concerning cells and genes.

	before	after
cells	3000	2995
genes	32000	6428
sparsity	94.826%	74.545%

Table 3.2.: Metrics of the data before and after the quality control module.

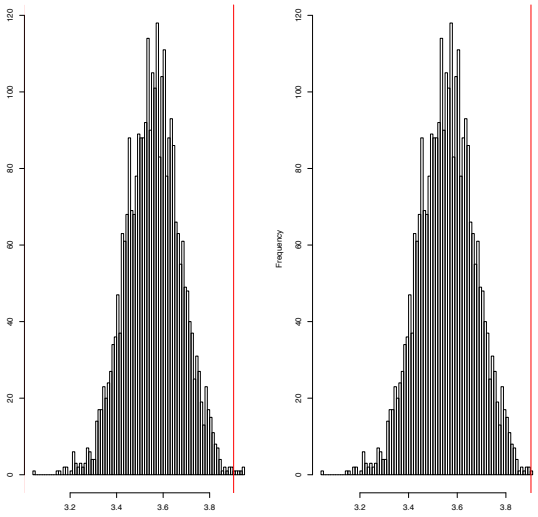

Figure 3.5.: Histograms of log_{10} UMI counts per cell before (left) and after (right) being processed by the quality control module. The *[filter.cells.by.umi.max]* cut off is indicated as a red vertical line.

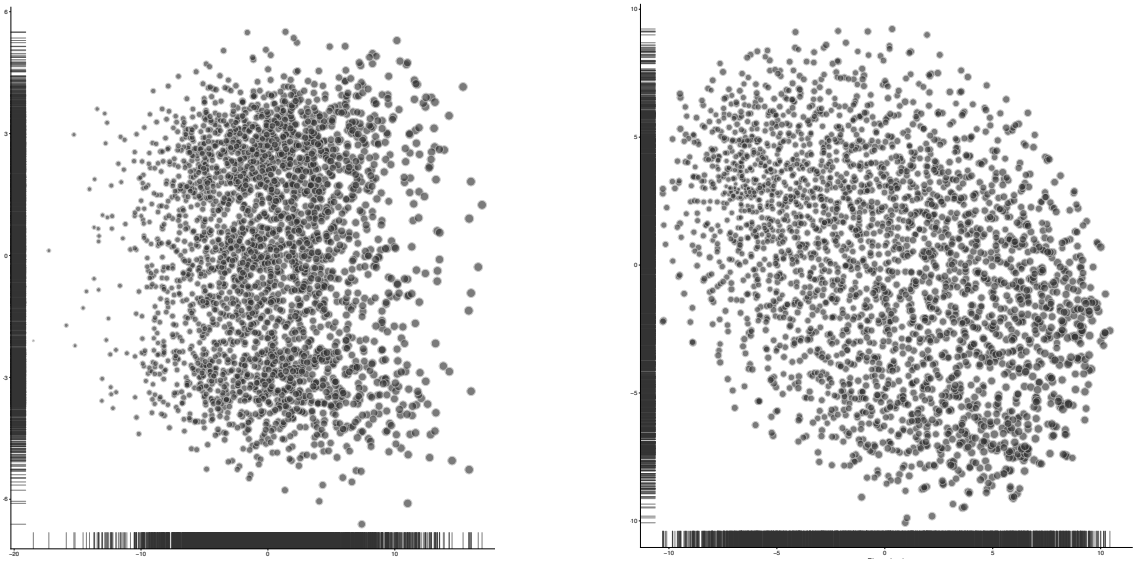

Figure 3.6.: PCA (left) and t-SNE (right) plot of the log_2 transformed data after being processed by the quality control module.

Having a look at Table 3.1 and 3.2, we can see that, although it is a simulated dataset, a lot of genes are not expressed, which is even more common in real datasets. Furthermore, we get rid of even more genes through the filtering process, which finally leads to a drastic reduction of dimensions. Cells are not filtered that heavily (only 5), because the simulation did not yield "damaged" cells or empty droplets as would be the case in a real dataset. The last important remark concerning the metrics is the decrease in sparsity of approximately 20% by the quality control module. As we have mentioned above, sparsity is a major challenge in scRNAseq analysis, and therefore critical to be reduced. We will see that real datasets sometimes have an even higher sparsity in the beginning.

Figure 3.5 nicely illustrates how the cut off parameters are used as thresholds for the filtering process. As we did not have to loose a lot of cells, the change in the histogram is not extreme, but the idea is apparent.

Having a first, simplified, look at the data in a space with reduced dimensions (two) in Figure 3.6, we can not make out any distinct clusters or populations. This can be explained by the rather high dispersion within the data, which we intentionally chose during the data simulation in Appendix A to challenge the capabilities of the workflow.

The plots describing the gene filtering process with the help of mitochondrial genes and ERCC spike-ins are not shown, because the simulated data does not contain any. We will see in Chapter 4.2 how important and impactful quality control with the help of mitochondrial genes can be. Furthermore, the histograms in Chapter 4.2 will present a greater discrepancy before and after the quality control module was performed.

3.5. Normalization

Having removed irrelevant data and dimensions, we need to account for the difference in sequencing depth of each cell.

Main Objective

It is important to ensure that certain cells do not distort, influence or even dominate the analysis to a greater degree than others, solely because of differences between cells caused by technicalities during the sequencing process. Therefore, we basically try to homogenize the data by different methods, without loosing real or relevant biological differences, which should steer the downstream analysis and drive the conclusions made by the analysis.

Implementation

There are a lot of different approaches to normalization of sequencing data. Ranging from standard methods, which are not very sophisticated, but established and accepted within the community, to rather recently developed approaches to tackle the novel challenges arising with the advent of scRNAseq data.

We will give an overview concerning the supported methods including a short description on the operating principles and its configuration shortcut. Mathematical analyses of the underlying theoretical aspects and a comparison of the different methods can be found in [Rei18]. We have also based the selection of normalization approaches on the insights presented in [Rei18].

Through the course of an analysis every method is applied and the result of each normalization is saved within a container of the *SingleCellExperiment* object. Thereby, the analyst can easily switch between different normalization results at a later time. The analyst chooses the normalization method, which should be used in the downstream analysis, with the parameter *[norm.method]* in the configuration file. If the analyst wants to use the raw quality controlled data in the downstream analysis, without loosing the flexibility of switching between different normalization methods at a later time (which would be the case if the whole module is skipped), the configuration *[norm.method=count]* can be used. This module can be extended by other normalization methods to ensure fast modifications and adaptions, when new methods are developed.

If not stated otherwise we used the function *normaliseExprs* from the package *scater* for the normalization, which already supports most of the presented methods.

Standard methods for normalization are the logarithm, counts per million and a combination of both. The application of the **logarithm** with the basis 2, including an initial offset (otherwise zeros would pose a problem), on the elements of the expression matrix is a wide spread first approach to normalization and was implemented manually *[norm.method=log2]*. The **counts per million** (CPM) approach, scaling the counts such that the total counts of each cell (aka library size) is one million, comes with the advantage of comparability across cells and therefore represents a good basis for the goal of normalization *[norm.method=cpm]*. We used the function *calculateCPM* from the *scater* package for that. The basic principle is to use the total number of counts of each cell as a size factor for the normalization of that cell. A lot of the following methods build on that principle and therefore make the results of different methods comparable. The **combination** of both, first calculate CPM values and then apply the logarithm, is also supported *[norm.method=log2_cpm]*.

Size Factor-, DESeq- or Relative Log Expression-normalization was developed for the analysis of bulk-RNAseq data. The idea is to determine a cell specific size

factor for the calculation of the effective library size (= an adjusted library size per cell). With the effective library size we apply the principle of CPM to get a comparable result *[norm.method=SF]*.

Upper Quartile normalization is based on the idea of using each cell's upper quartile (= 75th percentile) value as its size factor. Due to the fact that it was also developed with bulk-RNAseq data in mind the upper quartile is either zero or very close to zero when applied on scRNAseq data. Therefore, we parametrized the method with the 99th percentile as factor to determine the cell specific size factor *[norm.method=UQ]*. Again, we use the CPM approach and get normalized and comparable results.

Weighted Trimmed-Mean of M-values normalization as the name suggests, uses the weighted trimmed-mean of M-values (TMM) of each cell as size factor. The M-values are the gene-wise log-fold-changes. The trimming is performed by removing genes with M-values and absolute intensities outside of certain upper and lower percentage boundaries *[norm.method=TMM]*. The calculation is rather tedious, but tries to ensure robust results. With the obtained size factor per cell, the CPM approach results in normalized and comparable data.

Downsampling as normalization is the only representative of stochastic normalization methods in the workflow. The principle is to determine the smallest library size within the data to obtain a cell specific probability. With this probability and the actual count values of the cell we sample, with the help of a binomial distribution, the normalized expression values *[norm.method=DS]*. Unlike most of the other methods, it is not leveraging the CPM approach and therefore lacks the means to be compared to results of other normalization approaches in a meaningful manner. This method was directly implemented as we have just described.

Size Factor by Pooling Across Cells normalization also loosely called Lun Size Factor normalization due to the first authors name of the paper which presented it, tries to overcome the biggest novel challenges in normalization of scRNAseq data, namely sparsity and the assumption that most of the genes are not differentially expressed among different cells. The approach accounts for the sparsity of the matrix by pooling across cells with similar library size, summing them gene-wise up (to reduce the incidence of those problematic zeros) and calculating size factors for these pools *[norm.method=LSF]*. By solving a system of linear equations, the size factor for each individual cell can be determined. It is by far the most scRNAseq specific and at the same time sophisticated (and computational expensive) normalization method within the workflow. We implemented it with the help of the functions *quickCluster* for the pooling process and *computeSumFactors* for the size factor calculation, both from the package *scran*. Finally, we used the function *normalize*

from the package *scater* to normalize the values with the help of the calculated size factors.

Results & Output

After the data has been normalized by this module we are confronted with a decision: Which normalized dataset should be used for the downstream analysis? We want to state that we have not found a general best practice method, due to the variation across different datasets, based on biological and/or technical differences. Also it is important to mention that the most sophisticated method is not necessarily the best, because normalization naturally leads to distortion of the data to a certain extent and therefore simpler approaches, which yield similar normalization results, should be preferred. To help with this decision the workflow supplies the analyst with the following plots to inspect the results of the different normalization methods.

- A correlation plot describing the correlation of the matrices resulting from the application of different normalization methods with the help of miniaturized scatter plots and the spearman correlation coefficient in a comprehensive format based on only 20% randomly sampled cells to reduce computational effort. This format is provided by the function *corrplot* from the package *corrplot*.

- The Relative Log Expression (RLE) plots for each normalization result, which are a preferred method for the exploration of high dimensional data, are provided by a *scater* function called *plotRLE*. Therefore, they are the chosen means to easily compare normalization results. The goal is to find the method, which trims or homogenizes the data the most.

- The output of the *plotQC* function from the package *scater* visualizes the top ten quality control metrics (which were calculated in the import module) and metadata with the greatest influence on the data. These plots should be used in a qualitative way, or in other words to check if an improvement can be noticed by normalizing the data with different methods. The following module (Chapter 3.6) will be concerned with the quantification, analysis and possible removal of the influence from factors on/from the data. The plots are generated based on the raw data and two normalized datasets, namely by the combination of CPM and logarithm and the chosen one by the analyst (by the configuration parameter *[norm.method]*). Thereby, the user can compare the result when applying no, a standard or the preferred method.

As we already stated above, the workflow computes results for all the supported normalization methods to ensure a smooth analysis experience in case of a change in analysis strategy, which leads to the application of another normalization approach.

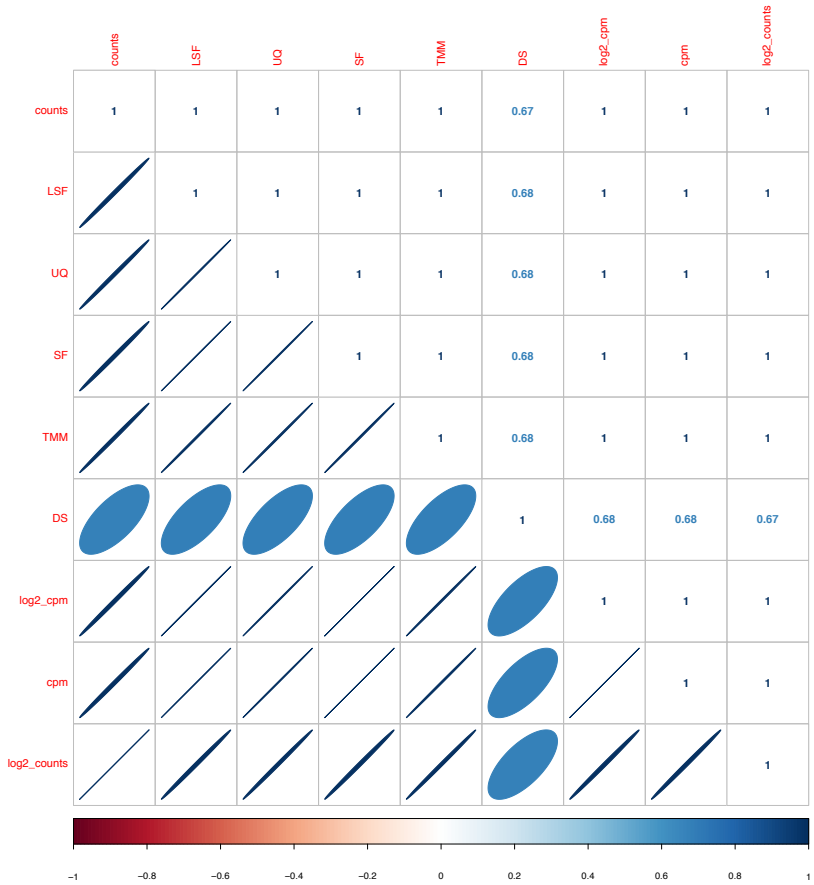

Figure 3.7.: Correlation plot for the comparison of the matrices generated by the supported normalization methods.

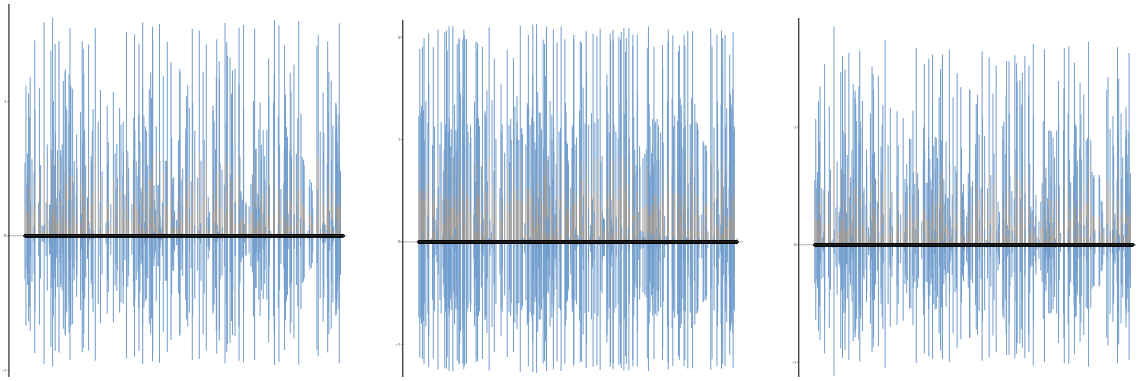

Figure 3.8.: RLE plots of the result matrices after applying the combination of CPM and logarithm (left) ranging from > -1 to < 2, the TMM method (middle) ranging from < -1 to > 2, and finally the LSF approach (right) ranging from > -1 to < 2 . The RLE values are on the y-axis and the cells on the x-axis.

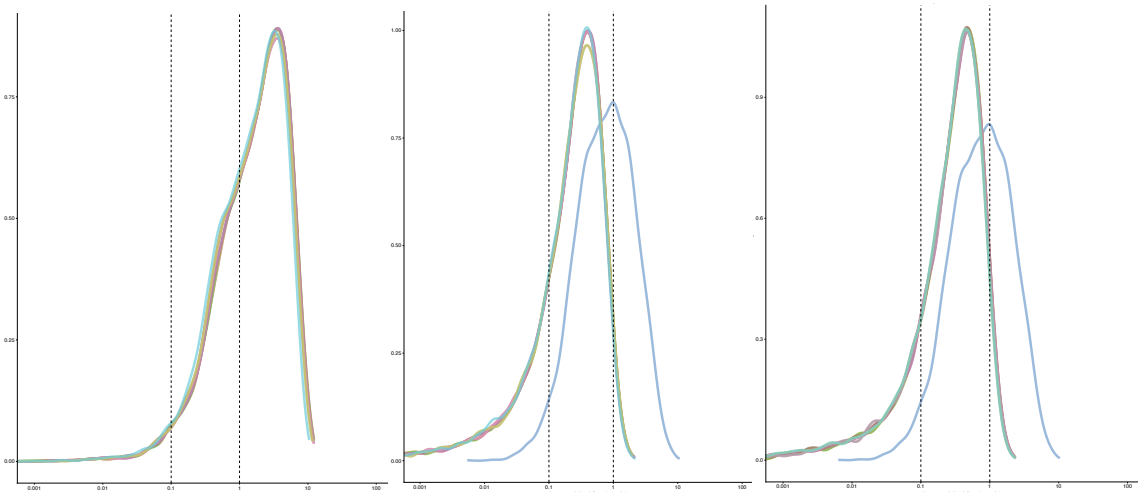

Figure 3.9.: QC plots of the top ten most influencing factors of the respective matrix with no (left), the combination of CPM and logarithm (middle) and the LSF normalization (right). Each color denotes a variable, not necessarily the same in every plot. The x-axis denotes the % of variance explained ($\log_{10}$−scale) and the y-axis the density. The dashed lines indicate 0.1% and 1%, respectively.

With the help of the **correlation plot** in Figure 3.7 we can check that the normalized as well as the raw data correlates (concerning the spearman rank correlation coefficient) nicely and therefore we can assume that we did not mess up the intrinsic structure of the data by the normalization process. Only the downsampling approach shows a significant difference compared to the other normalization results. This can be explained by the fact that it was not implemented according to the CPM approach, and therefore one can argue that there is no valid basis for a comparison in this case.

Looking at the **RLE plots** in Figure 3.8, we can immediately see that the LSF approach is the best option concerning the range of the RLE-values on the y-axis, compared to the results of the other two methods.

Last but not least, we can inspect the three generated **QC plots** in Figure 3.9. Here, we can see significant differences in the influence of certain metrics on the partly normalized data. It is quite obvious that both (the standard and the LSF) normalization approaches decrease the influence of certain metrics quite significantly by the factor of 10 compared to the raw data. Further analysis of the nature, origin and the influence on the data of these variables will be covered in the next Chapter.

Lets **summarize** the above stated observations: the correlation plot assures us that we did not change the data in an unintended way, the RLE plots indicate that the LSF approach homogenizes the data the most and the QC plots show that the standard and the more sophisticated normalization approach LSF are effective in reducing the influence of certain metrics on the data.

Therefore, we choose to use the LSF normalized data for the downstream analysis.

3.6. Confounding Factor Analysis

After the normalization we are able to start looking for influencing factors, which characterize the data. This is one of the most challenging and at the same time novel tasks coming with the unique nature of scRNAseq data. That is why we treat it as one of the centerpieces of the workflow.

Main Objective

This module tries to determine influencing factors and how much they describe the data, but also each other. Our aim is to decide which ones should not be allowed to describe the data to a certain degree, or in other words identify the confounding factors, and finally remove their influence. Otherwise, the influencing factors dominate the downstream analysis, although they do not represent valuable information or characteristics, which are the subject of the scientific question at hand.

Implementation

The module is manifold and we will start its description with a short theoretical introduction to confounding factor analysis, namely a definition and potential origins, which were assumed during the development of the workflow. After that, we work through the following steps of the module: detection & identification of influencing factors (not necessarily confounders), analysis of the found factors to determine their descriptive power concerning the data and each other, selection of the most powerful confounders and finally removal of their influence on the data.

The theoretical aspects and a discussion on the statistical and mathematical methods, which lead to this approach, can be found in [Rei18]. There, we explain the possibilities, complications and solutions, which lead to the following implementation.

Definition

The first thing we have to clarify and agree on is: What are potential confounding factors? We define them as variables, which explain more than a certain degree of the data's internal structure. These variables are basically attributes of the cells and can be seen as additional dimensions to the dataset (and not being genes) or simply metadata.

This leads us to the second big question: Where do they originate from? Some of them can be calculated directly from the data and others have to be provided by the analyst. The second possibility in this answer is very important to the

experimental setup, which generates the data, because if the variables in question are not documented properly there is no chance of removing or even quantifying their influence on the data.

Very common potential confounding factors of the first kind are the library size, number of expressed genes or variations in different formats as for example the influence of the top 500 expressed genes. Interesting examples of the second kind are processing batch, sample donor or experiment operator.

A last remark should be made about the different data types, which can occur. Mainly we have to differentiate between continuous and categorical variables. Often the influencing factors, which have to be supplied externally are categorical in nature (for example processing batch). The difference between those two data types has to be taken into account when choosing methods for the analysis of their influence on the data or each other.

Detection & Identification

We start by reducing the dimensions of the normalized dataset to decrease the computational effort. The strategy of our choosing is to perform Principal Component Analysis (PCA) and only use the most informative components. To obtain the number of the most informative components we implemented two ways. Either the analyst supplies the workflow with a fixed number via the configuration *[pcs.use]* or an automated approach is applied. The automated approach first conducts a PCA with the goal of keeping the first 50 principal components based on the approximate number of the most variable genes *[cf.number.variable.features]* provided by the analyst through inspection of the data (also with the help of the plots generated by this module). This is achieved through the function *runPCA* from the package *scater*. Next, we go through the calculated components from the most to the least informative and determine in each step how much additional variation was explained by the added principal component. If this value falls below a certain threshold *[pc.threshold.pct]*, then we have found the number of principal components to be used, excluding the last component. For the detection process only these most informative components are used in the following.

Now, we have to determine the influence of the supplied metadata on the data. This is done by calculating the coefficient of determination R^2 [Guj04] of a linear regression model [Gri00], computed by the function *lmFit* of the *limma* package, between every provided metadata variable and the chosen principal components. With the help of a threshold *[rsq.threshold.pct]*, we decide if the influence of a variable on the data should be investigated further. This is the case if one calculated R^2 value of a variable concerning any component is higher than or equal to the threshold. Otherwise, the variable's explanatory power is not of importance to the downstream analysis. By sorting them according to the most informative principal component, which is influenced the most by a variable, we get a ranked list of

potential confounders.

At last we apply a filter *[cf.blacklist]* on the remaining list of variables, because oftentimes the effect of certain metadata variables lie within the scope of the whole experiment and therefore will be subject to the analysis (for example treatment effect). Therefore, the influence of such variables on the data is not allowed to be removed. What we are left with is a ranked list of confounding factors.

Analysis

With a list of confounding factors we have to analyze their influence on each other. This is important, because if two factors, which influence each other heavily, are both removed, the effects potentially introduced and downstream results obtained are not foreseeable. Of course, a more rigorous analysis of their relationship and effect on the data may aid in the understanding of underlying mechanisms, but it is not always practicable in the course of an analysis. Therefore, we chose the pragmatic approach, which is automated, of determining their influence on each other and keeping, among a group of mutually influencing factors, only the most powerful one.

For the process of comparing the influence of confounding factors on each other, we have to consider the respective type of variable. Different variable types can not necessarily be compared by the same methods. The following list describes the solution to every possible combination of variable types. The first term is the type of the dependent variable and the second one the type of the explanatory or independent variable in the respective regression model.

- **Continuous versus Continuous** is modeled with the help of simple linear regression and the influence is determined by the standard R^2 as we have already seen before, when we investigated the influence of the potential confounding factors on the most informative principal components.
- **Continuous versus Categorical** is modeled with the help of multiple linear regression [Gri00]. This is achieved with the help of a design matrix, where every category of the categorical variable, but one, is seen as a single explanatory variable in the model. The influence is, again, determined by the standard R^2, with the help of the above outlined functions.
- **Categorical versus Continuous** can be seen as the inverse situation of the previous case and therefore we solve it by switching the positions of the variables, before modeling it in the same way and using the standard R^2 to determine the influence.
- **Categorical versus Categorical** can not be modeled with the help of standard linear regression models and therefore we have to build a multinomial logistic regression model [Cze12] and use McFadden's pseudo-R^2 [McF73] to determine the influence of the variables on each other. This was done with

the help of the function *multinom* from the package *nnet* to build multinomial logistic regression models and *logLik* from the package *stats* to determine the log-likelihood function of a model to obtain the McFadden pseudo-R^2 value.

One remark on the direction of influence. Only in the last case the "direction" of the influence is important, because McFadden's pseudo-R^2 is not-symmetric, in contrast to the standard R^2, which we use in the remaining scenarios.

The decision, if one factor influences another one in a significant way, is based on the same threshold *[rsq.threshold.pct]* as before, when we discussed the influence of potential confounding factors on principal components. In the end we are left with a list of the most impactful confounding factors, which do not influence each other in a significant way.

Removal

At last, but not least, we want to remove the influence of the remaining confounding factors from the data, because the influence has proven to be significant, unwanted and independent of each other. For that purpose we use multiple linear regression with all the confounding factors as explanatory variables and the dataset as dependent variable(s). In the case of categorical variables we split them up by category, as before. The residuals yielded by the multiple linear regression model are the components, which are independent of the explanatory variables and therefore, represent the corrected data, which we use in the following downstream analysis. To achieve that, we used the same method we have already used for the normalization in Chapter 3.5, namely *normaliseExprs* from the package *scater*, just with a different parametrization.

Results & Output

To navigate through this sophisticated process and comprehend the decisions made by the workflow, according to the supplied configurations, a multitude of visualizations and a text file naming the confounding factors, whose influence on the data has been removed, is provided. Of course, the *SingleCellExperiment* object was extended by a container with the corrected data on which the downstream analysis will be performed.

Following the above outlined process of confounding factor analysis the following outputs are supplied.

- A scree or elbow plot with the principal components ordered by their explanatory power on the x-axis and their respective eigenvalue on the y-axis. A red line indicates the number of components, which were taken, determined either through the provided parameter *[pcs.use]* or the automated approach explained above. It represents a common way to decide how many components

should be taken for the downstream analysis by looking for an "elbow" in the curve.

- Another way to investigate the PCA, is the plot with the principal components ordered by their explanatory power on the x-axis and the accumulated explained variance on the y-axis. This plot shows how much of the variance, within the data, is explained by the according number of principal components. Again, a red line indicates the chosen number for the analysis.

- For the analysis of the influence of potential confounding factors on the data or to be more precise on the previously chosen most informative principal components, a heatmap table is provided with the help of the function *aheatmap* from the package *NMF* with the factors as rows and the components as columns. The values are the respective R^2-value of the linear regression model. The whole table is reordered following a hierarchical clustering to show potential structures or groups. Every heatmap table in the following, also in the following modules, is generated by the same function as this one, if not stated otherwise.

- Analogue to the previous visualization, a heatmap table with the filtered confounding factors and their influence on each other is supplied. Here, it is important to take into account that the McFadden's pseudo-R^2 is not-symmetric and therefore the direction of the influence matters in the case of categorical versus categorical variables. In that case the value shows how much the respective variable in the row is influenced by the respective variable in the column. Again, the heatmap is hierarchically clustered to reveal potential intrinsic structures.

- The same quality control plot showing the influence of the top ten influencing factors, as we have already supplied in the course of the normalization module, is generated based on the confounding factor corrected matrix to visualize the (hopefully) drastic improvement.

- For every confounding factor, whose effect on the data was removed, a plot is generated visualizing the effect on the most influenced principal components before and after the removal, including the degree of influence determined by the R^2-value. These are either scatter plots, in the case of a continuous variable, or violin plots in the case of categorical variables. This feature is also provided by the function *plotQC* from *scater*.

- As already stated above, a text file is generated with the names of the most influential confounding factors, whose effect has been removed from the data.

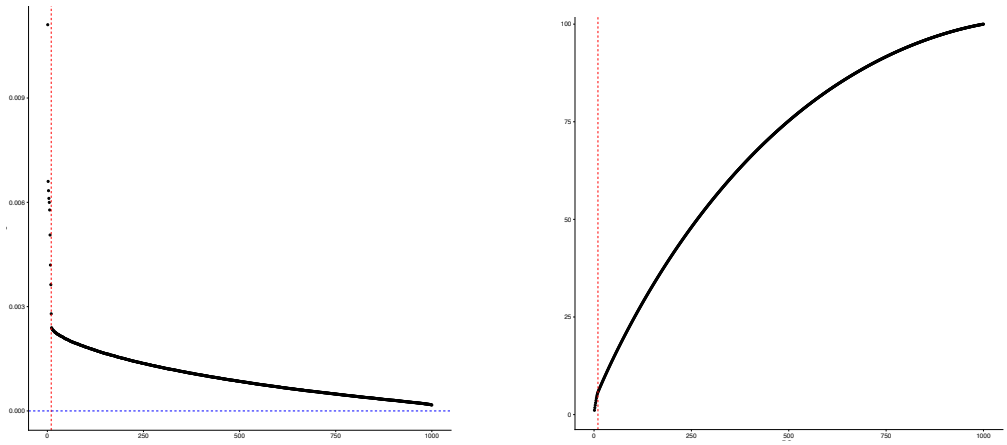

Figure 3.10.: Scree plot (left) and cumulative variance explained plot (right) ordered by the principal components' descriptive power within the data.

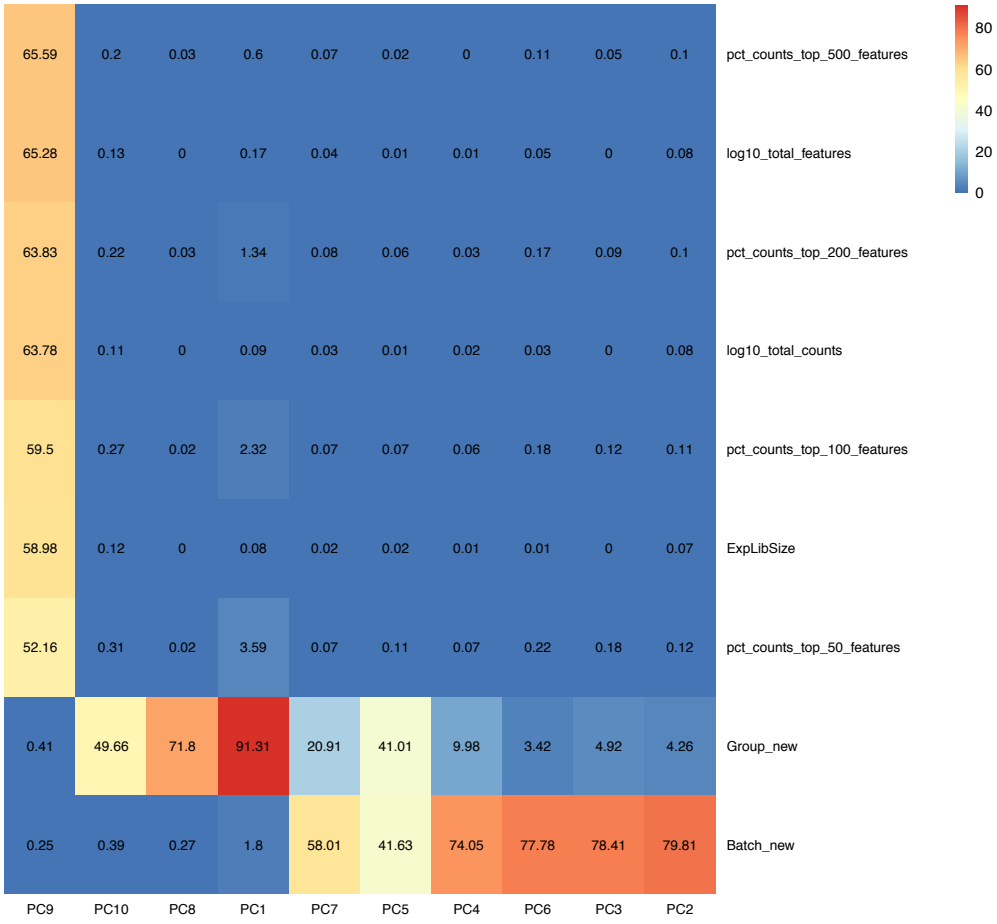

Figure 3.11.: Heatmap table displaying the influence in percentage, determined by the standard R^2, of metadata variables on the ten most informative principal components of the simulated dataset.

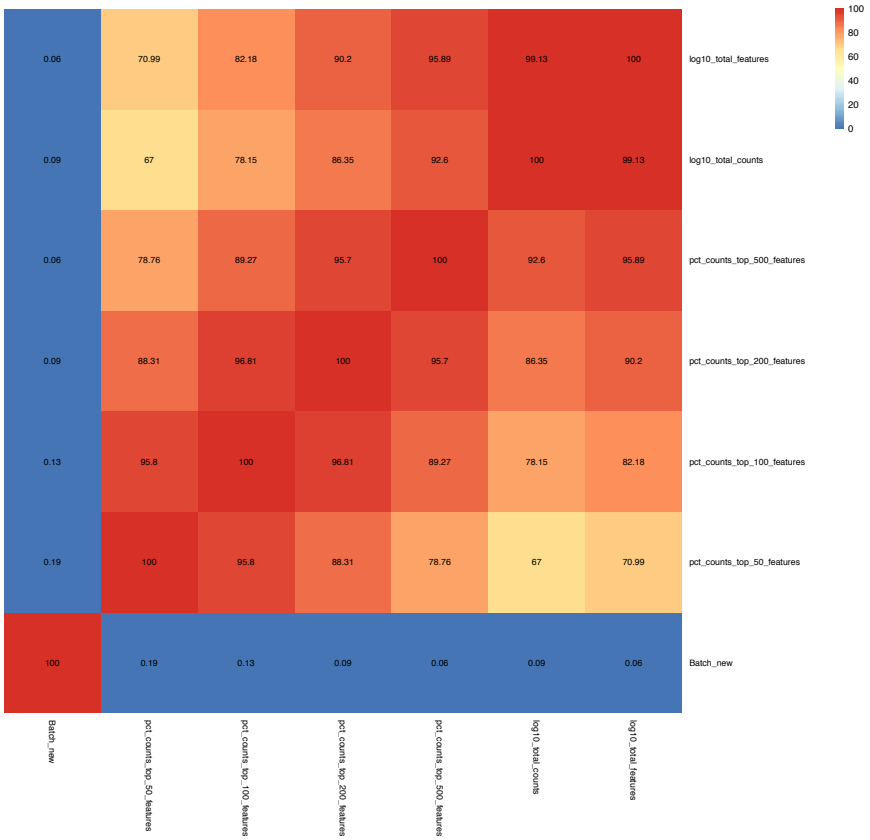

Figure 3.12.: Heatmap table displaying the influence in percentage, determined by the appropriate R^2, of metadata variables on each other.

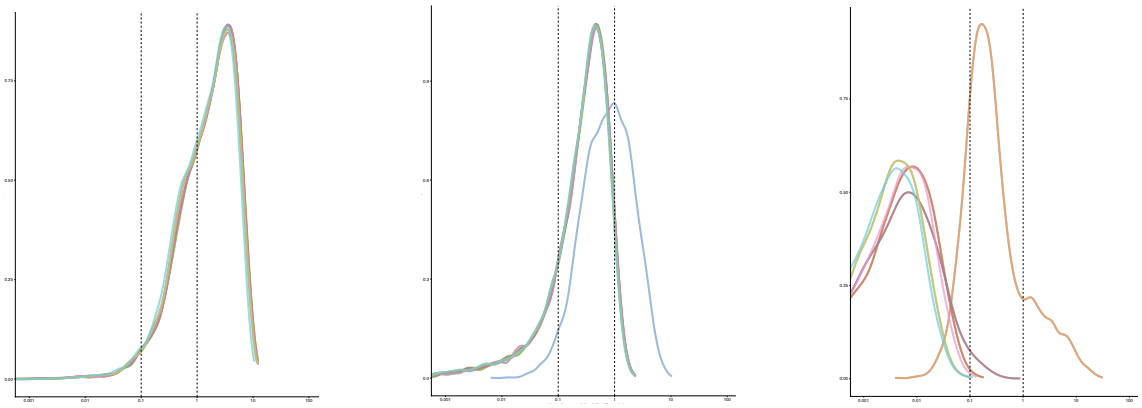

Figure 3.13.: QC plots of the top ten most influencing factors of the respective matrix with no normalization (left), the LSF normalization (middle) and the LSF normalization and removed confounding factors' influence (right). Each color denotes a variable, not necessarily the same in every plot. The x-axis denotes the % of variance explained ($\log_{10}$−scale) and the y-axis the density. The dashed lines indicate 0.1% and 1%, respectively.

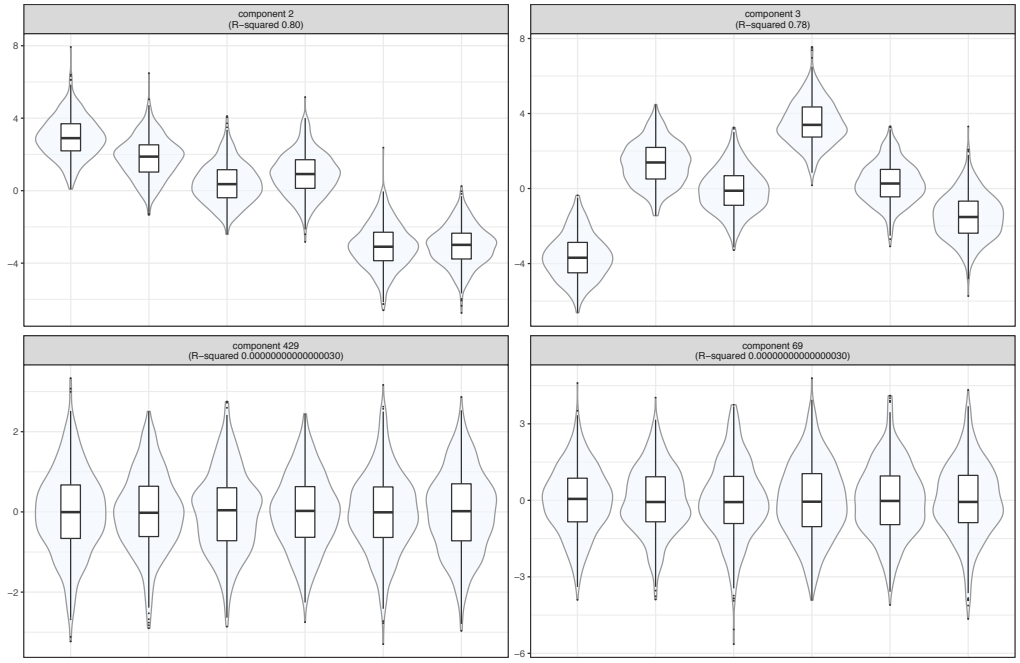

Figure 3.14.: Violin plots of the respective two most influenced principal components by the categorical variable Batch_new before (top) and after (bottom) the removal of the confounding factor's influence.

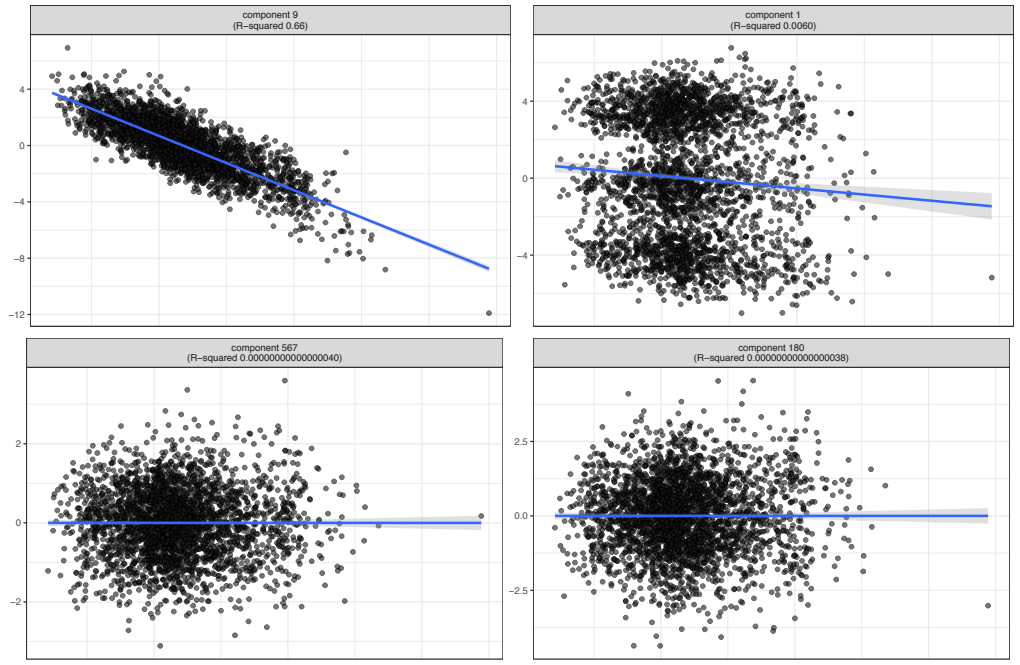

Figure 3.15.: Scatter plots of the respective two most influenced principal components by the continuous variable pct_counts_top_500_features before (top) and after (bottom) removal of the confounding factor's influence.

The scree plot in Figure 3.10 shows us clearly, that choosing the ten most informative components was a very good estimation concerning the parametrization. Although, we have to mention that the plot on the right in Figure 3.10 reminds us how low the accumulated variance by those components really is.

The heatmap in Figure 3.11 is clearly indicating, that the factor Group_new, which is the true clustering as set up during the simulation of the data, and Batch_new are explaining a great proportion of the variance within the data, represented by the first ten principal components. Furthermore, we can observe that all the other variables influence the ninth component. This may indicate that they all describe a similar aspect of the data.

Figure 3.12 consists of those factors, which influence at least one principal component in a significant way and were not on the blacklist. Additionally, it confirms the previously stated suspicion, that all the remaining confounding factors, apart from Batch_new, describe similar aspects in the data and therefore influence each other.

Following these observations the workflow concluded to remove the effect of the variables Batch_new and pct_counts_top_500_features. The latter one was the most descriptive factor compared to the other confounders, which it was influencing or influenced by.

For the comparison between pre and post confounding factor analysis module results, we have a look at Figure 3.13, where we see a drastic improvement from the raw to the normalized and finally the confounding factor corrected data. The only influencing factor left in the confounding factor corrected data plot is the variable Group_new, whose effect was not allowed to be removed.

Finally, Figure 3.14 and Figure 3.15 visualize how the respective confounder describes the data before and after it's effect on the data was removed. Complementing the visualization by violin or scatter plots respectively, we can compare two metrics. First, the component which was influenced the most before and after. In the case of Batch_new we have component 2 versus component 429 and in the case of variable pct_counts_top_500_features component 9 versus component 567. As if this was not already enough, the influence, denoted by the R^2-value, on the respective most influenced component decreased by a magnitude of 10^5, which basically means that there is nearly no influence left.

Although we are looking at a simulated dataset we can try to comment on the nature of the found and removed effect of the confounding factors, as we would with real data. The confounder Batch_new was intentionally introduced during the simulation of the dataset to test the confounding factor analysis capabilities of the workflow, therefore any further interpretation is rather pointless. The second confounding factor, pct_counts_top_500_features, on the other hand suggests that the 500 most expressed genes by count values described the data in a very significant way and would have dominated the following downstream analysis, although there are still 5928 other genes expressed in the data, which can aid in the characterization of each cell in a unique way.

3.7. Clustering

After having removed the influences, which may have driven the clustering process, but should not be allowed to characterize the data in a significant way, the confounding factors, we are ready to start with the final part of the workflow, namely the clustering of cells by populations.

This part represents a possible endpoint in a scRNAseq analysis and at the same time it is the starting point for every potential downstream analysis, which is based on a sound clustering by cell populations. Examples of such are the analysis of a specific cell type, pseudo time analysis to look for temporal changes or differential gene expression analysis to characterize (sub-)populations by their gene expression.

Main Objective

To overcome the major challenge of currently not having a best practice method or algorithm for the clustering of scRNAseq data, our approach is to implement every valid method (valid means, that the method has proven to work at least on a subset of scRNAseq datasets) and try to generate a lot of potentially meaningful results. These results are then analyzed by the subsequent module to reach a final result, or in our case a favorite and a consensus clustering.

Therefore, the main goal of this module is to produce a variety of potentially valid clustering results for further analysis, but also to capture every aspect or intrinsic structure within the data. This is ensured by looking at it from every possible angle, represented by different algorithms or approaches and respective parameter variations.

Implementation

We will start with some preparatory measures and then state the supported algorithms with short remarks on their underlying mechanisms.

Preparation

First, the new data, with removed effects of confounding factors, is subject to dimensionality reduction, because it speeds up the clustering process and is often a prerequisite for certain approaches. This is achieved by principal component analysis (PCA) as before in the confounding factor analysis module. For further elaboration on the implementation we refer at this point to the previous Chapter 3.6. As input for the clustering algorithms we only use the most informative principal components, determined by the above mentioned approach. Additionally, we use the previous and following two components as input for every algorithm. This increases the variation

of the input and thereby generates more, potentially valid, results, which help in obtaining a robust final clustering result.

A big challenge in cluster analysis is the number of clusters k to look for. Classic approaches, as for example k-means algorithms, need a k as input parameter. We implemented two ways to circumvent that issue to a certain degree.

The user can supply the number of anticipated clusters k directly with the help of the configuration parameter *[no.of.clusters]*. This is especially useful when domain knowledge is available, because then a confident assumption on the potential number of populations in a specific tissue type can be made. The provided assumption is then transformed into an interval by the addition and subtraction of two, which represent the boundaries of an integer interval for k, to enable a broader parameter spectrum to search in.

The other approach we chose to implement is algorithmic in nature and complements the parameter approach. Here, the expected number of clusters k is determined by leveraging the results of a clustering algorithm (*Seurat*'s SNN-Cliq implementation, more details below), which does not need a k as input parameter. After successful application of that clustering approach, k is obtained by determining the median across the number of clusters within all clustering results generated by this algorithm, and rounding it down to get an integer value. Again, this value is transformed into an integer interval by the same strategy as before described in the parameter approach.

Both approaches are used, as long as the parameter is supplied by the analyst, and the resulting intervals are unified to get a final list of k's as input parameters.

Having now a dimensionality reduced dataset and a list of different numbers of clusters k to look for as input parameters, we are ready for the clustering process.

Algorithms

We will outline all the approaches or algorithms, their implementation and involved packages, which partly state that they were designed for and successful in clustering certain scRNAseq datasets, in the following. To cover a large amount of reasonable possibilities we always computed a lot of different results per approach with the help of parameter variations. In every case we varied the number of clusters k to look for and the number of principal components as input, to cover a wider spectrum of potentially valid outcomes and avoid missing hidden structures. Furthermore, approach specific parameters were always varied in a reasonable way, for the same reason. We will not elaborate any further on the specific parametrization as it is beyond the scope of this work.

The following clustering algorithms, approaches or implementations are supported.

- *Seurat*'s shared nearest neighbor (SNN)-Cliq implementation to find clusters within preprocessed data, without the need of the parameter k.

- *ClusterExperiment* (CE) package provides a variety of different algorithms for the clustering of scRNAseq data, which can be divided in two categories.
  - TypeK algorithms, which need a number of clusters k to look for: pam, clara, kmeans, hierarchicalK and spectral.
  - Type01 algorithms, which are hierarchical in nature: hierarchical01 and tight.
- Single Cell Consensus Clustering (*SC3* package), a k-means based consensus approach with a standard implementation and one that is used instead in the case of more than 10000 cells, to reduce the computational effort. The second approach trains a support vector machine (SVM) on a clustered (by the standard approach) subset of the data and then classifies the rest by means of the trained SVM. We always force the second approach in addition to the standard approach, even if there are less than 10000 cells to cluster, to get additional variation in the results.
- The *pcaReduce* package provides a hierarchical clustering approach applied on datasets, which dimensions were reduced by the means of PCA. Two variations are presented, one based on sampling and the other one on merging in the course of the hierarchical clustering process.
- t-SNE combined with k-means implementation, where the dataset is dimensionality reduced by the means of t-SNE (*runTSNE* function from the *scater* package) resulting in a two dimensional map on which a standard *kmeans* implementation from the *stats* package is applied.
- Clustering through imputation and dimensionality reduction, provided by the *CIDR* package.

In total there are 14 different approaches implemented, which all yield more than one result depending on the number and variation of parameters. If one approach or specific parameter configuration does not yield a result it is simply skipped. This module can be easily extended by other clustering algorithms, whose results feed directly into the workflow without any additional effort.

Results & Output

The result of this module is a vast amount of different clustering results, due to the variety in algorithms and parameters. All of these clusterings are saved within the *SingleCellExperiment* object and two new objects, which are generated in the course of the module. These new objects are derived from the *Seurat* and the *ClusterExperiment* package, respectively, and contain all the same information, just fitted to the structure of the corresponding class. The workflow supports from this point on all three objects and updates them on any changes, which occur in the course of the subsequent module, because they have unique analysis and visualization capabilities we want to use.

The module does not generate any plots or visualizations, which is an exception to the basic principle of the module architecture of computation and visualization in an alternating manner. The reason is that this module is purely intended to cluster the data by different means to get the biggest possible set of clustering results for a more sophisticated analysis in the next module. Additionally, we have to mention that a lot of results are simply meaningless and therefore the effort to visualize them in different ways would be in vain. After the next module, the cluster analysis, we will present comprehensive visualizations concerning the best clustering results and how they came to be.

The only output, besides the above mentioned result objects, is a text file consisting of a list of all the clustering results generated. The names follow a standardized nomenclature and include the algorithm name, parameters, their values and the number of determined clusters. Depending on the algorithm, which generated the result, more or less parameters are stated in the name. One exemplary name would be *CEkmeans_pcs12_k08_k8_cluster*, which states the algorithm (*CEkmeans* = kmeans from *ClusterExperiment*), the number of used principal components used (*pcs12* = 12), the input parameter for the number of clusters to look for (*k08* = 8) and the number of determined clusters (*k8* = 8).

At this point we are not able to say anything about the analysis of the simulated dataset *sim_2_2* except that there were 216 clustering results obtained by the cluster module, without any information on their quality or validity. This represents the basis for the next module, which is the rigorous and comprehensible analysis of clustering results.

3.8. Cluster Analysis

The situation we see ourselves confronted with is having a huge amount of clustering results of unknown quality and validity with our final goal of clustering the data by cell populations. It is the result of applying a large amount of different clustering approaches with varying parameters on the data. This situation represents a big challenge and unsolved issue in general, but especially in scRNAseq analysis, due to the lack of consensus or best practice concerning the clustering process. This observation was confirmed in a recent publication by Freytag et al [FLNB17], which called this situation actually a “Cluster Headache”. Due to the fact that we tackle a critical, but very challenging issue with this module, we view it as the second centerpiece of the workflow.

Main Objective

To reach the goal of having a final best clustering result by cell populations, we implemented an approach, already theoretically presented in [Rei18]. The presented

approach did not concentrate on developing a new clustering algorithm (there are already quite a lot), which works only on certain groups of datasets, but rather on a broader framework to deal with that issue. Following that approach we quantify the quality of each clustering result by certain indices, which check for different aspects of the clusterings. With the help of these quantifications we sort the list of all results by the quality derived from the indices. Thereby, we can get the best x clustering results from the top of that list. Additionally, to these top clustering results, we implemented the computation of a consensus clustering, to deliver a robust and comprehensible solution. By implementing this approach, we hope to be able to circumvent the above stated issue in cluster analysis within scRNAseq analysis and solve it in a comprehensible and general way.

Implementation

Before we start with the rigorous analysis of the clustering results, we try to get rid of obvious low quality results, by filtering according to certain rules. After that, we strictly follow the outlined theoretical considerations of [Rei18] in the implementation process, which consists of

- the computation of various cluster indices per result,
- the determination of an order by quality by formulating and solving a multi-criteria decision making problem,
- the combination of the best clustering results to obtain a consensus and
- an optional comparison to a provided solution or clustering result.

Cleanup

To get rid of low quality results we filter them by the following rules and remove them from any further analysis.

- Clustering results which are empty (meaning that the algorithm could not determine any cluster) or contain less than 2 clusters, hence only one cluster was determined, are obviously of no further use. If more clusters are anticipated due to specific domain knowledge concerning the sample, the parameter *[min.no.of.clusters]* can be adjusted to the analysts needs.
- If clusterings contain more than 20% of not-clustered cells we dismiss the result, because it indicates that the approach in combination with the parameters did not seem capture the inherent structure within the data, which fit to most of the cells. Some not-clustered cells usually occur, due to not previously filtered multiplets or low quality cells. In case of expected low quality cells during the clustering process this threshold can be customized with the parameter *[not.clustered.pct]*.

- If the largest cluster consists of more than 90% of all cells within the data, it indicates significant under-clustering and we remove the result. Of course it is not entirely impossible that one cell population dominates the sample in such away. In that case the parameter *[biggest.cluster.max.pct]* can be adjusted accordingly.
- Clustering results with more clusters than twice the maximum of the anticipated number indicated by the input variable k, are also removed, because this indicates over-clustering. On the other hand, maybe there is some truth to the over-clustering concerning some cell populations and thereby indicating the existence of sub-populations. This could be investigated in an additional run of the workflow on certain clusters of a clustering result as outlined in Chapter 3.10.

Usually these measures lead to a significant reduction in the number of potentially valid clustering results. Thereby, the computational burden of the following steps is decreased drastically. All of the following computations and outputs only concern and describe these remaining clustering results.

Quality Measures

To quantify the quality of the clustering results in an objective manner we calculate for each result six different measures. Whereby we try to evaluate different aspects within the structure of the clustering results as for example compactness, density, distance between clusters and the statistical information of a clustering result concerning the data. For the first four indices, in the following list, this is achieved with the help of the function *intCriteria* from the package *clusterCrit*. The information criteria approach was implemented manually by leveraging the *lm* function within the *stats* package.

The following cluster indices were chosen for that purpose. A theoretical presentation and analysis of their properties and behavior can be found in [Rei18].

- Silhouette Index
- Calinski-Harabasz Index
- Tau Index
- C Index
- Akaike- & Bayesian information criteria approach, weighted with the help of the most informative principal components, developed and presented in [Rei18]

This set of approaches for the quantification of clustering result quality can be extended or edited at this point at any time to ensure flexibility, compatibility and adaptability concerning novel developments in the field of predicting more accurately the quality of clustering results derived from scRNAseq data.

Decision Making & Combination

Now, we have filtered out the low quality clustering results according to parametrized, specific rules. Then, we calculated for each of the remaining results six different quality measures. The problem at hand is to find a way to determine a ranking by all of the above mentioned quality measures. This is achieved by formulating it as a multicriteria decision making problem, where the quality measures represent the criteria and the clustering results the alternatives, between which a decision has to be made.

For that purpose we chose to use TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) [PZKS12], realized in the function *TOPSISLinear* (or as an alternative *TOPSISVector*) within the package *MCDM*. By applying TOPSIS on the clustering results and their according measures (equally weighted) we obtained a ranked list by quality. This approach directly yields the best clustering result concerning the quality measures at the top of that ranked list. It represents one potential endpoint of the analysis and we will call it in the following **favorite clustering**.

As we already argued in [Rei18], we do not want to loose the potential discoveries or detected internal structures within the remaining clustering results, especially derived from the top ranked ones, which were not chosen as the favorite clustering. Therefore, we additionally implemented the presented approach on leveraging the large amount of alternative clustering results to obtain a complementary solution.

Before, we present the final step of the implementation, the consensus, we have to remove duplicates within the ranked list of clustering results to avoid potential biases concerning an algorithm or number of clusters k. In other words, we remove every clustering result which was obtained by the same algorithm and yielded the same number of clusters k. Of course, we always remove the duplicates with lower quality.

Finally, we choose the top *[n.top.clustering]* clustering results and combine them to a consensus clustering. This is achieved by the function *combineMany* from the package *clusterExperiment* with a parameter for the smallest accepted cluster size *[minClusterSize.pct]* and a proportion parameter *[combine.proportion.use]*. The proportion parameter basically denotes the proportion of times in which a cell has to be in a certain cluster to be allocated to that cluster in the final consensus clustering result. We will call the result of this procedure the **combined clustering**. Thereby, we used the knowledge within the top *[n.top.clustering]* clustering results to obtain an additional and robust final clustering result.

Comparison (optional)

An optional part of this module is the comparison of the final results with an externally supplied clustering result in the form of metadata. This is for example a

previous clustering result, a published classification result for that dataset or some kind of rare ground truth (as in our case with the simulated dataset).

For the comparison we used two different measures, namely the Adjusted Rand Index (ARI) [JD88] provided by the function *adjustedRandIndex* from the package *mclust* and the Normalized Mutual Information (NMI) [WW07] provided by the function *NMI* from the package *NMI*. The metadata variable with which the comparison should take place has to be supplied via the parameter *[comparison*]*. These two metrics are measures for comparing two clustering results and yield values between 0 and 1, where 0 means absolutely no overlap and 1 that the two clustering results are exactly the same. Therefore, they can be interpreted as similarity in percentage, covering different aspects of a comparison. Of course, the comparison is performed with both final results, the favorite and the combined clustering result.

Results & Output

This module generates by far the most results and outputs. This can be easily explained by the fact that it is the end point of the analysis and tries to present the results in the most comprehensible way possible.

First and foremost the module results in the three objects (*SingleCellExperiment*, *ClusterExperiment* and *Seurat*) generated in the course of the clustering module, filled only with the remaining clustering results (after filtering them) and the newly generated combined clustering. We have not mentioned the favorite clustering separately, because it was not added by the module, only chosen.

To describe the results in more detail and a readable format the following files are generated by the module.

- A text file with a list of the names of the *[n.top.clustering]* best clustering results ranked by their quality according to TOPSIS.
- An object containing only the calculated clustering indices in the form of a data frame for separate inspection or additional analysis.
- An optional text file containing the calculated ARI and NMI values of the combined and favorite clustering compared to the supplied metadata variable *[comparison*]*.

Additionally, to make every step comprehensible and present the results in various ways, the following visualizations are provided.

- Hierarchically clustered heatmaps of the ARI and NMI values comparing every clustering result with each other. This is an easy way to check for similarities and differences among the clustering results.
- Barplots of the final and the best clustering results, to visualize how the combined clustering came to be and how similar the favorite clustering is compared to the other best clustering results. They are generated by the function

plotClusters from the package *clusterExperiment*. Every row represents a clustering result and every column a cell. The different clusters within a result are indicated by colors. Optional a barplot is generated containing the combined and favorite clustering and values of interest (which have to be provided as metadata), if they were supplied by the parameter *[ValueOfInterest]*, as rows. Examples of such values are processing batch or ground truth.

- Hierarchically clustered consensus matrix of the best clustering results as a heatmap, which is a cell by cell matrix describing the proportion of times that one cell is in the same cluster as another cell in respect to the best clustering results. It is provided by the *consensusmap* function of the *NMF* package. It can also be seen as the visualization of a similarity matrix, whose entries represent the similarity of two cells, if clusters are interpreted as groups of cells with similar features.
- Histograms of the number of found clusters within the clustering results.
- A heatmap table, similar to the one from the confounding factor analysis module in Chapter 3.6, describing the influence of the metadata on the best clustering results by the R^2-values of the corresponding linear regression model. This plot is useful to identify potential confounders in hindsight or to discover factors that drove the clustering process and to which degree.
- Scatter plots, describing the correlation between every clustering index to help understand their dependencies among each other. This is an important aspect to consider, when using the above described approach for ranking the clustering results by quality. The colored points represent the best clustering results.
- t-SNE maps of the data, colored by the combined and favorite clustering result are generated with the *TSNEPlot* function of the package *Seurat*. Furthermore, if provided by the parameters *[ValueOfInterest]* or *[GenesToPlot]*, t-SNE maps colored by values of interest or specific genes (with *Seurat's* function *FeaturePlot*) are provided.
- PCA plots of the data, colored by the combined and favorite clustering result, are obtained with the help of *Seurat's* function *PCAPlot*.

Before interpreting the visualizations and bringing the analysis of the simulated dataset to its conclusion, we wanted to mention that we only present a certain selection of plots and results due to lack of space and usefulness to the reader.

ground truth versus	ARI	NMI
favorite clustering	0.959	0.937
combined clustering	0.998	0.99

Table 3.3.: ARI and NMI values of the comparison with the ground truth, which was defined during the simulation of the data.

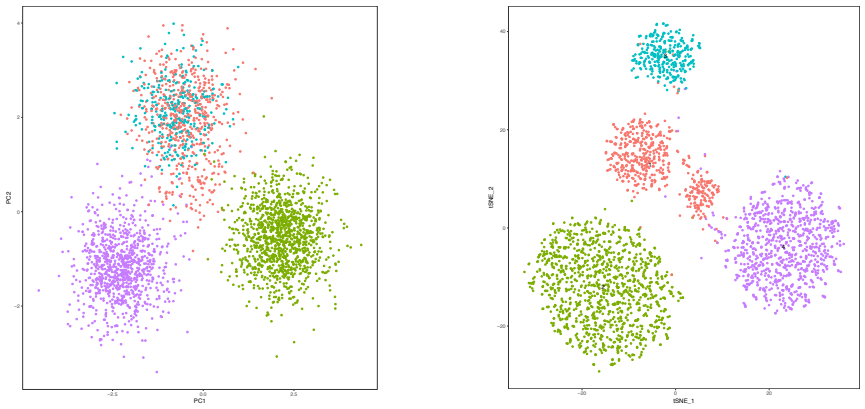

Figure 3.16.: PCA plot (left) and t-SNE map (right) of the processed data colored by the favorite clustering result, determined by the described approach.

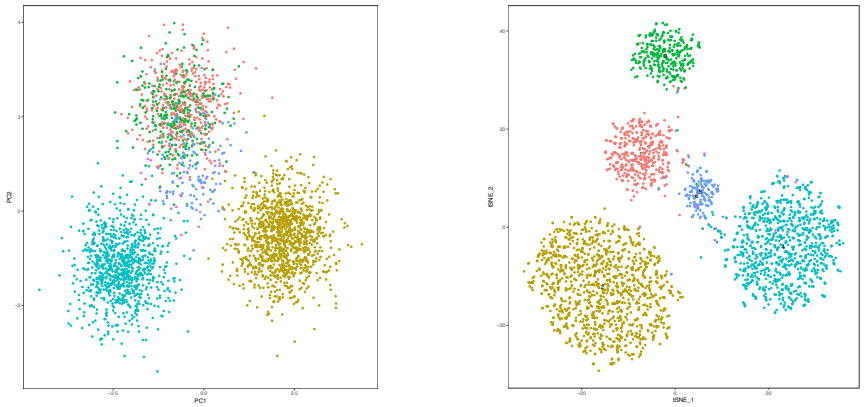

Figure 3.17.: PCA plot (left) and t-SNE map (right) of the processed data colored by the combined clustering result, determined by the described approach.

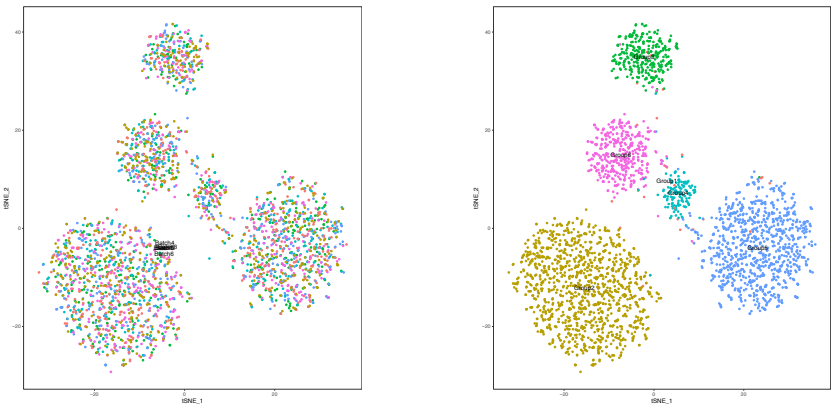

Figure 3.18.: t-SNE maps of the processed data colored by the values of interest Batch_new (left) and Group_new (= ground truth) (right).

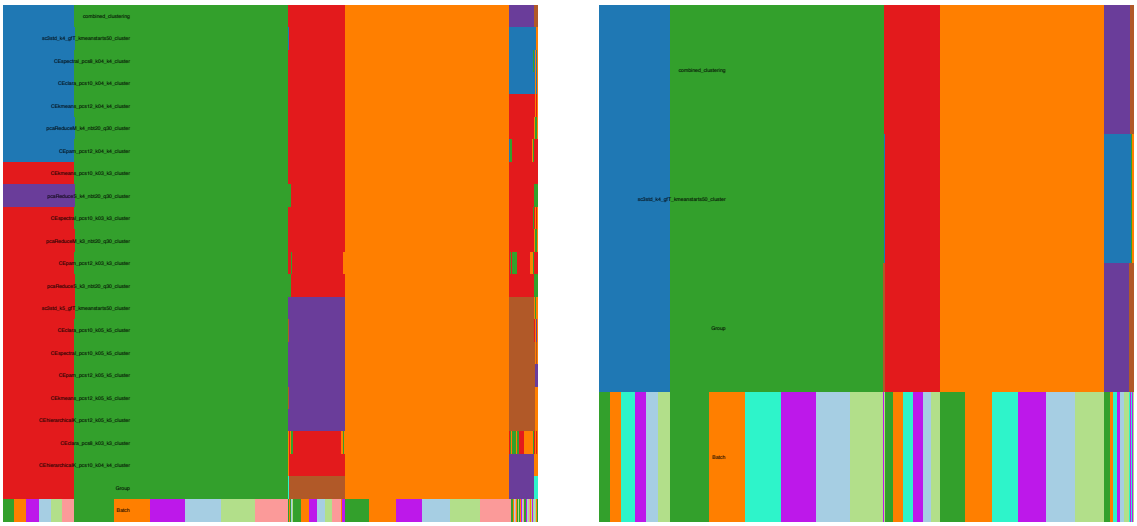

Figure 3.19.: Barplots of the combined and best clustering results with the values of interest Group and Batch (left) and of the combined and favorite clustering and the values of interest Group and Batch (right).

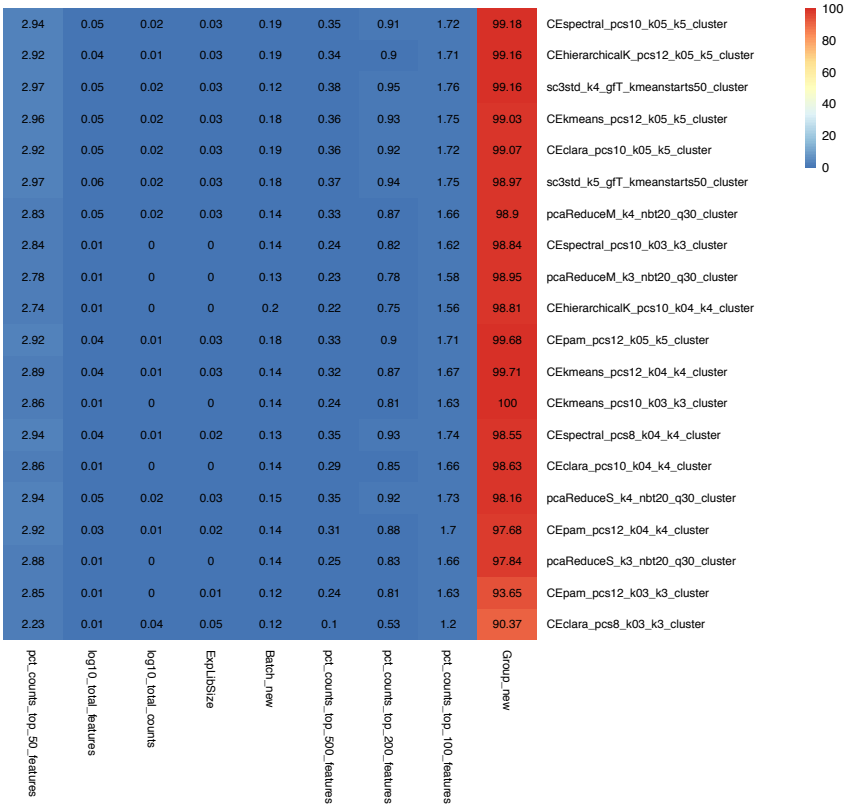

Figure 3.20.: Heatmap table visualizing the influence of selected factors on the best clustering results.

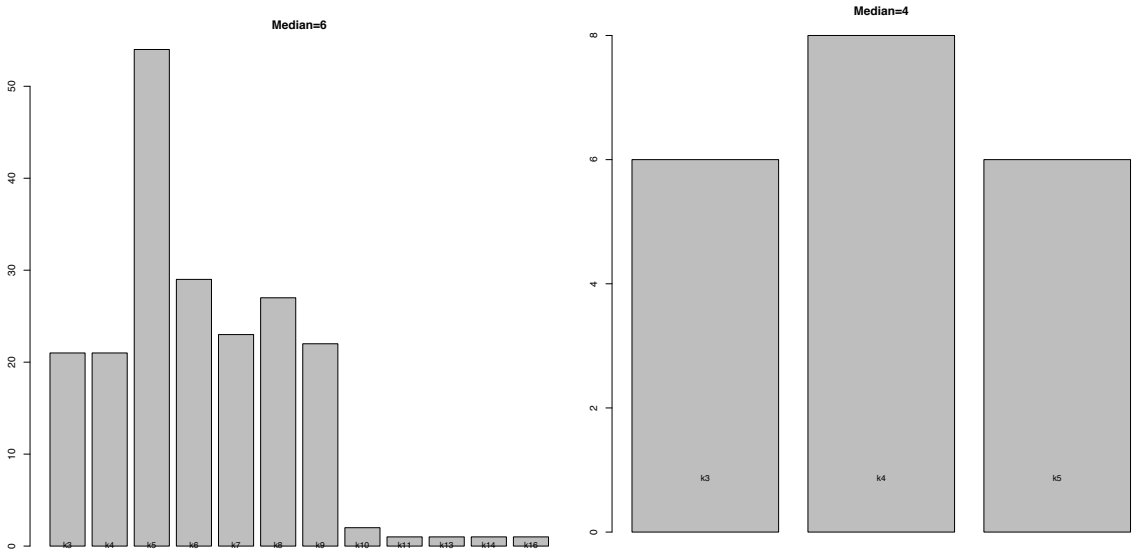

Figure 3.21.: Histograms of the number of found clusters within all (left) and the best clustering results (right).

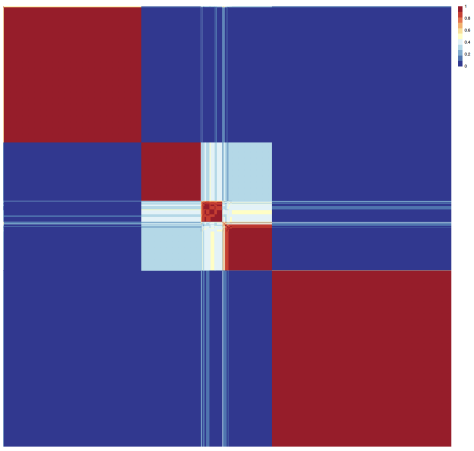

Figure 3.22.: Consensus matrix of the best clustering results.

Starting out by looking at the two comparison metrics, the ARI and NMI values, in Table 3.3 we can already state that the overlap with the ground truth clustering is nearly perfect. Although, at this point the result is not the main concern, because we want to present a full analysis of a simulated dataset with the workflow and not discuss the accuracy or performance, which is the subject of Chapter 4, we wanted to point out with how little effort on the analyst's side, this excellent result was achieved.

In Figure 3.16 and 3.17, we can see two different visualizations (PCA plot and t-SNE map) of the processed data, colored by the final favorite and combined clustering

results. Additionally, we present the t-SNE maps of two values of interest, namely the Batch_new and the Group_new (= ground truth) variables in Figure 3.18. Here, it is apparent that the batch attribute is distributed evenly among all clusters, which is very realistic when thinking about samples originating from the same source, but processed in different batches.

We can immediately appreciate that the "clustering" by the t-SNE map, the final clustering results and the ground truth correspond to each other to a high degree. This is not necessarily always the case (due to higher dimensionality within the data), even if the clustering is correct, but definitely a good indication of having found a sound clustering result in neatly pre-processed data.

The differences between the two final clustering results and the ground truth are distinguishable by comparing their respective t-SNE maps, but not easy to analyze or quantify. This can be achieved in a better way by looking at the barplots in Figure 3.19. These plots demonstrate, in a very comprehensible way, how the combined clustering came to be and visualize the similarity concerning the clustering among the best clustering results. Furthermore, the relation between clustering results and values of interest is presented in a clear way. For a close up inspection of the relation between only the two final clustering results and the values of interest a separate barplot is generated, as we can see in Figure 3.19 on the right. Here, we can see how similar the combined and favorite clustering are to the ground truth. For the purpose of quantifying their similarity we want to reference back to the comparison by metrics shown in Table 3.3.

Even deeper analyses of the results can be undertaken with the help of Figures 3.20, 3.21 and 3.22. Figure 3.20 visualizes in a very comprehensible way, similar to the confounding factor module, the influence of selected variables on the best clustering results. As expected none of the presented variables influence the clustering results in a significant way, apart from the ground truth denoted by Group_new. Figure 3.21 presents an overview of the number of found clusters k in all and especially the best clustering results. This gives a good impression concerning the most probable correct number of populations within the sample. We can see that the median among all results represents the real number of 6 clusters within the data, whereas the best clustering results yield 4 as their median. Nevertheless, the final combined clustering result yields the correct number of 6 clusters. Last but not least, Figure 3.22, shows the consensus matrix of the best clustering results and confirms what we have already suspected in Figure 3.19, that the best clustering results did not differ in their cluster allocation in significant ways. Only in the center of the matrix, representing the smallest of the final clusters, we can spot some discrepancies between the different clustering results.

In conclusion we can state that the analysis of the simulated dataset *sim_2_2* was more than successful and all the steps and decisions made by the workflow were presented in a comprehensible manner.

3.9. Final Result

This part quickly reiterates, but also puts an emphasize on, what the workflow actually delivers in the end.

Two solutions, a favorite and a combined clustering result, by cell populations are basically the final product of the workflow. Besides that, all of the analyses done by the different modules and the manipulations on the data are provided in a comprehensible way with additional visualizations. Thereby, we tried to ensure to adhere to the goals we defined in the beginning of this Chapter concerning the workflow.

The results of every module are always added to the preexisting ones, thereby it is ensured that the changes made to the data can be traced back and that the analyst is able to rerun specific modules at any given time. Therefore, in the end the user is presented with the entire history and description of the whole analysis and every decision made along the way.

The three final objects are the *SingleCellExperiment* object created in the first module and the *ClusterExperiment* and *Seurat* object, which were created in the course of the clustering process. All of them contain nearly the same information on the whole process, provided they support it. We recommend to keep working with the *SingleCellExperiment* object, because it houses the entire analysis history and was used from the beginning. The list of the best clustering results is also recommended for further use, because other solutions ordered by quality are convenient when alternatives are needed or the best clustering results are very different from each other.

3.10. How to Use the Workflow

Most of the time the workflow can not be used in a one and done manner, but rather as a tool in an iterative process of exploring the data at hand. The workflow should enable a flexible and comprehensible way to explore scRNAseq data, without doing repetitive tasks manually. Therefore, it is mostly automated. The results should aid in the decision making process of next steps or be seen as a basis for further exploration of different cell types, clusters or samples, always depending on the question and data at hand.

Therefore, it is often necessary to perform multiple (partial) runs on the same data until one gets a meaningful result or simply a robust basis for further exploratory analyses. These partial runs can be performed as an iterative process with different configuration parameter sets and comparisons.

A good example which uses the flexibility enabled by the workflows modular nature, is the problem of determining the initial set of parameters. If the analyst does not have any clue about the metrics or structure of the data at hand, we recommend to

stop the workflow after the first module, the import module, was performed. Then, investigate the output of the module by looking at the generated visualizations or load the obtained object to explore it even further. Having done that, the user can set the parameters in the configuration files in a more meaningful and informed way and let the workflow continue where it left off. This is only one scenario, which could occur quite often in the beginning of an analysis.

Another scenario, located on the opposite end of the workflow would be the following. Imagine the subject of an analysis is to look for an effect on cellular level on the sample as a reaction to a certain stimulus (treatment, radiation or disease). The literature suggests that the effect only or primarily occurs on a specific subset of cell populations. In this case the scientist could run the analysis as outlined above with the goal of clustering by cell population. Of course, the effect of the stimulus on the data has to be removed in the course of the confounding factor analysis to ensure that it does not dominate the clustering process. After having obtained a final clustering result, the targeted subset of populations, meaning one or more clusters, can be extracted. By using the original raw data of the extracted cells, the workflow can be rerun for the selected subset and the scientific question concerning the effect of the stimulus on certain cell populations can be investigated further.

A general recommendation, when looking for internal structures in the data of any nature is to go from rough clustering, with big but definite clusters, to smaller ones. This can be achieved by the above outlined approach of first dividing the data in big chunks and then rerun the workflow on one single cluster, which has the potential to inhabit the subpopulation we are looking for.

There are endless scenarios such as using different parameters for the quality control, switching normalization method or removing a certain confounding factor, which lead to multiple iterations and reruns of specific parts of the workflow. Thereby, different results for a comparison and exploratory analysis are generated.

For this reason, we called this approach a workflow and not a pipeline, because loops and branches in the process are possible and very likely to occur and pipelines usually operate in a “linear” way. After every module the analyst is encouraged to inspect the output of the module to decide on next steps. The idea is to intervene sooner and be a lot more flexible than with an one-off pipeline, which has no means of configuring the independent steps of the analysis.

4. Verification & Validation

> "Knowing is not enough, we must apply. Willing is not enough, we must do."
>
> Johann Wolfgang von Goethe

This chapter is dedicated to the process of verifying and validating the workflow, which we just described in detail throughout the previous chapter.

We define verification as the act of ensuring that the workflow does exactly what we intended it to do, provided that the assumptions we made are met by the data. This entails among other things the correct determination of clusters and the removal of real confounding factors. Validation on the other hand, describes the process of ensuring that the workflow performs well in a real setting, represented by real world datasets. In other words, by verification we check if the workflow does what it was designed to do concerning the specifications, functions and goals we defined before. By validation, we make sure that the workflow yields valid results according to its desired real world application. Similar definitions of these terms can be found in the field of software development and reliability [Pha03].

In concrete terms, the verification is achieved by applying the workflow on eight different simulated datasets to ensure the proper technical functionality. The validation, will be performed by applying the workflow on a real public dataset and comparing the result with published ones by the community. Additionally, the results will undergo a thorough biological examination by the means of a discussion at the end of the chapter.

We will always explain the setup of the respective process (verification or validation) and discuss the results. In the case of validation we will additionally comment on the analysis of the real dataset to ensure comprehensibility concerning the final results.

4.1. Verification of the Workflow with Simulated Datasets

The verification chapter consists of two parts. First, we will shortly describe the setup of the process. In the second part we will discuss the results and formulate a conclusion concerning the verification of the workflow.

The goal is to show that the workflow addresses all the stated challenges and issues in scRNAseq analysis and obtains, under controlled circumstances, excellent results.

4.1.1. Setup of the Verification

To verify the correct functionality of the specifications and goals of the workflow we simulated eight different datasets with the help of the package *splatter*. This package was exactly made for that purpose, meaning that it is intended to be used for the verification of programs like this workflow in a standardized way, by simulating scRNAseq data. We even estimated the main parameters with the help of the function *splatEstimate*, to base the simulated datasets on a real dataset.

We equipped all eight of the simulated datasets with an inherent clustering as ground truth for the comparison with the final results of the workflow. Four of the datasets were additionally separated into batches to simulate the common situation of having a strong confounding factor present in the data. Furthermore, each of the four datasets in each group (with and without batch separation) differ in their dispersion parameter and therefore represent increasing levels of difficulty concerning the goal of correct clustering. To be more precise the datasets ending with *_1* are not very disperse and should only check all the basic functions, because of that we expect nearly perfect results. Datasets ending with *_2* are significantly disperse, even more than estimated by the function *splatEstimate* based on the real dataset. To test the workflows limits, datasets with the ending *_3* are unrealistically disperse. Until now the dispersion has increased with the appended numbers. Datasets ending with *_4* represent an exception to that rule and are exactly as disperse as the function *splatEstimate* has estimated, which means their level of difficulty is positioned between *_1* and *_2* datasets. Therefore, the datasets with the ending *_2* represent the benchmark we want to achieve, because they are a more difficult task than the real dataset. The motivation for the datasets with the ending *_4* was to analyze if there could be any conclusions drawn concerning parametrization, like normalization method selection, concerning the subsequent analysis of the real dataset. For more details on the datasets, the simulation process, used parameters and the real dataset they are based on, we refer to Appendix A.

The last important thing we have to discuss, concerning the verification setup, is the following question: According to which metrics will the result be measured? As we have already implemented and described the ARI and NMI measures, in Chapter 3.8, for the comparison of the final results with externally supplied metadata, we propose to also use them for the quantification of success within the verification process.

4.1.2. Results & Conclusions

We will present the results of the verification process in two tables, one with and one without batch separation within the datasets. As in Chapter 3.8 we will display the ARI and NMI values of the final clustering results, which are the combined and the favorite clustering, compared to the simulated real clustering (= ground truth). Due

to the nature of the datasets (simulated) this is the best way to evaluate the performance of the workflow, because no biological knowledge can be applied to the results to check for plausibility. We will not go into the details concerning the analyses of the datasets. Each dataset was processed with the same configurations as input, only differentiating in the applied normalization method. The exact configuration parameters of the analyses are documented in Appendix A. To make the tables more readable we always colored the lowest/worst value per dataset and comparison metric (NMI and ARI) in **orange** and the highest/best in **green**. After presenting the results, we will try to draw some conclusions and formulate hypotheses for the analysis of the real dataset.

dataset	normalization	$ARI_{combined}$	$NMI_{combined}$	$ARI_{favorite}$	$NMI_{favorite}$
sim_1_1	DS	0.995	0.986	0.917	0.908
sim_1_1	log2cpm	0.976	0.967	0.97	0.952
sim_1_1	LSF	1	1	0.97	0.952
sim_1_1	SF	0.998	0.991	0.97	0.952
sim_1_1	TMM	0.97	0.952	0.97	0.952
sim_1_1	UQ	0.976	0.967	0.97	0.952
sim_1_2	DS	0.954	0.906	0.859	0.825
sim_1_2	log2cpm	0.996	0.984	0.965	0.943
sim_1_2	LSF	0.996	0.984	0.969	0.95
sim_1_2	SF	0.993	0.981	0.965	0.943
sim_1_2	TMM	0.996	0.983	0.921	0.9
sim_1_2	UQ	0.998	0.987	0.92	0.914
sim_1_3	DS	0.091	0.39	0.617	0.631
sim_1_3	log2cpm	0.241	0.513	0.875	0.811
sim_1_3	LSF	0.534	0.708	0.93	0.871
sim_1_3	SF	0.26	0.537	0.872	0.817
sim_1_3	TMM	0.798	0.772	0.896	0.834
sim_1_3	UQ	0.934	0.839	0.928	0.859
sim_1_4	DS	0.993	0.974	0.759	0.814
sim_1_4	log2cpm	0.969	0.951	0.969	0.949
sim_1_4	LSF	0.996	0.986	0.969	0.952
sim_1_4	SF	0.997	0.989	0.969	0.949
sim_1_4	TMM	0.996	0.984	0.756	0.823
sim_1_4	UQ	0.996	0.986	0.969	0.949

Table 4.1.: Results of the verification with simulated datasets without batch.

We will start with Table 4.1, where we present the results of the comparison to the ground truth for all the datasets, which were not influenced by the batch confounding factor. The results of the analysis of the first dataset, *sim_1_1*, are as expected

and there is not even any value below 0.9, which stands for itself. We even see a perfect match in the LSF normalized results of the combined clustering. The most interesting dataset, *sim_1_2*, due to its realistic and increased difficulty and similarity to the real dataset, yields also very good results with no value below 0.82 and in the section of the combined clustering not even below 0.9 as before. Another situation is presented, when we look at the most disperse dataset in this group, *sim_1_3*. Here, we can see that a lot of approaches do not work at all, with the lowest value at 0.091, and others perform astonishingly well, with the highest values around 0.93. Last but not least, the results of the most similar dataset, compared to the real dataset, *sim_1_4*, present themselves very good with no values lower than 0.756 and by excluding the four (from 24) lowest values not even lower than 0.949.

dataset	normalization	$ARI_{combined}$	$NMI_{combined}$	$ARI_{favorite}$	$NMI_{favorite}$
sim_2_1	DS	0.994	0.979	0.961	0.941
sim_2_1	log2cpm	0.976	0.967	0.969	0.95
sim_2_1	LSF	0.998	0.989	0.962	0.947
sim_2_1	SF	0.976	0.967	0.969	0.95
sim_2_1	TMM	≈ 1	0.998	0.962	0.945
sim_2_1	UQ	1	1	0.962	0.947
sim_2_2	DS	0.935	0.871	0.941	0.879
sim_2_2	log2cpm	0.997	0.993	0.961	0.944
sim_2_2	LSF	0.998	0.99	0.959	0.937
sim_2_2	SF	0.999	0.993	0.961	0.944
sim_2_2	TMM	0.996	0.982	0.960	0.937
sim_2_2	UQ	0.997	0.989	0.757	0.820
sim_2_3	DS	0.165	0.398	0.619	0.608
sim_2_3	log2cpm	0.427	0.636	0.918	0.845
sim_2_3	LSF	0.413	0.605	0.538	0.420
sim_2_3	SF	0.427	0.636	0.918	0.845
sim_2_3	TMM	0.762	0.755	0.882	0.794
sim_2_3	UQ	0.43	0.633	0.921	0.851
sim_2_4	DS	0.99	0.979	0.954	0.927
sim_2_4	log2cpm	0.998	0.993	0.962	0.948
sim_2_4	LSF	0.996	0.988	0.962	0.946
sim_2_4	SF	0.973	0.957	0.962	0.948
sim_2_4	TMM	0.998	0.994	0.962	0.946
sim_2_4	UQ	0.97	0.95	0.962	0.948

Table 4.2.: Results of the verification with simulated datasets with batch.

Moving on to the more challenging datasets in Table 4.2, where we present the results of the comparison to the ground truth for all the datasets, which were additionally

influenced by the batch confounding factor. Again, the results of the easiest dataset, *sim_2_1*, are as expected and the lowest value is 0.941. In one case (UQ), the combined clustering is even a perfect match when compared to the ground truth. The dataset with increased difficulty, compared to the real dataset, *sim_2_2*, which we have analyzed in detail in the course of Chapter 3, yielded very good results. To be more precise apart from the four lowest values, there are none lower than 0.935. As before the results from the most disperse and difficult dataset, *sim_2_3*, cover a big range from 0.165 to 0.921, with some approaches performing over all bad, some over all good and a few in a mixed manner. The most realistic dataset, *sim_2_4*, yielded excellent results with no value lower than 0.927.

According to the results of this limited sample of eight datasets, which were intended to check if all the specifications and goals of the workflow were met, we want to state that the overall performance was better than anticipated. Additionally, we want to mention that in all above listed configuration settings the batch confounding factor was automatically identified and its effect removed from the data. Therefore, we conclude that the workflow was verified by the above stated process and results. Before we continue with the validation of the workflow, we try to extract some knowledge, apart from the verification, out of the results from Table 4.1 and 4.2.

In the following we will list some conclusions and hypotheses, which we think can be drawn from the presented results.

- A higher dispersion within the data does not lead necessarily to worse results. This can for example be seen by comparing the results from *sim_2_4* and *sim_2_2*, where the latter one has a higher dispersion but not all the results are worse than the results of the former one.
- It seems that the consensus approach, which yields the combined clustering result, does not work so good with datasets, which have an extremely high dispersion, as for example in *sim_1_3* and *sim_2_3*. The favorite clustering approach on the other hand still delivered very good results. Therefore, we argue that the number of best clusterings *[n.top.clusterings]*, used in the generation of the combined clustering, is of grave importance.
- We can see that the most sophisticated normalization methods, TMM or LSF, do not always yield the best results. Therefore, we can not even formulate for simulated datasets a recommendation concerning the best normalization method. Following the fact that the datasets *sim_1_4* and *sim_2_4* are the most similar to the real dataset, we could try to predict which normalization method would yield the best results. Having said that, we will not do such a thing, because of the just mentioned unclear dynamics in the background, which lead to good or bad results concerning the applied normalization method.
- Another interesting observation is, that contrary to intuitive thinking, some datasets in the first group (without batch effect) performed not as good as their counterparts in the second group (with batch effect). This can be clearly

seen when comparing the best results of the combined clustering results in *sim_1_2* and *sim_2_2*.

We think, that we have tested a lot of different configurations, even unrealistic difficult ones, and reached satisfying and excellent results. Therefore, we conclude that the chosen methods and approach on the matter is valid and worth further investigation. In the next part we will investigate how good the workflow performs on a real dataset and in comparison with an already published analysis of exactly that dataset.

4.2. Validation of the Workflow with a Real Dataset

After having verified that the workflow meets its specifications and determines clusterings to a satisfactory degree, even in very disperse simulated datasets, we move on to the validation of the workflow by using it to analyze a real public scRNAseq dataset.

This chapter consists of three parts, in which we first describe the setup of the validation process, then give a brief summary of the analysis of the real dataset and finally discuss results and try to draw some conclusions.

4.2.1. Setup of the Validation

The basic idea is to analyze, with the help of the workflow, a publicly available scRNAseq dataset. We chose a rather small one with approximately 3000 PBMC cells (*pbmc3k*) from a healthy human donor provided by 10x Genomics. Thereby, we ensured reasonable computational effort and a sample which is made up of a lot of differently sized cell populations. Therefore, it is a good candidate for the validation of our workflow. A more detailed description, the origins and structure of the dataset can be found in Appendix A.

The most important condition we have to agree on is the metric or measure which decides if the workflow is validated or not. This is especially tricky, because we see ourselves confronted with the fact that we do not and can not have a ground truth, due to the fact that we deal with data from a real biological sample.

We propose two solutions to this conundrum. At first we will compare the results of the workflow to an established, published and widely accepted result from the authors of the package *Seurat* in form of a tutorial [Sat18]. The topic of the tutorial is the analysis of the same dataset we have chosen for the validation process and therefore provides a clustering, which we can use as standard of reference or pseudo ground truth. The deciding metrics would be again represented by the ARI and NMI values, which describe the overlap of the results of the workflow and the ones from the tutorial. The second solution is the application of biological knowledge about

the dataset in comparison to the results of the workflow. For example comparing expected to detected or identified cell populations. Both described solutions are not absolute, but the best we can do under these circumstances. We are convinced that the presented approaches will prove to be sufficient to validate the workflow.

4.2.2. Analysis of the Data with the Workflow

This chapter describes a reduced and optimized analysis of the chosen publicly available real dataset called *pbmc3k*. The goal is to briefly walk the reader through the whole process of analyzing a real biological dataset, in comparison to a simulated one as we have already seen in Chapter 3. The final results will be presented at the end of this part, but discussed in the next chapter.

Starting with the import module, the most interesting plots are the kneeplots, which are presented in Figure 4.1. Here, we can see in the UMI count vs. cells plot that the cut off parameter was chosen wisely, because the knee is rather distinct. The second kneeplot, describing the UMI counts vs. genes, shows us that the range of UMI counts concerning the genes is very wide, reaching from 0 to one gene with over 100000 counts. The last kneeplot, showing the relationship between genes and cells, visualizes the fact that at least more than 2000 cells express more than 500 genes due to the position of the knee. With the help of these plots the decisions concerning the filter parameters for the quality control module can be informed.

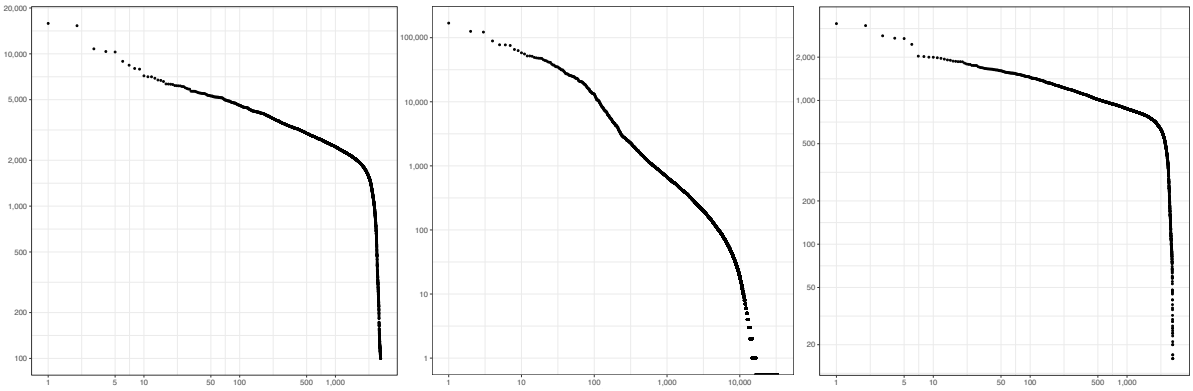

Figure 4.1.: Kneeplots of UMI counts vs. cells (left), UMI counts vs. genes (middle) and genes vs. cells (right) of the *pbmc3k* dataset.

The other plots of the import module, namely the density plots and histograms, will be partly shown in the output of the next module, the quality control module. There, we compare the histograms of the data before and after the quality control module was performed. This enables further fine tuning of the configuration parameters, but also better control over the process. Let us look at Table 4.3, where a summary of the quality control module is presented with the help of some metrics. We can see that, compared to the analysis of the simulated dataset *sim_2_2*, a significant

number of cells was filtered out. Furthermore, the sparsity before, but also after, the quality control step was quite high. The reduction in dimensions, e.g. genes, is again very drastic, but important.

	true	false
expressed genes	16668	16070
relevant cells	2448	519
relevant genes	6684	9984

	before	after
cells	2967	2448
genes	32738	6684
sparsity	97.609%	88.217%

Table 4.3.: Metrics summarizing the filtering process concerning cells and genes (left) and describing the data before and after the quality control module (right).

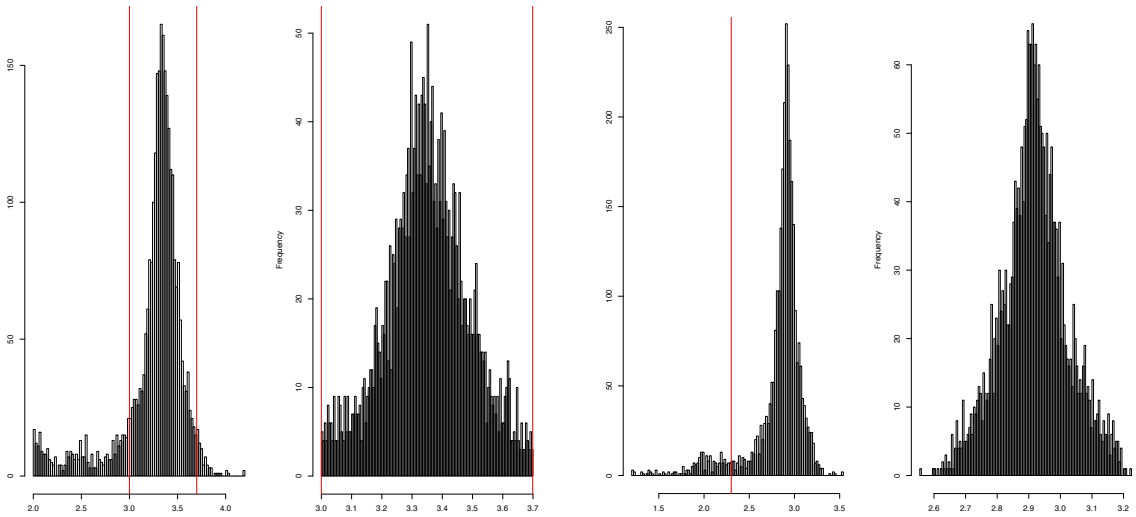

Figure 4.2.: Histograms of log_{10} UMI counts per cell (left half) and log_{10} number of expressed genes per cell (right half) always before (left) and after (right) being processed by the quality control module. The respective cut offs are indicated as red vertical lines.

The histograms in Figure 4.2 visualize the cell filtering process with the help of the cut off parameters. We can see that the cut offs were not too restrictive and we still kept a lot of, hopefully viable, cells. Figure 4.3, describes the cell filtering process according to the rules defined concerning the maximum percentage of mitochondrial genes allowed and the minimum of expressed genes per cell. We notice that a significant number of cells is filtered out due to high mitochondrial activity and that they also do not express a lot of genes in general.

Having a first look at the data in a space of reduced dimensions, apart from kneeplots, histograms or scatter plots of course, with the help of Figure 4.4, we can already make out at least three distinctive cell populations by looking at the

two dimensional t-SNE map and two populations by looking at the first two principal components of the PCA plot.

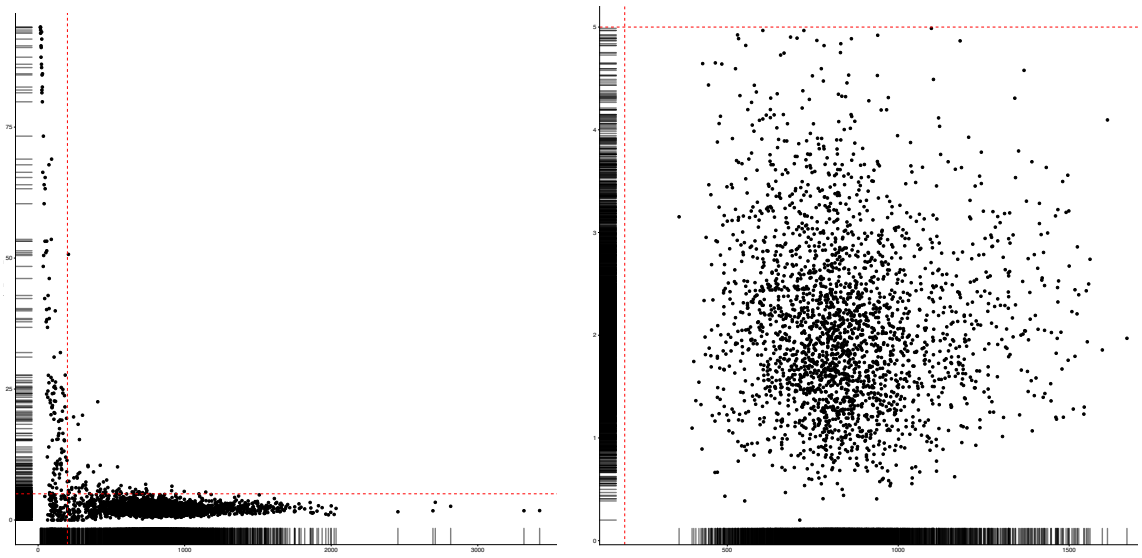

Figure 4.3.: Scatter plots displaying mitochondrial gene percentage vs. number of total genes expressed per cell before (left) and after (right) the quality control module. The respective thresholds are indicated by the red dotted lines.

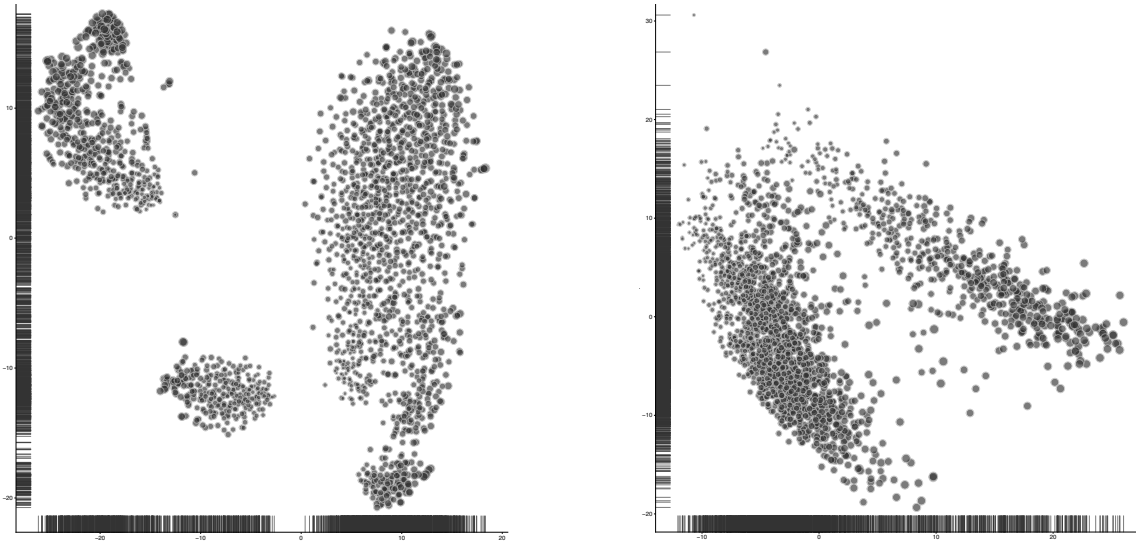

Figure 4.4.: t-SNE map (left) and PCA plot (right) of the log_2 transformed data after being processed by the quality control module.

Moving on to the normalization module we are faced with the task of selecting a normalization result, which hopefully leads to a good clustering result. We processed the dataset with all of the available normalization results until the end and

chose for this presentation of the analysis the weighted trimmed-mean of M-values (TMM) method, because it delivers the best results when compared to the pseudo ground truth provided by the *Seurat* tutorial. A comparison of the final results when choosing different normalization approaches will be discussed in the next part.

After having normalized the data with the chosen method, the confounding factor analysis module enters the stage. By looking at the heatmap tables of Figure 4.5 and 4.6 we can see how the workflow came to the conclusion of removing the effect of the following factors from the data. In Figure 4.5 we can spot sixteen variables from which fourteen influence one of the most informative principal components to a significant degree. By determining their influence on each other, as illustrated in Figure 4.6, we can make out three distinctive groups of factors, which describe each other in a significant way. The most powerful from each group, concerning the influence on the data, is selected and its effect on the data is removed.

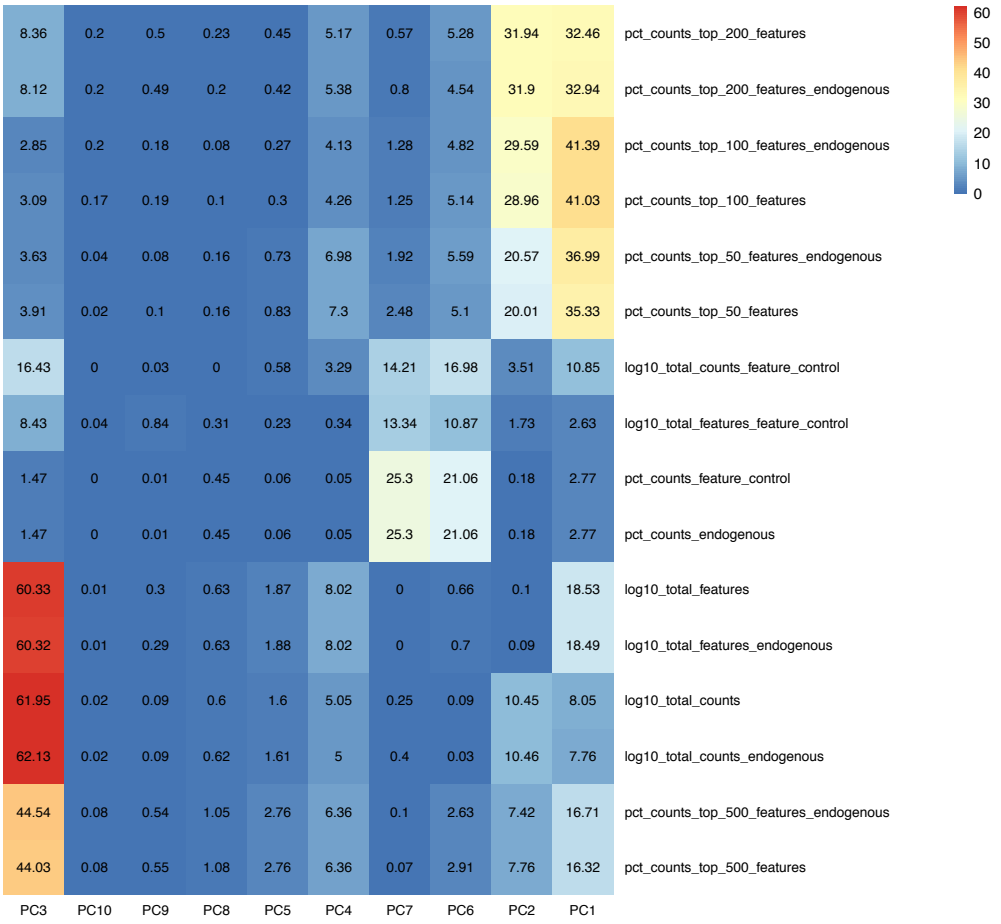

Figure 4.5.: Heatmap table displaying the influence of metadata variables on the ten most informative principal components of the *pbmc3k* dataset in percentage, determined by the standard R^2.

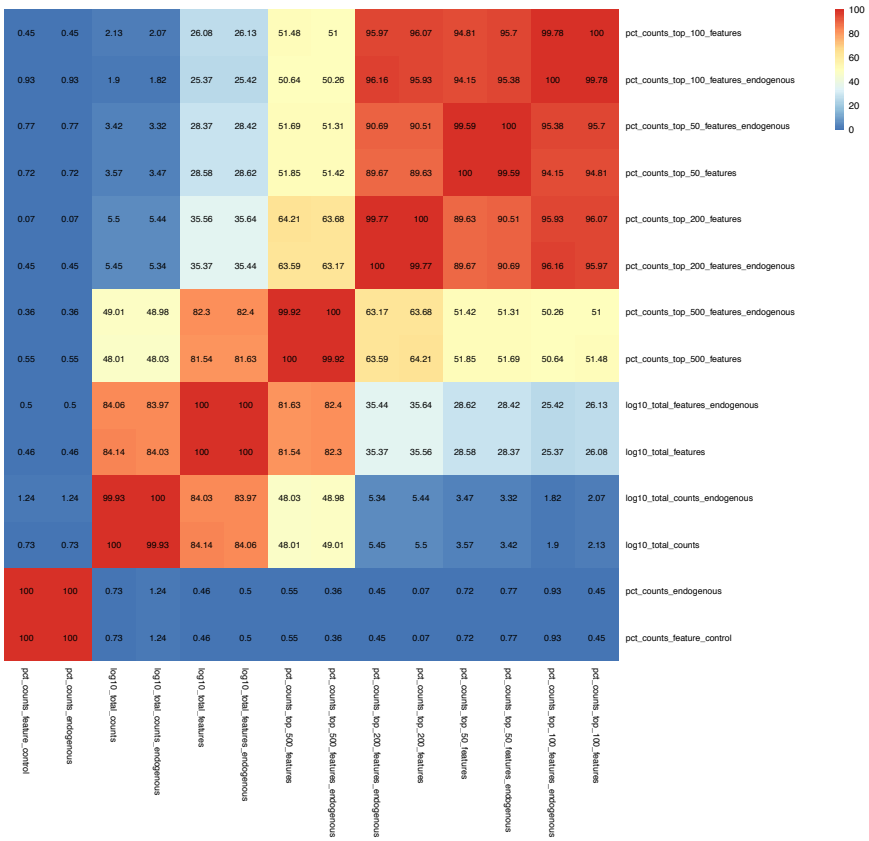

Figure 4.6.: Heatmap table displaying the influence of metadata variables on each other in percentage, determined by the appropriate R^2 measure.

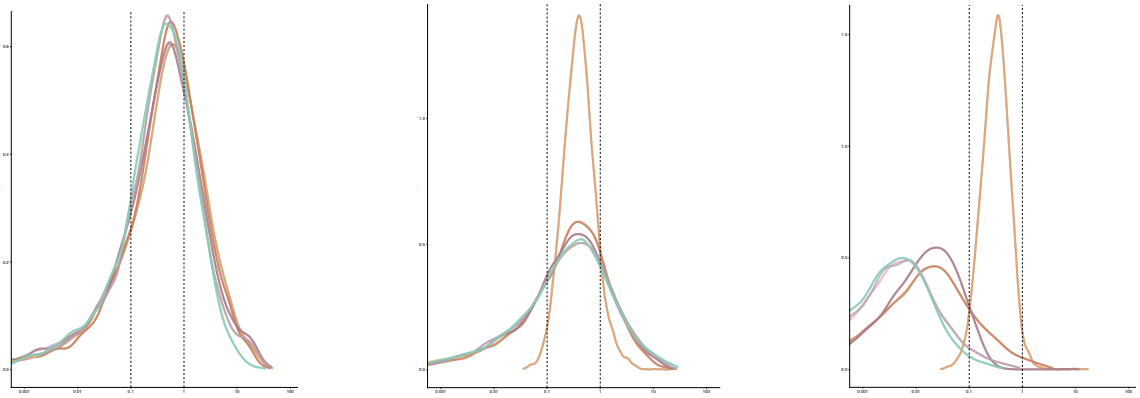

Figure 4.7.: QC plots of the top ten most influencing factors of the respective matrix with no normalization (left), the TMM normalization (middle) and the TMM normalization and removed confounding factors' influence (right). Each color denotes a variable, not necessarily the same in every plot. The x-axis denotes the % of variance explained ($\log_{10}$−scale) and the y-axis the density. The dashed lines indicate 0.1% and 1%, respectively.

We will briefly state the identified confounding factors, their description and possible implications and interpretations concerning their influence.

- pct_counts_top_100_features_endogenous represents the percentage of count values from the one hundred most expressed genes (= features), which are not ERCC spike in genes and therefore endogenous, compared to the counts of all genes. This significant influence indicates that the clustering process would have been dominated by these one hundred most expressed genes.
- log10_total_counts_endogenous describes the logarithm to the base ten of the total counts, only from endogenous genes. We are basically confronted with the fact that the library size (= total counts) still describes each cell in a significant way and thereby cells with a greater library size would be deemed as more important in the course of the clustering process.
- pct_counts_feature_control is the percentage of counts per cell only derived from feature control genes, which are in this case the mitochondrial genes. Although we filtered rigorously according to the expressed percentage of mitochondrial genes in the course of the quality control module, they apparently still describe the data in a significant way.

Finally, the plots of Figure 4.7 show us how the normalization and the additional removal of the effect of the identified confounding factors from the data transforms the data in regards to the most influencing factors in general. Again we want to mention that these plots should be used in a qualitative manner, because the quantitative part was done by the workflow. Nevertheless, the plots visualize nicely how the taken measures helped in this case with reducing the influence of these factors.

The clustering module, as the exception to the underlying basic architecture of the workflow, does not provide any visualizations. Only the number of initial and not filtered cluttering results is supplied. The workflow computed 174 potentially valid clustering results.

The last module, the cluster analysis module, on the other hand delivers a lot of visualizations and results. The best fifteen clustering results according to the described approach, ordered descending by quality, are

- *CEkmeans_pcs8_k06_k6_cluster*
- *CEclara_pcs10_k06_k6_cluster*
- *CEhierarchicalK_pcs8_k05_k5_cluster*
- *CEhierarchicalK_pcs10_k07_k7_cluster*
- *CEhierarchicalK_pcs10_k08_k8_cluster*
- *CEkmeans_pcs10_k05_k5_cluster*
- *seurat_k4_pcs12_res0.2_cluster*

- *CEclara_pcs10_k05_k5_cluster*
- *seurat_k5_pcs12_res0.3_cluster*
- *CEpam_pcs10_k05_k5_cluster*
- *CEhierarchicalK_pcs10_k09_k9_cluster*
- *CEkmeans_pcs12_k08_k8_cluster*
- *seurat_k6_pcs10_res0.5_cluster*
- *sc3std_k6_gfT_kmeanstarts50_cluster*
- *tSNEkmeans_k5_perp30_ntop1000_cluster*

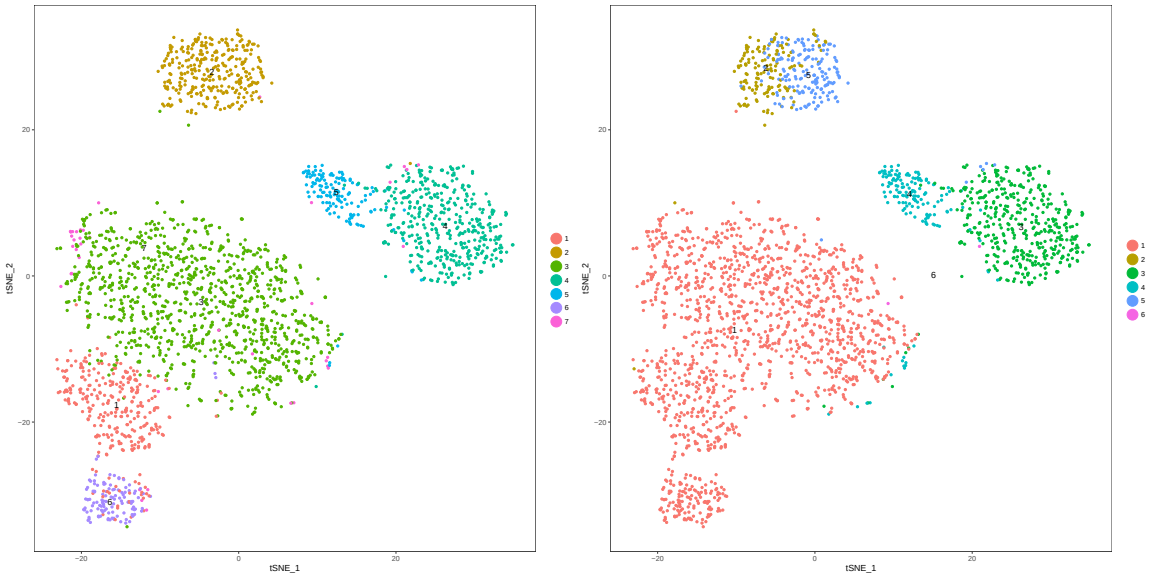

Figure 4.8.: Two dimensional t-SNE maps of the processed data colored by the final combined (left) and favorite (right) clustering result.

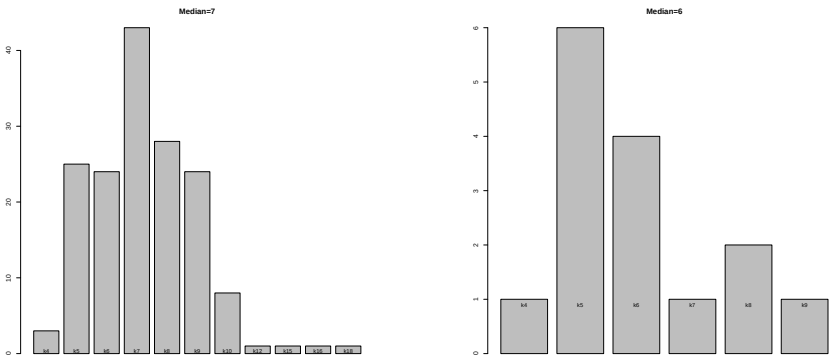

Figure 4.9.: Histograms of the number of found clusters k within all (left) and the best 15 clustering results (right).

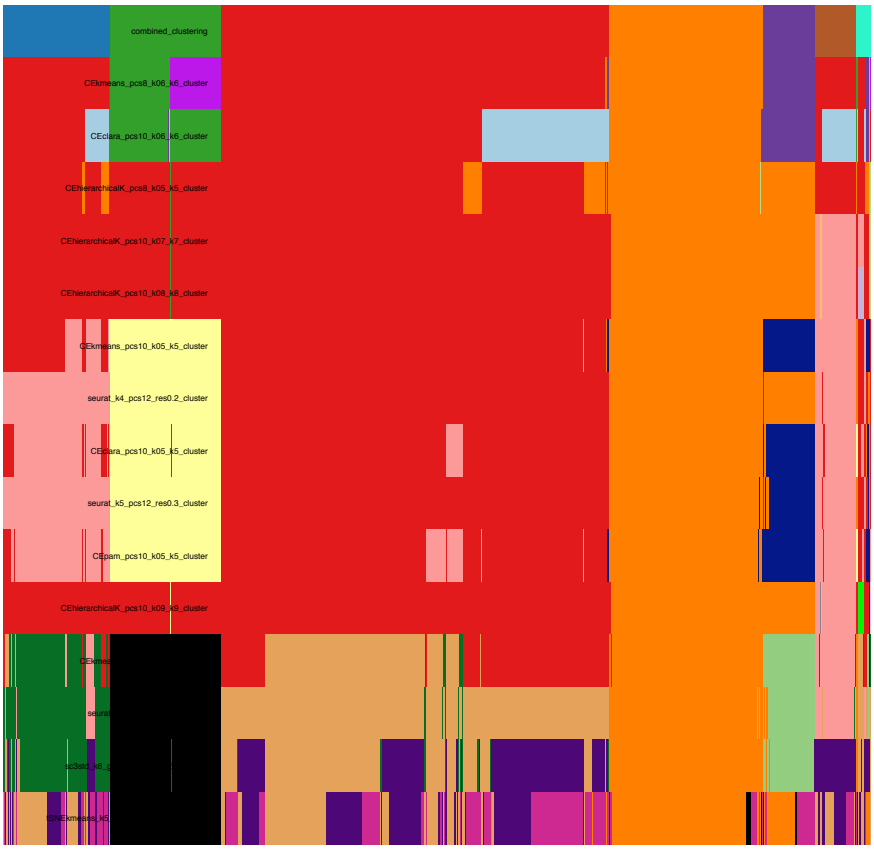

Figure 4.10.: Barplot of the combined and the 15 best clustering results.

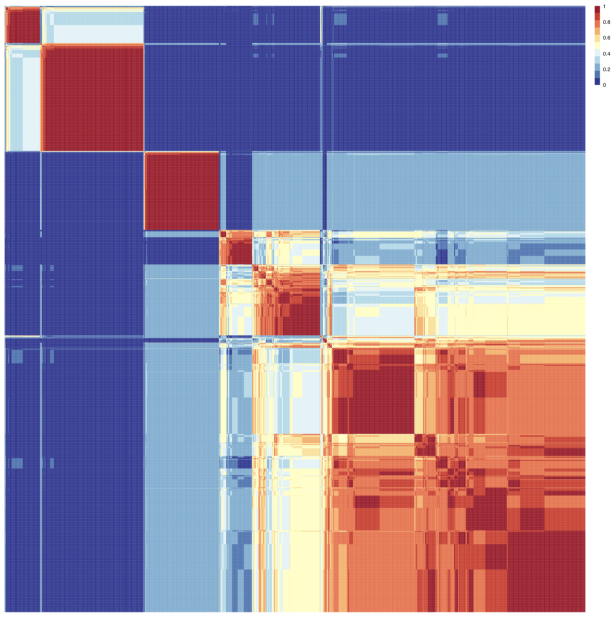

Figure 4.11.: Consensus matrix of the 15 best clustering results.

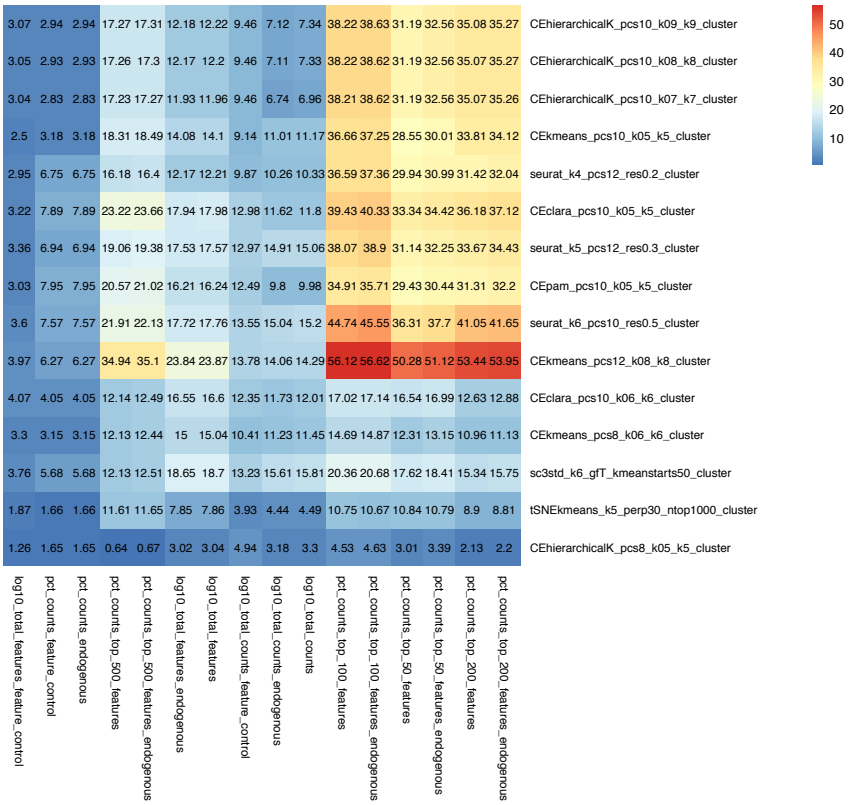

Figure 4.12.: Heatmap table visualizing the influence of selected factors on the 15 best clustering results.

By looking at the t-SNE maps in Figure 4.8 we can already see that the combined and the favorite clustering differ concerning the cluster allocation in two ways. First, in the top region of the t-SNE map, where the favorite clustering displays two and the combined clustering one cluster. The second difference lies in the left bottom quadrant of the t-SNE map, where the favorite clustering displays one, but the combined clustering three clusters. Furthermore, we can see the t-SNE map itself as an assurance that both final clustering results have found legitimate clusters, because the colored clusters agree with the ones determined by the t-SNE map.

Moving on to the histograms in Figure 4.9, we see that the most probable real number of clusters k within the data lies between six and seven, due to the fact that the median of all clustering results is 7 and the median of the best clustering results is 6. These numbers speak for the final clustering results, which have found six and seven clusters, respectively.

The barplot in Figure 4.10 shows us how the combined clustering came to be and how often the 15 best clustering results agree on the cluster allocation of cells. Additionally, we can investigate which clusters are the same in every clustering result as for example the orange one, which is cluster 4 in the combined clustering

result. This indicates the robustness of that cluster. In other situations it informs on potential subpopulations as for example it is the case in the first (blue), third (red) and sixth (brown) cluster within the combined clustering (first row), whose cells are all in the first cluster (red) of the favorite clustering (second row). We already mentioned this discrepancy between the combined and the favorite clustering result when we looked at the t-SNE maps in Figure 4.8.

Another way to investigate the robustness and composition of the clusters within the combined clustering result is visualized by Figure 4.11. In the top left quadrant, we can clearly see that three clusters were found in the same way by nearly all of the 15 best clustering results. Three or even four more are not so distinctively visible, but can be seen in the bottom right quadrant and could be subject to further investigation.

At last we can take a look at the factors which describe the 15 best clustering results in a significant way. This can be achieved by analyzing the heatmap table in Figure 4.12. Here we can for example see that the best clustering result, *CEk-means_pcs8_k06_k6_cluster*, can not be described in a significant way by any of the clustering results. In contrast to that, the 12th best clustering result, *CEk-means_pcs12_k08_k8_cluster*, which was determined by exactly the same algorithm but different parameters and input data, is significantly influenced by multiple factors.

The analysis and results presented in this part are comprehensible, give a well-rounded and informative image and constitute a solid basis for further explanatory data analysis. In the next and last part of the validation process we will try to analyze these results and draw some final conclusions.

4.2.3. Results & Conclusion

At last we want to put the results of the previous analysis into context and try to answer the question of validity concerning the workflow. This is done in two ways, as we have already outlined. First, by the comparison to a standard of reference or pseudo truth represented by the results of a tutorial analyzing the same dataset [Sat18]. Secondly, we will apply biological knowledge to the results of the analysis to figure out if the determined clusters represent known cell types, which can be identified by characteristic marker genes.

Comparison to a Pseudo Ground Truth

Similar to the verification process we will simply compare the results of the workflow, computed for every available normalization method, with the two different results provided by the tutorial from SatijaLab [Sat18].

The two solutions provided by the tutorial differ only by one cluster in solution 1 (with 8 clusters), which is separated into two clusters in solution 2 (resulting in

9 clusters). Due to the fact that they explain that phenomena biologically and algorithmically we compare the results of the workflow to both solutions in the following Tables 4.4 and 4.5. As before we always colored the best results per final clustering result and quality measure (column) in green and the worst in orange.

normalization	$ARI_{combined}$	$NMI_{combined}$	$ARI_{favorite}$	$NMI_{favorite}$
DS	0.875	0.831	0.724	0.795
log2cpm	0.614	0.767	0.338	0.509
LSF	0.872	0.821	0.346	0.52
SF	0.788	0.797	0.813	0.805
TMM	0.875	0.844	0.574	0.711
UQ	0.61	0.761	0.694	0.748

Table 4.4.: Comparison of the *pbmc3k* workflow results with solution 1.

When we compare the final results of the workflow, obtained by applying different normalization methods, with the results of solution 1 in Table 4.4, we notice that the best values are not as high as we have seen in the previous part, the verification. This is the case, because the best result before were exceptionally good in terms of the interpretation of the quality measures ARI or NMI, respectively, probably due to the fact that we dealt with simulated data. Additionally, we have to mention that we compare to a pseudo-truth and the sample is biological in nature and therefore may be subject to more irregular distributed variation.

Furthermore, it gets apparent why we chose the weighted trimmed-mean of M-values (TMM) method for the normalization when we presented the analysis of the *pbmc3k* dataset in the last section, because it yields the best results when comparing the consensus clustering approach to the results of the tutorial. In general, the consensus approach with the combined clustering as its results, yielded very good overlap to solution 1 with the lowest ARI value being 0.61 and the lowest NMI value being 0.761. The favorite clusterings on the other hand cover quite a wide range of values reaching from 0.338 to 0.813 in the ARI values and 0.509 up to 0.805 in the NMI values.

Here, we want to focus more on the combined clustering results, because the second best clustering result may overlap to a higher degree with solution 1 and this would involve interaction by the analyst. By focussing on the initial results without more intervention than necessary we can argue in respect of the automated aspect of the workflow.

Moving on to Table 4.5, where we compare the workflows results with solution 2, which, among other things, splits one cluster from solution 1 in two. The comparison to this, more clustered solution, is not as good as we have seen before. The difference between the two comparison results should not be surprising due to the fact that the solutions are also different. Therefore, it is clear that one fits better to some

results of the workflow than the other. Having said that, the results are still pretty good, probably because the major difference between solution 1 and solution 2 is the split of one cluster. Again, the range of quality measure values of the combined clustering results is quite narrow with the ARI values stretching from 0.613 to 0.644 and the NMI values from 0.735 to 0.772. When comparing the favorite approach with solution 2 we encounter a wide range of values for the ARI metric from 0.209 to 0.562 and the NMI measure from 0.448 to 0.734. By focusing rather on the combined clustering results than the favorite ones we would chose the TMM result again as the best to present in the analysis part from before.

normalization	$ARI_{combined}$	$NMI_{combined}$	$ARI_{favorite}$	$NMI_{favorite}$
DS	0.644	0.765	0.484	0.719
log2cpm	0.642	0.741	0.209	0.448
LSF	0.618	0.752	0.214	0.456
SF	0.613	0.735	0.562	0.734
TMM	0.633	0.772	0.368	0.642
UQ	0.631	0.735	0.464	0.675

Table 4.5.: Comparison of the *pbmc3k* workflow results with solution 2.

Obviously, the dependency on the normalization method is much higher in real datasets than in simulated datasets, which confirms the presented approach of always calculating the normalized values with every implemented method to enable easy switching between differently normalized datasets during an analysis.

The attentive reader will wonder how many clustering results within the best clustering results, which make up the combined clustering, were obtained by applying the SNN-Cliq clustering approach provided by the *Seurat* package, which was the tool used in the tutorial to analyze the dataset. The answer depends of course on the applied normalization method, but in the case of the TMM normalization method, three of the best fifteen clustering results were obtained by the SNN-Cliq algorithm, which is only a fifth and therefore not enough to represent the required majority for the decision making within the consensus approach.

We wanted to mention again that the results, solution 1 and solution 2 from [Sat18], do not represent a definite ground truth, but rather a standard of reference or pseudo ground truth. Nevertheless, in the absence of a better alternative we are confident that this comparison is meaningful as the authors of the tutorial have already published a lot of analyses in respected journals as for example Butler et al [BHS+18] in Nature Biotechnology. At last we want to emphasize that very similar results were achieved with the here presented workflow, without the application of any in depth biological know how (compared to the SatijaLab analysis) and not a lot of interaction was needed from the scientist.

In conclusion we can say that the comparison to the analysis results of SatijaLab

also yielded very good results, when looking at the final clustering results of both approaches.

Identification of Cell Types by Gene Expression

The second approach in the validation process is to identify the obtained clusters' cell types by applying biological knowledge. In this case with the help of marker genes. The question we want to ask ourselves is: Are the results biologically reasonable? Or in other words: Do we find the typical cell types, which are usually present in a PBMC sample?

For that purpose we present in Figure 4.13 fifteen t-SNE maps of which fourteen are colored by (differentially) expressed genes compared to the other cells and one is colored by the combined clustering result yielded by the workflow, as described in the previous part on the analysis of the *pbmc3k* dataset. Here, the color grey denotes no (significant) expression, violet weak expression and dark blue strong expression compared to the other cells.

In the following Table 4.6 we will go through the found clusters by number and color and try to determine the cell type with the help of the differentially expressed genes visualized in Figure 4.13. We sorted the gene names within the table by expression strength and marked the ones which were exclusively expressed by a certain cluster in **bold**.

cluster	color	strong expression	weak expression
1	red	NKG7**, CD8A**	IL7R, S100A4, LYZ
2	brown	**MS4A1**, **CD79A**	LYZ
3	green	IL7R, **CCR7**	S100A4, LYZ, CD4
4	turquoise	CST3, **CD14**, LYZ	S100A4, CD4
5	blue	CST3, **MS4A7**, FCGR3A, LYZ	S100A4, CD4, NKG7
6	violet	**GNLY**, NKG7, FCGR3A	S100A4, LYZ
7	pink	-	-

Table 4.6.: Gene expression of the clusters within the combined clustering result of the *pbmc3k* data ordered by expression strength and marked bold when exclusively expressed.

We try to figure out which clusters represent certain cell populations to check if they resemble the anticipated cell types of PBMCs outlined in Appendix A. This is done with the help of the online platform *Human Protein Atlas* available from www.proteinatlas.org and described in [TÅW+17], which among other things contains mRNA expression profiles from a diverse panel of human-derived cell lines and provided the gene encoding and protein function information used below.

We primarily focus on exclusively expressed genes by certain clusters and subsequently on the most frequently expressed ones.

- Cluster 1 is the only cluster expressing the gene CD8A, which is a cell surface glycoprotein found on most cytotoxic T lymphocytes. Therefore, we think it consists of **CD8 T cells**.
- Cluster 2 with two exclusively expressed genes, namely MS4A1 and CD79A can be easily identified as **B cell** cluster, due to the fact that the first one encodes a B-lymphocyte surface molecule and the second one encodes the Ig-alpha protein of the B-cell antigen component.
- Cluster 3 the biggest cluster in our result, is a little harder to classify by our gene selection, because both CCR7 and IL7R, which are the most expressed genes, occur in more than one cell type. The receptor encoded by the CCR7 gene is expressed in various lymphoid tissues and activates B and T lymphocytes. The protein encoded by the IL7R gene has been shown to play a critical role during lymphocyte development. Nevertheless, due to the fact that it expresses additionally CD4 and we have already identified the B cell cluster, we conclude that it mainly consists of **CD4 T cells.**
- Cluster 4 is the only cluster expressing the CD14 gene, which encodes a protein mostly found in monocytes. Thats why we conclude that this cluster consists of Monocytes expressing CD14 (**CD14+ Monocytes**).
- Cluster 5 characterizes itself by expressing the MS4A7 gene, which encodes proteins associated with mature cellular function in the monocytic lineage. Due to the fact that FCGR3A was also expressed very high, compared to the other clusters we denote it as the **FCGR3A+ Monocytes** cluster.
- Cluster 6 can be described by the expression of the gene GNLY and NKG7. The former gene encodes a protein, which is present in cytotoxic granules of cytotoxic T lymphocytes and natural killer cells. The latter one has the alternative name *natural killer cell protein 7*. This leads us to the conclusion that the cluster consists of **NK cells**.
- Cluster 7 consists of not-clustered cells, which are distributed all over the t-SNE map.

We want to mention that SatijaLab also used a similar approach to identify the clusters' cell types, in their tutorial [Sat18]. We tried to apply the same tactic and also used some of the same nomenclature concerning the cluster names to enable an easier comparison.

The additional cluster, which differentiates solution 1 from solution 2 in the tutorial of SatijaLab is founded in biological reasoning. They noticed that the CD4 T cell cluster could be divided by their cells' expression of the genes S100A4 and CCR7. They argued that this indicates a separation between memory and naive CD4 T cells. When looking at Figure 4.13, we can also see this separation in Cluster 3 (green) by those two markers. Furthermore, we see in Figure 4.10 that four of the best clustering results divided cluster 3 into two sub clusters.

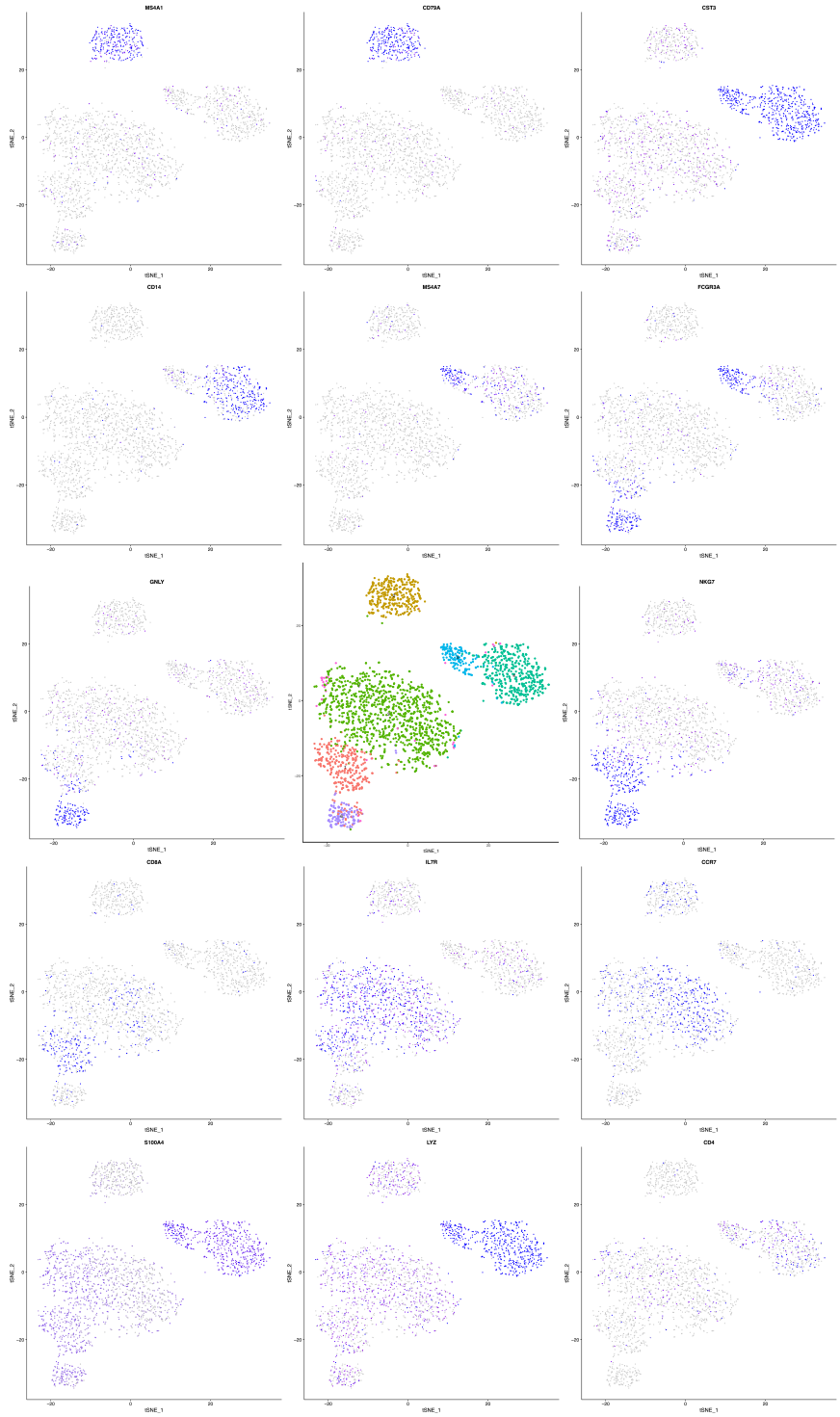

Figure 4.13.: t-SNE maps of the processed *pbmc3k* dataset colored by gene expression or combined clustering result, respectively. From left to right and top to bottom the t-SNE maps are colored by MS4A1, CD79A, CST3, CD14, MS4A7, FCGR3A, GNLY, combined clustering result, NKG7, CD8A, IL7R, CCR7, S100A4, LYZ and CD4.

Finally, comparing the identified cell types with the ones expected in a PBMC sample from Appendix A, we notice that the only cell type we could not find is dendritic cells, which represents by far the smallest cell population within PBMCs.

We think the workflow is validated by the methods and results presented here. In conclusion we state that the workflow does what it was designed for in an excellent manner.

One final remark has to be made at this point. The results obtained by the workflow represent a starting point for the scientists. Therefore, the potential of the applications for the workflow is greater than presented in this work. Starting out with a robust and comprehensible clustering result and a lot of configuration options or alternative solutions makes it easier to focus on what matters most in research: asking questions and driving the decision making processes, which lead to answers, with rigorous data analysis.

5. Conclusion & Outlook

"If there is effort, there is always accomplishment."
Kano Jigoro, founder of Judo

5.1. Conclusion

We started out by stating our motivation for this work. Followed by that, we gave a quick introduction to the topic of scRNAseq and how it is different compared to standard DNA sequencing or bulk-RNAseq. Then we explained scRNAseq with the 10x Genomics solution in detail, which enabled scRNAseq to rise to the top of the next generation sequencing field, due to high throughput and quality at a reduced price per experiment. This new approach results in scRNAseq data with novel challenges for the field of computational biology. These challenges are tackled by scRNAseq analysis with a variety of potential goals.

The next part of this work concerned itself with the development and presentation of a semi-automated workflow for the analysis of scRNAseq data. The applied methods and procedures were based on the theoretical considerations of Reichl [Rei18]. We formulated definite goals to be met additionally to the fact that the challenges of scRNAseq data have to be overcome. Therefore, we presented a specific module architecture to meet the outlined objectives. We briefly discussed that the interaction with the workflow is enabled via configuration files and explained their capabilities. Then we described the goal, implementation and results of each module in detail. To make it even more comprehensible we processed a simulated dataset along the way. This entailed, among other things, decision making, interpretation of outputs and discussion of visualizations. We put an emphasis on two modules, which we denoted as centerpieces of the workflow. The confounding factor analysis module and the cluster analysis module.

The confounding factor analysis module deals with a huge challenge in scRNAseq in general, that is the high susceptibility to noise, technical errors or other factors which influence the data generation in a significant way. Therefore, the identification and rigorous analysis of potential confounding factors and the removal of their effect on the data is critical during scRNAseq analysis. The cluster analysis module tackles another unsolved challenge within the realm of scRNAseq analysis, which is the robust and comprehensible clustering by cell populations according to the cells' gene expression levels. A lot of algorithms have been proposed to solve this problem,

which showed excellent results on some datasets, but failed completely in other cases. Therefore, we decided to focus on the analysis of potentially valid clustering results and implemented an approach, theoretically proposed in [Rei18], which yields two final clustering results.

We ended this part by reiterating what the final results of the workflow are and how the workflow is intended to be used in research.

The next part of this work was denoted to the verification and validation of the presented workflow. The purpose of the verification was to check if all the goals were met and that the workflow solves the identified challenges in a satisfactory manner. We conducted the verification with the help of eight simulated datasets, which inhabited different characteristics relating to prominent challenges in scRNAseq analysis such as confounding factors, high dispersion within the data and cell populations of varying sizes. These populations or clusters were induced in the course of the simulation process to have a real ground truth to determine the performance of the workflow. Therefore, the verification was done by processing all the datasets and comparing the final clustering results with the underlying simulated real clusterings. The comparison yielded excellent outcomes and even the simulated datasets with confounding factors and unrealistic high dispersion were analyzed properly.

The validation on the other hand concerned itself with the applicability of the workflow in a real world setting. Therefore, we analyzed a publicly available real world dataset. We presented the analysis in a comprehensible way and guided the reader through all the critical steps. The subsequent process of validation was split in two approaches. First, we compared the final results of the workflow with the results of a scRNAseq analysis tutorial, which analyzed the same dataset. In other words, we used the clustering results from another group as pseudo ground truth to compare our final clustering results with theirs. The outcome was very good and satisfactory. The other approach was to analyze the results with the help of biological knowledge about the sample and thereby identifying the cell type of each cluster with the help of marker genes. The identified cell types were in line with our expectations and therefore represented another confirmation concerning the validity of the workflow.

Therefore, we conclude that the developed and here presented semi-automated workflow for scRNAseq analysis met the outlined goals by rigorous verification of the functionality with simulated datasets and validation through the analysis of a real dataset and comparing the results to published ones.

Finally, we want to state that this workflow is not an one-off pipeline, but a dynamic and interactive workflow with automated components to aid the scientist in the tasks that cost time and effort so the focus can be on decision making, answering questions, research and further exploration of the data on a solid and informed base. We are confident that the workflow we developed and presented is more than the sum of its parts or modules and delivers robust and comprehensible results.

5.2. Outlook & Further Work

The work on such a workflow with the goal of developing a best practice way to analyze scRNAseq data is never done, but nevertheless we have to come to a close. The defined scope and goals were met and we obtained excellent results when putting the workflow through rigorous verification and validation processes. In regards to potential further work and outlooks the analysis of a lot more public datasets and the associated comparisons with other analysis results come to mind. This could aid in the understanding of the dynamics within scRNAseq datasets from different sources in connection to certain methods. Furthermore, this could lead to better understanding and selection of configuration parameters.

Apart from analyzing more data and comparing results with others, the modules were developed to be easily extended by novel approaches or methods. Therefore, we see the need for updates in this regard on a regular basis with the accompanied testing and comparing to results of previous versions of the workflow.

As we have stated throughout this work, we think of the results obtained by the workflow as a basis for further exploration and analysis. This leads to the idea of developing further downstream analysis modules, which are based on the final clustering results yielded by the workflow. Two examples of such downstream analysis modules would be tackling better differential gene analysis for a more accurate identification of cell types or sub populations, or pseudo time analysis to look for temporal changes within cell populations as a reaction to certain stimuli.

Another idea for next steps in the realm of scRNAseq analysis would be to involve machine learning methods such as neural networks, which could be trained with the knowledge generated by this workflow (the classification of cell populations). Thereby, standard workflows could be replaced by models resulting from such approaches. Training unsupervised approaches on datasets from different sources (sequencing technology wise) could lead to new insights on the "real" characteristics of certain cell populations. Of course, the discovery of new cell types or subpopulations is usually not possible with the help of such approaches, because the best model is only as good as its training set, which might not inhabit an unknown cell population. Therefore, the generated model would not be able to recognize the unknown population. However, exactly this kind of challenges make research interesting.

One last remark concerning standards and norms has to be made. Packages like *splatter*, which enabled us to simulate scRNAseq data in a standardized, comprehensible and reproducible way, are very important to collaborate in a meaningful way across the globe when working in such a field. Therefore, we argue that rigorous and comprehensible standards and norms, concerning formats, methods, datasets and procedures should be developed. We think that this is just the beginning of this new discipline and that it will yield or contribute to quite a lot of new discoveries.

A. Datasets & Configuration Parameters

This appendix will provide all the necessary information about the datasets used in the process of verification and validation concerning the presented scRNAseq analysis workflow's capabilities and functionalities.

The first part describes the real dataset we chose for the validation process, but also to estimate the parameters for the simulation of datasets used in the verification process. The second part deals with the simulated datasets and how they came to be. The last part provides the configuration parameters, which were used to process and analyze all datasets.

A.1. 3k PBMCs from a Healthy Donor

This dataset was used for the validation of the workflow and as a basis for the simulation of the datasets used in the verification process. It is a sample of approximately 3000 peripheral blood mononuclear cells (PBMCs) from a healthy donor, hence the abbreviated name *pbmc3k*. A PBMC is any peripheral blood cell having a round nucleus. The dataset was provided publicly online by 10x Genomics [10x16a], the company which micro-droplet capturing technique for scRNAseq was presented in Chapter 2.3.

In the following we will list multiple reasons, which lead to the selection of this dataset for the sake of validating the workflow and as a basis for the simulated datasets.

- The data is provided by the company, which invented the micro-droplet capturing technique we described in the beginning of this work.
- The sample size of approximately 2700 viable cells is high enough to encounter the novel challenges, which we described in Chapter 2.4.
- The number of cells is low enough to work on the dataset in an iterative manner for testing, refining and comparing of results without facing to much computational effort.
- Peripheral blood mononuclear cells (PBMCs) from a healthy donor host a wide range of cell types, compared to samples from specific organs or tissues (e.g.

liver), with varying cell population size. Therefore, we thought of it as a very good dataset for the validation process.

- Using a sample from a human donor makes the purpose and goals of this work more relatable and supports the claim of scRNAseq analysis position in medical or pharmaceutical research.

In the following we directly state some technical metrics from 10x Genomics [10x16a] concerning this dataset.

- Single Cell Gene Expression Dataset by Cell Ranger 1.1.0.
- Peripheral blood mononuclear cells (PBMCs) from a healthy donor.
- PBMCs are primary cells with relatively small amounts of RNA (~1pg RNA/-cell).
- 2,700 cells detected, sequenced on Illumina NextSeq 500 with ~69,000 reads per cell 98bp read1 (transcript), 8bp I5 sample barcode, 14bp I7 GemCode barcode and 10bp read2 (UMI)
- Analysis run with --cells=3000
- Published on May 26, 2016

Furthermore, it is of interest which cell types are usually present in a sample of PBMCs from a healthy donor. Therefore, we present a list of common cell types in human PBMCs and their anticipated frequency (varies across individuals) from Chapter 15 of [VCLE+15] on PBMCs.

- Lymphocytes $70 - 90\%$
  - CD3+ T cells $70 - 85\%$, consisting of CD4+ and CD8+ T cells with an approximate ratio of 2:1
  - B cells $5 - 10\%$
  - NK cells $5 - 20\%$
- Monocytes $10 - 20\%$
- Dendritic cells are rare $1 - 2\%$

A.2. Simulated Datasets

The eight simulated datasets served their purpose in the development and the verification of the workflow. They were generated with the help of the *R* package *splatter*, which provides a comprehensible and standardized way to simulate scRNAseq data. The package even provides the means to estimate the simulation parameters based on a real dataset with the function *splatEstimate*. We used this functionality to get an initial and realistic estimate for the parametrization of the function, based on the *pbmc3k* dataset.

The estimated parameters based on the *pbmc3k* dataset by the function *splatEstimate* of the package *splatter* were as follows.

```
A Params object of class SplatParams
Parameters can be (estimable) or [not estimable], 'Default' or 'NOT DEFAULT'.
Global:
(GENES)  (CELLS)  [Seed]
32738     2967    362015

27 additional parameters

Batches:
[BATCHES]        [BATCH CELLS]       [Location]      [Scale]
    4        741, 742, 741, 743       0.1             0.1

Mean:
(RATE)              (SHAPE)
13.1862567561821   0.513625055001981

Library size:
(LOCATION)             (SCALE)
7.46950097836897   0.795782986717212

Exprs outliers:
(PROBABILITY)          (LOCATION)             (SCALE)
0.0192464082407156    5.13929002084815    0.994144591961186

Groups:
[GROUPS]          [GROUP PROBS]
    7    0.01, 0.25, 0.1, 0.05,...

Diff expr:
[Probability]     [Down Prob]   [Location]    [Scale]
     0.1              0.5          0.1          0.4

BCV:
(COMMON DISP)            (DOF)
0.291449770028786    28.9998030313448

Dropout:
[Present]             (MIDPOINT)                (SHAPE)
  FALSE      -0.0200665832600224      -1.02201072763585

Paths:
[From]     [Length]     [Skew]     [Non-linear]   [Sigma Factor]
  0          100          0.5           0.1              0.8
```

The obtained parameters could have been used to simulate the desired datasets, but we had to manipulate them to fit our needs. Most of the estimated parameters were simply taken over from the estimation or simplified by rounding and only the

following were adapted, due to our experimental setup of the verification process.

- Batches, were artificially added in four of eight simulated datasets denoted by starting with *sim_2_* and served the purpose of testing the confounding factor analysis capabilities of the workflow. We decided for six batches of equal sizes to mimc a real scenario as good as possible.
- Groups, or clusters were very important to see if our cluster analysis approach works. We decided for six clusters of different sizes, described by a probability of a cell being in a group or cluster, due to the fact that this resembles usually the real case.
- The biological coefficient of variation (BCV) for each gene in each cell describes the underlying common dispersion across all genes. This is the main parameter we used to make the data more difficult to analyze.
- Dropouts were added in the course of the simulation, because they represent a major challenge in scRNAseq analysis.

Table A.1 presents the most important and only parameters we manipulated, between datasets, and used for the simulation. We did not include the parameters [GROUPS] and [BATCHES], because they can be directly derived from the parameters [GROUP PROBS] and [BATCH CELLS], respectively.

dataset	[GROUP PROBS]	[BATCH CELLS]	[COMMON DISP]
sim_1_1	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	3000	0.1
sim_1_2	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	3000	0.5
sim_1_3	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	3000	1.2
sim_1_4	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	3000	0.3
sim_2_1	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	$500, 500, 500, 500, 500, 500$	0.1
sim_2_2	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	$500, 500, 500, 500, 500, 500$	0.5
sim_2_3	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	$500, 500, 500, 500, 500, 500$	1.2
sim_2_4	$0.01, 0.4, 0.1, 0.05, 0.3, 0.14$	$500, 500, 500, 500, 500, 500$	0.3

Table A.1.: Estimated and used simulation parameters.

As we can see in Table A.1 and by comparing the parameters with the estimated ones from above, apart from simulated datasets ending with *_1,* all the parameters are chosen to make it more difficult for the workflow to detect the correct populations within the data.

To conclude this part we present all of the final simulation parameters, which were used to generate the eight simulated datasets. We used the symbol || in the parameter listing to distinguish between the different parameters depending on the dataset.

```
A Params object of class SplatParams
Parameters can be (estimable) or [not estimable], 'Default' or 'NOT DEFAULT'.
Global:
(GENES)  (CELLS)  [Seed]
32000     3000    123456

27 additional parameters

Batches:
[BATCHES]          [BATCH CELLS]          [Location]    [Scale]
   1||6      3000||500, 500, 500, 500, 500, 500      0.1          0.1

Mean:
(RATE)    (SHAPE)
10        0.1

Library size:
(LOCATION)  (SCALE)
   11         0.2

Exprs outliers:
(PROBABILITY)  (LOCATION)   (SCALE)
     0.02           5           1

Groups:
[GROUPS]              [GROUP PROBS]
    6        0.01, 0.4, 0.1, 0.05, 0.3, 0.14

Diff expr:
[Probability]    [Down Prob]  [Location]   [Scale]
      0.1             0.5         0.1         0.4

BCV:
(COMMON DISP)          (DOF)
0.1||0.5||1.2||0.3       30

Dropout:
[Present] (MIDPOINT) (SHAPE)
  TRUE       0         -1

Paths:
[From]      [Length]      [Skew]     [Non-linear]  [Sigma Factor]
  0           100          0.5           0.1            0.8
```

After the simulation process we manipulated the count values by dividing them by 10 to ensure that the quality control metrics resembled the model dataset *pbmc3k* even more.

A.3. Configuration Parameters

In Table A.2 and A.3 we present the configuration parameters, which were used in the analysis of the above described datasets. The analyses themselves are presented, described and discussed in Chapter 3, 4.1 or 4.2, respectively.

Default (per Chapter 3.2) and the following three parameters are not mentioned in the tables, namely all flags, normalization method *[norm.method]* and genes to plot *[GenesToPlot]*.

- All flags, were set TRUE, due to the fact that the presented analyses always conducted a full run of every module, including its visualization part.
- For every dataset and normalization method we ran the entire analysis, therefore we did not state it in the tables.
- Genes to plot were only needed in the case of the *pbmc3k* dataset, due to the fact that the others were simulated and did not contain any real genes. The plotted genes, which were used for the identification of the found populations as cell types, are listed in Chapter 4.2 and can be seen in Figure 4.13.

parameter name / dataset	*pbmc3k*	*sim_1_1*	*sim_1_2*	*sim_1_3*	*sim_1_4*
input.type	10X	matrix	matrix	matrix	matrix
MT.pattern	^MT-	^MT-	^MT-	^MT-	^MT-
filter.cells.by.umi.min	1000	1000	1000	1000	1000
filter.cells.by.umi.max	5000	8000	8000	8000	8000
filter.cells.by.geneexpression	200	200	200	200	200
filter.cells.by.MTpct	5	100	100	100	100
filter.cells.by.ERCCpct	100	100	100	100	100
filter.gene.by.cellexpression	2	2	2	2	2
filter.gene.by.no.-expression.per.cell	2	2	2	2	2
cf.blacklist	NA	Group	Group	Group	Group
read10X_min_-total_cell_counts	100	NA	NA	NA	NA
cf.number.variable.features	1000	1000	1000	1000	1000
no.of.clusters	7	7	7	7	7
pcs.use	10	10	10	10	10
ValueOfInterest	NA	Group	Group	Group	Group
minClusterSize.pct	1	1	1	1	1
n.top.clusterings	15	20	20	20	20
comparison	sol1,sol2	Group	Group	Group	Group

Table A.2.: Configuration parameters for the analysis of the datasets *pbmc3k*, *sim_1_1*, *sim_1_2*, *sim_1_3* and *sim_1_4*.

parameter name / dataset	*sim_2_1*	*sim_2_2*	*sim_2_3*	*sim_2_4*
input.type	matrix	matrix	matrix	matrix
MT.pattern	^MT-	^MT-	^MT-	^MT-
filter.cells.by.umi.min	1000	1000	1000	1000
filter.cells.by.umi.max	8000	8000	8000	8000
filter.cells.by.geneexpression	200	200	200	200
filter.cells.by.MTpct	100	100	100	100
filter.cells.by.ERCCpct	100	100	100	100
filter.gene.by.cellexpression	2	2	2	2
filter.gene.by.no.-expression.per.cell	2	2	2	2
cf.blacklist	Group	Group	Group	Group
read10X_min_-total_cell_counts	NA	NA	NA	NA
cf.number.variable.features	1000	1000	1000	1000
no.of.clusters	7	7	7	7
pcs.use	10	10	10	10
ValueOfInterest	Group, Batch	Group, Batch	Group, Batch	Group, Batch
minClusterSize.pct	1	1	1	1
n.top.clusterings	20	20	20	20
comparison	Group	Group	Group	Group

Table A.3.: Configuration parameters for the analysis of the datasets *sim_2_1*, *sim_2_2*, *sim_2_3* and *sim_2_4*.

List of Figures

List of Tables

R Packages

cidr	v.0.1.5 “Ultrafast and accurate clustering through imputation for single-cell RNA-seq data” - [LTH17]
clusterCrit	v.1.2.7 “Compute clustering validation indices.” - [Des16]
clusterExperiment	v.1.4 “Compare Clusterings for Single-Cell Sequencing” - [PRJ17]
corrplot	v.0.84 “A graphical display of a correlation matrix or general matrix.” - [WS17]
DelayedMatrixStats	v.1.0.2 “Functions that Apply to Rows and Columns of ’DelayedMatrix’ Objects” - [Hic18]
limma	v.3.34.9 “limma is an R/Bioconductor software package that provides an integrated solution for analysing data from gene expression experiments.” - [RPW+15]
Matrix	v.1.2-12 “Sparse and Dense Matrix Classes and Methods.” - [BM17]
MCDM	v.1.2 “Multi-Criteria Decision Making Methods for Crisp Data” - [Mar16]
mclust	v.5.4 “mclust 5: clustering, classification and density estimation using Gaussian finite mixture models” - [SFMR17]
NMF	v.0.23.6 “Provides a framework to perform Non-negative Matrix Factorization (NMF).” - [GS18] and [GS10]
NMI	v.2.0 “Normalized Mutual Information of Community Structure in Network” - [Wu16]
nnet	v.7.3-12 “Software for feed-forward neural networks with a single hidden layer, and for multinomial log-linear models.” - [VR02]
pcaReduce	v.1.0 “Hierarchical clustering of single cell transcriptional profiles” - [žY16]

SC3	v.1.7.7 “Consensus clustering of single-cell RNA-seq data” - [KKS+17]
scater	v.1.6.2 “Pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R” - [MCLW17]
scran	v.1.6.6 “Implements functions for low-level analyses of single-cell RNA-seq data.” - [LMM16]
Seurat	v.2.2.0 “A toolkit for quality control, analysis, and exploration of single cell RNA sequencing data.” - [SBH+18] and [BHS+18]
SingleCellExperiment	v.1.0.0 “Defines a S4 class for storing data from single-cell experiments.” - [LRK17]
splatter	v.1.1.8 “Splatter is a package for the simulation of single-cell RNA sequencing count data” - [ZPO17]
stats	v.3.6.0 “This package contains functions for statistical calculations and random number generation.” - [R C17]

Bibliography

[10x16a] 10x Genomics. pbmc3k - Datasets - Single Cell Gene Expression - Official 10x Genomics Support. 2016, https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.1.0/pbmc3k.

[10x16b] 10x Genomics. Single Cell 3 ' Solution. 2016, https://www.10xgenomics.com/single-cell/.

[10X17] 10X Genomics. Removal of Dead Cells from Single Cell Suspensions Improves Performance for 10x Genomics ® Single Cell Applications. Technical report, 2017.

[ABM] ABM Inc. NGS - Introduction | ABM Inc.

[BHS+18] Andrew Butler, Paul Hoffman, Peter Smibert, Efthymia Papalexi, and Rahul Satija. Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nature Biotechnology*, 2018, https://www.nature.com/articles/nbt.4096.

[BM17] Douglas Bates and Martin Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2017.

[Cze12] Scott A Czepiel. Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation. *Class Notes*, pages 1–23, 2012, papers3://publication/uuid/4E1E1B7E-9CAC-4570-8949-E96B51D9C91D.

[Des16] Bernard Desgraupes. *clusterCrit: Clustering Indices*, 2016.

[FLNB17] Saskia Freytag, Ingrid Lonnstedt, Milica Ng, and Melanie Bahlo. Cluster Headache: Comparing Clustering Tools for 10X Single Cell Sequencing Data. page 203752, oct 2017, https://www.biorxiv.org/content/early/2017/10/19/203752.

[Gri00] Karl Grill. Skriptum zur Vorlesung "Mathematical Statistics". 2000, https://institute.tuwien.ac.at/fileadmin/t/mathstoch/upload/ms0.pdf.

[GS10] Renaud Gaujoux and Cathal Seoighe. A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, 11, 2010.

[GS18] Renaud Gaujoux and Cathal Seoighe. *The package NMF: manual pages*, 2018.

[Guj04] Damodar N. Gujarati. *Basic Econometrics*. 4 edition, 2004.

[Hic18] Peter Hickey. *DelayedMatrixStats: Functions that Apply to Rows and Columns of 'DelayedMatrix' Objects*, 2018.

[Ill17] Illumina. An Introduction to Next-Generation Sequencing Technology. Technical Report 2, 2017.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.

[KKS+17] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, and Martin Hemberg. SC3: Consensus clustering of single-cell RNA-seq data. *Nature Methods*, 14(5): 483–486, 2017, https://www.nature.com/articles/nmeth.4236.pdf.

[Kul16] Jerzy K. Kulski. Next-Generation Sequencing - An Overview of the History, Tools, and "Omic" Applications. 2016, http://dx.doi.org/10.5772/61964 http://www.intechopen.com/books/colitis.

[LMM16] Aaron T L Lun, Davis J McCarthy, and John C Marioni. A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor. *F1000Res.*, 5: 2122, 2016.

[LRK17] Aaron Lun, Davide Risso, and Keegan Korthauer. SingleCellExperiment: S4 Classes for Single Cell Data, 2017.

[LTH17] Peijie Lin, Michael Troup, and Joshua W.K. Ho. CIDR: Ultrafast and accurate clustering through imputation for single-cell RNA-seq data. *Genome Biology*, 18(1), 2017, https://genomebiology.biomedcentral.com/track/pdf/10.1186/s13059-017-1188-0?site=genomebiology.biomedcentral.com.

[Mar16] Blanca A Ceballos Martin. *MCDM: Multi-Criteria Decision Making Methods for Crisp Data*, 2016.

[McF73] Daniel McFadden. Conditional logit analysis of qualitative choice behavior. 1973.

[MCLW17] Davis J. McCarthy, Kieran R. Campbell, Aaron T.L. Lun, and Quin F. Wills. Scater: Pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R. *Bioinformatics*, 33(8): 1179–1186, 2017.

[Nat15] National Human Genome Research Institute (NHGRI). All About The Human Genome Project (HGP) - National Human Genome Research Institute (NHGRI), 2015.

[Nat16] National Human Genome Research Institute (NHGRI). Newborn Screening Fact Sheet - National Human Genome Research Institute (NHGRI), 2016.

[NFM03] Donald D. Newmeyer and Shelagh Ferguson-Miller. Mitochondria: Releasing power for life and unleashing the machineries of death. *Cell*, 112(4): 481–490, 2003.

[Pha03] Hoang Pham. Software reliability and cost models: Perspectives, comparison, and practice. *European Journal of Operational Research*, 149(3): 475–489, 2003.

[PRJ17] Elizabeth Purdom, Davide Risso, and Marla Johnson. clusterExperiment: Compare Clusterings for Single-Cell Sequencing, 2017.

[PZKS12] Yi Peng, Yong Zhang, Gang Kou, and Yong Shi. A Multicriteria Decision Making Approach for Estimating the Number of Clusters in a Data Set. *PLoS ONE*, 7(7), 2012.

[R C17] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2017.

[Rei18] Stephan Reichl. *Mathematical Methods in Single Cell RNA Sequencing Analysis with an Emphasis on the Validation of Clustering Results*. Technische Universität Wien, Vienna, 2018.

[RPW+15] Matthew E. Ritchie, Belinda Phipson, Di Wu, Yifang Hu, Charity W. Law, Wei Shi, and Gordon K. Smyth. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic acids research*, 43(7): e47, 2015.

[RTL+17] Aviv Regev, Sarah A Teichmann, Eric S Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, Hans Clevers, Bart Deplancke, Ian Dunham, James Eberwine, Roland Eils, Wolfgang Enard, Andrew Farmer, Lars Fugger, Berthold Göttgens, Nir Hacohen, Muzlifah Haniffa, Martin Hemberg, Seung Kim, Paul Klenerman, Arnold Kriegstein, Ed Lein, Sten Linnarsson, Emma Lundberg, Joakim Lundeberg, Partha Majumder, John C Marioni, Miriam Merad, Musa Mhlanga, Martijn Nawijn, Mihai Netea, Garry Nolan, Dana Pe'er, Anthony Phillipakis, Chris P Ponting, Stephen Quake, Wolf Reik, Orit Rozenblatt-Rosen, Joshua Sanes, Rahul Satija, Ton N Schumacher, Alex Shalek, Ehud Shapiro, Padmanee Sharma, Jay W Shin, Oliver Stegle, Michael Stratton, Michael J T Stubbington, Fabian J Theis, Matthias Uhlen, Alexander van Oudenaarden, Allon Wagner, Fiona Watt, Jonathan Weissman, Barbara Wold, Ramnik Xavier, Nir Yosef, and Human Cell Atlas Meeting Participants. The Human Cell Atlas. *eLife*, 6: e27041, 2017, https://elifesciences.org/articles/27041.

[Sat18] Satija Lab. Seurat - Guided Clustering Tutorial. 2018, https://satijalab.org/seurat/pbmc3k_tutorial_1_4.html.

[SBH+18] Rahul Satija, Andrew Butler, Paul Hoffman, Jeff Farrell, Shiwei Zheng, Christoph Hafemeister, and Patrick Roelli. Package "Seurat", 2018.

[SFMR17] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery. mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. *The R Journal*, 8(1): 205–233, 2017, https://cran.r-project.org/package=mclust https://journal.r-project.org/archive/2017/RJ-2017-008/RJ-2017-008.pdf.

[Shl03] Jon Shlens. A tutorial on principal component analysis: derivation, discussion and singular value decomposition. 2: 1–16, 2003, https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf www.snl.salk.edu/ shlens/pca.pdf.

[STM15] Oliver Stegle, Sarah A. Teichmann, and John C. Marioni. Computational and analytical challenges in single-cell transcriptomics. *Nature Reviews Genetics*, 16(3): 133–145, 2015, http://dx.doi.org/10.1038/nrg3833.

[TÅW+17] Peter J. Thul, Lovisa Åkesson, Mikaela Wiking, Diana Mahdessian, Aikaterini Geladaki, Hammou Ait Blal, Tove Alm, Anna Asplund, Lars Björk, Lisa M. Breckels, Anna Bäckström, Frida Danielsson, Linn Fagerberg, Jenny Fall, Laurent Gatto, Christian Gnann, Sophia Hober, Martin Hjelmare, Fredric Johansson, Sunjae Lee, Cecilia Lindskog, Jan Mulder, Claire M. Mulvey, Peter Nilsson, Per Oksvold, Johan Rockberg, Rutger Schutten, Jochen M. Schwenk, Åsa Sivertsson, Evelina Sjöstedt, Marie Skogs, Charlotte Stadler, Devin P. Sullivan, Hanna Tegel, Casper Winsnes, Cheng Zhang, Martin Zwahlen, Adil Mardinoglu, Fredrik Pontén, Kalle von Feilitzen, Kathryn S. Lilley, Mathias Uhlén, and Emma Lundberg. A subcellular map of the human proteome. *Science*, 356(6340): eaal3321, 2017, https://www.proteinatlas.org/.

[VCLE+15] K. Verhoeckx, P. Cotter, I. López-Expósito, C. Kleiveland, T. Lea, A. Mackie, T. Requena, D. Swiatecka, and H. Wichers. *The Impact of Food Bioactives on Health: In Vitro and Ex Vivo Models*. 2015.

[VH08] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9: 2579–2605, 2008, http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf.

[VR02] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, 4 edition, 2002.

[Wan01] X Wang. The expanding role of mitochondria in apoptosis. *Gene Dev*, 15(214): 2922–33, 2001.

[WS17] Taiyun Wei and Viliam Simko. R package "corrplot": Visualization of a Correlation Matrix, 2017.

[Wu16] Tianhao Wu. *NMI: Normalized Mutual Information of Community Structure in Network*, 2016.

[WW07] Silke Wagner and Dorothea Wagner. Comparing Clusterings - An Overview. *Analysis*, 4769(001907): 1–19, 2007, https://publikationen.bibliothek.kit.edu/1000011477.

[ZPO17] Luke Zappia, Belinda Phipson, and Alicia Oshlack. Splatter: simulation of single-cell RNA sequencing data. *Genome Biology*, 2017, http://dx.doi.org/10.1186/s13059-017-1305-0.

[ZTB+17] Grace X.Y. Zheng, Jessica M. Terry, Phillip Belgrader, Paul Ryvkin, Zachary W. Bent, Ryan Wilson, Solongo B. Ziraldo, Tobias D. Wheeler, Geoff P. McDermott, Junjie Zhu, Mark T. Gregory, Joe Shuga, Luz Montesclaros, Jason G. Underwood, Donald A. Masquelier, Stefanie Y. Nishimura, Michael Schnall-Levin, Paul W. Wyatt, Christopher M. Hindson, Rajiv Bharadwaj, Alexander Wong, Kevin D. Ness, Lan W. Beppu, H. Joachim Deeg, Christopher McFarland, Keith R. Loeb, William J. Valente, Nolan G. Ericson, Emily A. Stevens, Jerald P. Radich, Tarjei S. Mikkelsen, Benjamin J. Hindson, and Jason H. Bielas. Massively parallel digital transcriptional profiling of single cells. *Nature Communications*, 8: 14049, jan 2017, http://www.nature.com/doifinder/10.1038/ncomms14049.

[žY16] Justina žurauskiene and Christopher Yau. pcaReduce: Hierarchical clustering of single cell transcriptional profiles. *BMC Bioinformatics*, 17(1), 2016.