Batch effect correction methods improve the evaluation of clusterability in single-cell transcriptomics data

Schmeits, C.J.

Batch effect correction methods improve the evaluation of clusterability in single-cell transcriptomics data

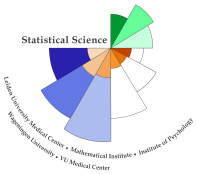
Caspar Schmeits (s1393871)

Internal supervisor: Dr. Szymon Kiełbasa (LUMC)
External supervisor: Dr. Stefan Semrau (LION)
Daily supervisor: Maria Mircea MSc (LION)

MASTER THESIS

Defended in October 2021

Specialization: Statistical Science



STATISTICAL SCIENCE
FOR THE LIFE AND BEHAVIOURAL SCIENCES

Abstract

In the past decade, experimental developments in the field of transcriptomics have enabled researchers to measure gene expression at the level of single cells, leading to a great increase in measurement resolution. Clustering the cells according to their gene expression profiles can aid the discovery of novel cell types. Unfortunately, it is often difficult to tell whether the established clusters are homogeneous or if sub-clustering might be possible. Recently, a new clusterability measure has been developed that aims to quantify the heterogeneity of gene expression within clusters. The so-called *SIGnal Measurement Angle* (SIGMA) is based on a result from random matrix theory which states that the singular values of a random matrix follow a known probability distribution. Singular values that strongly deviate from this distribution are likely to be caused by deterministic sources of variability, such as differences between cell types. However, the heterogeneity may also be caused by unwanted technical sources of variance, such as batch effects, which arise when data from several experiments are combined. Various methods exist for batch effect correction, but it is yet unclear whether they reduce batch effects to the extent that their effect on SIGMA is sufficiently eliminated. In this thesis, we compared the efficacy of three different batch correction methods (fastMNN, Harmony, and Seurat) on simulated data and on two empirical data sets. Their effectiveness was evaluated with several batch mixing metrics as well as by inspecting the singular values and vectors of the batch-corrected expression matrices. In conclusion, both fastMNN and Harmony worked well in most cases, but with unbalanced data sets (i.e., when one or more cell types were absent in one of the batches), it became increasingly difficult to decide whether singular values were batch effect-associated, because in those cases the batch effect also contained biological heterogeneity.

Foreword

With this thesis, I (hope to) have contributed to the investigation of batch effects in single-cell RNA-sequencing in relation to a newly developed clusterability measure. To this end, I have come up with ways to associate singular values of the expression matrix with batch effects, applied a new batch mixing metric in addition to existing metrics, and compared the effectiveness of three different batch effect correction methods on simulated and empirical data.

Data sources

Two data sets were analysed in this thesis. The human dendritic cell data were downloaded from the NCBI Gene Expression Omnibus (GEO) with accession number GSE94820 (Link: `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE94820`). The human lymphocyte data were kindly provided by Dr. Kees van Bergen from the Leiden University Medical Center (LUMC) and are not publicly available; the patients from which the samples were taken have been anonymized for the analysis.

Availability of R code

The R code that was used for generating the figures in this thesis is available in my GitHub repository: `https://github.com/Capspar/StatSci-Thesis`.

Acknowledgements

I would like to thank my supervisors Dr. Stefan Semrau and Dr. Szymon Kiełbasa for their valuable advice throughout the course of my thesis project. I especially want to thank my daily supervisor Maria Mircea MSc for her supervision and support, and for the good times at the office and outside of it. Also, I wish to express my gratitude to my parents for supporting me, financially and otherwise, and to all my friends and housemates who brought me social relief while carrying out this project during these unusual times. Finally, I want to thank my brother Vincent, who inspires me.

Contents

Chapter 1

Introduction

1.1 Gene expression and regulation

For increasing our understanding of many biological processes, it is at least as important to study *gene expression* as it is to study *genes*. Genes, which are stored in the form of DNA, contain the genetic information that is necessary for building proteins and other functional molecules. However, to build proteins, genes first have to be copied from DNA into *messenger RNA* (mRNA) by a process called *transcription*, after which the mRNA is used to create proteins, a process known as *translation*. Transcription and translation together may be referred to as gene expression. In living organisms, gene expression is typically heavily regulated, and for good reason, because not all gene products are needed everywhere at all times. For example, during the embryonic development of multicellular organisms, it is crucial that developmental genes are expressed only in specific regions of the developing body at specific points in time.

As different parts of an organism have different functions, the types of cells making up these parts are specialized for serving specific purposes. Cell types differ not only in terms of their structure, but also in terms of which genes they express. For instance, red blood cells are used to carry oxygen through the blood stream, and therefore, developing red blood cells express genes for making hemoglobin, a protein that binds oxygen molecules. White blood cells, on the other hand, are not involved in oxygen transport but are part of the immune system, so they do not express hemoglobin.

Cells utilize several mechanisms to regulate gene expression. Some of these mechanisms chemically modify the DNA in certain places (DNA methylation) or affect how tightly the DNA is packed together (histone modifications), thereby adjusting its accessibility for transcription. Additionally, proteins called transcription factors are able to bind to specific nucleotide sequences in the DNA in order to promote or repress the transcription of particular genes.

Analyzing gene expression, and especially changes in gene expression, is very useful for studying development and disease. In stem cell research, it has been shown that differentiating cells undergo shifts in gene expression as they develop into new cell types [30]. Also, the observation that cells of the same type form clusters in the gene expression space has led to the discovery of previously unknown cell types [22]. In medical studies, it is often of interest how gene expression is altered in diseased subjects compared to a control group, and the same holds for effects of drug treatments. In recent years, it has become possible to measure the gene expression of many genes and cells at the same time, sparking the emergence of the research field of *single-cell transcriptomics*.

1.2 Single-cell transcriptomics

Transcriptomics is the study of the *transcriptome*, the collection of all RNA transcripts in a cell, tissue, or organism at a certain point in time [12]. The transcriptome consists of messenger RNA (mRNA) molecules, which are intermediary molecules for producing proteins, as well as non-coding RNAs (e.g., ribosomal RNAs), which perform a number of functions inside cells.

The first attempts at measuring transcriptomes were made in the 1990s. Although several techniques have been developed over the years, the ever decreasing cost of high-throughput sequencing technologies has made RNA-seq the most popular contemporary technique for transcriptome quantification. RNA-seq involves extracting RNA molecules from a biological sample and sequencing them with a DNA sequencer [12]. In order for this to work, the RNA molecules have to be converted into complementary DNA (cDNA) molecules, which can be achieved with an enzyme called *reverse transcriptase*. Before sequencing, the cDNA is fragmented into smaller pieces and, optionally, labeled with short, random nucleotide sequences called *unique molecular identifiers* (UMIs) [8]. Subsequently, the cDNA is amplified (multiplied) to obtain larger amounts, which are more easily detectable. After sequencing, the resulting *reads* are aligned to a reference genome to determine from which genes they originate. If no reference genome exists, the sequencing reads can be aligned to each other (this is known as *de novo* assembly). The number of reads that map to each gene represents the abundance of RNA transcripts of that gene at the time the sample was taken. If the DNA fragments were labeled with UMIs, the exact number of detected RNA transcripts can be derived. In the former case, we denote the abundances as *read counts*, and in the latter case as *UMI counts*.

Originally, RNA-seq was applied on RNA samples extracted from many cells at the same time, effectively averaging gene expression over all the cells in the sample. This approach is now referred to as *bulk RNA-seq*. Soon, efforts were made to capture and sequence RNA transcripts from individual cells separately (*single-cell RNA-seq* or *scRNA-seq*). This development faced two major challenges [9]: (1) the capture of individual cells, and (2) the amplification of very small amounts of RNA from an individual cell. Cells can be captured manually [27], but this is very laborious and is only practical for samples consisting of few cells, such as early embryos. For larger-scale experiments, microfluidic devices have been created that either capture cells in wells on a well plate [11] or in tiny droplets inside an oil suspension [18]. Every well or droplet is supplied with enzymes that lyse the cell, and the RNA is fragmented and tagged with a cell-specific nucleotide sequence known as a *barcode*, which identifies the cell from which the RNA molecules originate. As described before, UMIs may be added to the fragments as well before the RNA is reverse-transcribed and amplified. Then, all RNA molecules are pooled together, sequenced, aligned, and counted, resulting in a matrix of read or UMI counts.

The advent of scRNA-seq has enabled researchers to study gene expression patterns in much greater detail. After all, gene expression can vary considerably between cells inside a tissue, depending on their cell types or states. Not only can gene expression profiles be used for identifying known cell types, they may also reveal new, hitherto unknown cell types. Another application of scRNA-seq is in the characterization of shifting expression patterns in differentiating cells [30].

1.3 Single-cell RNA-seq data analysis

A scRNA-seq data set consists of a matrix of read or UMI counts. Typically, the rows represent genes and the columns represent cells. For human cells, the number of protein coding genes is around 20 000 [6], so a number of rows in that order of magnitude can be expected. The number of columns (cells) obviously depends on the experiment and tissue, but can nowadays mount up to more than a million in extreme cases [35].

The processing of scRNA-seq data generally involves a number of steps [13]. First, cells and genes of low quality need to be removed, an operation known as *quality control* (QC). In this case, low quality means that a measurement does not reflect a biological state of interest due to technical or biological causes. For example, a cell may have been already dead at the moment it was captured, or the droplet or well contained two cells instead of one (such wells or droplets are called *doublets*). Cells are deemed to be of low quality if their *library size* (i.e., the total number of counts per cell) is very small or very large, or if they have a low number of detected genes. Other criteria, such as the proportion of mitochondrial gene counts relative to the library size, can be utilized as well. A high proportion of mitochondrial gene expression may indicate that the cell is missing cytoplasmic RNA that may have leaked out before cell capture, or that a cell is stressed. For gene quality control, genes that have been detected in at most a few cells are not informative and can also be removed.

The next step in the usual workflow is *normalization*, which aims to cancel out differences in the library sizes of cells that arose because of the sampling procedure [13]. After normalization, the data are generally log-transformed.

Often, we want to cluster the data into different cell types or cell states. Since clustering methods usually benefit from *dimension reduction* due to the elimination of uninformative noise, it is common to perform *feature selection* first [13]. Feature selection consists of selecting a number or proportion of genes that contribute the most to the variance between cells. Then, the number of dimensions can be brought further down by principal component analysis (PCA) or other dimension reduction techniques. Currently, the most commonly used clustering techniques for scRNA-seq data involve graph-based clustering, for example the Leiden algorithm [28].

Given that the resulting clusters represent true biological states, we want to find out to which cell type each cluster belongs. This is called *annotation* and can be achieved by identifying differentially expressed genes between clusters and comparing these genes to marker genes that are known from the literature to characterize certain cell types [13]. Using the annotated clusters, further types of analyses may include assessing the differences in cell type proportions between two or more subjects [13], differential expression analysis, or in the case of stem cell differentiation, trajectory inference (TI) [30]. The latter involves analyzing the changes in gene expression that cells undergo when they develop and differentiate into other cell types.

1.4 Batch effects

Whenever a scRNA-seq data set consists of subsets that have been generated under different technical and/or biological circumstances, it is almost always subject to artefacts known as *batch effects* [29]. Because the process of generating scRNA-data involves many steps and manipulations, the count matrix obtained at the end of the process is the product of a myriad of factors. If cell capture, reverse transcription, amplification, and sequencing are performed on separate batches of cells, slight differences in any of the steps introduce unwanted systematic differences between batches.

If not dealt with appropriately, batch effects influence the previously mentioned data processing steps. To begin with, quality control should generally be performed on the separate batches instead of the data as a whole, because outlier cells in one batch may be masked by other batches. Feature selection should also be done separately for each batch, as the variance introduced by batch effects could otherwise lead to the selection of genes that vary the most between batches instead of between cells within batches. Furthermore, if batch effects still persist during clustering, cells will be more prone to be clustered by their batch of origin rather than by their cell type.

Batch effects have been an issue since the introduction of microarrays (an older technique for measuring gene expression), and a number of methods have been developed to correct for them. Early methods, such as limma [25], used linear models to model batch effects in microarray data and remove them. However, the nonlinear nature of batch effects in scRNA-seq data has required the development of dedicated batch correction algorithms. Three such algorithms are compared in this thesis: fastMNN [5], Harmony [10], and Seurat [3]. All three of these methods estimate and correct for batch effects in local neighborhoods of the data, and can therefore be viewed as locally linear correction algorithms.

1.5 Batch effects and a new clusterability measure

When clustering data, it is always a question whether we are not clustering too coarsely or too finely: if there are too few clusters, some subpopulations remain hidden. On the other hand, if there are too many clusters, then some clusters are simply based on random noise. Recently, a clusterability measure has been developed that aims to answer this question: the signal measurement angle (SIGMA) [20]. Given a computed cluster, it will tell you whether subclustering may be necessary. The measure is based on the assumption that a scRNA-seq data matrix is the sum of a random noise matrix and a deterministic signal matrix. Briefly, SIGMA uses information about the distribution of singular values to evaluate the strength of the signal compared to the noise.

Since batch effects are deterministic signals, they have to be reckoned with when assessing clusterability with SIGMA. The objective of this thesis is to investigate how batch effects are manifested in the singular values and vectors of scRNA-seq count matrices, and whether batch effect correction methods can adequately remove the influence of batch effects from the singular values/vectors (and thus from SIGMA). We will begin by introducing SIGMA and the three batch effect correction algorithms fastMNN, Harmony, and Seurat. Then, we will examine how batch effects influence the singular values and vectors of several simulated scRNA-seq matrices and apply batch correction. Additionally, batch correction will be applied to two real data sets in order to conclude whether they sufficiently remove batch effects in light of SIGMA.

Chapter 2

Methods

2.1 Random matrix theory

As is typically the case with biological data, scRNA-seq data can be very noisy, meaning that a substantial amount of random variation exists between observations. It turns out that, after the appropriate normalization and scaling steps, scRNA-seq matrices largely obey certain properties derived from random matrix theory. One of these properties is the probability distribution of singular values, which is known. In the following subsections, singular value decomposition and its relation to random matrices and to SIGMA is introduced.

Singular value decomposition

Any $m \times n$ matrix $\mathbf{X}$ can be decomposed into a matrix product $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ known as the *singular value decomposition* (SVD), where $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n)$ is orthogonal and contains the *left singular vectors* $\mathbf{u}_1, \ldots, \mathbf{u}_n$, $\mathbf{V} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$ is orthogonal and contains the *right singular vectors* $\mathbf{v}_1, \ldots, \mathbf{v}_n$, and $\mathbf{\Sigma}$ is a diagonal matrix containing the *singular values* $\sigma_1, \ldots, \sigma_n$ such that $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. The number of nonzero singular values is equal to the rank of $\mathbf{X}$.

The Marchenko-Pastur distribution

A matrix with random variables as entries is known as a *random matrix*. It has been shown that under certain assumptions, the singular values of a random matrix follow a Marchenko-Pastur distribution: let $\mathbf{X}$ be an $m \times n$ matrix containing independent and identically distributed random variables with mean 0 and variance 1. Then, for $m, n \to \infty$ and $1 < \frac{m}{n} < \infty$, the density of the singular values of $\mathbf{X}$ is

$$\rho(\sigma; c) = \begin{cases} \frac{\sqrt{(b-\sigma^2)(\sigma^2-a)}}{\pi\sigma c}, & \text{if } a \leq \sigma^2 \leq b \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

where $c = \frac{m}{n}$, $a = (1 - \sqrt{c})^2$, and $b = (1 + \sqrt{c})^2$ [20].

In the case of single-cell RNA-seq data, an expression matrix $\tilde{\mathbf{X}}$ can be regarded as the sum of a random *noise matrix* $\mathbf{X}$ and a *signal matrix* $\mathbf{P}$. The signal matrix $\mathbf{P}$ should ideally contain all the relevant biological information with respect to the cell state, that is, fold changes of differentially expressed (DE) genes between cell types. Unfortunately, it is impossible to disentangle $\mathbf{P}$ from $\tilde{\mathbf{X}}$. However, if after the appropriate scaling steps we were to examine the distribution of the singular values of $\tilde{\mathbf{X}}$, we would find that most of the singular values are concentrated together. We call this the bulk distribution. Additionally, if the signal is strong enough, there will be at

least one singular value positioned above the bulk as a result of the signal matrix (Figure 2.1). For the effect that $\mathbf{P}$ has on the randomness of $\tilde{\mathbf{X}}$, $\tilde{\mathbf{X}}$ is said to be *perturbed* by $\mathbf{P}$. Usually, the singular values resulting from a signal appear above the bulk distribution. The magnitude of a singular value that results from the signal matrix represents the strength of the signal.

Note that we made some assumptions for Equation 2.1 to hold. In particular, we assumed that the dimensions of the random matrix are infinite. Of course, this is never the case for scRNA-seq expression matrices, but it turns out that even for random matrices with finite but relatively large dimensions, the distribution of singular values will approximate the MP distribution very well. There is, however, a slight chance that a singular value will exceed $\sqrt{b}$, the upper boundary of the bulk distribution. This probability is known and is described by the Tracy-Widom distribution [20]. In practice, strong signals (i.e., if there is a strong degree of differential gene expression) are always located far above the bulk distribution).

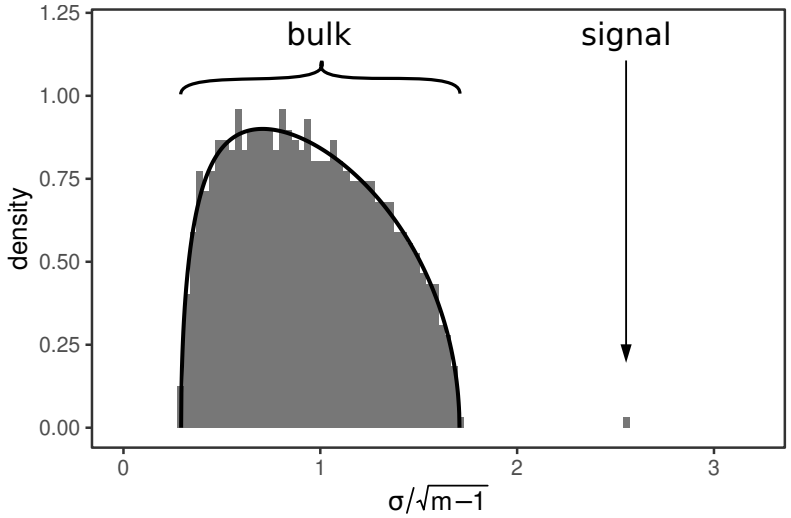


Figure 2.1: Singular values of a perturbed random matrix. The random 2000×1000 matrix $\tilde{\mathbf{X}}$ with entries $\tilde{x}_{ij} \sim \mathcal{N}(0,1)$ has been perturbed by a rank-1 matrix $\mathbf{P} = \mathbf{u}\mathbf{v}^T$, where $\mathbf{u} = (1,1,\ldots,1)^T$ is a 2000×1 vector of ones and $\mathbf{v} = (0,0,\ldots,0,0.1,0.1,\ldots,0.1)^T$ is a 1000×1 vector with 500 zeros and 500 values of 0.1. When divided by $\sqrt{2000-1}$, the majority of singular values form a compact bulk distribution that follows the theoretical Marchenko-Pastur density with parameter $c = \frac{2000}{1000} = 2$. However, as a result of the perturbation, one singular value deviates from the bulk distribution. This singular value is denoted as a signal.

The signal measurement angle (SIGMA)

Low-rank perturbation theory allows us to make certain statements about the signal matrix $\mathbf{P}$ [1, 20]. First of all, it is possible to derive the singular values of $\mathbf{P}$ from the largest singular values of $\tilde{\mathbf{X}}$. Secondly, although we do not know the singular vectors of $\mathbf{P}$, there is a relationship between each singular value of $\mathbf{P}$ and the angle between the corresponding right singular vectors of $\tilde{\mathbf{X}}$ and $\mathbf{P}$ [20]:

$$\langle \tilde{\mathbf{v}}_i, \mathbf{v}_i \rangle^2 \xrightarrow{\text{a.s.}} \begin{cases} 1 - \frac{c\left(1+\theta_i^2\right)}{\theta_i^2\left(\theta_i^2+c\right)} & \text{if } \theta_i \geq c^{1/4} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Here, $\tilde{\mathbf{v}}_i$ and $\mathbf{v}_i$ are the ith right singular vectors of $\tilde{\mathbf{X}}$ and $\mathbf{P}$, respectively, and θ_i is the ith singular value of $\mathbf{P}$. Since the norms of the singular vectors are equal to 1, the inner product $\langle \tilde{\mathbf{v}}_i, \mathbf{v}_i \rangle$ is equal to the cosine of the angle between $\tilde{\mathbf{v}}_i$ and $\mathbf{v}_i$.

We define the *signal measurement angle* (SIGMA) as $\langle \tilde{\mathbf{v}}_1, \mathbf{v}_1 \rangle^2$, the squared cosine of the angle between the first right singular vectors of $\tilde{\mathbf{X}}$ and $\mathbf{P}$, corresponding to the largest singular value of $\mathbf{P}$, θ_1. From Eq. 2.2, it can be observed that SIGMA lies between 0 and 1, and that the larger θ_1, the closer SIGMA will be to 1. In this way, SIGMA relates the strength of the perturbation signal $\mathbf{P}$ to a quantity between 0 and 1.

2.2 Batch effect correction methods

Since the advent of single-cell sequencing, a number of methods have been developed to deal with batch effects. Such methods are necessary because we want to be able to pool data together for analysis in order to achieve greater resolution and statistical power for answering research questions. Batch correction usually consists of taking one batch as a *reference* batch, and correcting the expression values in the *target* batches in such a way as to match the expression in the reference batch. Initially, batch correction methods from bulk RNA-seq and microarray data analysis, such as limma [25] and ComBat [7], were used for scRNA-seq data as well. For example, limma consists of a simple linear model that estimates batch effect coefficients and sets them to zero, so that only the other covariates and the residuals remain. However, while bulk RNA-seq deals with the average gene expression over many cells in a tissue, scRNA-seq measures a separate gene expression profile for each cell, which greatly increases the resolution of the measurement. It also means that batch effects manifest themselves on a single-cell level and are not averaged out anymore over a whole tissue, leading to more complex batch effects if different types of cells are differently affected by batch effects. This requires novel methods that specifically deal with scRNA-seq data.

I chose to compare three methods that are generally considered good [29]: fastMNN, Seurat Integration, and Harmony. In the following subsections, I will describe each of these methods in detail. Note that all descriptions below deal with the integration of two batches. When working with multiple batches, the default strategy of all three algorithms is to start by integrating the first two batches (according to the input order), then to integrate the third batch with the result of the first integration, and so on. The order of batch integration can matter, and can usually be configured by the user in most algorithms.

FastMNN

FastMNN is an improved version of the MNNCorrect algorithm developed by Haghverdi et al. (2018) [5]. MNNCorrect relies on finding *mutual nearest neighbors* (MNNs) between two batches. For each cell in batch 1, its k nearest neighbors in batch 2 are computed, and *vice versa*. Two cells, one from each batch, are considered an MNN pair if they are contained in each others' sets of nearest neighbors. The underlying assumption is that MNN pairs between the batches represent two cells of the same cell type or state. When all pairs of MNNs have been identified, pair-specific batch effect vectors are computed by taking the difference between the expression vectors of each MNN pair. Next, cell-specific correction vectors are determined by Gaussian kernel smoothing. That is, the correction vector for each cell will be a weighted average of the batch correction vectors of all MNN pairs to which the cell belongs and of the surrounding MNN pairs. The advantage of this approach over bulk RNA-seq methods is that it can handle batches with unequal cell type compositions, because in the case that a batch contains a cell type that

is not present in other batches, no MNN pairs will be formed, and the correction vector for such cells is only based on the smoothing effect from surrounding MNN pairs.

FastMNN improves the MNNCorrect algorithm by first performing principal component analysis (PCA) on the normalized expression values and and then applying the correction in the principal component space. Not only does this result in a much shorter computation time (particularly for the nearest neighbor search), it also tends to remove high-dimensional technical noise. The PCA is performed in a way that ensures equal contribution of each batch to the basis vectors of the PC space regardless of batch size. Namely, instead of centering the gene expression values around the overall expression mean, they are centered around an (unweighted) average of batch means [15].

Another difference with MNNCorrect is that fastMNN also performs *batch orthogonalization* after MNN identification but before applying the batch correction, which means that the cells in the target batch are projected onto the hyperplane perpendicular to the average batch effect vector over all cells. This removes any variation between the cells in the direction of the average batch effect. The idea behind this step is that it is assumed that the batch effect is orthogonal to the biological effects, so that any variation along the batch effect vector must be due to technical noise. The actual batch correction is applied after batch orthogonalization, and takes place in PC space as well.

Seurat Integration

The Seurat integration algorithm [3] is part of the Seurat single-cell analysis package in R. Its mode of operation is similar to fastMNN, but with some key differences.

First, instead of PCA, Seurat uses a variation on *canonical correlation analysis* (CCA) for dimension reduction. Given an $n \times p$ matrix $\mathbf{X}$ and an $n \times q$ matrix $\mathbf{Y}$, it looks for pairs of unit vectors $\mathbf{u} \in \mathbb{R}^p$ and $\mathbf{v} \in \mathbb{R}^q$ in the feature spaces of $\mathbf{X}$ and $\mathbf{Y}$ such that the covariance between the projections $\mathbf{Xu}$ and $\mathbf{Yv}$ is maximized. Mathematically, it aims to find

$$\underset{\mathbf{u}\in\mathbb{R}^p,\mathbf{v}\in\mathbb{R}^q}{\arg\max}\ \mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v} \quad \text{s.t. } \|u\|_2^2 = \|v\|_2^2 = 1,$$

the solutions to which are the left and right singular vectors of the cross-covariance matrix $\mathbf{X}^T\mathbf{Y}$ [26, 34]. When applied to two batches of scRNA-seq data, let $\mathbf{X}$ and $\mathbf{Y}$ be the standardized expression matrices of batch 1 and 2 such that the gene expression for each cell has mean 0 and variance 1. Let n be the number of genes, and let p and q the numbers of cells in batch 1 and batch 2, respectively. A dimension reduction can be obtained by embedding the cells in batch 1 onto the first k left singular vectors of $\mathbf{X}^T\mathbf{Y}$ as cell embeddings of batch 1, and embedding the cells in batch 2 on the first k right singular vectors of $\mathbf{X}^T\mathbf{Y}$. To make the cell embeddings comparable between the two batches, they are L2-normalized so that the Euclidean norm of each cell expression vector is equal to 1.

Next, Seurat computes MNN pairs between the cell embeddings of the two batches, but calls them *anchor pairs* which consist of a *query cell* (in the batch that is to be corrected) and an *anchor cell* (in the reference batch). Unlike in fastMNN, where all MNN pairs are used for correction, Seurat applies a filtering step and a scoring step to mitigate the effects of incorrect anchor pairs representing cells of different biological states. First, anchor pairs of which the anchor and the query cell are far away from each other in the original high-dimensional space (i.e., if the anchor cell is not among the k nearest neighbors of the query cell) are filtered out. Then, every anchor pair gets a score based on the shared neighbor overlap between the query cell and the anchor cell in both the reference and the query batch. Subsequently, the anchors get weighted by a function of the anchor score and the distance between anchor and query cell.

Correction vectors for the query cells are calculated using a combination of Gaussian kernel smoothing and the anchor scores and weights. In contrast to fastMNN, Seurat calculates the correction vectors in the original data space instead of in a dimensionally reduced space.

Harmony

Harmony is an iterative algorithm developed by Korsunsky et al. (2019) [10] that involves a series of clustering and correction steps. The algorithm is motivated by the premise that, had we known all the cell types, batch effects could be regressed out within each cell type. The clusters computed in the clustering step serve as proxies for the cell types that are present in the data. Since the actual number of cell types is unknown, the number of clusters K is defined to be $\min(100, \frac{N}{30})$. Based on the cluster assignments, cell-specific corrections are applied by modeling the cell expression with a mixture model and removing the estimated batch-dependent variation from the model. Then, the corrected data are used for clustering in the next iteration. Like fastMNN, Harmony operates on a reduced-dimensional PCA embedding of the gene expression space.

Let N be the total number of cells in the data set, let B be the number of batches, and let d be the number of dimensions in the PC space. The clustering step consists of a soft k-means clustering. The word *soft* entails that a cell is not assigned to a single cluster, but is given a probability for belonging to each of the clusters. The clusters are estimated with the following objective:

$$\arg\min_{R,Y} \sum_{i,k} R_{ki}\|Z_i - Y_k\|^2 + \sigma R_{ki} \log R_{ki} + \sigma\theta R_{ki} \log\left(\frac{O_{ki}}{E_{ki}}\right)\phi_i$$

$$\text{s.t. } \forall i\, \forall k\, R_{ki} > 0,\ \forall i \sum_{k=1}^{K} R_{ki} = 1,$$

where

- R_{ki} is the probability that cell i belongs to cluster k;
- Z_i is the d-dimensional expression vector of cell i in PC space;
- Y_k is the current centroid of cluster k;
- O_{ki} is the observed number of cells in batch b that are 'soft-assigned' to cluster k;
- E_{ki} is the expected number of cells in batch b that are 'soft-assigned' to cluster k under the assumption that the batch assignment and the cluster assignment are independent;
- ϕ_i is a one-hot vector indicating to which batch cell i belongs.

The objective function consists of a sum over three terms. The first term is the soft k-means clustering objective, the second term provides entropy regularization, governed by the hyperparameter σ, and the third term is a diversity penalty term, governed by σ and θ. With high entropy regularization ($\sigma \gg 0$), the algorithm favors clusters of equal size. With a high diversity penalty ($\theta \gg 0$), it favors clusters with high batch diversity. Thus, the two latter terms ensure that the clusters are of similar size and contain similar numbers of cells from every batch. The optimal soft cluster assignments R and centroids Y are estimated by an expectation-maximization (EM) procedure.

After estimating R and Y, the cluster assignment matrix R is used to compute a unique correction vector for each cell. Each cell expression vector Z_i is modeled with a mixture of linear

models:

$$Z_i = \sum_k R_{ki} W_k^T \phi_i^* + \epsilon_i$$

Here, $\phi_i^* = \begin{pmatrix} 1 \\ \phi_i \end{pmatrix}$ is the one-hot batch vector augmented with an intercept term, such that the collection of the column vectors $\phi_1^*, \ldots, \phi_N^*$ forms the $(B+1) \times N$ design matrix ϕ^*. W_k^T are the cluster-specific parameters of the linear model, estimated by weighted least squares:

$$W_k = (\phi^* \text{diag}(R_k) \phi^{*T} + \lambda I)^{-1} \phi^* \text{diag}(R_k) Z^T,$$

where R_k are the probabilities of each cell to belong to cluster k. A ridge penalty λ is added to the non-intercept parameters to avoid singularity of $\phi^* \text{diag}(R_k) \phi^{*T}$. The goal of this model is to remove batch-dependent variation while retaining batch-independent variation. Therefore, the expression vector Z_i is corrected by retaining only the cluster-specific intercepts $W_{k[1,\cdot]}$ and the cell-specific residual ϵ_i from the model:

$$\hat{Z}_i = \sum_k R_{ki} W_{k[1,\cdot]} + \epsilon_i$$

$\hat{Z}$ is used for computing new clusters in the next iteration. To avoid over-correction, however, the estimated batch-dependent variation is always subtracted from the original embedding Z, not from the corrected embedding $\hat{Z}$ from the previous iteration.

Dealing with low-dimensional fastMNN and Harmony outputs

The SIGMA algorithm needs a full expression matrix as input to work properly. However, fastMNN and Harmony only return corrected principal component scores, typically of much smaller dimensions than the original gene space. This makes it impossible to check whether the bulk of the singular value distribution follows the theoretical Marchenko-Pastur distribution given the dimensions of the expression matrix, and in particular to decide whether a singular value significantly deviates from the bulk. Seurat on the other hand, while computing the anchor pairs in a reduced space, calculates its correction vectors in the original gene expression space, and hence does not have this problem. Here, a strategy is presented that attempts to overcome the low-dimensional output of fastMNN and Harmony.

Both fastMNN and Harmony initially perform dimension reduction by PCA and use the principal component scores for computing batch correction vectors. While both methods do this by default when provided with an expression matrix, it is also possible to compute principal component scores beforehand and to use them as input to fastMNN and Harmony directly, while skipping the internal dimension reduction step. The advantage of manually calculating the principal components is that we are able to obtain a rotation matrix in addition to the scores matrix, which is not the case if the PCA is performed using the internal dimension reduction step inside fastMNN or Harmony. In our approach for dealing with the low-dimensional fastMNN and Harmony outputs, we will use the rotations for reconstructing back a modified expression matrix.

Let $\mathbf{X}$ be an $N \times P$ expression matrix where N is the number of cells, P is the number of genes, and $P > N$ (w.l.o.g.). Assuming that $\mathbf{X}$ is of full rank, the PCA results in an $N \times N$ scores matrix $\mathbf{T}$ and a $P \times N$ rotation matrix $\mathbf{V}$, such that $\mathbf{X} = \mathbf{T}\mathbf{V}^T$. Now, let $\mathbf{T}_{1:d}$ be the submatrix containing the first d columns of the scores matrix $\mathbf{T}$, and let $\mathbf{T}_{(d+1):N}$ be the submatrix containing the rest of the columns, such that $\mathbf{T} = (\mathbf{T}_{1:d}, \mathbf{T}_{(d+1):N})$. The idea is to use $\mathbf{T}_{1:d}$ as input for fastMNN and Harmony, yielding corrected scores $\tilde{\mathbf{T}}_{1:d}$, and pasting the corrected scores back with the rest of the columns to create a *partially corrected scores matrix*

$\tilde{\mathbf{T}}_d = (\tilde{\mathbf{T}}_{1:d}, \mathbf{T}_{(d+1):N})$. Then, $\tilde{\mathbf{T}}_d$ is multiplied back with the original rotation matrix $\mathbf{V}$ to obtain a new $N \times P$ expression matrix $\tilde{\mathbf{X}}_d = \tilde{\mathbf{T}}_d \mathbf{V}^T$ representing the original gene space. In short, we have only corrected the first d principal components of $\mathbf{X}$, leaving the other principal components unchanged. By applying this strategy with fastMNN and Harmony, an expression matrix of the original dimensions can be returned even though batch correction takes place in a reduced-dimensional space. The number of principal components to be corrected, d, is chosen by examining the degree of batch mixing for different values of d. The different metrics for assessing how well two or more batches are mixed are described in the next section.

2.3 Evaluation of batch mixing

The goal of batch effect correction is to combine data from different batches in such a way that they become indistinguishable. Our specific goal is to determine whether current batch effect correction methods are able to remove batch effects from singular vectors and thereby remove their influence on SIGMA. Therefore, in order to compare the results of different batch effect correction methods, we need ways to evaluate the degree of mixing of the batches in the gene expression space. An intuitive way of evaluating this degree is to assess the distribution of batches on a local level. In this section, two currently existing evaluation metrics for batch mixing will be discussed, namely the k-nearest neighbors batch effect test (kBET) and the local inverse Simpson index (LISI). Subsequently, a novel metric will be introduced: the local Kullback-Leibler divergence.

Existing evaluation metrics

k-Nearest Neighbors Batch Effect Test (kBET): The k-nearest neighbors batch effect test (kBET) [4] determines local neighborhoods in the data space with k-nearest neighbor graphs and performs a χ^2-test by comparing local proportions of cells originating from each batch to the global batch proportions. The test is performed a number of times on different random samples from the data while counting the number of times the null hypothesis is rejected, resulting in a rejection rate. The smaller the rejection rate, the better the batches are mixed.

Local Inverse Simpson Index (LISI): The local inverse Simpson index (LISI) [10] was developed by the creators of Harmony. The Simpson index $S = \sum_i p_i^2$, where p_i is the proportion of cells originating from batch i, is a measure for diversity that is often applied in ecology to quantify species diversity [24]. Its inverse, $\frac{1}{S} = \frac{1}{\sum_i p_i^2}$ represents the 'effective' number of batches present in a data set. If the data consists of two equally sized batches, then $\frac{1}{S} = 2$. In contrast, if the batch sizes are very skewed, $\frac{1}{S}$ will be just above 1. As with kBET, the inverse Simpson index is computed in local neighborhoods with the use of KNNs.

Local Kullback-Leibler divergence

A drawback of LISI is that it achieves its maximum if the local batch sizes are equal, regardless of the global batch distribution. For this reason, another metric was devised based on the *Kullback-Leibler (KL) divergence* between the local and the global batch proportions. Let P and Q be two discrete probability distributions on a sample space $\mathcal{X}$. In this case, $\mathcal{X}$ is the set of batches in the data. The KL divergence from Q to P is defined as

$$D_{KL}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}.$$

Note that in general, $D_{KL}(P\|Q) \neq D_{KL}(Q\|P)$. Particularly, if there exists an $x \in \mathcal{X}$ such that $P(x) > 0$ and $Q(x) = 0$, but no x exists such that $Q(x) > 0$ and $P(x) = 0$, then $D_{KL}(P\|Q)$ becomes infinite, whereas $D_{KL}(Q\|P)$ does not. For this metric to be useful, we want to avoid infinite values. Therefore, it is best to choose P to represent local batch proportions and Q to represent global batch proportions, since global batch proportions are necessarily greater than 0 (otherwise, a batch would not exist), which is not the case for local batch proportions.

Another property is that $D_{KL}(P\|Q) \geq 0$, and if $P(x) = Q(x)$ for all $x \in \mathcal{X}$, then $D_{KL}(P\|Q) = 0$, so the KL divergence is minimized if the local and global batch proportions are equal.

Similarly to in kBET and LISI, the local neighborhoods are determined by finding the KNNs of every cell or of a random subset of cells. Then, the average local KL divergence can be computed over these neighborhoods.

Under the null hypothesis that batches are well mixed, the local batch proportions will fluctuate randomly around the fixed global batch proportions, and thus the average KL divergence will behave as a random variable. Consequently, a null distribution of the average KL divergence for well-mixed batches can be constructed by permuting the batch labels of the cells and calculating the average KL divergence for many such permutations. The actual average local KL divergence can be tested against the null distribution to test for significance. Note that this is an option for using this method as a hypothesis test; in this thesis, only the local KL divergences were computed.

Identifying batch effects along singular vectors

For SIGMA, we are interested in singular values that deviate from the bulk distribution of singular values of the expression matrix: significantly deviating singular values indicate the presence of heterogeneity within clusters. The question is whether the heterogeneity has a biological cause (e.g., if the cluster contains cellular subtypes) or a technical one (e.g., a batch effect). In order to determine whether a deviating singular value may (in part) be caused by a batch effect, we can evaluate the degree of batch mixing along its associated right singular vector. If the batches are badly mixed along the singular vector, it signifies that a batch effect contributed to, if not caused the presence of the high singular value. Therefore, assessing batch mixing for all significantly deviating singular value will help decide whether the calculation of SIGMA was distorted by a batch effect.

In order to assess batch mixing along a single singular vector we used Welch's t-test to test for the difference between the means of the batches in cases where only one cell type and two batches were present. Since scRNA-seq samples are usually relatively large in terms of numbers of cells, it is not very important that the data be normally distributed along the singular vector to achieve high statistical power [14]. Alternatively, one of the aforementioned batch mixing metrics, kBET, can be used to identify batch effects in singular vectors (since it provides a hypothesis test out of the box), but it is considerably slower than the t-test, especially if one wants to examine every singular vector. In fact, even performing the t-test already took a substantial amount of time, prompting the creation of an alternative t-test function that computes Welch's t-statistics in a vectorized manner in R. After computing the p-values from the t-test, they were corrected for false discovery rate (FDR) by a Benjamini-Hochberg procedure. Singular vectors with corrected p-values below 0.05 were regarded as *batch effect-associated singular values* (BSVs).

For batch effect correction of data sets consisting of more than one cell type, we used a two-sample Anderson-Darling (AD) test [21] instead of a t-test for finding batch effects in singular vectors. This test was chosen in order to deal with the multimodal distributions that form along singular vectors as a result of the presence of multiple cell types in the data. Because the two-sample AD test can be used for finding significant differences between empirical distributions that

deviate from normality, it was deemed more suitable than the t-test, which assumes normally distributed data. Akin to the more well-known Kolmogorov-Smirnov test, the AD test computes a test statistic based on the difference between the empirical cumulative distribution functions (ECDFs) of the two samples. The AD statistic is defined as

$$A^2_{nm} = \sum_{i=1}^{N-1} \frac{(M_i N - ni)^2}{i(N-i)},$$

where m and n are the number of observations in batch 1 and batch 2, respectively, M_i is the number of observations in batch 1 that are less than or equal to the ith smallest observation in the two batches combined, and $N = m + n$ [21]. The AD statistics were calculated according to this equation and standardized in the same way as in the function ad.test() from the kSamples package [23]. Subsequently, p-values were computed based on the standardized statistics using the function ad.pval() from the same package. As with the aforementioned t-test, the p-values were FDR-corrected with Benjamini-Hochberg and all singular vectors with p-values below 0.05 were considered BSVs.

2.4 Data simulation

The simulation of scRNA-seq data was performed with the splatSimulate() function from the Splatter package [33]. Splat consists of a hierarchical model for gene expression that includes parameters for controlling the numbers of cells and genes, the presence of outlier genes, the library size of the cells, the number of dropouts, and the strength of the mean-variance relationship of gene expression for each cell. The core of Splat is a gamma-Poisson model, a Poisson distribution for which the mean parameter is itself a random variable following a gamma distribution. The resulting distribution is equivalent to a negative binomial distribution.

Simulation with Splat proceeds in several steps. First, given a pre-specified number of genes N and cells M, the mean expression of each gene is generated from a gamma distribution. Next, to allow for a few highly expressed genes, some of the generated gene means are randomly selected and inflated by a certain factor. Both the frequency and the inflation factors of these outlier genes are controlled by hyperparameters. At this point, the gene means λ'_j, $j = 1, \ldots, N$, are the same for each cell. Next, the expected library sizes of the cells are simulated from a log-normal distribution and are used to proportionally adjust the gene means for each specific cell, so that each combination of gene i and cell j has a specific mean expression parameter $\lambda'_{i,j}$. Subsequently, to enforce an inverse relationship between the expression mean and variance of each gene, a factor called the *biological coefficient of variation (BCV)* is generated from a scaled inverse chi-squared distribution with parameters partly based on the current gene mean. The BCVs and the current cell-gene means $\lambda'_{i,j}$ are used as parameters for generating a new set of parameters $\lambda_{i,j}$ from a gamma distribution. Then, the gene count $Y'_{i,j}$ for gene i and cell j is simulated by a Poisson distribution with mean $\lambda_{i,j}$. Finally, technical dropouts are modeled with a logistic function which dictates the probability that a certain gene expression should be set to 0, thereby leading to the final gene expression values $Y_{i,j}$.

Splatter gives the option to create differential gene expression between groups of cells by multiplying some gene means in each group with random factors generated from a log-normal distribution with mean μ_g and standard deviation β_g, thus creating cell types with distinct gene expression. The larger μ_g and β_g, the larger the differences in gene expression between two groups. Similarly, there is an option for creating batch effects. In this case, within each batch, *every* gene mean is multiplied by a random factor drawn from a log-normal distribution with

mean μ_b and standard deviation β_b. Here as well, larger values of μ_b and β_b generate more extreme differential expression between batches (i.e., the batch effect becomes stronger).

2.5 Data preprocessing

All data sets, simulated or otherwise, were analysed with the scater package [19], which provides the SingleCellExperiment class in which expression matrices can be stored together with metadata of the cells and genes, and with low-dimensional representations of the data. In addition, it comes with some functions for quality control, dimension reduction, and visualization.

A few preliminary steps were necessary before batch correction, namely quality control (QC) and normalization. QC was needed to remove outlier cells and genes, and then normalization was applied to correct for the differences in library size between cells. Both steps will now be explained in more detail.

Quality control

Quality control (QC) serves to remove cells and genes from the data that show unusual or uninformative expression [13]. For cell QC, histograms of the library sizes (total number of reads) and total number of genes detected are inspected in order to manually determine thresholds below which cells should be removed. A small library size or number of detected genes may indicate cells of bad quality, meaning that their gene expression pattern is unreliable. Cells for which that library size or number of detected genes appeared distinctively below the bulk of the library size distribution were removed. Additionally, cells with a very high relative expression of mitochondrial genes were removed, because this could indicate that cytoplasmic RNA has leaked out from the cell prior to cell capture, whereas the mitochondrial RNA has been retained inside the mitochondria. For gene QC, which was applied after cell QC, all genes that did not have more than one count in at least two cells were treated as undetected and removed from the data. The function 'nexprs' from the scater package was used to identify such genes.

Log-normalization

After QC, the gene expression needs to be scaled to account for cell-to-cell differences arising from the sequencing procedure. This scaling step, called *normalization*, is necessary to make valid comparisons between cells further on in the analysis. A simple normalization procedure is counts-per-million (CPM) normalization [13], a form of library size normalization where the gene counts of each cell are divided by its library size and multiplied by 1 million. However, if differentially expressed (DE) genes are present, which is most likely the case if the data contains multiple cell types, using just the library size is too simple because it can obscure meaningful biological effects. For example, a cell type might show much upregulation for certain genes compared to other cell types, whereas their housekeeping genes show normal expression levels. In this case, applying library size normalization will scale down the expression of housekeeping genes a lot, giving the appearance of downregulation compared to other cell types, even though that was not the case. Also, the presence of technical dropouts (zero counts) may cause a bias in the estimation of the library size factors. However, while the presence of DE genes and technical dropouts is most problematic for differential expression analysis, we will not perform such analyses and will therefore stick with library size normalization.

Finally, the counts are divided by the size factors and $\log_2$-transformed. Since scRNA-seq data usually contains many zero counts, a value of 1 is added to all normalized counts prior

to the transformation to avoid computing logarithms of 0. The resulting values are denoted as *logcounts*.

Feature selection

Feature selection refers to selecting a subset of the detected genes in order to remove unimportant noise and keep those genes that contribute most to the heterogeneity in the data [13]. Specifically, we are looking for *highly variable genes* (HVGs). The scran package [17] was used for identifying HVGs. The function modelGeneVar() first computes the per-gene expression mean and variance. Subsequently, the gene variance is fitted against the gene means by locally weighted scatterplot smoothing (LOWESS) in an attempt to decompose the variance into a technical and a biological component: the fitted values from the estimated mean-variance trend are regarded as the technical components, and the residuals are considered to represent the biological components. Evidently, this method rests on the assumption that a large majority of genes exhibit mostly technical variance, therefore contributing most to the estimated mean-variance trend. By assuming that the per-gene variance estimates are normally distributed around the mean-variance trend, modelGeneVar() then computes p-values based on the ratio between the biological and technical variance components, i.e., the ratio between the residuals and the mean-variance trend. A predefined number or proportion of genes (typically 1000-2000 genes) with the lowest p-values and therefore with the highest biological-to-technical variance ratios are selected as HVGs.

Because batch effects can significantly contribute to the heterogeneity of gene expression, scran resolves to model gene-variance trends and to compute technical and biological variance components and p-values for each batch separately. The p-values per gene are then combined using Fisher's method.

2.6 Description and analysis of simulated data sets

The simulations were divided into four different scenarios: (1) two batches of one cell type, (2) multiple batches of one cell type, (3) two batches of two cell types, and (4) an unbalanced data set with four cell types and two batches, where one of the cell types is absent from the second batch.

For the first scenario, two batches of 300 cells, each with 15 000 expressed genes, were simulated. Three different batch effect mean (μ_b) and standard deviation (β_b) parameters were chosen to compare different batch effect strengths: high ($\mu_b = \beta_b = 0.1$), medium ($\mu_b = \beta_b = 0.07$), and low ($\mu_b = \beta_b = 0.04$). All other parameters had their default values.

In the second scenario, instead of varying the batch effect strength, the default $\mu_b = \beta_b = 0.1$ were used, and the number of batches was varied, simulating three data sets containing $B = 3, 4,$ and 5 batches, respectively. The number of cells per batch was $\frac{600}{B}$. The number of simulated genes was 15 000.

The third scenario tested different batch effect strengths, akin to Scenario 1, but now with two cell types instead of one. We simulated 300 cells per batch, 15 000 genes, and the cells were given equal probability of belonging to one of two cell types. The three different batch effect strengths were $\mu_b = \beta_b = 0.12, 0.15, 0.18$.

For the fourth and final scenario, two batches of 400 cells and 15000 genes were simulated, with four different cell types. The cell type probabilities were set to 0.4, 0.2, 0.2, and 0.2. The default batch effect mean and standard deviation parameters $\mu_b = \beta_b = 0.1$ were used. To create imbalance between the batches, all cells of the fourth cell type (76 cells) were removed from Batch 2.

In all scenarios, after removing undetected genes and log-normalizing, feature selection was applied by filtering for the 10% most highly variable genes (HVGs) with the modelGeneVar() function from scran, resulting in a bit less than 1500 genes for each simulation (not exactly 1500, because some undetected genes were removed from the 15000 original genes before selecting the top 10%). Next, the logcounts were scaled row-wise such that each row had a mean expression 0 and a variance of 1, and then scaled column-wise in the same manner. Scaling both row- and column-wise is necessary for the bulk distribution of singular values to follow the appropriate MP distribution. Then, the singular values and vectors of the scaled expression matrices were computed and examined.

Batch effect correction with fastMNN, Harmony, and Seurat was applied to Scenarios 1 and 4 using different values for the dimensionality parameter d. For Scenario 4, the batch mixing metrics were calculated based on $k = 75$ nearest neighbors. Before computing the SVDs, the matrices of logcounts were scaled row- and column-wise such that they had a mean of 0 and a variance of 1.

2.7 Description and analysis of empirical data sets

Dataset 1: Dendritic cell data

The first data set that was analysed consists of human blood dendritic cells (DCs) from a study by Villani et al. in 2017 [31]. Four DC types were isolated from blood by fluorescence-activated cell sorting (FACS), providing a ground truth of the cell identities. The data were prepared exactly as described in Tran et al. (2020) [29], resulting in two batches of 384 cells each. Both batches contained 96 plasmacytoid DCs (pDCs), 96 *double negative* cells (i.e., negative for the CD1C and CD141 cell type markers), 96 CD1C cells, and 96 CD141 cells. Note that the batch sizes and cell type compositions are perfectly balanced (before quality control).

For cell quality control, all cells with $\log(D+1) < 8$ were removed, where D is the number of detected genes per cell. No percentage of relative mitochondrial expression could be calculated because the data set did not contain any mitochondrial genes. After cell and gene QC, the top 10% highly variable genes were selected, resulting in a a data set with 749 cells (371 in batch 1 and 378 in batch 2) and 1906 genes. Note that library size normalization was not necessary because the data had already been CPM-normalized, so therefore the counts could directly be $\log_2$-transformed before feature selection. Batch correction with fastMNN, Harmony, and Seurat was applied to the logcounts using the strategy described in Section 2.2, and with several different dimensionality parameters $d = 10, 20, \ldots, 100$, and the degree of batch mixing for each d was examined using kBET, LISI, and the local KL divergence as mixing metrics. The number of nearest neighbors for computing the mixing metrics was set to $k = 90$. Before computing the SVDs, the uncorrected and corrected matrices were scaled row- and column-wise such that they had mean 0 and variance 1.

Dataset 2: Lymphocyte data

The lymphocyte data set consisted of cells from lymph nodes of two patients with a form of B-cell lymphoma. The data were kindly provided by Dr. Kees van Bergen from the Leiden University Medical Center. The cells had been sorted by FACS to separate the (diseased) B-cells from the rest. This data set contained the remaining cells, which were supposedly healthy. From each patient, a sample of the lymph nodes was taken twice: the second sample ("K3nB") was taken a year after the first sample ("K2nB"). The two samples K2nB and K3nB per patient were regarded as two batches. This means that the technical batch effect is confounded with

a biological effect, namely due to biological variability in time. The two batches of patient 1 consisted of 1128 and 1784 cells, and the two batches of patient 2 consisted of 1596 and 776 cells. The batch sizes were therefore considerably less balanced than in data set 1. Other than with the dendritic cell data, no ground truth of cell types was available for the lymphocyte data, but the data were likely to contain at least CD4+ and CD8+ T-cells, B-cells, and natural killer (NK) cells, according to the information provided by Dr. Van Bergen.

For cell quality control, the two patients were processed separately, and since the batches were very unbalanced, they were also processed separately within each patient. First, 13 mitochondrial genes were identified and separated from the endogenous (nuclear) genes. The mitochondrial genes were only used to compute relative mitochondrial expression, and were disregarded in the subsequent analyses. Thresholds for library size and number of detected genes for each patient and batch are shown in Table 2.1. No thresholds were used for mitochondrial expression, since the other thresholds already seemed sufficient.

	Batch	Library size	Number of detected genes
Patient 1	K2nB	1000	500
	K3nB	1250	600
Patient 2	K2nB	1000	650
	K3nB	-	650

Table 2.1: Thresholds for cell quality control of the lymphocyte data per patient and batch. All cells with library size or number of detected genes below the indicated values were removed. For batch K3nB of Patient 2, no library size threshold was chosen.

After cell and gene quality control, scale factors for normalization were computed with the logNormalizeCounts() function from scater, which uses library size normalization. Feature selection was performed by selecting the top 20% HVGs using the modeGeneVar() function. The final data sets contained 1711 genes and 2866 cells (1090 cells in K2nB, 1776 cells in K3nB) for Patient 1, and 1712 genes and 2263 cells (1489 in K2nB, 774 in K3nB) for Patient 2. Again, batch correction with fastMNN, Harmony, and Seurat was performed using dimensionality parameters $d = 10, 20, \ldots, 100$ and the degree of batch mixing was evaluated with kBET, LISI, and with the local KL divergence, again all with $k = 90$ nearest neighbors. As with the simulated scenarios and the dendritic cell data, the uncorrected and corrected matrices were scaled row- and column-wise to have mean 0 and variance 1 before computing the SVD.

2.8 Software

Throughout this section, a number of R packages have been mentioned that were used for the simulations and for data analysis. An overview of the used packages and their versions is listed in Table 2.2. All analyses were conducted in R version 4.1.0.

Package	Version	Used for	Reference
scater	1.20.0	SingleCellExperiment class, Quality control, PCA, UMAP	[19]
splatter	1.16.1	scRNA-seq data simulation	[33]
scran	1.20.1	Feature selection	[16]
batchelor	1.8.0	fastMNN batch correction	[5]
Harmony	1.0	Harmony batch correction	[10]
Seurat	4.0.3	Seurat batch correction	[3]
kBET	0.99.6	kBET batch mixing metric	[4]
lisi	1.0	LISI batch mixing metric	[10]
FNN	1.1.3	Fast KNN search for kBET and local KL	[2]
kSamples	1.2-9	p-value computations for the Anderson-Darling test	[23]
SIGMA	0.0.0.1	Signal measurement angle	[20]
tidyverse	1.3.2	Data manipulation and plotting (mainly ggplot2)	[32]

Table 2.2: Overview of R packages used in the analysis.

Chapter 3

Results

First, we will examine how batch effects are manifested in simulated data. Using the Splatter package, several different scenarios were simulated. Three different batch effect correction methods (fastMNN, Harmony, and Seurat) were deployed for correcting the simulated batch effects. The resulting observations and conclusions were used for analyzing the two real-world data sets: the dendritic cell data and the lymphocyte data.

3.1 Batch effects in simulated data

We start by investigating the behavior of the singular values and vectors of simulated expression matrices containing batch effects. Four different scenarios were examined: (1) two batches of one cell type, (2) multiple batches of one cell type, (3) two batches of two cell types, and (4) two batches of four cell types with one cell type missing from the second batch.

Scenario 1: Two batches of one cell type

A simple scenario to start with is one where all the cells are of the same biological type and there are two batches. In that case, the only biological variation is cell-specific noise, and the only other variation is due to the batch effect. To explore how a batch effect influences the singular value distribution of the (scaled) expression matrix, three data sets of 600 cells were simulated with Splatter, each consisting of two batches of equal size, and each with a different set of location (μ_b) and scale (β_b) parameters determining the strength of the batch effect.

The singular value histograms and the first two right singular vectors of the three gene expression matrices are shown in Figure 3.1. In each of the three simulations, most of the singular values followed the expected Marchenko-Pastur (MP) distribution. Two singular values could be observed falling outside the bulk distribution: one singular value was located above the bulk distribution and another was equal to 0. The former arose from the batch effect, which introduced a nonrandom signal in the expression matrix. Accordingly, let us denote such large singular values as *signal singular values* (SSVs). The three histograms demonstrate that the stronger the batch effect, the greater the SSV becomes.

The other deviating singular value, the one equal to 0, was a consequence of scaling the matrix row-wise and column-wise prior to computing the SVD. In particular, for expression matrices with fewer cells than genes ($N < P$), centering the rows of the matrix introduces some linear dependence among its columns by removing a degree of freedom and thus a rank. Since it is known that the rank of a matrix is equal to its number of non-zero singular values, one singular value must therefore turn zero. In fact, if the scaling step is omitted, the expression matrix will

contain an extremely high singular value reflecting the general mean expression. Scaling moves this singular value to zero. In expression matrices with more cells than genes ($N > P$), it is the centering of columns rather than rows that leads to a rank reduction. Because the scaling step involves both row- and column-wise scaling, all expression matrices will lose a rank from scaling regardless of whether the number of cells is larger or smaller than the number of genes.

Now, let us take a look at the right singular vectors corresponding to the deviating large singular values. Since the cells (observations) are represented as columns in a single-cell expression matrix, each right singular vector has the same length as the number of cells, and multiplying the matrix of right singular vectors $\mathbf{V}$ with the diagonal matrix of singular values $\mathbf{\Sigma}$ will give the principal component scores $\mathbf{V}^T\mathbf{\Sigma}$. The singular values determine the importance of each principal component. Therefore, examining a right singular vector is equivalent to examining the "unweighted" scores of a single principal component. The scatter plots in the lower row of Figure 3.1 display the elements of the first two right singular vectors. In completely random matrix, the elements of the singular vectors are expected to be normally distributed with mean 0, which was clearly not the case for the first singular vector in all three scenarios (Figure 3.1). Instead, the vector elements (cells) were divided over two distributions dictated by the batch effect. For high batch effect strengths (Fig. 3.1A), the two distributions were very distinct. The weaker the batch effect (Fig. 3.1B&C), the less distinct the two distributions became.

Scenario 2: Multiple batches of one cell type

What happens to the singular value histogram if the data consists of more than two batches? To investigate this, expression matrices containing three batches of 200 cells each, four batches of 150 cells, and five batches of 120 cells were simulated, so that the total number of cells was exactly 600 for all three expression matrices. The results are shown in Figure 3.2.

The upper row of Figure 3.2 shows the singular value histograms for the three different experiments. In the scenario with three batches (Fig. 3.2A), two SSVs were present, in the scenario with four batches (Fig. 3.2B) there were three SSVs, and the five-batch scenario (Fig. 3.2C) had four singular values above the bulk distribution. From simulated experiments with up to 20 batches, it was observed that the number of singular values above the bulk is equal to the number of batches minus 1. This seems to be related to the number of hyperplanes needed to (optimally) separate the batches.

Scenario 3: Two batches of two cell types

In this section and in the next, we will combine technical variability and biological variability by simulating data sets consisting of two or more cell types in two batches, to inspect the balance between technical (batch) and biological (cell type) effects. The presence of multiple cell types in single-cell data is ubiquitous, and a simple and often encountered data integration problem is one consisting of two batches, so these two scenarios reflect simple but typical batch effect correction problems. First, two batches of 300 cells were simulated. Approximately 150 cells of each cell type were present in each batch. This scenario was repeated three times with three different batch effect strengths ($\mu_b = \beta_b = 0.12, 0.15, 0.18$) to alter the magnitude of the batch effect with respect to the magnitude of the biological effect.

The singular value histograms and scatter plots of the first two right singular vectors of the three scenarios are displayed in Figure 3.3. Two SSVs could be observed, one of which was due to the batch effect, and the other was due to the two cell types. In the first scenario (Fig. 3.3A), the cell type effect was the strongest, as the effect of cell type was almost completely captured by the first singular vector, and the batch effect was mostly manifested in the direction

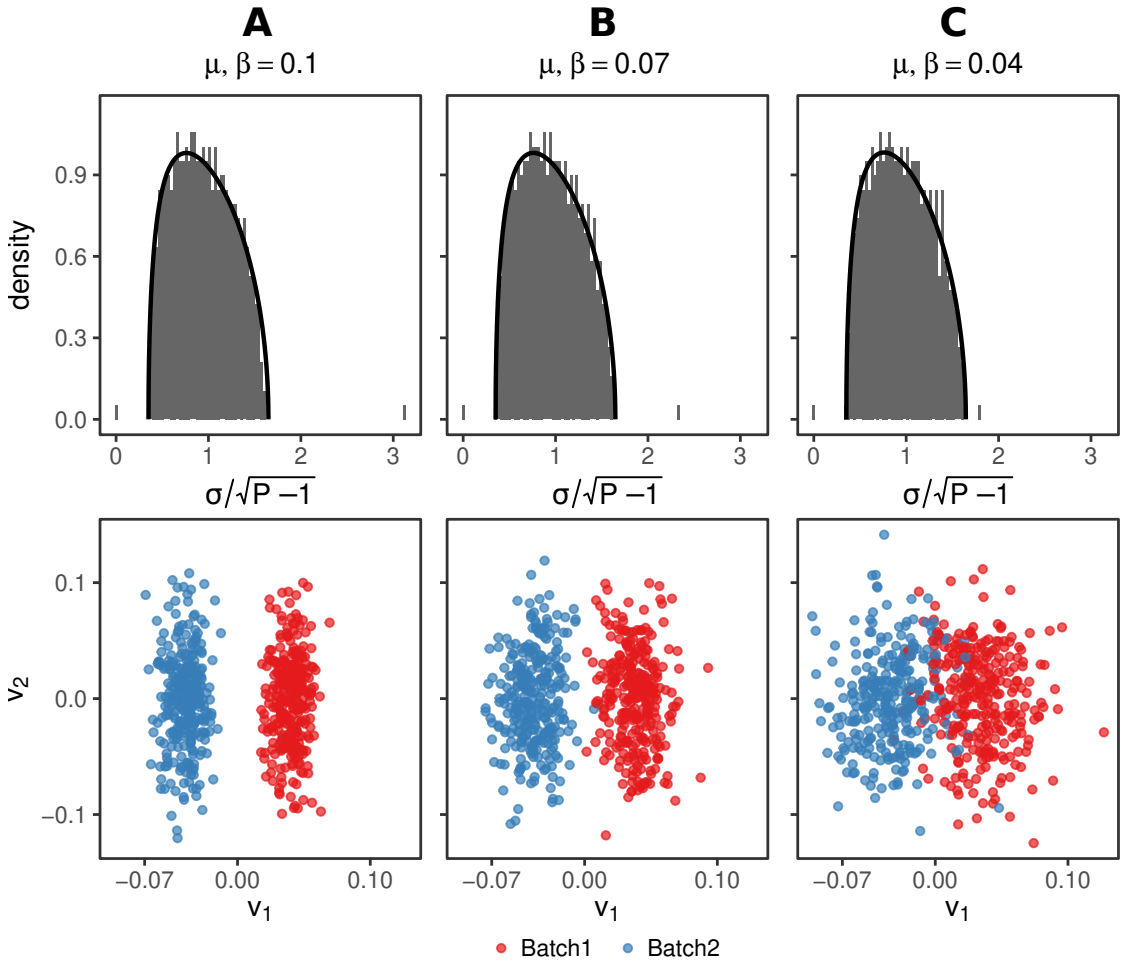


Figure 3.1: The effect of batch effect strength on the singular values and vectors of the scaled expression matrix. Three different batch effect strengths were simulated with Splatter, as indicated by the parameters μ and β above the three columns. The upper row contains singular value histograms of the expression matrices. The solid black lines superimposed on the histograms describe the theoretical MP densities given the dimensions of the expression matrices. The lower row depicts the elements of the first two right singular vectors of the expression matrices.

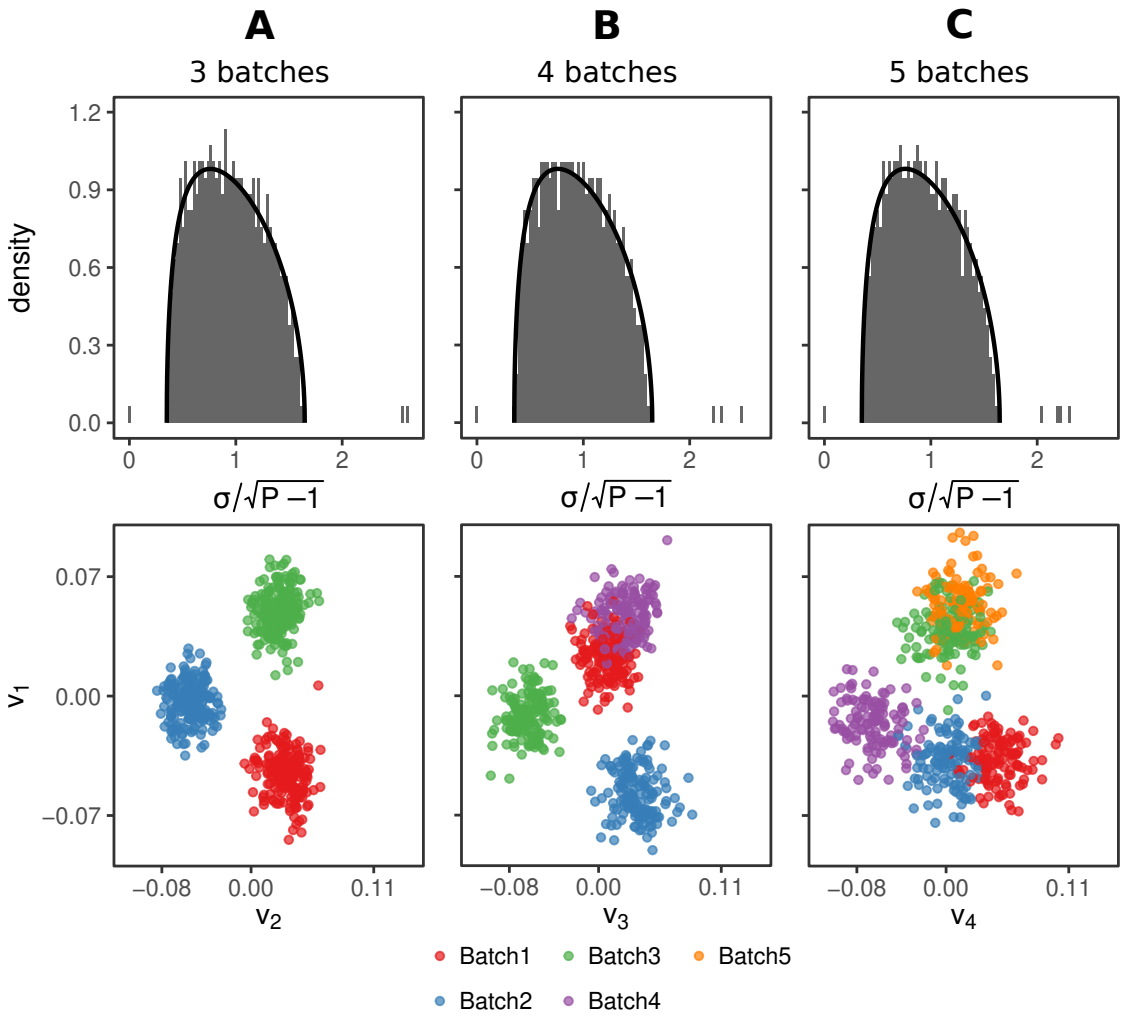


Figure 3.2: Singular value histograms (upper row) and singular vector scatter plots (lower row) of simulated data sets consisting of multiple batches. For the three-batch data set (A), the first two singular vectors were plotted. For the four-batch data set (B), the first and third singular vectors were plotted. For the five-batch data set, the first and fourth singular vectors were plotted.

of the second singular vector. In this case, it could be said that the highest singular value was associated with the cell type effect, and the second highest singular value was associated with the batch effect. However, when the magnitudes of the batch effect and the cell type effect became more similar, this did not hold anymore. As the SSVs got closer to each other, both effects seemed to spread over the first two singular vectors (Fig. 3.3B&C). This makes sense, because singular vectors point in the directions of the most variance, and if the cell types and and the batch effect contribute equally to the total variance in the data, both effects will provide equal singular values and will also be manifested in equal proportions on both of the first two singular vectors. In conclusion, batch effects and cell type effects are not necessarily captured by separate singular vectors, but can be divided over multiple vectors depending on the relative magnitudes of cell type and batch effects.

Scenario 4: Unbalanced cell type compositions

In the fourth scenario, four cell types were simulated in such a way that Batch 1 contained all four cell types, and Batch 2 contained only three of the four cell types, thereby creating an imbalance in the cell type compositions of the two batches. In Figure 3.4A, the singular value histogram shows four SSVs, which is as expected given that the (technical) batch effect contributed one SSV, and the differences between the four cell types contributed three SSVs. From the scatter plots (Fig. 3.4B), we can see that the batch effect was mostly captured by the second, third, and fourth singular vectors, whereas the cell type effects could be found along all four singular vectors. Furthermore, the distinction between the fourth cell type and the other cell types was mostly captured by the fourth singular vector. Since all other singular values were positioned inside the bulk distribution, it was assumed that no other singular vectors carried batch or cell type effects. This assumption can be tested by performing t-tests on the right singular vectors, although no t-tests were performed here (in the subsequent sections about batch correction, we did perform hypothesis tests to detect batch effect-associated singular values).

Scenario 4 shows us that when the batch effect consists not only of technical effects, but also of biological effects (i.e., the presence of a cell type that is not found in the other batch), it becomes harder to attribute a SSV solely to technical differences between batches because the difference in cell type composition is also involved.

3.2 Batch effect correction of simulated data

Now, we will use several of the previously simulated scenarios to test out the three different batch correction methods. One problem to overcome with respect to batch correction was the fact that fastMNN and Harmony work in a reduced-dimensional space and return low-dimensional corrected matrices, such that the SVD would only yield a low number of singular values of which the distribution is unknown. In section 2.2, a strategy was devised to overcome the low dimensionality, which boiled down to computing the maximum number of principal components from the expression matrix, but only correcting the scores of the first d principal components with fastMNN or Harmony. To test whether this strategy indeed returned adequate singular value histograms, and to find out for which values of d it works best, it was implemented on the expression matrix of Scenario 1 (one cell type, two batches). Although Seurat does not return low-dimensional results, it does have a dimensionality parameter defining the number of canonical correlation vectors that are being used for finding the anchor pairs, and so this parameter was also called d.

It can be observed in Figure 3.5B that for both fastMNN and Harmony, the bulk of singular values still fitted the MP distribution very well after batch correction of the first d components

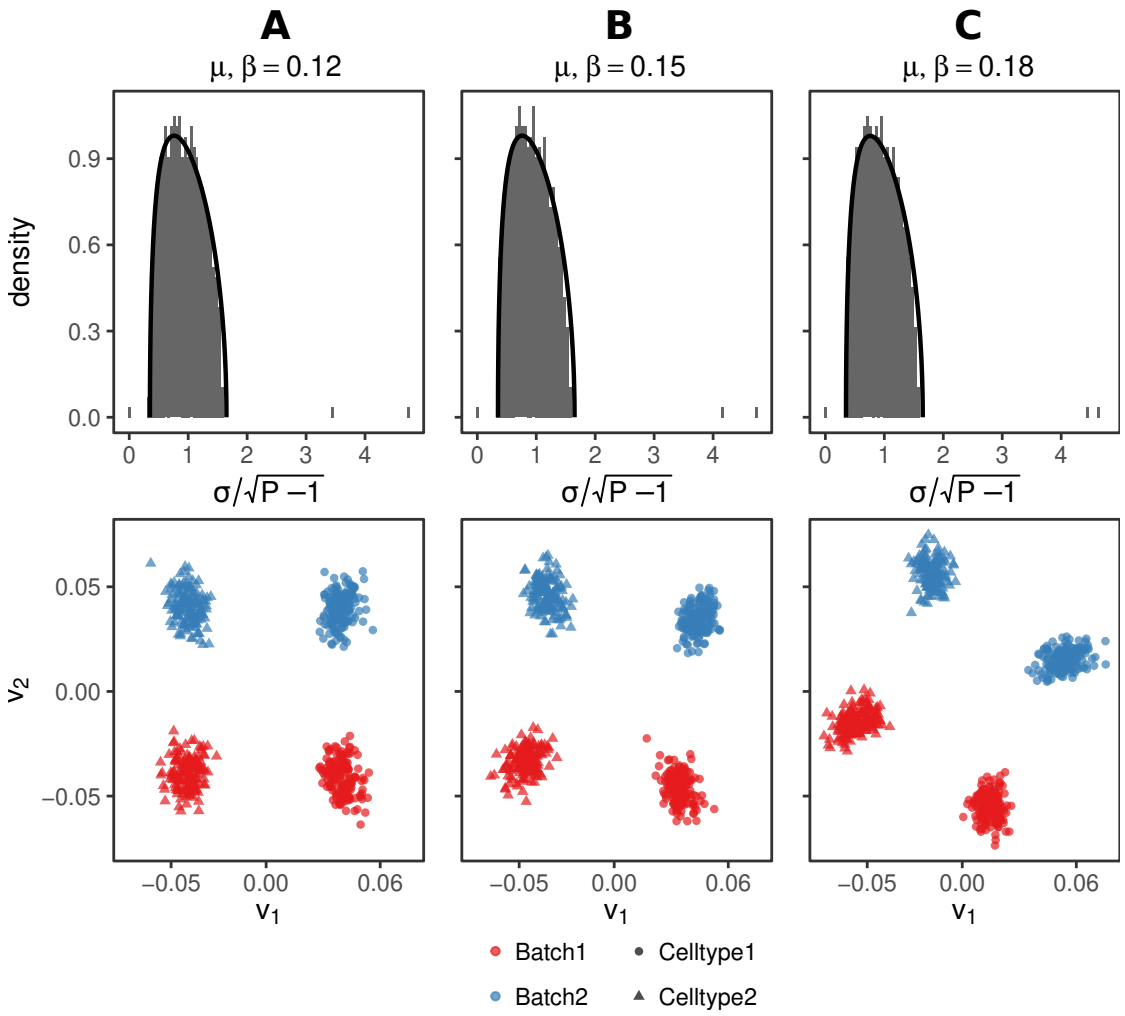


Figure 3.3: Simulated scRNA-seq data with two cell types and different batch effect strengths. The upper row contains singular value histograms of the expression matrices, with superimposed theoretical MP densities. The lower row shows the elements of the first two right singular vectors of each expression matrix, with colors representing batches and shapes representing cell types.

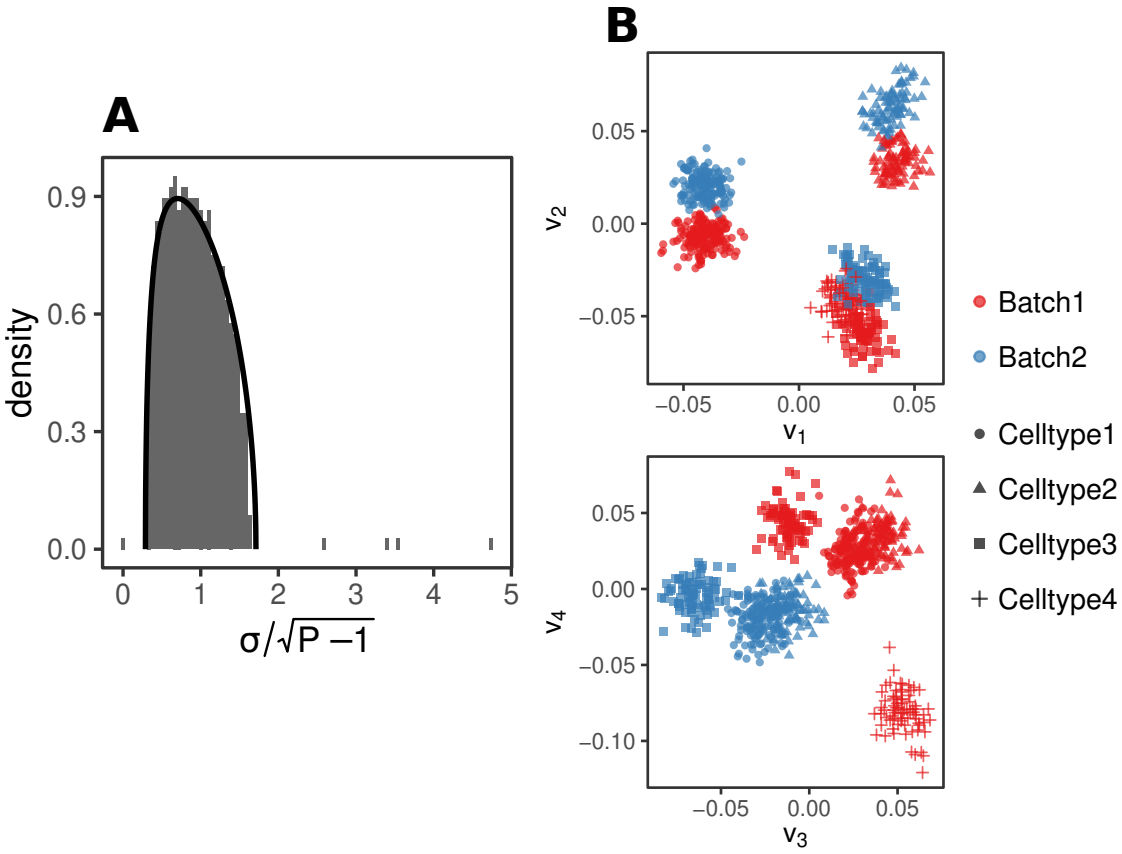


Figure 3.4: Singular values and vectors of a simulated data set containing two batches with unbalanced cell type compositions. Batch 1 contained four cell types, while Batch 2 contained only three of the four cell types. The histogram of singular values is shown in (A). In (B), the elements of the first four right singular vectors have been plotted, with colours representing batches and shapes representing cell types.

and reconstruction of the expression matrix. This was true for low, moderate, and high values of d. Also, it can be noticed that the high SSV associated with the batch effect (as indicated by the asterisk under the histogram in 3.5A) had been removed (but not at $d = 600$ for fastMNN and Harmony), but in all cases, there was still at least one singular vector present along which a significant batch effect could be detected (Welch's t-test, $p_{BH} < 0.05$; raw p-values were corrected with a Benjamini-Hochberg procedure), which is again indicated by the asterisks under the histograms. In fact, in the case of fastMNN, for low to moderate values of d, a singular vector containing a batch effect could be found at one of the two singular values equal to 0, meaning that it was effectively removed from the data at the cost of one matrix rank, since the rank of a matrix is equal to its number of nonzero singular values. The loss of one rank can be attributed to the PCA that is carried out before batch correction, since it involves centering of the gene means, while another rank is lost because the rows and columns are scaled before computing the SVD. Together, this explains the fact that the fastMNN histogram had two singular values equal to 0. The SSV was not removed by fastMNN at $d = 600$, likely because correcting too many of the principal components is less effective due to the extra noise.

In the case of Harmony (Fig. 3.5B, second row), the SSV containing the batch effect was also removed, but instead of the batch effect appearing at a singular value of 0, it appeared on the edges of the bulk distribution. With $d = 2$, the batch effect seemed to have spread over several singular vectors at the lower boundary of the bulk distribution, and to one singular vector at the upper boundary. It must be noted, however, that the results of the Harmony algorithm varied slightly among runs (caused by the random initialization of cluster centroids in the clustering steps), and they seemed to vary more for low d. In some cases with $d = 2$, the batch effect did not spread over multiple vectors and the result was similar to those for higher values of d. For moderate values of d, the batch effect consistently showed up along a single singular vector at the lower bulk boundary. Like fastMNN, Harmony was ineffective for $d = 600$ due to noise from all the extra principal components. Also, akin to fastMNN batch correction, Harmony batch correction led to the loss of one rank due to centering for the PCA, but strangely enough, scaling the rows and columns did not remove another rank, unlike with fastMNN. It is yet unclear why this happens.

With Seurat, the high SSV was also removed, and the batch effect moved to one or more singular vectors at the upper edge of the bulk distribution. Seurat batch correction produced two singular values at 0, which, as with fastMNN, can be attributed to a scaling step prior to dimension reduction, and to scaling the rows and columns before computing the SVD.

In Fig. 3.5C, the largest singular values that could be associated with a batch effect were determined for different values of d. Batch effect association was determined by applying a t-test on the right singular vector corresponding to the singular value (this procedure was described in Section 2.3). Since the magnitude of a singular value reflects the amount of variance in the data that is explained by its corresponding singular vector, small *batch effect-associated singular values* (BSVs) indicate that the importance of the batch effect has been sufficiently minimized to a point where it contributes less to the total variance than cell-specific noise. Hence, the lower the maximum BSV after batch correction, the better the result is considered. In this respect, fastMNN gave better results for $d \leq 50$ while Harmony performed better for moderate to high values of d, except for $d = 600$. For Seurat, the number of dimensions seemed to have little effect on the location of the batch effect among the singular values. Note that Seurat was only tested for up to $d = 250$, because the maximum number of dimensions to be chosen for CCA was 299 (the size of the smallest batch minus 1), and the next value of d to be tested with the other two methods was $d = 300$.

In conclusion, the strategy for overcoming the low dimensionality of fastMNN and Harmony batch correction worked, and all three batch effect correction methods were effective in reducing the importance of the batch effect with low to moderate values of the dimensionality d. Since the magnitude of the BSV reflects the importance of the batch effect in the data, fastMNN can be considered the most effective, followed by Harmony and Seurat. In the next section, we will evaluate the performance of the three methods on a data set with more than one cell type and with unbalanced cell type compositions per batch, which was simulated in Scenario 4.

Comparing fastMNN, Harmony, and Seurat in Scenario 4

The three batch correction methods were tested on simulation scenario 4, which consisted of two batches, one of which contained four cell types, while the other batch contained only three of the four cell types. To assess the performance of the three methods in this scenario, we used the three batch mixing metrics kBET, LISI, and the local KL divergence (Fig. 3.6A-C). For kBET and local KL, low values indicate good mixing of the batches and therefore, better batch effect correction. For LISI on the other hand, higher values indicate better batch mixing. As can be observed, fastMNN achieved the best batch mixing for $d = 10$ according to all three

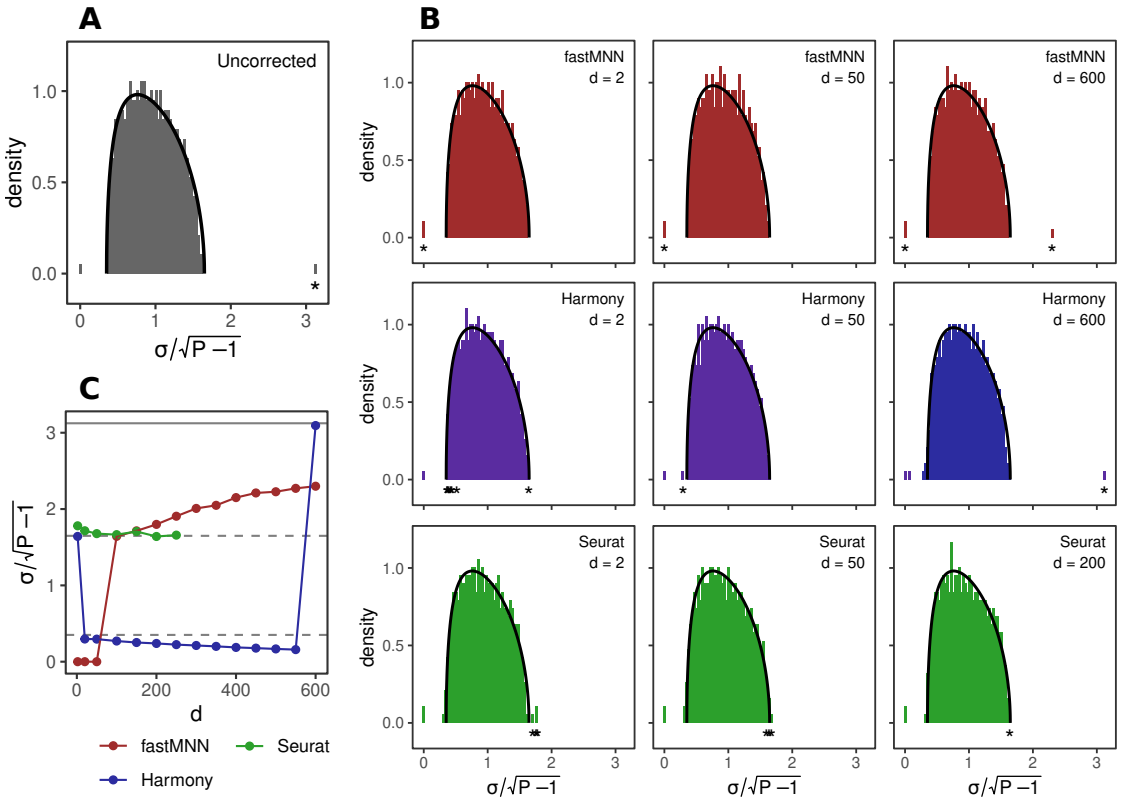


Figure 3.5: Batch correction in a simple simulated scenario with two batches of one cell type. In (A), the singular value histogram before batch correction is shown. The asterisk under the largest singular value indicates a significant difference in means between batches along its corresponding right singular vector (Welch's t-test, $p < 0.05$). In (B), the three batch effect correction methods were applied with low, intermediate and high dimensionality parameters d. For fastMNN and Harmony, d refers to the number of principal components on which the correction was applied as described in Section 2.2. For Seurat, d refers to the number of canonical correlation vectors used for determining the anchor pairs. Again, asterisks under singular values indicate the presence of a batch effect along the corresponding right singular vector. In (C), the maximum singular value for which a batch effect was present along its right singular vector was followed for the three batch correction methods with $d = 2, 20, 50$, and from there in steps of 50 until $d = 600$ (in the case of Seurat only until $d = 250$). The dashed grey lines represent the theoretical boundaries of the bulk distribution, and the solid gray line at the top represents the largest singular value before batch correction.

metrics, getting progressively worse for higher values of d (although still better than before batch correction; this is shown at $d = 0$). With Harmony, batch mixing was greatly improved as well, regardless of the value of d. Seurat seemed to perform the worst, since the three metrics did not improve much compared to before batch correction (especially kBET and KL).

In Figure 3.6D, all singular values have been plotted before and after batch effect correction. Before batch correction (at $d = 0$), four SSVs were present above the bulk of the singular value distribution, three of which were batch effect-associated (according to the Anderson-Darling test, $p_{BH} < 0.05$). After batch correction with fastMNN, only three SSVs remained (for all $d \in 10, \ldots, 100$). The batch effect was retained in the third singular vector, which can be expected because the batches did not only contain technical differences but also biological differences. Additionally, a BSV appeared at a singular value of 0, indicating a reduction in the importance of the batch effect. With Harmony, a similar pattern could be observed, except that BSVs appeared at the lower end of the bulk distribution, and that the choice of d influenced the third singular value: the higher d, the lower the importance of the third singular vector in the corrected data. With Seurat, batch correction also removed one of the SSVs, but introduced a batch effect at the first singular value (except with $d = 20$) and at a number of singular values at the upper edge of the bulk distribution. Also, the magnitudes of all three remaining SSVs were impacted by the choice of d.

Concluding, fastMNN and Harmony both seem to give clean results by removing one singular value from among the four SSVs, while presumably keeping biological differences between the batches intact. Unfortunately, the devised Anderson-Darling test for identifying batch effects along singular vectors could not distinguish between the technical part of the batch effect that needed to be removed and the biological part that was of interest (i.e., the extra cell type that was present in only one of the batches). In the following section, we will continue by applying the batch effect correction methods to two actual scRNA-seq data sets, to investigate whether batch effects can be successfully removed in light of the signal measurement angle (SIGMA).

3.3 Empirical data

Dendritic cell data

After investigating how batch effects and biological effects are manifested in the singular values and vectors of simulated scRNA-seq data, and after observing how the three batch correction methods affect these manifestations, we now turn to a real data set. The dendritic cell (DC) data set contained two batches with four types of DCs: pDCs, CD1C cells, CD141 cells, and double negative cells. After preprocessing and feature selection, the data set contained 749 cells (371 in batch 1 and 378 in batch 2). The data are visualized in Figure 3.7 with a uniform manifold approximation and projection (UMAP), a nonlinear dimension reduction technique that aims to perserve the local as well as the global structure of the data. As expected, four very distinct clusters appeared, reflecting the four dendritic cell types present. Additionally, a slight batch effect could be observed: within each cluster, the cells grouped together by batch. Also, a small number of cells fell slightly outside some of the clusters. These cells may be regarded as outliers, and either have a biological cause (e.g., a few cells of an unknown cell type are present in the data) or were the result of technical noise (e.g., some cells are of bad quality).

First, it was necessary to find out which values of the dimensionality parameter d worked best for each of the three batch effect correction methods, which was decided based on the three batch mixing metrics kBET, LISI, and local KL divergence, and on the locations of the BSVs. In Figure 3.8A-C, it can be seen that kBET reported the lowest rejection rate for fastMNN around $d = 50$, LISI gave the highest index for $d = 100$, and the KL divergence is the lowest again

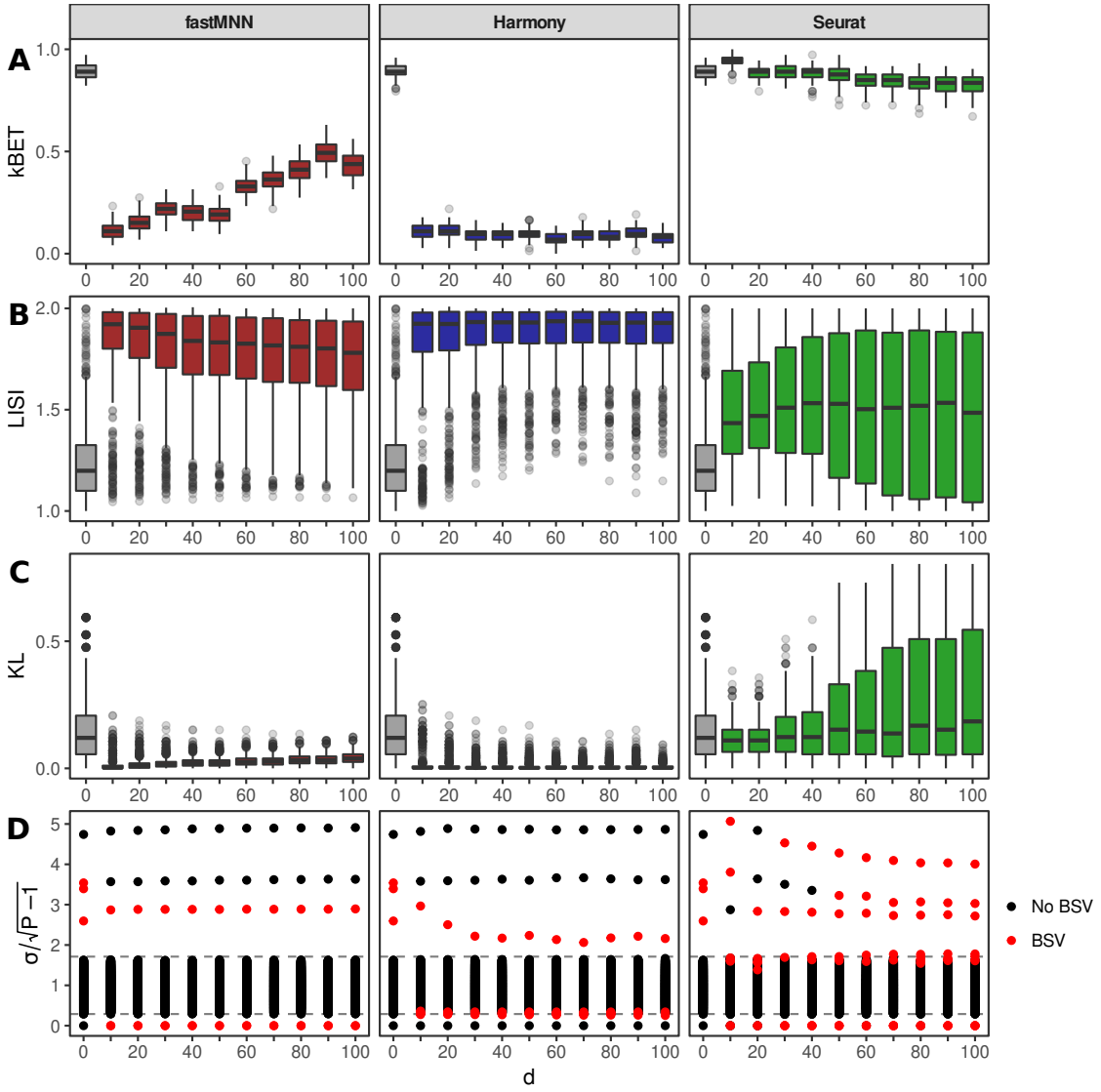


Figure 3.6: Batch effect correction in the fourth simulation scenario. In this scenario, one of the batches contained only three of the four cell types, leading to different cell type compositions per batch. Three batch mixing metrics were utilized to quantify the efficacy of the three batch correction methods with respect to the dimensionality parameter d (A-C). The grey box plots at $d = 0$ reflect batch mixing before batch effect correction. With kBET (A) and the local KL divergence (C), lower values indicate better batch mixing, whereas with LISI (B), higher values indicate better batch mixing. In (D), the singular values of each corrected matrix were plotted against d. The dashed horizontal lines delineate the lower and upper bounds of the theoretical Marchenko-Pastur distribution given the dimensions of the expression matrix. The red dots indicate batch effect-associated singular values (BSVs).

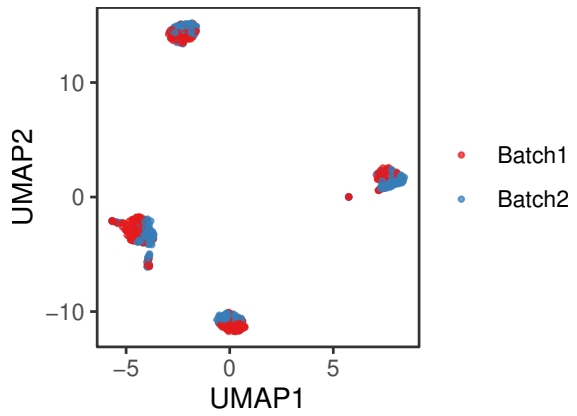


Figure 3.7: Uniform Manifold Approximation and Projection (UMAP) of the dendritic cell (DC) data before batch correction. The colors represent the batch from which each cell originates.

around $d = 40$ and $d = 50$. Oddly enough, batch correction with Harmony resulted in worse batch mixing scores than with the uncorrected data (i.e., higher kBET rejection rates and KL divergences, and lower LISIs), and the value of d did not make much of a difference. For Seurat, $d = 10$ seemed to give the best batch mixing scores.

For choosing the best d, the locations of batch effect-associated singular values within the singular value histograms were also considered (Figure 3.8D), where low batch effect-associated singular values were preferred. For fastMNN, the batch effect appeared at a singular value of 0 for every d. Therefore, the optimal d for fastMNN was decided to be $d = 50$ based on kBET and the local KL divergence. With Harmony, batch effect-associated singular values appeared at the upper boundary and lower boundary of the bulk distribution for $d = 10$ and $d = 20$, but only at its lower boundary from $d = 30$ onward. Since the three batch mixing metrics did not differ much with respect to d, the dimensionality for Harmony was set at $d = 30$. With Seurat, batch effect-associated singular values were present at both bulk boundaries until $d = 50$, so the choice of d was based solely on the batch mixing matrix, which were optimal for $d = 10$.

Let us now inspect the singular value histograms before and after batch correction with the chosen values of d (Figure 3.9). The three highest singular values signify the distinctions between the four cell types. Before batch correction, several additional SSVs were situated just above the bulk distribution, most of which were associated with a batch effect.

Batch correction had a similar effect on the distribution of singular values as to what was observed for the simulated data: while fastMNN moved the batch effect to a single singular value at 0, Harmony moved it to a few singular values on the lower end of the bulk distribution, indicating that the importance of the batch effect was reduced by a lot. This result is puzzling, because it contradicts the conclusions of the batch mixing metrics. We have not been able to establish the cause of this contradiction. Lastly, Seurat also performed well this time, moving the batch effect to below the bulk of the distribution. Additionally, the bulk distribution after batch correction with Seurat seemed to (visually) fit the theoretical MP distribution slightly better than in the other cases.

With all three correction methods, several SSVs remained above the upper bulk boundary, even though the batch effect was removed from their corresponding singular vectors. Taking a look at the elements of the fourth singular vector before and after batch correction (Figure 3.9), we can see that this is likely due to the presence of the aforementioned small number of outliers.

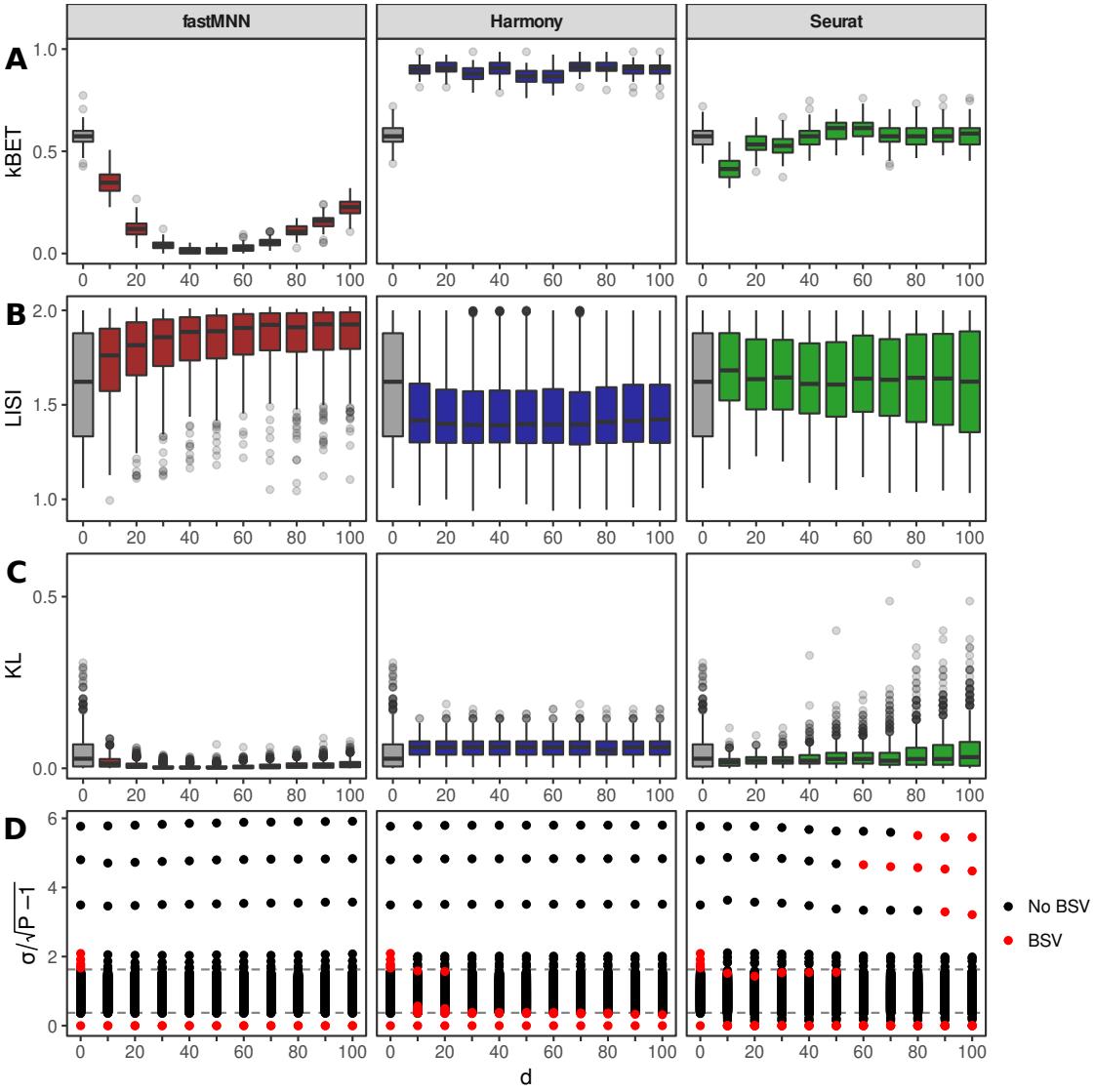


Figure 3.8: Batch effect correction of the dendritic cell data. Three batch mixing metrics were used to assess the efficacy of the three batch correction methods with respect to the dimensionality parameter d (A-C). With kBET (A) and the local KL divergence (C), lower values indicate better batch mixing, whereas with LISI (B), higher values indicate better batch mixing. The grey box plots at $d = 0$ reflect batch mixing before batch effect correction. In (D), the scaled singular values of each corrected matrix were plotted against d. The dashed horizontal lines delineate the lower and upper bounds of the theoretical MP distribution given the dimensions of the expression matrix. The red dots indicate BSVs.

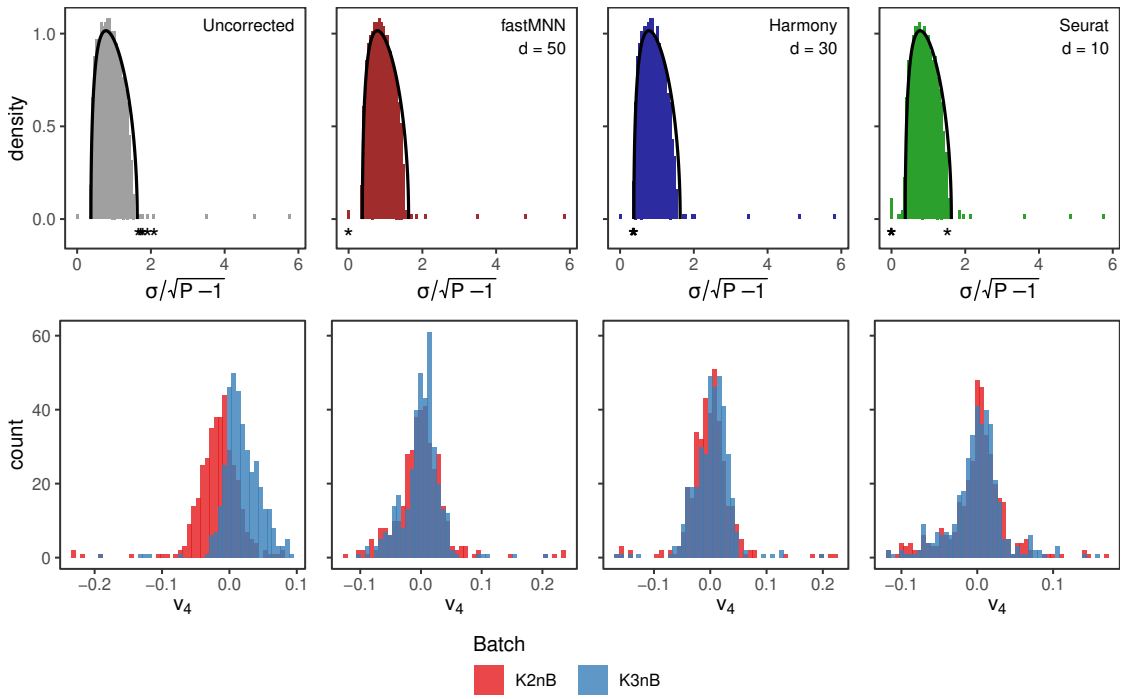


Figure 3.9: Singular value histograms and the fourth singular vectors of the dendritic cell data before and after batch correction. The choice of the dimensionality d for each method was based on the three batch mixing metrics and on the locations of batch effect-associated singular values, deciding on $d = 50$ for fastMNN, $d = 30$ for Harmony, and $d = 60$ for Seurat. The singular value histograms including theoretical MP densities (solid black lines) are displayed in the upper row, where asterisks mark BSVs. The lower row depicts histograms of the elements of the fourth right singular vectors.

Removing them might also remove those SSVs (but no outlier removal was attempted).

A note of caution should be made here: when evaluating specific singular vectors (for example, when we evaluate the fourth singular vector in Fig. 3.9), we cannot be certain that it still represents the same source of variance across the different batch correction results. With the first simulation scenario, we have seen that when successful, batch correction reduced the importance of the batch effect by "moving down" the BSV to zero or closer to zero (Fig. 3.5). As a consequence, some if not all singular vectors changed in their order of importance: what was formerly the first singular vector (corresponding to the largest singular value), became the last singular vector (corresponding to the smallest singular value). Unfortunately, it seems very hard to uniquely identify singular vectors in general (if there is any continuity at all between the different batch correction results), complicating the comparison of singular vectors.

In conclusion, with this data set it was possible to observe the batch effect in the SVD, and all three batch effect correction methods performed well in reducing the importance of the batch effect in the singular value distribution, given that the optimal d was chosen. Of the three, fastMNN gave the cleanest result. This means that if SIGMA were to be computed on the cell clusters, we can be confident that it is not being inflated because of batch effects. As we already observed in simulation scenario 4, and as we will see again in the next section, associating batch effects to singular values gets more problematic with less balanced data sets.

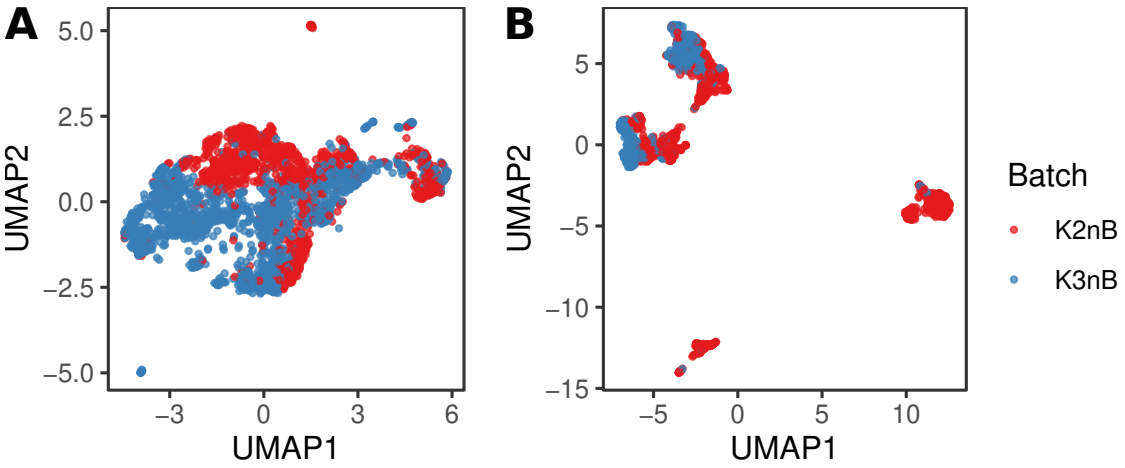


Figure 3.10: UMAPs of the lymphocyte scRNA-seq data before batch correction. The UMAPS were calculated and plotted separately for Patient 1 (A) and Patient 2 (B).

Lymphocyte data

The lymphocyte data set contained lymph node samples from two patients. From both patients, two biological samples were taken, the second a year after the first. The two samples are referred to as "K2nB" and "K3nB", and we will regard them as batches for batch correction. After quality control, the two batches of Patient 1 contained 1090 and 1776 cells, and the two batches of Patient 2 contained 1489 and 774 cells. The top 20% highly variable genes were selected, resulting in 1711 genes for Patient 1 and 1487 genes for Patient 2. Note that in contrast to the dendritic cell data, the number of (selected) genes for each of the two patients was less than the total number of cells, which was not the case in any of the simulation scenarios nor for the dendritic cell data. Also, there was no *a priori* information available on the cell types that were contained in the data except from the fact that they are lymphocytes. The data from two patients were analysed separately. UMAPs of the two data sets prior to batch correction are shown in Figure 3.10. In both cases, a batch effect was present. With Patient 1 (Fig. 3.10A), there were two small groups of cells (in the top left and in the bottom of the plot) at some distance from the main groups of cells, one of which originated from K2nB, and the other from K3nB. With Patient 2 (Fig. 3.10B), four clusters could be discerned in the UMAP, two of which were populated by both batches, but the other two of which were mostly composed of cells from K2nB.

As with the previous data set, batch correction was performed with fastMNN, Harmony, and Seurat using dimensionality parameters d that were chosen by evaluating batch mixing and the batch effect-associated singular values. In Figure 3.11A-C, batch mixing was evaluated for the data from Patient 1. For fastMNN, the choice of d did not seem to matter a lot; kBET and the local KL divergence seemed to be optimal around $d = 50$, whereas LISI was optimal for slightly lower d (but it made little difference). For Harmony, kBET did not improve much with any value of d compared to $d = 0$ (no correction), while LISI and local KL divergence did show some improvement, with all three metrics becoming slightly worse for higher d. For Seurat, the best metrics were obtained for $d = 10$.

The number and location of BSVs (Figure 3.11D) were not very informative for deciding on an optimal dimensionality, except perhaps in the case of Harmony, where several values of d led to a reduction in the number of BSVs. Therefore, the best values of d were decided to be $d = 50$

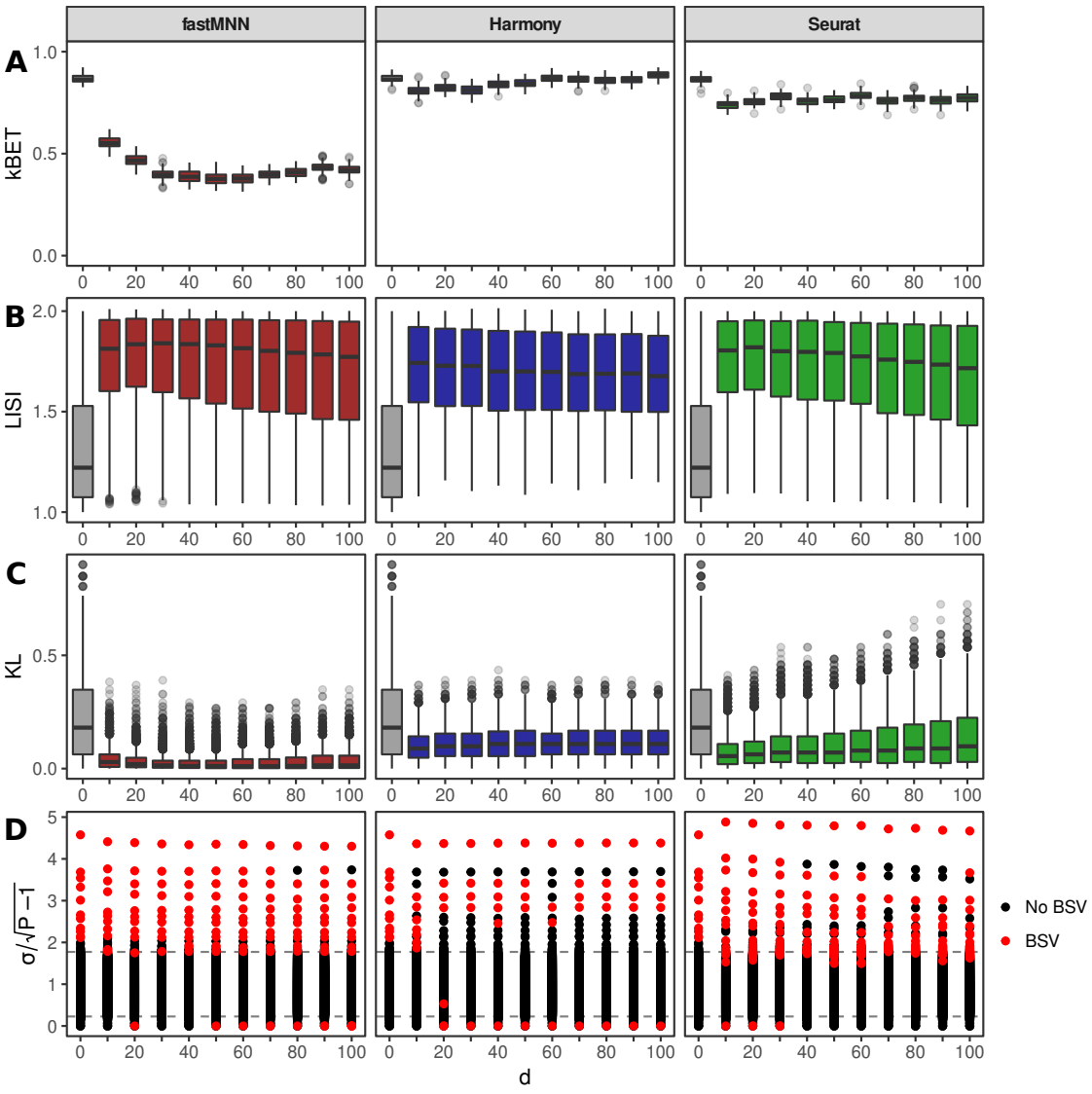


Figure 3.11: Batch effect correction of the lymphocyte data of Patient 1. The three batch mixing metrics kBET (A), LISI (B), and the local KL divergence (C) were used to assess the efficacy of the three batch correction methods with respect to the dimensionality parameter d. The grey box plots at $d = 0$ reflect batch mixing before batch effect correction. In (D), the scaled singular values of each corrected matrix with respect to d are shown. The dashed horizontal lines delineate the lower and upper bounds of the theoretical MP distribution given the dimensions of the expression matrix. The red-coloured singular values indicate BSVs.

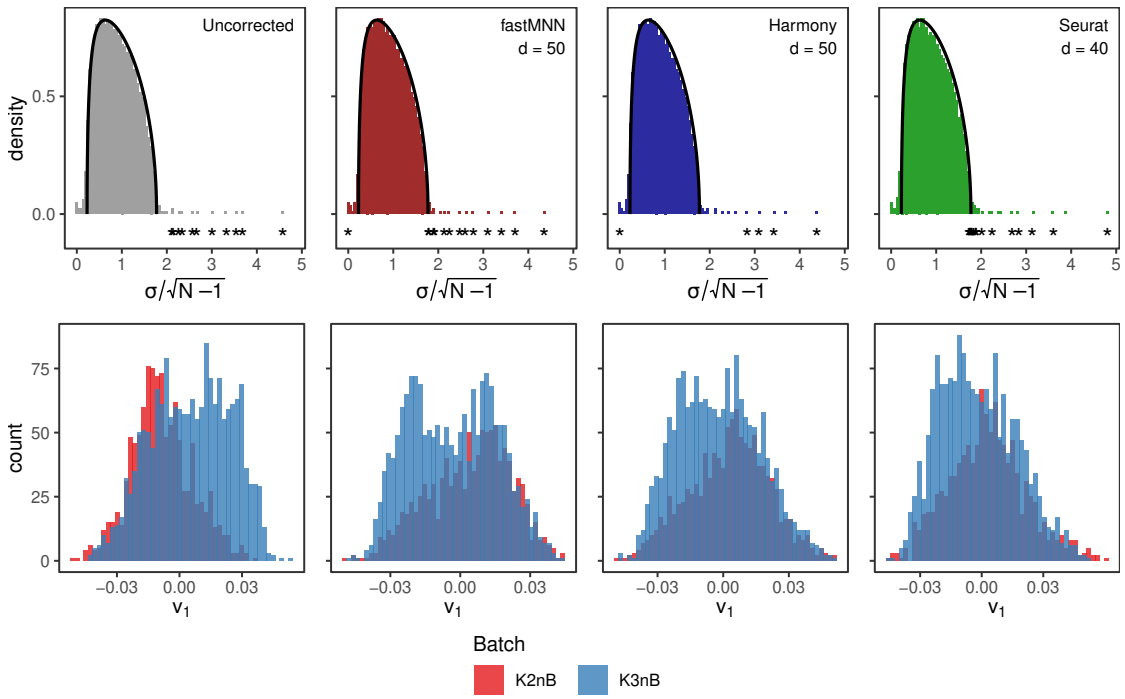


Figure 3.12: Patient 1 singular value histograms and the elements of the first singular vectors before and after batch correction. The upper row shows singular value histograms including the theoretical MP densities, with asterisks marking the BSVs. The lower row shows the elements of the first singular vector. The choice of the parameter d for each batch correction method was based on the batch mixing metrics and on the number and location of batch effect-associated singular values.

for fastMNN and Harmony, and $d = 40$ for Seurat. For fastMNN, the batch mixing metrics were leading for the choice of d because there was only little difference in batch-associated singular values. For Seurat, $d = 40$ seemed to result in slightly fewer BSVs than for lower d.

The resulting singular value histograms and the elements of, in this case, the first singular vectors after batch correction with the three methods are shown in Figure 3.12. Neither of the methods seemed to reduce the number of BSVs among the SSVs. Notably, the highest singular value remained batch effect-associated for all three methods. The presence of the batch effect in the corresponding (first) singular vector became clear when examining the elements of the vector (shown in the lower row of Figure 3.12): the elements from the two batches followed different distributions, both before and after batch correction. However, this does not mean that the batch correction was necessarily unsuccessful. In fact, removing technical effects while preserving meaningful biological differences between the batches is exactly the purpose of the three correction methods in question. It may very well be that the differences along the first singular vector reflected a biological effect that was correctly preserved during batch correction.

For Patient 2, batch correction with fastMNN seemed to improve the batch mixing metrics only very slightly compared to before batch correction (Fig. 3.13A-C), but seemed to improve a little with increasing d. They also changed little, if not worsened, after Harmony batch correction. Seurat seemed to bring about the best mixing of batches according to the three metrics: especially LISI and the local KL divergence were optimal at low d. From the number of BSVs before and

after batch correction (Fig. 3.13D), it can be concluded that both fastMNN and Harmony reduced the number of singular vectors containing a batch effect, which was also the case with the dendritic cell data but not with Patient 1.

Based on Figure 3.13, $d = 50$ was chosen for fastMNN batch correction (admittedly, a quite arbitrary choice), $d = 30$ for Harmony, and $d = 10$ for Seurat. The resulting singular value histograms and first singular vectors (Figure 3.14 reveal that the batch effect was still present in the first singular vector, and guessing from the singular vectors depicted, it was most likely confounded with the biological effect of the suspected extra cell type in batch K2nB. In simulation scenario 4, we have seen the same: even though batch correction seemed successful, the batch effect was still present among SSVs, because of the biological variation between batches.

In conclusion, it seems that the more unbalanced the data, the harder it becomes to identify solely the technical batch effect. The batch effect also stays present among many of the SSVs after batch correction, probably because of confounding with biological effects. The batch mixing metrics seemed to favor fastMNN for Patient 1, and Seurat for Patient 2 (but only slightly), so perhaps Seurat works a little better in unbalanced cases.

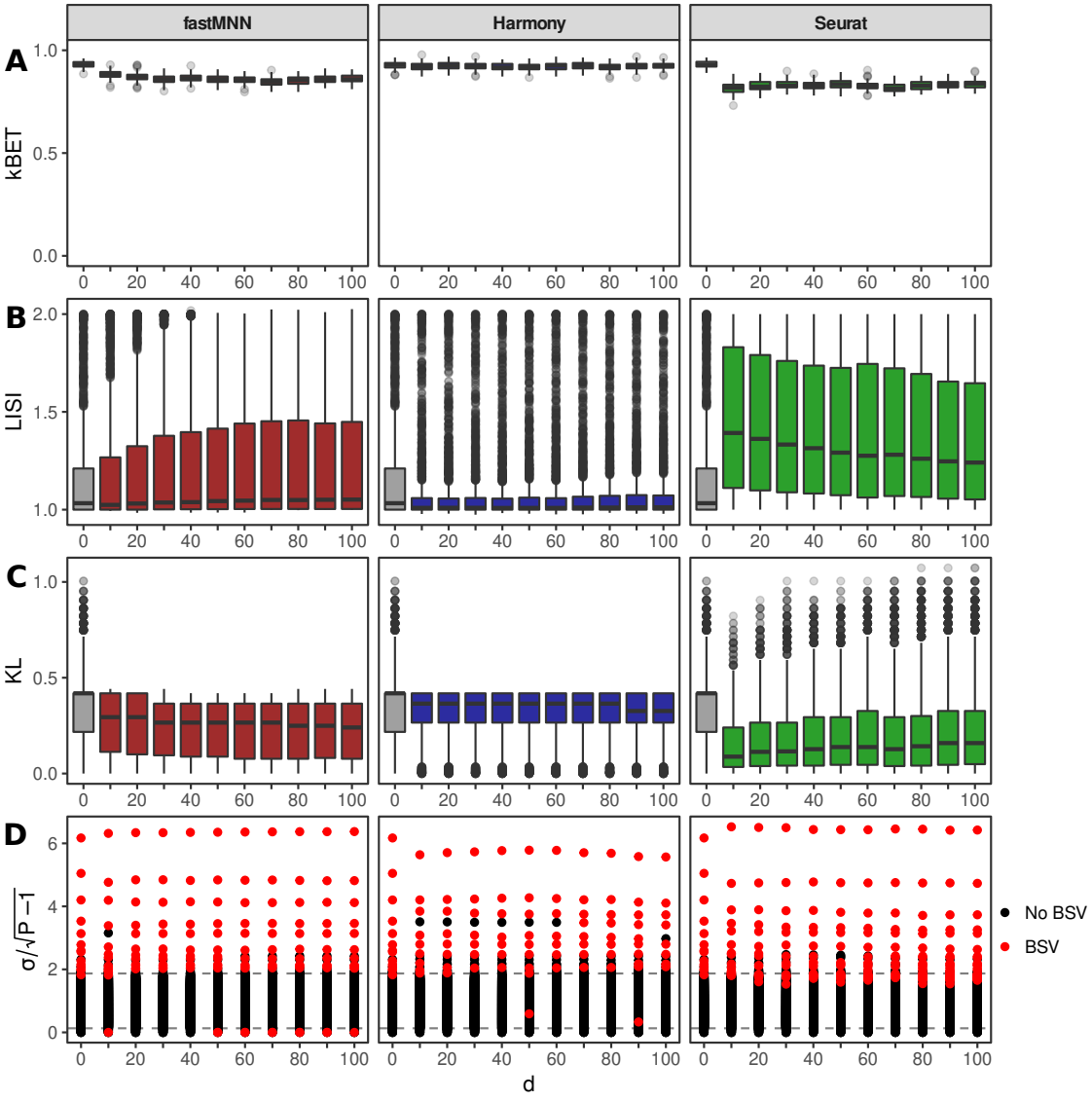


Figure 3.13: Batch effect correction of the lymphocyte data of Patient 2. The three batch mixing metrics kBET (A), LISI (B), and the local KL divergence (C) were used to evaluate the performance of the three batch correction methods with respect to the dimensionality d. The grey box plots at $d = 0$ reflect batch mixing before batch effect correction. In (D), the scaled singular values of each corrected matrix with respect to d are shown. The dashed horizontal lines delineate the lower and upper bounds of the theoretical MP distribution given the dimensions of the expression matrix. The red-coloured dots indicate BSVs.

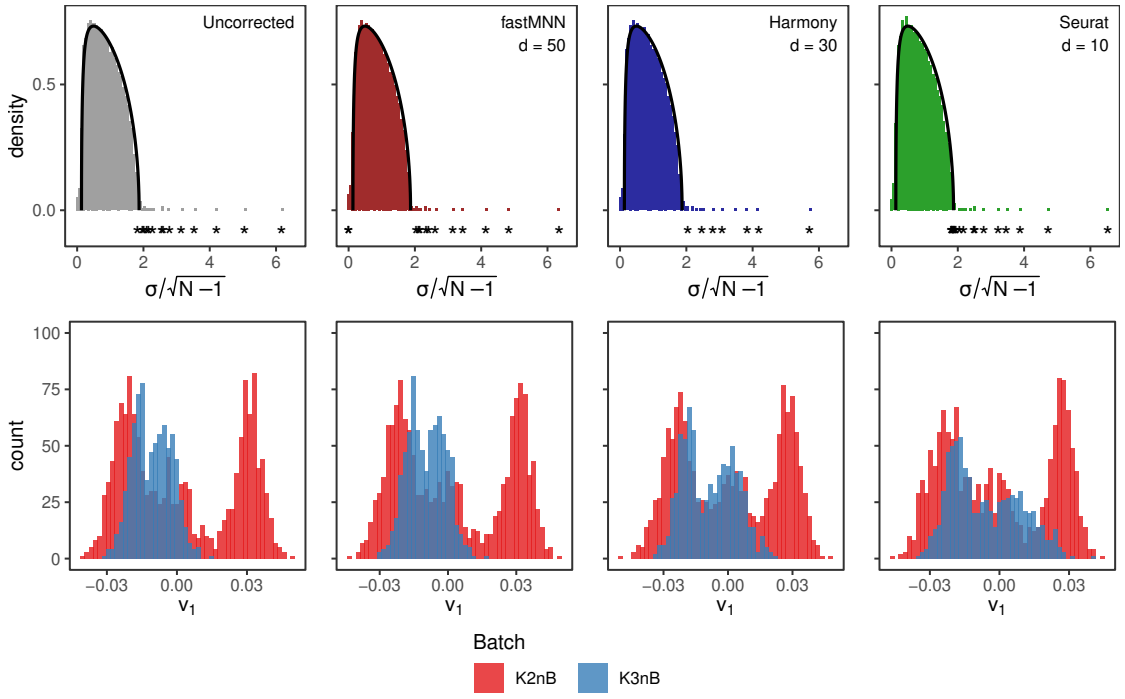


Figure 3.14: Patient 2 singular value histograms and first singular vectors before and after batch correction. The upper row shows the singular value histograms including the theoretical MP densities, with asterisks marking the BSVs. The lower row shows the elements of the first right singular vector. Again, the choice of d was based on the three batch mixing metrics as well as the number and locations of batch effect-associated singular values.

Chapter 4

Discussion

4.1 Conclusions

Best batch effect correction method

In the simulated scenarios, all three batch effect correction methods proved efficient in removing batch effect-associated singular values (BSVs) from among the signal singular values (SSVs) in the singular value histograms of the expression matrices. Each method affected the singular value distributions in a specific way: after fastMNN correction, the batch effect was manifested at a singular value of 0, with Harmony, it appeared on the lower end of the bulk distribution of singular values, and with Seurat, it appeared at the upper end of the bulk distribution. In this sense, fastMNN can be regarded as the most effective, because any trace of a batch effect had been moved to a singular value of 0, removing all of its importance from the data. Interestingly, although the importance of the batch effect was reduced by batch correction, it was never completely removed from the data, as it could still be detected along at least one of the singular vectors.

With the dendritic cell data, the batch effect was manifested slightly above the bulk distribution of singular values, among SSVs that were probably caused by outliers. Batch effect correction was successful, showing the same pattern of BSVs after batch correction as with the simulated data. According to the batch mixing metrics, fastMNN achieved the best batch mixing of the three methods. Most of the singular values previously containing the batch effect remained in the same positions after batch correction, suggesting that they would also have been present if there had been no batch effect. Therefore, the batch effect spreads over singular vectors that also contain other signals (in this case, the presence of outliers). This phenomenon was also observed in the simulated "2c2b" scenario (see Figure 3.3). Presumably, the outliers contributed to similar proportions of variance in the data as the batch effect.

With the lymphocyte data, it was much harder to conclude whether batch correction was successful, because SSVs were still associated with batch effects (according to the Anderson-Darling test) after correction. This can most likely be attributed to the lack of balance with respect to batch size and cell type composition, which causes biological differences between the batches. A good batch correction method retains the biological differences while removing the technical differences between batches, but the AD-test was not able to distinguish between these two types of effects. For Patient 1, the batch mixing metrics seemed to favor fastMNN, whereas for Patient 2, Seurat seemed to give the best batch mixing. The behavior of BSVs after batch correction did not match the patterns observed in the simulations and with the dendritic cell data: although the number of BSVs was reduced, some of the remaining BSVs still showed up

above the bulk distribution. In this respect, Harmony reduced the number of BSVs by the most, retaining only two BSVs for both patients.

Concluding, fastMNN and Harmony seemed to perform the best in most cases, based on the batch mixing metrics and sometimes on the fates of BSVs in the singular value distribution. Only for the lymphocyte data of Patient 2, Seurat seemed to perform better.

Implications for SIGMA

As SIGMA is usually based on the largest signal singular value (SSV) in the data, it is important to consider (1) how batch correction affects the largest SSV, and (2) whether the largest SSV is associated with batch effects before and after batch correction. In the analysis of the dendritic cell data, we have seen that the largest SSV changed slightly with batch correction. In the lymphocyte data, it was also slightly altered. The effects of batch correction on the largest singular value were not consistent: in some cases the resulting singular value was a bit higher, in other cases a bit lower than before batch correction. In conclusion, SIGMA will not substantially change after batch correction of these data sets, indicating the presence of heterogeneity. Note that SIGMA does not always use the highest SSV in its computation. This is because the algorithm contains a regression step that aims to correct singular values for the influence of unwanted sources of variability between cells. Some SSVs might be more affected by this correction than others, which may in some cases change the order of the SSVs. If the SSV reflects a true biological source of heterogeneity, however, it is not expected to change by much.

As for the second consideration, the largest singular values in the lymphocyte data were batch effect-associated, and remained so after batch correction, most likely due to confounding with a biological effect. Caution should be taken when calculating SIGMA if this is the case. If the batch effect is confounded with biological effects (i.e., batches are unbalanced), then the SSV is "genuine" because it reflects a *biological* effect, but it is still unclear how severely the singular value (and therefore SIGMA) is affected by the *technical* part of the batch effect.

4.2 Limitations

Validity of batch mixing metrics on unbalanced data

With the real data sets, the three batch mixing metrics kBET, LISI, and the local KL divergence mostly agreed with each other on what was the best batch correction method and dimensionality parameter, but not always. For example, kBET did not report a great improvement in rejection rate after batch correction of the lymphocyte data of Patient 1, while LISI and the local KL improved substantially. Also, the batch mixing metrics generally did not correspond very much to the locations of BSVs for different values of d in any of the data sets. In the dendritic cell data, for instance, kBET and KL were optimal around $d = 50$ with fastMNN, but the batch effect was manifested at 0 for all $d \in \{10, 20, \ldots, 100\}$. In general, the distribution of BSVs among the singular values changed most for the simulated data and the dendritic cell data, and least for the lymphocyte data.

Other batch correction parameters

The parameter that we focused on in order to optimize the three batch effect correction methods was the dimensionality d. With fastMNN and Harmony, d referred to the number of principal components on which batch correction was applied, and with Seurat it referred to the number of canonical correlation vectors used to find the anchor pairs. Of course this is not the only variable

that can be tweaked in the three algorithms. In fastMNN, the number of nearest neighbours to find was left on the default $k = 20$. In Seurat, three related parameters $k.anchor$, $k.filter$, and $k.score$ can be chosen for the number of anchors to use in anchor finding, filtering and scoring, respectively. These parameters were left on their default values as well. The parameters θ and σ governing cluster size and diversity in Harmony were also not changed from their default values. All these parameters could influence the performance of the batch correction methods with the two data sets.

Assessing the presence of batch effects along singular vectors

We used Welch's t-test for detecting batch effects in the singular vectors in the simplest simulation scenario. To accommodate the non-normal distributions along singular vectors of more complex data sets, an Anderson-Darling test was used instead. Unfortunately, any test that distinguishes batches from each other is not able to distinguish technical effects from biological effects.

Multiple batches

In this thesis, we have only dealt with integrating two batches of data. In the presence of more than two batches, a t-test is not sufficient for detecting batch effects along singular vectors. A natural generalization would be to utilize a one-way ANOVA with an F-test for equality of batch means. Since the ANOVA may be more sensitive to non-normality of the batches, however, perhaps it is better to utilize one of the batch mixing metrics here, as all of them are suitable for analyzing multiple batches.

Alternatively, it seems that the Anderson-Darling test can also be generalized to more than two samples [23]. Ideally, the ad.test() function from the kSamples package [23] should be rewritten to run in a vectorized manner, which was only implemented for $k = 2$ samples in this thesis.

Inspecting histograms at cluster level

All the examined singular value histograms were calculated from complete expression matrices. However, SIGMA is normally calculated separately for clusters within the data, to assess whether sub-clustering is needed. Therefore, it makes sense to examine the singular value histograms and the presence of batch effects on a cluster level as well. However, it may not be very straightforward to do this, because the clusters are based on batch-corrected data. If the batch effect can still be detected in a cluster, it entails that the batch effect correction was not successful and, consequently, that the clusters may well be incorrect.

4.3 Suggestions for further work

Estimating biological effects in singular vectors

The detection of batch effects along singular vectors would become much more useful if it was possible to distinguish between technical and biological effects. With the knowledge of biological clusters, it may be possible to assess whether the variance along a singular vector is best explained by the batches or by cell clusters. Still, if the batch effects are confounded with biological effects, this is not very meaningful, because the biological effect is contained in the batch effect. Moreover, as mentioned in section 4.2, the clustering procedure also depends on the success of batch correction, and therefore it complicates comparisons between batch effects and biological effects.

Regressing out batch effects while computing SIGMA

Ultimately, we want to know if SIGMA reports genuine biological heterogeneity in a cluster. A possible way to adapt SIGMA is to correct the singular values by reducing them based on an estimate of the amount of variance explained by the batches. In fact, a functionality already exists in the SIGMA package that enables the user to compensate for unwanted technical variability. It works by creating linear models with the right singular vectors $\mathbf{v}_i$ as dependent variables and specified covariates such as the library sizes of the cells as independent variables. The adjusted R^2 of the models are used to shrink the eigenvalues: $\tilde{\lambda_i} = (1 - R^2_{adj,i})\lambda_i$, where of course $\lambda_i = \sigma_i^2$, the square of singular value in question. Since the adjusted R^2 reflects the proportion of variance in the singular vector explained by the covariates, and the eigenvalue represents the variance in the data explained by the singular vector, it makes sense to scale the eigenvalue in such a way. The calculation of SIGMA then proceeds with the shrunken singular values $\tilde{\sigma_i} = \sqrt{\tilde{\lambda_i}}$. To regress out the batch effects using this method, we could simply add the batch assignments as dummy variables in the design matrix of the linear model.

Unfortunately, the possibility of the batch effect being confounded with meaningful biological effects is again problematic, because it will probably cause SIGMA to be reduced too much, and therefore the heterogeneity in the cluster will be underestimated. This could be salvaged if we could find a way to distinguish between the "wanted" and "unwanted" parts of the batch effect along the singular vector, and only regress out the unwanted part. As current batch effect correction methods aim to do exactly this, but in a higher dimensional expression space, perhaps they can serve as inspiration.

Bibliography

[1] Florent Benaych-Georges and Raj Rao Nadakuditi. "The singular values and vectors of low rank perturbations of large rectangular random matrices". In: *Journal of Multivariate Analysis* 111 (2012), pp. 120–135. ISSN: 0047-259X. DOI: https://doi.org/10.1016/j.jmva.2012.04.019. URL: http://www.sciencedirect.com/science/article/pii/S0047259X12001108.

[2] Alina Beygelzimer et al. *FNN: Fast Nearest Neighbor Search Algorithms and Applications*. R package version 1.1.3. 2019. URL: https://cran.r-project.org/package=FNN.

[3] Andrew Butler et al. "Integrating single-cell transcriptomic data across different conditions, technologies, and species". In: *Nature Biotechnology* 36.5 (May 2018), pp. 411–420. ISSN: 1546-1696. DOI: 10.1038/nbt.4096. URL: https://doi.org/10.1038/nbt.4096.

[4] Maren Büttner et al. "A test metric for assessing single-cell RNA-seq batch correction". In: *Nature Methods* 16.1 (Jan. 2019), pp. 43–49. ISSN: 1548-7105. DOI: 10.1038/s41592-018-0254-1. URL: https://doi.org/10.1038/s41592-018-0254-1.

[5] Laleh Haghverdi et al. "Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors". In: *Nature Biotechnology* 36.5 (May 2018), pp. 421–427. ISSN: 1546-1696. DOI: 10.1038/nbt.4091. URL: https://doi.org/10.1038/nbt.4091.

[6] *Homo sapiens annotation report*. https://www.ncbi.nlm.nih.gov/genome/annotation_euk/Homo_sapiens/106/. Accessed: 2021-09-22. Feb. 2014.

[7] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. "Adjusting batch effects in microarray expression data using empirical Bayes methods". In: *Biostatistics* 8.1 (Apr. 2006), pp. 118–127. ISSN: 1465-4644. DOI: 10.1093/biostatistics/kxj037. eprint: https://academic.oup.com/biostatistics/article-pdf/8/1/118/25435562/batch+effect+supplementary+materials.pdf. URL: https://doi.org/10.1093/biostatistics/kxj037.

[8] Teemu Kivioja et al. "Counting absolute numbers of molecules using unique molecular identifiers". In: *Nature Methods* 9.1 (Nov. 2011), pp. 72–74. DOI: 10.1038/nmeth.1778. URL: https://doi.org/10.1038/nmeth.1778.

[9] Aleksandra A. Kolodziejczyk et al. "The Technology and Biology of Single-Cell RNA Sequencing". In: *Molecular Cell* 58.4 (May 2015), pp. 610–620. DOI: 10.1016/j.molcel.2015.04.005. URL: https://doi.org/10.1016/j.molcel.2015.04.005.

[10] Ilya Korsunsky et al. "Fast, sensitive and accurate integration of single-cell data with Harmony". In: *Nature Methods* 16.12 (Dec. 2019), pp. 1289–1296. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0619-0. URL: https://doi.org/10.1038/s41592-019-0619-0.

[11] Yalan Lei et al. "Applications of single-cell sequencing in cancer research: progress and perspectives". eng. In: *Journal of hematology & oncology* 14.1 (June 2021). PMC8190846[pmcid], pp. 91–91. ISSN: 1756-8722. DOI: 10.1186/s13045-021-01105-2. URL: https://doi.org/10.1186/s13045-021-01105-2.

[12] Rohan Lowe et al. "Transcriptomics technologies". In: *PLOS Computational Biology* 13.5 (May 2017), e1005457. DOI: 10.1371/journal.pcbi.1005457. URL: https://doi.org/10.1371/journal.pcbi.1005457.

[13] Malte D. Luecken and Fabian J. Theis. "Current best practices in single-cell RNA-seq analysis: a tutorial". In: *Molecular Systems Biology* 15.6 (2019), e8746. DOI: https://doi.org/10.15252/msb.20188746. eprint: https://www.embopress.org/doi/pdf/10.15252/msb.20188746. URL: https://www.embopress.org/doi/abs/10.15252/msb.20188746.

[14] Thomas Lumley et al. "The Importance of the Normality Assumption in Large Public Health Data Sets". In: *Annual Review of Public Health* 23.1 (2002). PMID: 11910059, pp. 151–169. DOI: 10.1146/annurev.publhealth.23.100901.140546. eprint: https://doi.org/10.1146/annurev.publhealth.23.100901.140546. URL: https://doi.org/10.1146/annurev.publhealth.23.100901.140546.

[15] Aaron T. L. Lun. *A description of the theory behind the fastMNN algorithm*. https://marionilab.github.io/FurtherMNN2018/theory/description.html. Accessed: 2021-05-25.

[16] Aaron T. L. Lun, Karsten Bach, and John C. Marioni. "Pooling across cells to normalize single-cell RNA sequencing data with many zero counts". In: *Genome Biology* 17.1 (Apr. 2016), p. 75. ISSN: 1474-760X. DOI: 10.1186/s13059-016-0947-7. URL: https://doi.org/10.1186/s13059-016-0947-7.

[17] 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". eng. In: *F1000 research* 5 (2016), pp. 2122–2122. ISSN: 2046-1402.

[18] Evan Z. Macosko et al. "Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets". In: *Cell* 161.5 (May 2015), pp. 1202–1214. DOI: 10.1016/j.cell.2015.05.002. URL: https://doi.org/10.1016/j.cell.2015.05.002.

[19] Davis J. McCarthy et al. "Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R". In: *Bioinformatics* 33.8 (Jan. 2017), pp. 1179–1186. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw777. eprint: https://academic.oup.com/bioinformatics/article-pdf/33/8/1179/25150420/btw777.pdf. URL: https://doi.org/10.1093/bioinformatics/btw777.

[20] Maria Mircea et al. "A clusterability measure for single-cell transcriptomics reveals phenotypic subpopulations". In: *bioRxiv* (2021). DOI: 10.1101/2021.05.11.443685. eprint: https://www.biorxiv.org/content/early/2021/05/13/2021.05.11.443685.full.pdf. URL: https://www.biorxiv.org/content/early/2021/05/13/2021.05.11.443685.

[21] A. N. Pettitt. "A Two-Sample Anderson–Darling Rank Statistic". In: *Biometrika* 63.1 (1976), pp. 161–168. ISSN: 00063444. URL: http://www.jstor.org/stable/2335097.

[22] Tian Qin et al. "Single-cell RNA-seq reveals novel mitochondria-related musculoskeletal cell populations during adult axolotl limb regeneration process". In: *Cell Death & Differentiation* 28.3 (Oct. 2020), pp. 1110–1125. DOI: 10.1038/s41418-020-00640-8. URL: https://doi.org/10.1038/s41418-020-00640-8.

[23] Fritz Scholz and Angie Zhu. *kSamples: K-Sample Rank Tests and their Combinations*. R package version 1.2-9. 2019. URL: https://cran.r-project.org/package=kSamples.

[24] E. H. Simpson. "Measurement of Diversity". In: *Nature* 163.4148 (Apr. 1949), pp. 688–688. DOI: 10.1038/163688a0. URL: https://doi.org/10.1038/163688a0.

[25] Gordon K. Smyth and Terry Speed. "Normalization of cDNA microarray data". In: *Methods* 31.4 (2003). Candidate Genes from DNA Array Screens: application to neuroscience, pp. 265–273. ISSN: 1046-2023. DOI: https://doi.org/10.1016/S1046-2023(03)00155-5. URL: https://www.sciencedirect.com/science/article/pii/S1046202303001555.

[26] Tim Stuart et al. "Comprehensive Integration of Single-Cell Data". In: *Cell* 177.7 (2019), 1888–1902.e21. ISSN: 0092-8674. DOI: https://doi.org/10.1016/j.cell.2019.05.031. URL: https://www.sciencedirect.com/science/article/pii/S0092867419305598.

[27] Fuchou Tang et al. "mRNA-Seq whole-transcriptome analysis of a single cell". In: *Nature Methods* 6.5 (Apr. 2009), pp. 377–382. DOI: 10.1038/nmeth.1315. URL: https://doi.org/10.1038/nmeth.1315.

[28] V. A. Traag, L. Waltman, and N. J. van Eck. "From Louvain to Leiden: guaranteeing well-connected communities". In: *Scientific Reports* 9.1 (Mar. 2019), p. 5233. ISSN: 2045-2322. DOI: 10.1038/s41598-019-41695-z. URL: https://doi.org/10.1038/s41598-019-41695-z.

[29] Hoa Thi Nhu Tran et al. "A benchmark of batch-effect correction methods for single-cell RNA sequencing data". In: *Genome Biology* 21.1 (Jan. 2020), p. 12. ISSN: 1474-760X. DOI: 10.1186/s13059-019-1850-9. URL: https://doi.org/10.1186/s13059-019-1850-9.

[30] Cole Trapnell et al. "The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells". In: *Nature Biotechnology* 32.4 (Mar. 2014), pp. 381–386. DOI: 10.1038/nbt.2859. URL: https://doi.org/10.1038/nbt.2859.

[31] Alexandra-Chloé Villani et al. "Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors". In: *Science* 356.6335 (2017). ISSN: 0036-8075. DOI: 10.1126/science.aah4573. eprint: https://science.sciencemag.org/content/356/6335/eaah4573.full.pdf. URL: https://science.sciencemag.org/content/356/6335/eaah4573.

[32] Hadley Wickham et al. "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43 (2019), p. 1686. DOI: 10.21105/joss.01686.

[33] Luke Zappia, Belinda Phipson, and Alicia Oshlack. "Splatter: simulation of single-cell RNA sequencing data". In: *Genome Biology* 18.1 (Sept. 2017), p. 174. ISSN: 1474-760X. DOI: 10.1186/s13059-017-1305-0. URL: https://doi.org/10.1186/s13059-017-1305-0.

[34] Yi Zhang. *Dimension Reduction (PCA, ICA, CCA, FLD, Topic Models)*. https://www.cs.cmu.edu/$\sim$tom/10701_sp11/recitations/Recitation_11.pdf. Accessed: 2021-05-27.

[35] Grace Zheng. *Our 1.3 million single cell dataset is ready to download*. https://www.10xgenomics.com/blog/our-13-million-single-cell-dataset-is-ready-to-download. Accessed: 2021-09-22. Feb. 2017.