MASTER THESIS

# Batch Noise & Gene/Gene Correlations in Single Cell RNA Sequencing Data

Andreas Berglund

May 20, 2021

UNIVERSITY OF COPENHAGEN

# Batch Noise & Gene/Gene Correlations in Single Cell RNA Sequencing Data


| | |
|---|---|
| Author: | Andreas Berglund Nielsen |
| Supervisor: | Associated Professor Ala Trusina |
| Supervisor: | Alexander Valentin Nielsen |

Complex Models of Life
Niels Bohr Institute



Submitted to the University of Copenhagen
May 20, 2021


UNIVERSITY OF COPENHAGEN

## Abstract

This thesis investigates the influence batch noise has on the clustering and identification of different cell types as well as quantifying different gene/gene correlation patterns within scRNA-seq.

The influence of the batch noise is investigated through a series of simulations in which the cell type in our synthetic scRNA-seq data is represented using a colouring scheme inspired by the RGB colour code.

From our simulation we illustrated that a relatively small batch noise would make the clustering algorithm cluster based on batch rather than cell type. This is the case even when cells of the same cell type, but different batch is closer than cells of different cell type but same batch. Our investigation shows well defined regions of good clustering (based on cell type) and bad clustering (based on batch). Furthermore, we demonstrated that data correction methods can improve the clustering of our synthetic data by moving it from regions of bad clustering into regions of good clustering. The same applies when examining real scRNA-seq data.

We found from the fact that clustering based on batch occurs even when cells of the same cell type but different batch is closer than cells of different cell type but same batch illustrates that there are other aspects than the magnitude of the batch noise that causes inaccuracies in clustering.

Inspired by the method for identifying cellular populations we sat forth identifying the different gene/gene correlation patterns in scRNA-seq data. By vectorising the 2D histogram of the classical gene/gene scatter plot we could group similar gene/gene correlation patterns using a clustering algorithm. Using a custom distance measurement we managed to identify five different gene/gene correlation patterns.

# Acknowledgements

First of all, I would like to extent my deepest gratitude to associate professor Ala Trusina and Ph.d student Alexander V. Nielsen who have been two amazing supervisors throughout my master thesis. I have really enjoyed our collaboration and the inspiring and motivating talks we have had. I have always left our meetings with a better understanding and the motivation to explore more exciting questions. Even tough the corona virus has made this process a different experiment from what I had anticipated you have managed to help me keep my spirit high and for that you have my deepest thanks.

A special and heartfelt thanks goes to my girlfriend Caroline Nielsen for being an amazing moral support in these stressful times and for always supporting me and cheering me on. She has also earned my endless gratitude for proofreading all the spelling mistakes my dyslectic brain would otherwise have scattered throughout this thesis.

Finally, I would like thank the rest of my family and friends for always being willing to help in whatever way they could. Especially thanks to my mother Jytte for letting me mess up her living room whenever I needed a change of scenery from my own apartment.

# Table of Contents

# List of Figures

Chapter 1

# Introduction

Since the first published example of single cell RNA sequencing (scRNA-seq) in 2009 [19] the interest and success of this method have only increased over the years. The invention of scRNA-seq has allowed us to measure the expression level of nearly all genes in a single cell over hundreds of thousands individual cells. Our knowledge of some of the most vital questions in biology and medicine have been greatly advanced in large because of this method. Researchers have used scRNA-seq to discover, amongst other, many new and rare cellular populations (cell types), inferred new regulatory networks in gene expressions, as well as mapped the development path from a single cell to an organism with multiple organs and billions of different cells [15, 3]. The latter of which was awarded Science Magazines *Breakthrough of the Year* in 2018 [17]. Single cell RNA sequencing has allowed researchers to address biological questions that for many years had been out of reach.

The success of scRNA-seq has in recent years led to a substantial increase in more complex and large-scale experiments [1]. As an example this could be the construction of a comprehensive atlas of all human cell types and sub-types including their activity states, dynamic transitions, physical locations, and lineage relationships through development [22]. Such an extensive endeavour requires the joining of data from many different experiments with different compositions of cell types, taken from many different individuals and under different circumstances. Although, the new single-cell data offers tremendous opportunities they also pose a comprehensive challenge to the computational biology community [23].

One of the major problems facing the computational part of the analysis of single cell data is the relatively strong technical noise which can arise from many different parts of the scRNA-seq data pipeline [3, 20]. This technical noise is in nearly all cases very difficult to distinguish from real biological variations. A part of this noise, which in recent years has gained a lot of attention, comes from cells being handled in distinct groups. This is commonly referred to as *batch noise* or *batch effect*. These batch effects can arise in many different situations, from the small-scale effects occurring when cells are sequenced in different lanes or harvested at different time points to larger scale effects when cells are grown in different laboratories or stemming from different species. If these batch effects are not properly corrected for variance in gene expression caused by technical noise can be attributed as an effect caused by biological mechanisms which can then lead to serious misinterpretation of the data.

A famous example of this is was the comparison of the gene expression between human and mouse tissue by Lin et al., 2014 [13]. From their analysis of the cells

they came to the conclusion that cells would appear more similar based on species rather than cell type - i.e. human brain cells were found to be closer to human heart cells than they were to brain cells from a mouse. Given the fact that we for decades have studied the effectiveness of medical treatments on mice under the assumption that human and mice cells are comparable at some level, such a revelation would be revolutionary. This was, however, shown by Gilad & Mizrahi-Man, 2015 [4] to be caused by batch noise not accounted for during data analysis. When they reexamined the data, now correcting for batch noise, they found that cells would group together with cells of the same cell type not dependent on species.

This leads us to the the main question of this thesis; can a relatively small batch noise lead to problems in clustering of cell types and the subsequent analysis of new cell types? - i.e. how will a small batch noise affect the similarity between cells of different types and batches, and will this cause the most commonly used clustering algorithms to cluster cells based on batch instead of cell type - even if the distance between cells of the same cell type is smaller than cells from the same batch?

Our aim is to explore these questions through a series of simulations. In these simulations we will try to investigate the influence of batch noise represented in a simple and intuitive way.

In the second part of this thesis we consider another but related problem. A popular use of scRNA-seq data is the inferring of gene/gene regulatory networks. This relies heavily of the Pearson correlation coefficient but there may be nonlinear or non-monotonic correlation patterns in scRNA-seq data. In this regard we want to see what correlation patterns exists in real scRNA-seq data.

Inspired by the success of identifying different cell types using unsupervised clustering methods on scRNA-seq data we decided to apply these techniques in the investigation of gene/gene correlation - i.e. can we use clustering methods used in the scRNA-seq workflow to identify different types of gene/gene correlation? And can this information help us identify genes that plays an important role in temporal decision making?

## Chapter Overview

**Chapter 2** contains a brief overview of the current best practice for the single cell RNA sequencing workflow from the isolation of individual cells to the different types of down stream analysis.

**Chapter 3** describes the different predefined methods used in the subsequent chapters. This includes methods for batch correction, dimension reduction, clustering, and similarity scores between two partitions.

**Chapter 4** introduces a method for generating synthetic data (RGB cells) and other methods used to examine the main question put forth by this thesis - i.e. the effect of a small batch noise on clustering and identification of cell types. Moreover, this chapter also contains the resulting analysis on the effect of batch noise on the clustering and identification of cell types when examined using the before mentioned methods. This examination is performed on both simulated and real data.

**Chapter 5** contains the approach followed and the subsequent analysis of the second question raised by this thesis - i.e. what types of correlations exists in scRNA-seq data.

Chapter 2

# Background

In this chapter we will provide a quick overview of the technical and bioinformatical best practice workflow of scRNA-seq analyses. A more in-depth look into the methods used as a part this thesis is presented in Chapter 3.

The scRNA-seq workflow can broadly be split into three parts (1) the isolation of single cells & RNA counting, (2) The preprocessing & cleaning of the data, and (3) the downstream analysis of the data [15, 9, 6].

## 2.1 Single Cell Isolation & RNA Counting

The very first step is as the name suggest the isolation of the individual cells in a sample. This can be done using many different methods and is largely dependent on the type of experiment preformed [9, 15]. These methods range from extracting single cells from a suspension using pipettes to highly automated methods that very effectively can isolate single cells inside tiny droplets [9]. The next step is to extract and count the RNA present in each cell. This is done by preparing an RNA-seq library. Again, this can be done with a variety of different techniques but they mostly follow a set of common steps. Firstly the cell needs lysis in order to access the RNA. This RNA is then reverse transcribed into cDNA (complementary DNA), which is amplified and sequenced. The goal of sequencing is to get the order of the nucleotide from the pieces of cDNA obtained during amplification. The result of which is called a read. These reads are then subjected to a quality control (QC) where the bad reads are removed, after which the remaining reads are aligned to the desired genome. Alignment to a genome is when the order of nucleotides given by the read is matched to a specific location inside the genome. From this, the number of reads aligned to each gene in the genome can then be counted. This information is stored in the count matrix ($N(cellular\ ID) \times M(genes)$), where each count represents the successful capture, reverse transcription, and sequencing of an mRNA molecule in the given cell [15, 9, 6].

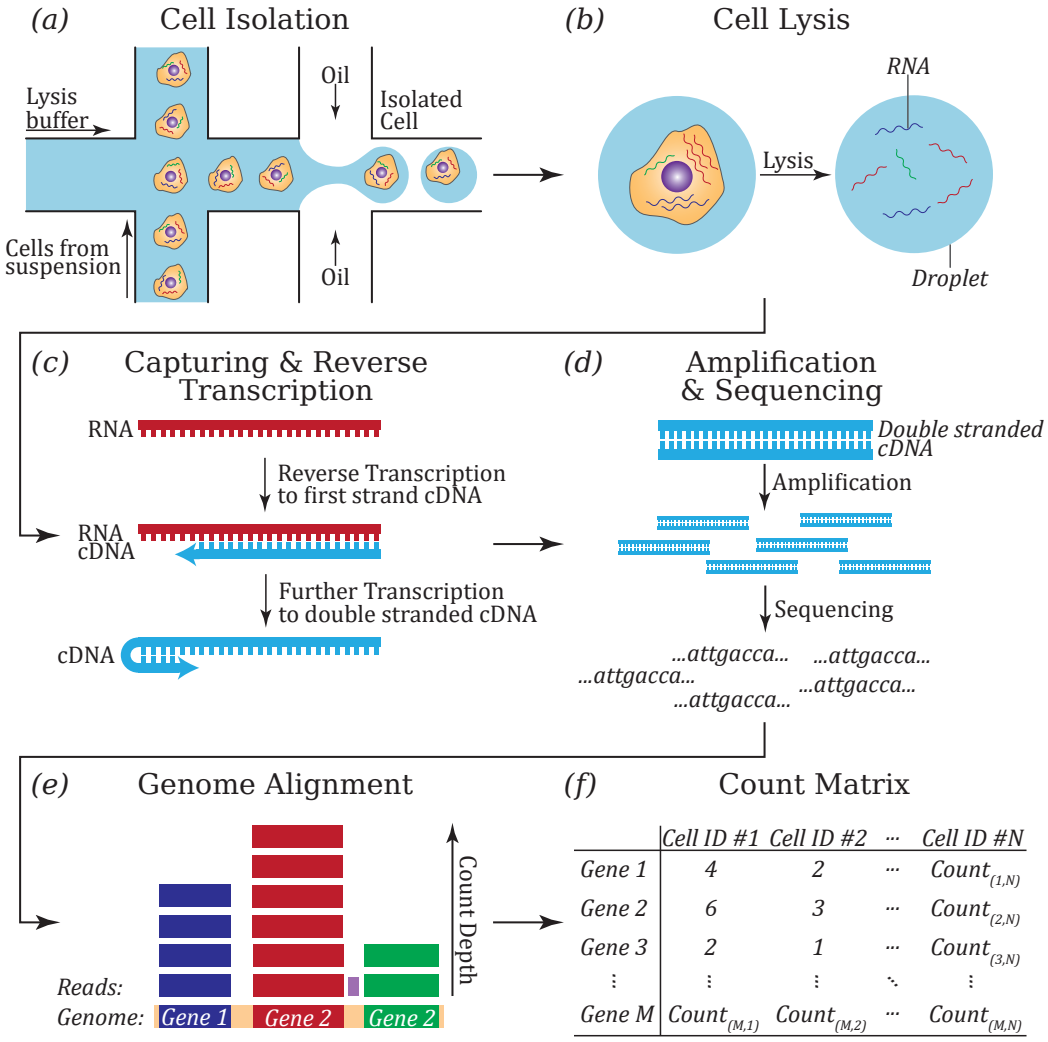


**Figure 2.1 | Illustration of single cell isolation and mRNA counting:** This figure illustrates the steps from the isolation of cells to the counting of aligned reads. *(a)* The cells are first isolated, here illustrated by the microfluidic method where each cell is isolated in tiny droplets. *(b)* The cell then lyses thereby freeing the cellular RNA. *(c)* The RNA is then captured and reversed transcribed to cDNA. *(d)* The cDNA is amplified and sequenced into reads which *(e)* is then aligned to a genome. *(f)* The number of reads aligned to gene is counted and stored in the count matrix. Reads that do not align to a gene in the genome, like the purple read in *(e)*, are not counted in the count matrix.

## 2.2 Preprocessing & Data Cleaning

Preprocessing is the next step in the scRNA-seq pipeline. This is an extremely vital step and has a big impact on the quality of the downstream analysis.

The step begins by performing a second round of QC. This time it is a QC of the cellular ID in the in the count matrix and is most commonly implemented based on three aspects: (1) the number of counts per cell (count depth), (2) the number of genes per cell, and (3) the fraction of counts from mitochondrial genes [15]. These parameters can be used to identify dying cells, cells with a broken membrane, or situations in which the isolation of a cell has failed and more than one cell was

isolated together. For example, the mRNA in the cytoplasm will leak out of a cell with a broken membrane. This will result in a low count depth, few observed genes, and a high fraction of mitochondrial genes. It is important to consider these aspects jointly when performing this QC since each parameter in isolation can have a perfectly valid biological explanation. This is the case when a very large cell will have a larger number of counts than a small cell would have. It is therefore recommended to be as allowing as possible when performing QC in order to not filter out any viable cells.

### 2.2.1 Normalisation

The next step in preprocessing is the normalisation of the count matrix. As stated previously, each count in the count matrix represents the successful capture, reverse transcription, and sequencing of an mRNA molecule in the given cell. The number of counts per gene (count depth) is the measure of how expressed the gene is. However, the count depth for otherwise two identical cells can vary due to sampling effects inherent to the methods used to capture and count the mRNA molecules. In the downstream analysis cells will be compared based on the count data. Therefore, it is vital that the relative gene expression between cells is correct. This is the problem that the normalisation step is meant to resolve.

A very popular normalisation method is the *counts per million* ($CPM$) protocol which is calculated by

$$CPM_{i,j} = 10^6 \frac{c_{i,j}}{\sum_j c_{i,j}}, \tag{2.1}$$

where $CPM_{i,j}$ is the normalised count and $c_{i,j}$ the un-normalised count for the $i'$th cell and the $j'$th gene, respectively. The factor of $10^6$ is called the scale factor and is what contributes the $M$ in $CPM$. The key assumption behind the $CPM$-method is that all cells in the data set initially have the same number of mRNA molecules and the variance in count depth is only due to sampling effects. It is a very simple and easy to interpret method for normalisation and will in many cases perform adequately [9, 6]. When choosing a normalisation method it is important to consider the experiment from which the data is coming. Some scRNA-seq data sets consist of a relative heterogeneous cell population with differing cell size and mRNA molecule counts. The $CPM$-method is in these situations not sufficient and more complex normalisation methods are needed [15].

After normalisation is is often recommended to do a $log(x+1)$-transform of the data. To a large extend, this is performed because the gene expressions can crudely be said to follow a log-normal distribution. The $log(x+1)$-transform will in this case make the gene expressions follow a normally distributed, which is an assumption for many of the downstream analysis methods [15].

### 2.2.2 Data Integration & Batch Correction

Normalisation can filter out the technical effect happening due to the variations that occurs during sampling. However, this is not the only effect one can wish to remove. For some experiments it can be desirable to even remove some biological effects in order to amplify the signal of the biological phenomena of interest. Although, in

most cases it is desirable to have the additional technical effects that have not been mitigated by the normalisation removed. [9, 6, 15].

These additional technical effects are commonly referred to as *batch effects*. There can be many different reasons for the batch effects, ranging from cells harvested at different times to cells grown in different laboratories.

The data can experience the batch effect on many different levels both between groups of cells in the same experiment, between experiments performed in the same lab, or between data sets from different laboratories. Here, it is common to distinguish between the first and the last two examples. This is due to the fact that the batch effect arising in these scenarios requires different levels of corrections. When the data is gathered from a single experiment the removal of batch effect is commonly known as *batch correction*. On the other hand, when the data comprised of data from different experiments the problem of correcting for batch effect is called *data integration* [15].

A more detailed explanation of the best performing methods for batch correction and data integration can be found in section 3.1.

### 2.2.3 Feature Selection, Dimension Reduction & Visualisation

Single cell RNA sequencing data sets can contain expression values for up to 25,000 genes a large part of which will hold no information about the underling biological variance. These are genes where the expression varies very little between the different cells in the data set - i.e. genes that are expressed equally across all of the cells. In order to ease the computational tasks in downstream analysis and reduce the noise generated from these low variance genes it is useful to reduce the dimensionality of the data set. This can be done using a wide range of methods and is performed in a series of steps. The first step is commonly known as *feature selection*. This is where the aforementioned low variance genes are filtered out leaving only the high variance genes (HVG). A very popular method for selecting HVGs is binning genes based on their mean expression and then selecting the one with the highest variance-to-mean ratio in each bin [15].

After the selection of the HVGs one can choose to further reduce the dimensions. This is now done with dedicated dimension reduction algorithms (section 3.2). These algorithms embed the data from the expression matrix into a lower dimensional space, which is designed to capture the underlying structure of the data in as few dimensions as possible [15].

Dimension reduction is largely performed for two reasons, namely *visualisation* and *summarisation*. The goal of visualisation is to capture the information in the data set in two or three dimensions. These reduced dimensions can be used to construct a visual representation of the data using a two or three dimensional scatter plot, respectively. The goal of summarisation on the other hand is to reduce the data to its essential components by finding the inherent dimensionality of the data. This will greatly ease the computational task in the downstream analysis. The reason being that the manifold of the biological effects can be sufficiently described using fewer dimensions than the number of genes. The number of output components from these methods are therefore not prescribed. However, the variance in the data described by each component becomes smaller and smaller for higher components thereby making higher components less and less important [15].

There are dedicated methods for both dimensional reduction in visualisation and summarisation. While the leading components generated by summarisation method can be used to visualise the data, a dedicated method for visualisation will provide a better representation of the inherent variability in the data set. On the other hand, the two or three dimensional output of the visualisation method should never be used for summarisation. One should be cautious when using the reduced dimensions for downstream analysis because vital biological information can be lost [9, 15].

A more in-dept explanation of the dimension reduction method used for visualisation in this thesis can be found in section 3.2.

We have here given a brief overview of the preprocessing of the data generated by the successful capturing, reversed transcription and sequencing of cellular mRNA. In summary the preprocessing can be divided into five stages: (1) QC of the cells in the data set, (2) normalisation of data, (3) correction of technical and undesirable biological effects, (4) feature selection of high variance genes, and (5) dimension reduction for visualisation and summarisation.

## 2.3 Downstream Cell- & Gene-Level Analyses

After the raw count data has been preprocessed different types of downstream analyses can be performed in order to extract information about the underlying biological system. These methods are usually split into two groups: gene- and cell-level approaches. The first of which works with inferring regulatory networks by detecting genes with correlated expressing profiles. Analyses made on a cellular level will focus on either clustering cells with similar gene expressions into groups of the same cell type or measure small changes in gene expressions between cells in order to infer trajectories in cellular development.

We have in this thesis primarily worked with the clustering and community detection algorithm used in the identification of cell types. We will therefore primarily focus on this part of the downstream analysis.

### 2.3.1 Cell Type Identification

There exists many different methods for clustering cells with similar gene expressions together into smaller groups. These methods can generally be divided into the two before mentioned groups : clustering and community detection algorithms.

Clustering algorithms are classical unsupervised machine learning problems where cells are assigned to clusters by minimising the intra-cluster distance. Community detection algorithms are based on graph-partitioning, where cells are represented as notes and each cell has an etch connecting it to its $K$ nearest neighbours. A common aspect to both clustering and community detection algorithms is that they rely on a parameter - which is chosen by the user - that determines the number of clusters. It is therefore important to know at what level of granularity one wishes to examine the data before implementing the algorithm of choice.

The graph based methods have in general been shown to perform better than the clustering algorithms based only on distance. A more detailed explanation of

the Leiden community detection algorithm used throughout this thesis can be found in section 3.3.1.

The next step in the identification of cell types is the labelling of each cluster with a meaningful biological annotation. This is done by matching the gene's signature for each cluster to so called *marker genes*. This annotation of biological identity is vital for determining the quality of the partition generated by the algorithm of choice. The *adjusted Rand index* ($ARI$) is a measurement of the quality of such partition and is explained further in section 3.3.2.

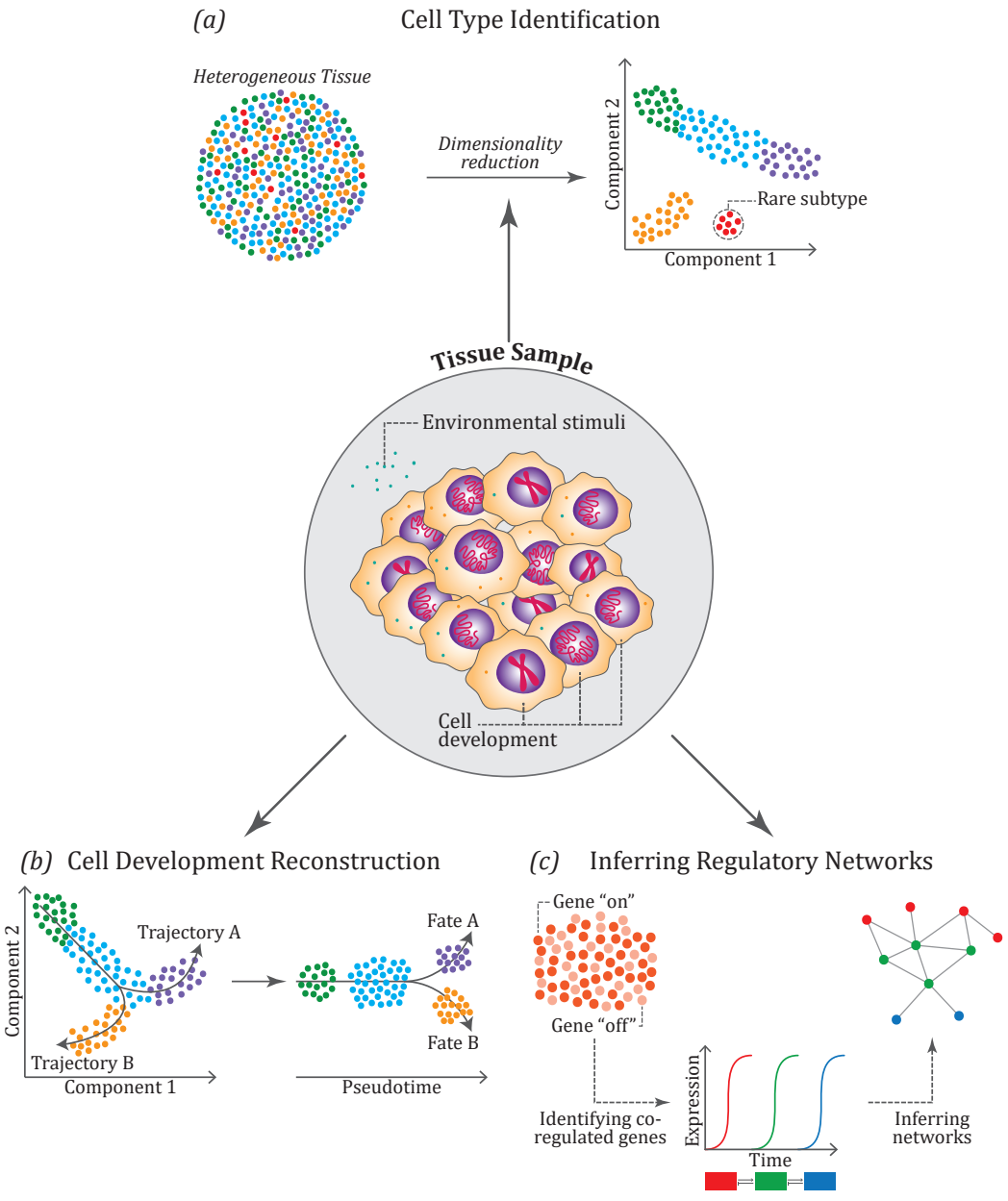


**Figure 2.2 | Illustration of the different downstream analyses for scRNA-seq data:** This figure illustrates the most common downstream analyses for scRNA-seq data. *(a)* illustrates how through dimension reduction and clustering algorithms we can identify different cell types and discover new rare subpopulations. *(b)* shows another popular cell-level analysis. The reconstruction of cell development. Again, through dimension reduction and clustering algorithms we can identify different development trajectories and from this we can establish a development path. *(c)* shows the gene-level analysis where co-regulating genes are identified to infer regulatory networks.

Chapter 3

# Methods

During the years of research into the application of single cell sequencing a number of methods have been proposed for the different aspects of the scRNA-seq pipeline as described in Chapter 2. In this chapter we will give a brief explanation of the different methods used for *batch correction*, *clustering*, *visualisation/dimensional reduction*, and *partition comparison*.

## 3.1 Methods for Batch Correction & Data Integration

The list of methods used for batch correction and data integration is quite long. An extensive benchmark analysis of a vast number of these methods was performed by Luecken et al., 2020 [14], where each method was scored based on the level of batch correction and biological conservation - i.e. how much biological variance was conserved. Based on this list we chose three different methods; *ComBat*[10], *Harmony*[11], and *Scanorama*[7]. All of these methods scored high on both batch correction and biological conservation but are different in both the approach and when each is suitable to use [14]. *ComBat* and *Harmony* are both linear methods and they performed best when applied to less complicated batch correction problems. *Scanorama* on the other hand is a nonlinear graph method that performs the best when tackling more complicated data integration problems but can sometimes over correct in less complicated scenarios [14].

### 3.1.1 ComBat

The *ComBat* method was originally developed for bulk gene expression experiments but is still commonly used for scRNA-seq data as well [14]. *ComBat* belongs to a family of batch adjustment methods called location and scale (L/S) adjustments. The idea behind these adjustment methods is that the batch effect can be modelled out by standardising the means and variances across batches. This is done by assuming a model that will represent the gene expression. The *ComBat* method uses a model for the observed gene expression $Y_{ijg}$ for gene $g$ from sample $j$ in batch $i$ expressed by,

$$Y_{ijg} = \alpha_g + X\beta_g + \gamma_{ig} + \delta_{ig}\epsilon_{ijg}, \tag{3.1}$$

where $\alpha_g$ is the overall gene expression, $X$ is a design matrix that contains information about sample conditions, $\beta_g$ is the gene specific vector of regression coefficients corresponding to $X$. $\gamma_{ig}$ and $\delta_{ig}$ represents the additive and multiplicative batch

effects for batch $i$ and gene $g$, respectively. The $\epsilon_{ijg}$ factor represents the measurement error and is assumed to be normally distributed with an expected value of zero [10].

*ComBat* uses the *empirical Bayes* method for determining the estimators from the linear regression in eq. (3.1) [10]. This is a method where the prior distribution is determined from the data contrary to the standard Bayesian method where the prior is fixed beforehand. After the estimators are obtained the observed data $Y_{ijg}$ can be adjusted for batch effects to the corrected gene expression

$$Y^*_{ijg} = \frac{Y_{ijg} - \hat{\alpha}_g - X\hat{\beta}_g - \hat{\gamma}_{ig}}{\delta_{ig}} + \hat{\alpha}_g + X\hat{\beta}_g, \tag{3.2}$$

where $\hat{x}$ denotes the estimation of the $x$'th parameter [10].

### 3.1.2 Harmony

*Harmony* is a method developed by [11] that tackles the batch effect by projecting cells into an embedded space where cells will group based on cell type. This method starts by embedding the cells in a lower dimensional space before the cells are grouped into clusters containing data from multiple data set (Figure 3.1a). To perform this clustering [11] coined a soft version $K$-means clustering algorithm where each cell could potentially be assigned to multiple clusters. The idea behind this was to capture the smooth transition between cell stages. Moreover, the clustering algorithm also penalised clusters with a low diversity of cells from different experiments.

After clustering *Harmony* will calculate a centroid (the centre point of a cluster) for each cluster as well as a data specific centroid - i.e. a centroid for each cluster where only cells from the same data set are uses (Figure 3.1b). The distance between these centroids are then used to compute a cluster and a data specific correction factor (Figure 3.1c). Finally, each cell is assigned an average of these correction factors proportional to its level of soft assignment to each of the corresponding clusters. This will potentially provide a unique correction factor for each cell. The cells are then corrected (Figure 3.1d) and the four steps are repeated until the cell cluster assignments are stable [11].

### 3.1.3 Scanorama

*Scanorama* is a method developed by [7] for integrating scRNA-seq data from multiple experiments. The main idea behind the *Scanorama* method is analogous to the algorithms used in panorama stitching where overlapping features of a set of images can be stitched together into a single image (Figure 3.2a). *Scanorama* can in the same manner stitch multiple scRNA data sets together by identifying overlapping regions. This is achieved by a generalisation of the mutual nearest neighbours (MNN) approach introduced by [5]. Whereas the original MNN method would find similar elements between only two batches the *Scanorama* method can perform this on multiple data sets at the same time. However, to integrate more than two data sets in the original MNN method one had to select a reference data frame where each other data set are successively integrated into the reference data set one by one (Figure 3.2e). This made these methods vulnerable to over-correction and dependent on the order in which data sets are integrated. The *Scanorama* method

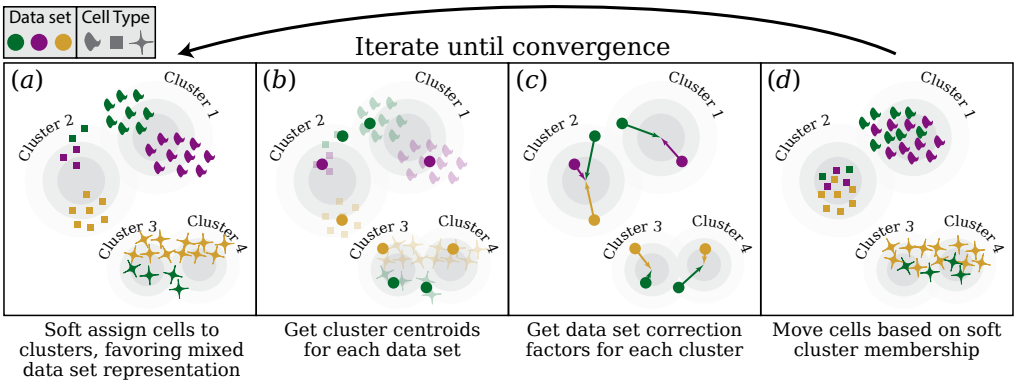


**Figure 3.1 | Overview of the *Harmony* algorithm:** Data sets is in this figure represented using colours and different cell types using shapes. *(a)* The *Harmony* algorithm uses soft clustering to assign the different cells to potentially multiple clusters. A penalty term ensures that the diversity of data sets within each cluster is maximised. *(b)* The *Harmony* algorithm then calculates global centroids for each cluster, as well as centroids specific to each data set for each cluster. *(c) Harmony* will now calculate a correction factor for each combination of clusters and data sets based on the centroids. *(d)* Finally, *Harmony* corrects each cell with a cell specific factor gained from a linear combination of the correction factors from step *(d)* weighted by its soft cluster assignments from step *(a)*. Steps *(a)* through *(d)* is repeated until convergence. - Image taken from [11].

can to an extent alleviate these problems by considering matches between all pairs of data sets [7].

### Mutual Nearest Neighbours (MNN)

The mutual nearest neighbour method for batch correction is an approach developed by [5]. This method identifies cells based on similar gene expression profiles between data sets from different experiments. This is achieved by firstly performing a nearest neighbour search with the cosine distance (eq. (3.3)) as a metric of similarity between two batches [5],

$$cosine.distance(\vec{x},\vec{y}) = 1 - \frac{\vec{x}\cdot\vec{y}}{\|\vec{x}\|\ \|\vec{y}\|} = 1 - \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}}. \tag{3.3}$$

Let us consider two scRNA-seq experiments where each consists of a data set containing a single batch - batch 1 and batch 2. For each cell in batch 1 the aforementioned nearest neighbour search is performed in order to find the $k$ cells in batch 2 with the smallest cosine distance to the cells in batch 1. This constitutes the $k$ nearest neighbours in batch 2. The same thing is performed for all cells in batch 2. In this manner we have a list of the $k$ nearest neighbours for all cells in both batch 1 and batch 2. A pair of cells from each batch will be considered to be mutual nearest neighbours if they are contained in each others set of nearest neighbours (Figure 3.2b - 3.2f). These pairs of cells are interpreted as cells belonging to the same cell type and the difference between them can be used to correct the batch effect between the two batches [5].

One way *Scanorama* differs from MNN is instead of performing the nearest neighbour search in high-dimensional space it first performs a dimension reduction routine

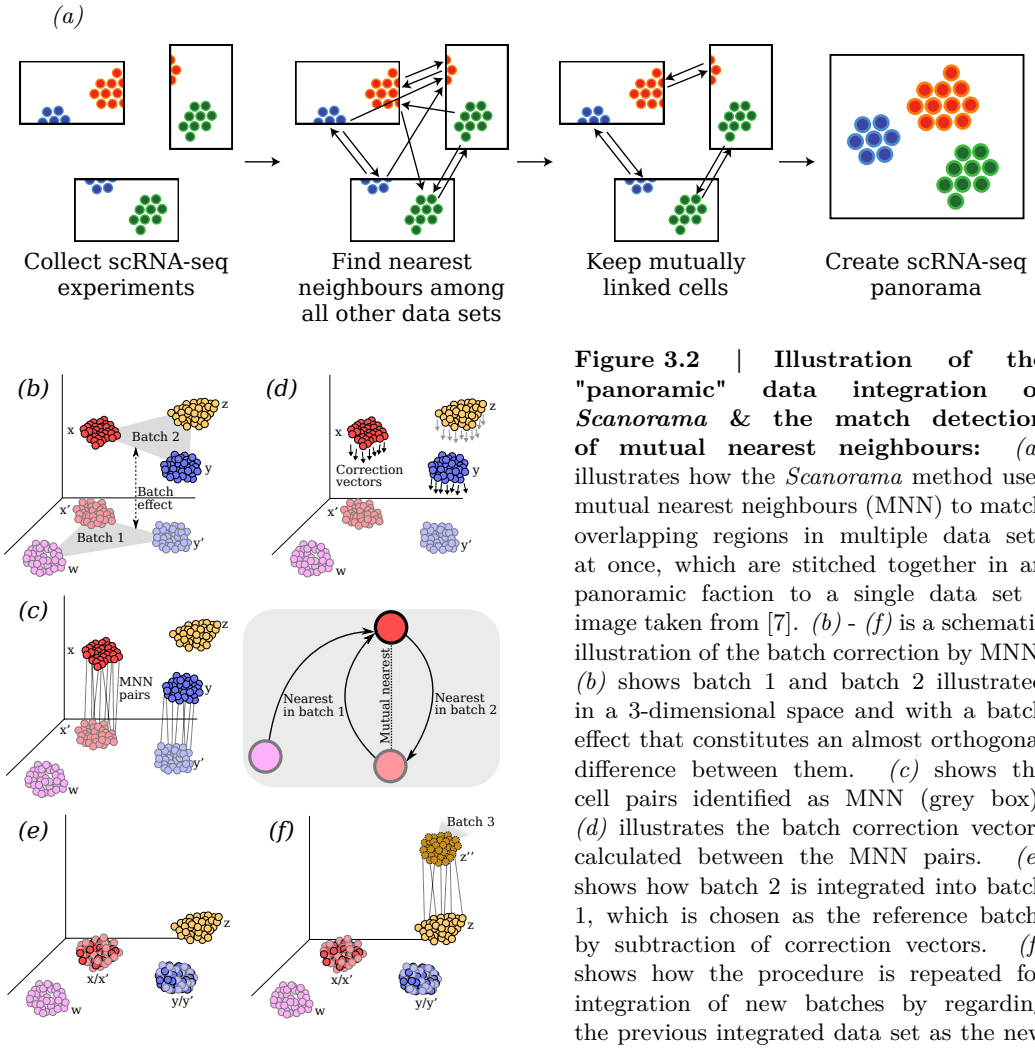


**Figure 3.2 | Illustration of the "panoramic" data integration of *Scanorama* & the match detection of mutual nearest neighbours:** *(a)* illustrates how the *Scanorama* method uses mutual nearest neighbours (MNN) to match overlapping regions in multiple data sets at once, which are stitched together in an panoramic faction to a single data set - image taken from [7]. *(b)* - *(f)* is a schematic illustration of the batch correction by MNN. *(b)* shows batch 1 and batch 2 illustrated in a 3-dimensional space and with a batch effect that constitutes an almost orthogonal difference between them. *(c)* shows the cell pairs identified as MNN (grey box). *(d)* illustrates the batch correction vectors calculated between the MNN pairs. *(e)* shows how batch 2 is integrated into batch 1, which is chosen as the reference batch, by subtraction of correction vectors. *(f)* shows how the procedure is repeated for integration of new batches by regarding the previous integrated data set as the new reference - image taken from [5].

where the gene expression profile of each cell is embedded in a lower dimensional space. This greatly decreases the computational burden such a search would otherwise have entailed. Moreover, the *Scanorama* method can by considering all data sets at the same time integrate these data sets without merging batches that has no overlapping regions [7].

## 3.2 Methods for Dimensions Reduction & Visualisation

As mentioned in Chapter 2 scRNA-sec data sets may consist of up to tens of thousands different genes. Such high dimensionality can prove to be difficult for some downstream analysis methods to handle and makes visualisation unfeasible. For these purposes different types of dimension reduction methods exists and differ based on whether the purpose is downstream analysis or visualisation [15]. The methods used for downstream analysis tries to capture as much of the biological variance in as few components as possible. *Scanorama* and *Harmony* are examples of downstream analysis methods that rely on these types of dimension reductions

[7, 11]. The first couple of components generated by such dimension reduction algorithms can be used for visualisation but there exists methods which embed data in a lower dimensional space for the explicit purpose of visualisation. We did not perform dimensional reduction for our main analysis. It was only used as a part of the *Scanorama* and *Harmony* methods. We will therefore only focus on the *t*-SNE visualisation methods which was heavily used in this thesis.

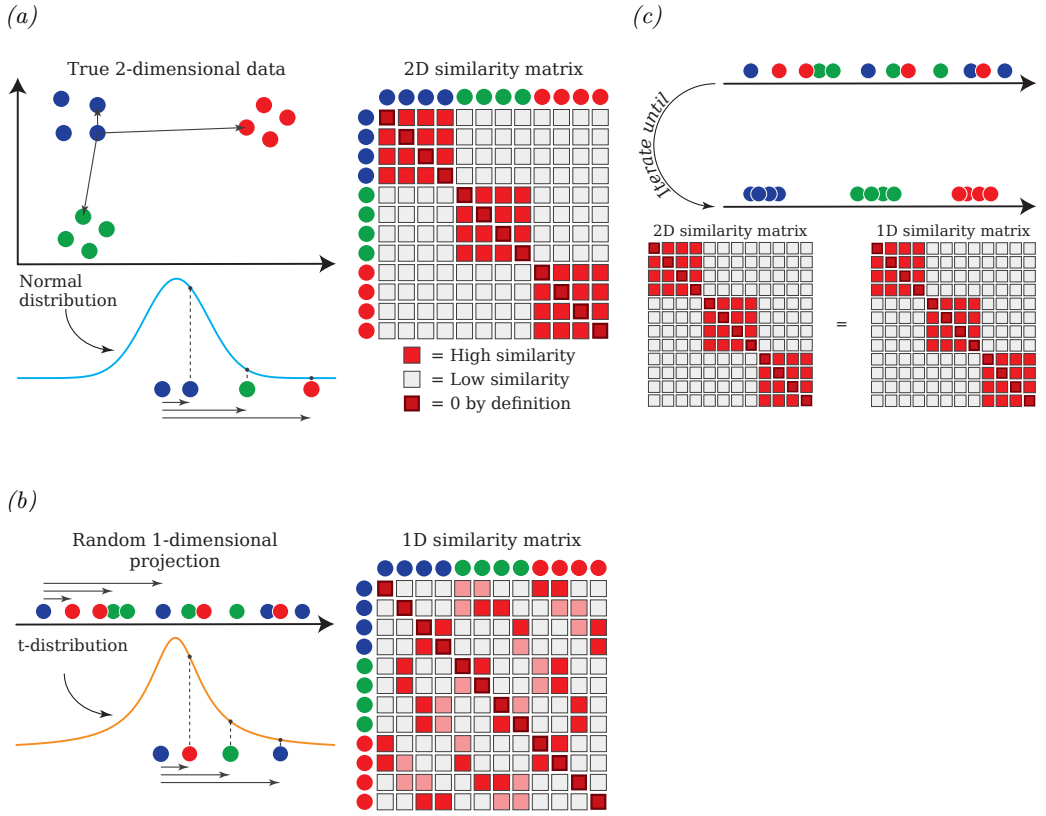


**Figure 3.3 | Illustration of the reduction from two to one dimensions using *t*-SNE:** *(a)* shows (left) the true 2D representation of the data. Bottom left shows the use of the Gaussian distribution in the determination of the similarity score. To the right is an illustration of the similarity matrix. *(b)* shows the initial random projection of the data into 1D. Below which the use of the *t*-distribution in the determination of similarity scores. (right) Similarity matrix for the low-dimensional case. *(c)* illustrates how the *t*-SNE algorithm updates the projection of the lower dimensional space until the similarity scores are the same for both high- and low-dimensional data.

### 3.2.1 *t*-Distributed Stochastic Neighbour Embedding

*t*-Distributed Stochastic Neighbour Embedding (*t*-SNE) is a method used to map a high-dimensional data set down to two or three dimensions [21]. This is done by firstly turning the Euclidean distance between two points in the high-dimensional data set ($x_i$ and $x_j$) into a conditional probability $p_{j|i}$ that represents the similarity between the two points. This similarity score $p_{j|i}$ is the probability that the point $x_i$ will choose $x_j$ as a neighbour given that neighbours are picked proportional to the probability density function (PDF) of a Gaussian distribution centred at $x_i$ (Figure

3.3a). This similarity score $p_{j|i}$ is given by

$$p_{i|j} = \frac{exp\left[-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right]}{\sum_{k \neq i} exp\left[-\frac{\|x_k - x_j\|^2}{2\sigma_i^2}\right]}, \tag{3.4}$$

where $\sigma_i$ is the variance of the Gaussian distribution which is determined from the perplexity. This parameter is determined by the user and is a soft measure for the number of neighbours [21].

The similarity score determined this way is not symmetric - i.e. $p_{j|i}$ is often not equal to $p_{i|j}$. To make it symmetric we can implement a parameter $p_{ij} = p_{ji}$ that is the average of $p_{j|i}$ and $p_{i|j}$.

The high-dimensional data is now randomly projected into the desired low-dimensional space. The similarity score for the projected points in the low-dimensional data set is then calculated. The main idea is the same, the Euclidean distance between two points ($y_i$ and $y_j$) is again converted into a conditional probability $q_{ji}$. The only difference is that the probability is now proportional to the PDF of a student $t$-distribution with one degree of freedom (Figure 3.3c) - hence the "$t$" in $t$-SNE [21]. This similarity score $q_{ji}$ is given by

$$q_{ij} = \frac{\left(1 + \|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l} \left(1 + \|y_k - y_l\|^2\right)^{-1}}. \tag{3.5}$$

The idea is now to incrementally change the mapping of points so that mismatch between $pji$ and $q_{ji}$ is minimised (Figure 3.3b). This is done by a gradient descent where each component is given by,

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ji} - q_{ji})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}. \tag{3.6}$$

These steps are repeated until the projection in the low-dimensional space is stable and the mismatch between $pji$ and $q_{ji}$ is minimised [21].

## 3.3 Methods for Cell Identification

The first step in identifying different cell populations is the grouping of cells based on the similarity of gene expression profiles. There are two approaches to generate these groupings. The first is the classical clustering algorithms known from unsupervised machine learning problems. This can for example be the $K$-means algorithm used in the Harmony method. The second approach uses community detection methods. These are graph based methods where the scRNA-seq data is represented on a graph where each cell is a note and the edges represent the similarity between the cells. Cells are then grouped based on the links between the cells in so-called communities - hence the name community detection algorithms. A community in a network is defined by a set of highly interconnected nodes which means there is a large amount of links inside the communities compared to links between them.

The graph based methods are in general faster than the clustering algorithm. Because of the sometimes very large data sets analysed in scRNA-seq it is therefore recommended to use the community detection methods [15]. The Leiden algorithm is such a method.

### 3.3.1 The Leiden Clustering Algorithm

The Leiden algorithm is an improvement to the Louvain community detection method [18]. The Louvain method is composed of two phases; (1) moving nodes between communities and (2) aggregating a new network based on the communities [2]. The Leiden algorithm is very similar but it includes a step where the partitions of the communities are refined before aggrated into a new network [18]. Both methods iterates over these steps until the desired resolution is achieved.

The first step in both the Louvain and Leiden algorithm is the moving of nodes between communities. In order to evaluate this step we need a measure for the density of a community. This is commonly measured by the so-called modularity,

$$Q = \frac{1}{2m} \sum_{i,j} \left[ A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \quad Q \in [-1, 1], \tag{3.7}$$

where $A_{ij}$ is the weight between the nodes $i$ and $j$, $k_i = \sum_j A_{ij}$ is the sum of the weights of all the edges attached to node $i$, $c_i$ is the community of node $i$, and $m = \frac{1}{2} \sum_{ij} A_{ij}$. The modularity score, $Q$, is weighted by a resolution parameter which determines how many communities the algorithm will partition the network into. A higher resolution will lead to more communities whereas a lower resolution will lead to fewer communities [2, 18].

Given a weighted network of $N$ nodes we can now start by assigning a different community to each node in the network - i.e. in the first pass there will be as many communities as there are nodes in the network (Figure 3.4a). Then for each node $i$ we consider its neighbours $j$ and evaluate the change in modularity that would arise from removing $i$ from its own community and placing it in the community of $j$. The $i$'th node will then be placed in the community where the change in modularity gives the highest possible value. If all the possible changes are negative the $i$'th node stays in its own community [2]. The change in modularity $\Delta Q$ coming from moving a node $i$ into community $C$ is computed by the following equation,

$$\Delta Q = \left[ \frac{\Sigma_{in} + k_{i,in}}{2m} - \left( \frac{\Sigma_{tot} + k_i}{2m} \right)^2 \right] - \left[ \frac{\Sigma_{in}}{2m} - \left( \frac{\Sigma_{tot}}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right], \tag{3.8}$$

where $\Sigma_{in}$ is the sum of weights insight the community $C$, $\Sigma_{tot}$ is the sum of weights of links that comes into nodes in $C$, and $k_{i,in}$ is the sum of weights of all the links between the $i$'th node and the nodes in $C$. This process is repeated for all nodes until no further improvements can be made (Figure 3.4b). This means that the same node can be assigned multiple times [2].

The next step is where the Leiden algorithm differ by introducing a refined partition $\mathcal{P}_{refined}$ of the network (Figure 3.4c). In order to construct this refined partition we start just as in phase one by assigning each cell its own community. Just as before we go through each cell evaluating the change in modularity by moving it through different communities. The difference between the moving phase

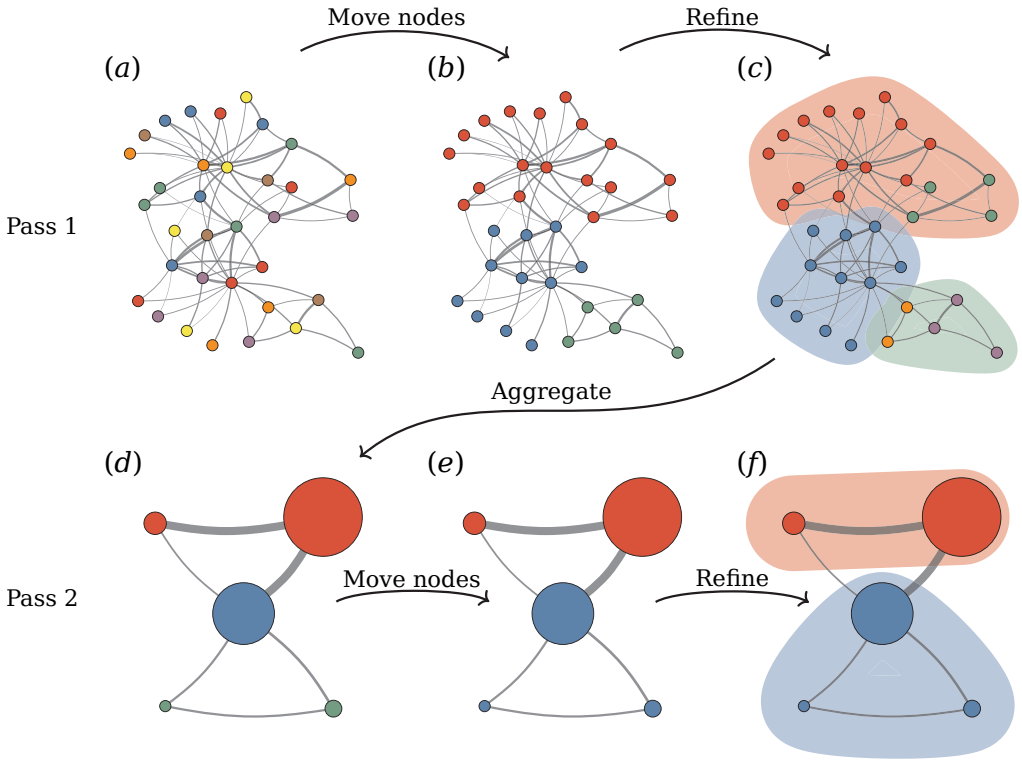


**Figure 3.4 | Illustration of the steps in the Leiden clustering algorithm:** *(a)* shows the initial assignment of cells into individual communities. *(b)* shows the final partition $\mathcal{P}$ after all the cells have been moved to the community that provided the highest modularity score. *(c)* illustrates the partitioning of the cells into the refined partition $\mathcal{P}_{refined}$ (colour of the nodes) compared to the original partition $\mathcal{P}$ (coloured regions). *(d)* shows the new aggregated network where the new nodes are based on $\mathcal{P}_{refined}$ and the initial community assignment is based on $\mathcal{P}$. *(e)* and *(f)* illustrates the moving of nodes and the generation of a refined partition for the new aggregated network, respectably. These steps are repeated until no more improvements can be made. - Image taken from [18].

is that the note is not assigned greedily to the community that yielded the largest increase in modularity but may be merged with any community where the change in modularity is positive. Communities in the original partition $\mathcal{P}$ may therefore be split into multiple subcommunities in $\mathcal{P}_{refined}$ [18].

In the third step a new network is aggregated where each new node corresponds to a community in $\mathcal{P}_{refined}$ but the initial partition of the new network is based on $\mathcal{P}$. This new network can then be reapplied at the first step by starting a new pass of the algorithm (Figure 3.4e and 3.4f). These steps are repeated until no further improvements can be made [2, 18].

### 3.3.2 Measure of Agreement Between Two Partitions

In order to determine the *goodness of clustering* we need to define a measure of agreement that can compare the resulting clustering against some external criteria. In our case each cell has one true cell type and our clustering algorithms will also only assign each cell to one cluster. We can therefore use the measure of agreement between two partitions as a *goodness of clustering* measure [24]. The adjusted Rand index ($ARI$) is such a measurement.

### Rand Index & Adjusted Rand Index

Suppose that we have $n$ cells that are divided into $m$ different cell types which has been clustered into $k$ clusters by a clustering algorithm. We can then define two partitions; $U = \{u_1, \cdots, u_m\}$ and $C = \{c_1. \cdots, c_k\}$ where $U_i$ determines the number of cells of the $i$th cell type and $c_i$ the number of cells in the $i$th cluster. We can then define four parameters:

***a:*** Is the number of cells that belongs to the same cell type in $U$ and the same cluster in $C$.
***b:*** Is the number of cells that belongs to the same cell type in $U$ but in a different cluster in $C$.
***c:*** Is the number of cells that belongs to a different cell type in $U$ but the same cluster in $C$.
***d:*** Is the number of cells that belongs to a different cell type in $U$ and a different cluster in $C$.

Here $a$ and $d$ can be interpreted as agreement and $c$ and $b$ as disagreement. From these parameters we can define the *Rand index* as

$$Rand\ index = \frac{a+d}{a+b+c+d}. \tag{3.9}$$

This is a simple measure of the agreement between two partitions. The Rand index lies between 0 and 1 where 1 corresponds to perfect agreement and 0 to no agreement [24].

A problem with the Rand index is that it does not correct for randomness in the clustering. The adjusted Rand index $ARI$ is a modified version of the Rand index that does just that [14]. The general form of the $ARI$ is

$$\frac{index - expected\ index}{maximum\ index - expected\ index}. \tag{3.10}$$

Just as in the case for the Rand index an $ARI$ of 1 will correspond to perfect agreement between the two partitions. An $ARI$ of 0 on the other hand corresponds to an index equal to the expected index - i.e. random clustering. This was the lower boundary for the Rand index which will therefore make the adjusted Rand index a more sensitive measure of agreement [14, 24]. It was shown by Hubert & Arabie, 1985 ([8]) that the expected index can be written as,

$$E\left[\sum_{i,j}\binom{n_{ij}}{2}\right] = \left[\sum_i \binom{n_{i.}}{2}\sum_i \binom{n_{.j}}{2}\right] / \binom{n}{2}, \tag{3.11}$$

where $n_{ij}$ is the number of cells of type $i$ in cluster $j$, $n_{i.}$ the number of cells of type $i$, $n_{j.}$ the number of cells in cluster $j$, and $n$ the total number of cells[1]. This makes the adjusted Rand index,

$$\frac{\sum_{i,j}\binom{n_{i,j}}{2} - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}. \tag{3.12}$$

In our simulation we used `scikit-learn`'s implementation of the $ARI$ function.

[1] An explanation of the notation used can be found in Table 7.2 in Appendix

Chapter 4

# Simulating Batch Effects using Synthetic RGB-Cells

The aim of this master thesis is to examine the influence a small batch noise can have on the subsequent clustering and identification of different cell types. Our concern is that when we combine multiple sample collections the otherwise relative small difference between batches (compared to the difference between cell types) will cause the clustering algorithms to cluster the cells based on the batch rather than cell type. We hypothesise that when comparing the euclidean distance between cells of different cell types undesirable clustering will occur even when cells of the same cell types are closer to each other than cells of the same batch. Here we define undesirable clustering as when the clustering algorithm groups cells together based on batch rather than cell type.

To explore these phenomena we have decided to investigate them through a theoretical simulation. By choosing a simulation we can focus our investigation on only a few aspects at the time. This will make it easier to get a simple and intuitive understanding of what influence each of the desirable aspects will have on a given downstream analysis. This would not be possible by analysing real data only because the influence of each aspect would be very difficult to isolate. For this purpose we have coined the concept of an RGB-cell that can be used to simulate a simplified version of scRNA-seq data sets. The goal of this simulation is thereby to construct a simple method that can provide an intuition for the influence of the batch noise.

## 4.1 Synthetic RGB-Cells for Simulation

The inspiration for the RGB-cells was drawn from the RGB colour model. In the same manner as red, green, and blue light is added together to create a specific colour in the colour model we can combine the different expressions of "red", "green", and "blue" genes together to create an RGB-cell of a given colour/cell type. In this manner we can create as many different cell types as there are colours in the RGB colour model.

The goal of these RGB-cells is to be able to generate batches composed of populations of cells with different colours. These colours will then represent the different cell types inside these batches.

### 4.1.1 Constructing an RGB-Cell

In our simulation a cell is represented as a vector $\vec{c}$ that contains the gene expression of $N$ genes,

$$\vec{c} = \{g_1,\ g_2,\ g_3,\ \cdots,\ g_N\}, \tag{4.1}$$

where $g_i$ is the gene expression of the $i$th gene. These $N$ genes are then divided into three subgroups representing the three colours red, green, blue,

$$\vec{c} = \{\overbrace{R_1,\ R_2,\ \cdots,\ R_{N_R}}^{Red\ genes},\ \overbrace{G_1,\ G_2,\ \cdots,\ G_{N_G}}^{Green\ genes},\ \overbrace{B_1,\ B_2,\ \cdots,\ B_{N_B}}^{Blue\ genes}\}. \tag{4.2}$$

The combined expression of these genes will then determine the colour/cell type of the cell.

Genes within the same colour group will have a high positive correlation whereas the correlation between cells of different subgroups will depend on the final colour of the cell. These different correlations are illustrated in the top of Figure 4.1a.
To generate a cell of a specific cell type we define a set of vectors that represents the base colours; red, green, and blue,

$$\vec{e}_r = \{\overbrace{1,\ 1,\ \cdots,\ 1}^{Red\ genes},\ \overbrace{0,\ 0,\ \cdots,\ 0}^{Green\ genes},\ \overbrace{0,\ 0,\ \cdots,\ 0}^{Blue\ genes}\} \tag{4.3}$$

$$\vec{e}_g = \{\overbrace{0,\ 0,\ \cdots,\ 0}^{Red\ genes},\ \overbrace{1,\ 1,\ \cdots,\ 1}^{Green\ genes},\ \overbrace{0,\ 0,\ \cdots,\ 0}^{Blue\ genes}\} \tag{4.4}$$

$$\vec{e}_b = \{\overbrace{0,\ 0,\ \cdots,\ 0}^{Red\ genes},\ \overbrace{0,\ 0,\ \cdots,\ 0}^{Green\ genes},\ \overbrace{1,\ 1,\ \cdots,\ 1}^{Blue\ genes}\}. \tag{4.5}$$

These vectors span the entire colour space and from this every cell type can be generated by a weighted sum of the base vectors,

$$\vec{c} = w_r\vec{e}_r + w_g\vec{e}_g + w_b\vec{e}_b, \tag{4.6}$$

where $w_R$, $w_G$, and $w_B$ is the weight of the corresponding base vector. As an example, we can in this fashion generate a red cell, $\vec{c}_r$:

$$\begin{aligned}\vec{c}_r &= 255\vec{e}_r + 0\vec{e}_g + 0\vec{e}_b \\ &= \{\underbrace{255,\ 255,\ \cdots,\ 255}_{Red\ genes},\ \underbrace{0,\ 0,\ \cdots,\ 0}_{Green\ genes},\ \underbrace{0,\ 0,\ \cdots,\ 0}_{Blue\ genes}\}.\end{aligned}$$

since the $RGB$ colour code for red is $\{R = 255, G = 0, B = 0\}$.

### 4.1.2 Adding Cell/Cell Variation to RGB-Cells

#### Intrinsic Variation

The next step in our simulation is to construct a population of multiple cells of the same cell type. In order to do so we need to introduce some variation in the gene expressions for the different cells. Noise will of course exist in cells of the same cell type. This is largely due to the stochastic nature of transcription. This variation is added in two separate steps. The first step introduces variance into the

three different gene colour groups red, green, and blue. This is done by introducing randomness to the aforementioned weights.

In our simulation we have only used three different cell types - red, green, and blue - and the weights for each of these types were generated as follows,

$$\vec{W}_R = \{w_r \sim U(0,1),\ w_g \sim U(0,n_I),\ w_b \sim U(0,n_I)\}, \quad \text{weights for red cells}$$
$$\vec{W}_G = \{w_r \sim U(0,n_I),\ w_g \sim U(0,1),\ w_b \sim U(0,n_I)\}, \quad \text{weights for green cells}$$
$$\vec{W}_B = \{w_r \sim U(0,n_I),\ w_g \sim U(0,n_I),\ w_b \sim U(0,1)\}, \quad \text{weights for blue cells,}$$

where $x \sim U(a,b)$ is a number drawn from a uniform distribution between $a$ and $b$ and $n_I$ is the simulations *intrinsic noise* parameter. This parameter will thereby determine how "polluted" each cell type is - i.e. a high $n_I$ will make the population of cells express colours not associated with its cell type. The addition of this noise is illustrated by the difference between Figure 4.1a and 4.1b. This illustrates how before adding any noise (4.1a) we start with three groups of identical cells. After the introduction on noise weights (4.1b) we can see that we now get variation between the cells in the different colour groups. From the $R_j - R_i$ plot in the top of Figure 4.1b we can however see that there is no variance between the genes in each colour group for a given cell - i.e. all the red genes in an individual cell has the same expression. The variation only exists between cells. This is also true for the green and blue genes.

The goal of the second step in the addition of intrinsic variation is to add variation to every single gene without considering the colour group it belongs to. Thereby, resolving the aforementioned problem of identical gene expression within a cell. The variation in every single gene expression is added by multiplying a number drawn from a normal distribution with $\mu = 1$ and $\sigma = 0.5$ ($\mathcal{N}(1, 0.5)$) to every single gene expression (Figure 4.1c). This part of the intrinsic variance is kept constant and is thus not used as a parameter in the simulation.

The distribution of gene expression is commonly log-normal distributed [15]. This is one of the reasons why it is recommended to do the $log(x+1)$-transformation after normalisation in order to make the distribution normal. However, for the sake of simplicity we decided to work in a normal scheme from the beginning - illustrated by the use of the normal distribution in the step above. The main reason was that this gave us a better intuition for how to add batch noise as will later be explained. We did however also test if our result would be comparable when working in the log-normal scheme (Section 7.3 in Appendix).

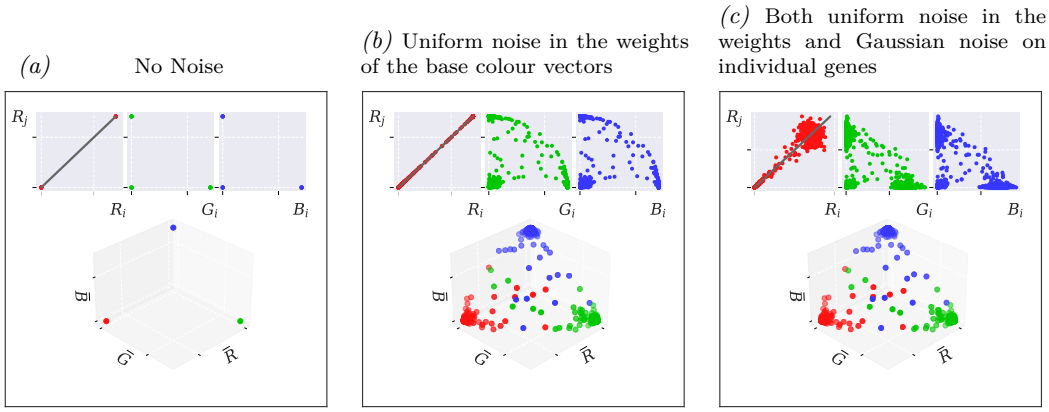


**Figure 4.1 | Illustration of the addition of intrinsic noise:** *(a)* - *(c)* illustrates how the two intrinsic noise steps generates variance in the gene expressions. *(a)* shows the batch when no noise have been added, *(b)* when only the uniform noise in the weights of the base colour vectors is added, and *(c)* when both the uniform noise in the weights and the Gaussian noise in each individual gene have been added. Each cell in the batch is represented as a dot in the 3D scatter plot at the bottom of *(a)*, *(b)*, and *(c)*. Each cell has here been reduced to a three dimensional object where the coordinates are the average expression of all the red ($\bar{R}$), green ($\bar{G}$), and blue ($\bar{B}$) genes, respectively. All the scatter plots contain the same number of points even this is not obvious to see in *(a)*. The top figure in *(a)*, *(b)*, and *(c)* shows the correlation between - from left to right - a red and a red gene, a red and a green gene, and a red and a blue gene. The function $y = x$ is indicated with the grey line in the red vs. red plot.

### Batch Noise

We are as previously stated interested in examining the influence of a batch noise on the clustering and identification of cell types. This batch noise can be implemented in many different ways but it depends on the underlying distribution. The log-normal distribution is highly associated with multiplicative noise [12]. We decided as stated before to work in a normal distributed system. A multiplicative noise in a log-normal distributed system will take the form of an additive noise in a normal distributed system [12]. This was the main reason why we chose to work in the normal scheme from the beginning because we found the idea of an additive batch noise more intuitive.

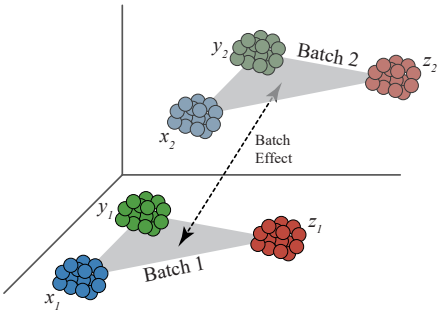


**Figure 4.2 | Illustration of the influence of the batch noise:** This figure uses a 3-dimensional illustration to show how the batch noise will shift the entire batch in a given direction and with a specific magnitude which in our simulation is determined by the batch noise vector $\vec{n}_B$.

The batch noise is now generated by adding the same *batch noise vector* $\vec{n}_B$ to every single cell within the batch. Thereby shifting each cell in the direction of this batch noise vector (Figure 4.2). Every entry in $\vec{n}_B$ is a number drawn from the uniform distribution $U(-n_B/2, n_B/2)$, where the width $n_B$ is the parameter that determines the level of the batch noise in the simulated data. Here it is important to notice that there is a difference between the *batch noise vector* $\vec{n}_B$ and the *batch noise magnitude* $n_B$.

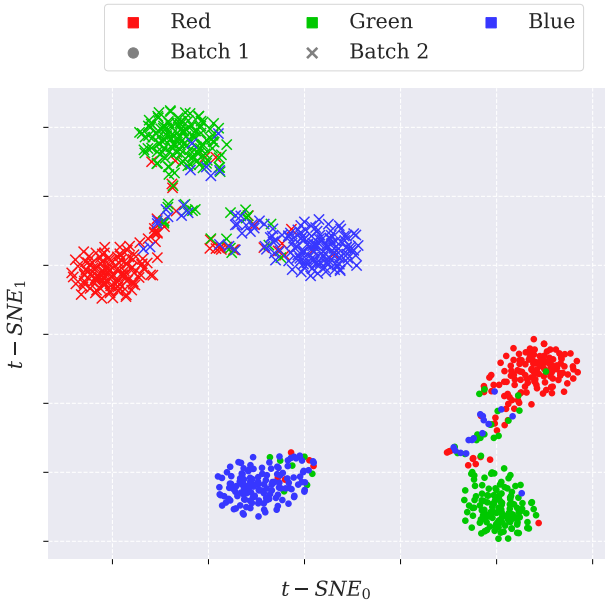


**Figure 4.3 | Example of two batches generated using the RGB-cell method:** This plot shows the $t$-SNE of a data set containing two batches with 150 red, green and blue cells in each. Both batches were generated with intrinsic noise, $n_I = 0.15$ and batch noise, $n_B = 0.075$. The colours indicate the different cell types and the marker style indicates the different batches.

After the addition of batch noise the gene expressions is then normalised using the *counts per million*, $CPM$ method (eq. (2.1)). In this way we have combined three different cell types into a batch.

We now have a framework for creating batches containing populations of cells of different cell types where each cell has been subjected to the same batch noise. The plot in Figure 4.3 shows the $t$-SNE of a data set comprised of two batches each containing 150 red, green, and blue cells. Both batches were generated using the same intrinsic noise, $n_I$ and level of batch noise, $n_B$ but with a different batch noise vector thereby shifting each batch in different directions. Here we can see how the $t$-SNE has grouped the different cell in such fashion that cells of the same cell type do not co-localise but were instead grouped closer to cells from its own batch.

### 4.1.3 Intrinsic & Extrinsic Distance to Quantify Differences Between Cell Types Within and Between Batches.

We are as stated previously interested in seeing what a relatively small batch noise will do to the pairwise distance between cells and how this will influence the clustering and identification of cells. Before addressing a method for measuring this influence we will introduce four different classes of cells in order to make referencing easier;

**Class** $A$**:** Cells of the same cell type and same batch
**Class** $B$**:** Cells of the same cell type but different batch
**Class** $C$**:** Cells of different cell type but same batch
**Class** $D$**:** Cells of different cell type and different batch

We can with these classes of cells in mind now define three different measurements that we can use to examine the influence of the batch noise.

**Average cluster width** ($w_c$) is a measurement of how much variance exists between the cells of the same cell type and batch (illustrated by magenta arrows in Figure

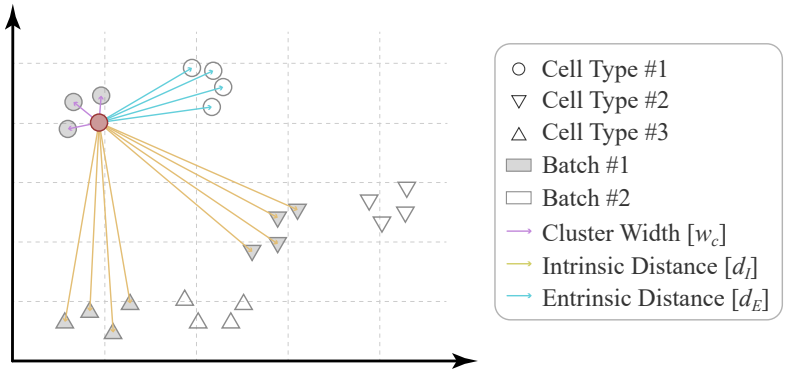


**Figure 4.4 | Illustration of cluster width $w_c$ , intrinsic distance $d_I$ , and extrinsic distance $d_E$ :** This figure illustrates how the parameters cluster width $w_c$, intrinsic distance $d_I$, and extrinsic distance $d_E$ are measured. The purple arrows shows how the pairwise distance is measured between cells of the same cell type and batch (Class $A$) for a single cell. The average of this measurement for all cells in both batches will give the cluster width $w_c$. The blue arrows indicates the pairwise distance between cells of the same cell type but different batch (Class $B$). The average of this measurement for all cells is the extrinsic distance $d_I$. Lastly the yellow arrows shows the pairwise distance between cells of different cell type but of the same batch (Class $C$). The average of which taken over all cells is the intrinsic distance $d_I$.

4.4). This is measured by averaging the pairwise distance between all cells of the same type and within the same batch (Class A) and is calculated by,

$$w_c = \frac{1}{N_B}\left(\sum_{k}^{N_B}\frac{1}{N_{C,k}}\sum_{m}^{N_C}\frac{1}{M_{k,m}}\sum_{j\neq i}\left\|\vec{c}_{im_ik_i}-\vec{c}_{jm_jk_j}\right\|\right), \text{ when } m_i = m_j \ \wedge \ k_i = k_j. \tag{4.7}$$

The parameter $N_B$ is the number of batches, $N_{C,k}$ is the number of different cell types in the $k$th batch, and $M_{k,m}$ is the number of cells in the $m$th cluster and $k$th batch. The $\vec{c}_{im_ik_i}$ term is the gene expression vector of the $i$th cell of the $m_i$th cell type from the $k_i$th batch.

In this way we end up with a single number that describes the average width of every single population of cells that exists in the data set.

**Intrinsic distance** ($d_I$) is a measurement that indicates the difference between cells of different cell types but in the same batch (illustrated by yellow arrows in Figure 4.4). This is measured as the average pairwise distance between all cells of different cell types within the same batch (class $B$).

The intrinsic distance $d_I$ is calculated in the same manner as eq. (4.7) but under the condition that $m_i \neq m_j \wedge k_i = k_j$.

**Extrinsic distance** ($d_E$) is a measurement of how much difference the batch noise introduces to cells of the same type (illustrated by cyan arrows in Figure 4.4). This is measured by averaging the pairwise distances between all cells that share cell type but are from different batches (class $C$).

The extrinsic distance $d_E$ is calculated in the same manner as eq. (4.7) but under the condition that $m_i = m_j \wedge k_i \neq k_j$.

There exists as mentioned in Chapter 2 many different methods for normalisation. Even different variations of the CPM method can be used. The scaling factor can in these cases vary from the usual $10^6$ by different factors of ten. The choice of normalisation will influence both cluster width $w_c$, intrinsic distance $d_I$, and extrinsic distance $d_E$. We can by introducing the relative intrinsic and extrinsic distances, $ID$ and $ED$,

$$ID = \frac{d_I}{w_c} \quad \text{and} \quad ED = \frac{d_E}{w_c}, \tag{4.8}$$

make the measurements consistent across normalisation methods as long as it uses global scaling. This will moreover also reduce the number of measurements needed to analyse the influence of the batch noise from three to two.

An illustration of how the relative pairwise distances are influenced by the intrinsic noise and the batch noise is shown in Figure 4.5. From this figure we can see how an increase in the level of the batch noise (Figure 4.5a and 4.5b) pushes cells of the same cell type away from each other. In the beginning when the level of batch noise $n_B$ is low cells of the same cell types but the different batches (class $B$) are still closer than cells of different cell types (class $C$). When the batch noise is sufficiently large the two batches are pushed so far away from each other that cells of class $C$ are now closer than cells of class $B$. On the other hand, when the intrinsic noise $n_I$ is increased (Figure 4.5c and 4.5d) we see how the pairwise distance between cells of different cell types begin to decrease and becomes harder to distinguish. In the end they will completely overlap.

We can now by combining different values of $n_I$ and $n_B$ construct batches with varying $ID$ and $ED$ (Figure 4.5e and 4.5f). This works very well for exploring the influence a batch noise will have on clustering and identification of cell types and how different data integration and batch correction methods can help to alleviate the problems caused by the batch noise.
An overview of the terminology used can be found in Table 7.1 in Appendix.

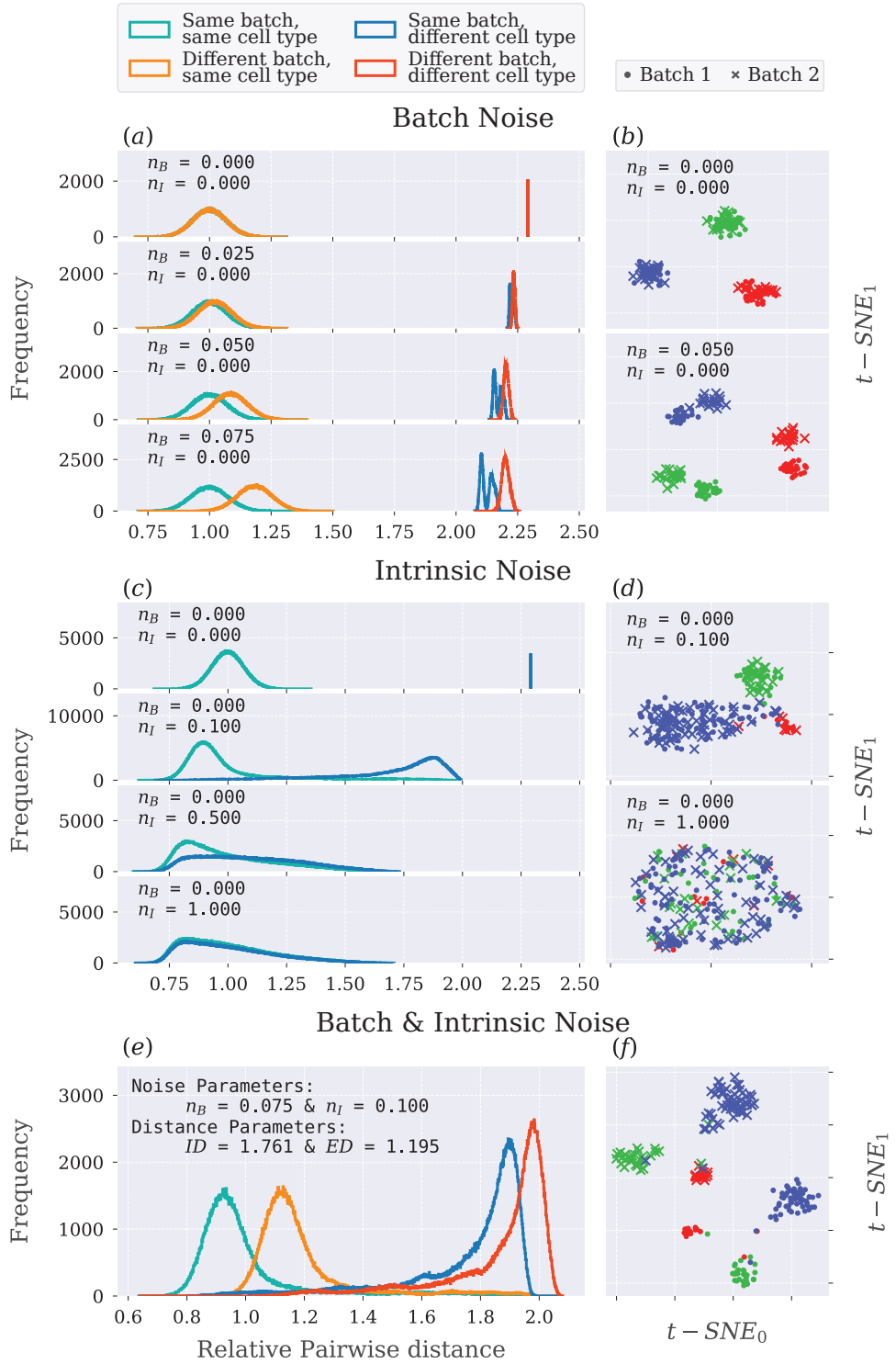


**Figure 4.5 | Effect of intrinsic noise and batch noise on relative intrinsic and extrinsic distances:** The subfigures *(a)* and *(c)* show how the relative pairwise distance changes as $n_I$ and $n_B$ varies. In *(a)* $n_I$ is kept at zero whereas $n_B$ is varied as indicated in the top left of each of the histograms. In *(c)* $n_B$ is kept at zero and $n_I$ is varied. Subfigure *(e)* is an example of when both $n_I$ and $n_B$ is above zero. Subfigures *(b)*, *(d)*, and *(f)* shows the $t$-SNE of the data set at different values of $n_B$ and $n_I$ as indicated in the top left corner of each.

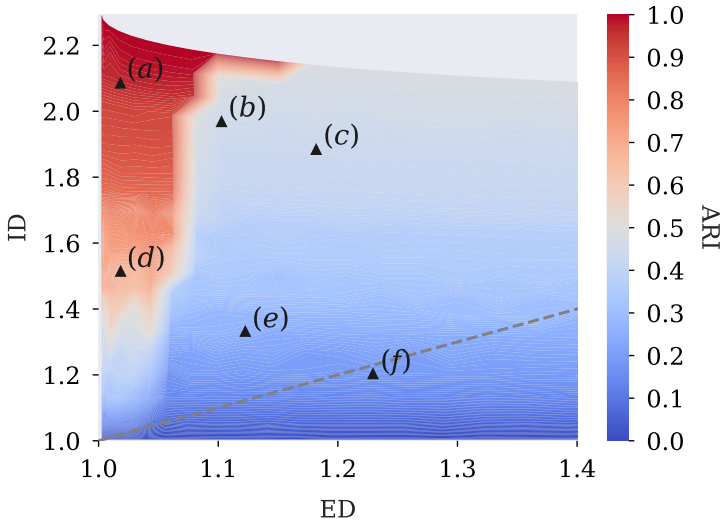


**Figure 4.6 | Maximum *ARI* as a function of *ED* and *ID* for data with no correction:** This figure shows the maximum $ARI$ at different values of $ED$ and $ID$. The grey line indicates the $ID = ED$ cutoff. Above this line when $ID > ED$, cells of the same cell type but different batch (class $B$) will be closer than cells of different cell type but same batch (class $C$). Below the line the opposite is the case.

## 4.2 Clustering & Identifying Cells based on Cell Type

Now that we have a method for constructing batches containing populations of cells of different types with variation in the gene expression we can begin to tackle the main question put forth by this thesis - i.e. how will a relatively small batch noise influence the clustering and identification of cells.

| Cell Type | Batch 1 | Batch 2 | Sum |
|---|---|---|---|
| Red | 100 | 100 | 200 |
| Green | 300 | 300 | 600 |
| Blue | 900 | 900 | 1800 |
| Sum | 1200 | 1200 | 2400 |
| **Red Genes** | **Green Genes** | **Blue Genes** | **Sum** |
| 100 | 100 | 100 | 300 |

**Table 4.1 | Composition of the batches in a standard data set.:** This table shows the number of red, green, and blue cells together with the number of corresponding genes for the standard data set used in this thesis. Both batches are created using the same intrinsic noise $n_I$ and level of batch noise $n_B$ but different batch noise vectors $\vec{n_B}$.

Firstly, we need to decide on a framework for how to evaluate the partition generated by a cluster algorithm at different levels of batch noise. We are in this part of the thesis interested in the identification of cell types. We will therefore evaluate the influence the batch noise has by how good the clustering algorithm is at grouping cells of the same cell type together in the same cluster. To measure this we use the adjusted Rand index $ARI$ discussed in Section 3.3.2 which can take a value between zero and one where an $ARI$ of 0 represent a completely random partition and 1 a partition that overlaps perfectly with the predefined cell types.

This was done by constructing a standard data set (Table 4.1) containing two batches of red, green, and blue cells. These batches was constructed for 20 different $n_B \in [0, 0.12]$ and $n_I \in [0, 1]$. For each combination of $n_B$ and $n_I$ the Leiden clustering algorithm was used to cluster the cells. The resolution was chosen such that we obtained the maximum $ARI$. A contour plot of the best $ARI$ at different combination of $ID$ and $ED$ is shown in Figure 4.6.

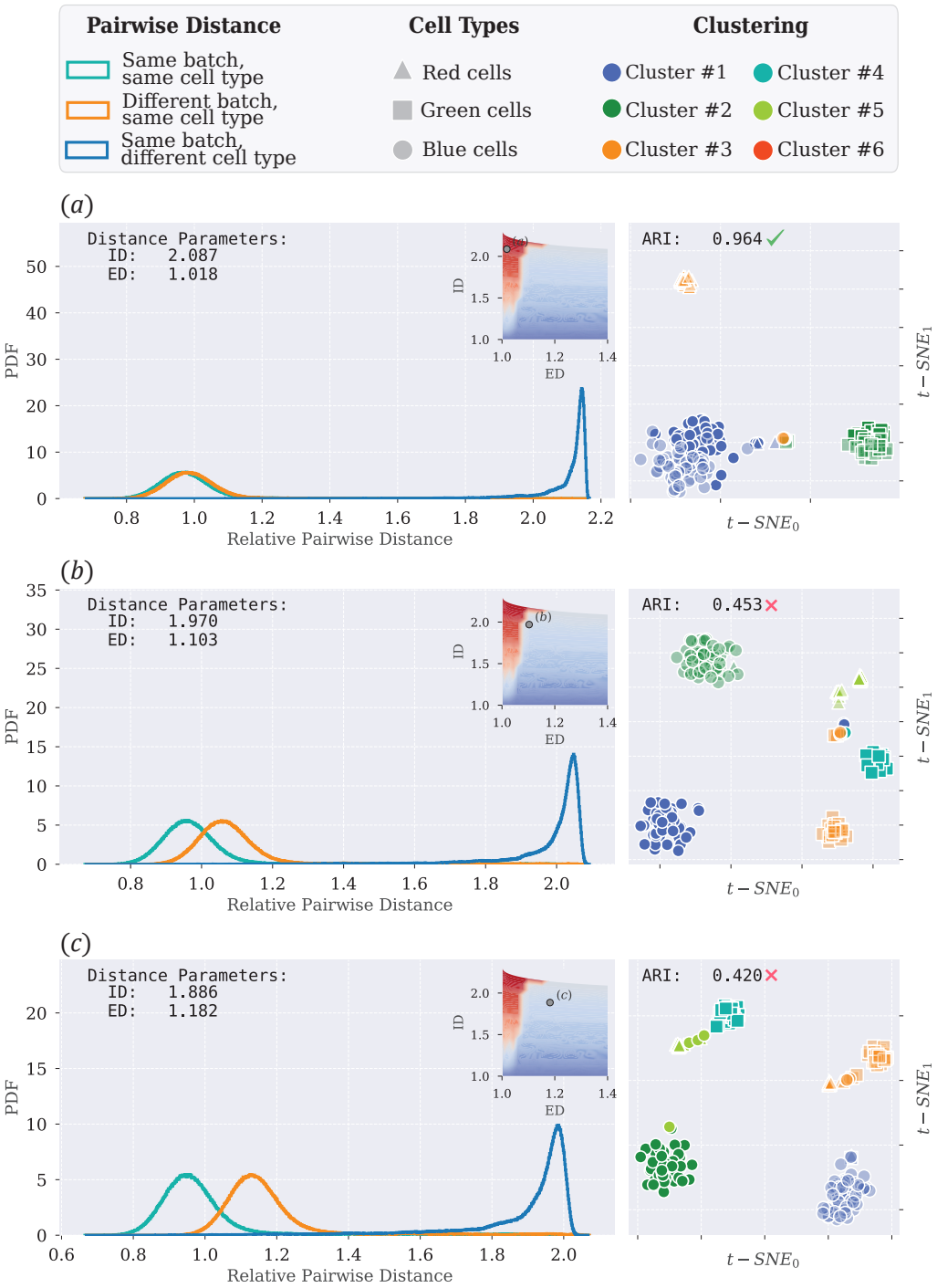


**Figure 4.7 | Relative pairwise distance & best possible clustering - part 1:** This figure shows (left) the relative pairwise distance and (right) the $t$-SNE for the best possible clustering determined by the maximun $ARI$ using the Leiden algorithm for the one region where clustering works well ($a$) and the two regions where the clustering algorithm fails to cluster based on cell type ($b$) and ($c$). This is also indicated in the contour plot in the top right corner of the pairwise distance plot. The transparency in the $t$-SNE plot indicates the two different batches. The check mark or cross mark next to the $ARI$-value in the $t$-SNE plot is also an indicator of good or bad clustering, respectively.

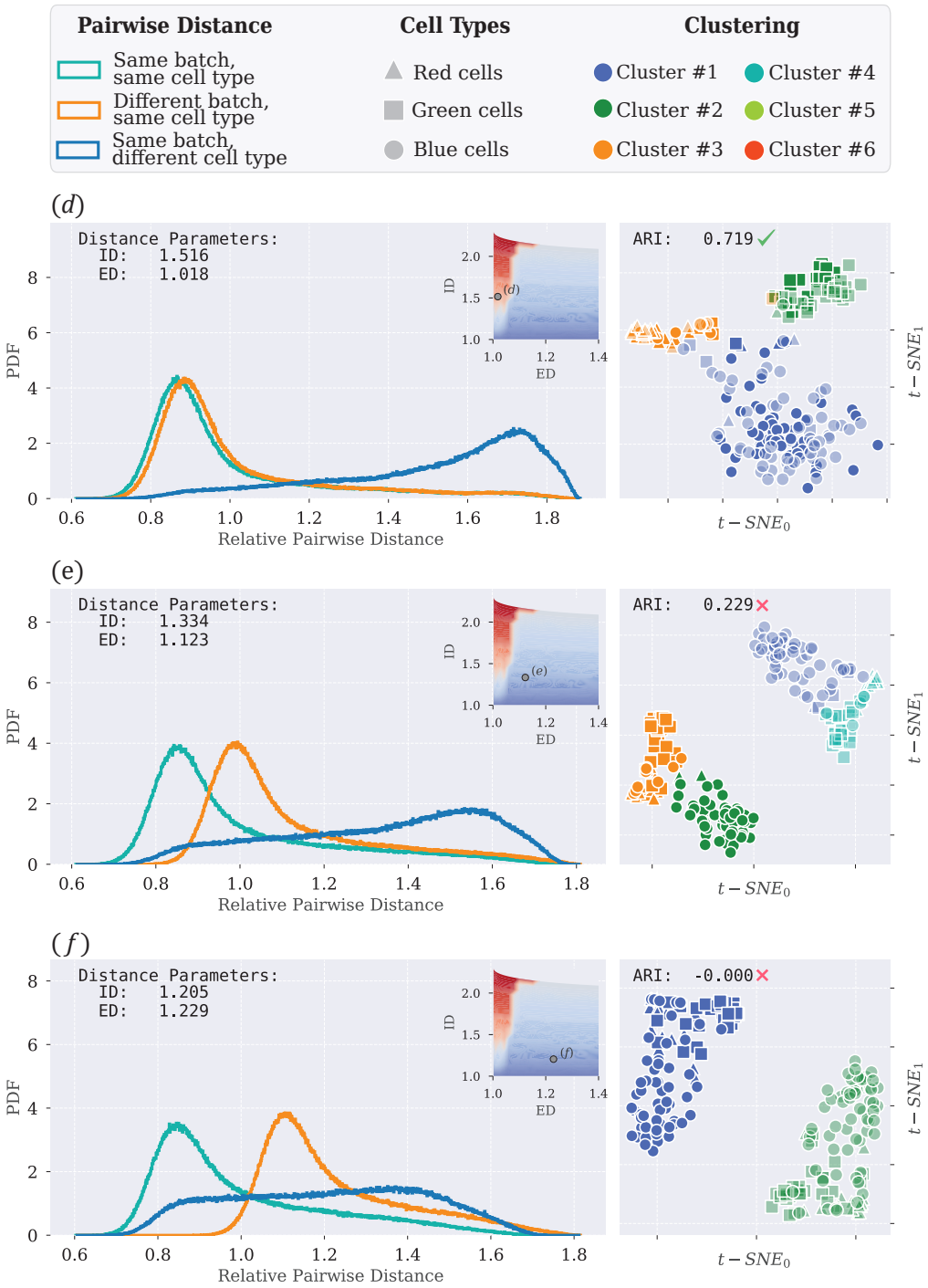


**Figure 4.8 | Relative pairwise distance & best possible clustering - part 2:** This figure shows (left) the relative pairwise distance and (right) the $t$-SNE for the best possible clustering determined by the maximun $ARI$ using the Leiden algorithm for the one region where clustering works well ($d$) and the two regions where the clustering algorithm fails to cluster based on cell type ($e$) and ($f$). This is also indicated in the contour plot in the top right corner of the pairwise distance plot. The transparency in the $t$-SNE plot indicates the two different batches. The check mark or cross mark next to the $ARI$-value in the $t$-SNE plot is also an indicator of good or bad clustering, respectively.

We can see from Figure 4.6 that a relatively low level of batch noise can have a huge influence on the clustering algorithm's ability to cluster cells together based on cell types. The grey line in Figure 4.6 indicates where $ID = ED$. This shows that the clustering algorithm will have problems clustering based on cell type even when cells of the same cell type but different batch (class $B$) is closer than cells of a different cell type but same batch (class $C$) - i.e. when $ID > ED$. There seems to be a rather sudden change in $ARI$ around an $ED \approx 1.1$ where clustering becomes very difficult if the batch noise pushes the distance beyond this point. There are of course exceptions to this rule. When the $ID$ goes below $\sim 1.4$ we also see a decrease in the $ARI$. This is not surprising since at these low levels of $ID$ it is hard to distinguish different classes of cells from each other.

These phenomena are explored in more detail by looking at the distribution of pairwise distances and a $t$-SNE visualisation (Figure 4.7 and 4.8) for the combination of $ID$ and $ED$ marked *a-f* in Figure 4.6.

We can see from the distribution of pairwise distances that even the slightest increase in distance between cells of the same cell type but different batches will cause the clustering algorithm to cluster based on batch as well as cell type. This is especially evident in Figure 4.7b. The distribution of the pairwise distances shows very clearly that cells of the same cell type are a lot closer than cells of different cell types but same batch (Figure 4.7b (left)). However, the clustering algorithm have still decided to split cells of the same cell type into different clusters, as show in the $t$-SNE plot (Figure 4.7b (right)). It is only when the distribution of the pairwise distance between cells from class $A$ and class $B$ are more or less on top of each other - i.e. when $ED \sim 1$ - that we see clusters based only on cell types (Figure 4.7a and 4.8d).

From this we demonstrate that a relatively low level of batch noise can make it very hard for the clustering algorithm to generate clusters based on cell types - i.e. such that across batches all cells of the same type will cluster together. We hypothesised that the correlation of the batch noise between cell types could be a factor in the emergence of batch effects. An examination of this hypothesis can be found in Section 7.4 in Appendix.

## 4.3 Clustering & Identification with Data Integration & Batch Correction

As previously discussed, a great effort has been put into different methods for data integration and batch correction in order to combat the problems that arise from the batch noise [15, 14]. With the *RGB*-method we have created a framework for analysing distances between classes of cells. This will allow us to explore the effect of the different data integration and batch correction methods. We are interested in seeing if the application of these methods will change the values of $ED$ and maybe also $ID$ in order to move a data set from one of the regions in Figure 4.6 where clustering failed to one where clustering succeeded.

We have chosen the methods; *Scanorama*, *Harmony* and *ComBat,* where *ComBat* is a batch correction method typically used when the batches has been generated under more similar conditions. *Scanorama* on the other hand is a more complicated data integration method which can be used on more complex data sets [14]. These

methods have been shown to be some of the most efficient and accurate methods for combating the troublesome batch effect [14].

In order to illustrate the effect we generated a batch in the same manner as described in Table 4.1 situated at a value of $ID$ and $ED$ far out into the region where the clustering algorithm could no longer perform good clustering. Hereafter, the batch noise was corrected using the *scanpy* libraries implementation of the aforementioned methods. The result of this correction is shown in Figure 4.9.

We can see in the $ARI$ contour plots that the original data is placed far out at a very large $ED$ (Figure 4.9a (left)). It is both evident from the $ARI$ value and the $t$-SNE plot of best clustering that the clustering does not work here (Figure 4.9a (right)). After correction some interesting phenomena can be observed. The first thing being that for this amount of batch noise and intrinsic noise it is only *ComBat* and *Scanorama* that moves the data into the region where the clustering algorithm can perform good clustering (Figure 4.9b and 4.9c). Here, it is interesting that the *ComBat* method only decreases $ED$ - i.e. it only targets the component of bad clustering that is related to batch noise. The *Scanorama* method, on the other hand, will both decrease the $ED$ as well as increasing the $ID$ so not only will it move cells of the same cell type closer to each other it will also move cells of different cell types further away making the clustering even easier. This artefact can be due to the fact that *ComBat* is a "simple" batch correction algorithm whereas *Scanorama* is a more complicated data integration method. For this level of $ED$ and $ID$ the *Harmony* method was the only one that did not manage to move the data into the region of good clustering (Figure 4.9d). Here, it is interesting to see that in the same way - and to the same degree - as in the *Scanorama* method it increased the $ID$. However, it did not manage to correct for the batch noise. The $ED$ decreased but not enough for it to be moved into the region of good clustering.

In the in pairwise distance plot for the *Scanorama* and *Harmony* method we observe another interesting phenomena (Figure 4.9c (left) and 4.9d (left)). Although it is faint, the distribution of pairwise distances for cells of the same batch and the same cell type (class $A$) and cells of the same cell type but different batch (class $B$) now have a second peak around 0.25. This indicates that the correction methods have split the cells into two groups; one small group which has been moved further away from cells of a different type but same batch (class $C$) and another much larger group which distance compared to class $C$ has not change significantly. To investigate this phenomena we decided to examine how each of the data integration and batch correction methods worked on a cell type level. To do this we measured the relative pairwise distance and calculated the $ED$ and $ID$ for each cell type separately in contrast to before where the values of $ID$ and $ED$ were combined and averaged within cells of the same class ($A$, $B$, $C$) (Figure 4.10).

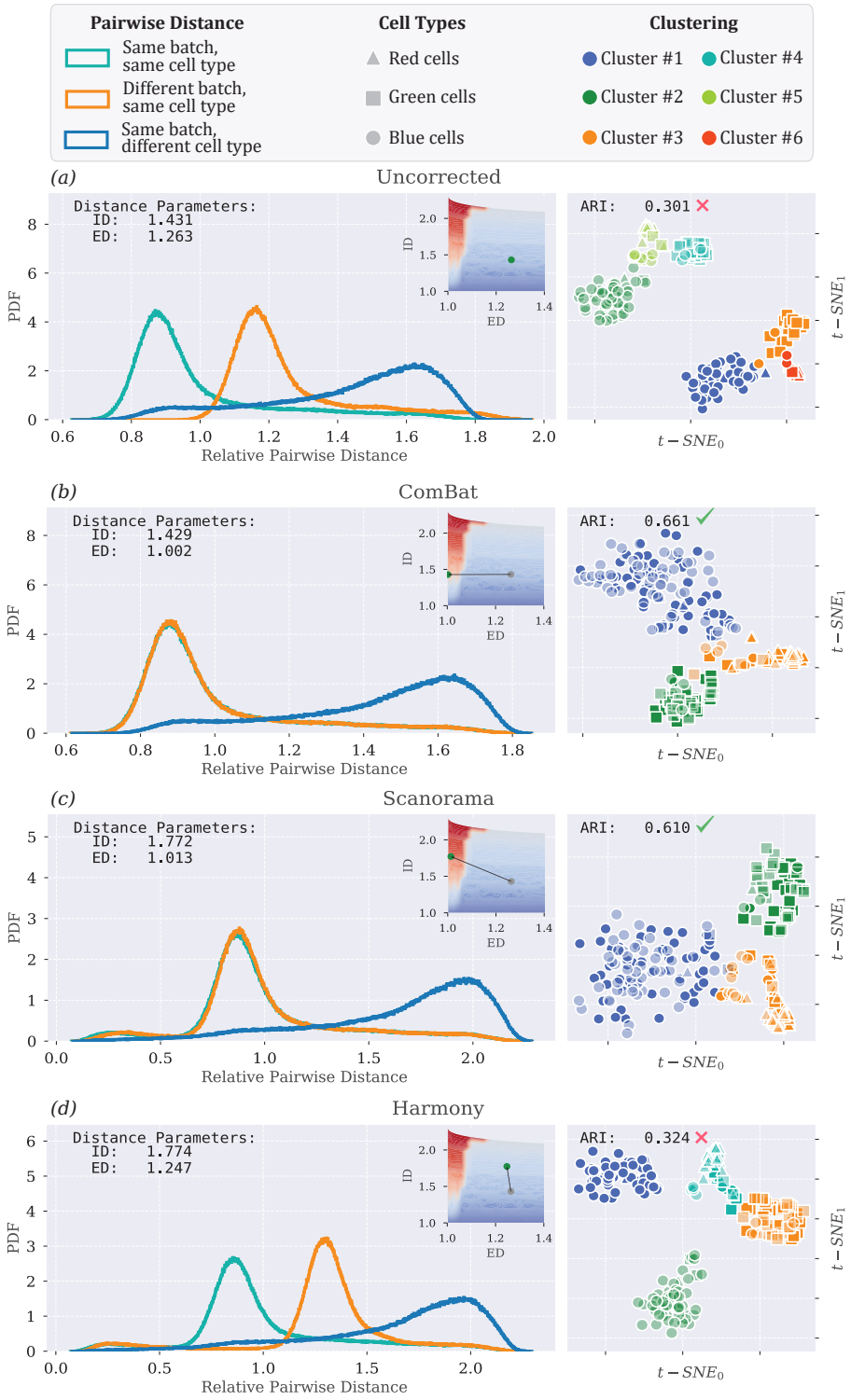


**Figure 4.9 | Relative pairwise distance & best possible clustering for corrected data:** This figure shows (left) the relative pairwise distance and (right) the $t$-SNE for the best possible clustering determined by the maximum $ARI$ using the Leiden algorithm. The title of each plot indicates what if any correction method was used. The contour plot shows the shift in $ID$ and $ED$ after the data correction - grey: uncorrected, green: corrected. This indicates if the correction method have shifted the data into a region of good clustering (red) or bad clustering (blue/blue-grey). The transparency in the $t$-SNE plot indicates the two different batches. The check mark or cross mark next to the $ARI$-value in the $t$-SNE plot is also an indicator of good or bad clustering, respectively.

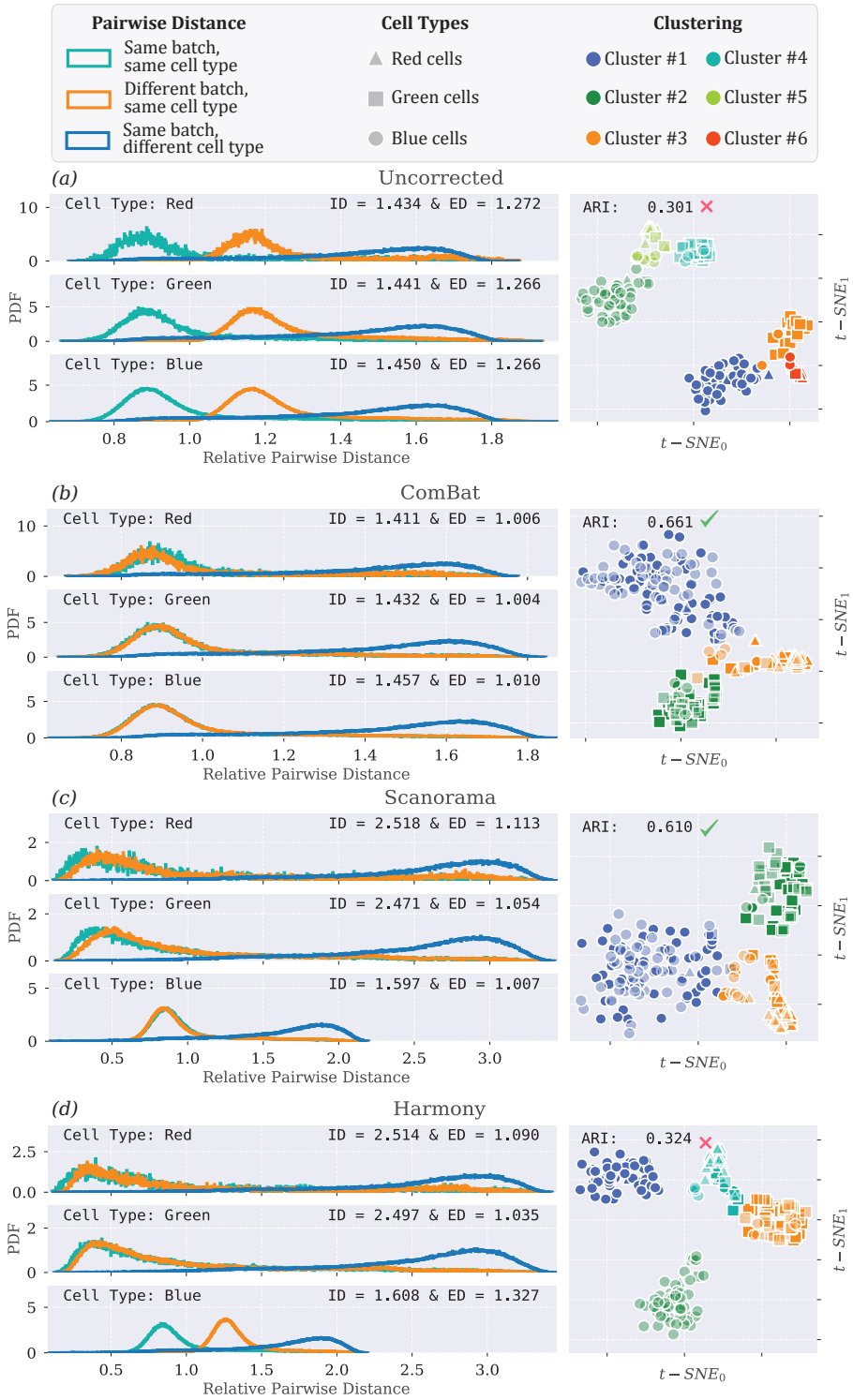


**Figure 4.10 | Relative pairwise distance for each cell type & best possible clustering for corrected data:** This figure shows (left) the relative pairwise distance for each cell type: red, green, and blue in the data set and (right) the $t$-SNE for the best possible clustering determined by the maximum $ARI$ using the Leiden algorithm. The title of each plot indicates what if any correction method was used. The transparency in the $t$-SNE plot indicates the two different batches. The check mark or cross mark next to the $ARI$-value in the $t$-SNE plot is also an indicator of good or bad clustering, respectively.

From this we again see how the *ComBat* method keeps the internal structure constant but moves the other batch closer (Figure 4.10b). However, something very interesting occurs in the case of the *Scanorama* and *Harmony* methods (Figure 4.10c and 4.10d). For the red and green cells both methods perform quite well and we see the same phenomena as before where cells of the same cell type is moved closer together and cells of different cell types are pushed further apart. The interesting thing, however, occurs for the blue cells. Here we see that both of the methods do not, to the same extent, correct the data as for the two previous cell types. It is also for this cell type that the *Harmony* fails in its correction. This we can see from the *Harmony*'s *t*-SNE plot where we see the red and green cells have come together into nice clusters whereas the blue cells remain separated into two different clusters (Figure 4.10d (right)). The difference in cluster sizes, with blue being the largest, explains why the second peak was much smaller - i.e. the larger number of cells will also result in an proportional larger number of pairwise distances.

The key aspect to take away from this analysis is that the good data integration and batch correction methods will most importantly decrease the distance between cells of the same cell types in different batches and in some cases even making cells of different cell types more distinguishable by moving them away from each other.

## 4.4 Extrinsic & Intrinsic Distances in Real Data

It would be interesting to see if the phenomena of bad clustering despite $ID > ED$ observed in our simulation is present in data from real scRNA-seq experiments. For this purpose we used data on blood cell types (*hematopoiesis*) from [5]. The data was obtained from the MarionLab's github page [16]. To make the analysis more simple we only used the cell types *granulocyte–monocyte progenitor* (GMP), *common myeloid progenitor* (CMP), and *megakaryocyte–erythrocyte progenitor* (MEP) since these where the only ones present in both batches, *Paul* and *Nestorowa*. There are 3543 cells of the before mentioned cell types in the data with 3543 genes each. A more thorough review of how the cells are distributed is shown in Table 4.2.

| Cell Type \Batch | Nestorowa | Paul | *Sums* |
|---|---|---|---|
| CMP | 328 | 481 | 801 |
| GMP | 123 | 1154 | 1277 |
| MEP | 362 | 1095 | 1457 |
| *Sums* | 813 | 2730 | 3543 |

**Table 4.2 | Composition of the haematopoietic data set:** This table shows how the cells are distributed according to cell type and batch in the data of blood cell types from [5] used in our analysis.

Inspired from our results obtained in Figure 4.10 we decided to look at the $ED$ and $ID$ for each individual cell type (Figure 4.11).

The first thing that jumps to mind when looking at the results is the big difference between $ID$ and $ED$ for the different cell types (Figure 4.11). We can see that the CMP cells - both before and after the correction - are very hard to distinguish from both cells of different cell types as well as cells from different batches. This phenomena is very likely explained by the expected development path of haematopoietic cell types (Figure 4.12). Here we can see the CMP is an intermediate step in the formation of GMP and MEP cells [5].

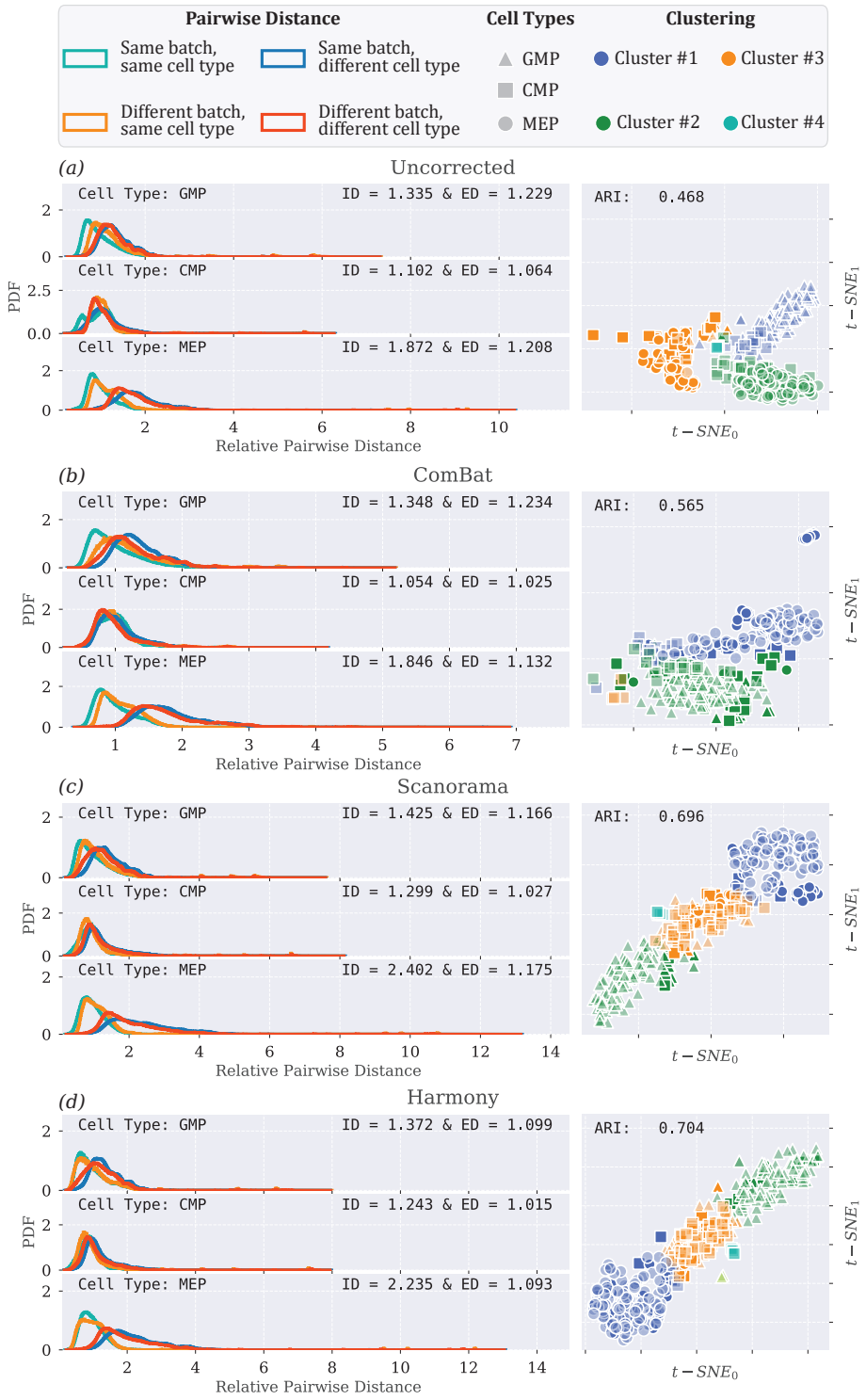


**Figure 4.11 | Relative pairwise distance & best possible clustering for corrected and uncorrected blood type data:** This figure shows (left) the relative pairwise distance for each of the three cell types; GMP, CMP, and MEP from [5]'s blood type data set used in our analysis. The $t$-SNE to the right shows the clustering with the heights $ARI$ obtained by the Leiden algorithm. The title of each plot indicates what correction method if any was used. The $ED$ and $ID$ for each of the cell types are shown in the top right corner of the relative pairwise distance plot. The transparency in the $t$-SNE plot indicates the two different batches.

The tendency of the clustering algorithm to cluster cells based on batch rather than cell types even when $ID$ is larger $ED$ as observed in our simulation is also present in the real uncorrected data (Figure 4.11a). We can see from the $t$-SNE plot of the clustered uncorrected data (Figure 4.11a (right)) that each cluster more or less only contains cells from one batch - i.e. each cluster is ~100% homogeneous regarding batch.

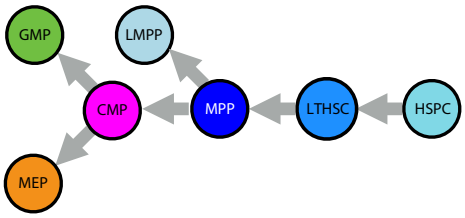


**Figure 4.12 | Development path of haematopoietic cell types:** This image illustrates the expected development path of haematopoietic cell types. Image taken from [5]

However, when it comes to the corrections we see very different results than in our simulation. This time it is very clearly *Scanorama* and *Harmony* which performed the best correction (Figure 4.11c and 4.11d). Both methods managed to increase the difference between $ID$ and $ED$ for all cell types and improve the $ARI$ score by a significant amount. The improvement is also evident from their respective $t$-SNE plots (Figure 4.11c (right) and 4.11d (right)). Here we see the GMP and MEP cells grouped at different ends with the intermediate cell type CMP sandwiched in between.

When comparing the relative pairwise distance plot for the uncorrected data (Figure 4.11a (left)) with that for *ComBat* (Figure 4.11b (left)) we see that *ComBat* have not managed to correct the distances in any significant way. On the contrary it removed the small difference between $ID$ and $ED$ that existed for the CMP cells before correction. However, when looking at the clustering in the $t$-SNE we see that the small level of correction *ComBat* managed to perform on the GMP and MEP cells did improve the clustering of these cells quite a bit (Figure 4.11b (right)). This is also evident in the improvement of the $ARI$.

## 4.5 Discussion & Recap

The key aspect to take away from this analysis of the influence of batch noise on clustering and identification of cells is that it is not only the magnitude of the batch noise that makes the clustering algorithms cluster based on batch instead of cell type. From Figure 4.6 we could see that this would occur even when $ID > ED$. There was almost a hard border at $ED \approx 1.1$ between good clustering (red area) and bad clustering (blue/blue-grey area). This indicates that there are other parameters besides the magnitude of batch noise which influences clustering of cells. We hypothesised as stated earlier that one of these parameters could be the correlated nature of the batch noise - i.e. that all cells in a batch is affected by the same batch noise. We briefly explored this hypothesis by trying to decorrelate the batch noise by rotating the batch (Section 7.4 in Appendix). The results gained from rotating the batches showed no difference compared to the results from before the rotation. We believe that this could have been caused due to many of the *axes of rotation* in the 300-dimensional gene space would introduce a negligible change in the correlation, even for relatively large angles. This is of course only a hypothesis and would need further exploration.

After this examination of batch noise we explored how different batch correction and data integration methods would affect the data sets' position in our $ID/ED$-landscape and how this would influence the clustering. For this purpose we used *ComBat*, *Harmony*, and *Scanorama*. From this we saw that all the batch correction methods improved the data sets' position in the $ID/ED$-landscape but only *ComBat* and *Scanorama* managed to move the data set into the region of good clustering. The *ComBat* method was the most consistent between these two in its correction of the data set. Applying an equal amount of correction for all cell types whereas *Scanorama* applied a different level of correction to the group of blue cells compared to the red and green cells. However, $ARI$-value is comparable between the two methods and the difference could very easily be due to fluctuations in the simulations. The *Harmony* method as stated before did not manage to move the data set into a region of good clustering because the method did not manage to correct the blue cells. It is interesting to see how both *Scanorama* and *Harmony* had some issues with the cell type that contained the most cells (blue cells).

Lastly we used our concepts of $ED$ and $ID$ to examine real scRNA-seq data on blood cell types from [5]. We were interested in seeing if we would observe the same phenomena of bad clustering despite $ID > ED$. Before applying any correction we did indeed see bad clustering (clustering based largely on batch rather than cell type) even though $ID > ED$ for most of the cell types. However, as one can expect the picture becomes a bit more complicated when looking at real data. We can see that the distance between different cell types is quite a bit smaller than in our simulated data. This is probably in a large extend due to the fact that the data examined the development path of hematopoietic cell types and we would therefore expect the cells to be relatively similar.

When we apply the three batch correction algorithms we again see an improvement in the relationship between $ID$ and $ED$ and a better clustering based on cell types. This time it was the *Scanorama* and *Harmony* methods which managed to perform the best correction whereas *ComBat* did not sufficiently manage to correct for the batch effect. The fact that the optimal batch correction method differ between the simulated and the real data could very well be an illustration of the difference between simple batch effect correction performed by *ComBat* and the more complicated data integration problems tackled by *Scanorama* and *Harmony*.

Chapter 5

# Clustering Gene/Gene Correlations in scRNA-seq data

Inspired by the success of identifying different cell types we will in this chapter apply the same clustering algorithms to identify different gene/gene correlations in scRNA-seq data. For this purpose we will use the scRNA-seq data from Brickman's lab[1]. This data set contains 4,671 different cells and expression for 9,732 genes. This data set is comprised of cells from 8 different batches and contains 16 different cell types. In order to ease the computational burden we limited the data to only the four gut-cells in the data set - *DE-gut*, *Foregut*, *Hindgut*, and *Midgut*. This reduced the number of cells to 1625.

| Data Information | |
|---|---|
| Number of cells: | 1,625 |
| Number of genes: | 9,732 |
| Number of batches: | 8 |
| Number of cell types: | 4 |

**Table 5.1 | Information about the data used in the investigation of gene/gene correlations:** This table contains general information on the data set from Brickman's lab when filtered to only include *gut-cells*.

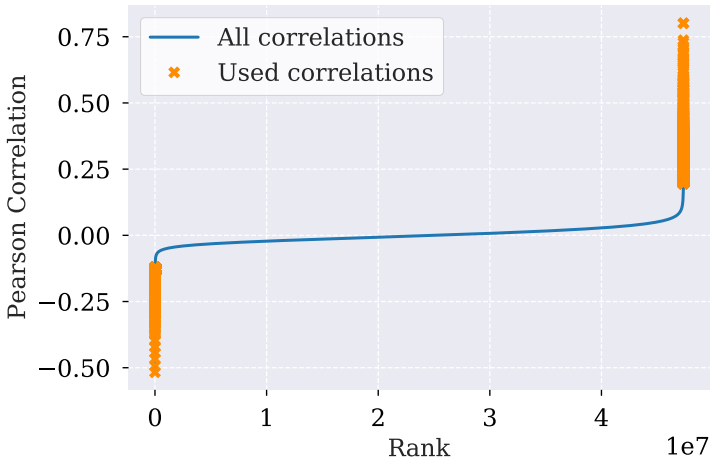


**Figure 5.1 | The unique gene/gene correlation ranked by their Pearson correlation coefficient:** This figure shows all (blue line) the unique gene/gene pairs filtered in the data set together with the total 20,000 gene/gene pairs chosen for further investigation (orange crosses).

If we used all of the 9,732 genes we would end up with $\sim$ 50 million unique gene/gene correlations. Again, this would make the computational task extremely burdensome. We therefore limited our investigation to the 10,000 highest positive and the 10,000 highest negative correlated gene/gene pairs (Figure 5.1) determined by the Pearson correlation. By doing so we have also managed to exclude the most uncorrelated ($\rho \sim 0$) gene/gene pairs.

[1] Publication in review

## 5.1 Creating a Matrix of Vectorised 2D Histograms of Gene/Gene Correlation Patterns

In order to use the clustering mechanism used in scRNA-seq we need to find a way to store the information of the correlation between all the unique gene/gene pairs in a matrix. This was done by vectorising a 2D histogram of the pairwise gene correlation. Before constructing this matrix we need to go through five steps. Firstly, we filter out the cells where neither of the genes in the gene/gene pair are expressed. Next, we use the $min - max$ normalisation,

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)},$$

to rescale the gene expression for each gene. This will scale the range of each gene expression such that it will lie in $[0, 1]$. This will have no influence on the Pearson correlation coefficient but it will make the comparison of gene/gene pairs easier. This allows us to create the classical gene/gene scatter plot (Figure 5.2a) which is then binned into 625 bins with 25 bins on each axis (Figure 5.2b). Before vectorising the 2D histogram we need to add a spatial influence - i.e. the frequency in each bin is altered based on its neighbours. The reason for this is best illustrated with a simplified example. If given a vector $\vec{v} = \{1, 0, 0, 0, 1\}$ where each entrance represents a bin in a histogram, the distance between this vector and the vector $\vec{v}_1 = \{1, 0, 0, 1, 0\}$ will be the same as to $\vec{v}_2 = \{1, 1, 0, 0, 0\}$. We are not interested in a cell placed in a neighbouring bin having just as big an influence on the distance between gene/gene pairs as a cell placed in a bin on the opposite site of the histogram. This problem can be solved by applying a Gaussian filter with $\sigma = 1$ on the 2D histogram (Figure 5.2c). The convoluted 2D histogram can then be vectorised (Figure 5.2d). These steps are performed for all the desired gene/gene pairs and stored in a matrix (Figure 5.2e).

## 5.2 Custom Distance Measurement for Comparing Gene/Gene Pairs

Before clustering the different gene/gene pairs together we need to determine which metric to use when measuring the distance between gene/gene pairs. In the case of our regular scRNA-seq data where each row corresponds to a different cell we used the euclidean distance between cells as a measure of similarity. This is however not suitable as a stand-alone measure for the distance between the vectorised gene/gene pairs. As mentioned earlier we only included the unique gene/gene pairs in our data set. This means that we only vectorised the histogram for $gene_j$ vs. $gene_i$ and excluded $gene_i$ vs. $gene_j$ since this would give the exact same Pearson correlation coefficient. However, when measuring the euclidean distance between the gene pairs the orientation of the scatter plot can have a big influence. It is therefore important to measure the distance to both orientations in order to get the full picture.

To do this we first need to de-vectorise the data back into the matrix form of the 2D histogram,

$$\mathbf{A_m} = \text{vec}^{-1}_{(25,25)}(\vec{g}_m), \tag{5.1}$$

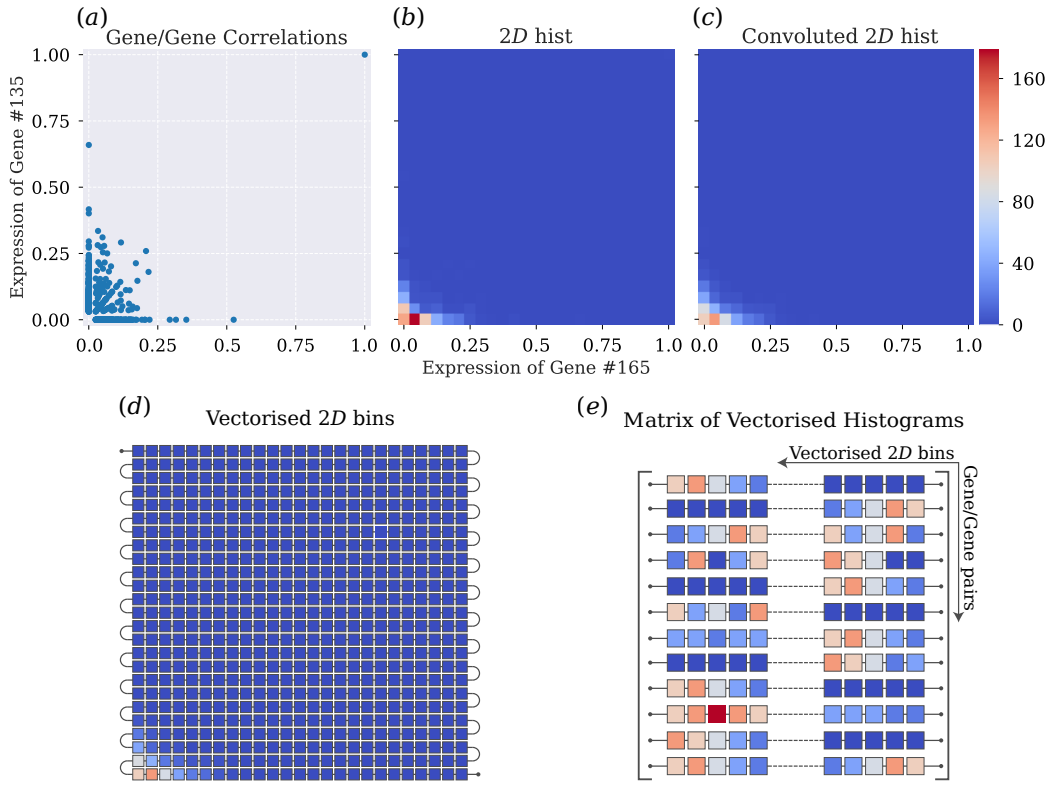


**Figure 5.2 | The steps in vectorising the gene/gene correlation patterns:** The vectorisation of gene/gene correlation patterns starts with the classical gene/gene scatter plot *(a)*. This scatter plot is then turned into a 2D histogram *(b)* which is convoluted using a Gaussian filter with $\sigma = 1$ *(c)*. The convoluted 2D histogram is then vectorised *(d)* and stored in a matrix where the rows represent the different gene/gene pairs and the columns are the bins from the 2D histogram *(e)*.

where $\vec{g}_m$ is the $m$th gene/gene pair, $\mathbf{A_m}$ is the de-vectorised matrix, and $\text{vec}^{-1}_{(a,b)}(\vec{v})$ is the inverse vectorisation function. This will rescale a vector $\vec{v} \in \mathbb{R}^{ab}$ to a matrix $\mathbf{A} \in \mathbb{R}^{a,b}$. The matrix form is then transposed and re-vectorised,

$$\vec{g}_m^{\,\prime} = \text{vec}(\mathbf{A_m}^T), \tag{5.2}$$

where $\vec{g}_m^{\,\prime}$ is the vectorised form of the 2D histogram with the opposite orientation compared to $\vec{g}_m$.

We can now measure the distance between the $m$th and the $k$th gene/gene pair for both orientations of $m$,

$$d^{(1)}_{k,m} = ||\vec{g}_k - \vec{g}_m|| \tag{5.3}$$

and

$$d^{(2)}_{k,m} = ||\vec{g}_k - \vec{g}_m^{\,\prime}||. \tag{5.4}$$

We then set the smallest as the "true" distance between the $m$th and the $k$th gene/gene pair,

$$d_{k,m} = \min\left(d^{(1)}_{k,m},\ d^{(2)}_{k,m}\right). \tag{5.5}$$

This is also illustrated in pseudocode in Algorithm 5.1.

**Algorithm 5.1** Custom distance function for comparing gene/gene pairs.

```
def custom_distance(v1, v2):
    '''
    Parameters:
        v1 and v2: Two N-dimensional vectors.
    '''
    d1 = norm(v1 - v2)                          # calculating euclidean distance
    A = v2.reshape([sqrt(N), sqrt(N)])          # reshaping the vector back to 2D
    A_T = A.T                                   # transposing the matrix
    v2_prime = A_T.reshape(N)                   # vectorising the transposed matrix
    d2 = norm(v1 - v2_prime)                    # calculating euclidean distance of new vector

    return min(d1, d2)                          # returns the smallest distance
```

## 5.3 Clustering Gene/Gene Pairs based on Vectorised 2D Histograms

Now that we have a metric for measuring the distances between the gene/gene pairs we can once again use the Leiden clustering algorithm on our vectorised gene/gene data. By using a resolution of 0.05 the Leiden clustering algorithm returned five different clusters [2] (Figure 5.3f). The average frequencies in each bin was calculated for each of the five different clusters still accounting for the orientation of the gene/gene pair. The five average gene/gene pairs are then de-vectorised and $log(x + \epsilon)$-transformed, where $\epsilon$ is an extremely small number[3]. This was done in order to make the variance appear more visible (Figure 5.3a - 5.3e).

Out of these five subpopulations of gene/gene correlation patterns we find cluster #1 especially interesting. This cluster exhibited a mix between positive and anti-correlated behaviour (Figure 5.3a) which is a trait we hypothesise could be an indicator of gene regulation important in cellular decision making. Clusters #2, #3, and #5 (Figure 5.3b, 5.3c, and 5.3e) appears to be very similar. We can also see from the $t$-SNE plot (Figure 5.3f) of the clusters that these three clusters are mostly made up of negative correlations. These clusters could very well have been merged if a lower resolution had been chosen. Cluster #4 exhibits classical positive correlated behaviour which is also evident from the $t$-SNE plot.

[2] Five random gene/gene pairs for each of the five clusters is plotted in Figure 7.3 in Appendix.
[3] We used the `finfo(float).tiny` value of $\sim 2 \cdot 10^{-308}$ from the *numpy* library

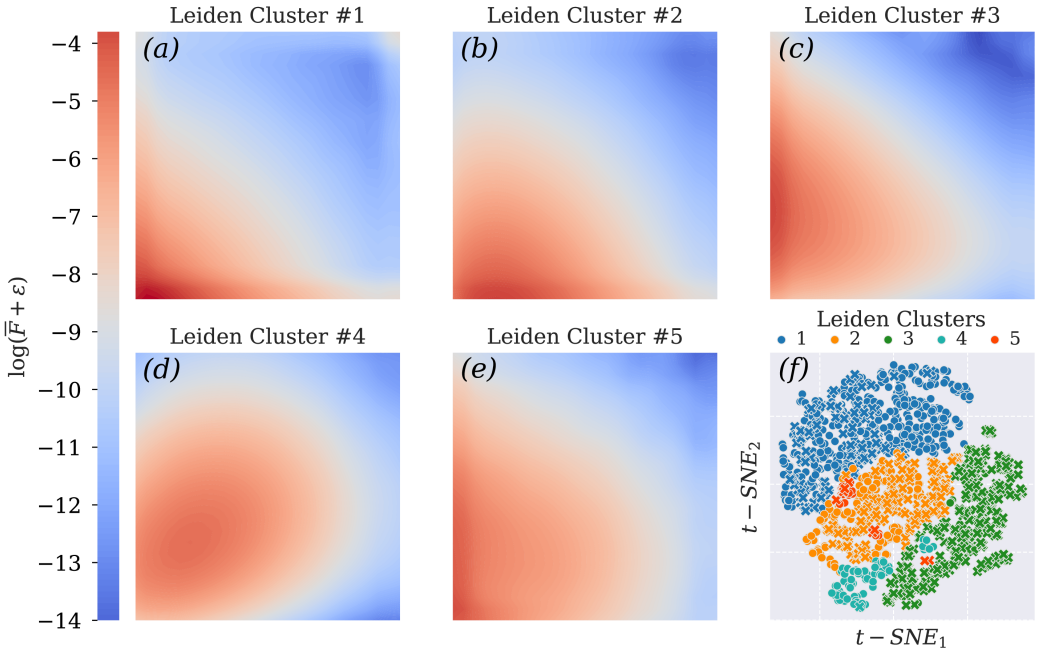


**Figure 5.3 | Average gene/gene configuration & *t*-SNE visualisation for the five Leiden clusters:** *(a)* - *e* shows the average frequency $\bar{F}$ of the de-vectorised 2D histogram of the gene/gene pairs from their respective cluster. This average has been $\log\left(\bar{F}+\epsilon\right)$-transformed where $\epsilon$ is a very small number in order to better illustrate the differences in average frequency. *(f)* shows the *t*-SNE plot coloured according to the five different clusters with the markers indicating positive (circles) and negative (crosses) Pearson correlation coefficients.

Chapter 6

# Discussion & Conclusion

This thesis has mainly focused on the examination of the influence of a batch noise on the euclidean distance between cells and the subsequent influence on clustering and identification of different cell types. This was primarily performed through a theoretical simulation. We therefore coined the idea of an RGB-cell with inspiration from the *RGB* colour code where the colour of a cell represents its type. The idea behind this simulation was to make a simple and intuitive way for us to examine the influence of different magnitudes of batch noise. In order to make a simple scheme for analysing the influence of a batch noise we defined the relative intrinsic distance $ID$ and the relative extrinsic distance $ED$ as the average distance between cells of different cell types inside a batch and the average distance of the same cell types between batches, respectively. The batch noise influence on the distance between cells is in this way reduced to two parameters making the comparison between different magnitudes of batch noise easier.

We initially showed how an increase of the magnitude of the batch noise caused the batches to drift apart - i.e. the euclidean distance between cells of the same cell type between batches increases with the magnitude of the batch noise (Figure 4.5a). After this we examined the influence of the batch noise on clustering identification of different cell types. This showed that the clustering algorithm had a hard time clustering cells based on cell types even for relatively small levels of batch noise. This was evident from the fact that even when cells of the same cell type but of different batches is closer than cells of a different type but same batch - i.e. when $ID > ED$. We could see there existed almost a hard border around $ED \sim 1.1$ which separates regions of good clustering (based on cell type) from regions of bad clustering (based on batch).

Furthermore, we applied our scheme of analysis using $ID$, $ED$, and $ARI$ to examine the effect of three popular batch correction methods; *ComBat*, *Harmony*, and *Scanorama* on our RGB-simulation. From this examination we could show how both *Scanorama* and *ComBat* managed to shift the data set's position in the $ID/ED$-landscape from the region of bad clustering into the region of good clustering. This was not the case for the *Harmony* method but on further examination we could see that this was due to bad correction of the largest cell type group (blue cells). This was also though to a lesser extend observed in the *Scanorama* method as well.

Lastly we used our analysis scheme on real data for blood cell types from [5]. We could see the same phenomena with cells clustering based on batch even though $ID$ is larger than $ED$. The picture was however a little bit more complicated because

we used data examining the development path of blood cells thereby making the different cell types a lot more similar than the red green and blue cells from our simulation. However, when we apply the three batch correction algorithms we again see an improvement in the relationship between $ID$ and $ED$ and a better clustering based on cell types. This time it was the *Scanorama* and *Harmony* methods which managed to perform the best correction whereas *ComBat* did not sufficiently manage to correct for the batch effect. The fact that the optimal batch correction method differ between the simulated and the real data could very well be an illustration of the difference between simple batch effect correction performed by *ComBat* and the more complicated data integration problem tackled by *Scanorama* and *Harmony*.

From this analysis we concluded that other factors than the magnitude of the batch noise must play a role in the emergence of batch effects. We hypothesised that the correlation of the batch noise between different cell types in a batch was one of these factors. We tried to decorrelate the batch noise by rotating the batch plane around a random vector. However, we encountered a couple of problems in our approach of de-correlation. Firstly, it was hard to find a rotation vector that produced a measurable difference without having to rotate the batch-plane an exorbitantly large angle. Lastly, the biggest problem was the lack of a single measurement that could quantify the "correlated-ness" of the batch noise between cell types. This would have made the search for a suitable vector of rotation much easier and is a vital parameter for more detailed analysis of the influence of the correlated nature of the batch noise on clustering of cell types.

In addition to analysing the batch effects we also examined the different types of correlation patterns present in real scRNA-seq data from Brickman's lab. This analysis returned five different subpopulations of correlation patterns. Here it is especially cluster #1 and #4 which stands out. Cluster #4 exhibited a clearly positive correlated behaviour between the genes whereas the genes in cluster #1 exhibited a mix between positive and anti-correlated behaviour. We would have liked to examine the temporal aspect of the gene/gene correlation for the gene pairs in cluster #1 because we hypnotised that this type of correlation pattern might be an indicator for genes important in cellular decision making. However, this was not possible do to time constraints.

## 6.1 Outlook

From the result of our RGB-simulation presented in this thesis we have been inspired to apply our method of analysis to other interesting factors regarding batch noise and batch correction. One thing we find very intriguing was how *Scanorama* and *Harmony* performed different levels of correction for different groups of cell types. It would therefore be interesting to examine both the absolute size of cell populations as well as the size factor between them and what influence this can have on clustering and batch correction. Moreover, the examination could also be performed with more similar cell types - e.g. different shades of red. Lastly, we have also considered adding genes that do not correspond to a certain colour and are therefore not usable for the distinguishing of cell types. It could be of interest to see how these features could influence the clustering and batch correction of the data.

# Bibliography


[1] Nikolas Barkas, Viktor Petukhov, Daria Nikolaeva, Yaroslav Lozinsky, Samuel Demharter, Konstantin Khodosevich, and Peter V. Kharchenko. Wiring together large single-cell rna-seq sample collections. *bioRxiv*, 2018.

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics*, 2008(10):P10008–12, 2008.

[3] Philip Brennecke, Simon Anders, Jong Kyoung Kim, Aleksandra A Kołodziejczyk, Xiuwei Zhang, Valentina Proserpio, Bianka Baying, Vladimir Benes, Sarah A. Teichmann, John C. Marioni, and Marcus G. Heisler. Accounting for technical noise in single-cell rna-seq experiments. *Nature methods*, 10(11):1093–1095, 2013.

[4] Yoav Gilad and Orna Mizrahi-Man. A reanalysis of mouse encode comparative gene expression data. *F1000 research*, 4:121–121, 2015.

[5] Laleh Haghverdi, Aaron T. L. Lun, Michael D. Morgan, and John C. Marioni. Batch effects in single-cell rna-sequencing data are corrected by matching mutual nearest neighbors. *Nature biotechnology*, 36(5):421–427, 2018.

[6] Stephanie Hicks. Analysis of single cell rna-seq data: 2018 bioinfosummer workshop, December 2018.

[7] Brian Hie, Bryan Bryson, and Bonnie Berger. Efficient integration of heterogeneous single-cell transcriptomes using scanorama. *Nature biotechnology*, 37(6):685–691, 2019.

[8] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[9] Byungjin Hwang, Ji Hyun Lee, and Duhee Bang. Single-cell rna sequencing technologies and bioinformatics pipelines. *Experimental & molecular medicine*, 50(8):1–14, 2018.

[10] W. Evan Johnson, Cheng Li, and Ariel Rabinovic. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics*, 8(1):118–127, 04 2006.

[11] Ilya Korsunsky, Nghia Millard, Jean Fan, Kamil Slowikowski, Fan Zhang, Kevin Wei, Yuriy Baglaenko, Michael Brenner, Po-Ru Loh, and Soumya

Raychaudhuri. Fast, sensitive and accurate integration of single-cell data with harmony. *Nature methods*, 16(12):1289–1296, 2019.

[12] Echard Limpert, Werner A. Stahel, and Markus Abbt. Log-normal distributions across the sciences: Keys and clues. *BioScience*, 51(5):341–352, 2001.

[13] Shin Lin, Yiing Lin, Joseph R. Nery, Mark A. Urich, Alessandra Breschi, Carrie A. Davis, Alexander Dobin, Christopher Zaleski, Michael A. Beer, William C. Chapman, Thomas R. Gingeras, Joseph R. Ecker, and Michael P. Snyder. Comparison of the transcriptional landscapes between human and mouse tissues. *Proceedings of the National Academy of Sciences - PNAS*, 111(48):17224–17229, 2014.

[14] Malte D. Luecken, Maren Buttner, Kridsadakorn Chaichoompu, Anna Danese, Marta Interlandi, Michaela F. Mueller, Daniel C. Strobl, Luke Zappia, Martin Dugas, Maria Colome-Tatche, and Fabian J. Theis. Benchmarking atlas-level data integration in single-cell genomics. `https://www.biorxiv.org/content/early/2020/05/27/2020.05.22.111161`, may 2020.

[15] Malte D Luecken and Fabian J Theis. Current best practices in single-cell rna-seq analysis: a tutorial. *Molecular systems biology*, 15(6):e8746–n/a, 2019.

[16] Aaron Lun and Marioni Laboratory. Further mnn algorithm development. `https://github.com/MarioniLab/FurtherMNN2018`, 2019.

[17] Elizabeth Pennisi. Breakthrough of the year: Development cell by cell. `https://vis.sciencemag.org/breakthrough2018/finalists/#cell-development`, 2018.

[18] Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), Mar 2019.

[19] Brian B Tuch, John Bodeau, Asim Siddiqui, Ellen Nordman, Fuchou Tang, Xiaohui Wang, Nanlan Xu, M Azim Surani, Yangzhou Wang, Catalin Barbacioru, Clarence Lee, and Kaiqin Lao. mrna-seq whole-transcriptome analysis of a single cell. *Nature Methods*, 6(5):377–382, 2009.

[20] Po-Yuan Tung, John D. Blischak, Chiaowen Joyce Hsiao, David A. Knowles, Jonathan E. Burnett, Jonathan K. Pritchard, and Yoav Gilad. Batch effects and the effective design of single-cell gene expression studies. *Scientific reports*, 7(1):39921–39921, 2017.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[22] Allon Wagner, Aviv Regev, and Nir Yosef. Revealing the vectors of cellular identity with single-cell genomics. *Nature biotechnology*, 34(11):1145–1160, 2016.

[23] Chen Xu and Zhengchang Su. Identification of cell types from single-cell transcriptomes using a novel clustering method. *Bioinformatics (Oxford, England)*, 31(12):1974–1980, 2015.

[24] Ka Yee Yeung and Walter L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics (Oxford, England)*, 17(9):763–774, 2001.

Chapter 7

# Appendix

## 7.1 Explanation & Abbreviation of Parameters in the *RGB*-simulation

| Parameter Name | Abbreviation | Explanation |
|---|---|---|
| Average cluster width | $w_c$ | The average pairwise distance between all cells of the same batch and cell type calculated using eq. (4.7) under the conditions $m_i = m_j \wedge k_i = k_j$. |
| Batch | - | A collection of cells of different cell types that experience the same batch noise. |
| Batch noise magnitude | $n_B$ | This parameter determines the width of the uniform distribution $U(-n_B/2, n_B/2)$ from which each element in the batch noise vector $\vec{n}_B$ is drawn. |
| Batch noise vector | $\vec{n}_B$ | This parameter is the vector that are added to all the cells in the batch no matter the cell type. |
| Cluster | - | A collection of cells that have been clustered tougher by a clustering algorithm. |
| Extrinsic distance | $d_E$ | The average pairwise distance between all cells of the same cell type but different batches calculated using eq. (4.7) under the conditions $m_i = m_j \wedge k_i \neq k_j$. |
| Intrinsic distance | $d_I$ | The average pairwise distance between all cells of different cell types but of the same batch calculated using eq. (4.7) under the conditions $m_i \neq m_j \wedge k_i = k_j$. |
| Intrinsic noise | $n_I$ | This parameter determines the scale of the uniform distribution $U(0, n_I)$ from which the colour weights $(\omega_r, \omega_g, \omega_b)$ is drawn. |
| Population | - | A collection of cells that exhibits the same cell type. |
| Relative intrinsic distance | $ID$ | $\frac{d_I}{\bar{w}_c}$ |
| Relative extrinsic distance | $ED$ | $\frac{d_E}{\bar{w}_c}$ |

**Table 7.1 | Glossary list for parameters used in the *RGB*-cells simulation:** This table contains the name, abbreviation and a short explanation of the different terms and parameters used in the *RGB*-cells simulation.

## 7.2 Notation used for the Adjusted Rand Index

| Class \Cluster | $c_1$ | $c_2$ | $\cdots$ | $c_k$ | *Sums* |
|---|---|---|---|---|---|
| $u_1$ | $n_{1,1}$ | $n_{1,2}$ | $\cdots$ | $n_{1,k}$ | $n_{1.}$ |
| $u_2$ | $n_{2,1}$ | $n_{2,2}$ | $\cdots$ | $n_{1,1}$ | $n_{2.}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $n_{1,1}$ | $\vdots$ |
| $u_m$ | $n_{m,1}$ | $n_{m,2}$ | $n_{1,1}$ | $n_{m,k}$ | $n_{.m}$ |
| *Sums* | $n_{.1}$ | $n_{.2}$ | $\cdots$ | $n_{.k}$ | $n$ |

**Table 7.2 | Notation for the contingency table for comparing two partitions:** This table contains an explanation of the notation used in eq. (3.11) and (3.12).

## 7.3 Multiplicative Batch Noise

We wanted to see if we could produce similar results as described in section 4.2 using a log-normal scheme and a multiplicative noise. In order to do this we changed the second step in adding intrinsic noise form a normal distribution $\mathcal{N}(1, 0.5)$ to a log-normal distribution Lognormal$(1, 0.5)$. Moreover, the distribution from which the entries in the batch noise vector $\vec{n}_B$ was drawn was change to the uniform distribution $U(1 - n_B/2, 1 + n_B/2)$. Lastly, after the counts per million (CMP) normalisation the data was this time $log(x + 1)$-transformed. Besides this the approach was the same as in section 4.2.

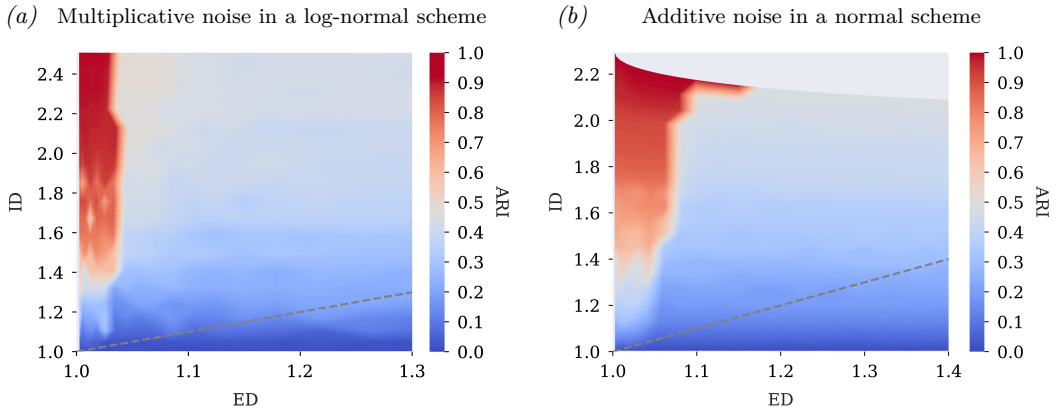


**Figure 7.1 | Maximum *ARI* as a function of *ED* and *ID* for multiplicative & additive batch noise:** *(a)* shows the maximum $ARI$ at different values for of $ED$ and $ID$ for data generated using multiplicative noise. The grey line indicates the $ID = ED$ cutoff. Above this line when $ID > ED$, cells of the same cell type but different batch (class $B$) will be closer than cells of different cell type but same batch (class $C$). *(b)* is the same plot as *(a)* but for data generated using additive noise. This is the same data presented in Section 4.2.

This analysis showed similar results for both additive and multiplicative batch noise (Figure 7.1). In both cases we observed bad clustering even when $ID$ was larger than $ED$. The hard border between regions of good clustering and bad clustering we discussed in the case of additive noise is also present for multiplicative noise

(Figure 7.1a). However, the region of good clustering for multiplicative noise is a bit narrower.

## 7.4 Rotation of the Batch Plane for Decorrelation of Batch Noise

In Section 4.2 we demonstrated that it is not only the magnitude of the batch noise that gives rise to the batch effect we see in the clustering of the cells. We hypothesise that the correlation of the batch noise can be one of the other parameters which causes this batch effect. We reason that when the batch noise is correlated between cells of different cell types the clustering algorithm will cluster cells together based on this correlation even for small magnitudes. The idea is that it will do this because the number of cells which have experienced the same batch noise is a lot larger an will therefore over shadow the effect of the otherwise more similar cells of the same type in the other batch. In order to explore this we therefor need a way to decorrelate the batch noise such that each cell type in a batch will be affected differently by the batch noise.

We hypothesise that one way to decorrelate our data without changing the internal structure of the the batch is by rotating the entire hyperplane in which the batch lies. By rotating the batch instead of varying the the batch noise vector we do not add any additional intrinsic noise to the system. A rotation will not only keep the structure inside the different cell populations but also the internal configuration of the different cell types. By changing the batch noise felt by each cell population while keeping the internal structure constant we have a way of isolating the effect the correlation will have on the clustering and identification of the cell types. This is done by defining a random axis that goes through the centre of the batch. The entire batch is then rotated around this axis at some predefined angle.

Now we have a method for trying to decorrelate the batch noise we can see if this have any effect on the clustering and identification of the cells. We construct the batches in the same manner as in section 4.2 with the same number of cells and genes. This time, however, after the batches have been shifted by the batch noise we rotated the batch around a random axis of rotation at a constant angle of $\pi/12$ *rad*. Once again, these batches was constructed for the same 20 different values of $n_B$ and $n_I$. For each combination of $n_B$ and $n_I$ the Leiden clustering algorithm was used to cluster the cells. In Figure 7.2 a contour plot of the maximum $ARI$ at different combination of $ID$ and $ED$ is shown.

The results shown in Figure 7.2a does not appear to vary significantly from the results in 4.6. This becomes evident from the contour plot of the the difference in the $ARI$ between rotated and non-rotate data ($ARI_{rotated} - ARI_{non-rotated}$) shown in Figure 7.2b. Here we see that the results obtained from data with non-rotated batch noise and data with rotated batch noise varies very little and the variation can easily be explained from random fluctuations in the simulation.

This indicates that either it was not the correlation of the batch noise that causes cells of class $A$ to cluster together with cells of class $C$ instead of class $B$ despite the shorter distance to the latter class as we hypothesised or that our method rotating the batch plane did not decorrelate the batch noise in any measurable seance. It

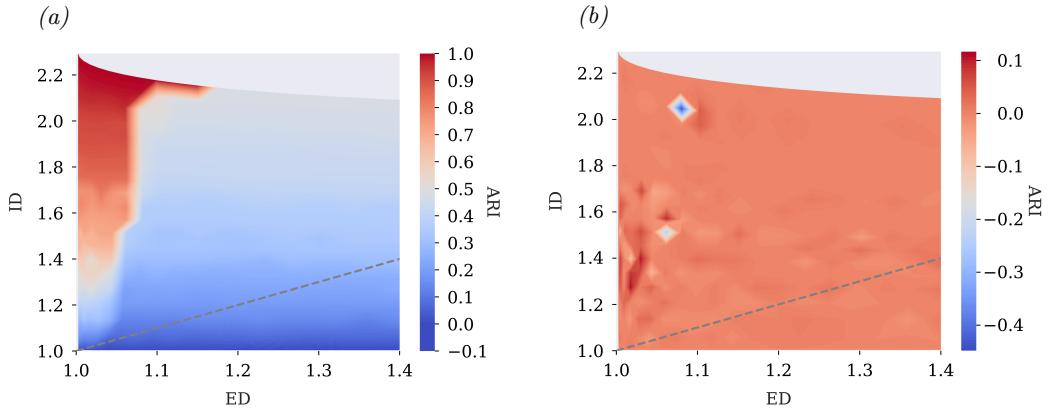


**Figure 7.2 | Maximum *ARI* as a function of *ED* and *ID* for rotated batches:** *(a)* shows the maximum $ARI$ at different values for of $ED$ and $ID$ for data rotated $\pi/12$ $rad$ around a random axis through the centre of the batch. The grey line indicates the $ID = ED$ cutoff. Above this line when $ID > ED$, cells of the same cell type but different batch (class $B$) will be closer than cells of different cell type but same batch (class $C$). *(b)* shows the difference in the $ARI$ between rotated and non-rotate data ($ARI_{rotated} - ARI_{non-rotated}$). The non-rotated data is the data presented in Section 4.2.

could very well be the case that it is hard to find axis in 300-dimensional space where a rotation of $\pi/12$ $rad$ provides a measurable level of decorrelation. We will need a measurement that could determine at what level the batch noise is correlated between cell types in order to answer these questions. This would have made the search for a suitable axis of rotation much easier and is a vital parameter for more detailed analysis of the influence of the correlated nature of the batch noise on clustering of cell types.

## 7.5 Five Random Gene/Gene Correlations for the Five Determined Clusters

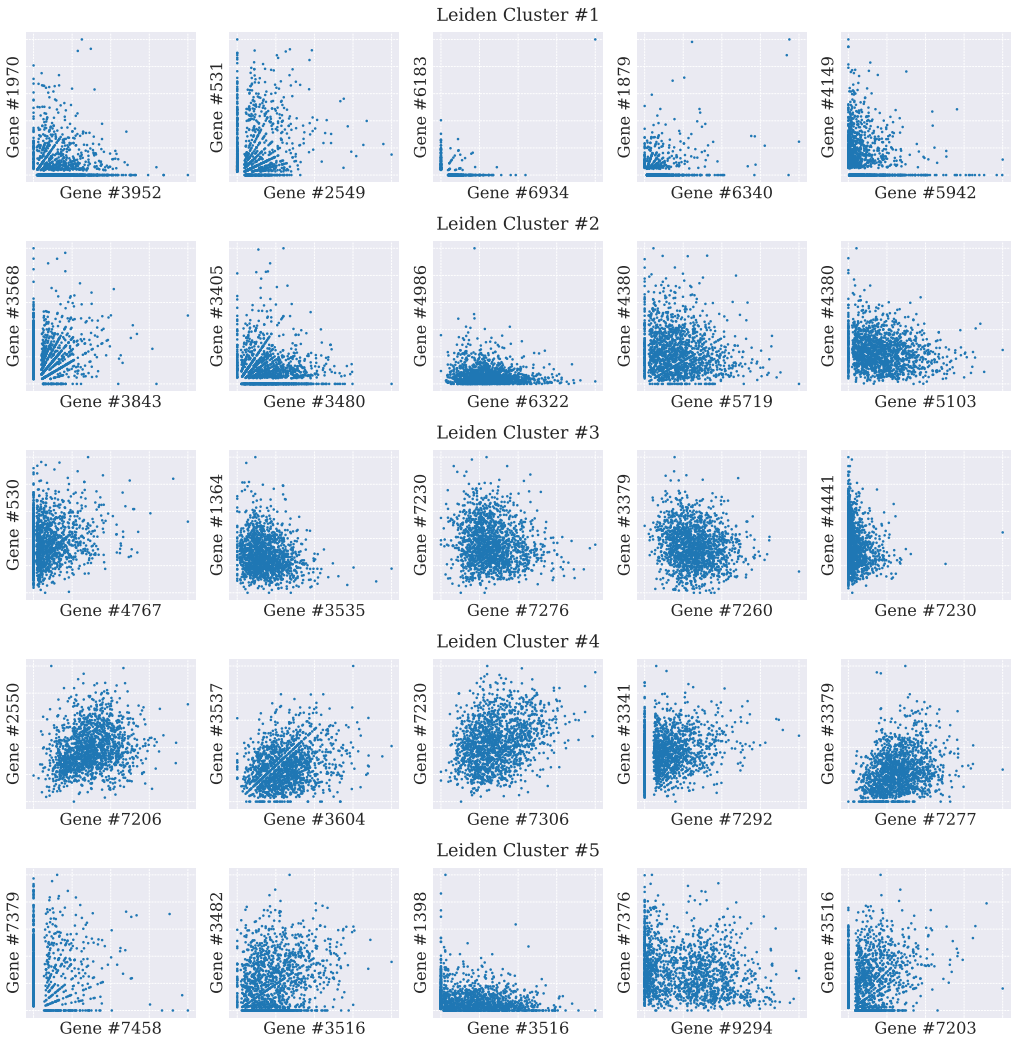


**Figure 7.3 | Five random gene/gene pairs for each cluster:** This figure illustrates five random gene/gene pairs from each of the five clusters generated by the Leiden algorithm. The orientation of the gene/gene pair have been been determined by the smallest distance (using eq. (5.5)).