Identifying transcriptional signatures of cell-stemness

Rachel Jackson

Year 4 Project
School of Mathematics
University of Edinburgh
April 2021

Abstract

This project aims to characterise the transcriptional programs driving cell-stemness by applying Non-negative Matrix Factorisation to large single-cell RNA-sequencing (scRNA-seq) datasets. This report will describe a cross-validation pipeline to determine an optimal number of factors for matrix factorisation, before validating this approach using simulated single cell RNA-sequencing data. This cross-validation pipeline will then be applied to a real single cell RNA-sequencing dataset, namely the mouse embryonic retinal dataset from Lo Giudice et al. (2020). Finally, we will discuss potential improvements to this approach in order to find the elusive cell-stemness transcriptional signature.

This project report is submitted in partial fulfilment of the requirements for the degree of BSc Mathematics and Biology.

I would like to thank my supervisors Dr Andrew Papanastasiou, Dr Catalina Vallejos and Dr Natalia Bochkina for all of their help and support throughout this project.

Contents

Chapter 1

Introduction

Single cell RNA-sequencing (scRNA-seq) is a genomic approach used for the quantitative analysis of RNA (ribonucleic acid) molecules from individual cells in a biological sample [39]. RNA is a polymeric molecule which has essential functions in coding, decoding, regulation and expression of genes. RNA is made up of the ribonucleotide bases adenine (A), cytosine (C), guanine (G), and uracil (U). RNA has a wide range of functions in the cell:

- messenger RNA (mRNA) is transcribed from genes and is used as the blueprint for the translation of proteins
- transfer RNA (tRNA) brings specific amino acids to the ribosome, which are then matched up to the mRNA blueprints to synthesise proteins.
- ribosomal RNA (rRNA) is used as the building blocks of ribosomes.
- other types of RNA within cells (e.g. small RNA, micro RNA and small nuclear RNA) have diverse functions [6].

scRNA-seq quantifies the mRNA present for each of the genes expressed in a cell. scRNA-seq has a broad range of applications, with uses emerging in oncology [28], infectious diseases [3] and in developmental biology [25]. In the context of developmental biology, a particularly important biological process is cellular differentiation. Differentiation is the process by which cells become increasingly specialised from multipotent stem cells via expression of genes specific to that cell type [34]. Cell stemness defines how much a cell is alike a stem cell, i.e. it is a measure of the potential a cell has to differentiate into a finite set of cell types. In developmental biology, scRNA-seq can be used to predict differentiation trajectories based on which genes are expressed in these cells.

During this project, scRNA-seq data will be utilised in order to characterise the transcriptional programmes driving stemness and differentiation in retinal cells. In particular, we wish to utilise Non-negative Matrix Factorisation (NMF) to identify the transcriptional programmes defining these processes.

1.1 Overview of project aims

The aims of this project are:

- To implement an NMF (Non-negative matrix factorisation) and cross-validation analysis pipeline to tune input NMF parameters.
- To evaluate the proposed analysis pipeline in terms of its ability to resolve underlying structure of simulated synthetic scRNA-seq data.
- Analysis of real scRNA-seq data from a mouse embryonic retinal dataset utilising NMF and cross-validation techniques.
- To interpret the factors found by NMF. We hypothesise that one of these factors may represent a gene expression programme which defines stemness. This factor would have high expression in stem cells and low expression in differentiated cells.

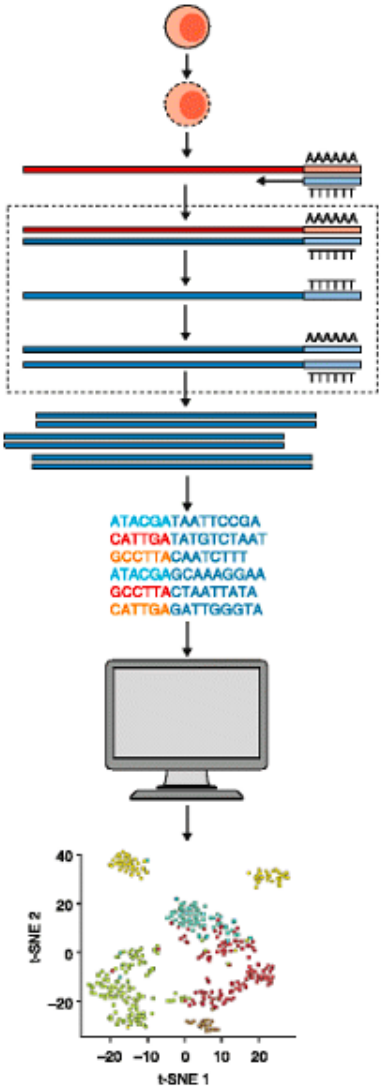
1.2 Overview of single-cell RNA-sequencing

Multiple scRNA-seq protocols have been proposed [12]. One of the most popular protocols is the one implemented by the 10× Genomics platform which is commercially available [13]. In 10× Genomics assays, individual cells are isolated using microfluidic technology in a continuous oil phase.

scRNA-seq library generation requires several steps. Firstly, cells are lysed (i.e. the cell membrane is broken down). mRNA is reverse transcribed into first strand cDNA (complementary DNA). Second strand cDNA is synthesised and cDNA amplification is carried out to create the final cDNA library via Polymerase Chain Reaction (PCR). PCR is the process by which segments of DNA are amplified. The DNA strands are separated by heating to 90°C. The temperature is then lowered to 55°C for primers to anneal. The strand is then extended by DNA polymerase when the temperature is increased to 72°C [46]. Repeated rounds of this reaction amplifies the DNA segment at rate 2^n, where n is the number of rounds of amplification.

The reverse transcription step incorporates a Unique Molecular Identifier (UMI) which identifies the individual mRNA transcript, and a barcode which identifies the cell from which the transcript originated. This allows assignment of each read to its original cell after cDNA library amplification [13]. This process is shown in Figure 1.1.

From this process, we can determine the number of copies of each gene in each cell and so construct a genes × cells expression matrix. Each entry in this matrix will give the number of mRNA copies transcribed from each gene in an individual cell. This is high dimensional count data (thousands of genes in thousands or even millions of cells), and a wide range of approaches have been proposed to characterise cell-to-cell heterogeneity using scRNA-seq data [54]. An example of different underlying types of heterogeneity are shown in Figure 1.2.



1. Isolate single cells from a tissue sample. Lyse the cell membrane in a way that preserves cellular mRNA

2. mRNA is converted to cDNA using **reverse transcription**. During this step a UMI which identifies the transcript and a barcode which identifies the cell from which the transcript originated from are incorporated

3. Amplify cDNA using **PCR**, then pool all the cDNA libraries together

4. Pool cDNA sequencing libraries and sequence using Next Generation Sequencing

5. Obtain counts matrix which contains the expression level of each gene in each cell. Quality control and downstream processing of the dataset can then be carried out (including dimensionality reduction such as t-SNE plots)

Figure 1.1: An overview of the scRNA-seq workflow (Figure adapted from [12]).

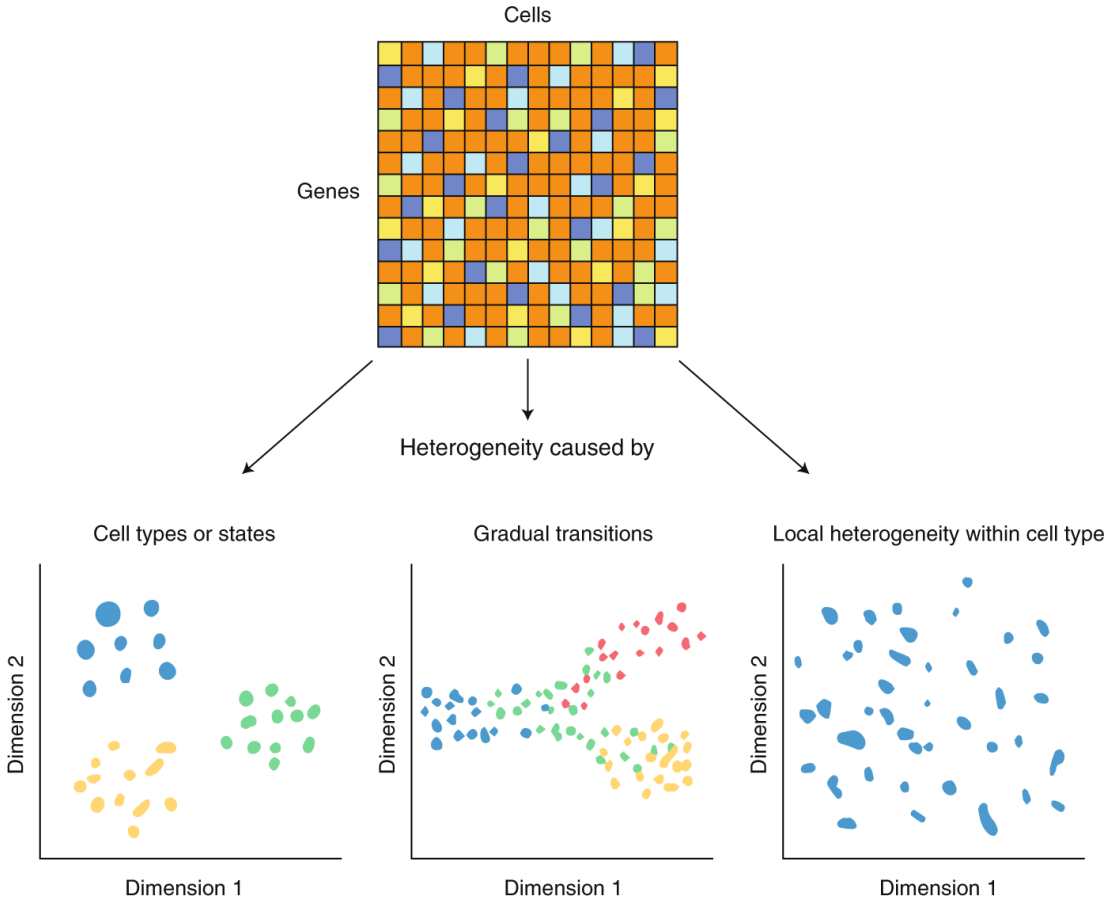


Figure 1.2: The genes × cells matrix obtained from single cell RNA-sequencing is highly dimensional. Dimensionality reduction techniques can be used on this data to visualise the underlying biological heterogeneities of this dataset. Such biological structure can include distinct cell types (left) and gradual transitions through a process, such as the cell division cycle or cell differentiation (middle). Locally, within a seemingly homogeneous population, cell-to-cell heterogeneity may be due to intrinsic or extrinsic sources of noise (right) (Figure taken from [43]).

1.3 Overview of the report

In subsequent chapters, this report will firstly introduce the Lo Giudice et al. (2020) dataset and discuss the initial processing of the dataset. Methods utilised in the project are described in Chapter 3. In particular, Non-Negative Matrix Factorisation and the cross-validation pipeline are described. To validate this cross-validation approach, we utilise simulated scRNA-seq data (the generation of which is described in Chapter 4). The results of NMF and cross-validation on this simulated data are shown in Chapter 4. Finally, in Chapter 5 we utilise this cross-validation pipeline on the Lo Giudice et al. (2020) dataset to obtain the optimal number of factors to use for NMF. NMF is then applied to this dataset to visualise the underlying biology. Potential improvements to this method are discussed in Chapter 6.

Chapter 2

Exploratory Analysis

The data used in this project was introduced in Lo Giudice et al. (2020). In their paper, Lo Giudice et al. describes the scRNA-seq analysis of the embryonic mouse retinal cells. This dataset contains 6067 retinal single-cell transcriptomes that show step wise progression of gene expression during the early stages of differentiation in retinal cells. A large diversity of cell states are present in the retina; for example, retinal ganglion cells (RGCs), photoreceptor cells and amacrine cells (as shown in Figure 2.1). The Lo Giudice et al. (2020) dataset contains a trajectory of cells from undifferentiated early progenitor cells to differentiated cells such as Retinal Ganglion Cells (RGCs) [25].

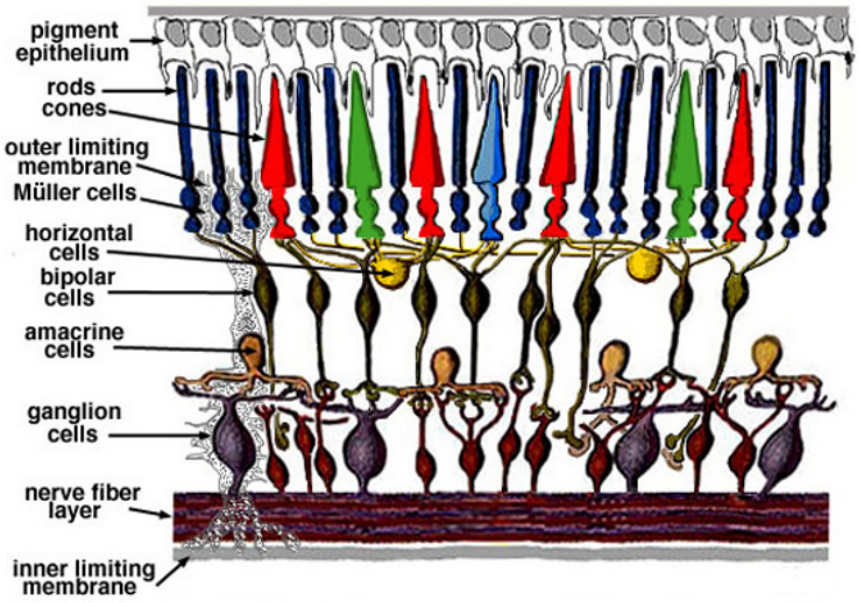


Figure 2.1: Diversity of cells in the retina (Figure extracted from [20]).

2.1 The Lo Giudice et al. (2020) dataset

Mice aged 12-30 weeks were impregnated simultaneously. These mice were ethically sacrificed at E15.5 (the stage of the embryo 15 days after vaginal plug formation). Thirty retinas were extracted in ice-cold L-15 medium, microdissected and incubated in 200μl of cell dissociation solution consisting of papain (1mg/ml)-enriched HBSS (Hanks' Balanced Salt Solution) at 37°C for 12 minutes with titration every 2 minutes. The cells were further dissociated by up-and-down pipetting. The reaction was then stopped by addition of 400μl of ovalbumin(2mg/ml)-enriched HBSS and the cell suspension was passed through a cell strainer to remove cell aggregates. Centrifugation of the resulting solution was carried out for 5 minutes at 500g at 4°C. The pellet was then resuspended in cold HBSS, and the solution was ensured to have a concentration of cells of 410cells/μl. scRNA-seq was carried out according to the 10× Genomics protocol. Two lanes of results were obtained since two channels were used during the scRNA-seq processing [25].

2.2 Initial processing of the data

Before any analysis of the dataset could be carried out, it must be ensured that only high quality data is used in the analysis. To ensure only high quality data is present, we perform quality control (e.g. to remove outlier cells). Moreover, the data is normalised to remove technical artefacts.

2.2.1 Quality control

After the generation of a genes × cell transcript counts matrix, quality control (QC) is carried out on the data to ensure only high quality cells are retained. Firstly, outlier cells are detected based on adaptive thresholds for library size (total sum of counts across all genes for each cell), number of genes expressed and the percentage of mitochondrial genes expressed. Mitochondrial RNA is contained in a separate organelle, so if the mitochondrial gene percentage is high, this indicates that the cells may have been damaged, resulting in cytoplasmic leakage [17]. Large library sizes may indicate that due to technical error two cells have been labelled as the same cell (a doublet). These outliers are then discarded.

2.2.2 Normalisation

Normalisation of data aims to remove differences in sequencing coverage (the number of unique reads that include a given nucleotide in the reconstructed sequence) between cells so that they do not interfere with comparisons of expression profiles between cells. These differences often arise due to technical difficulties in cDNA capture or PCR amplification efficiency across cells. Following standard scRNA-seq analysis pipelines [1], the data is normalised for each gene i in cell j according to the formula:

$$\tilde{Y}_{ij} = \frac{Y_{ij}}{\hat{s}_j},$$

where Y_{ij} is the expression count of gene i in cell j and $\hat{s_j}$ is the cell-specific size factor calculated using using computeSumFactors() in the *scran* R package [26].

2.3 Exploratory analysis

Dimensionality reduction techniques were used in order to visualise the highly dimensional scRNA-seq data. In order to give some clarity to what is visualised, cell types were assigned to each cell in the data set using CellAssign [55].

2.3.1 Assigning cell types

CellAssign

CellAssign is an algorithm designed to assign single cells to a cell type by probabilistic assignment, using the expression of marker genes in each cell. Let $\mathbf{Y}$ be a cell $\times$ gene transcript count expression matrix for J cells I genes. In this population of cells assume that we have C cell types, each defined by the higher expression of specific marker genes. The relationship between cell types and marker genes is encoded through a binary matrix ρ, where $\rho_{ic} = 1$ if gene i is a marker gene for cell type c and 0 otherwise. CellAssign introduces the categorical indicator vector $\mathbf{z} = \{z_1, ..., z_j\}$, where $z_j = c$ if cell j is of type c. Let s_j be the size factor for cell j and $\mathbf{X}$ be a $P \times N$ matrix of P covariates (such as sample of origin). Let y_{ji} denote the counts of each gene i in each cell j; $y_{ji}|z_j = c \sim \mathcal{NB}(\mu_{jic}, \phi_{jic})$, where $\mathcal{NB}$ is the negative binomial distribution. μ_{jic} represents the mean expression and ϕ_{jic} is a μ-specific dispersion [55]. ϕ_{jic} is defined as a sum of radial basis functions dependent on the modelled mean μ_{jic} as proposed in Eling et al., 2019 [9]. Then the model is:

$$\mathbb{E}[y_{ji}|z_j = c] = \mu_{jic},$$

where

$$\log \mu_{jic} = logs_j + \delta_{ic}\rho_{ic} + \beta_{i0} + \sum_{p=1}^{P} \beta_{ip}x_{pj},$$

with the constraint that $\delta_{ic} > 0$.

δ_{ic} corresponds to the average log-fold change that gene i is overexpressed in cell type c. β_{i0} represents the base-line expression of gene i, while β_{ip} represent the expression for the other covariates.

To assign cells to cell types, CellAssign infers the probability that each cell is of a given type, i.e. we compute $p(z_j = c|\mathbf{Y}, \hat{\mathbf{\Theta}})$, where $\hat{\mathbf{\Theta}}$ is the estimates of the model parameters [55].

In our analysis, CellAssign was applied using marker genes for each cell type from Clark et al. (2019) [7].

2.3.2 Dimensionality Reduction Visualisations

Two dimensionality reduction techniques were used to visualise this data set:

Uniform Manifold Approximation and Projection (UMAP) was used to visualise the dataset. UMAP is a dimensionality reduction technique that can be used for visualisation of datasets [27]. The UMAP algorithm is founded on three assumptions about the data:

1. The data is uniformly distributed on a Riemannian manifold;
2. The Riemannian metric is locally constant (or can be approximated as such);
3. The manifold is locally connected.

From these assumptions it is possible to model the manifold with a topological structure. The embedding is found by searching for a low dimensional projection of the data that has the closest possible equivalent topological structure. The full details of the UMAP algorithm are found in McInnes and Healy (2018) [27].

Another dimensionality reduction technique is **t-Distributed Stochastic Neighbour Embedding** (t-SNE). t-SNE minimizes the Kullback-Leibler divergence between the joint probabilities p_{ij} in the high dimensional space and the joint probabilities q_{ij} in the low-dimensional space.

- The values of p_{ij} are defined to be symmetrised conditional probabilities, such that:
$$p_{ij} := \frac{p_{j|i} + p_{i|j}}{2n},$$
where n is the number of datapoints and $p_{j|i}$ is
$$p_{j|i} = \frac{\exp\{(-||x_i - x_j||^2/2\sigma_i^2\}}{\sum_{k\neq i}\exp\{(-||x_i - x_k||^2/2\sigma_i^2\}},$$
where σ_i^2 is the variance of the Gaussian that is centred at x_i.
The similarity of datapoint x_i to datapoint x_j $(j \neq i)$ is $p_{j|i}$, the conditional probability that datapoint x_i would pick datapoint x_j as its neighbour if neighbours were picked in proportion to their probability density under a Gaussian distribution centred at x_i.

- The q_{ij} are defined by a Student t-distribution with one degree of freedom:
$$q_{ij} = \frac{(1 + ||y_i - y_j||)^{-1}}{\sum_{k\neq l}(1 + ||y_i - y_l||)^{-1}},$$
where the y_i and y_j are the low-dimensional counterparts of x_i and x_j.

- The Kullback-Leibler divergence between the two joint probability distributions P and Q is given by:
$$C = \sum_i \sum_j p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \text{ [44].}$$

If the map points y_i and y_j correctly model the similarity between the high-dimensional data x_i and x_j, then p_{ij} and q_{ij} will be equal. Therefore, minimising the Kullback-Leibler divergence of P and Q gives us a low-dimensional representation of the high-dimensional data.

The UMAP and t-SNE representations for the Lo Giudice dataset are shown below in Figure 2.2 and Figure 2.3 respectively. Since no clear clustering is shown to occur, matrix factorisation may be helpful to elucidate the underlying structure of this dataset. Both plots show an almost symmetrical split which corresponds to the two technical replicates contained within the LoGiudice dataset. Cells on both plots were labelled using CellAssign.

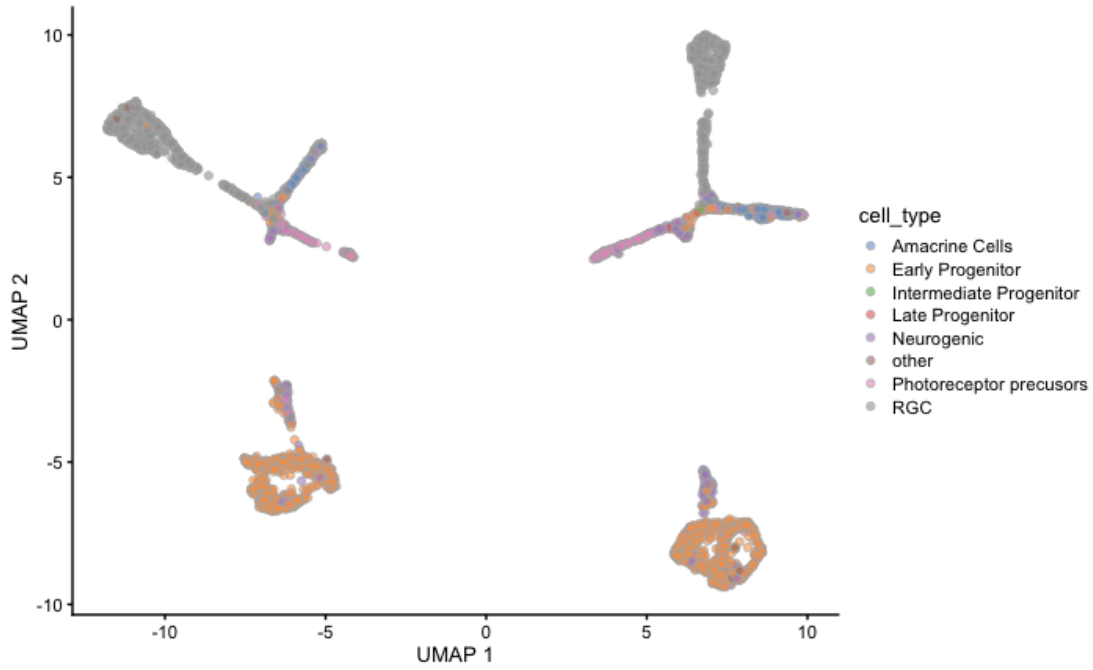


Figure 2.2: UMAP for the LoGiudice dataset for both Lanes 1 and 2.

In Figure 2.4 we compare CellAssign cell labels with the expression of key marker genes for each cell type. For all of the marker genes, the highest level of expression is at the cells where CellAssign has assigned the cell type corresponding to that marker gene. Overall, this suggests that CellAssign is successfully identifying each cell type, however this is reliant on the correct identification of marker genes in Clark et al. (2019).

From this exploratory analysis, we can now understand in which cells we expect to obtain stemness scores corresponding to stem cells. In the t-SNE plot this will be in the regions where Early progenitors were identified. These plots provide guidance as to which cells we can expect to identify as stem cell-like using the methods described in later chapters. For simplicity, subsequent analyses will only consider one of the technical replicates.

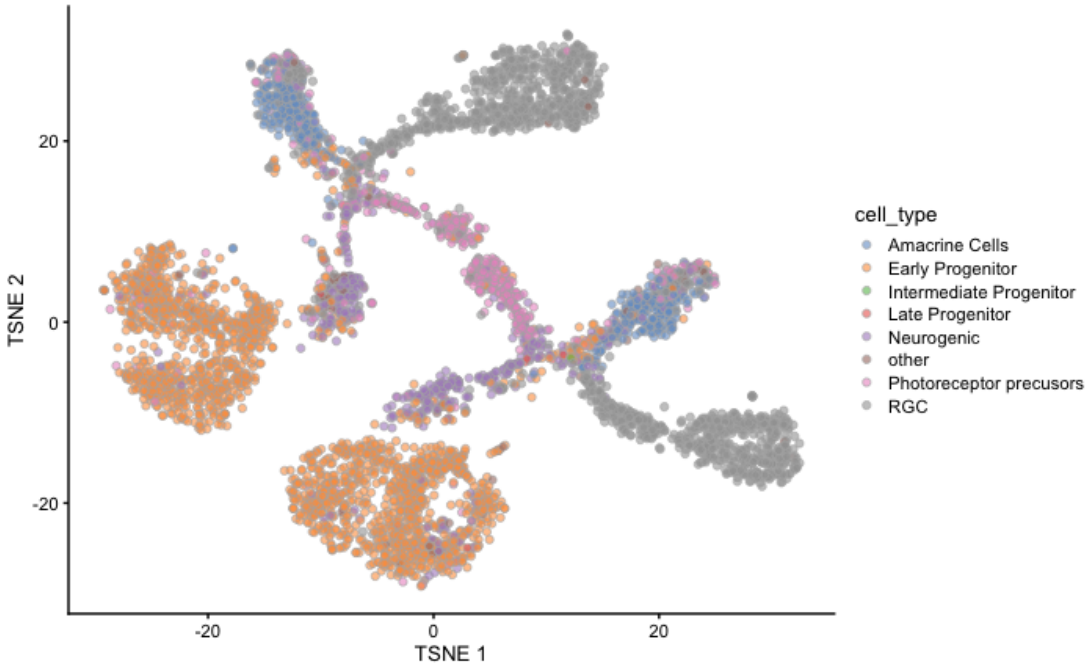


Figure 2.3: t-SNE for the LoGiudice dataset for both Lanes 1 and 2

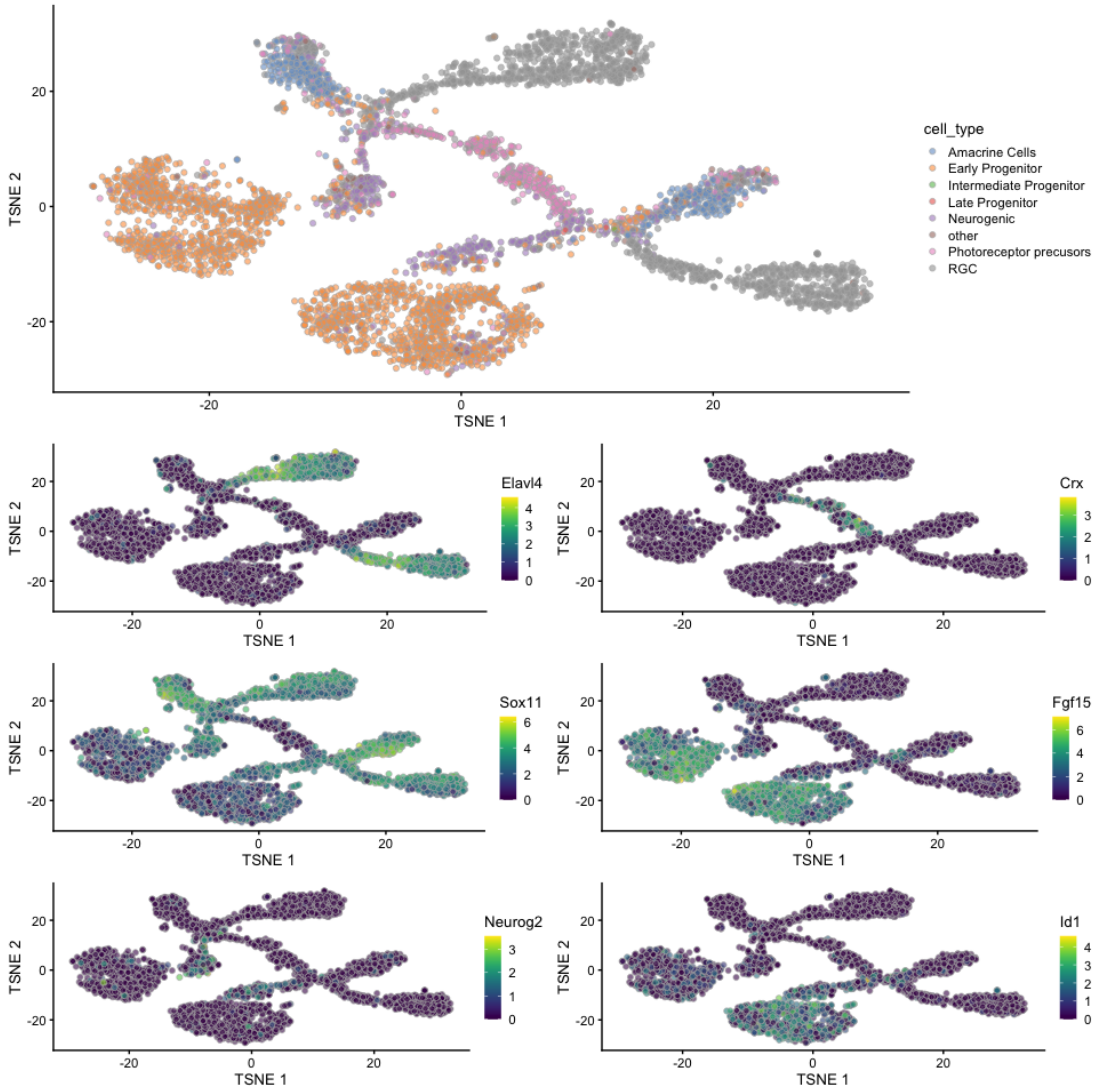


Figure 2.4: Validation of CellAssign, comparing gene expression of key marker genes with how the cells are labelled by CellAssign. The top panel shows the t-SNE plot labelled using CellAssign. Beneath we have plots corresponding to the following marker genes: *Elav14* is a marker gene for Retinal Ganglion Cells (RGC cells); *Crx* is a Photoreceptor precursor marker; *Sox11* is an Amacrine cell marker; *Fgf15* is an Early Progenitor cell marker; *Neurog2* is a Neurogenic cell marker; *Id1* is a Late progenitor marker.

Chapter 3

Methods

The following sections describe methods thought to be able to be utilised to determine the differentiation status of cells. Matrix factorisation methods can reveal the underlying structure of the gene expression matrix, and therefore may be able to identify components which indicate stemness. We consider matrix factorisation methods rather than clustering methods (such as K-means clustering) as we are looking for a signature that follows a gradient from undifferentiated to differentiated which will not be well defined in distinct clusters. In particular, we focus on Non-negative Matrix Factorisation (NMF). We compare NMF both with methods which assign cell types to the cells within the dataset, and also with existing stemness scoring methods. We expect that NMF will be able to be used to differentiate between stem cells and differentiated cells.

3.1 Matrix Factorisation Methods

Matrix Factorisation (MF) methods are unsupervised techniques which reveal low-dimensional structure of high dimensional data, whilst preserving as much of the structure of the original data. In general, MF methods approximate an input data matrix $\mathbf{Y}$, as a product of two matrices, the amplitude matrix $\mathbf{U}$ and the pattern matrix $\mathbf{V}$, which are also referred to as factors. We can also write this as $\mathbf{Y} \approx \mathbf{UV}$.

3.1.1 Overview of Matrix Factorisation Methods

MF methods include principal component analysis (PCA), non-negative matrix factorisation (NMF) and independent component analysis (ICA) (PCA and ICA are described in detail in Appendix A). These methods have a diverse array of applications across many different fields. For example, ICA has been used in the following areas:

- Separation of audio signals (as described in the cocktail party problem) [22]
- Analysis of electroencephalogram (EEG) and magnetoencephalography (MEG) signals in neuroimaging [16]

- Division of multiple access communication schemes in telecommunication [16]
- Finding hidden factors in financial data [16].

The aim of this project is to explore if matrix factorisation methods can give useful insights into the underlying structure of scRNA-seq datasets, and if we can use these methods to identify factors which successfully predict cell-stemness.

3.1.2 Non-negative Matrix Factorisation

Given an $n \times m$ non-negative matrix $\mathbf{Y}$, where n is the number of cells and m is the number of genes, we wish to find non-negative matrix factors $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{Y} \approx \mathbf{UV}$. i.e. we factorise the $\mathbf{Y}$ matrix into an $n \times r$ matrix $\mathbf{U}$ and an $r \times m$ matrix $\mathbf{V}$. This is achieved by solving the following optimization problem:

$$\min_{U \in M_+^{n \times r}, V \in M_+^{r \times m}} \rho(\mathbf{Y}, \mathbf{UV}),$$

where M_+ denotes the set of non-negative matrices and ρ denotes a cost or objective function. The general formulation of NMF as an optimisation problem is given by minimising the cost function with respect to $\mathbf{U}$ and $\mathbf{V}$, subject to the constraints $\mathbf{U}, \mathbf{V} \geq 0$. r is specified to be less than m or n so that we obtain a compressed version of the original data matrix. The above approximation can be rewritten column by column as $\mathbf{y} \approx \mathbf{Uv}$, where $\mathbf{y}$ and $\mathbf{v}$ are the corresponding columns of $\mathbf{Y}$ and $\mathbf{V}$. Therefore, each data vector $\mathbf{y}$ is approximated by a linear combination of the columns of $\mathbf{U}$ weighted by the components of $\mathbf{v}$. Hence, $\mathbf{U}$ can be regarded as containing a basis for the linear approximation of the data in $\mathbf{Y}$. Since relatively few basis vectors are used to represent many data vectors, a good approximation requires the basis vectors to use structure contained in the data. We need to define a cost function ρ to quantify the quality of this approximation [23].

Cost functions A cost function can be constructed using some measure of distance between two non-negative matrices $\mathbf{A}$ and $\mathbf{B}$.

- For example, the Frobenius norm is given by the distance between $\mathbf{A}$ and $\mathbf{B}$:

  $$\rho(A, B) = ||\mathbf{A} - \mathbf{B}|| = \left(\sum_{ij} (A_{ij} - B_{ij})^2 \right)^{\frac{1}{2}}.$$

  This is lower bounded by zero and vanishes if and only if $\mathbf{A} = \mathbf{B}$.

- Another cost function is given by the Kullback-Leibler divergence (as given in the scipy.special.kl_div() function [45]):

  $$\rho(A, B) = D(\mathbf{A}||\mathbf{B}) = \sum_{ij} (A_{ij} \log \left(\frac{A_{ij}}{B_{ij}} \right) - A_{ij} + B_{ij}).$$

This cost function is again lower bounded by zero and vanishes if and only if $\mathbf{A} = \mathbf{B}$. [23].

NMF implementation in Python We use the scikit-learn NMF function implementation in Python [32]. This function was used throughout this project with either the Frobenius norm or the Kullback-Leibler Divergence as the cost function.

Optimisation does not general have a closed form solution, so we consider a numerical approach. In the implementation we use, NMF uses a coordinate descent algorithm to solve the optimisation problem [32]. Coordinate descent algorithms solve optimisation problems by successively performing approximate minimisations along coordinate directions. Each subproblem is a lower-dimensional minimisation problem and thus can be solved more easily than the full problem [51]. In general, at each iteration of the algorithm, the new value of $\mathbf{U}$ or $\mathbf{V}$ is found by adding a factor to the current value, where the factor depends on the quality of the approximation. Another implementation of NMF is consensus non-negative matrix factorisation (cNMF). Due to time constraints, this method was not used, however, it is discussed in Appendix A [21].

3.1.3 Comparison of Matrix Factorisation methods

In the table below, the matrix factorisation methods Independent Component Analysis (ICA) and Principal Component Analysis (PCA) are compared to NMF [37]. Full descriptions of PCA and ICA can be found in Appendix A.

	PCA	ICA	NMF
Components can be ranked by variation of the original data explained.	✓	✗	✗
Learns factors which are statistically independent.	✗	✓	✗
All features can be assumed to have equal weight.	✗	✓	✓
Constrains elements of U and V matrices to be greater than or equal to zero.	✗	✗	✓
Solutions vary depending on initialisation of algorithm.	✗	✓	✓

3.1.4 NMF for the analysis of scRNA-seq data

NMF learns two sets of low-dimensional representations (in each matrix factor) from high-dimensional data: one defines the relationships between cells, whilst the other defines relationships between genes. Matrix factorisation decomposes the pre-processed data matrix into two related matrices which describe its sources of variation: a cells×factors matrix $\mathbf{U}$ and a factors×genes matrix $\mathbf{V}$ (see Figure 3.1). The columns of the cells×factors matrix quantify the sources of variation between cells, whilst the rows of the factors×genes quantify the sources of variation between genes.

The values in each column of the cells×factors describe the contributions of a cell to that factor, whilst the values in each row of the factors×genes matrix

describe the relative contributions of genes to a factor. Each factor could represent a specific characteristic of a cell; for example, a cell type (e.g. Retinal Ganglion Cell) or an attribute of a subset of cells (e.g. the cell cycle stage of a cell). Transcriptional signatures - which define the genes involved in these processes- can be learnt by comparing relative weights in each row of the factors×genes matrix [37].

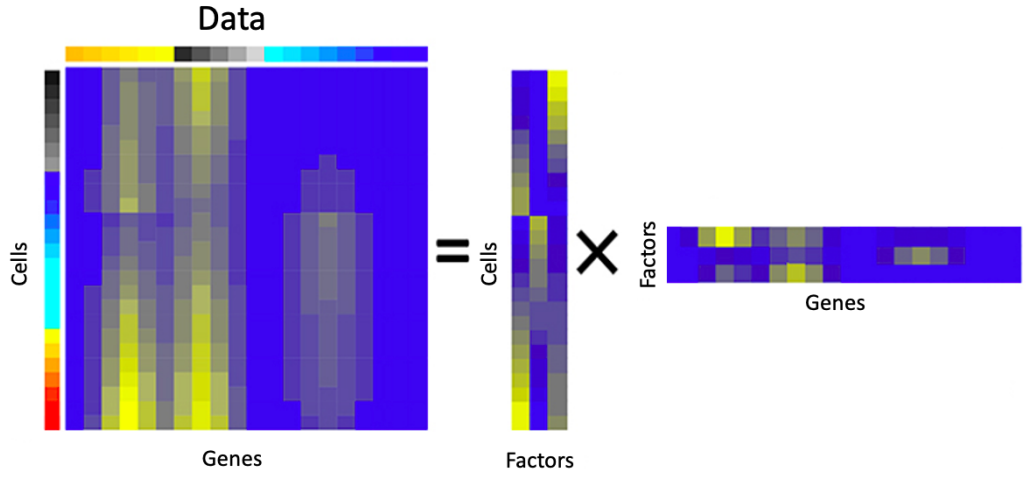


Figure 3.1: The pre-processsed input data matrix is approximated by the matrix product of the cells×factors and factors×genes matrices (the output of matrix factorisation methods). Yellow entries denote high values, whilst blue entries indicate low values (Figure adapted from [37]).

3.2 Cross-validation to determine the number of NMF factors

Cross-validation techniques are used to assess how the results of the analysis will generalise to an independent dataset. In a prediction problem, a model is usually given a dataset of known data on which training is run (a training data set), and a dataset of unknown data (test data) against which the model is tested. In this instance, we wish to determine how the model will perform to predict the structure of scRNA-seq datasets in general.

3.2.1 Cross-validation for NMF

Here we wish to use cross-validation in order to obtain the optimal number of factors for NMF. Increasing the number of factors in matrix factorisation improves the approximation, but may cause overfitting which means that the learned factorisation may fail to generalize to other datasets generated from the same biological conditions.

For matrix factorisation methods, cross-validation is more complicated than for supervised models. The primary concern is how to split the data, as leaving

out a whole row/column will lead to not all parameters being estimated.One approach that allows all parameters to be estimated is to use a speckled, random hold-out pattern, as proposed in [50]. In this approach, non-negative matrices $\mathbf{U}$ and $\mathbf{V}$ are determined via the optimization

$$\min_{U \in M_+^{n\times r},\, V \in M_+^{r\times m}} \rho(\mathbf{M} \circ \mathbf{Y}, \mathbf{M} \circ \mathbf{UV})$$

where $\mathbf{M}$ denotes a random binary masking matrix. In practice, this is non-trivial as most publicly-available NMF implementations do not easily accommodate omitting random entries from the optimization. One way to get around this is to obtain the matrices U and V via a slightly different optimization [49], namely,

$$\min_{Z \in M_+^{n\times m}, U \in M_+^{n\times r}, V \in M_+^{r\times m}} \rho(\mathbf{Z}, \mathbf{UV}),$$

where $\mathbf{Z}$ is constrained by $\mathbf{M} \circ \mathbf{Z} = \mathbf{M} \circ \mathbf{Y}$. The details of the algorithm we have implemented to perform this optimization are found in Algorithm 1 and are illustrated in Figure 3.2. The key point is that for all masked entries $\rho(\mathbf{Z}_{ij}, (\mathbf{UV})_{\mathbf{ij}}) = 0$, making this optimization equivalent to running NMF on only the non-masked entries of the matrix $\mathbf{Y}$. This allows us to quantify training and test costs, ρ_{train} and ρ_{test}, and therefore to perform a true cross-validation of the NMF approach. Use of Frobenius norm or Kullback-Leibler Divergence depends on which cost function was used for NMF. The training cost is calculated for unmasked entries (training data) whilst the test cost is calculated for mask entries (test data).

For the Frobenius norm:

$$\rho_{\text{test}} = ||(1 - \mathbf{M}) \circ (\mathbf{UV} - \mathbf{Y})||_F,$$

where $\mathbf{Y}$ is the data matrix after selection for highly variable genes and $\mathbf{M}, \mathbf{U}$ and $\mathbf{V}$ are as defined in Algorithm 1.

$$\rho_{\text{train}} = ||\mathbf{M} \circ (\mathbf{UV} - \mathbf{Z})||_F.$$

For the Kullback-Leibler Divergence:

$$\rho_{\text{test}} = \sum_{ij} (1 - M_{ij}) \circ \mathbf{W}_{ij}, \text{ where } \mathbf{W}_{ij} = \begin{cases} Y_{ij} \log\left(\frac{Y_{ij}}{(UV)_{ij}}\right) - Y_{ij} + (UV)_{ij} & \text{if } Y_{ij} > 0 \\ (UV)_{ij} & \text{if } Y_{ij} = 0 \end{cases}$$

$$\rho_{\text{train}} = \sum_{ij} M_{ij} \circ \mathbf{W}_{ij}, \text{ where } \mathbf{W}_{ij} = \begin{cases} Z_{ij} \log\left(\frac{Z_{ij}}{(UV)_{ij}}\right) - Z_{ij} + (UV)_{ij} & \text{if } Z_{ij} > 0 \\ (UV)_{ij} & \text{if } Z_{ij} = 0 \end{cases}$$

$(UV)_{ij}$ is an entry of the two non-negative matrices $\mathbf{U}$ and $\mathbf{V}$ multiplied together and thus is always positive .

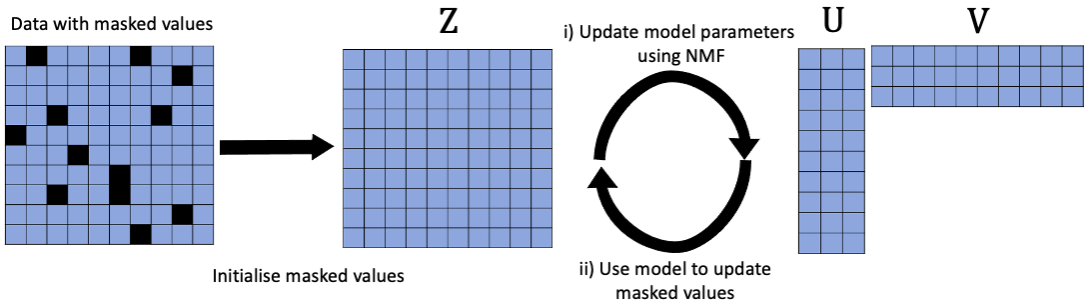


Figure 3.2: Illustration of the cross-validation by imputation procedure. Firstly, the matrix is masked by replacing the randomly selected masked values with zero, generating a matrix $\mathbf{Z}$. For a specified number of iterations, the following steps are repeated: i) The model parameters are updated by running NMF and decomposing $\mathbf{Z}$ to $\mathbf{U}$ and $\mathbf{V}$. ii) $\mathbf{U}$ and $\mathbf{V}$ multiplied together approximate $\mathbf{Z}$; therefore the masked values are updated updating by replacing the masked entries with the approximations given by NMF. After finishing this loop, a cost can be calculated which indicates how well NMF was able to approximate the matrix with the specified number of components (also referred to as factors). The idea for this approach to cross-validation comes from the Alex H. Williams blog [49], and the procedure is defined in more detail in Algorithm 1.

Algorithm 1 Cross-validation procedure

Input: $\mathbf{Y}$ = normalised counts matrix that has been filtered for the top 2000 highly variable genes, and r= number of factors

1: 30% of the matrix is randomly selected to be masked. The masking matrix $\mathbf{M}$ is defined such that:

$$M_{ij} = \begin{cases} 0 & \text{if entry is masked} \\ 1 & \text{if entry is not masked.} \end{cases}$$

2: Initialise $\mathbf{Z}^{(0)} = \mathbf{M} \circ \mathbf{Y}$, set $l = 1$ (where $\circ$ denotes the Hadamard product).

3: **for** *iteration* l **do**

Solve the optimisation:

$$(\mathbf{U}^{(l)}, \mathbf{V}^{(l)}) = \min_{U \in M_+^{n \times r}, V \in M_+^{r \times m}} \rho(\mathbf{Z}^{(l-1)}, \mathbf{UV}),$$

Update the masked values (test dataset)

$$\mathbf{Z}^{(l)} = \mathbf{Z}^{(l-1)} + (1 - \mathbf{M}) \circ (\mathbf{U}^{(l)}\mathbf{V}^{(l)})$$

Calculate the cost function for test and train datasets

$$\rho_{\text{train}}^{(l)} = \rho(\mathbf{M} \circ \mathbf{Z}, \mathbf{M} \circ \mathbf{U}^{(l)}\mathbf{V}^{(l)})$$

$$\rho_{\text{test}}^{(l)} = \rho(((1 - \mathbf{M}) \circ \mathbf{Y}), ((1 - \mathbf{M}) \circ (\mathbf{U}^{(l)}\mathbf{V}^{(l)})))$$

Set $l = l + 1$

end

4: Repeat 3: for L iterations, where L is chosen such that the training cost $\rho_{\text{train}}^{(l)}$ converges.

5: Set $\rho_{\text{train}}^{(\text{final})} = \rho_{\text{train}}^{(L-1)}$, $\rho_{\text{test}}^{(\text{final})} = \rho_{\text{train}}^{(L-1)}$

Output:$(\rho_{\text{train}}^{(\text{final})}, \rho_{\text{test}}^{(\text{final})})$

To compare different numbers of factors, this algorithm can be independently applied for a range of number of factors. At the end of each repetition of Algorithm 1, we can consider the final training and test costs. The number of factors determined is independent of the choice of random masking matrix. This is ensured by running this cross-validation procedure for several different random masking matrices, each defined by a different random seed. The number of factors at the minimum value of the test cost will give us the optimal number of factors for NMF (as shown in the plot in Figure 3.3).

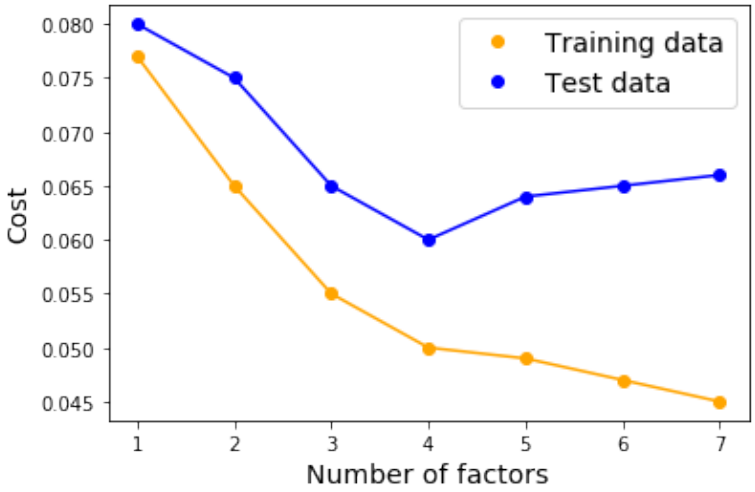


Figure 3.3: An illustration of test and training costs plotted for different numbers of factors. Here, the minimum training cost is at 4 factors, so this is considered to be the optimal number of factors to use for NMF in this example.

3.3 Determining cell differentiation status

The following section describes CytoTRACE [10], an existing method developed to determine the stemness scores of cells (i.e. the similarity of a cell to a stem cell). Alternative methods to determine cell-stemness have also been developed, for example StemID and SCENT which are described in Appendix B [11, 40]. In Chapter 5, CytoTRACE scores will be compared to the signatures obtained by NMF.

3.3.1 CytoTRACE

CytoTRACE is a method used to determine how differentiated a cell is, utilising raw scRNA-seq data. The method is based on the assumption that a less differentiated cell will express more genes. CytoTRACE scores range from 0 to 1. A low score signifies a differentiated cell whilst a high score signifies a stem cell-like cell. CytoTRACE uses single-cell gene counts and smoothing of covariant gene expression to predict ordered differentiation states.

Census Transformation

The transcript count per single cell is inferred by Census transformation [33]. Similar to the normalisation used in our analysis, this is used as a pre-processing step before CytoTRACE scores are generated for each cell. Census transformation is computed using:

$$y^*_{ij} = G_j \frac{y_{ij}}{\sum_{q=1}^{n} y_{qj}}, \forall i \in \{1, ..., n\}, \forall j \in \{1, ..., k\}$$

where $\mathbf{Y}$ is an $n \times k$ matrix with n genes and k cells. Y_{ij} is the number of transcripts assigned to gene i in cell j. The resulting gene expression matrix $\mathbf{Y}^*$ is $\log_2$-normalised after adding a pseudo-count of 1. G_j is the number of detected genes in cell j. $\mathbf{Y}' = \log_2(\mathbf{Y}^* + 1)$ is used as an input for CytoTRACE.

Gene Counts Signature

This method is based on the premise that genes for which the expression patterns correlate best with overall gene counts will best predict cell ordering by differentiation status.

Definition 3.3.1. The *Gene counts signature (GCS)* is defined as the geometric means of expressed genes that are most correlated with gene counts (the top 200 genes are used).

Smoothing covariant gene expression

After census transformation the data is still noisy, with considerable variance between cells. Assuming that transcriptionally similar cells will occupy similar differentiation states, a 2-step smoothing process is applied to improve on GCS

using nearest-neighbour graphs.

Before smoothing, the expression matrix $\mathbf{Y}'$, was filtered to a set of genes expressed in at least 5% of cells. The *dispersion index* was calculated for each gene by taking the ratio of its variance to its mean across all cells. The top 1000 most dispersed genes were selected to compute a $k \times k$ similarity matrix, D.

Let D_{ij} be the Pearson correlation coefficient between cells i and j. To convert to a Markov matrix, all diagonal elements were set to zero and the sum of each row was normalised to 1. The null similarity coefficient was calculated by using the mean of D after setting diagonal entries to zero. The 2-step smoothing method involved the following steps:

1. **Non-negative least squares regression (NNLS)** was applied to solve the optimisation problem:

$$\vec{y^*} = \arg\min_{\vec{y} \geq 0} ||A\vec{y} - G\vec{C}S||$$

   where $G\vec{C}S_R = A\vec{y^*}$.

2. **A diffusion process** is simulated and applied for 10,000 iterations or until convergence (mean($|\vec{d_{t+1}} - \vec{d_t}| \leq 10^{-6}$):

$$\vec{d_{t+1}} = 0.9 \cdot A\vec{d_t} + 0.1 \cdot \vec{d_0}$$

   where $\vec{d_0} = G\vec{C}S_R$.

The final output is a vector with k elements, containing the predicted ordering of every cell [10].

Chapter 4

Analysis of Simulated scRNA-seq data

Before applying matrix factorisation and cross-validation methods to the Lo Giudice data, we wished to verify that these methods worked as we expected. To do this we used simulated scRNA-seq data. The Splatter statistical framework described by Zappia et al. (2017) was used to simulate scRNA-seq data. This was implemented in Python as described by Kotliar et al. (2019) [21, 53].

Using NMF on simulated data where we know the ground truth allows us to identify if the method defined for cross-validation in Algorithm 1 in Chapter 3 identified the optimal number of factors. Without this step, it would be difficult to interpret the factorisation on real data with confidence.

4.1 Generation of Simulated Data

4.1.1 The Splatter simulation

The Splatter simulation described by Zappia et al. captures many features of real scRNA-seq data, including outlier genes that are highly expressed, differing library sizes between cells and sparse data [53]. scRNA-seq data is often sparse; expression is only observed for relatively few genes in each cell. The observed zero counts are due to biological and technical causes. Not all genes are expressed in each cell, and so biological zeros occur when a gene is not expressed in that cell. Technical zeros (or dropouts) occur when an expressed RNA molecule is not captured by the sequencing technology. Mean-variability trends are also observed in scRNA-seq datasets. This is where the higher the mean gene expression, the lower the variability of the expression of that gene. Genes which have low expression are found to have high variability in their expression.

The model uses parametric distributions with hyper-parameters estimated from real data. The hyper-parameters estimated from real data include outlier probability and expected library size, as illustrated in Figure 4.1.

The central concept of the Splatter simulation is the Gamma-Poisson hierarchical model where mean expression level from each gene is sampled from a Gamma distribution, and the count for each cell is then subsequently sampled

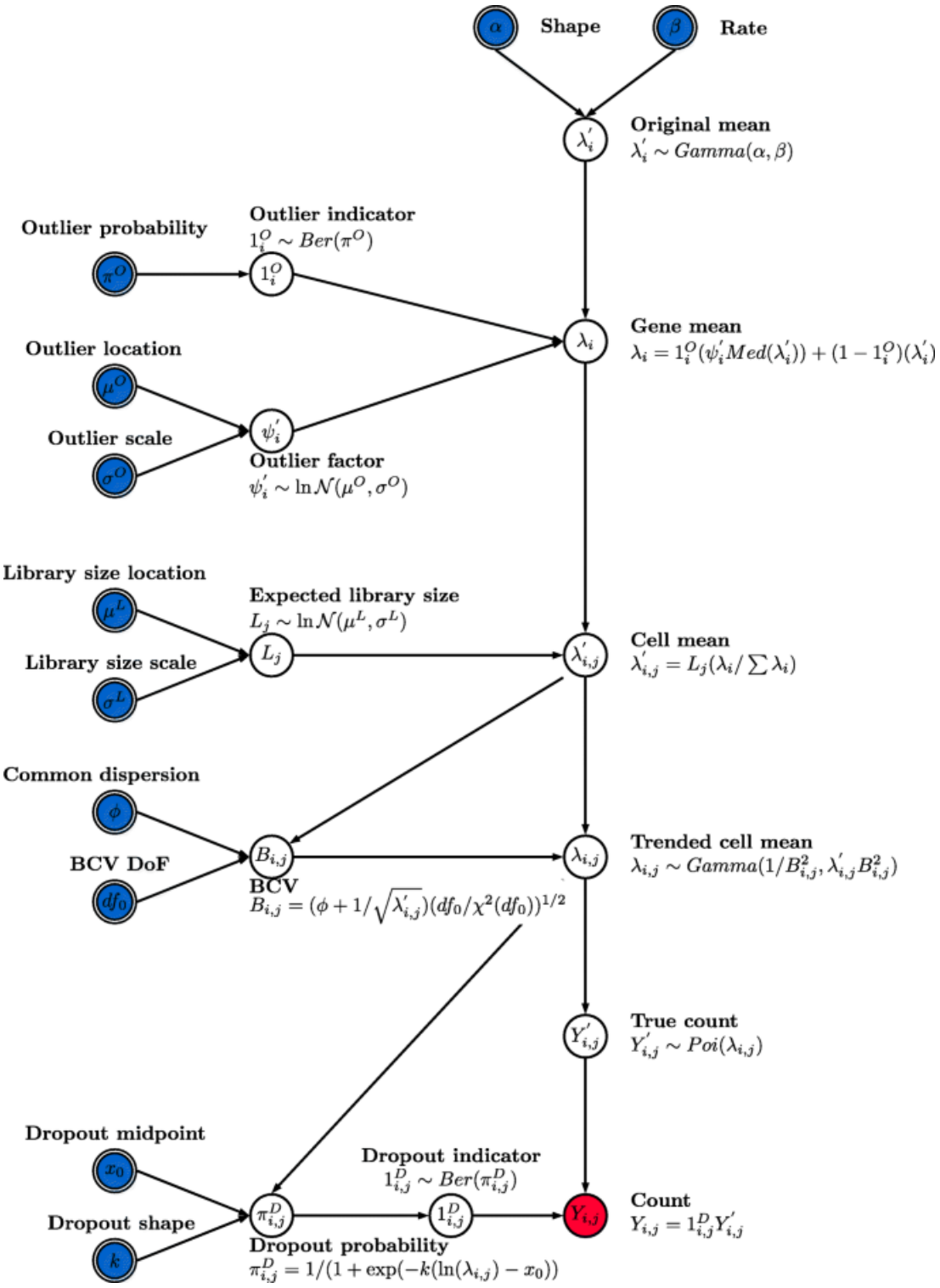


Figure 4.1: The Splatter simulation model. Blue shading indicates input parameters that can be estimated from real data. Red shading is the final output count matrix (Figure extracted from [53]).

from a Poisson distribution. Modifications include expression outliers and a mean-variability trend in the simulation.

The simulation begins by generating means from a gamma distribution $\lambda_i' \sim \Gamma(\alpha, \beta)$, where i corresponds to a gene and α and β are estimated from real data. To better capture extreme expression levels, the probability that a cell is an outlier (π^O) is incorporated. These outliers are added to the simulation by updating the simulated mean with the median of the simulated gene means multiplied by an inflation factor:

$$\lambda_i = 1_i^O(\psi_i' \text{Med}(\lambda_i')) + (1 - 1_i^O)\lambda_i'.$$

The inflation factor is sampled from a log-normal distribution $\psi_i' \sim \ln\mathcal{N}(\mu^O, \sigma^O)$. The outlier indicator 1_i^O is defined to be $1_i^O \sim \text{Bernoulli}(\pi^O)$. π^O, μ^0 and σ^0 are estimated from real data.

Library sizes are modelled using a log-normal distribution such that $L_j \sim \ln\mathcal{N}(\mu^L, \sigma^L)$, where μ^L and σ^L are estimated from real data. The simulated library sizes L_j for each cell j are used to proportionally adjust the normalised gene means for each cell such that:

$$\lambda_{ij}' = L_j \left(\frac{\lambda_i}{\sum \lambda_i} \right).$$

This allows the number of counts per cell to be altered, independent of the underlying gene expression levels specified by λ_i. The mean-variability trend is enforced by using a simulated Biological Coefficient of Variation (BCV) for each gene using an inverse chi-squared distribution:

$$B_{ij} = (\phi + 1/\sqrt{\lambda_{ij}'})(\text{df}_0/\chi^2(\text{df}_0))^{\frac{1}{2}}.$$

ϕ and df_0 are estimated from real data.

A new set of means are simulated from a Gamma distribution using the BCV values such that $\lambda_{ij} \sim \Gamma(1/B_{ij}^2, \lambda_{ij}' B_{ij}^2)$. A matrix of counts is then generated from a Poisson distribution: $Y_{ij}' \sim \text{Pois}(\lambda_{ij})$.

To simulate dropouts, elements of the count matrix are multiplied by a dropout indicator function to obtain the final output:

$$Y_{ij} = 1_{ij}^D Y_{ij}',$$

where the dropout indicator is $1_{ij}^D \sim \text{Bernoulli}(\pi_{ij}^D)$. The probability of dropout, π_{ij}^D is defined to be $\pi_{ij}^D = 1/(1 + \exp(-k(\ln(\lambda_{ij} - x_0)))$, where the dropout midpoint x_0 and shape k are estimated from real data. [53].

4.1.2 The scsim simulation

In the Python-based scsim implementation used for this project, the cells j are rows of the final count matrix and the genes i are columns (so the transpose of the Y_{ij} count matrix output for the Splatter simulation description above is obtained). Background gene expression is simulated as in Splatter and scsim is additionally

designed to allow the generation of different gene expression programmes:

- Cell identity programmes $Z_{I(j)}$ for cell j, which are equivalent to cell types. This is defined by some genes being consistently more highly expressed in a group of cells.
- Activity programme Z_a which represents a continuous cell process such as the cell cycle. This denotes a pattern of gene expression that is shared between different cell types.

Firstly a baseline mean is generated for each gene. In each gene expression programme, a subset of genes are differentially expressed as determined by the 'deloc' parameter. A 'deloc' parameter of 1.5 means that genes defining this cell type or activity programme are increased by a $\log_2$ fold change of 1.5. The Z vectors are made up of the mean gene expression parameter for each gene for that programme. The probability of a gene being differentially expressed in a programme was 2.5%. Cells were randomly assigned an identity programme with equal probability for each class such that $\mathbb{P}(\text{cell type } c) = \frac{1}{n}$, where n is the number of cell types and $c \in \{1, ..., n\}$. 30% of cells of a specified subsection of cell types were chosen to express the activity programme at a usage ρ_j, which was uniformly distributed between 10% and 70% ($\rho_j \sim \text{Uniform}(0.1, 0.7)$). The cell mean is computed in a different way than as in Splatter, and it specified as:

$$\lambda'_{ij} = L_j(\rho_j Z_a + (1 - \rho_j)Z_{I(j)}).$$

ρ_j is zero if the cells do not express the gene expression programme. L_j is the simulated library size, simulated as described above. $I(j)$ is the identity for cell j [21]. The main inputs required for scsim single cell RNA-seq data simulation are summarised in the table below:

Input	
−−seed	Simulation seed
−−numsims	Number of simulations to run
−−deloc	Differential expression location parameter
−−K	Number of identity programmes
−−nproggroups	Number of groups expressing the activity programme
−−ncells	Total number of cells
−−doubletfrac	% doublet cells (set to zero in our simulations)

Doublet cells refer to when two cells have been captured in an individual cell sample so there is double the gene product present for this gene. Any doublet cells should have been removed from our data set during quality control. Details of the other Splatter parameters used by scsim are found in Kotliar et al. [21].

Unless otherwise specified, we use seed=14, numsims=1, ncells=2500 and doubletfrac=0.0. K is equal to the number of cell types in the simulation, and nproggroups is the number of cell types where the activity programme is expressed (set to zero if no activity programme is included in the simulation). deloc is the differential expression location parameter, defined for each simulation below.

4.2 Overview of Simulated Data Analysis

After the generation of simulated scRNA-seq data, the analysis pipeline described in Figure 4.2 was applied (which emulates what will be done when analysing real data). First, the genes were filtered such that only genes that were detected in at least 0.005% of cells were kept. Before running NMF on the simulated data set, the data were first normalised using library counts (the vectors of transcript counts generated for each cell were normalised to sum to 1). A variance-stabilising transformation could then be applied. The variance-stabilising transformations considered were the $\log(\mathbf{Y}+1)$, $\sqrt{\mathbf{Y}}$ and Freeman-Tukey $\sqrt{\mathbf{Y}}+\sqrt{\mathbf{Y}+1}$ transformations [41, 30, 47]. The top 2000 most highly variable genes were then selected out of the 10000 simulated to be used for further analysis. NMF was then run using the Kullback-Leibler cost function on this 2500 cells $\times$ 2000 genes matrix, with the number of components varied to visualise the optimal number of factors. The analysis was then visualised using the plots shown in the next section, which visualise the ability of NMF to uncover the ground truth of the simulated scRNA-seq data (i.e. whether NMF can obtain factors which define cell type and activity programme gene expression patterns).

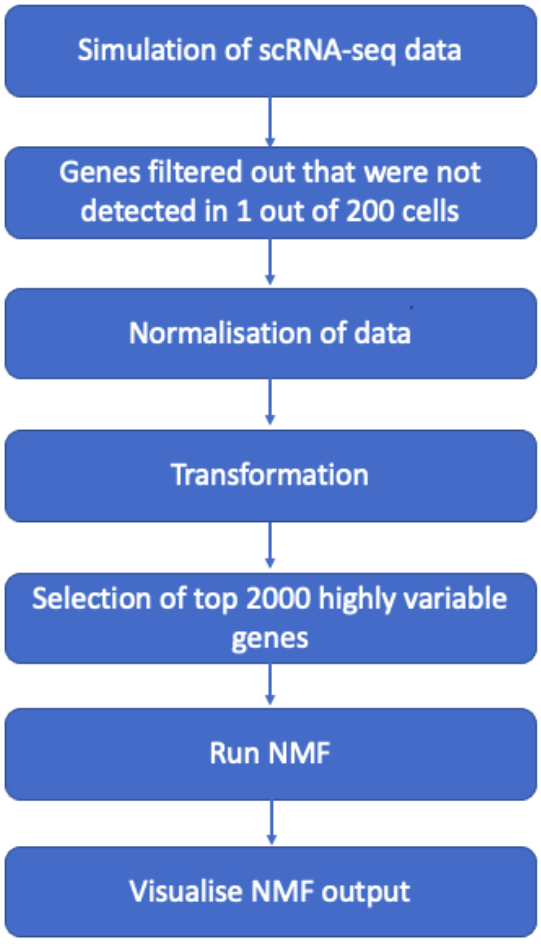


Figure 4.2: Overview of the analysis pipeline for the simulated scRNA-seq dataset.

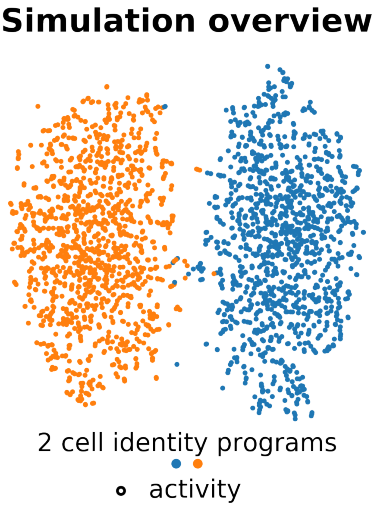
4.3 Simulation of 2 Cell Types

Here we consider the case where we have two cell identity programmes. These cell identity programmes define cell type. For example, in real data these two cell types could be Retinal Ganglion cells and Endothelial cells. The differential expression location parameter is set to 1.0.

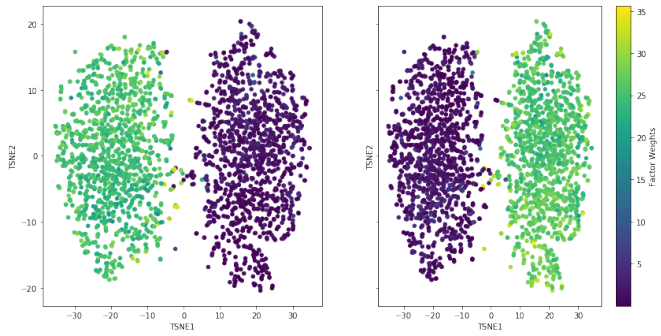
In this case we assume that the gene expression in the 2 cell types is distinct and do not have shared gene expression programmes defining activity.

NMF was run using the Kullback-Leibler cost function with 2 components, as this is the expected number of factors required when 2 cell identity programmes are present. We see that the two factors appear to represent each cell type. Overfitting occurs when NMF is run with 3 or more factors (we do not identify a clear signal from specifying additional factors).

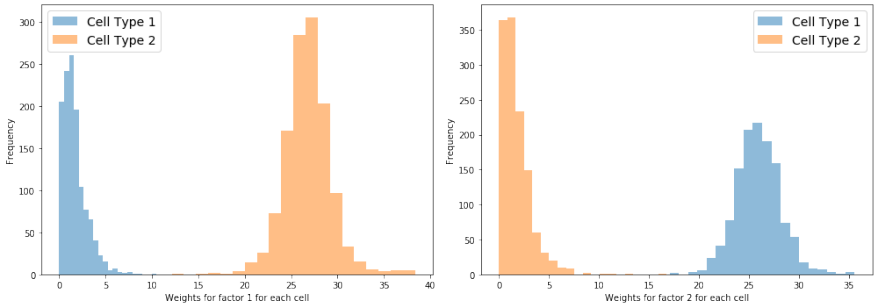
The ground truth for the 2 cell type simulation is shown in Figure 4.3a, which shows how the cell populations are split into two cell types. From Figures 4.3b and 4.3c we can clearly see that Factor 1 describes cell type 2 and Factor 2 describes cell type 1.



(a) Ground truth for the 2 cell type simulation with no activity programmes.



(b) t-SNE plot generated using scsim of the simulation of two cell types and no gene activity programme. The plots shows two cell types coloured by weights of factor 1 (on left) and factor 2 (on right) respectively.



(c) Histogram of the two cell types and no activity programme simulation. The left panel shows the highest weights for factor 1 are from cell type 2, whilst the right panel shows that the highest weights for factor 2 are from cell type 1.

Figure 4.3: Results for the 2 cell type simulation with no activity programmes.

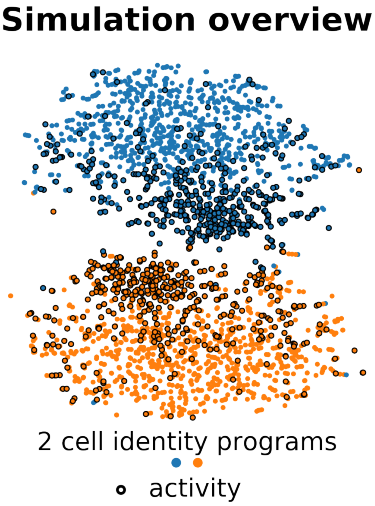
4.4 Simulation of 2 Cell Types with 1 Activity Programme

Here we consider the case where we have two cell identity programmes and an activity programme. For example this activity programme could be defining a gene expression programme involved in the cell cycle.

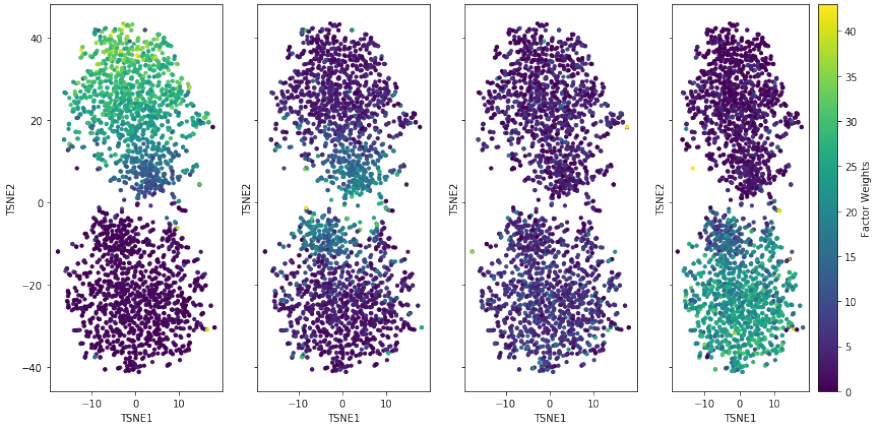
The differential expression location parameter is set to 1.0 and 1.5 where described below. Firstly consider the case where the differential expression location parameter is 1.0.

Here we obtain the ground truth in Figure 4.4a. When we run NMF with 4 components (but not less) we are able to obtain factors which explain the cell types and the activity programme (as shown in Figures 4.4b and 4.5). This additional noise component may be due to background gene expression in the simulation.

We then examine this same set up where where the differential expression location parameter is set to 1.5. The ground-truth for this simulation is shown in Figure 4.6a. Here, we obtain factors which explain the 2 cell types and activity programme using NMF with only 3 components (Figures 4.6b and 4.7). The increased 'deloc' parameter means that the genes defining the cell identity and activity programmes are more differentially expressed, and so these factors are easier to extract from the background gene expression in the simulation. This means NMF requires fewer factors to explain the underlying structure of the data.



(a) Ground-truth for the 2 cell type and 1 activity programme simulation, with differential expression location parameter 1.0.



(b) t-SNE plot generated using scsim of the simulation of two cell types and 1 gene activity programme. Factor 1 describes cell type 1; Factor 2 describes the activity gene expression programme; Factor 4 describes cell type 2. The 3rd factor captures noise in the sample.

Figure 4.4: Results for the 2 cell type and 1 activity programme simulation, with differential expression location parameter 1.0.

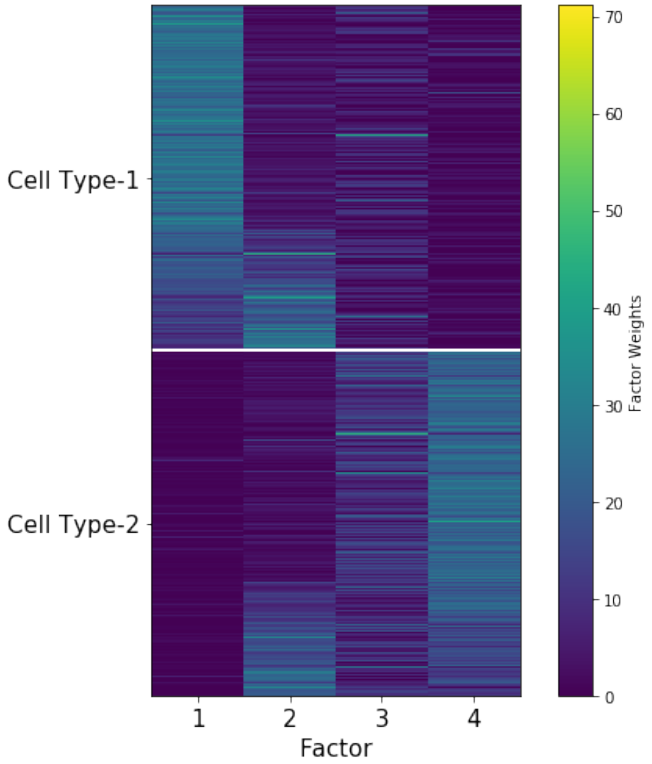
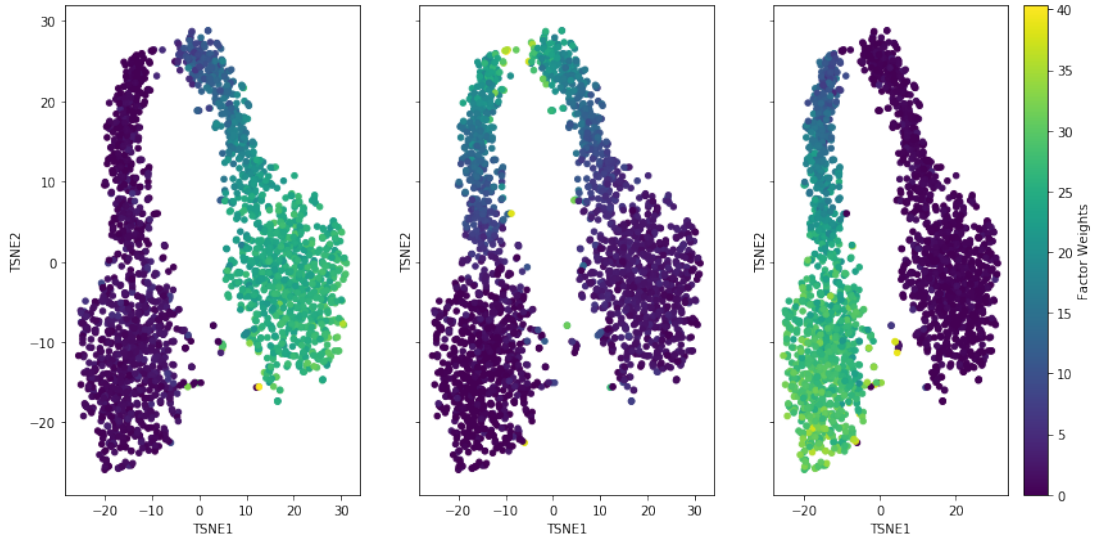


Figure 4.5: Heatmap of the cell type factors obtained by NMF with 4 factors for the 2 cell types and 1 activity programme simulation for differential expression location parameter 1.0. To aid visualisation, cells were sorted first by cell type then by activity programme usage status (on or off) before plotting the heatmap for the corresponding factor weight for each cell. Factor 2 corresponds to the activity programme, as we see it is expressed in 30% of cells in cell types 1 and 2. The 3rd factor is capturing noise present in the sample. Factors 1 and 4 correspond to cell type 1 and 2 respectively.

(a) Ground-truth for the 2 cell type and 1 activity programme simulation, with differential expression location parameter 1.5.



(b) t-SNE plot generated using scsim of the simulation of two cell types and 1 gene activity programme. Factor 1 describes one cell type; Factor 2 describes the activity gene expression programme; Factor 3 describes the other cell type.

Figure 4.6: Results for the 2 cell type and 1 activity programme simulation, with differential expression location parameter 1.5.

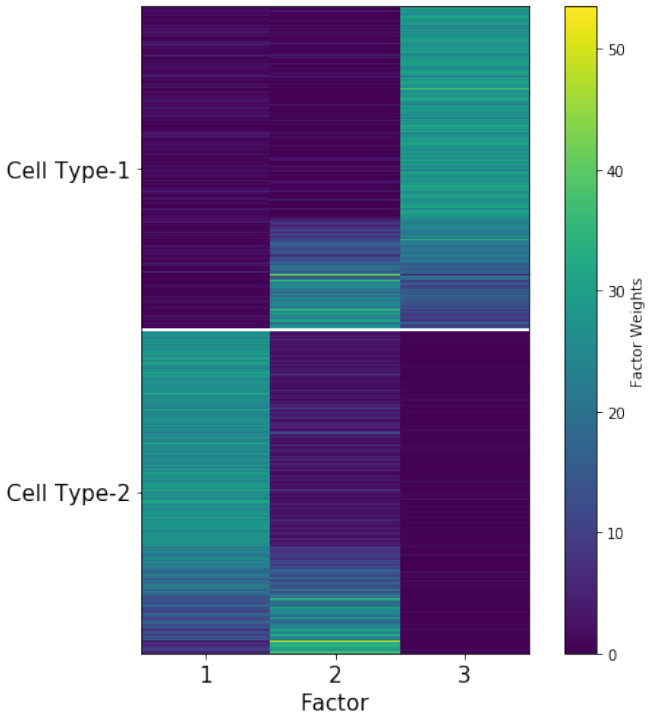


Figure 4.7: Heatmap of the cell type factors obtained by NMF with 3 factors for the 2 cell types and 1 activity programme simulation. To aid visualisation, cells were sorted first by cell type then by activity programme usage status (on or off) before plotting the heatmap for the corresponding factor weight for each cell. Factor 2 corresponds to the activity programme, as we see it is expressed in 30% of cells in cell types 1 and 2. Factors 1 and 3 correspond to cell type 2 and 1 respectively.

4.5 Simulation of 7 Cell Types with 1 Activity Programme

Here we examine a more complex simulation with 7 cell types and 1 activity programme. The gene expression programme defining the activity programme is expressed in 3 of the 7 cell types. 30% of the cells in these 3 cell types express the activity programme (As shown in Figure 4.8 which shows the ground truth for this simulation). The differential expression location parameter is set to 1.5. 7 cell types were chosen as this is also the number of cell types used by CellAssign in the Lo Giudice dataset.

The best factorisation is found when the number of components specified for NMF is 9. For 8 or less components not all of the identity programmes are well defined by the factors. For the heatmap in Figure 4.9a, we see that the cell types are each defined by a factor, and the activity programme is defined by factor 2. Factor 4 appears to capture underlying noise in the sample. This noise is likely to be due to background gene expression in the simulation.

Figure 4.9b shows a heatmap which indicates the correlation between the gene weights for each factor and the genes specifying each cell type or activity programme. Comparing to what we see in Figure 4.9a, we can see that the factors for the genes weights that specify each cell type are the same as those in the heatmap (the entries where the highest correlations occur). The activity programme is again found to be denoted by factor 2. This shows that the interpretation from the cell weights matrix does correspond to the gene weights matrix interpretation for simulated scRNA-seq data.

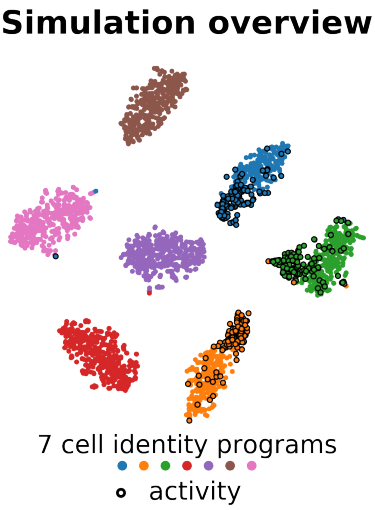
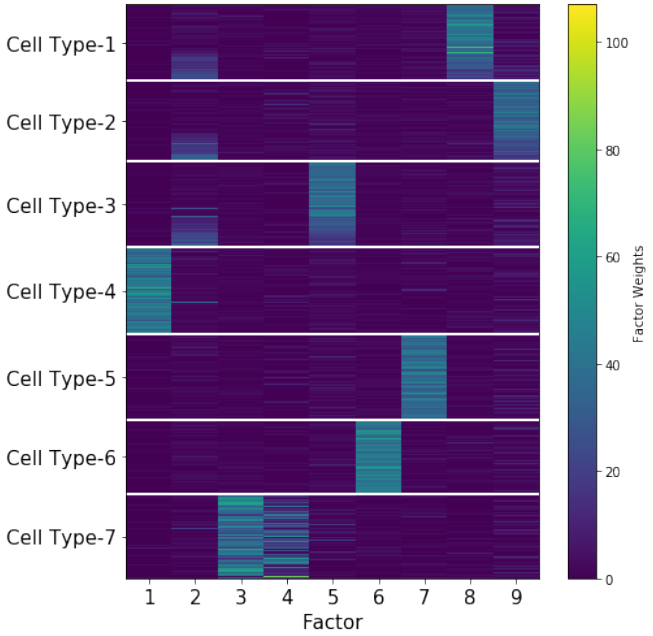
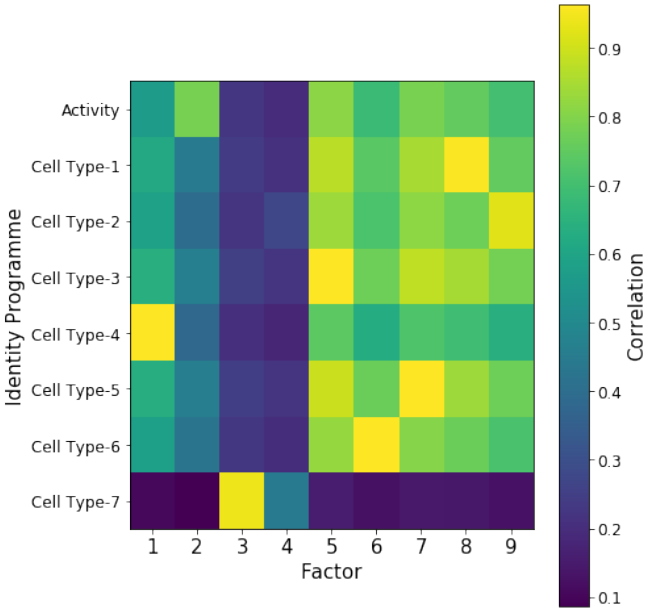


Figure 4.8: Ground-truth for the 7 cell type and 1 activity programme simulation, with differential expression location parameter 1.5.



(a) Heatmap of the cell type factors obtained by NMF with 9 factors for the 7 cell type and 1 activity programme simulation. To aid visualisation, cells were sorted first by cell type then by activity programme usage status (on or off) before plotting the heatmap for the corresponding factor weight for each cell. Factor 2 corresponds to the activity programme, as we see it is expressed in 30% of cells in cell types 1-3. The 4th factor is capturing noise present in the sample.



(b) Gene weights correlation plot: Each entry gives the Pearson correlation between the gene weights for each factor and the genes determining each activity programme/cell type.

Figure 4.9: Results for the 7 cell type and 1 activity programme simulation, with differential expression location parameter 1.5.

4.6 Results of Cross-validation on Simulated Data

So far this chapter has demonstrated that it is possible to simulate single cell RNA-sequencing data sets that have known biology (i.e. a known number of cell types and activity programmes). The remainder of this chapter will discuss how the cross-validation method described in Chapter 3 performs on these simulated data sets.

First, the cross-validation process described in Algorithm 1 in Chapter 3 was run on the normalised counts matrix obtained without applying a variance-stabilising transformation (Figure 4.10). In this figure, for the Kullback-Leibler Divergence the test cost minimum for all masking matrix random seeds is obtained at 3 factors for the 2 cell types, 1 activity program case (differential expression location parameter here is 1.5). The training cost is given by the pink line, where the training costs for all seeds have plotted directly on top of one another. For the Frobenius norm, the minimum occurs at 2 factors, which from comparing to the heatmap in Figure 4.7, we know that this is the incorrect number of factors. Figure 4.10c and 4.10d show the results for the 7 cell type, 1 activity programme simulation. From the heatmap in Figure 4.9a, we would expect a minimum at number of components/factors equal to 9. However, this is not observed in either of these plots.

After obtaining these results it was clear that the minimums of these plots were difficult to interpret in the more complex simulated cases which would be closest to the real data scenario. Therefore, 3 transformations of the data were considered. The aim of these transformations was to stabilise the variance of the data, potentially enhancing the underlying signal. By making the data more Gaussian, the Frobenius norm should act as a better cost function.

Firstly a $\log(\mathbf{Y}+1)$ transformation of the data matrix $\mathbf{Y}$ was considered. This log transformation was shown to be unhelpful, as incorrect mimimums which did not reflect the ground truth for each case were identified (Figure 4.11). This agrees with the findings of Townes et al. (2019) which reported a distortion of scRNA-seq data under log transformation [41]. For the 2 cell type, 1 activity programme case, both the Kullback-Leibler Divergence and the Frobenius norm identified minimums at 4 factors. However, when we run NMF for these log-transformed simulations and plot heatmaps, the optimal number of factors was found to be 3. For the 7 cell type, 1 activity programme case, no clear minimum was identified for the Kullback-Leibler Divergence cost function (Figure 4.11c); the Frobenius norm cost function identified a minimum at 8 factors (Figure 4.11d). Whilst this might be the number of components expected for 7 cell types + 1 activity programme, running NMF and plotting heatmaps on this transformed data reveals that 8 factors does not fully characterise the underlying ground truth.

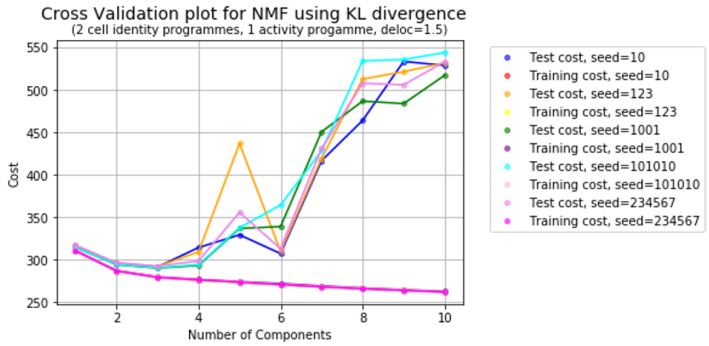
Two further transformations were considered. Firstly, a square-root transformation ($\sqrt{\mathbf{Y}}$) and also a variation on this called the Freeman-Tukey transform ($\sqrt{\mathbf{Y}}+\sqrt{\mathbf{Y}+1}$). The Freeman-Tukey transform was used by Wagner (2019) to stabilise variance on scRNA-seq data (which is Poisson distributed)[47]. Here we find that these transformations both improve the interpretability of the cross-validation plots obtained (Figure 4.12 and 4.13).

In Figure 4.12, we see that the square-root transformation correctly identifies

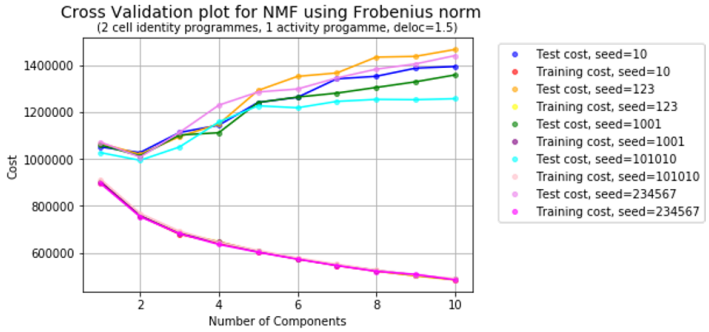
3 as the optimal number of factors for both the Kullback-Leibler Divergence and the Frobenius norm. The 7 cell-type, 1 activity programme case is less consistent between the two cost functions. The Kullback-Leibler Divergence shows two similar minimums; one at 7 and one at 9 factors. The first minimum at 7 factors is likely to be as there are 7 cell types (i.e. 7 well defined clusters of the data). The second minimum at 9 factors is likely to correspond to the number of factors required for this transformation of the data to fully visualise the groundtruth of the dataset, as was seen in the heatmap in Figure 4.7. This shows that if we obtain multiple minimums, then both need to be considered to ensure the number of factors chosen leads to a factorisation which fully explains the ground truth. The Frobenius norm is uninformative for this case as it does not show a consistent minimum for all the seeds defining the random masking matrix.

In Figure 4.13, we see that the Freeman-Tukey transformation also correctly identifies 3 as the optimal number of factors for both the Kullback-Leibler Divergence and the Frobenius norm. Again, the 7 cell-type, 1 activity programme case is less consistent between the two cost functions. The Kullback-Leibler Divergence shows two similar minimums; one at 7 and one at 9 factors. Frobenius norm is once again uninformative for this case as it fails to show a consistent minimum for all seeds defining the random masking matrix.

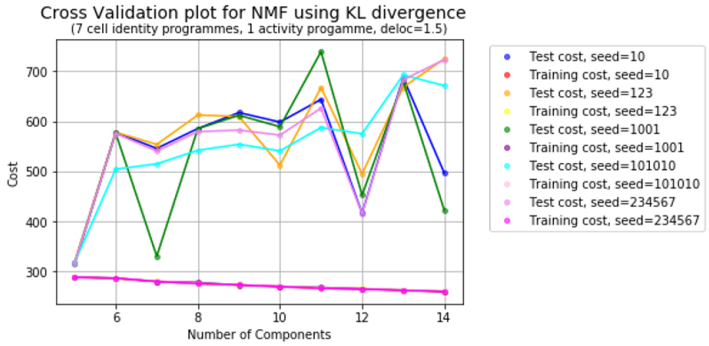
Since, Kullback-Leibler Divergence appears to give the most consistent minimums for all masking matrix random seeds, we will only consider this cost functions for our real data analysis. Comparing Figures 4.12 and 4.13, there appears to be little difference in the results for the square-root and Freeman-Tukey transformations in our simulated data. Therefore, we will consider both of these transformations when analysing the real data. Whilst both transformations look to be as good as each other in the simulations, real data is much more complex and so it is still worth considering both transformations as slight differences in the simulated data could lead to much larger differences in the real data analysis.



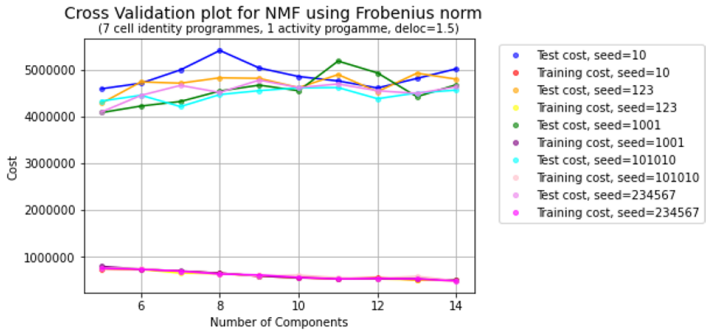
(a) KL Divergence plot for 2 cell identity programme, 1 activity programme case.



(b) Frobenius norm plot for 2 cell identity programme, 1 activity programme case.

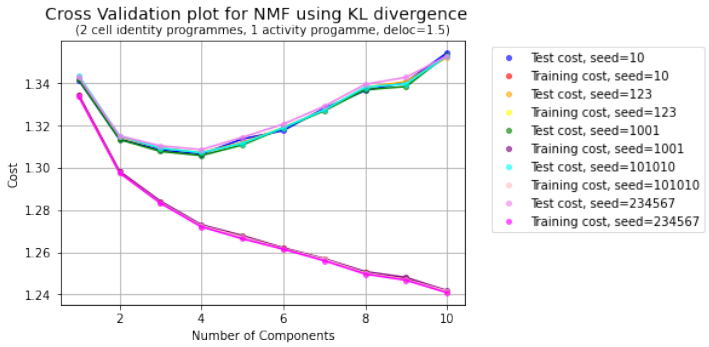


(c) KL Divergence plot for 7 cell identity programme, 1 activity programme case.

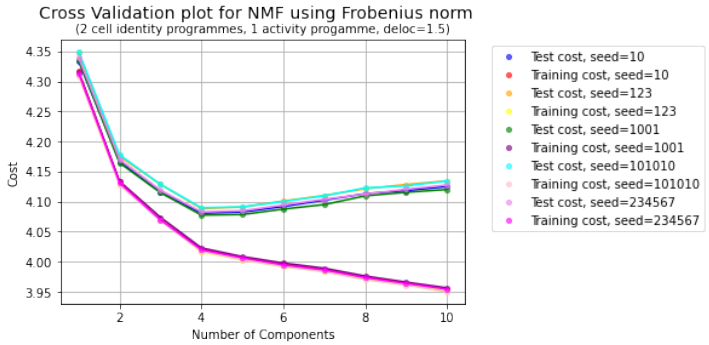


(d) Frobenius plot for 7 cell identity programme, 1 activity programme case.

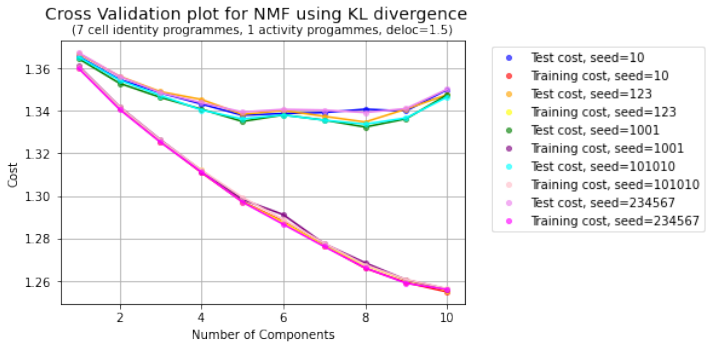
Figure 4.10: Cross-validation plots plotting both training and test costs for the non-transformed data over a range of seeds (indicated in the figure legends). In the 2 cell type, 1 activity programme case we expect a minimum at 3. In the 7 cell type, 1 activity programme case we expect the minimum to be at 8 or 9 (9 if noise is found in 1 component as seen in 4.9a).



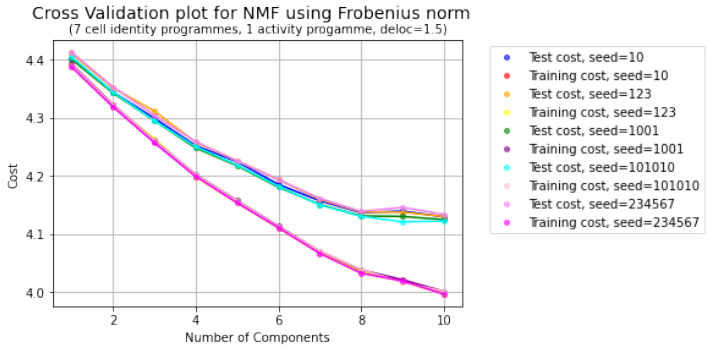
(a) KL Divergence plot for 2 cell identity programme, 1 activity programme case.



(b) Frobenius norm plot for 2 cell identity programme, 1 activity programme case.

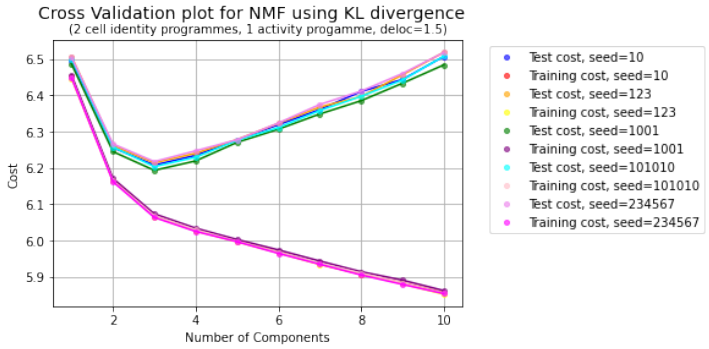


(c) KL Divergence plot for 7 cell identity programme, 1 activity programme case.

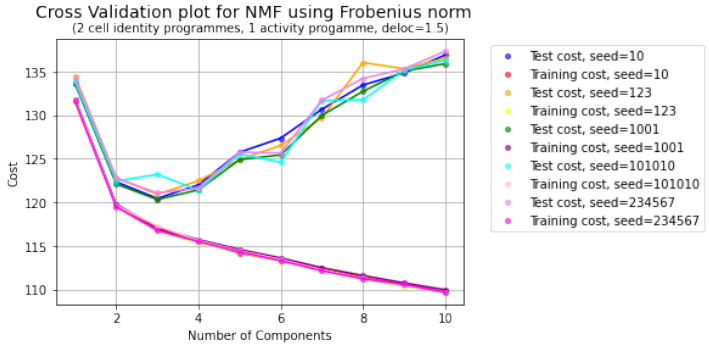


(d) Frobenius plot for 7 cell identity programme, 1 activity programme case.

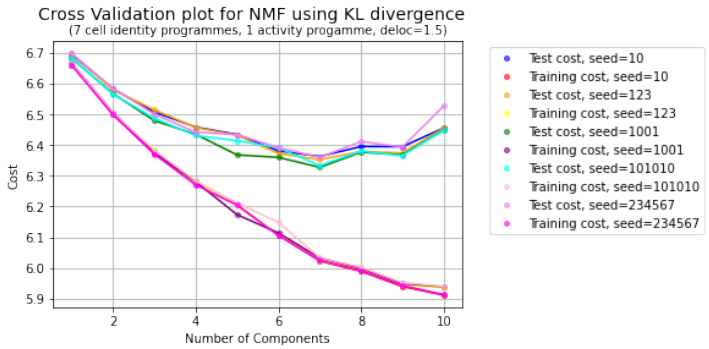
Figure 4.11: Cross-validation plots plotting both training and test costs for the log transformed data over a range of seeds (indicated in the figure legends).



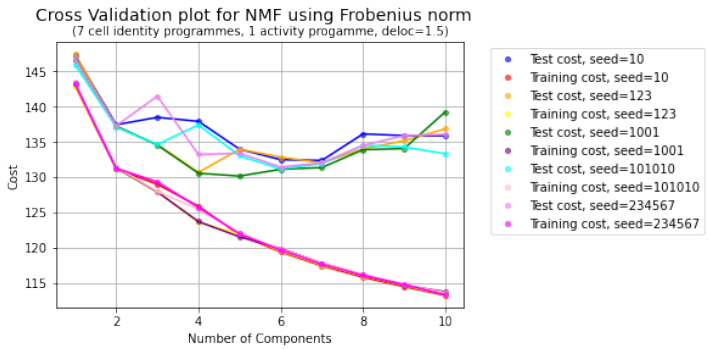
(a) KL Divergence plot for 2 cell identity programme, 1 activity programme case.



(b) Frobenius norm plot for 2 cell identity programme, 1 activity programme case.

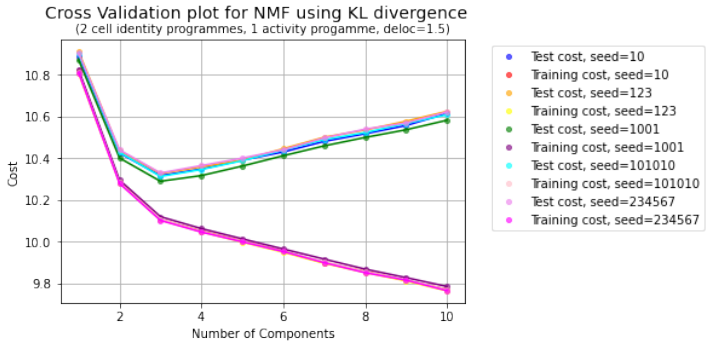


(c) KL Divergence plot for 7 cell identity programme, 1 activity programme case.

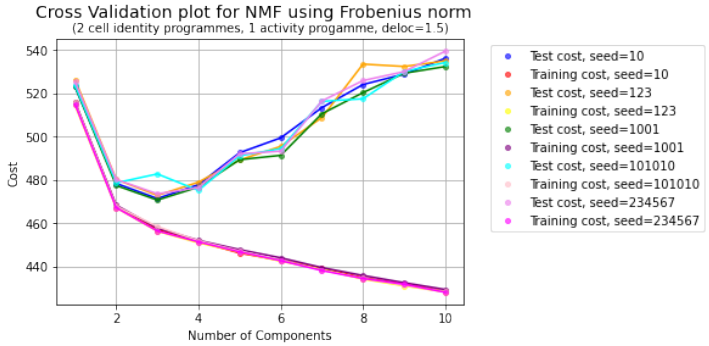


(d) Frobenius plot for 7 cell identity programme, 1 activity programme case.

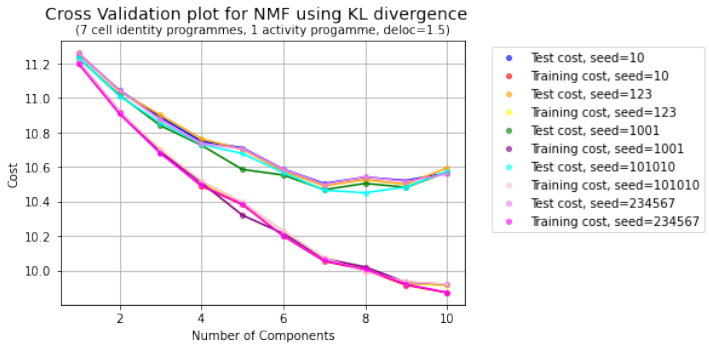
Figure 4.12: Cross-validation plots plotting both training and test costs for the square-root transformed data over a range of seeds (indicated in the figure legends).



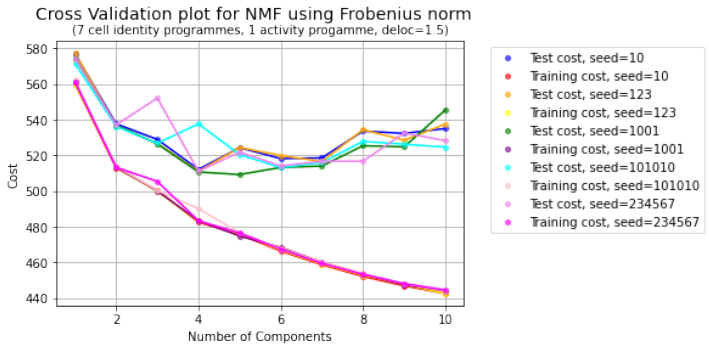
(a) KL Divergence plot for 2 cell identity programme, 1 activity programme case.



(b) Frobenius norm plot for 2 cell identity programme, 1 activity programme case.



(c) KL Divergence plot for 7 cell identity programme, 1 activity programme case.



(d) Frobenius plot for 7 cell identity programme, 1 activity programme case.

Figure 4.13: Cross-validation plots plotting both training and test costs for the Freeman-Tukey transformed data over a range of seeds (indicated in the figure legends).

Chapter 5

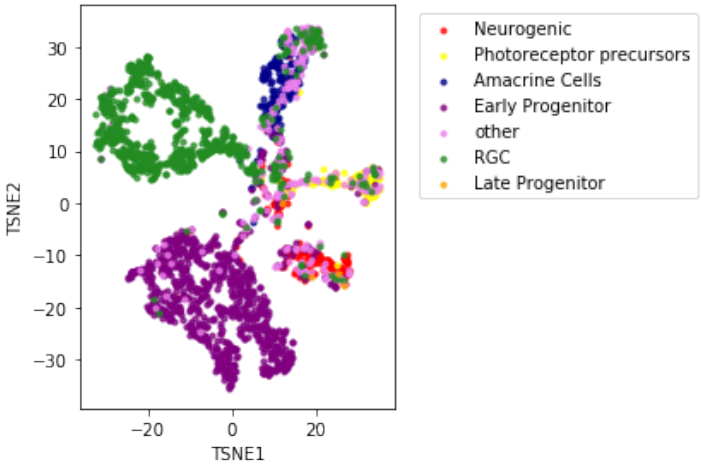
Real scRNA-seq data analysis

This section will firstly define some basic biology against which we can compare the factors we obtain from running Non-negative Matrix Factorisation (NMF) on the Lo Giudice dataset. Then, the previously described cross-validation procedure will be carried out on two transformations of the data (the square-root and Freeman-Tukey transformations), and a number of factors to use for NMF will be identified. After this, a biological interpretation will be attempted for each of the NMF factors obtained. Only one of the two lanes of the Lo Giudice et al. (2020) dataset were used in this analysis to avoid the requirement for batch correction.

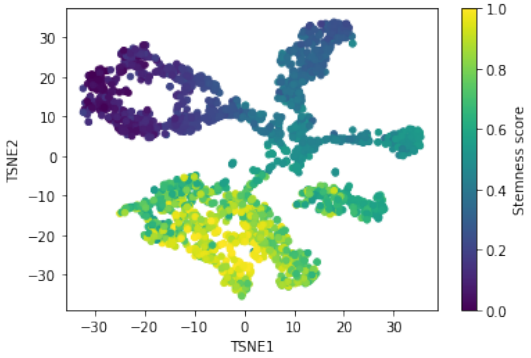
5.1 Basic biology of the Lo Giudice dataset

Stemness is one biological variable that should vary within the Lo Giudice dataset. To measure this, CytoTRACE was implemented on the Lo Giudice dataset, using 'fast' mode since the number of cells was over 3000 [10]. The highest CytoTRACE scores should correlate with the least differentiated cells. However, CytoTRACE scores of 1.0 occur for Retinal Ganglion cells which are known to be differentiated cells. Low CytoTRACE scores occur in the region of the t-SNE plot denoting Early progenitor cells, which are known to be stem cells. Therefore it seems that CytoTRACE scores give the inverse measure of cell-stemness for this dataset. Therefore, to get an indication of cell-stemness directly, we consider $1-$(CytoTRACE score). Cell-stemness can be directly compared with CellAssign annotations to confirm that the highest cell-stemness scores do indeed correlate with the cells that are the least differentiated in the dataset, i.e. Early Progenitor cells (Figure 5.1). The use of the inversion of CytoTRACE as a stemness score is discussed further in Chapter 6.

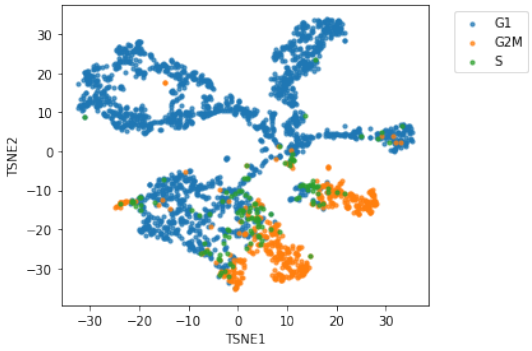
Additionally, another part of the underlying biology which may differentiate between cells is the cell division cycle stage. The cell division cycle is active in cell types which are currently dividing (those that are not differentiated). There are 4 stages to this cycle: G1 growth phase, S DNA replication phase, G2 growth phase and M mitosis phase. It is during M phase that the two daughter cells are separated [29]. In Figure 5.1c, the Lo Giudice dataset is coloured by cell cycle phase. The cell cycle phase scores and annotations were obtained by implementing Cyclone (method described in [35]).



(a) CellAssign attributions for each cell.



(b) Stemness scores for each cell.



(c) Cell Cycle scores for each cell.

Figure 5.1: Lo Giudice data t-SNE plots coloured by CellAssign, Stemness and Cell Cycle Phase respectively.

5.2 Cross-validation and Non-negative Matrix Factorisation applied to the Lo Giudice dataset

Cross-validation and Non-negative Matrix Factorisation (NMF) were run as described in previous chapters. This procedure was carried out for two transformations of the dataset (the square-root and Freeman-Tukey transformations).

Highly variable genes for the real data were selected by using functions included in the *scry* package. Firstly, the mean-variance trend of the data was modelled using modelGeneVar(), before getTopHVGs() was used to obtain the number of genes specified with the highest variance between different cells [38].

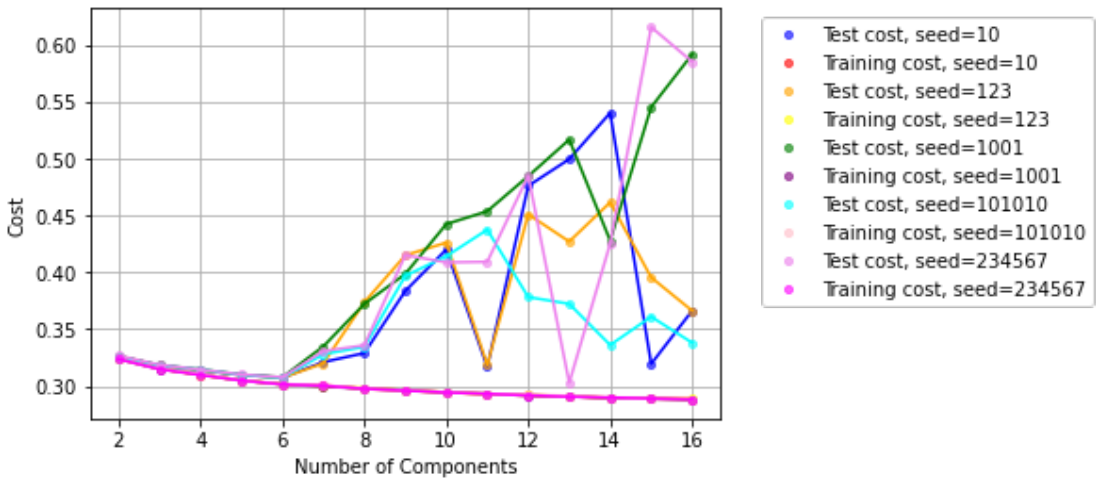
Freeman-Tukey transformation The Freeman-Tukey transformed data cross-validation plot is shown in Figure 5.2a. From this, since a minimum of the test cost is obtained at 6 factors, this was taken to be the optimal number of factors for this dataset. Therefore, NMF was run for 6 factors for the Freeman-Tukey transformed data. A t-SNE plot coloured by the cell weights for each factor is shown in Figure 5.2b.

Square-root transformation The square-root transformed data cross-validation plot is shown in Figure 5.3a. From this, since a minimum of the test cost is obtained at 7 factors, this was taken to be the optimal number of factors for this dataset. Therefore, NMF was run for 7 factors for the square-root transformed data. A t-SNE plot coloured by the cell weights for each factor is shown in Figure 5.3b.

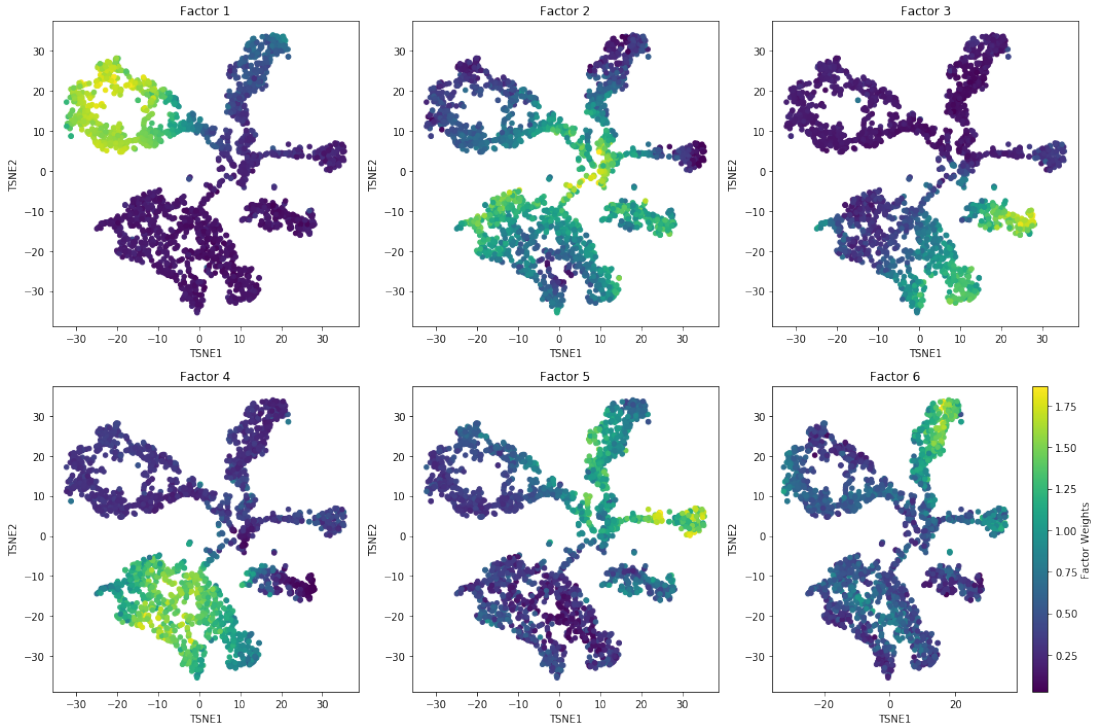
Comparing the t-SNE plots for the Freeman-Tukey transform (Figure 5.2b) with the cell type labels in the CellAssign plot (Figure 5.1a), it is apparent that Factors 1 and 4 correspond to Retinal Ganglion Cells and Early Progenitor cells respectively. Factors 2, 5 and 6 correlate with Neurogenic cells, Photoreceptor precursors and Amarine cells respectively. Factor 3 appears to correspond to G2-M cell cycle phase cells (as shown in 5.1c). Therefore, this factor indicates cells which are currently undergoing cell division.

The square-root transformation NMF cell factors appear to be more distinctly defined. Comparing the t-SNE plots for the square-root transform (Figure 5.3b) with the cell type labels in the CellAssign plot (Figure 5.1a), it is apparent that Factors 1 and 5 correspond to Retinal Ganglion Cells and Early Progenitor cells, while Factors 2,3 and 7 correspond to Photoreceptor precursors, Neurogenic cells and Amacrine cells respectively. Photoreceptor precursor cells here appear to be better defined than in the CellAssign plot where many of these cells are labeled 'other', suggesting that the marker genes from Clark et al. (2019) used to define this cell type may be able to be improved [7]. Comparing Factor 6 of the square-root transform t-SNE (Figure 5.3b) with the cell cycle phases plot (Figure 5.1c), it appears that this factor corresponds to cells in G2/M phase.

The reasoning for Factor 5 (Figure 5.3) is less clear. This factor appears to evenly divide each cell type, suggesting that it may represent an underlying bias of the dataset that has not been accounted for. Checking by plotting against the Quality Control metrics (i.e. library size, number of genes expressed and

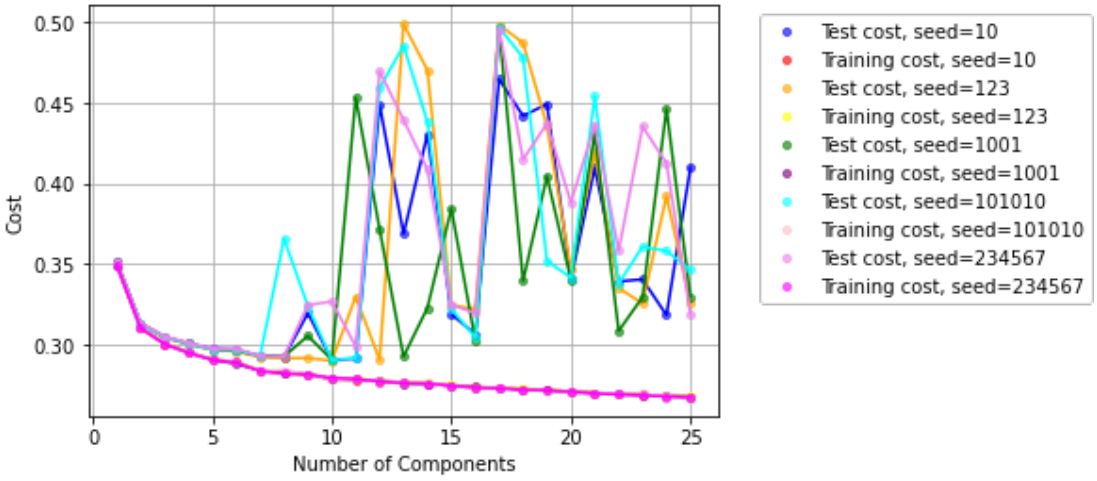


(a) Cross-validation plot for Freeman-Tukey transformed data.

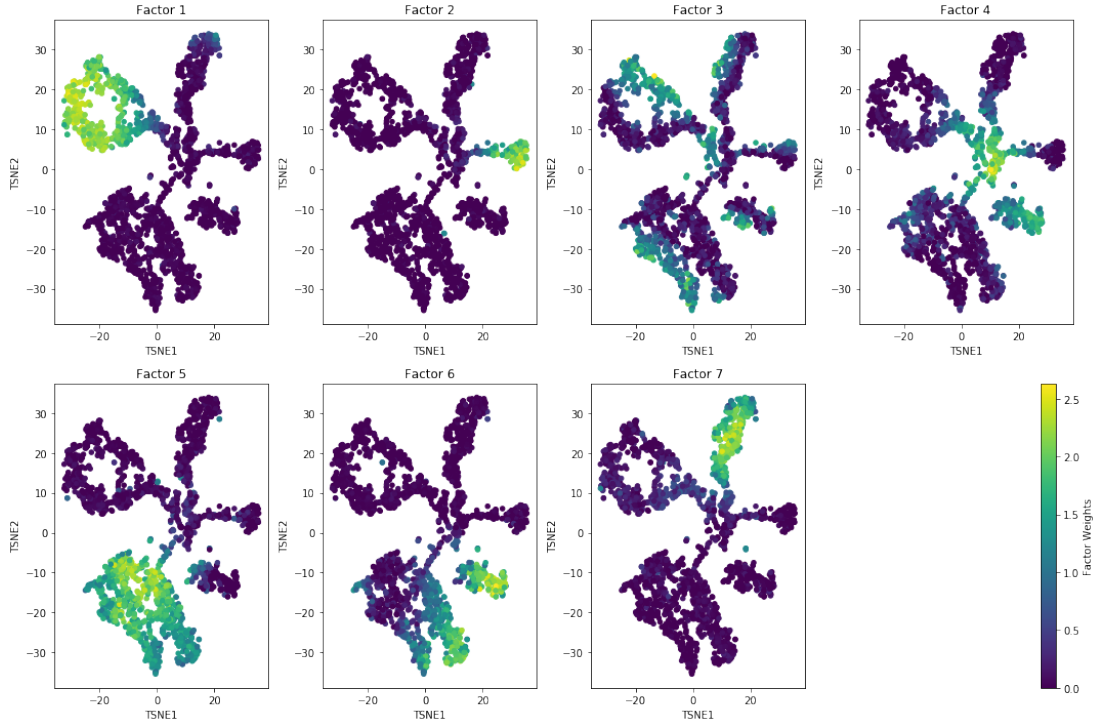


(b) t-SNE plots coloured by the cell weights of NMF factors 1-6.

Figure 5.2: Cross-validation and t-SNE plots coloured by NMF factors 1-6 for the Freeman-Tukey transformed Lo Giudice data.



(a) Cross-validation plot for the square-root transformed data.



(b) t-SNE plots coloured by the cell weights of NMF factors 1-7.

Figure 5.3: Cross-validation and t-SNE plots coloured by NMF factors 1-7 for the square-root transformed Lo Giudice data.

mitochondrial percentage for each cell), it was clear that this did not correlate with any of these metrics (these plots are shown in Appendix F). An alternative explanation could be given by some unknown biological aspect of the dataset. For example, this could be due to the age of the embryo that the cells were taken from (information not available in the original paper). Another explanation for Factor 5 could also be explained by noise in the dataset, as seen in the earlier simulations of scRNA-seq data in Figure 4.9.

5.2.1 Scaling of the dataset

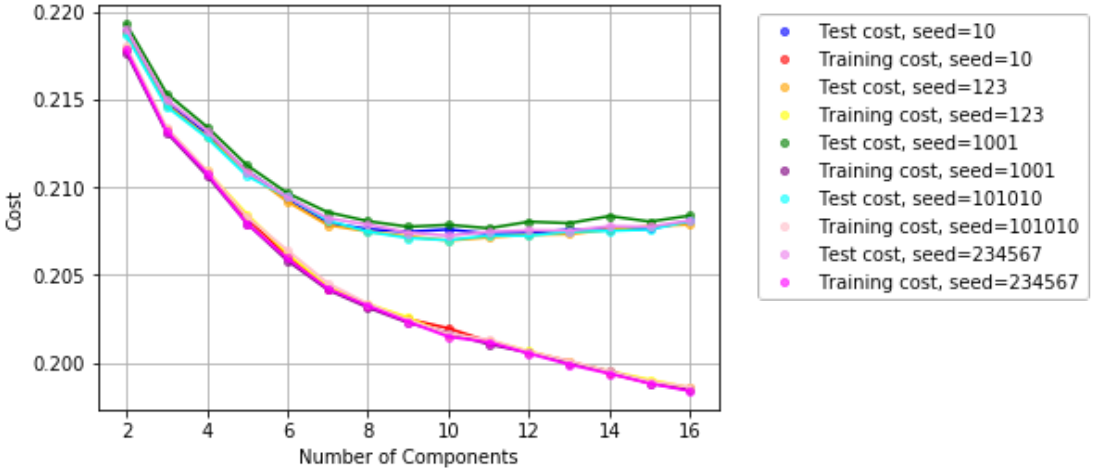
When looking at the highest weighted genes for each of the 6 and 7 factors of the Freeman-Tukey and square-root transforms respectively, the top gene for all factors was the non-coding RNA *Malat1*. This gene was very highly, but differentially expressed compared to other genes and so came up as the top gene for all factors.

After noticing the high prevalence of some genes in the dataset, a scaling was calculated for each gene to try to reduce the bias towards highly expressed genes in the factors. This scaling was given by:

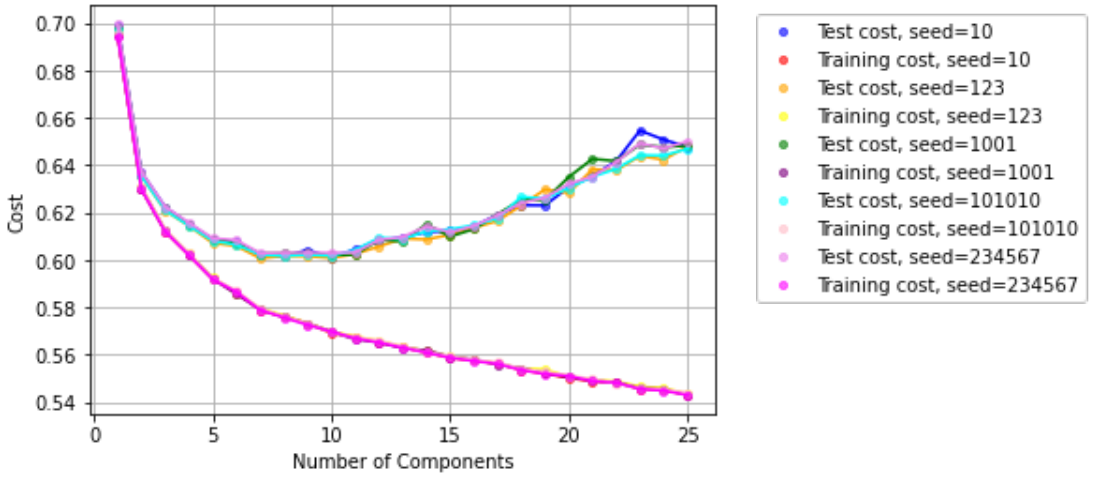
$$\tilde{\mathbf{Y}}_{ij} = \frac{\mathbf{Y}_{ij}}{\sigma_{ij}},$$

where σ_{ij} is the standard deviation for each gene. This is important, as most of the markers used to distinguish biological differences of cells are transcription factors which are less highly expressed in cells. These transcription factors are proteins which influence the expression of other genes, and so are key in determining cell type or cell activities. Additionally, since protein genes are easier to interpret, the genes were filtered to be protein-coding genes only.

When running the cross-validation on this scaled dataset, we see much less variable results between seeds for higher numbers of factors (Figure 5.4). However, these plots seem to have minimums which fall across a range of values, so we run NMF for several values of the minimum and select the number of factors where the factors on the t-SNE plots appear clearest defined. From these plots, we select the number of factors for the scaled Freeman-Tukey transform to be 9 factors and for the square-root transform to again be 7 factors.



(a) Cross-validation plot for the scaled Freeman-Tukey transformed data.

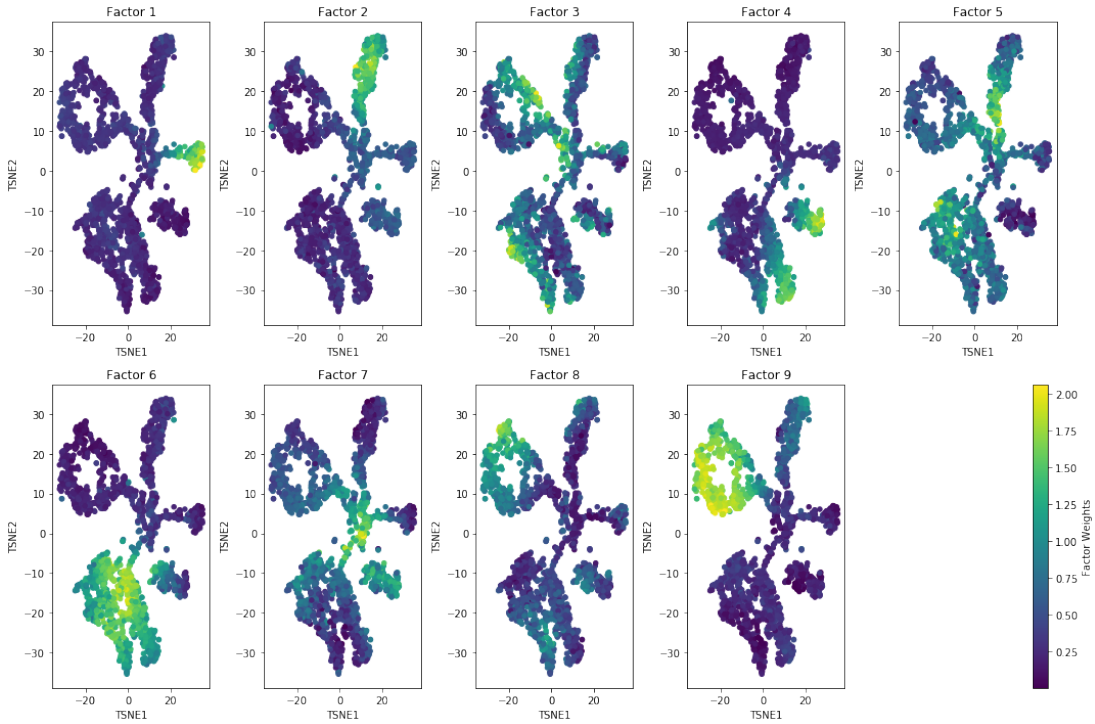


(b) Cross-validation plot for the scaled square-root transformed data.

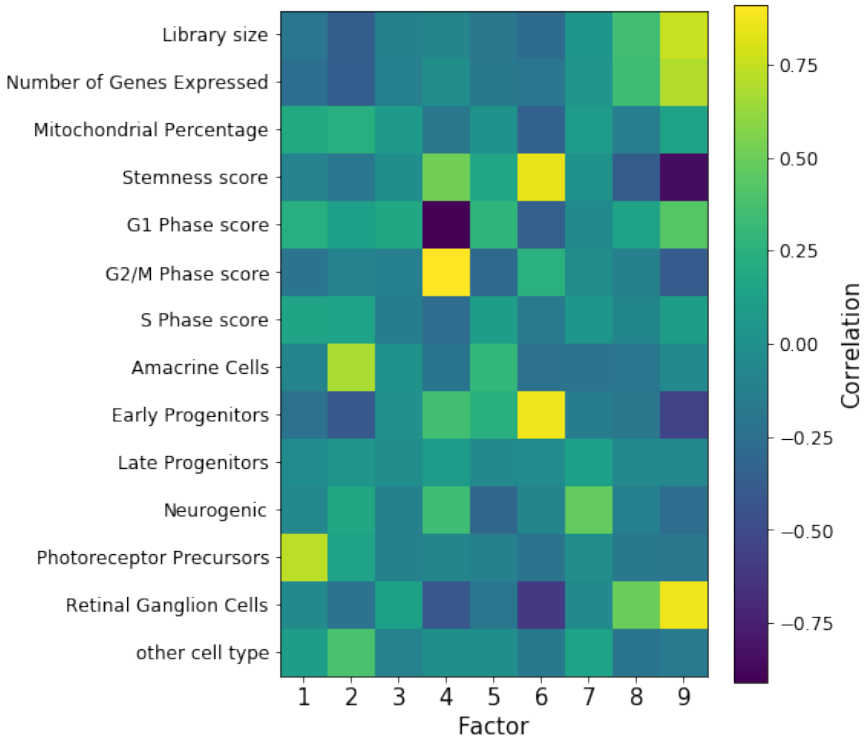
Figure 5.4: Cross-validation plots for scaled Lo Giudice data.

The results of NMF on the scaled Freeman-Tukey transformed data are shown in Figure 5.5. Figure 5.5b shows the correlations between the cell factor weights and the scores for each set of metadata for each cell. This includes the quality control metrics (library size, number of genes expressed and mitochondrial percentage), the stemness score, the cell cycle phase scores, and the scores for each cell type. These correlations match up well with CellAssign annotations and the t-SNE plots coloured by cell weight. In this correlation plot we see that the Photoreceptor Precursor probability correlates with Factor 1, Amacrine cells with Factor 2, Early Progenitor cells with Factor 4 and Retinal Ganglion Cells correlate with Factor 9. Factor 7 has some correlation with Neurogenic probability, however, this is a weaker correlation. Factor 4 is very strongly correlated with G2/M score. Factors 3, 5 and 8 have no correlation with any of the metadata explored, so these factors could be explained by another variable or be due to noise in the dataset and so should be investigated further.

The results of NMF on the scaled square-root transformed data are shown in Figure 5.6. Figure 5.6b shows the correlations between the cell factor weights and the scores for each set of metadata for each cell. The correlations again correspond nicely to what is seen in the CellAssign and t-SNE plots. Here we see that Factor 1 correlates with Retinal Ganglion Cells, Factor 2 with Photoreceptor Precursors, Factor 4 with Neurogenic cells. Factor 5 with Early Progenitor cells and Factor 7 with Amacrine cells. Factor 6 correlates with the G2/M phase of the cell cycle. Factor 3 is not correlated with any of the metadata, so needs further investigation. Overall, both scaled transformations show that the metadata correlate with the factors we would expect, and so provide further evidence that we are finding real transcriptional differences within the data.

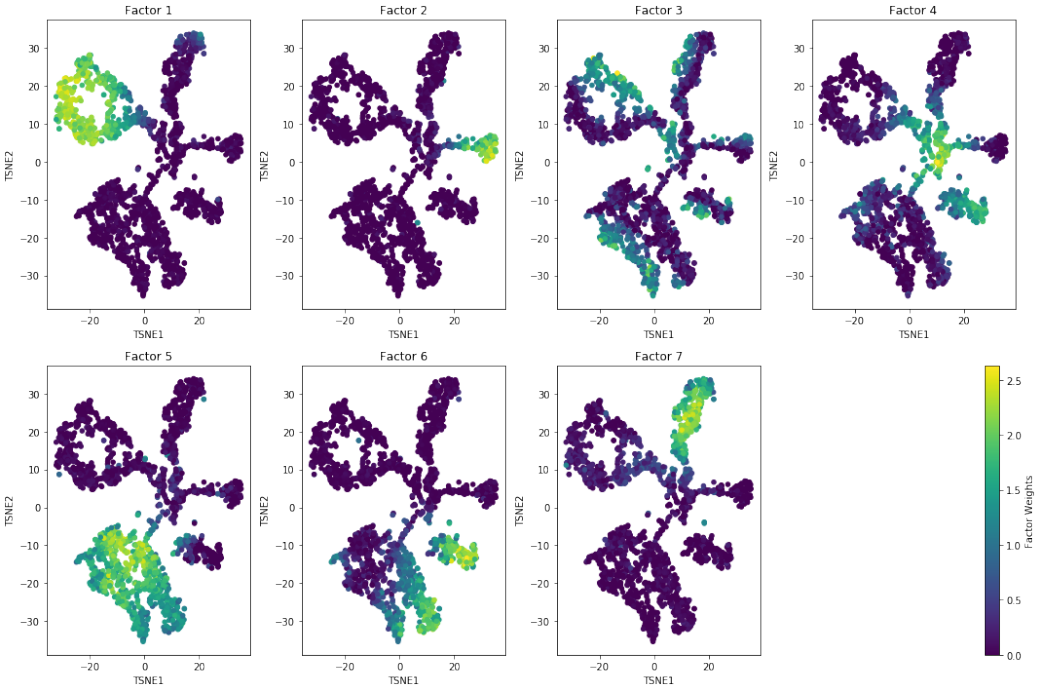


(a) t-SNEs plot for the cell weights of NMF factors 1-9.

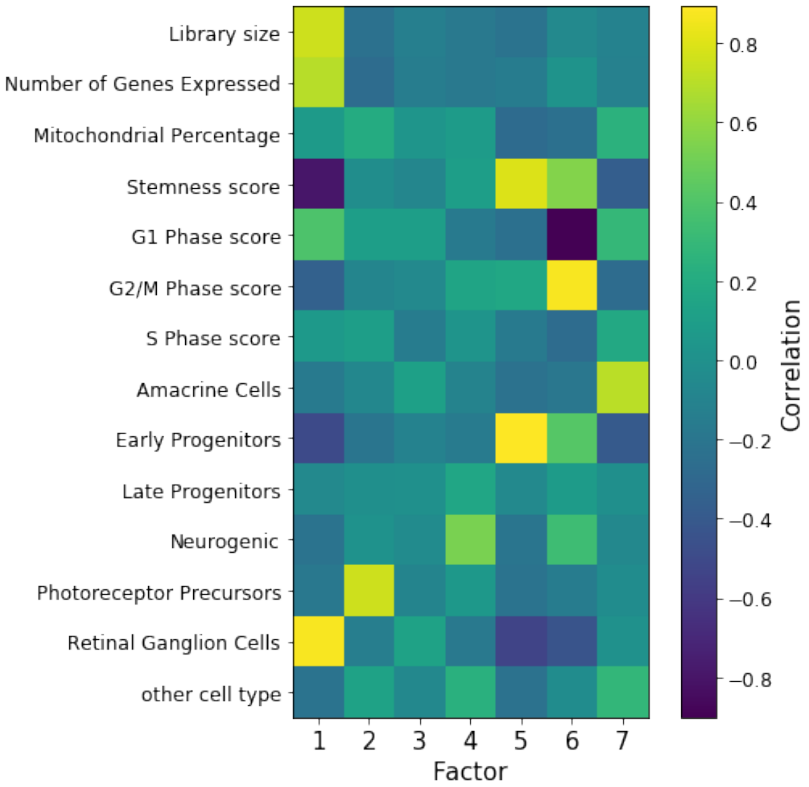


(b) Correlation heatmap of correlations between each factor and each element of metadata.

Figure 5.5: Results for the NMF with 9 factors run on scaled Freeman-Tukey transformed Lo Giudice data.



(a) t-SNEs plot for the cell weights of NMF factors 1-7.



(b) Correlation heatmap of correlations between each factor and each element of metadata.

Figure 5.6: Results for the NMF with 7 factors run on scaled square-root transformed Lo Giudice data.

5.2.2 Top genes for each factor

Finally, we considered the genes which were most upregulated in each of the factors obtained for NMF. These gene factor weights are obtained from the rows of the factor $\times$ genes matrix. The gene weights are then ordered from highest to lowest to obtain the genes with the highest weights for each factor. These highest weighted genes denote transcriptional signatures for each of the factors. To allow easier interpretation of the factors, only genes encoding transcription factors (TFs) are shown as TFs are often used to define different cell types and biological processes. The tables of the highest genes for each of the cases described previously in this chapter are found in Appendix D. These cases are:

- 6 factor NMF for the Freeman-Tukey transformed data
- 7 factor NMF for the square-root transformed data
- 9 factor NMF for the scaled Freeman-Tukey transformed data
- 7 factor NMF for the scaled square-root transformed data

Some interpretations of the role of each factor can be made by considering the most highly expressed TF gene in each factor. For example, for the scaled square-root transformed NMF with 7 factors, for factor 1, the top TF gene *Ebf1* is required for RGC differentiation, whilst the 2nd highest weighted gene *Isl1* was one of our marker genes for Retinal Ganglion cells [7, 5].The top TF for factor 2 is *Thrb*, which is a gene associated with photoreceptor development [31]. This provides further evidence, other than localisation of the high cell factor weights, that this factor is encoding photoreceptor precursor cells. *Zfp36l1* is a gene with high weighting in factor 5, and this gene is known to be expressed in progenitor cells and so provides further evidence for our previous annotation of this factor [52]. *Pax6* is the highest transcription factor identified in factor 7 which denotes Amacrine cells and this is again a CellAssign marker gene used to define this same cell type [7]. Some of the highest weighted TF genes in factor 6 are associated with mitosis. *Mis18bp1* is the 3rd highest TF gene in this factor. The protein transcribed from this gene is Mis18bp1, which is required for the association of CENP-A to centrosomes. The gene encoding CENP-A, *Cenpa* is the 4th highest TF gene in this factor and is essential for mitosis to occur successfully [36]. This shows that the factor associated with G2/M phase is defined by genes involved in mitosis.

Overall, using both square-root and Freeman-Tukey transformations of the scRNA-seq dataset (with and without scaling), NMF elucidated factors which explained the underlying biology of the dataset.

Chapter 6

Discussion

During this project, I implemented a cross-validation analysis pipeline which can be used to determine the optimal number of factors for matrix factorisation methods. NMF was able to recover the underlying ground truth of simulated single cell RNA-sequencing datasets. Here it was also identified that the square-root and Freeman-Tukey transformations were useful in helping us identify the optimal number of factors more easily. On the real data, both square-root and Freeman-Tukey transformations of the scRNA-seq dataset were considered, and NMF on these transformed datasets elucidated factors which explained the underlying biology.

However, this approach needs to be developed further. Of note, in both Figures 5.2a and 5.3a, the variability in the test cost between masking matrix seeds increases greatly above 7 factors. One reason for this could be that our data matrix is only approximately 2500 cells $\times$ 2000 genes. This is relatively small for a dataset of this type. This means that each cell type will only be present in a subset of these cells, and small cell types do not appear to define the factors found by NMF (details of how many cells are in each cell type can be found in the table in Appendix C). In our plots, it appears that there is little consensus between seeds for higher numbers of factors where we may see these less common patterns in the dataset. It could be that for a dataset of this size, more than 7 factors is very difficult for NMF to resolve with this size of dataset.

In future, to increase the size of the dataset, firstly we could integrate both lanes and also look for similar datasets to merge with this dataset, or use larger existing datasets to compliment this work. To be able to integrate datasets, we would need to use batch-effect correction methods such as LIGER to correct for technical and biological batch effects within the dataset [42]. Batch-effect correction is important as often even in scRNA-seq data under the same technical and biological conditions, different replicates often show a distinct batch effect as seen in Figure 2.3 where the 2 lanes of Lo Giudice cluster separately in a t-SNE plot. LIGER is a recent method which uses NMF to obtain a low-dimensional representation of the input data before performing clustering and using a shared factor neighbourhood graph to connect clusters with similar neighbourhoods. Having identified clusters, normalisation is carried out to make this dataset match a reference dataset, and thus accomplishing batch correction[42]. This method of batch-effect correction would be particularly interesting to explore, as it illus-

trates the vast arrays of uses that NMF has in the analysis of scRNA-seq data.

The variability in the test cost between seeds increasing greatly above 7 factors is not observed in all of our cross-validation plots. We see an improvement in the cross-validation plots in Figure 5.4 by scaling each gene by the standard deviation, however, this scaling could be inflating genes with low variances excessively, and so alternative methods should be considered for improving interpretability of cross-validation plots beyond 7 factors.

As an additional check that this cross-validation procedure works, the optimal number of factors could be determined using the Cophenetic Correlation Coefficient (CCC). NMF factors can be considered to be clusters of genes; this coefficient gives a measure of degree of fit of a set of clusters to a dataset. Here the maximum correlation for a number of factors would indicate that this is the best number of factors to use for the matrix factorisation method in question. More details of this method are given in Appendix D [4]. The reason a cross-validation approach was chosen is that CCC may not work well if the data is not well clustered. Since the Lo Giudice dataset contains a differentiation trajectory, and so has cells in intermediate cell states, CCC may not give a good estimate of the best number of factors to use for this dataset.

There are multiple matrix factorisation methods that can be applied to scRNA-seq datasets [37]. Whilst only NMF was considered in this project, the cross-validation method implemented in this report could also be used for both PCA and ICA. Implementing the same cross-validation procedure on other matrix factorisation methods would also be interesting. A direct comparison between PCA, ICA and NMF on the same dataset would allow us to decide which of these methods is best for uncovering the underlying data structure of developmental scRNA-seq datasets.

The required inversion of CytoTRACE to obtain a stemness score highlights an issue with CytoTRACE. Inversion of CytoTRACE scores as seen here (such that the lowest scores give undifferentiated cells) has been observed in other datasets [10, 24]. For example, this is observed in myelin forming cells dataset in the Gulati et al. (2020) Supplementary Information [10]. This inversion of stemness observed in the CytoTRACE scores is likely to be due to the underlying assumption of the CytoTRACE algorithm which assumes that the least differentiated cells have the highest number of different genes expressed. In this dataset, this assumption is not true; we observe that the highest numbers of genes expressed occur instead in the Retinal Ganglion Cells, the most differentiated cells in the dataset (Figure in Appendix F). Therefore, it is likely that CytoTRACE doesn't generalise to all developmental scRNA-seq datasets. The Supplementary Information of Gulati et al. also notes that other stemness methods did not work on datasets where CytoTRACE was inverted. Therefore, obtaining new methods to determine cell-stemness is likely to be required to obtain a method that applies to all developmental scRNA-seq datasets.

If through further improvement of the methods discussed in this report we were able to find a stemness transcriptional signature, this would be useful for denoting stem cells in cell populations. For example, a stemness transcriptional signature could be compared to genes upregulated in what are called 'cancer stem cells' to see if these cancerous cells are undergoing true reversion to an

undifferentiated state. Cancer stem cells have the capacity to self-renew and to give rise to progeny that are different from themselves, similar to typical stem cells [8]. If there are some differences between cancer stem cells and non-cancer stem cells, then these differences can be exploited by treatments allowing specific targeting of cancer stem cells without risking non-cancer stem cells which are vitally important for replenishing damaged tissues.

Overall, matrix factorisation methods allow us to uncover the underlying biology of datasets in an unsupervised manner. Here, we implemented a cross-validation method that allowed us to optimise the NMF factorisation so that we could obtain meaningful factors that gave us insight into the structure of the dataset.

Bibliography

[1] Amezquita, R. et al., Orchestrating Single-Cell Analysis with Bioconductor. Version 1.0.6, R version 4.0.3, Bioconductor 3.12 (2020). `https://bioconductor.org/books/release/OSCA/` [Accessed: 20th March 2021].

[2] Banerji, C. R. S. et al., Cellular network entropy as the energy potential in Waddington's differentiation landscape. *Scientific Reports*, **3**, 1-7 (2013).

[3] Bossel Ben-Moshe, N. et al., Predicting bacterial infection outcomes using single cell RNA-sequencing analysis of human immune cells. *Nature Communications*, **10**, 3266 (2019).

[4] Brunel, J.P. et al., Metagenes and molecular pattern discovery using matrix factorization. *National Academy of Sciences*, **101(12)**, 4164-4169 (2004).

[5] Chuang, J. H. et al., Expression profiling of cell-intrinsic regulators in the process of differentiation of human iPSCs into retinal lineages. *Stem cell research & therapy*, **9(1)**, 140 (2018).

[6] Clancy, S., RNA Functions. *Nature Education*, **1(1)**, 102 (2008).

[7] Clark, B. S. et al., Single-Cell RNA-Seq Analysis of Retinal Development Identifies NFI Factors as Regulating Mitotic Exit and Late-Born Cell Specification. *Neuron*, **102(6)**, 1111-1126 (2019).

[8] Dawood S, et al., Cancer stem cells: implications for cancer therapy. *Oncology*, **28(12)**, 1101-7, 1110 (2014).

[9] Eling, N. et al., Correcting the Mean-Variance Dependency for Differential Variability Testing Using Single-Cell RNA Sequencing Data. *Cell systems*, **9(4)**, 401–413 (2019).

[10] Gulati, G. S. et al., Single-cell transcriptional diversity is a hallmark of developmental potential. *Science*, **367**, 405-411 (2020).

[11] Grün, D. et al., De Novo Prediction of Stem Cell Identity using Single-Cell Transcriptome Data. *Cell Stem Cell*, **19**, 266-277 (2016).

[12] Haque, A. et al., A practical guide to single-cell RNA-sequencing for biomedical research and clinical applications. *Genome Medicine*, **1**, 1-12 (2017).

[13] Hwang, B. et al., Single-cell RNA sequencing technologies and bioinformatics pipelines. *Experimental & Molecular Medicine*, **50**, 1-14 (2018).

[14] Hyvärinen, A., New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit. *Advances in Neural Information Processing Systems*. **10**, 273–279 (1998).

[15] Hyvärinen, A. et al., What Is Independent Component Analysis? In: Independent Component Analysis. New York, USA: *John Wiley & Sons*, 145-164 (2001).

[16] Hyvärinen, A. et al., Part IV: Applications of ICA. In: Independent Component Analysis. New York, USA: *John Wiley & Sons*, 391-448 (2001).

[17] Ilicic, T. et al., Classification of low quality cells from single-cell RNA-seq data. *Genome Biology*, **17**, 29 (2016).

[18] James, G. et al., An Introduction to Statistical learning. Vol. 103. New York: *Springer*, Chapter 9 & 10, 337-417 (2013).

[19] Johnson, R.A. and Wichern, D.W., Applied Multivariate Statistical Analysis. (6th edition), *Pearson Prentice Hall*, Chapter 11 (2007)

[20] Kolb, H. et al., Webvision: The Organization of the Retina and Visual System. Salt Lake City (UT): *University of Utah Health Sciences Center* (1995).

[21] Kotliar, D. et al., Identifying gene expression programs of cell-type identity and cellular activity with single-cell RNA-seq. *eLife*, 8:e43803 (2019).

[22] Kutz, J.N., Chapter 16: Independent Component Analysis. In: Data-Driven Modeling & Scientific Computation : Methods for Complex Systems & Big Data. (1st edition), Oxford: *Oxford UP* (2013).

[23] Lee, D. and Seung, H., Learning the parts of objects by non-negative matrix factorization. *Nature*, **401**, 788–791 (1999).

[24] Li, D.-Q. et al., Single-cell transcriptomics identifies limbal stem cell population and cell types mapping its differentiation trajectory in limbal basal epithelium of human cornea. *Ocular Surface*, **20**: 20-32 (2021).

[25] Lo Giudice, Q. et al., Single-cell Transcriptional Logic of Cell-Fate Specification and Axon Guidance in Early-Born Retinal Neurons. *Development*, **146(17)**:dev178103 (2019).

[26] Lun. A.T.L. et al., A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor. *F1000Research*, **5**, 2122 (2016).

[27] McInnes, L. and Healy, J., UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, *ArXiv e-prints* 1802.03426 (2018).

[28] Mendes Levitin, H. et al., Single-Cell Transcriptomic Analysis of Tumor Heterogeneity. *Trends in Cancer*, **4(4)**, 264-268 (2018).

[29] Morgan, D. O., Chapter 1: The Cell Cycle. In: The Cell Cycle: Principles of Control, London:*New Science Press* (2007).

[30] Nason, G. P., Chapter 6: Multiscale Variance Stabilization. In: Wavelet Methods in Statistics with R. Use R!, New York: *Springer* (2008).

[31] Ng, L. et al., Developmental expression of thyroid hormone receptor beta2 protein in cone photoreceptors in the mouse. *Neuroreport*, **20(6)**: 627–631 (2009).

[32] Pedregosa, F. et al., Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, **12**, 2825-2830 (2011).

[33] Qiu, X. et al., Single-cell mRNA quantification and differential analysis with Census. *Nature Methods*, **14**, 309–315 (2017).

[34] Ralston, A. and Shaw, K., Gene expression regulates cell differentiation. *Nature Education*, **1(1)**, 127 (2008).

[35] Scialdone, A. et al., Computational assignment of cell-cycle stage from single-cell transcriptome data. *Methods*, **85**, 54-61 (2015).

[36] Srivastava, S. et al., Posttranslational mechanisms controlling centromere function and assembly. *Current Opinion in Cell Biology*, **52**, 126–135 (2018).

[37] Stein-O'Brien, G. L. et al., Enter the Matrix: Factorization Uncovers Knowledge from Omics. *Trends in Genetics*, **34(10)**,790-805 (2018).

[38] Street, K. et al., scry: Small-Count Analysis Methods for High-Dimensional Data. R package version 1.2.0 (2020) `https://bioconductor.org/packages/scry.html` [Accessed: 12th February 2021].

[39] Tang, F. et al., mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods*, **6**, 377–382 (2009).

[40] Teschendorff, A. E. and Enver, T., Single-cell entropy for accurate estimation of differentiation potency from a cell's transcriptome. *Nature Communications*, **8**, 15599 (2017).

[41] Townes, F. W. et al., Feature selection and dimension reduction for single-cell RNA-seq based on a multinomial model. *Genome Biology*, **20**, 295 (2019).

[42] Tran, H.T.N. et al., A benchmark of batch-effect correction methods for single-cell RNA sequencing data. *Genome Biology* **21**, 12 (2020).

[43] Vallejos, C., Exploring a world of a thousand dimensions. *Nature Biotechnology*, **37**, 1423–1424 (2019).

[44] van der Maaten, L. and Hinton, G., Visualising Data using t-SNE. *Journal of Machine Learning Research*, **9**, 2579-2605 (2008).

[45] Virtanen, P. et al., SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, **17(3)**, 261-272 (2020).

[46] Wages, J.M. Jr., "POLYMERASE CHAIN REACTION." *Encyclopedia of Analytical Science*, 243–250 (2005).

[47] Wagner, F., Straightforward Clustering of single-cell RNA-seq data with t-SNE and DBSCAN. *Cold Spring Harbor Laboratory*, 770388 (2020).

[48] Westad, F. and Kermit, M., 2.14 Independent Component Analysis. In: Brown, S.D., Tauler, R. and Walczak, B., Comprehensive Chemometrics. *Elsevier*, 227-248 (2009).

[49] Williams, A. H., How to cross-validate PCA, clustering, and matrix decomposition models (2018).
`http://alexhwilliams.info/itsneuronalblog/2018/02/26/crossval/`
[Accessed: 31st January 2021].

[50] Wold, S., Cross-Validatory Estimation of the Number of Components in Factor and Principal Components Models. *Technometrics*, **20:4**, 397-405 (1978).

[51] Wright, S. J., Coordinate descent algorithms. *Mathematical Programming*, **151**, 3-34 (2015).

[52] Wu, F. et al., Zfp36l1 and Zfp36l2 balances proliferation and differentiation in the developing retina. *Cold Spring Harbor Laboratory*, 2020.12.15.422926 (2020).

[53] Zappia, L. et al., Splatter: simulation of single-cell RNA sequencing data. *Genome Biology*, **18**:174 (2017).

[54] Zappia, L. et al., Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. *PLOS Computational Biology*, **14(6)**:e1006245 (2018).

[55] Zhang, A. W. et al., Probabilistic cell-type assignment of single-cell RNA-seq for tumour microenvironment profiling. *Nature Methods*, **16**, 1007-1015 (2014).

Appendix A

Alternative Matrix Factorisation Methods

A.1 Principal Component Analysis

Principal Component Analysis (PCA) aims to reduce the dimensionality of a high dimensional dataset whilst accounting for as much of the original variation in the original dataset as possible. This process transforms a set of intercorrelated variables $\mathbf{x} = (x_1, x_2, ..., x_p)^T$ into a set of uncorrelated variables $\mathbf{y} = (y_1, y_2, ..., y_p)^T$. $y_1, y_2, ..., y_p$ are termed the principal components and are linear combinations of the original variables $x_1, x_2, ..., x_p$ such that:

$$y_1 = l_1^T \mathbf{x} = l_{11}x_1 + ... + l_{1p}x_p$$

$$\vdots$$

$$y_p = l_p^T \mathbf{x} = l_{p1}x_1 + ... + l_{pp}x_P.$$

We now wish to find vectors of coefficients $l_1, ...l_p$ which maximise the variance of the projections $l_1\mathbf{x}, ..., l_p\mathbf{x}$. Each principal component y_i is a projection of $\mathbf{x}$ into one-dimensional space, where components of x are weighted by the elements of l. The best explanation of the variance is given by this weighting.

The matrix of coefficients $\mathbf{L} = [l_{ij}]$ producing the principal components is the result of solving:

$$\max_{l_i} l_i^T \Sigma l_i \text{ such that } l_i^T \Sigma l_i = 0 \text{ and } l_i^T l_i = 1 \text{ for i} = 1, ..., p, \text{ with } j \neq i,$$

where Σ is the covariance matrix [19].

A.2 Independent Component Analysis

Independent Component Analysis (ICA) reveals independent factors from a set of signals that have been mixed, which are assumed to be mutually independent. ICA builds on the concepts of PCA, extending it to extract interleaved datasets.

In general we can describe this N dimensional system as: We observe N

random variables $x_1, ..., x_N$, which are modelled as linear combinations of M random variables $s_1, ..., s_M$:

$$x_i(t) = a_{i1}s_1 + a_{i2}s_2 + ... + a_{iM}s_M, 1 \leq i \leq N,$$

where a_{ij}, for $i = 1, ..., N$ and $j = 1, ..., M$ are some real coefficients to be determined. By definition the s_j are mutually independent.

In matrix form, this is expressed as:

$$\mathbf{x} = \mathbf{A}\mathbf{s},$$

where $\mathbf{x}$ are the mixed signal measurements, $\mathbf{s}$ are the original measurements to be determined and $\mathbf{A}$ is the matrix of coefficients which determines how signals are mixed as a function of the physical system of interest [22].

ICA is a generative model, which means it describes how the observed data are generated by a process of mixing the components s_j. The ICs (Independent Components) s_j cannot be directly observed and the mixing coefficients a_{ij} are assumed to be unknown. We must estimate the s_j and a_{ij} using the observed x_i [15].

Estimation of the true underlying sources $\mathbf{u}_m$ and the unmixing system $\mathbf{W} = \mathbf{A}^{-1}$ can occur under the assumption that the sources s_m are statistically independent, where:

$$\mathbf{U} = \mathbf{X}\mathbf{W}.$$

A.2.1 Restrictions in ICA:

1. There must be at least as many sensor observations as there are source signals, that is, $N \geq M$.

2. The independent components in $\mathbf{S}$ are always mutually independent.

3. Only one source signal may follow a Gaussian distribution (the rest must be Non-Gaussian).

4. The order of the independent components and their variances cannot be determined.

If the first assumption is not satisfied then the mixing system $\mathbf{A}$ does not have full rank. Therefore, the matrix is rank deficient and so there is insufficient information as M parameters cannot be estimated with less than M data points.

The second assumption implies that if we let the components of $\mathbf{S}$ have zero-mean then their joint pdf (probability density function) is factorial:

$$p_s(\mathbf{S}) = \prod_{m=1}^{M} p_{s_m}(s_m).$$

Without loss of generality, we can assume mixture variables and independent components have zero-mean. If zero-mean is not satisfied then pre-processing of

the data can be carried out to make this hold. The mixture variables can be centred to have zero mean by:

$$\mathbf{x} = \mathbf{x}' - \mathbb{E}(\mathbf{x}').$$

This also makes the independent components also zero-mean, since:

$$\mathbb{E}(\mathbf{s}) = \mathbf{A^{-1}}\mathbb{E}(\mathbf{x}).$$

Since the mixing matrix remains the same, this pre-processing has no affect on the estimation of $\mathbf{A}$ [15].

The third assumption is necessary since Gaussian distributions are second-order and only require mean and covariance to completely determine the distribution. ICA requires higher-order statistical information in order to separate mixed sources [48].

The fourth assumption is necessary since components of $\mathbf{s}$ and $\mathbf{A}$ are unknown, so their positions within the transform are arbitrary.

Estimation of the independent components is carried out by optimising various criteria, as discussed in the following section.

A.2.2 Estimation of Independent Components:

The following sections describe methods by which the Independent Components can be estimated.

Independence by Minimum Mutual Information: Infomax (Information-maximisation) is an ICA estimation principle that maximises the output entropy or information flow through a system. If the matrix $\mathbf{X}$ is used as the input of the system, then the output $\mathbf{y}$ is of the form: $y_m = g_m(\mathbf{u}_m) = g_m(\mathbf{w}_m^T\mathbf{X})$, where g_m is a non-linear transfer function, u_m is the source to be estimated and w_i are the weights. The goal is to maximise the entropy of $\mathbf{y}$.

The joint entropy for a system of two outputs y_1 and y_2 is given by:

$$H(y_1, y_2) = H(y_1) + H(y_2) - I(y_1, y_2).$$

Maximising the joint entropy requires the individual entropies to be maximised, whilst the mutual information $I(y_1, y_2)$ is minimised. When $I(y_1, y_2)$ is zero, y_1 and y_2 are independent. This can be generalised for n variables $y_1, ...y_n$:

$$H(y_1, y_2, ..., y_n) = H(y_1) + H(y_2) + ... + H(y_n) - I(y_1, y_2, ..., y_n).$$

N.B. The mutual information is defined as

$$I(Y, X) = H(Y) - H(Y|X).$$

The non-linear transfer function g_m is chosen such that it is an increasing function with values between 0 and 1, and the maximum entropy for the bounded output occurs when the probability distribution is uniform. If each g_m is chosen

to be the cumulative distribution function of its estimated source $\mathbf{u}_m$ to be estimated, then statistical independence between the outputs y_m is achieved as well as independence between the estimated sources $\mathbf{u}_m$[48].

Independence by Maximum Non-Gaussianity: The central limit theorem states that the distribution of a sum of independent random variables tends towards a Gaussian distribution, under certain conditions. Assume the data $\mathbf{x}$ is distributed according to the ICA data model:

$$\mathbf{x} = \mathbf{A}\mathbf{s}.$$

Estimating the independent components can be achieved by finding the right combinations of the mixture variables, since

$$\mathbf{s} = \mathbf{A}^{-1}\mathbf{x}.$$

Thus an estimate of the independent components can be given by $\mathbf{u} = \mathbf{b}^T\mathbf{x} = \sum_i b_i x_i$, where $\mathbf{b}$ is a linear combination of the x_i.

This is equivalent to $\mathbf{u} = \mathbf{b}^T\mathbf{A}\mathbf{s}$, a linear combination of the s_i, where $\mathbf{q} = \mathbf{b}^T\mathbf{A}$ are the coefficients. Therefore we have $\mathbf{u} = \sum_i q_i s_i$. If $\mathbf{b}$ was one of the rows of $\mathbf{A}^{-1}$ then the linear combination $\mathbf{b}^T\mathbf{x}$ would equal one of the independent components s_i. We therefore wish to determine $\mathbf{b}$ such that it approximately equals a row of $\mathbf{A}^{-1}$. Since a sum of independent variables is more gaussian than the original variables, $\mathbf{u} = \mathbf{q}^T\mathbf{s}$ will be more gaussian than the individual s_i. This is least gaussian when the sum equals one of the s_i. In this case only one of the q_i is nonzero. Additionally, we can see from this that non-Gaussianity is equivalent to independence.

In practice, we do not know the values of $\mathbf{q}$, however, we do know that

$$\mathbf{q}^T\mathbf{s} = \mathbf{b}^T\mathbf{x}.$$

Therefore, we can vary $\mathbf{b}$ and find $\mathbf{b}$ which maximises the non-gaussianity of $\mathbf{b}^T\mathbf{x}$ [15].

Unlike for infomax where the random variable had largest variance when the distribution was uniform, here we have variables with equal variance. Then a Gaussian variable has the largest entropy value and we can thus use entropy as a measure of non-Gaussianity. Here we use a normalised version of differential entropy called negentropy, defined as:

$$\mathcal{J}(\mathbf{u}) = H(\mathbf{u}_{\text{Gauss}}) - H(\mathbf{u}),$$

where $\mathbf{u}_{\text{Gauss}}$ is a Gaussian random variable with the same covariance matrix as $\mathbf{u}$ [48]. Hyvärinen (1998) described an approximation of negentropy as

$$\mathcal{J}(\mathbf{u}) = \sum_{p=1}^{P} k_p[E\{G_p(\mathbf{u})\} - E\{G_p(\nu)\}]^2,$$

where k_p are some positive constants, G_p some non-quadratic (contrast) functions

and ν is a standardised Gaussian variable such that $N(\sigma^2 = 1, \mu = 0)$ [14].

Independence by Minimum Nonlinear Correlation: The fourth order cumulant, which indicates higher order covariance, is given by:

$$\text{cum}(u_i, u_j, u_k, u_l) = E(u_i u_j u_k u_l) - E(u_i u_j)E(u_k u_l) - E(u_i u_k)E(u_j u_l) - E(u_i u_l)E(u_j u_k),$$

where u_i, u_j, u_k, u_l are elements of the random vector $\mathbf{u}$ [15]. It can be shown that if the random process $\mathbf{u}$ follows a multivariate Gaussian distribution, all cumulants of order 3 and higher vanish. A contrast function ϕ that uses the sum of squared fourth order cross-cumulants can be constructed as a function of the separating matrix:

$$\phi(\mathbf{W}) = \sum_{ijkl \neq ijkl} \text{cum}(u_i, u_j, u_k, u_l)^2.$$

This contrast function measures the mutual information between cross-cumulants. Making the cumulant matrix as diagonal as possible is equivalent to minimising non-linear correlations. Thus components in $\mathbf{U}$ become statistically independent [48].

A.3 Consensus non-negative matrix factorisation (cNMF)

We will now consider the special case of consensus Non-negative Matrix Factorisation, as described by Kotliar et al. (2019) [21]. R replicates of NMF are run on the same normalised dataset with the same number of components K but with different randomly selected seeds. This results in R instances of the usage matrix $\mathbf{U}^{(r)}$ (N cells $\times$ K programmes) and programme matrices $\mathbf{V}^{(r)}$ (K programmes $\times$ H genes).

Therefore we obtain $\mathbf{U}^{(r)} \times \mathbf{V}^{(r)} \approx \mathbf{Y}$. For each replicate, the rows of $\mathbf{V}^{(r)}$ are normalised to have length 1. The $\tilde{\mathbf{V}}^{(r)}$ are then concatenated vertically into a single $RK \times H$ dimensional matrix, $\mathbf{V}$, where each row is a component from a replicate. Components with high mean Euclidean distance from their L nearest neighbours are filtered out according to the following criteria:

If $D(V_l, L) < \tau$, then $\mathbf{V}^{(\mathbf{f})} = \begin{bmatrix} \dots \\ V_l \\ \dots \end{bmatrix}$ for $l = 1...Rk$, where $L = \rho R$ and

$D(V_l, L) = \frac{1}{L} \sum_{V_n \in N_L(V_l)} ||V_l - V_n||$. V_l is the *lth* row of $\mathbf{V}$, $N_l(V_l)$ is the set of L nearest neighbours of V_l and

$\mathbf{V}^{(\mathbf{f})}$ is the matrix of rows which passed the L nearest neighbours threshold filter. Varying ρ and τ determines which replicate components are filtered out and which are kept. ρ denotes the fraction of NMF replicates to be used as nearest neighbours, whilst τ is a distance threshold that denotes how close a component must be to its nearest neighbours in Euclidean space to be sufficiently matching. The rows are then clustered using K-means clustering with the Euclidean distance

metric, using the same number of clusters as the number of components for the NMF runs [21].

K-means Clustering: K-means clustering is carried out using the following algorithm:

1. Randomly assign a number from 1 to K to each of the observations (these are the initial cluster assignments). Initialise the cluster centroids for the K clusters $(\mu_1, ...\mu_k)$.

2. Repeat until convergence:

   (a) Find the minimum distance from each point i to each cluster j by computing:
   $$c^{(i)} := \arg\min_j ||y^{(i)} - \mu_j||^2$$

   (b) To update the centroid value, for every j set:
   $$\mu_j := \frac{\sum_{i=1}^{m} \mathbb{I}(c^{(i)} = j) y^{(i)}}{\sum_{i=1}^{m} \mathbb{I}(c^{(i)} = j)}$$ [18].

$A_k = \{\text{rows l assigned to cluster k}\}$ contains the indices of the rows of $\mathbf{V^{(f)}}$ assigned to the kth cluster. Each cluster of replicate components is then collapsed down to a single consensus vector by taking the median value for each gene across the components in the cluster:

$$\mathbf{V}_{kj}^{(c)} = \text{median}(\{V_{lj}^{(f)} \text{for } l \in A_k\})$$

for l indexing over the rows of $\mathbf{V^{(f)}}$ and j indexing over the columns (genes), where the median is taken separately for each gene j. This defines a $K \times H$ consensus programmes matrix $\mathbf{V}^{(c)}$, where (c) denotes consensus. The gene expression programme components are then normalised using Manhattan distance. A consensus usage matrix is then fitted using NMF with the component matrix fixed to $\tilde{\mathbf{V}}^{(c)}$, i.e. solving the optimisation

$$\min_{U_{i1}...U_{ik} \geq 0} ||\tilde{Y}_i - \sum_{k=1}^{K} U_{ik} \tilde{V}_k^{(c)}||,$$

where $G_k^{(c)}$ is the H-dimensional normalised consensus programme for the kth gene expression programme, i indices over cells. We are minimising with respect to the U_{ik}, which are constrained to be non-negative. These coefficients are then concatenated and normalised so that the values for each cell sum to 1 (again using Manhattan distance) to generate the consensus usage matrix $\tilde{U}^{(c)}$:

$$\tilde{\mathbf{U}}^{(c)} = \begin{bmatrix} \tilde{U}_{11} & \cdots & \tilde{U}_{1K} \\ \vdots & \ddots & \vdots \\ \tilde{U}_{N1} & \cdots & \tilde{U}_{NK} \end{bmatrix}.$$

Appendix B

Alternative Stemness-scoring Methods

B.1 StemID

StemID is a method used to infer a network of differentiation trajectories between cell clusters and use this to predict stem-cell identity from scRNA-seq data.

B.1.1 Dimensionality reduction

Firstly, the algorithm reduces the dimensions of the transcript counts for each gene into a lower dimensional space in order to maintain only the number of dimensions required to represent all point-to-point distances between cells.

Definition B.1.1. The distance d_{ij} between cells i and j is defined as

$$d_{ij} = 1 - \rho_{ij},$$

where ρ_{ij} is the Pearson's correlation coefficient of the transcriptome of these cells.

B.1.2 Derivation of Differentiation Trajectories

The medoid m_i of cluster i is connected to the medoid m_j of all other clusters $j \in [1, ..., i-1, i+1, ..., N]$.

For each cluster k in cluster i the vector $z_{i,k} = y_{i,k} - m_i$ connecting position $y_{i,k}$ to m_i is projected onto each link $l_{i,j} = m_j - m_i$ between cluster i and j. Projections $p_{k,i,j}$ are calculated using the dot product,

$$p_{k,i,j} = |z_{i,k}| \cdot \cos \alpha_{k,i,j},$$

where the angle $\alpha_{k,i,j}$ is the angle between $z_{i,k}$ and $l_{i,j}$.

The cell is then assigned to the link with the longest projection.

B.1.3 Randomisation of Cell Positions

Randomisation is used to determine if a link has significantly more assigned cells than expected by chance. Randomisation is performed by sampling new cell positions from a uniform interval with boundaries given by the data. Cluster medoids are constant to maintain topology.

For each cluster it is computed how many cells are assigned to each link to another cluster. The distribution of expected cells on each link is sampled by repeating the randomisation 2000 times. For a p-value threshold of $P < 0.01$, 2000 repetitions is sufficient to calculate the 1% quantile with sufficient confidence. The p-value for each link is derived as the quantile of the distribution generated by randomisation corresponding to the actual number of cells on the link.

B.1.4 Link Score

To assess confidence in a particular link, a link score is computed that reflects its coverage by cells.

Definition B.1.2. The *link score* is defined as the maximum difference between two neighbouring cell positions after re-scaling the length to one.

B.1.5 Transcriptome Entropy

Transcriptome entropy E_j of cell j is

$$E_j = \sum_{i=1}^{N} p_{i,j} \log_N p_{i,j},$$

where $p_{i,j} = \frac{n_{i,j}}{N}$. $n_{i,j}$ is the number of transcripts of gene i in cell j. N equals the number of transcripts in each cell (equivalent for all cells due to downsampling).

The median delta-entropy ΔE_k is computed for each cluster k, defined as

$$\Delta E_k = \text{median}_{j \in k}(E_j) - \min_l(\text{median}_{j \in l}(E_j)),$$

where l denotes the link score.

To predict stem cell identity, a stemness-score is calculated for each cluster k, given by:

$$s_k = l_k \cdot \Delta E_k,$$

where l_k denotes the number of significant links of cluster k [11].

B.2 SCENT

SCENT (Single cell entropy) is a method used to infer differentiation potential of cells using scRNA-seq data. SCENT approximates differentiation potency of a single cell by computing the signalling entropy of each cell.

B.2.1 Computation of signalling entropy

The gene expression profile of a sample is integrated with a protein-protein interaction (PPI) network. This utilises the *mass action principle*, i.e. we assume the probability of interaction in a given sample is proportional to the product of the expression values of the corresponding genes in the sample. Thus, the weight of an edge between protein i and protein j is defined as:

$$w_{ij} \propto E_i E_j,$$

where E_i denotes the normalised RNA-seq read count of gene i.

Viewing the edges as signalling interactions, we can define a random walk on the network (assuming that we normalise the outgoing weights of a node to be 1).

This results in a stochastic matrix, P, over the network with entries

$$p_{ij} = \frac{E_j}{\sum_{k \in N_i} E_j}, \forall j \in N_i,$$

where N_i denotes the neighbours of protein i [2].

Definition B.2.1. Given the stochastic matrix p_{ij}, the signalling entropy is defined as the entropy rate over the weighted network, i.e.

$$SR = \sum_{i=1}^{n} \pi_i \sum_{k \in N_i} p_{ik} \log p_{ik},$$

where π is the steady-state probability satisfying $\pi P = \pi$ and $\sum \pi_i = 1$. SR can be divided by the maximum possible SR, $\max SR$, found as described by Teschendorff et al. (2017) to generate a normalised signalling entropy value, $\widetilde{SR}$ with a value between 0 and 1 [40].

B.2.2 Inference of differentiation states

Signalling entropy is hypothesised to be representative of differentiation state of single cells. After normalisation, $\widetilde{SR}$ can be transformed using:

$$y(\widetilde{SR}) = \log_2(\frac{\widetilde{SR}}{1 - \widetilde{SR}}).$$

The optimal number of differentiation states is then estimated as described by Teschendorff et al.(2017), as well as state-membership probabilities for each cell.

Definition B.2.2. For K inferred differentiation states, the normalised local Shannon information for each gene i is defined as:

$$\widetilde{SI} = -\frac{1}{\log K} \sum_{k \in K} p_k \log p_k,$$

where p_k is the probability distribution of the differentiation state k for a population of $N \in \mathbb{N}$ cells.

Clustering is then performed on a filtered set of cells using the partitioning-around-medoids (PAM) algorithm. The cells are filtered to genes which drive most variation (assessed using singular value decomposition). Landmarks are identified using differentiation state-cluster combinations which contained at least 1-5% of single cells. Inference of lineage trajectories is computed using centroids of gene expression, using cells within the landmark and genes used in the PAM clustering.

Appendix C

Table of CellAssign Markers

Below is a table of the different marker genes taken from Clark et al. (2019) which were used to define each cell type in the CellAssign implementation onto Lane 1 of the Lo Giudice et al. (2020) dataset [7, 25]. The final column also gives the number of cells denoted as each of the cell types in the Lo Giudice dataset (or by 'other' if no cell type was assigned).

Cell Type	**Differentiation status**	**Marker genes**	**Number of cells**
Amacrine Cells	Differentiated	*Dlx1, Sox11, Pax6* and *Sox4*	177
Early Progenitor Cells	Undifferentiated	*Fgf15, Ccnd2* and *Sfrp2*	953
Late Progenitor Cells	Undifferentiated	*Hes5* and *Id1*	4
Neurogenic Cells	Partially Differentiated	*Rax, Dll1, Neurog2, Hes6, Olig2* and *Dll4*	189
Photoreceptor Precursors	Partially Differentiated	*Opn1sw* and *Crx*	114
Retinal Ganglion Cells	Differentiated	*Isl1* and *Elavl4*	791
Other	N/A	N/A	239

Appendix D

Top Genes Tables

The genes in the following tables are the highest weighted transcription factor genes in each factor obtained from NMF.

For the Freeman-Tukey transformed data, NMF with 6 factors returned the following top genes for each factor:

idx	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6
1	*Klf7*	*Hmgb1*	*Hmgb2*	*Id3*	*Otx2*	*Id3*
2	*Ebf1*	*Id3*	*Hmgb1*	*Fos*	*Rorb*	*Klf7*
3	*Isl1*	*Sox4*	*Klf7*	*Hes1*	*Sox11*	*Hmgb1*
4	*Zfp36l1*	*Zfp36l1*	*Pax6*	*Sox4*	*Neurod1*	*Rorb*
5	*Sox11*	*Egr1*	*Tsc22d1*	*Id2*	*Id3*	*Ssrp1*
6	*Pou6f2*	*Hmgb2*	*Rorb*	*Rorb*	*Sox4*	*Sox4*
7	*Pou4f1*	*Klf7*	*Tcf25*	*Zeb2*	*Ssrp1*	*Mis18bp1*
8	*Otx2*	*Zfp207*	*Id2*	*Carhsp1*	*Hmgb1*	*Zeb2*
9	*Onecut2*	*Meis2*	*Id3*	*Zfhx3*	*Zfp36l1*	*Ybx1*
10	*Id2*	*Tfdp2*	*Meis2*	*Cnbp*	*Klf7*	*Pbx1*

For the square-root transformed data, NMF with 7 factors returned the following top genes for each factor:

idx	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
1	*Ebf1*	*Neurod1*	*Sox11*	*Hmgb1*	*Hmgb1*	*Hmgb2*	*Sox11*
2	*Isl1*	*Neurod4*	*Myef2*	*Rorb*	*Fos*	*Hmgb1*	*Pax6*
3	*Klf7*	*Otx2*	*Rorb*	*Sox11*	*Hes1*	*Mis18bp1*	*Rorb*
4	*Sox11*	*Meis2*	*Hmgb1*	*Cnbp*	*Rorb*	*Rorb*	*Sox4*
5	*Pou6f2*	*Thrb*	*Hsf2*	*Hmgb2*	*Zfp36l1*	*Hmgb3*	*Hmgb1*
6	*Hmgb1*	*Hmgb1*	*Gatad1*	*Hmgb3*	*Hmgb2*	*Pax6*	*Tsc22d1*
7	*Tsc22d1*	*Rorb*	*Zfp207*	*Neurod1*	*Id3*	*Ssrp1*	*Onecut2*
8	*Pou4f1*	*Crx*	*Klf13*	*Carhsp1*	*Cnbp*	*Cnbp*	*Carhsp1*
9	*Pax6*	*Cnbp*	*Khsrp*	*Rbpj*	*Id2*	*Sox11*	*Hmgb3*
10	*Tcf25*	*Onecut2*	*Ybx1*	*Zfp422*	*Pax6*	*Id2*	*Cnbp*

For the scaled Freeman-Tukey transformed data, NMF with 9 factors returned the following top genes for each factor:

idx	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8	Factor 9
1	*St18*	*Pias1*	*Ebf3*	*Tsc22d1*	*Id3*	*Id3*	*Egr1*	*Id2*	*Klf7*
2	*Lcorl*	*Rorb*	*Id2*	*Pax6*	*Sox4*	*Sox4*	*Zfp36l1*	*Hmgxb4*	*Pou6f2*
3	*Crx*	*Zeb1*	*Hmga1*	*Hmgb2*	*Zic5*	*Hes1*	*Hmgb1*	*Ikzf5*	*Onecut2*
4	*Rxrg*	*Sox11*	*Neurod4*	*Zfp260*	*Zeb2*	*Zfhx3*	*Id3*	*Hmga1*	*Ebf1*
5	*Khsrp*	*Atf4*	*Gpbp1l1*	*Tcf25*	*Tmf1*	*Tcf4*	*Dach1*	*Gpbp1l1*	*Isl1*
6	*Nr2f2*	*Zic5*	*Zfhx4*	*Mysm1*	*Zeb1*	*Fos*	*Zfp207*	*Crx*	*Zfp36l1*
7	*Kmt2c*	*Ssrp1*	*Sox4*	*Klf7*	*Six3*	*Hbp1*	*Sox4*	*Zfhx4*	*Pou4f1*
8	*Neurod1*	*Khsrp*	*Id3*	*Hmgb1*	*Crx*	*Cnbp*	*Zfp266*	*Ebf3*	*Zbtb20*
9	*Neurog2*	*Tsc22d3*	*Sox11*	*Meis2*	*Neurog2*	*Gatad1*	*Ikzf5*	*Tfap2d*	*Ybx1*
10	*Tmf1*	*Crx*	*St18*	*St18*	*Dlx1*	*Plagl1*	*Meis2*	*Pias1*	*Mis18bp1*

For the scaled square-root transformed data, NMF with 7 factors returned the following top genes for each factor:

idx	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
1	*Ebf1*	*Thrb*	*Myef2*	*Hmgb1*	*Hmgb1*	*Hmgb2*	*Pax6*
2	*Isl1*	*Neurod4*	*Hsf2*	*Cnbp*	*Cnbp*	*Hmgb1*	*Rorb*
3	*Klf7*	*Crx*	*Gatad1*	*Rorb*	*Zfp36l1*	*Mis18bp1*	*Sox4*
4	*Pou6f2*	*Meis2*	*Sox11*	*Hmgb3*	*Hes1*	*Cenpa*	*Sox11*
5	*Tsc22d1*	*Neurod1*	*Rorb*	*Rbpj*	*Id3*	*Hmgb3*	*Tsc22d1*
6	*Ebf3*	*Otx2*	*Khsrp*	*Carhsp1*	*Rorb*	*Foxn4*	*Hmgb1*
7	*Tcf25*	*Prdm1*	*Klf13*	*Sox11*	*Fos*	*Terf1*	*Onecut2*
8	*Pou4f1*	*Tulp1*	*Zbtb44*	*Neurod1*	*Id2*	*Rorb*	*Tfap2b*
9	*Zfhx3*	*Lhx4*	*Tmf1*	*Zfp207*	*Plagl1*	*Ssrp1*	*Carhsp1*
10	*Hmgb1*	*Lcorl*	*Zbtb21*	*Zfp422*	*Hmgb3*	*Ascl1*	*Hmgb3*

Appendix E

Cophenetic Correlation Coefficient

The Cophenetic Correlation Coefficient is an alternative method to determine the number of factors for NMF. Since the NMF algorithm may not converge to the same solution depending on the random initial seed, we can use this to check how strong the clustering is for each number of factors. If a clustering into k classes is strong, we would expect that sample assignment to clusters would vary little from run to run.

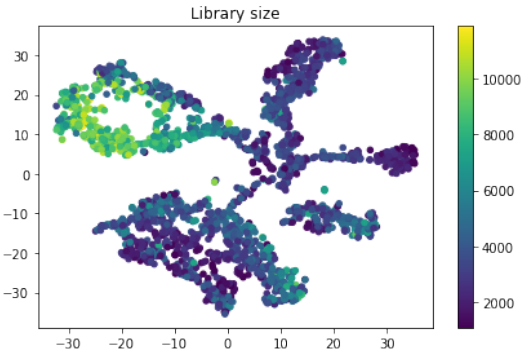
For each run, the sample assignment can be defined by a connectivity matrix C of size $M \times M$, with entry $c_{ij} = 1$ if samples i and j belong to the same cluster, and $c_{ij} = 0$ if otherwise. The consensus matrix $\bar{C}$ is the average connectivity matrix over many clustering runs (run until $\bar{C}$ is stabilised). The entries of $\bar{C}$ range from 0 to 1 and reflect the probability that samples i and j cluster together. If a clustering is stable, the entries of $\bar{C}$ will be close to 0 or 1. The dispersion between 0 and 1 thus measures the reproducibility of the clusters with respect to random initial conditions. By using the off-diagonal entries of $\bar{C}$ as a measure of similarity among samples, we can use average linkage to reorder the samples of $\bar{C}$.

The Cophenetic Correlation Coefficient $\rho_k(\bar{C})$ gives a quantitative measure of stability for each value of k by indicating the dispersion of the consensus matrix $\bar{C}$. ρ_k is computed as the Pearson correlation of two distance matrices: the first, $I - \bar{C}$, is the distance between samples in the consensus matrix, and the second is the distance between samples in the linkage used in the reordering of $\bar{C}$. In a perfect consensus matrix (all entries = 0 or 1), the Cophenetic Correlation Coefficient equals 1. When the entries are scattered between 0 and 1, the Cophenetic Correlation Coefficient is < 1. We observe how ρ_k changes as k increases. We select values of k where the magnitude of the Cophenetic Correlation Coefficient is maximised [4].

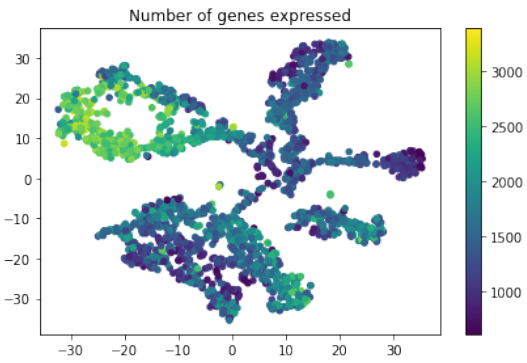
Appendix F

Quality Control Plots

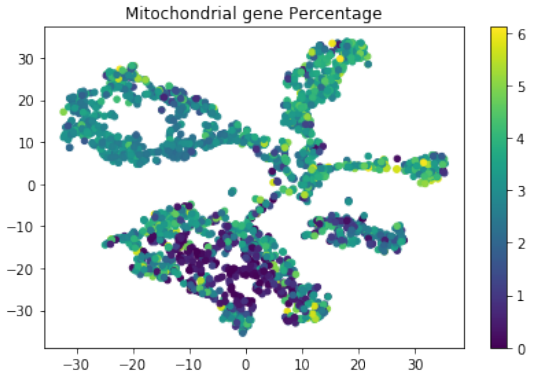
The Quality Control metrics described in Chapter 2 are used here to colour t-SNE plots for Lane 1 of the Lo Giudice dataset (Figure F.1). The largest library sizes and numbers of genes expressed are found in the area of the plot corresponding to Retinal Ganglion cells (the most differentiated cells in the dataset).



(a) t-SNE plot coloured by Library size of each cell.



(b) t-SNE plot coloured by number of genes expressed in each cell.



(c) t-SNE plot coloured by the percentage of genes expressed which are mitochondrial genes in each cell.

Figure F.1: Lo Giudice data lane 1 t-SNE plots coloured by Quality Control metrics.

Appendix G

Code Appendix

The full details of the code used in this project can be found at the GitHub repository linked below:

`https://github.com/VallejosGroup/RachelJacksonHonoursProject`