Computational methods for cytometry

by

Alice Yue

M.Sc., Simon Fraser University, 2017
B.Sc., University of Victoria, 2015

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2022

Declaration of Committee

Name:	**Alice Yue**
Degree:	**Doctor of Philosophy**
Thesis title:	**Computational methods for cytometry**
Committee:	**Chair:** Kay C. Wiese Professor, Computing Science
	Maxwell W. Libbrecht Supervisor Assistant Professor, Computing Science
	Cedric Chauve Committee Member Professor, Mathematics
	Ryan R. Brinkman Committee Member Adjunct Professor, Computing Science
	Martin Ester Examiner Professor, Computing Science
	Cynthia Guidos External Examiner Professor, Immunology University of Toronto

Abstract

Flow cytometry (FCM) is a high-throughput single-cell biotechnology commonly used to study the immune system in clinical and research settings. We present solutions to two problems in an FCM data analysis pipeline. The first problem is to identify cell populations within FCM samples. The second problem is to pinpoint the biomarkers or cell populations that can be used to help classify FCM samples (e.g. diseased vs healthy).

This thesis covers topics in computational biology and is intended for readers with basic knowledge in the field.

Keywords: Computational biology; flow cytometry; single-cell data

Dedication

To my mother.

Acknowledgements

I would like to show gratitude to my committee (supervisors: Cedric Chauve, Maxwell Libbrecht, Ryan Brinkman; defence chair: Kay Wiese; defence internal examiner: Martin Ester; defence external examiner: Cynthia Guidos). Words cannot express how thankful I am for your time, dedication, and for being role models in the field.

Also big hugs and appreciations to my family, friends, and labmates for your warm companionship and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cytometry is a biotechnology that analyzes single cells and is routinely used in the research and diagnosis of diseases of the immune system, such as leukemia and lymphoma [366]. As such, we will discuss cytometry in the context of the immune system (i.e. biological samples we use will contain immune cells from organs such as the spleen, bone marrow, and blood [169]).

In this thesis, we present solutions to two problems in the analysis of cytometry data: 1) identify cell populations within cytometry samples and 2) find biomarkers or cell populations that can be used to help classify cytometry samples (e.g. diseased vs healthy).

1.1 Cytometer

A cytometer is a high-throughput apparatus capable of simultaneously measuring multiple cell characteristics or *features* per single cell for all cells in a given biological sample [57, 58]. Users of cytometry harness this single-cell data to sort cells [86] into their respective cell type or cell population based on the proteins each cell contains. The assumption is that we can identify the cells in each cell population by the presence or absence of a unique combination of proteins. The end goal of cell sorting is to evaluate the immune condition of the subject by analyzing the cell population composition of their cytometry sample.

1.1.1 Flow Cytometry (FCM)

Flow cytometry (FCM) is a high throughput technology that uses the flow cytometer machine to measure cell features as the intensity of fluorescence (light of a certain wavelength or colour) emitted by the fluorochromes in the markers on the cells in a given processed sample. To prepare this sample, users first combine the sample with a cocktail of markers. *Markers* contain antibodies that attach to a certain target protein on the cells in the sample. These markers also contain fluorochromes that emit fluorescence upon stimulation. Hence, the different markers, or coloured light, present on a cell represent the types of proteins the cell contains. Once in the flow cytometer, the cells in this processed sample are aligned

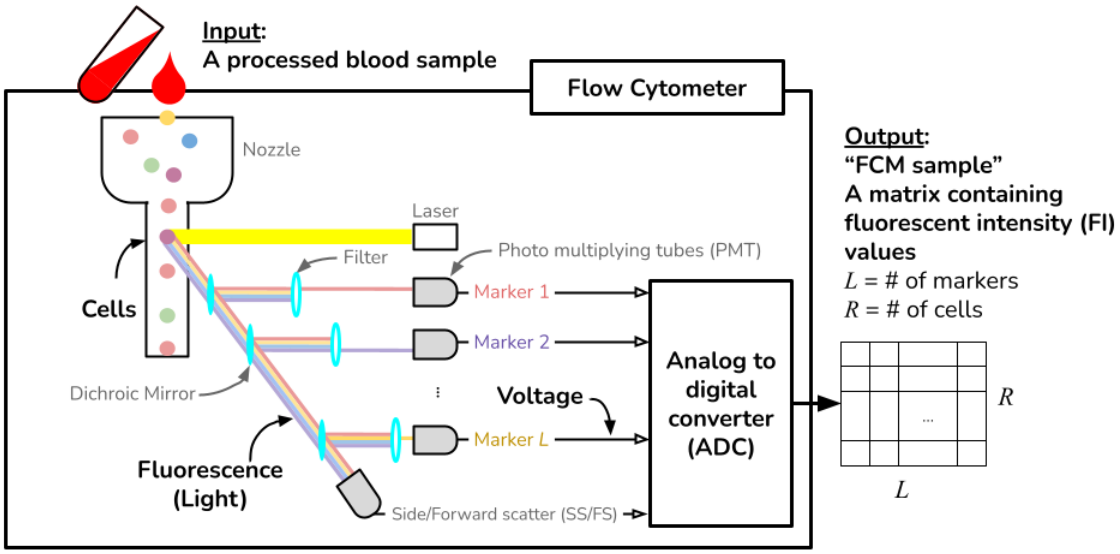

Figure 1.1: FCM machinery and how it analyzes biological samples. The data used to create the scatterplot in this figure is from [2].

into a single file stream by sheath fluid in a component of the machine called the flow cell. After the cells are focused, they are passed through a laser one at a time. Stimulated by the laser, the fluorochrome on the markers on each cell emits fluorescence. This fluorescence is filtered by filters and then detected by an array of photo multiplying tubes (PMT) — each measuring light of a certain range of wavelength.

For a given sample, the flow cytometer outputs the brightness of detected fluorescence as continuous *fluorescence intensity (FI)* values. In addition to markers, flow cytometers also detect a cell's physical characteristics including its size or forward scatter (SS) and granularity or side scatter (SS). These, together with the FI values, are given to the user as a file in the Flow Cytometry Standard (FCS) format [335]. This file includes a $R \times L$ matrix where R is the number of cells, L is the number of markers and physical characteristics. We will refer to the latter dimension as markers for brevity. Usually, a biologist would analyze a single biological sample using different *panels*, or sets of markers. This would result in multiple matrices per sample, one per panel. Currently, a flow cytometer is capable of measuring 10,000 cells per second and can handle 40 marker panels [274]. However, 12-15 marker panels are still most commonly used [250].

We also acknowledge that FCM can be used to analyze non-cellular particles such as individual proteins, and RNA (when no available marker attaches to a protein target of interest but fluorescent in situ hybridization can be conducted with a corresponding RNA transcript) [250].

First commercialized in 2009, DVS Sciences marketed another variation of the FCM called mass spectrometry. While FCM detects the FI, mass spectrometry uses a mass spectrometer to detect the mass-to-charge ratio of ionized chemical species on markers. Theoretically, mass spectrometry is more precise because a single chemical species can only be detected as a single mass-to-charge ratio value. In contrast, fluorescence emitted by a single type of fluorochrome can be detected on a range of wavelengths. Another advantage of mass spectrometry over FCM is the availability of commercial solutions that can measure larger panels (40+ markers) than FCM [337, 250]. However, the drawback of mass spectrometry is that it can only analyze up to 1,000 cells per second [256].

In 2012, Sony commercially released another variant of the flow cytometer called the spectral flow cytometer [100]. While conventional flow cytometers use optical filters to filter light into desired ranges of wavelength, spectral flow cytometers use a prism to fragment light across the full emission spectrum. This means that the former requires an array of PMTs to detect each range of wavelength as opposed to being able to use a single detector. This also means that the user would not have to manually configure PMTs based on the markers they are using.

The mass, spectral, and conventional flow cytometer all output their data in the FCS format. As FCS data can be analyzed the same way, we will refer to FCS data produced by any cytometer as *FCM samples*.

1.1.2 FCM sample considerations

While we assume that the information in any given FCM sample to be our ground truth (e.g. contains the correct count of cells in the actual biological samples and the FI value always correlate with the number of a certain protein on a cell), in reality, they are approximations of and can deviate from the ground truth or what is actually in the biological sample. We discuss these nuances in the context of the traditional flow cytometer.

Before biological samples can be put into the flow cytometer, the machine needs to be calibrated on a monthly to annual basis to ensure accurate markers [250]. One type of calibration is to use calibration and size reference beads, for which we know the true size and granularity. After putting these beads into the flow cytometer, the machine's parameters are calibrated until the output FS and SS are what we know the beads to be.

While we mention in our experiments that cell counts are given, during an experiment, additional steps need to be taken to measure absolute cell count. One example is to use cell counting beads. These are given as a suspension of microspheres (i.e. the "beads") at a known concentration. These microspheres emulate cells (e.g. lymphocytes), have a known FS and SS, and emit known fluorescence. This suspension is added to a biological sample such that the ratio of microspheres to sample volume is known. This way, we can calculate cell concentration by inferring the sample volume from the number of microspheres analyzed by the flow cytometer [75, 271]. More specifically, the concentration of cells per µL Z in a

biological sample can be calculated as:

$$Z = \frac{AC}{BD}$$

where A is the number of beads added to the sample and B is the total volume of the sample. In addition, C and D are the cell count and the bead count given by the flow cytometer output respectively.

Upon obtaining the raw FCM sample, another variable we have to control for is the FI values. For every experiment, the FI value at which it is considered 'bright' or 'positive' may be different and need to be determined by the user. This is a non-trivial problem. Antibodies on the markers should bind to proteins on a cell in a desired 'specific' manner via its antigen-binding site. However, they can also bind to a cell's endogenous Fc receptors (a type of protein present on e.g. macrophages and neutrophils) or bind 'nonspecifically' via ionic and hydrophobic interactions. The latter two bindings result in false-positive signals and need to be excluded. To do so, one can use FMO (fluorescence minus one) control samples. These samples contain all markers except for the one being controlled for such that they serve as true negative samples [300]. For example, if we want to find a threshold for three markers, we need to prepare three biological FMO samples where the first, second, and third samples contain all markers except the first, second, and third markers respectively. Plotting the cells from the first sample on the first marker, we take the 99th percentile FI value as the threshold that separates cells associated with FI values that are positive and negative for the first marker. The same can be done for the other two markers. Another solution is to use biological controls; for example, one type of biological negative control is a sample whose cells do not contain the protein of interest. An alternative negative control is a sample where the desired binding site of its cells is blocked by the same marker without fluorescence. Another method to reduce undesired bindings is by finding the right concentration of markers to mix in with a sample. Determining the right concentration of markers is also key for increasing the signal-to-noise ratio, reducing costs (by not overusing markers), and producing more true positive signals [75]. For any marker, we can find the best concentration via titration. This process starts by mixing multiple samples with different concentrations of the marker of interest. All samples contain cells that are positive and cells that are negative for the said marker. For each sample, we calculate how well separated the two groups of cells are from each other in terms of their FI. This quality is quantified using the stain index:

$$\frac{MFI_p - MFI_n}{2SD}$$

where MFI_p and MFI_n are the mean FI of the cells that are positive and negative for the marker, respectively. SD is the standard deviation of FI values for the cells that are negative for the marker. The chosen concentration would have the highest stain index.

1.2 FCM Bioinformatics

FCM bioinformatics appeared as a new sub-field of bioinformatics in the early 2000s [308, 267], focused on computationally storing, organizing, and analyzing high-dimensional FCM samples. Manual analysis is currently the norm leaving room for human errors, subjectivity, and variability [49, 308, 105, 166, 267, 237]. These, and recent advances in FCM sample standards [335], dissemination routes [334], analytical platforms [333, 65], and benchmark data sets [5, 6, 124], have driven scientists to develop an increasing amount of computational tools to complement and possibly replace manual FCM sample analysis.

1.3 Data analysis pipeline, motivations, and contributions

Taking an FCM sample as input, we put it through the data analysis pipeline outlined below.

1. Preprocess data: Clean and transform data for downstream analysis (see Section 2.1).
2. **Problem 1**: Cell population identification: Classify the cells in each sample into their respective cell populations.
3. **Problem 2**: Biomarker identification: Identify cell populations that can act as biomarkers that help classify samples (e.g. from healthy vs diseased subjects).

This thesis addresses Problems 2 and 1 in that order. As an overview of this thesis:

1. Chapter 2 details each step of the FCM data analysis pipeline and how it incorporates the problems and methods we developed — which will be described in subsequent chapters.
   (a) Chapter 3 is a comprehensive depth review of clustering-based methods for FCM and sc/RNAseq data sets (towards Problem 1).
   (b) Chapter 4 presents a method and a novel numerical metric to address Problem 2.
   (c) Chapter 5 addresses Problem 1 by proposing a new method for cell population identification. This method will show that it is possible to identify cell populations such that the results are visually interpretable.
   (d) Chapter 6 proposes a second method for Problem 1. Chronologically, this method is the most recent method being developed during the writing of this thesis.
2. Chapter 7 provides conclusive remarks and future work.

This work furthers our capacity to analyze single-cell FCM samples and creates opportunities to investigate problems such as identifying human immunodeficiency diseases of unknown origin.

Chapter 2

Data analysis pipeline and background

2.1 Preprocessing

Given an FCM sample (i.e. its $R \times L$ (cell $\times$ marker) matrix with FI values), we apply the following preprocessing protocols. We refer readers to [290] for more details.

Compensation is a first step that aims to reclassify FI values that were assigned to the incorrect marker. This error arises because fluorescence is detected on a range of wavelengths. For example, a yellow coloured fluorescence on marker A could be detected as a light or dark yellow with an orange hue. If the same FCM detects an orange coloured fluorescence on marker B as a light yellowish-orange, then the FCM could misclassify these FI values to the wrong marker. Continuing our example, the dark yellow and light orange FI could be misclassified as fluorescence from markers B and A respectively.

After compensation, we clean the data by removing cells with maximum or negative FS and SS values. This is because 'cells' with abnormally small or large FS and SS values may be debris or large non-biological particles.

Next we transform our FI values to amplify signals in our data for downstream analysis. We use the logicle transform [275] which is the parametrized biexponential function:

$$S(x; a, b, c, d, f) = a \cdot exp(bx) - c \cdot exp(-dx) + f$$

which is a generalization of the hyperbolic sine function:

$$sinh = \frac{exp(x) - exp(-x)}{2}$$

where x is the FI value to be transformed. We chose logicle transform because it spreads data out like a log transform to amplify smaller signals which may otherwise be missed. However, unlike log, logicle transform maintains near-linear scales around 0 such that those signals remain detectable [275].

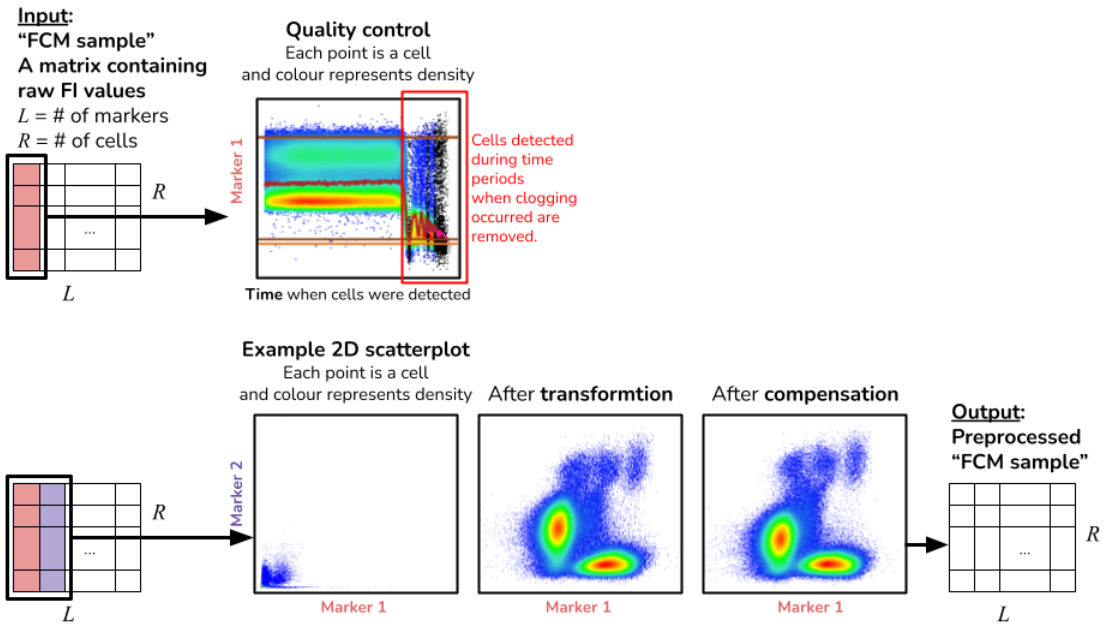

Figure 2.1: FCM data preprocessing steps visualized: quality control, transformation, and compensation. The data used to create the scatterplot in this figure is from [2].

For quality control, we use flowClean [108], a freely available tool that deletes anomalies in our FCM samples. Anomalies occur when the flow of cells through the cytometer is interrupted by clogging, back-pressure, and other machine-related issues. During these periods, there would either be a lack of data or particles would be clumped together and cannot be separated for downstream single-cell analysis.

2.2 Problem 1: Cell population identification

After obtaining a preprocessed FCM sample, we can proceed to solve Problem 1. A *cell population* is a group of cells that have similar FI values for the same group of markers. Accurate identification of cell populations in FCM is imperative for information discovery in real-world applications. For Chapter 4, we use the method specified at the end of this section for cell population identification; later, we introduce alternative ways to do so in Chapters 5 and 6.

2.2.1 Manual Gating

Gating is the process of drawing *gates* or borders around cells of the same cell population on 2D scatterplots. Each point on these plots represents a cell plotted on two markers. Threshold (1D) gates specify a FI threshold value that classifies cells as having an FI that is greater than, or, less than or equal to this threshold. Polygon (2D) gates are polygons on the scatterplot that enclose a group of cells of the same cell population. An upside to manually

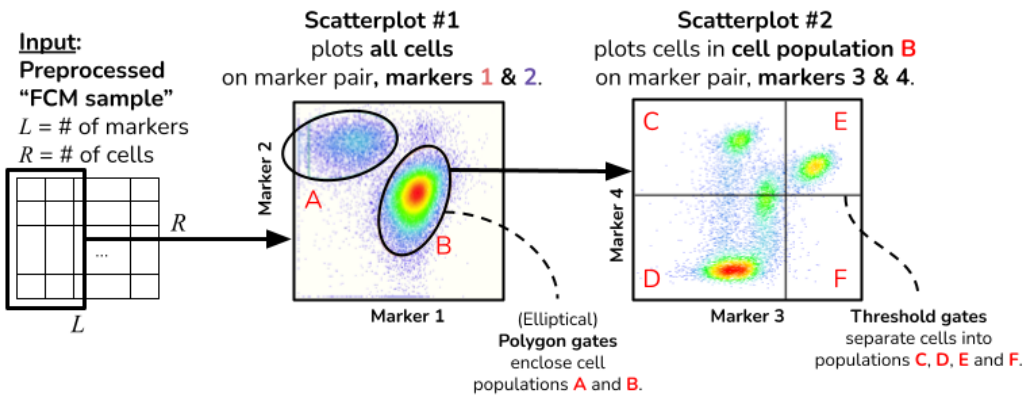

Figure 2.2: A toy example of a gating strategy which gates cell populations A-F from a given preprocessed FCM sample. This gating strategy contains two scatterplots, on which there are two elliptical polygon gates, and two threshold gates.

identifying cell populations is that the user has full control over which cell populations they are trying to find. The user is also able to interpret the results by incorporating their experience and knowledge schema into their gating. After a cell population is gated on a scatterplot, the cells in this cell population can be isolated and further plotted on another marker pair to be further gated. The instruction manual that specifies which marker pairs each cell population should be plotted on and how each scatterplot should be gated is called a *gating strategy* (Figure 2.2). Everything on this gating strategy is specified by an experienced human expert determined by domain knowledge. Therefore, in this thesis, when we conduct gating, we assume that all marker pairs and gates are given by a human expert via the gating strategy and that all cell populations found are known cell populations.

Gating produces interpretable results that aid users in drawing biological conclusions. We say that the results of cell population identification are *interpretable* if:

1. Cell clusters are presented in a way that has already been or allows humans to easily label a group of cells as a known cell population and

2. Results are presented such that the motivation for the resulting cell grouping is clear; this way, the user can determine whether or not they want to keep or discard the results.

For example, manual gating satisfies these two requirements because I) humans define the gates according to their experience on where cell populations should be positioned on a scatterplot and II) the gates are displayed on 2D scatterplots, as opposed to being in high-dimensional space. For II), we acknowledge that there are dimensionality reduction algorithms but their results reflect only limited aspects of the original data. For example, a

t-SNE (t-distributed stochastic neighbour embedding) [229] plot only preserves the distance between points and not the actual values of the points.

2.2.2 Related works

Given how interpretable manual gating results are, it has not been replaced by the over 50 more efficient computational cell population identification algorithms that have been developed [201, 383]. We list these methods in Table 2.1.

Unsupervised clustering methods

Unsupervised clustering methods cluster or group cells with similar FI for similar groups of markers, usually directly in high-dimensional space. These methods are 'unsupervised' because they do not require the user to provide *training* or reference samples whose cell populations have already been identified by a human expert. These methods are great for discovering cell population clusters that satisfy some criteria for natural cluster structure in FI space. Examples of cluster structure criteria are that the clusters need to be convex, have high-density centres, or conform to a certain statistical distribution. A pro of using clustering is that almost everything is automated and there is a chance that the algorithm can isolate new unknown cell populations. However, one issue is that some desired cell populations may not conform to the defined criteria and therefore cannot be found. Another issue is that with fully automated methods, its results cannot be easily incorporated into the user's knowledge schema. For example, the user may not be able to interpret which cluster comprises cells that belong to a certain cell population of interest. Researchers have attempted to resolve this by adding extra post-processing steps. For example, clusters can then be manually validated and re-gated onto 2D scatterplots one cell population at a time using tools such as GateFinder [9] (for polygon gates), HyperGate [34] (for rectangular gates), and C2G (cluster to gate) [395]. By replicating the manual gating retrospectively, and providing local low-dimensional visualizations of clusters, human experts can more easily interpret and provide semantics for each cluster. Other post-processing procedures include dimension reduction for visualizing all clusters (i.e. global high-dimensional visualization) and matching clusters between FCM samples such that the shared clusters can inherit some human given semantic (e.g. a common cell population label). While these workarounds improve interpretability, having to use multiple tools together poses accessibility challenges. Furthermore, the more steps there are, the higher the risk of errors, as small errors at each step can be propagated to downstream analysis (i.e. increased degrees of freedom). See a comprehensive review of clustering methods in Chapter 3

Supervised classification and automated gating methods

Most supervised classification methods also directly classify cells in high-dimensional space. These methods are called 'supervised' because they require and use training samples to

Table 2.1: Methods for cell population identification in FCM.

Tool	Type
ACDC [206] CytOpT [114] DeepCyTOF [211] Deep learning with transformers [388] flowLearn [227] Linear discriminant analysis [1] Multi-layer perceptron [111, 386]	Supervised Classification
Discriminative gate learning [171] FlowDensity [240] Mondrian process [170] OpenCyto [104]	Supervised learning to automate gating
bayesFlow [177] CCAST [16] CytometricFingerprinting [301] FLAME [284] FlowClust [223, 103] FlowGM [66] flowMatch [27] flowMeans [4] ImmunoCLUST [332] phenoGMM [306] Sequential Dirichlet process mixtures [148] SWIFT [265]	Unsupervised Clustering (Mixed-model-based)
ACCENSE [320] ClusterX [63] DensVM [33] FLOCK [285] auto/FlowGrid [96] floptics [338] FlowPeaks [121] Misty Mountain [342]	Unsupervised Clustering (Density-based)
CLARA [336] SamSPECTRAL [404] PhenoGraph [209] X-shift [310]	Unsupervised Clustering (Graph-based)
ASPIRE [89] BayesFlow [177]	Unsupervised Clustering (Bayesian-based)

train a classifier that learns to take a new unlabelled *testing* FCM sample as input and outputs the cell population label of each cell in the sample. A challenge with these methods is to simultaneously obtain high accuracy results, use only one or few training samples, and maintain interpretability of results. Early methods of supervised learning methods are neural networks that take the FI values of a single cell as input and output how likely it belongs to each of the pre-specified cell populations. For example, [111] used 2 layer Kohonen maps, [386] used a 3 layer multi-layer perceptron, and, more recently, [211] experimented using a 5 layer neural network. While the first two methods require abundant training samples, [211], [388], and [114] mitigates this requirement by aligning training sample data distributions to those of the training samples. This strategy is called domain adaptation. One downside of this strategy is that it assumes that the relative abundance of cells in each cell population is consistent across samples — this is not always the case. ACDC [206], however, does not have this con, but assumes and requires the user to provide information on whether their desired cell populations have a high or low FI value for each marker — it assumes that all cell populations can be segmented using threshold gates, which is also not always the case. Conversely, the CyTOF linear classifier [1] assumes that all cell populations are convex clusters. It models the FI values of cells in the same cell population as a normal distribution and uses a nearest median classifier to assign cells to their corresponding cell population.

While manual gating is labour intensive, its results are visually interpretable and give users control over which cell populations are identified — exemplifying these as drawbacks of existing computational methods for cell population identification [308]. In response, [240] and [104] proposed to computationally replicate the process of manual gating. Subsequently, [227] and [171] further automated this process by using supervised methods to gate cells with threshold gates in 1D.

Supervised vs. unsupervised cell population identification

In this thesis, we explore two automated gating methods to identify cell populations in a supervised manner. The reason for this is that in FCM, gating strategies are well-established and cell populations of interest are already known. Flow cytometry was first invented in the 1960s and formally described in 1972 [47]. While clustering methods were proposed by the 1950s for applications such as protein, RNA, DNA, and phylogeny analysis [328], computational biology only started taking off in the 80s [119] when desktop computers became prominent. However, outside of a few exceptions [111], we only started seeing computational flow cytometry papers being published after a dramatic increase in the number of markers that can be measured simultaneously (up to 18) in the 2000s [267]. By then, human experts have already amassed a repeatedly tested, and widely recognized knowledge base of how specific cell populations should be gated and how gating strategies should be designed [366]. This is in contrast to cell population identification in scRNAseq (single-cell RNA sequencing) data which predominantly uses unsupervised bi/clustering methods or proba-

bilistic methods that refer to databases that know how much of each transcript is present in a specific set of cell types [403, 20]. See examples of scRNAseq clustering methods and their commonalities/differences with flow cytometry clustering methods for cell population identification in our depth survey Chapter 3.

2.3 Cell hierarchy

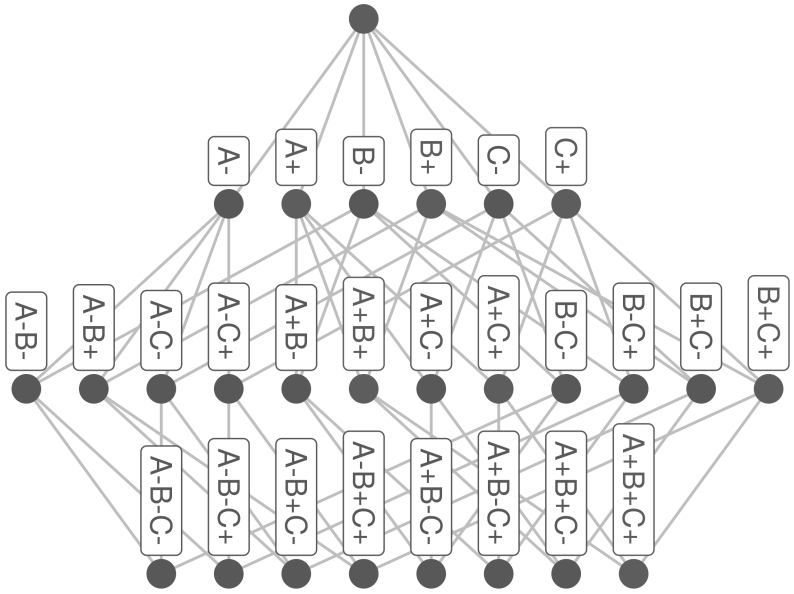

Figure 2.3: An example of a cell population hierarchy representation of an FCM sample and its cell populations defined by markers A, B, and C.

After obtaining the cell populations, we visualize the relationships amongst cell populations using the cell population hierarchy. A *cell population hierarchy*, or cell hierarchy for short, is a directed acyclic graph where nodes represent cell populations and edges represent the relationship between cell populations (Figure 2.3). In the cell hierarchy, a cell population is a set of cells with similar FI values for a set of $0 \leq \ell \leq L$ markers. We define a marker condition as a combination of a marker and a positive$^+$ or negative$^-$ expression indicator. For example, A^+B^- contains two marker conditions, A^+ and B^-, and represents a cell population whose cells have FI greater and less than the given thresholds for markers A and B respectively. We define the ℓ'th layer of the cell hierarchy as the set of all nodes whose label contains exactly ℓ unique marker conditions. It follows that a cell hierarchy has $L + 1$ possible layers with the 0th layer containing the root cell population comprising all cells.

In the cell hierarchy, each edge points from a 'parent' cell population to its 'child' sub-population defined by the addition of one marker condition. For example, if there are three markers A, B, C, then there are edges from the node representing the cell population labelled A^+ to the nodes labelled A^+B^+, A^+B^-, A^+C^+, and A^+C^-.

2.4 Problem 2: Biomarker identification

In clinical environments, a typical use case for FCM samples is to differentiate whether they came from a healthy or diseased person. The problem of making a diagnosis is categorized as a supervised classification problem because we know what classes we are looking for: healthy vs diseased. We will refer to these sample groups as *classes* containing the binary classes 'control' and 'experiment(s)' (e.g. healthy vs diseased) with the latter containing one or more classes based on the number of experiments done.

In exploratory research, we often do not know what effects the experiment has on immune cell populations; therefore, we need to conduct biologically meaningful unsupervised clustering of FCM samples. Sorting samples into naturally occurring groups helps us determine whether our experiments had: 1) no effect (are the same as control FCM samples), 2) significant effect(s), or 3) significant effect(s) that are similar or different from other experiments. More details can be found in [399].

Once we have solved Problem 1 and labelling of FCM samples, we need to identify *biomarkers*, driver cell populations whose features differentiate between the identified sample classes. For example, a biomarker could be the cell population that has significantly more cells in the FCM sample from a diseased patient than one from a healthy individual. We can then use that cell population as a biomarker to indicate whether or not a person is diseased. We propose a method to do this in Chapter 4.

Chapter 3

Clustering in flow cytometry and sc/RNAseq

This chapter is a comprehensive review of clustering methods for cell population identification. On top of FCM, we will also focus on clustering methods in sc/RNAseq data to identify cell populations (scRNAseq) and group samples (RNAseq). For sc/RNAseq, our input is a matrix with cells (scRNAseq) or samples (RNAseq) on the rows. On the column, we have the genes. Inside the matrix are the expression of, abundance, or read count of RNA transcripts associated with each gene. Since the features for sc/RNAseq are both transcripts, the clustering methods used in RNAseq can and have been adapted for cell population identification in scRNAseq. The reason we chose these two data types is that they are major contributors to big data in bioinformatics [244]. As the rows and columns in FCM and sc/RNAseq samples are different, we will refer to them as *biological objects* (or cells for FCM and scRNAseq, and samples for RNAseq) and *features* respectively. We also use the term point interchangeably with object to fit with the context of the methods described.

We organize the contents of this review as follows. First, this section briefly introduces the concepts of clustering, common clusering algorithms, and how data is preprocessed in preparation for clustering. Next, we split off into two sections, one for each data type. In each of these chapters, we talk about the strategies different methods use to cluster these data sets.

3.1 Preliminaries

Clustering is the process of grouping similar objects such that the objects in the same group or cluster have similar feature values and those in different clusters have dissimilar feature. More formally, clustering clusters biological object vectors, each referred to as $x_i \in^{\mathrm{m}}$ where $i \in \{1, ..., n\}$. Another way to represent clustering is through C, a function that maps each

x_i to a cluster

$$C : x_i \rightarrow \{w \in \mathbb{R} | 0 \leq w \leq 1\}^K$$

$$\sum \{w \in \mathbb{R} | 0 \leq w \leq 1\}^K = 1$$

where K is the number of clusters, with each cluster labelled with an index $1, .., k, ..., K$. If each object belongs to one cluster, we call this *hard clustering* and $C(x_i) = \{w \in \mathbb{R} | 0 \leq w \leq 1\}^K$ would contain one 1 and $K-1$ 0's. The position of 1 represents the cluster index k that x_i belongs to i.e. $C(x_i)_k$. Another category of clustering is *fuzzy clustering*, where $\{w \in \mathbb{R} | 0 \leq w \leq 1\}^K$ may contain multiple values greater than 0 representing the probability of an item belonging to a cluster rather than a definitive single cluster assignment. However, we will also talk about a subcategory of clustering where we find clusters one at a time. Here, an object may belong to no clusters, in which case the summation to 1 will not hold.

The most common input into a clustering algorithm is as a matrix $n \times m$ matrix D, a collation of x_i objects on the rows and m features on the column — here we refer to each feature column as $y_j \in \mathbb{R}^n$ $j \in \{1, ..., m\}$ and an element in D as D_{ij}. Clustering algorithms can also go through the process of converting this D into a *distance matrix* A' and/or *similarity matrix* A. These are $n \times n$ matrices where each value is a result of comparing two objects i.e. a value describing how distant or similar two objects are from each other. This comparison is done using a distance or similarity metric, some of which are described in section 3.2.

A can also be used to initialize an alternative representation called a *graph*. A graph $G = (V, E)$ is an abstract data structure that contains a set of *nodes* V connected together by a set of *edges* E. The edges are represented as a $|V| \times |V|$ *adjacency matrix* where the contents of this matrix contain the value associated with the edges connecting any two unique nodes. If this matrix is binary, a 1/0 would indicate that there is/no edge between two nodes. If this matrix contains positive continuous values, the edges would be weighted. Typically, each node would represent an object, in which case the adjacency matrix would be analogous to a similarity matrix. One example of finding clusters in a graph is to start with a complete graph and iteratively remove edges that have small similarity values until only K connected components are left. A connected component is a cluster of nodes connected via one or more edges but do not have edges connecting it to other connected components. To complete the notation, we define the $|V| \times |V|$ *degree matrix* R as a matrix whose diagonal consists of the degree of a node, or $R_{ii} = \sum_j A_{ij}$. The degree of a node when A is binary is the number of edges connected to that node.

3.2 Distance & similarity metrics

A *distance metric* or function d describes how distant two objects or value sets (x_i and $x_{i'}$) are from each other in the form of a single numerical value. A distance metric follows a set of rules:

1. Distance values must be positive: $d(x_i, x_{i'}) = 0$ if the two objects are identical, and > 0 otherwise
2. Distance values are symmetric: $d(x_i, x_{i'}) = d(x_{i'}, x_i)$
3. Distance values should respect triangular inequality: $d(x_i, x_{i'}) \leq d(x_i, x_{i''})d(x_{i'}, x_{i''})$

Some of the most common distance metrics for comparison of any two object vectors include the Euclidean and Manhattan distance, also known as the L2 and L1 norms, respectively.

$$d_{L2}(x_i, x_{i'}) = ||x_i - x_{i'}||^2 = \sqrt{\sum_j (D_{ij} - D_{i'j})^2}$$

$$d_{L1}(x_i, x_{i'}) = |x_i - x_{i'}| = \sum_j |D_{ij} - D_{i'j}|$$

If we are comparing two clusters of objects, the distance function d would be referred to as a *link*. Some common link functions include the complete, single, and unweighted average links. These are the maximum, minimum, and mean pairwise distances between all points in one cluster and all points in the second cluster, where the distance metric can be Euclidean, Manhattan, or any other function provided by the user. We can also use the link to calculate the distance between a pair of *centroids* — a centroid being a single point representation of a cluster.

If a method needs to use, instead, a similarity matrix, common ways to convert A' into A is to inverse A' or get the exponent $A = exp(-\frac{A'}{max(A')})$.

Clustering methods can also directly use correlation metrics, a flexible definition of similarity. Pearson correlation is a metric that evaluates how two vectors or value sets compare with each other in terms of their covariance *cov* and standard deviation δ from their mean; so instead of directly comparing the magnitude of two sets of values, it emphasizes the similarity in their trajectory. Using this, it can evaluate how correlated or proportional two value sets are with each other.

$$corr_{Pearson}(x_i, x_{i'}) = \frac{cov(x_i, x_{i'})}{\delta_{x_i} \delta_{x_{i'}}}$$

$$cov(x_i, x_{i'}) = E(x_i x_{i'}) - E(x_i)E(x_{i'})$$

$$var(x_i) = cov(x_i, x_i)$$

$$\delta = \sqrt{var(x_i)}$$

where E as a function is the expected value or the mean. The output of the Pearson correlation metric ranges from -1 (negatively correlated) and 1 (positively correlated) with 0 being not correlated at all. The Spearman correlation, on the other hand, is defined as the Pearson correlation of x_i and $x_{i'}$ after their values have been converted to ranks. Disregarding the original values mean that Spearman finds the monotonic relationship between two sets of values.

Another term used to describe similarity metrics is a *kernel function*. A common kernel function is the Gaussian or radial basis function kernel

$$d_{Gauss}(x_i, x_{i'}) = exp\left(-\frac{||x_i - x_{i'}||^2}{2\delta^2}\right)$$

where δ is a given parameter. The motivation of this separate branch of similarity metric development is to transform data into another often nonlinear space where clusters of points can be separated linearly. Other examples of kernels include the linear, polynomial, and Laplacian kernels.

3.3 Dimensionality reduction

Given an input mtrix, any methods opt to reduce the dimensionality or the number of features in this matrix. Dimentionality reduction is useful as many methods have a hard time dealing with high dimensional data, a phenomenon coined as 'the curse of dimensionality' [35]. Though having lots of features mean there is more data, there could be features introducing noisy uninformative signals and the data space becomes more sparse. As dimensionality m increases, the difference between the maximum and minimum distance d over the minimum distance between a set of random points and a fixed point in space becomes indistinguishable. This is because, distance metrics, such as the Euclidean, and their calculated value becomes more influenced by m than the magnitude of difference between values in objects x_i and $x_{i'}$ [40].

$$\lim_{x \to \infty} E\left(\frac{d_{max} - d_{min}}{d_{min}}\right) \to 0$$

As all objects become a similar distance apart, clustering methods become unable to identify any distinct clusters. We cover a few popular dimensionality reduction methods here. For a more comprehensive review, please refer to surveys [330, 373, 54].

Principal component analysis (PCA) was formulated in 1901 [278] and has since become one of the most common method for linear dimensionality reduction. PCA takes a matrix as input and uses an orthogonal transformation to transform the features into a new set of principal components (PC). Each PC can be treated as a new feature that is composed of a linear combination of the original features. The first PC represents the maximum amount of

variance in the data. The second PC represents the highest variance in the data while being constrained to be orthogonal to the first PC, and so on. By extracting the first few PCs, PCA not only allows users to obtain a matrix with a smaller feature space, but the lower-dimensionality also allows for visualization on a 2-3D plane. PCA scales well with large data sets and preserves the close and long distances between points in high-dimensional space. Most methods apply it to data that has been scaled such that PCA is not biased towards any one feature with larger values. However, PCA assumes that the data set follows a normal distribution, which can be restrictive for data sets that follow other statistical distributions or separates in nonlinear space.

The t-distributed stochastic neighbour embedding (t-SNE) [229], on the other hand, is a nonlinear method to reduce dimensionality. Thought-SNE only preserves local distances, this makes it an ideal tool for visualization. It first calculates a distance matrix (usually via a Gaussian kernel) A' and then it aims to learn a user-specified $m' < m$ dimensional D' such that its distance matrix A'' is similar to A'. This is equivalent to minimizing the Kullback-Leibler divergence (KLD) of the distributions A' and A'' via gradient descent:

$$KL(A'||A'') = \sum_{i \neq i'} A'_{ii'} \log \frac{A'_{ii'}}{A''_{ii'}}$$

Gradient descent works by adjusting parameter values or 'walking' a certain step per iteration towards the gradient or derivative of the KLD to minimize this objective function. The number of steps and step size are user given parameters. The location in solution space or the parameter values at initialization is random. Depending on what implementation of gradient descent one is using and its initialization, t-SNE may output different locally optimal results.

Another popular method to reduce dimensionality and conduct clustering simultaneously is spectral clustering [59, 102]. Spectral clustering is a collection of methods that use the spectrum or eigenvalues of a similarity matrix A of D to perform clustering. The spectrum or the dimensionality reduced features here are the eigenvectors for the K largest eigenvalues of the normalized Laplacian matrix of A, $L = R^{-\frac{1}{2}} A R^{-\frac{1}{2}}$. Like PCs, spectrums have a closed-form solution which makes it simple to implement. As well, these are features created from a similarity matrix and not directly from the original matrix D. Much like in t-SNE, an advantage to this is that one can use nonlinear similarity metrics, such as the Gaussian kernel, to bring the data into nonlinear space. This way, the spectrums would describe the objects based on their relationship with each other in that nonlinear space where the data may be more easily separated.

3.4 Clustering methods

In this section, we go over the major types of clustering methods: hierarchical, segmentation, model-based, density-based, graph-based, and self-supervised deep learning-based clustering. The first three types of clustering are cluster points around a centroid and can also be categorized as *centroid-based clustering*. Hence they tend to find convex clusters.

3.4.1 Hierarchical clustering

Hierarchical Clustering [176] is a hard clustering method and one of the most commonly adopted methods in bioinformatics [387] due to its intuitive algorithm and interpretable results. Given a x_i, its C outputs a length K vector that contains one 1 and $K-1$ 0's where the placement of 1 represents which cluster x_i belongs to. Hierarchical clustering can be split into two types: agglomerative and divisive.

Agglomerative clustering is where it initializes with n clusters, each containing one point. Given a link function (see examples in section 3.2), it calculates the pairwise distance between all the clusters and merges a pair of the closest clusters. If K is not/given, this merging procedure is done over and over until there is only one/K cluster/s left. Agglomerative clustering has a computation complexity of $O(n^3)$ given that it needs to compute the pairwise link between clusters in each iteration. However, depending on the link function, this complexity can be decreased. For example, [323, 81] uses the heap data structure to reduce run time down to $O(n^2 \log n)$.

Divisive clustering, on the other hand, does the opposite. It initializes with one cluster containing all points and then splits this cluster into two subclusters based on the weakest or most distant link. If K is not/given, then it repeats this split on each of the resulting clusters until it ends up with n/K total clusters. As there are $O(2^n)$ ways to split a cluster of n objects, many methods resort to heuristics to reduce the complexity. For example, when DIANA [182] splits a cluster Q into two smaller clusters A and B, it starts with $A = Q$ and an empty B. It initializes B by putting in the object A that has the maximum total distance from all other objects in A. DIANA proceeds to move object i from A into B one at a time such that i maximizes

$$max_i d'(i, A \backslash i) - d'(i, B)$$

where d' is a user-given link function. When the maximum value of this criterion becomes < 0 or when there is only one object left in A, it stops moving objects and the split is complete. Depending on the heuristic used, divisive clustering can be more computationally complex, but it has been shown to produce better clusters as it considers global as opposed to local distributions as in agglomerative clustering [181].

These methods are called hierarchical because the agglomerating and division process can be illustrated as a binary tree. A binary tree is defined as a type of graph where all nodes (except for the root node) has one parent node and two child nodes (except the leaf nodes). The root and leaf nodes are the starting and resulting clusters for divisive clustering, and the reverse for agglomerative clustering. Each node represents a cluster and its two child nodes represent the clusters it split into or is merged from. The binary tree visualization allows the user to interpret how the clusters came to be, hence making it a popular option for bioinformatics [204]. However, a limitation to this method is that once clusters are split or merged, these actions cannot be fixed at a later time. As well, the shape of the clusters is dependent on d'. For example, if the link is the mean pairwise Euclidean distance between every pair of points in two clusters, the clusters will end up being convex in Euclidean space.

3.4.2 Segmentation-based clustering

Also, a hard clustering method, one of the most popular Kmeans [263]. Kmeans is a segmentation clustering method where object clusters are 'segmented' apart from each other. It was formally described in 1955 and is still being used regularly today [167]. Its objective function is

$$min \sum_{k} \sum_{C(x_i)_k=1} d'(x_i, \mu_k)$$

where d' is a link function and μ_k is the centroid of cluster k. Originally, μ_k was defined as an mD mean of feature values of all points in cluster k, and d' is the squared error of the Euclidean distance between x_i and mu_k. Other variations of Kmeans give different definitions for the centroid and distance function. For example, Kmedoid uses the median as the centroid, instead of the mean, to account for the fact that means are easily skewed by outlier points. Another example is kernel Kmeans [83] who makes use of nonlinear link distance functions to find clusters formed in nonlinear space. Regardless, the objective functions all serve to minimize within-cluster variation, thus suitable for finding convex clusters in their respective space.

The optimization of this objective function, however, even with $K = 2$, is an NP-hard problem. The most well-known algorithm to optimize this objective function begins by 1) randomly selecting K points as the cluster centres, and then 2) assigning each point to the cluster centre it is closest to. 3) Once these clusters are formed, it finds the centroid of each newly formed cluster. 2) and 3) are repeated iteratively until convergence i.e. no objects are assigned to a different cluster compared to the previous iteration of the algorithm. This procedure allows Kmeans to be computed very efficiently in $O(nKmr)$ where r is the number of iterations, which in the worst case is $2^{\Omega(\sqrt{n})}$. Outcome-wise, the resulting clusters are heavily influenced by the initially chosen cluster centroids. We will see how some methods try to overcome this by choosing better initial centroids or by refining the cluster results post-clustering.

3.4.3 Model-based fuzzy clustering

A popular method for fuzzy clustering is *mixed model clustering*. This method assumes that data is generated using a mixture of K individual statistical distributions each containing point belonging to a cluster. In the Gaussian finite mixture model (GMM)'s case, it assumes that the points in each cluster can be fit or modelled as a Gaussian distribution $\mathcal{N}(x_i|\mu_k, \delta_k) = \frac{exp\left(\frac{(x_i-\mu_k)^2}{2\delta}\right)}{\sqrt{2\pi\delta}}$ represented by parameters: an mD mean of the features in cluster k as the centroid μ_k and the standard deviation of the points in cluster k, δ_k. Depending on where each point is located relative to these K clusters, the points are assigned to each cluster with a weighted $0 \leq w_k \leq 1$ probability of:

$$P(x_i) = \sum_{k=1}^{K} w_k \mathcal{N}(x_i|\mu_k, \delta_k)$$

To infer the parameters for each cluster, GMM can use expectation maximization (EM) [294]. It first initializes with K random sets of distribution parameters in D space. Using the Bayes rule, it determines how likely these distributions generated D — this is the expectation we want to maximize:

$$\sum_{k} P(k|x_i) = \frac{P(x_i|k)P(k)}{\sum_{k'=1}^{K} P(x_i|k')P(k')}$$

where $P(x_i|k) = \mathcal{N}(x_i|\mu_k, \delta_k)$ and $P(k) = \mathcal{N}(\mu_k, \delta_k)$. In other words, this step evaluates for each point, its $C(x_i)_k$ based on how well it fits into each cluster k. This is analogous to the cluster reassignment step in Kmeans. Finally, these probabilities are used to adjust the cluster distribution parameters such that they best represent the current cluster assignment. For GMMs, this would involve recalculating the μ_ks and δ_ks based on the updated cluster assignment. Its run time is analogous to that of Kmeans as these two steps are repeated until convergence.

To overcome the issue of needing to know K beforehand, many methods use an infinite mixture model instead. For simplicity, we will assume that our distributions are Gaussians, though this can be changed based on user need. One popular tool to facilitate the dynamic changes in K is the Dirichlet process (DP). $DP(T, \alpha)$ is a distribution of distributions commonly used to generate parameters. The T is a base distribution and α is a user given parameter specifying how different the distributions T' generated by the DP should be from the original T. α is set with the property such that as $\alpha \to \infty$, $T' = T$. Using a stick-breaking analogy, the DP process generates distributions T' via

$$T' = \sum_{k=1}^{\infty} \beta_\ell \delta_{z_k}(z)$$

where β_ℓ is a weight drawn from a beta distribution $beta(1, \alpha)$ and $\delta_{z_k}(z)$ is 1 if $z = z_k$. Note that the expectation of a value being drawn from the distributions T and T' is always the same. Since the DP has an unbounded k, we can theoretically generate an infinite amount of distributions or clusters to evaluate. This way, we can thoroughly inspect what K is optimal. In the context of producing clusters, another way to look at the posterior of a DP is via the Chinese restaurant analogy. Assuming each cluster is a Gaussian, it would have a mean and covariance. In the analogy, each cluster k is thought of to be a table at a restaurant with P_k number of people sitting at it, each person being a data point. Each table also contains the parameters required for a specified statistical distribution. In the Gaussian distribution, this would be the mean and covariance. When a new person enters the restaurant, he/she has a probability of γ to sit at a new table (produce a new cluster) or a probability of P_k to join table k. A common way to conduct this seating assignment is, like in GMM, via sampling — Gibbs sampling, a Markov chain Monte Carlo (MCMC) algorithm and specialization of the Metropolis-Hastings algorithm. Given a multivariate distribution, it initializes by randomly guessing several tables, their parameters, and the people or points who are assigned to them. The assignment of each point w'_i (not to be confused with the final w_k vector) is determined by the conditional probability

$$P(w'_i = k|w_{-i}, \gamma, x_i, \theta_k, D) = P(w'_i = k|w'_{-i}, \gamma)P(x_i|\theta_k, D)$$

where θ_k are parameters for distribution k. w'_{-i} here is simply the assignment of all points other than i. The sampling process continues to refine these parameters until convergence. In theory, unlike Kmeans, this process does not require a good initialization, but to speed things up, a good initialization or even a hypothesized prior as to what parameters the model could have is desirable.

Although model-based fuzzy clustering has sound statistical grounds to incorporate and infer lots of parameters, it is slower than hard clustering methods such as Kmeans or hierarchical clustering. As well, it requires robust assumptions on what type of distribution best represent the data.

3.4.4 Density-based clustering

In contrast with previously mentioned centroid-based clustering methods, density-based clustering methods emphasize finding connected groups of points regardless of their shape. An example method in this category is DBSCAN (density-based spatial clustering of applications with noise) [95]. DBSCAN starts by calculating which point contains a minimum amount of points within its neighbourhood. A *neighbourhood* refers to a certain number of points within a certain distance radius of the said point in D space. Both the number of points threshold and radius parameter are given by the user. Points that contain more than the number of points within the given radius are assumed to belong to dense areas of

the data space. Since it also assumes cluster borders should be sparse while cluster innards should be dense, these points are also assumed to be on the inner non-border areas of clusters. Points that do not meet this criterion are either outlying or lie on cluster borders — depending on whether they are or are not a part of the neighbourhood of a point that does meet the criteria. This assumption intrinsically defines the structure of a cluster allowing DBSCAN to function without needing the user to input K. However, DBSCAN does not respond well to situations where different clusters have different densities — because it has a global threshold for density. As well, this category of methods tends to be computationally expensive. For example, DBSCAN requires one to compare each pair of points to confirm all neighbourhoods hence a $O(n^2)$ runtime. To speed things up, DENCLUE (density clustering) [154] uses a grid-based approach by binning or cutting up the space of D into mD 'cubes'. It first gets rid of outlying cubes that contain a low number of points and then conducts DBSCAN with the remaining cubes as the pseudo-points (one or more points that collectively act as a single point). Prototype DBSCAN [92] uses the same strategy but takes as input, the result of any clustering algorithm. This way, it already has an initial parameter K and many prototype clusters to act as pseudo-points. The final clusters obtained are the meta-clusters made up of groups of initial prototype clusters. Although density-based methods are good for finding arbitrarily shaped clusters, the original algorithm and the updated versions do not deal well with high-dimensionality. Like the other distance-dependent algorithms, points become sparse as dimensionality increases, making the clusters harder to separate based on density.

3.4.5 Graph-based clustering

Graph-based clustering is a broad categorization of clustering methods that use the graph representation to depict data. Most commonly, these algorithms start with a graph G by representing a point or a group of points as a node. Then they connect these nodes using edges whose weights are based on some similarity or distance metric.

Creating a minimum spanning tree (MST) is one way to remove edges between nodes and segment the data. An example of a method that creates and uses the MST is MOCK (multi-object optimization clustering) [141]. It first calculates a distance matrix off of which it creates a fully connected graph where the nodes are the points and the edges between nodes are weighted by the distances. On this graph, Prim's algorithm [130] finds the MST in linear time and it drops edges that represent the longest distance between points. Edges are dropped until K clusters of interconnected points remain or no more edges represent distances farther than a given threshold.

Many variations of edge trimming exist, but more often than not, instead of making an MST, one can use other more simple heuristics. For example, whether or not the edge weights are under or over a threshold. One can also specify to leave on a maximum number of edges connected to each point. The latter creates a K-nearest neighbour (KNN) graph

where K represents the degree or the maximum number of edges that connect each point to its neighbourhood of the K closest nodes.

Taking the KNN graph as input, the Louvain algorithm [45] incorporates this data structure with agglomerative hierarchical clustering. Louvain uses a greedy iterative two step procedure to cluster the cell objects. Much like in hierarchical clustering, each node starts out as its own cluster. In the first phase, a metric is calculated for each cluster to evaluate how much modularity Q the cluster would gain by merging with any one of its neighbouring clusters. Modularity change by adding node v to cluster u here is defined as

$$Q = \left[\frac{\sum_{in} + 2e_{v,in}}{2e'} - \left(\frac{\sum_{tot} + e_v}{2e'}\right)^2\right] - \left[\frac{\sum_{in}}{2e'} - \left(\frac{\sum_{tot}}{2e'}\right)^2 - \left(\frac{e_v}{2e'}\right)^2\right]$$

where $\sum_{in}$, $\sum_{tot}$, e_v, $e_{v,in}$, and e' are the sum of edge weights inside u, incident to all points in u, incident to point v, between point v & points in u, and between all nodes in the graph respectively. Looking at the merge that would cause a maximum modularity gain, if this gain is a positive value, then the merge is conducted. This is repeated until no more merges occur. In other words, the second step repeats the first step on a new graph where the merged clusters are treated as starting nodes and the edge weight is the total sum of distance between the old nodes in the merged clusters. Louvain is a popular graph-based clustering method as it not only refines its results, it also runs efficiency in $O(n \log n)$ time.

While spectral clustering is mainly associated with its unique method to reduce dimensionality, the spectrums found can also be used to find connected component clusters within the original similarity matrix or adjacency matrix A of graph G. The Fiedler eigenvalue [101] of a graph corresponding to A is defined as the second smallest eigenvalue of L. This value is 0 if G is a connected graph i.e. all nodes are connected to all other nodes through an edge or a collection of connected edges, a path. The larger this value though, the more connected the graph is (more edges present), so it can also be used as a graph evaluation metric. However, in clustering, it can also partition a graph. The eigenvector corresponding to the Fiedler eigenvalue is the Fiedler eigenvector. As this vector has a value associated with each node, it can indicate how well connected each node is with the other nodes in the graph. A smaller/larger or negative/positive number means that the node is not/well connected or does/not have many edges connecting it to other nodes. One strategy is to segment the nodes into clusters via the sign of their associated values in the Fiedler eigenvector.

3.5 Cluster evaluation

While model-based clustering can use expectation to pick out the best set of clusters, other methods, such as Kmeans and hierarchical clustering, also require a way to evaluate their clustering results. As there is no ground truth in unsupervised clustering, evaluation of results is done using internal evaluation. For example, given distance metrics, the Dunn

index is formulated as the minimum inter-cluster distance over the maximum intra-cluster distance. minimizing this means that one would have tight dense convex clusters that are far from other clusters. Another metric commonly used to evaluate the intra-cluster homogeneity and inter-cluster heterogeneity is the silhouette index. The silhouette index is a value assigned to each point i

$$silhouette(x_i) = \begin{cases} 1 - \frac{a(x_i)}{b(x_i)} & \text{if } a(x_i) < b(x_i) \\ 0 & \text{if } a(x_i) = b(x_i) \\ \frac{b(x_i)}{a(x_i)} & \text{if } a(x_i) > b(x_i) \end{cases}$$

$$a(x_i) = \frac{\sum_{i \neq i', i' \in C_{i'}} d(x_i, x_{i'})}{|C_{i'}| - 1}$$

$$b(x_i) = min_{i \neq i'} \frac{\sum_{i' \in C_{i'}} d(x_i, x_{i'})}{|C_{i'}|}$$

where $C_{i'}$ is the cluster object i' is in and d is a user given distance metric. Given that each point has a numeric value indicating its quality, the silhouette index makes for good visualizations for interpretation.

Going back to model-based clustering, common cluster evaluation metrics other than its objective include the Bayesian information criteria [314] $BIC = |\theta| \log(n) - 2 \log(\ell)$ and the Akaike information criteria $AIC = 2|\theta| - 2ln(\ell)$. ℓ is the likelihood of the model evaluated and θ is the set of all parameters in the model. The first term in both metrics attempts to regularize the number of parameters or degree of freedom the model allows for to prevent overfitting i.e. we do not want a better score simply because there are more parameters. The second term is then there to ensure that the clustering model appropriately fits the data in the form of a likelihood function with respect to the data and parameters tuned.

3.6 Clustering in cytometry

FCM samples usually contain relatively few (around 10) features or markers and a large amount (thousands) of objects or cells. However, the clustering methods used also need to account for when there are up to 50 markers following the capacity of FCM [319]. The goal of clustering cells in FCM samples is to identify previously known and unknown cell populations based on the FCM sample's data structure.

As many methods consist of the steps, preprocessing, clustering, and postprocessing, we organize this section so that each method is discussed in the context of these steps. An overarching theme we focus on is the challenges each procedure aims to overcome.

(a) A common challenge is the need to find biologically relevant non/convex clusters. While traditional clustering methods, such as Kmeans and GMM, focus on finding

convex clusters, **cell populations could be of 1a) arbitrary shape, and even 1b) multimodal**. However, traditional methods have survived due to being computationally efficient and statistically sound. Therefore, many authors tend toward these methods and try to circumvent their disadvantages by adding engineering tweaks to their methodology.

(b) Similarly, another challenge is the need to find **rare cell populations**. As these populations can be small and overlapped by larger populations, traditional methods would merge them into those large populations. Hence, there is a need to effectively identify when rare cell populations exist and maintain them as result.

(c) While clustering methods need less user input than other types of methods, they still require parameters such as the number of clusters K desired. Unless the user has a particular set of cells he/she is looking for, K is difficult to define. Therefore, many methods resort to automatically finding such parameters to reduce decision fatigue on the user.

(d) Outside of finding particular populations, methods also need to adapt to circumstances when D consists of a **high-dimensionality** m. Many classical methods depend on the distance metric employed to understand the relationship between points. However, in high dimensions, these distances become sparse and uninformative. As well, higher dimensions can result in unmanageable computational complexity.

(e) In addition, since clustering methods do not incorporate user knowledge, clusters in different samples are unmatched i.e. we do not know which clusters represent which cell populations that may exist across multiple samples. To this end, many clustering methods add a step to **match clusters across samples** or directly build this into the clustering procedure.

(f) Finally, it is desirable to make the results more **interpretable**. This may be solved via extra visualizations, or even to make the clustering process more similar to manual cell population identification.

3.6.1 Preprocessing

We focus on two preprocessing procedures that many methods utilize to contort their data in preparation for the actual clustering step. These two types of preprocessing include changing how data is represented and automatically defining parameters needed by the clustering algorithm.

Data representation

While most methods use D directly, many choose to transform D to facilitate its clustering process, usually to mitigate challenges (a) and (b).

Challenge (b) is especially problematic when it comes to decreasing the computational complexity when clustering high-dimensional data. Therefore another way to prepare D is to first reduce its dimensionality. ACCENSE [320], and ClusterX [63] apply a 2D t-SNE (t-Distributed Stochastic Neighbor Embedding) [229] to get a projection of the original data. Although t-SNE is stochastic and not ideal for conducting dimensionality reduction representative of the original data [269], it works and is practical on specific data sets [383]. One of the reasons for this is because t-SNE transforms D nonlinearly such that downstream algorithms can find clusters that separate in nonlinear space, whereas if the original D is used, these algorithms would only find clusters that separate in linear space. While t-SNE accomplishes this while reducing dimensionality, methods can also simply resort to data transformation. For example, flowClust [223] uses Box-Cox [309] to transform the data such that it has an approximately normal distribution, before initializing and modelling it using GMM.

Another alternative representation of D is a graph. Each cluster or point can be summarized as a node in a graph connected using edges whose weights are defined by a distance/similarity metric. In SamSPECTRAL's case [404], it uses the heat kernel similarity metric between the FI values of each point [193] — scaling D into negative exponential space. Directly using the original graph is convenient for algorithms that require graph inputs. However, having a fully connected graph (where all nodes are connected to all other nodes) can result in high computational complexity. Therefore, X-shift [310] also represents D in graph form with edge weights defined by the Mahalanobis distance; but before clustering, it opts to shed some of its edges by converting this fully connected graph into a KNN graph, where K is user-defined. On top of shedding edges, one can also refine the edge weights using uniquely graph-based heuristics. For example, after obtaining the KNN graph where edge weights are based on the Euclidean distance, PhenoGraph [209] redefines its edge weights as a graph heuristic that involves the Jaccard metric which measures how many shared neighbours two connected points have. Since this method is focused on finding rare cell populations for the Challenge (b), the refinement allows PhenoGraph to embed the notion of density into the edge weights. This way, if a large less dense population is masked on top of a smaller denser population, they can be differentiated.

Defining the number of clusters

After obtaining the original or alternative representation of D, another concern one might have is how to define any required clustering parameter(s) for Challenge (c). A common parameter is K or the number of clusters one would expect to get as a result. While earlier methods such as [263] maintain this requirement, later methods attempt to mitigate this need. Aside from pure density-based and infinite mixture model-based methods, many other clustering methods treat the process of finding K as a preprocessing procedure. While we include examples of cluster evaluation in section 3.5

flowMeans, an extension of flowMerge [8], finds K by detecting the amount of modes or peaks in every eigenvector of D using [91]. If the same clusters are projected onto multiple eigenvectors (i.e. overlaps), these clusters are merged via agglomerative hierarchical clustering. The link between clusters used for merging is the modified symmetric version of the Mahalanobis distance between points of two clusters

$$d(X_1, X_2) = min\left(\sqrt{(X_1 - X_2)\delta_{X_1}^{-1}(X_1 - X_2)^\top}, \sqrt{(X_1 - X_2)\delta_{X_2}^{-1}(X_1 - X_2)^\top}\right)$$

where X_1 is the collection of x_i objects in cluster 1, X_1 is the mean of X_1, and δ_{X_1} is the covariance matrix of X_1. Agglomerative hierarchical clustering continues to merge clusters until the distances between all clusters abruptly changes from very far to very near. Similarly, SamSPECTRAL [404] directly estimates K as the number of maximum value (=1) eigenvalues of A before the eigenvalues abruptly start decreasing.

flowPeaks [121] uses an adhoc way to find $K = median(K_j)$ where K_j is the number of clusters found for each dimension j of D using the Freedman Diaconis formula [112]

$$K = median_{j=1,\ldots,m}\left(\frac{(max(x_j) - min(x_j))}{\{2 \cdot IQR(x_j) \cdot n^{-\frac{1}{3}}\}}\right)$$

where IQR is the interquartile range or the difference between the 75th and 25th percentile of the data.

On the other hand, many mixture model-based methods test a range of K to find the one that produces a result with the maximum likelihood. FLAME is one of the earlier methods to implement this strategy for FCM. It first clusters D using the skewed t-distribution mixture model on a range of Ks via EM [218]. It then finds the best clustering for each sample that minimizes the scale-free weighted ratio (SWR). This is a weighted ratio of the average intra-cluster and inter-cluster scale-free Mahalanobis distance normalized for variance in shape, dispersion, and orientation to minimize the influence of outliers. Other metrics that evaluate clusterings include the BIC (used in flowClust [223] and flowGM [66]) and the integrated classification likelihood (ICL) score [42] (used in immunoClust [332]). While the former is a common metric, it often overestimates the number of clusters. Therefore, immunoClust uses a normalized ICL such that it is only slightly biased towards results with fewer clusters.

Finally, it is also possible to find K first by using a clustering method that does not require K and then using the result of that clustering as a springboard for another clustering algorithm. FLOCK (FLOw Clustering without K) [285] initializes its clusters by using DENCLUE [154]. It uses the grid-based approach to conduct density-based clustering and then merges the grids that contain more points than a user-given threshold. Finally, K is defined as the resulting number of clusters found.

3.6.2 Clustering

The first published example of applying clustering on FCM data uses Kmeans. However, well-established traditional methods have a con in that it does not consider the unique challenges in FCM. For example, Kmeans and fuzzy model-based clustering methods meet a wall with Challenge (a) and (b). They usually find large convex clusters (or whatever distribution is used in the mixture model) depending on a user given K, unideal for Challenge (c). Nevertheless, it is desirable to exploit the fact that these methods are efficient and statistically sound. On the other hand, density-based clustering can find arbitrarily shaped clusters but they are inefficient when used on high-dimensional data as in Challenge (d). Extra postprocessing steps also need to be taken to satisfy challenges (e) and (f).

Finding convex clusters

In this section, we introduce methods that use a centroid-based clustering algorithm to initialize or find clusters. Although we do hope to mitigate Challenge (a), many methods resort to using classical algorithms to quickly initialize clusters before evaluating whether or not these clusters are valid or would need further refinement. In addition, most cell populations do conform well to convex clusters, therefore there is still a good market for these methods.

flowMeans [8] and flowPeaks [121], for example, uses Kmeans++ [22]. Given K, Kmeans++ is the same as Kmeans except it uses a different approach when initializing the cluster centroids and choosing new centroids with each iteration. It attempts to mitigate situations where a random initialization may affect the clustering outcome. When Kmeans++ is finding its initial cluster centroids, it repeatedly selects K random points. The final K centroids are chosen according to how close they are to the previously chosen random centroids, close being good. This ensures that the final starting centroids are in a densely populated area likely to be a cluster centre. After assigning each point to a cluster centroid, it defines a cluster by overlaying it with a Gaussian distribution who is represented by its parameters: cluster assignment w_k, mean μ_k, and smoothed covariance matrix δ_k (whose density distribution is binned based on a user given parameter). Assuming that a cluster is a dense patch of points, it proceeds to use the hill-climbing search algorithm [194] to approximate a point of largest density in each cluster to act as the new cluster centroids. The maximum step size used in the algorithm is based on the cluster size as specified by $\theta_k = min_{i=1,\ldots,m}\sqrt{\delta_k}$. This way, the algorithm will not take too long, or miss any peaks if the step size is too small or large respectively.

Likewise, FLOCK [285] also uses a centroid-based method, but it initializes its clusters by using DENCLUE [154] as mentioned in section 3.6.1. Taking the initial clustering results from DENCLUE, it initializes the Kmeans algorithm by assigning the mean for each DENCLUE cluster as cluster centroids. Points are then assigned to a cluster centroid to

whom it is closest in Euclidean space. Thereon after, it proceeds with Kmeans to get the final clusters.

A third example of a method that directly uses a traditional convex clustering algorithm is X-shift [310]. X-Shift uses its KNN graph, like in Kmeans, to cluster around centroids. These are points of maximum local density or 'hubness' i.e. it is amongst the KNN of a large amount of its KNN. Each non-maxima point is then connected to one of these centroids via a density-ascending path in the graph. Connected points are then shown as clusters.

Finding arbitrarily shaped clusters

While some methods adapt density-based methods directly (e.g. auto/FlowGrid [96] and floptics [338]) to find multi/unimodal clusters, other methods, such as Misty Mountain [342] and ACCENSE [320], assume that clusters should be unimodal but not necessarily convex clusters. These methods first represent the data using a kernel density estimate. One can imagine the kernel density estimate or histograms as an additional dimension. To find the density peaks, Misty Mountain starts from the maximum value of this density dimension and slowly descends to 0. On the way, density peaks will slowly emerge. When any two peaks start to merge, these two peaks would be recorded as separate clusters. If there are peaks that remain independent, then, in the end, these peaks would also be recorded as their clusters. In the context of previous methods, Misty Mountain assumes a cluster can be represented by a density peak or centroid and splits the clusters at the density valleys. Conversely, ACCENSE finds the peaks via a traditional metric from [168]. Apart from the peak finding methodology, the difference between Misty Mountain and ACCENSE is two-fold. First, Misty Mountain does not require the user to specify a bin width to create the kernel density estimate. It does so by using [191], a probabilistic approach, to determine the best number of bins for discretization in each dimension with which it builds the multidimensional density histogram. Another difference is that ACCENSE uses the kernel density estimate of a 2D representation of D created with t-SNE. Lowering the dimensionality makes the clustering more efficient in compliance to Challenge (d).

Also engineered to find arbitrarily shaped clusters for Challenge (a), FLAME [284] is an example of a method that adapts the classical GMM algorithm. Given the K found in section 3.6.1, it assumes that the cell populations represent a skewed multivariate probability distribution in Box-Cox transformed space i.e. it uses the skewed t-distribution instead of the convex Gaussian distribution in GMM on Box-Cox [309] transformed D.

Finding rare cell populations & matching cell populations across samples

A way to isolate rare cell population clusters for Challenge (b) is by first eliminating outliers and noise to make rarer cell populations easier to detect [31, 369]. flowGM [66] does this using expert knowledge by applying two rounds of default filters. These filter out uninteresting objects such as debris and dead cells. The filter here refers to a series of points in

D space that encircles a set of target points. In between filtering, flowGM conducts multiple iterations of GMM clustering on the refined D based on a given K and chooses the clustering with the highest likelihood.

A second strategy is to find cell populations simultaneously across all samples, hence also mitigating Challenge (e). This way, if a cell population is rare in one sample but emphasized in another, it would not be missed in the former sample. The H part of HDPGMM (hierarchical Dirichlet process GMM) [76] represents that this method does just that. It shares information and finds the same clusters (i.e. same Gaussian distribution parameter sets) across all samples. More specifically, it generates GMM clusters that are different across samples only in its w, or the number of objects in each cluster. To achieve this, it adds a layer of parameterization for w_k

$$w_k = w'_k \prod_{\ell=1}^{k-1}(1 - w'_\ell) \quad for \quad k = 2, ..., K-1$$

where $w'_k \sim \beta(1, \alpha)$, and α is given by the user. Hence, the priors, or the given variables, here are the α's, μ's, and δ's. Once the model is defined, HDPGMM uses [351] with the Metropolis within the Gibbs approach to calculate the posterior and account for the non-conjugate conditional distributions of w' and α [123].

Putting all of these together, BayesFlow [177] builds on HDPGMM to simultaneously cluster across samples and eliminate outliers at the same time — all this done via a few more parameters. First, to account for outliers, it adds a component

$$P(x_{ih}) = \sum_{k=1}^{K} w_{kh}\mathcal{N}(x_{ih}|\mu_{kh}, \delta_{kh}) + w_{0h}\mathcal{N}(x_{ih}|\mu_{0h}, \delta_{0h})$$

for each sample h, where subscript 0 represents outlier points and their distribution parameters which are the same across samples. The clusters, however, are different across samples but are connected via a latent layer of variables. This is based on the assumption that for each cluster k, its parameters μ_{kh} and δ_{kh} are generated by the same normal and inverse Wishart distribution respectively. The parameters for these distributions are hyper-parameters that can be tuned by the user. The sampling of latent variables here is done via MCMC.

Like BayesFlow, ASPIRE [89] accounts for outliers, over-clusters cells with HDPGMM, and then merges those clusters to mitigate challenges (a), (b), and (e). However, it builds all of these into a single statistical model using a non-parametric approach by adding two more components. The first additional component is random effects, to account for variation between samples. The second additional component is a layer of GMM which assumes that a cell population is a meta-cluster composed of a potentially infinite mixture of clusters from each sample. This accounts for the fact that a meta-cluster may be skewed or multi-modal.

More precisely, ASPIRE first models each point using a GMM to find the same clusters across samples. Up until this point, ASPIRE is equivalent to HDPGMM [76]. To account for outliers, the parameters of this GMM are generated by a discrete distribution modelled by a DP like in HDPGMM-RE [185]. In HDPGMM-RE, this DP would be directly parameterized by α and a base Gaussian distribution. However, ASPIRE replaces this base distribution with another discrete distribution that represents a meta-cluster. The parameter of this distribution, in turn, is generated by an overarching base distribution modelled by a second DP parameterized by γ and a Gaussian distribution.

Another common strategy is to incorporate the framework of agglomerative or divisive hierarchical clustering. Both of these methods can find rare clusters (resolving Challenge (b)), multimodal clusters, and depending on the link, arbitrarily shaped clusters (resolving, in part, Challenge (a)). Phenograph [209] employs the Louvain method to incorporate the graph representation into a hierarchical framework, while SWIFT (scalable weighted iterative flow clustering technique) [265] and immunoClust [332] incorporate hierarchical clustering with model-based clustering. Expanding on immunoClust, it performs over-clustering via divisive hierarchical clustering; but at every step, one cluster can be split into multiple clusters via GMM and the EM algorithm [110, 109]. In each split, immunoClust uses GMM to separately cluster for $K = 1, ..., K'$ and chooses the one clustering with the best ICL score [42] as described in section 3.6.1. The algorithm stops when it deduces that all the clusters should no longer be split. Another example of a method that uses divisive hierarchical clustering is cytometree [72]. cytometree incorporates ideas from manual threshold gating by assuming each cell is either positive/negative or have/not each marker. It creates the hierarchy by starting with a root node representing all cells. For each marker, it models the cells on this node as a mixture of two Gaussian distributions i.e. a GMM with $K = 2$:

$$P_j^{(v)}(x_i) = w\mathcal{N}(x_i|\mu_1, \delta_1) + (1 - w)\mathcal{N}(x_i|\mu_2, \delta_2)$$

where v is the node or the current cell population. Sticking with a criterion that does not depend on n_v (the number of cells in population v), cytometree uses the normalized difference between the Akaike criterion (AIC) [73] for the two clusters:

$$s_j^{(v)} = \frac{AIC_1 - AIC_2}{n^{(v)}}$$

This gauges whether or not it is appropriate to accept the two cluster GMM result. The marker with maximum $s_j^{(v)}$ is chosen, and if it is above a user-given threshold, the node v is split according to the two clusters at the said marker. Otherwise, if the threshold is not met or there are too few cells in v (e.g. less than 50 cells may cause $s_j^{(v)}$ to be invalid), the splitting is terminated at that particular cell population node i.e. a leaf node. Once the binary tree of cell populations is built, the leaf nodes are treated as the final cell populations. The same procedure can be done to split each population, for a marker, into three subpopulations.

3.6.3 Postprocessing

After clustering the cells together, methods can opt to incorporate several postprocessing steps to refine the clusters and interpret them. Generally, the three types of postprocessing done are evaluation and refinement of clusters, cell population matching and labelling across samples, and visualization or communication of results.

Finding multimodal and arbitrarily shaped clusters

Rather than embedding it into the clustering process, the refinement and merging of clusters can also be done as a postprocessing step after initial clustering is completed. Merging convex clusters allow methods to come up with multimodal arbitrarily shaped clusters.

X-shift [310], for example, overcomes the original clustering algorithms' tendency to find convex clusters by merging two clusters whose inter-cluster distance is below a user-given threshold. While X-shift defines this distance as the inter-centroid Mahalanobis distance, the Kmeans++ method, used in flowMeans [8], uses the Euclidean distance. In contrast, SamSPECTRAL [404] assumes that clusters should have one highest density point that phases off smoothly toward the edge of a cluster. Hence, after conducting spectral clustering on a KNN graph of D, SamSPECTRAL proceeds to merge two clusters if the ratio between the maximum edge weight within the clusters and the sum of all pairwise edge weights between the points in two clusters is greater than a user-specified threshold. This method finds semi-multimodal clusters with the potential to be arbitrarily shaped. However, no two clusters with equally large density peak centroids would be merged.

In addition to using cluster centroids, BayesFlow uses the Bhattacharyya distance [116] between Gaussian distribution clusters to also incorporate covariance. This allows it to recognize that a denser cluster within a large sparse cluster should be kept separate. Two thresholds are used to determine whether or not to merge clusters. If the distance is over the larger threshold or lower than the smaller threshold, the two clusters are not/merged. If two clusters are separated by a distance that falls between the two thresholds, then Hartigan's dip test for unimodality [145] must also be satisfied. In other words, if two clusters are not sufficiently far apart, then they would need to have sufficiently different centroids to be determined as separate clusters.

Cell population matching and labelling

While methods like HDPGMM, Bayesflow, and ASPIRE match clusters across samples while clustering, many methods choose to add this as a separate step post-clustering.

After clustering all the samples with its mixture model, FLAME [284] proceeds to identify modes in each population based on the μ's in the mixed model parameters found. These means or cluster centroids are pooled from across all the samples and then clustered using Kmeans. The centroid used in this Kmeans is the median. The optimal number of meta-

clusters is measured by the resulting average silhouette index [305]. Once the meta-cluster centroids are identified, the cluster centroids found for each sample is individually matched to the closest meta-cluster centroid as defined by the scale-free Mahalanobis distance via imperfect bipartite matching i.e. one or more original cluster can be assigned to the same meta-clusters, resulting in them merging.

MetaCyto [160], on the other hand, is a tool made exclusively for identifying common labels across studies and samples. It first clusters all the samples using a standard clustering method. Like in cytometree [72], each marker is subsequently split into two regions by using a threshold. It tests multiple thresholds and uses the one that splits up the cells most appropriately as defined by the silhouette index [305]. As in manual gating, each cluster is then labelled based on which of the two regions it is in for each marker, Hence, the cell population labels are consistent across samples.

Visualization

After obtaining the cell population clusters, one can mitigate Challenge (f) by obtaining interpretable visualizations that can communicate the results.

As it is difficult for users to comprehend cell population identification in high-dimensional space, methods resort to dimensionality reduction techniques to display the cell/populations on a 2D surface [229]. Traditional dimensionality reduction methods include self-organizing maps (as used in flowSOM [367]), PCA [389] and t-SNE [229] (as a part of viSNE [14], one-SENSE [68], and PhenoGraph [209]). In addition, CLARA [336] also tries to proportionally represent edge weights or distance in mD space on a 2D surface. It shows each cell as a point or node whose distance to other nodes is dictated by a force-directed weighted graph. The original edge weights on this graph are based on the cosine distances between the median FI of cells.

On a cell population level, SPADE [286] uses agglomerative hierarchical clustering on a down-sampled set of cells and then organizes and colours them as grouped on an MST. Emphasizing the notion of cell populations, cytometree [72] directly shows the users its divisive hierarchical binary tree. This not only shows users its clustering process, but it also does so without loss in performance and accuracy [383].

Within an application context, tools such as RchyOptimyx [268], gEM/GANN [359] and FloReMi [368] use already labelled clusters and samples. This is so that they can further mark and display cell populations that discriminate between different classes of samples to help with understanding the validity of found clusters [5]. These visualizations help transfer knowledge obtained from clustering to a downstream research application. A review of the available visualization software is covered in [239].

3.6.4 Alternative solutions

Clustering methods suffer from the curse of dimensionality. [269] corroborates this statement empirically by testing some common methods such as Phenograph, X-shift, and flowMeans, on a simulated 20 marker data set. The same is done for visualization tools such as t-SNE. Although strategies, such as over-clustering and refinement, can help to mitigate some issues [307], if one has prior knowledge of what cell populations one wants to find, automated gating and supervised methodologies can be good alternatives. In addition to the alternatives mentioned in Section 2.2.2, Tools that make their own gating strategy include FAUST [131], and cytometree [72]. While gates can be of any shape, they are typically marker thresholds that separate cells with high or low FI (i.e. have/not a marker). While FAUST also defines gates, it uses a rule-based gating strategy as opposed to a human-defined one. FAUST defines up to four gates by identifying the valleys along with their taunt string density estimation for each marker that passes a unimodal dip test for p values $< .25$. It then repeats this process on subpopulations of cells split up by these initial gates. Gating strategies are terminated if there are too few cells, the process is done up to three times, or if no markers satisfy the dip test. The cytometree method uses the same recursive method and termination conditions but uses different ways to define its gate and quality score. It defines a single gate for each marker by modelling each marker's FI values as a mixture of two Gaussian distributions. Instead of the dip test, it accepts the marker's gate with the highest normalized difference between Akaike criterion (AIC) score above a user-given threshold. Unsupervised gate search requires that these methods also attach a gate unification step across samples. flowType [268], on the other hand, uses user-given gates already unified across samples to create an exhaustive cell population hierarchy that would contain all possible gating strategies and cell populations for cell population discovery and analysis.

3.6.5 Remarks

Clustering of cell populations in FCM bioinformatics is focused on finding and discovering various naturally occurring cell populations in terms of data distribution. However, for these methods to be useful, they should find cell populations that allow the user to understand the condition of the subject whose sample is being analyzed. To this end, efficient ways of purpose-driven cell population comparison across samples need to take place. For this purpose, unsupervised methods that use existing packages, such as Voom, EdgeR, Limma [381], and flowGraph (SpecEnr) [400] exist to identify M/DCPs.

3.7 Clustering in sc/RNAseq

Clustering in sc/RNAseq is done to analyze the abundance of various RNA strands to understand what genes are being expressed in a biological sample. In contrast with the pre-

vious section, this section focuses on a data matrix where the number of features (columns) greatly outnumbers that of objects (rows) [74]. This phenomenon is common in bioinformatics given the relatively high cost to analyze a sample versus the ever-growing amount of throughput per sample [296]. Unlike in FCM where manual gating is still the dominant way to identify cell populations, automated methods, including clustering, are almost always used over manual analysis in sc/RNAseq analysis [390]. Like the previous chapter, we will first go into the technology that produces sc/RNAseq data, how preprocessing, clustering and postprocessing procedures are performed on this data, and finally mention some alternative solutions and remarks.

3.7.1 RNAseq

When it comes to human identity, our genome, or our DNA (deoxyribonucleic acid), lies at the heart of who we are. To function, different sections of the DNA are copied inside of the cell nucleus and released into the cell plasma to be translated into proteins that would execute functions within our body. These short copies of the DNA are called RNA (ribonucleic acid). RNAseq data, in turn, represents what RNA sequences or sections of the genome are being used by a cell at a snapshot in time within a given sample. In the previous section, we talked about how FCM can identify cells using the proteins they contain. In this section, we focus on how these cells and cells' activities can be analyzed by understanding what sections of the genome are being expressed via RNA.

sc/RNAseq, or single-cell / whole transcriptome shotgun sequencing, is a protocol that produces data to analyze the RNA of any eukaryotic cell. Many papers also refer to this type of data as transcriptomics. This encompasses any protocol that analyzes the RNA or the transcriptome, including RNAseq's predecessor, the microarray. Microarray also analyzes the transcriptome but was made obsolete because it is not as good at quantifying very low and highly expressed genes. It also requires researchers to know precisely what sequences they are looking for beforehand. Nevertheless, the same algorithms used on microarray data can be and have been used effectively on RNAseq data [199]. Also, since the same methods can also be used across scRNAseq and RNAseq data sets, we refer to all of these methods as clustering methods for RNAseq.

Sample preparation contains three steps [74]. The first step is to 1) isolate desired RNA segments from a tissue. Usually, this includes filtering the nucleic acids and using an enzyme called deoxyribonuclease (DNase) to remove DNA from the sample. Once we are left with the RNA, the next step is to 2) isolate a specific type of RNA, mRNA. While 90% of the RNA in the cell is ribosomal or rRNA, we are interested in the 1-2% of RNA called messenger or mRNA which contains the final sequences that are to be translated into proteins. These are extracted using one of two methods: enrichment and depletion. Enrichment picks out mRNA by exploiting the fact that, unlike rRNA, most mRNA contain at the end of their sequences, a poly(A) tail which can be selected for using poly(T) oligomers. However, for

enrichment to be successful, it requires large amounts of mRNA with high RIN (RNA integrity number) i.e. there should be a minimum amount of degradation at sequence ends and lots of mRNA strands from known exon sequence regions. Exons here refer to regions of a genome that remains on an RNA after intron regions are removed by RNA splicing — note that this means intron regions are ignored in enrichment. However, some tissues may not meet these quality requirements (e.g. biopsy samples) and some organisms simply do not produce polyadenylated mRNAs with poly(A) tails (e.g. bacteria). Therefore, another viable option is to remove the rRNAs using compounds such as specific locked nucleic acid (LNA). 3) Finally, once mRNAs are isolated, these are reverse transcribed into optionally labelled complementary or cDNA. Then, these are amplified and fragmented with enzymes into appropriate amounts and lengths to be analyzed as reads. Note though there are newer protocols, such as third-generation sequencing tools (e.g. PACBIO and Oxford Nanopore) that provide longer read sequences. Since the output of these technologies once prepared can be analyzed by similar tools, we will not differentiate between them in this review.

scRNAseq and depth

When one conducts an experiment, they need to choose what they want to analyze as their objects [254] and the resolution or depth of their features [254, 349].

Object-wise, in RNAseq, the object is a biological sample such that all the cells within a tissue are ruptured and the RNA materials are pooled, processed, and then analyzed; recently, scientists have developed a more granular option to analyze a single cell as an object in *scRNAseq*. In scRNAseq, each cell is unruptured and kept intact before being put through an extra preprocessing step. The cells are isolated into tubes, wells or droplets via manual or microfluid separated fluorescence-activated cell sorting (FACS). These cells are individually lysed and their mRNAs are reverse transcribed. Depending on the protocol used, the resulting cDNA (complementary DNA) may be already attached to cell-specific identification barcodes or these barcodes can be attached after amplification and fragmentation of the cDNA. Afterwards, the cDNA are pooled and analyzed as with any other RNAseq experiment.

The other dimension, depth, affects the quality of the features we are to analyze on our objects. *Depth* is the average number of times any original mRNA nucleotide is sequenced and amplified. More depth means that any sequence is more abundant and therefore the chance of it being picked up accurately over noise during the analysis is higher. Although it would be ideal to have infinite depth, there is a limited monetary budget. For highly expressed sequences in most eukaryotic cells, 5 million reads will usually suffice, while up to around 100 million reads have been used for detecting lowly expressed sequences [326]. In the case of scRNAseq, costs are further amplified as even smaller numbers of mRNA sequences per cell need to be amplified. In this case, researchers need to balance an additional trade-off between cell count and read depth. Usually, droplet-based scRNAseq scans 20,000 cells at

a sequencing depth of 10,000 reads [64]. If depth is of importance, some studies have set their depth to 50 million while analyzing less than a few hundred to a thousand cells [210].

Data acquisition

Regardless of the object of analysis and depth, RNAseq outputs a $n \times m$ matrix consisting of the object (sample or cell) on the rows and the sequence features on the columns. The content of this matrix is a count proportional to how many mRNA strands contain a certain sequence. We acknowledge that in RNAseq papers, this matrix is referred to as a $m \times n$ matrix where the features and objects are on the row and column of this matrix respectively. However, we transpose this matrix to keep it consistent with the input matrix of FCM.

When the raw data comes out, the original features are simply the read fragments we talked about earlier. To extract the meaning behind these reads, some preprocessing procedures need to be done. The first step is to control for quality. To account for duplicated reads, sequencing error, contamination, and other issues, researchers use standard tools such as FastQC (popular on the Illumina company platform) [97], NGSQC [79], FASTX-Toolkit [142], and Trimmomatic [46]. Some things they look out for include degradation in read quality going towards the 3' end of reads while temporarily or permanently removing artifacts and labels such as adaptor sequences and barcodes for collective analysis. After obtaining the quality-controlled version of the reads, these can either be used directly for analysis. In this case, the sequence features would simply be unique reads [384]. Most of the time, they are further annotated by being aligned to a corresponding genome or transcriptome using aligners such as RSeQC [377] and Qualimap [117]. Taking the human genome as an example, they can be mapped to databases such as Ensembl [161] or USCS [180]. By aligning reads to these data sets, we would know which genes or transcripts the reads belong to. This informs the users on which genes are being expressed in the sample or cell, and this allows for features to have annotated meaning. Other forms of annotation include linking or assembling all the reads via overlapping sequence bits. Tools that do so include SOAPdenovo-Trans [391] and Trinity [129] (and for longer reads [129]). In summary, each feature can correspond to a certain nucleotide base [70], read, a certain sequence, transcript, or most commonly, a longer sequence annotated as a gene. For simplicity, when we refer to our features, we will refer to features as if the reads were aligned and annotated at the level of a gene.

Now that the features are set for the $n \times m$ matrix, many methods would assume that further data preprocessing be done. The first one of those is normalization for RNAseq samples. One example of this is to adjust the count values such that the increase or decrease in count between objects is comparable [137]. When different samples have different cell counts, we may be misled to believe more of a certain gene is translated in one sample than in another when in reality, that sample just has more cells expressing the gene. One solution can be to simply use scRNAseq. However, a more economical solution is to use methods

such as the TMM (trimmed mean) from the edgeR package [298]. By assuming that most counts do not change, it takes the normalized mean ratio of all counts in one object over a reference object. This ratio is then multiplied with all counts in the non-reference object to get the normalized counts. Other tools that accomplish a similar purpose include the relative log expression (RLE) from the DESeq2 package [224], median ratio normalization (MRN) [247], PoissonSeq [213], and for certain cases, quantile normalization [153]. Additionally, one can choose to rid of small technical or external confounding factors using tools such as Rhcpp [261], RUV [295], COMBAT [340], and PEER [339]. Most of these tools model the normalized data as a linear combination of true expression counts, confounding factors, and an irreducible error term. These can be estimated by statistical sampling and other optimization algorithms. Finally, the count values are typically logged during preprocessing, or methods can specify otherwise.

We refer to this preprocessed $n \times m$ matrix as our input data set. The rows can either represent samples or cells and the columns we will refer to as features or genes.

3.7.2 Motivation, application, & problem

Through clustering scRNAseq cells, we can understand the cell type composition in a sample as we have done in FCM — but this time, instead of identifying cells using markers, we do so with gene sequence counts [254, 322, 362, 405]. On the traditional RNAseq side, we can compare how different experimental conditions relate to each other by clustering the RNAseq samples they affect.

Given that we know what clusters the objects belong to, one task thereon after is to interpret which features are DCPs and differentiate between these clusters (analogous to Problem 1). A more formal term for this research problem is differential expression (DE) analysis. The first DE metrics compared features based on pairwise object clusters to see whether the features' distribution is bimodal or unimodal. In this category, methods have explored metrics such as the p-value on t-statistics and chi-square tests for binary data. Other examples of these metrics include kurtosis and bimodality index (BI). Kurtosis here is the fourth standardized moment which tests the heaviness of tails in a distribution [29], used in methods such as PACK [354]. Meanwhile, BI, an alternative to the popular BIC metric has risen in popularity because it can also rank genes according to their bimodality as opposed to only indicating whether they are bimodal or not, as in kurtosis. However, these statistical distribution based tests require that the assumption on the data distribution to accurate and is not robust when there is too much noise [94, 149]. Another common way to tackle this problem is via multiple/linear regression. This method is still used to this day, commonly via package limma [327]. With the addition of significance testing, limma typically provides a good list of features that researchers may be interested in. Other packages incorporate statistical models, such as using the Poisson or its generalization, the negative binomial distribution to represent feature counts and evaluate their significance for DE [297, 17, 143].

These packages assume that for each feature, its counts should form separate distributions under different object clusters. The parameters of these distributions are often estimated using a sampling approach based on Bayesian principles, with hyperparameters that can be optionally tweaked by the user [348]. While there is no consensus on how to best estimate DE, researchers usually use many packages together to interpret and communicate the clustering results they desire [317, 293, 329]. Another way to perform DE is to incorporate it into the clustering process by clustering both dimensions at the same time. We will dive into methods that perform this later in the chapter.

Knowing what features correspond with what object clusters have significant implications in biological systems analysis. One way to utilize the object and feature clusters is to use the found features as biomarkers as we have done for FCM in Chapter 4 [189, 136, 135, 352]. When given a new object, one can use biomarkers to identify what cluster, condition, or tissue the new object may belong to. Biomarker discovery and verification have significant implications — one of which is that it may guide protocols on what to test for in clinical diagnosis [390]. Systems analysis also implicates what biological pathways are affected/s by these features or object clusters. One way to do so is via enrichment analysis e.g. hypergeometric tests [56] on existing databases. Enrichment can also be seen as a query to a database to see what processes the biomarkers are involved in and see if features affecting the same cluster are involved in a similar process(es). Existing databases include interaction networks KEGG [179], the gene ontology [23], DAVID [321], Babelomics [251], protein sequence dataset, Swissprot [28], conserved protein domain PFAM [107], InterPro [162], and RNA sequence repositories Rfam [118], and mirBase [195]. Knowing what pathways these features are involved in gives meaning to the state of a cell population or sample cluster. In addition, these states can also help researchers predict what shape and even function(s) the translated proteins may perform. For example, if they are involved in communication between cells then perhaps these genes are involved in changing the state of a cell. In addition, if the protein associated with our biomarker is similar to another protein involved in a particular pathway, we could deduce that the former protein may also be involved in that pathway [126, 51]. If hierarchical clustering is performed on the features, researchers could even assume an inter-gene hierarchical functional relationship [346]. One can also use the feature clusters to simplify existing knowledge bases. As well, if a temporal aspect is measured, the cell and sample states can also be analyzed on a timeline to understand the progression of expression in the observed system [236].

To put the clustering and DE to use, a popular application of these results is in precision medicine. The goal of precision medicine is to improve patient outcomes by applying precise treatment based on a person's phenotype. However, to create a custom treatment for every person is expensive, and it is difficult to show that a treatment's success is statistically significant when testing a medical hypothesis for approval. On the other hand, for diseases such as cancer, treating the entire population the same way may not be viable i.e. one treatment

may not be effective on patients whose cancer cells express different gene sequences or mutations. One of the first examples of precision medicine that expanded the field attempted to group cancer patients into subtypes based on gene dysregulation [358, 233]. Therefore, clustering patients into small groups and developing treatments for those individual groups has become a suitable compromise [273, 71, 62, 151, 222, 78]. Amongst the precision medicine-related papers released to date, transcriptomics makes up the second-largest set of data used in their methodology, after genomics data [273]. With most of the researchers using clustering as their main methodology for this application, a growing number of papers are starting to use these results to inform treatment decisions directly, especially in the domain of cancer — breast cancer in particular [208, 245].

On the topic of treatment, another major player interested in the clustering of RNAseq data is pharmaceutical companies. One purpose here is to hypothesize what compounds might react to what drugs i.e. trug-target interaction. RNAseq clustering allows one to understand what patient group expresses which genes and therefore what compounds may exist in that patient group's biological system. These compounds can then be queried to see which drugs may be effective in those particular patient groups. Given biomarkers of patient clusters on whom a drug has been effective on, researchers can match their biomarkers to that of new patients and see if the same drug can be applied. Furthermore, assuming that similar compounds have similar effects, companies can use clustering to see if they can repurpose existing drugs to target new compounds for involvement in treatment. RNAseq clustering also proves to be useful during the drug testing phase. Here, by experimenting with different drugs, and analyzing the effects of those drugs on gene expression, we can evaluate the drugs' efficacy.

To impact these application contexts, one needs to start with clustering of RNAseq data, whose methods we will go through in the following section.

Given a large number of features, a challenge in RNAseq clustering is the need to understand exactly which features are significantly associated with which cell populations or sample clusters. Having mainly looked at 1D clustering methods in the previous section, in this section, we also discuss the different multidimensional clustering techniques that have emerged. These are not only the object dimension but also the feature dimension. We maintain the organization of clustering techniques by their phases: preprocessing, clustering, and postprocessing. Given the dimensionality differences, RNAseq clustering methods have different challenges they face. Therefore, the tasks in each step may differ from that of FCM clustering methods. More specifically, some challenges RNAseq clustering methods face are as follows.

(a) In contrast with FCM, there is little emphasis on finding a particular type of cluster of interest; instead, we aim to find *robust clusters*. RNAseq is prone to noise, outliers, and missing values amongst batch effects [53, 188, 48]. This issue is magnified by the

fact that there are few objects to cluster and depth or feature quality decisions vary across the experiment.

(b) Another challenge is the emphasis put on *multidimensional clustering*. Since understanding what features contribute to the object clusters is important, RNAseq motivated the creation of a specific type of subspace clustering, biclustering, and more recently triclustering. Although one can put aside DE analysis until after object clustering, simultaneously clustering all provided dimensions is becoming more and more popular. However, the imbalance of dimensions (small n, large m) and different dimensional domains (e.g. temporal, context, gene, object) poses a challenge to finding multidimensional clusters that make sense (e.g. temporal dimension should be order-preserving) [238].

(c) As with any task is desirable to be able to have methods *automatically specify parameters*, especially K [52, 355, 364, 262, 187]. For this, a variety of quality metrics are put in place to guide methods on how many clusters are a good amount to go with. We will see that similar tactics used in FCM are also used in RNAseq, so we go over this challenge briefly while describing the methods.

(d) Finally, the clustering results should be *interpretable* to the user. In this section, we not only focus on visualization but also on how methods try to integrate other data types to interpret the context for why the clustering results turn out the way they did. For example, what do the features that differentiated the object clusters have in common in the context of a biological system.

3.7.3 Preprocessing and imputing missing data

Before clustering, many methods choose to prepare the data such that it becomes suitable for analysis. Many preprocessing steps are there to deal with the missing data, noise, and outliers that are common in RNAseq data. As well, to deal with high-dimensionality, some methods choose to conduct an extra step of feature selection (filtering in good features), dimension reduction (representing features in alternative space) and use alternative data representation techniques to summarize the data. While many of the latter two strategies overlap with that of FCM, all of these become more important due to the much larger features space in RNAseq data — reminding us that many clustering methods do not scale well in high-dimensional space and the need to mitigate Challenge (a) for finding robust clusters. As many RNAseq methods opt to merge the process of finding K for Challenge (c) with its clustering algorithm, we will defer then to when we describe the clustering solutions themselves.

To mitigate Challenge (a) for finding robust clusters, the amount of missing data in D should be minimized. Traditional methods of dealing with this include imputation or

the process of filling in counts for missing genes. One way to impute missing values is via a probabilistic model. Given a data set from the general population, if the counts of known genes are of the observed value for a specific object, one can find the most probable expression value for the missing gene [147]. Other methods for dealing with missing data include network smoothing [302], using an autoencoder [345], and matrix factorization [413].

Data representation

Methods can use either a matrix D or a graph as input. In matrix form, methods can choose to directly use the original matrix, transform the values in the matrix, or they can opt to downsize the matrix as in the following sections. Transformation allows one to bring D into nonlinear space. As in FCM, it is also common in RNAseq to use log or Box-Cox transform [360]. If one chooses to use a graph as input, methods often face the issue of having to decide on an appropriate metric to accurately calculate a distance or similarity between objects. We saw that in FCM, many methods opted for traditional distance metrics such as Euclidean and Manhattan. These metrics assume that each feature is independent and can be biased toward highly expressed genes. Therefore, RNAseq clustering methods often opt for scale-invariant methods such as the cosine [188, 156], Pearson correlation, and Spearman's correlation distance metrics [406] — which fortunately have shown empirically better results for RNAseq [173, 344], especially when used on stochastic clustering methods such as Kmeans (or Kmedoid, where the cluster centre is the median) [132].

Feature selection

Feature selection is the process of picking out good features while removing noisy or low-quality ones. Because there are no standards on what should be and should not be removed, many tools resort to heuristics to conduct this step. For example, it is still common for one to simply delete 10% of the lowest average expressed genes in a scRNAseq experiment [90]. Seurat [311] and [48] use a more heuristic assumption in that they assumes it is okay to remove genes y_j with low variance. Hence, the highest quality genes are those ranked with a high coefficient of variation. They assume that genes with a homogeneous count across all objects are unlikely to differentiate between objects of different clusters. Instead, these genes may simply be noise if, their counts are low, or housekeeping genes that are used as a control for experiments that want to identify highly variable genes. A term to describe these kinds of assumptions is the highly variable genes (HVG) method of gene filtering. Furthermore, M3Drop [18] focuses on dropping low-quality genes, or genes that have too many missing values. This situation may happen when there is a failure in reverse transcription or enzyme reactions. It assumes that the number of missing values, or dropout rate $dropout_j$, a gene has should conform to the Michaelis-Mentin function of that gene's average log expression

μ_j

$$dropout_j = \frac{\mu_j}{\mu_j + M}$$

where M is the Michaelis constant. This means that the lower the μ_j, the more dropouts a gene should have since rarer genes are harder to detect. More specifically, the dropout should fit onto a Michaelis-Mentin equation as a logistic curve, and those genes that deviate away from this curve should be removed. Note that if having values of 0 is equivalent to having missing values, deviation from this rule on the opposite side of the curve can also mean a DE gene. In a similar respect, [192] rank genes by how far they are from a moving median while also deleting low-quality genes that have a mean count under 10 million reads. After log transforming the count values, the moving median here is the fitted line between the mean gene counts and the $MCCV_j$ mean-corrected coefficient of variation (CV_j). Hence the distance from this median is:

$$DM_j = MCCV_j - MED_j$$

$$MCCV_j = log_{10}CV_j^2 - MCR_j$$

where MCR_j is the mean corrected residual of CV_j^2 and MED_j is the residual produced upon fitting $MCCV_j$ on the log transformed gene length of gene j. The length of each gene here is calculated as the union of all of its exons according to the Ensembl database.

One can also get rid of low-quality data by identifying outliers. An example of this is DensityCut [85]. DensityCut uses random walks, a process that walks through the edges of a graph with a probability proportional to the weight on each edge. Through several iterations of such walks, one can determine the proportion of times each node has been passed through. The ones that have been passed through rarely can then be labelled as outliers.

Dimensionality reduction

Another way to reduce the number of features of a data set is via dimensionality reduction. One way to do so is through PCA. PCA scales well with large data sets, however, it assumes that the data is distributed as a Gaussian in linear space. With or without modifications, PCA is the most commonly used dimensionality reduction method in RNAseq to date [19]. It is used in methods such as PCA-based Seurat [311], Ascend [316], CIDR with zero-imputed similarities [217], [397], SC3 [189], TSCAN [172], and iWGCNA [254]. Another common method to conduct dimensionality reduction is t-SNE. Since t-SNE preserves distances between points at a local level, it is a popular choice as a visualization tool. Nevertheless, it still proves practical for dimensionality reduction as it is used in many state-of-the-art tools, such as Monocole [287]. Like t-SNE, diffusion maps [288] are also able to represent data in nonlinear space. However, in contrast to t-SNE, it preserves both local and long-

distance relationships between points. Nevertheless, it has a strong assumption that the data presented is smooth in low dimensions so it is more suitable to be used on data with more than 1000 objects that can be clustered into a few distinct clusters — a viable choice for some large scale experiments and scRNAseq [288]. Finally, and more recently, methods have opted to use autoencoders to learn a non-linear embedding of the original data. For example, scCCESS [122], randomly splits up the input, column-wise, into N matrices and trains N autoencoders to produce a lower-dimensional embedding of the original data.

Deep learning-based imputation

Starting in 2018, we saw a profound increase in the application of deep learning; imputation was no exception [374]. The most commonly used architecture for self-learning deep learning models is the autoencoder. The autoencoder is a multi-layer network that contains an input layer, one or more fully connected encoding layers, an embedding layer, one or more fully connected decoding layers, and an output layer. The input and output layers are of size m, and the encoding, embedding, and decoding layers are of size $< m$ with the embedding layer being the smallest layer. AutoImpute [345] was amongst the earliest methods to use the autoencoder for imputation. As done traditionally with autoencoders, they train the autoencoder with the original input data and force it to reproduce the input at the output layer. The assumption is that if a smaller embedding layer can be used to produce the original input, then the embedding layer should be a sufficient lower-dimensional representation of the input. Since we train the autoencoder on many objects, then the embedding layer should be a representation that can reproduce a version of the original input that is free of noise, batch effect, and missing data. Though later methods also used the autoencoder, the main differences between these methods are either that they use different types of regularization layers to optimize for some objective, or that their method has a post-processing step where they feed the encoding of their encoder into an extraneous neural network to attain desired results. Other methods that use the autoencoder for imputation are SAUCIE [15], scScope [82], and scGNN [376] which embed this step with clustering.

Later on, DCA [93] also came up with an autoencoder imputation method but it assumed that scRNAseq, unlike RNAseq, does not conform with the ZINB (zero-inflation negative binomial) distribution but instead that counts produced based on UMI (unique molecular identifiers) follow a negative binomial distribution. Therefore, it added a layer to regularize or force the model to fit the input data onto this distribution. GraphSCI [292], on the other hand, assumed that they could obtain better imputations by accounting for gene-gene interaction by inputting an existing gene-gene interaction graph along with the given RNAseq transcript counts into a graph convolutional neural network. The vector embedding given by this network is then put into an autoencoder to obtain the final imputed data set. scIGANs [393], on the other hand, experimented with using a GAN (generative adversarial network), a popular network used to generate images. Its generative model generated a 100-

gene matrix or image-shaped output containing random values for each cell type. This model is trained until the discriminative model cannot tell the difference between the generated data and the original input data. Imputation of the input data is done based on the generated data. An issue with this approach is that it requires preliminary knowledge of the cell populations first or generates it through KNN.

Although scGNN and scIGANs significantly outperform other methods, they are significantly slower than the alternatives (hundreds of seconds for 50 cells) [374]. To improve efficiency, DeepImpute [21] separated the input data set column-wise groups and trained each group on their autoencoder such that the number of nodes in each layer of the autoencoder can be reduced. DeepMc [259] Also tried to reduce the number of features by performing matrix factorization and selecting only the 1,000 top features to train a 4-layer neural network shaped like an autoencoder but without the embedding layer. While faster than the alternative, even the fastest of these deep learning imputation methods still perform significantly slower than the alternatives [374].

3.7.4 Clustering

Traditional clustering methods

Motivated to analyze heterogeneity among cancer patients, many homegrown statistical tests to find DE genes [277, 145, 260] and clustering methods have been created. The earliest of those were created to find bimodal clusters. Setting $K = 2$ by default, methods like PACK (profile analysis using clustering and kurtosis) [354] and its predecessor [353] runs several instances of GMM. However, high dimensionality means that the data tend to be sparse, so if the data has a heavy tail or has extreme values, these basic procedures may result in false-positive results. However, it is effective if the assumption that the data distribution is normal holds. To understand how the data might fit on other distributions, SIBER [360] applies three types of distribution for its mixture model on Box-Cox transformed D: Gaussian, Poisson, and negative binomial distributions (and in some cases the t-distribution [39]). The Gaussian distribution is used for continuous data, while the Poisson and negative binomial distribution are specialized for binary or count data — with the latter being popular due to its ability to model overdispersion, where genes with small counts have especially high variance and uncertainty. Model-based clustering algorithms would eventually go on to find multiple clusters as in [214] with GMM, and TSCAN [172].

Hence, this brings us back to the importance of being able to find robust clusters as in Challenge (a). One approach in this step is to use consensus clustering. This approach combines many clustering instances of single or multiple clustering methods to remove inconsistencies and keep reliable clusters. Examples of this types of method include MetaCell [30], SC3 [189], and PCAReduce [397, 134, 217]. As in PhenoGraph, MetaCell [30] takes as input a similarity matrix A, creates a KNN graph, and makes a new similarity matrix where the similarity between objects is based on how many shared neighbours they had

in the KNN graph. It then samples a subset of objects several times, each time applying Kmeans clustering. Using these Kmeans results, a final similarity metric is generated based on how many times each object was clustered together in the sampling process. It then applies another round of Kmeans to this final similarity matrix and outputs this as the final result. Aside from clustering several times on subsets of data, one can also choose to cluster several times on the whole data. SC3 and PCAReduce start from a PCA dimensionality reduced matrix of D or the eigenvectors of the D's graph Laplacian whose distance between nodes is defined by the Euclidean, Pearson, or Spearman distance metrics. From there, they over-clustering with Kmeans several times to understand which objects are always clustered together. Again, the number of times an object is clustered together is then treated as the new similarity matrix between objects. Finally, they use this matrix to conduct hierarchical agglomerative clustering to obtain the final results. SAFE [396] and Clust [3] are examples of methods that combine or ensemble together results from multiple clustering algorithms. SAFE merges these results by first using cluster validation indices, such as the Jaccard, to find the best overlapping clusters. It then creates a distance matrix that contains the distance between all pairwise clusters. This matrix is converted into a graph on which it partitions to create meta-clusters i.e. the final clustering result. Consensus clustering is especially beneficial for stochastic algorithms that have random initialization or can easily fall into a local, not global optima. Ensembling together multiple results also help get rid of noisy inconsistent results — which have been shown to occur consistently for all methods [254]. Consensus clustering can also help to aggregate clustering results that have used different K values to pick out which K provides the most consistent results to mitigate Challenge (c) [52, 355, 364, 262, 187].

A clustering method that does not require an explicit K to initialize is hierarchical clustering. Given its natural ability to help users visualize its results via a dendrogram, it is the most used clustering algorithm for RNAseq, in particular for the patient stratification application [273]. Dendrosplit [406] opts to test each iteration of hierarchical clustering with a quality statistic to understand what K to finish off clustering at. It starts with a similarity matrix A made via Pearson or Spearman correlation. Using this, Dendrosplit performs agglomerative hierarchical clustering with the complete link — the maximum of pairwise distances between the points of two clusters. It stops merging any two clusters if $-\log min_j p(D_{I_1 j}, D_{I_2 j}))$ become lower than a threshold. p here is the p-value from the Welch's T-test for the jth gene of clusters containing object indices from subscript indicated clusters 1 and 2, I_1 and I_2. Finally, it is also worth mentioning other distance-dependent method groups including density-based and graph-based clustering. Like in FCM, density-based clustering is popular for clustering scRNAseq data because it can find arbitrarily shaped clusters [234, 55, 174, 312, 312]. An example of a method that uses density metrics in conjunction with hierarchical clustering is DensityCut [85]. DensityCut initializes by over-clustering. Using the random walk results found previously, it creates a density estimate

from the number of times each object is walked over during a random walk. It assigns the objects closest to the density peaks as cluster centroids. Using these centroids, objects are moved along their gradient directions toward each centroid. Whichever centroid they move towards defines the cluster they are assigned. This is done via the hill-climbing search algorithm [194]. It then takes these clusters and applies hierarchical agglomerative clustering to see which clusters should be merged. The link here is defined on the density distribution previously created where for every two clusters, the lowest point or density valley between the two clusters is compared with the lesser of the two centroids or the height of the lower density peak from the two clusters. If this ratio is above a certain threshold, the two clusters are merged. On the other hand, graph-based algorithms have also taken off with notable packages such as Seurat [311] which successfully applied Louvain onto PCA reduced RNAseq data. Note that again, since many hierarchical methods use some form of a distance metric, they can be sensitive to noise and outliers in a large feature space [405].

Multidimensional clustering

Challenge (b) represents a unique problem in data sets with a large or multiple feature space(s). In precision medicine, for example, it is more informative to simultaneously select for and understand what features contribute to making a patient group unique. These features can inform downstream decisions such as etiology and treatment [242]. Up until now, we have been clustering in a single object dimension. Multidimensional clustering, on the other hand, clusters multiple dimensions at once. The terms, biclustering and triclustering, were first coined in the bioinformatics field to define methods that cluster two and three dimensions simultaneously respectively. Their ability to help users interpret results allowed them to grow in popularity and enter mainstream use today [152]. In triclustering, the additional dimension would often represent a context such as time, location, and condition.

In this section, We start by discussing the general heuristics and statistical tests used to define a good multidimensional cluster, whether that be a bicluster or tricluster. We then go into some of the algorithms used to find clusters that satisfy these criteria. Hence, the goal of multidimensional clustering is to find a subspace in D that satisfies some homogeneity criteria at a specified quality or statistical significance. Note that most of the methods mentioned are designed specifically for biclustering but many of them can be extended for use in triclustering.

Formally, each cluster k here is a subspace of D with indices $I_k \subseteq \{1, ..., n\}$ and $J_k \subseteq \{1, ..., m\}$ for biclustering, and additionally the third dimension $H_k \subseteq \{1, ..., s\}$ for triclustering. For convenience, we call the first dimension objects, the second dimension features, and the third dimension slices. As we refer to rows as x_i and columns as y_j, in the triclustering, we refer to a matrix slice in the third dimension as z_h. As well, we stick with our notation of having D_{ijh} represent a subspace or element of D. Finally, we represent the mean of a subset of D via μ. For example, if we say μ_{iJH}, this represents the mean of x_i.

Heuristics Usually an element in D can be modelled as a constant $c \in \mathbb{R}$ that is effected by an irreduceable error term $e \in \mathbb{R}$ and its cluster $\Theta_{ijh} =$ in some combination of α_k, β_k, γ_k, where $\alpha_i k = D_{iJ_kH_k}$, $\beta_j k = D_{I_kjH_k}$, $\gamma_h k = D_{I_kJ_kh}$.

$$D_{ijh} = c + \Theta_{ijh} + e_{ijh}$$

We assume element D_{ijh} is a part of none or one to few cluster(s) k which consists of indices (I_k, J_k, H_k).

One of the most common objective function is the mean squared error (MSE)

$$MSE = \frac{1}{|D|} \sum_{i,j,h} (D_{ijh} - f(i,j,h))^2$$

where $f(i,j,h)$ is the algorithm modelled D_{ijh} after applying effects of the found clusters. The main goal of this objective function is to understand how well a methods' model reflects that of the real data. Another set of objective functions are standards of quality that a cluster needs to meet at a threshold or maximize.

In either case, what these objective functions help to achieve is homogeneous quality. Homogeneity is a heuristic that defines the assumptions a method has about a good cluster. These assumptions can pertain to several aspects of a bicluster or tricluster and can vary depending on user needs. Some of these assumptions can be structural aspects of a cluster, such as shape, size, overlap, and position of the cluster. These are usually defined intrinsically based on how an algorithm is designed. For example, most algorithms require clusters to be maximal. A maximal cluster, is a cluster that meets a homogeneity criteria and at the same time does not wholly contain a smaller cluster that does so as well i.e. cluster $Q = (I, J, H)$ is maximal if there is no (I', J', H') s.t. $I \subseteq I' \wedge J \subseteq J' \wedge H \subseteq H'$. Algorithms that stick to this criterion include hierarchical clustering. It implicitly requires clusters to be convex with high density. Since it iteratively merges clusters according to this criterion, it would not output a result that has two clusters where one is wholly contained in another. Other criteria include sensitivity to distribution of noise [164, 324], missing data (an ongoing challenge [414], and different data types (continuous, discretized [325, 392], or symbolic data [150]). These can be controlled by error terms added to a homogeneity criterion modelled by, for example, a Gaussian distribution. Meanwhile, the handling of different data types can also be handled by having multiple objective functions. However, putting aside extraneous requirements, the crux of a homogeneity criterion can be categorized as those trying to find clusters with values that are:

- Constant: all elements in a cluster have similar values i.e. $\Theta_{ijh} = 0$; or all elements in a row/column have similar values.

- Additive: all elements in a cluster is effected similarly by a summation unique to a cluster e.g. $\Theta_{ijh} = \alpha_{ik} + \beta_{jk} + \gamma_{hk}$.
- Multiplicative: all elements in a cluster is effected similarly by a product unique to a cluster e.g. $\Theta_{ijh} = \alpha_{ik}\beta_{jk}\gamma_{hk}$.
- Order preserving: elements in each row/column increase or decrease in additively/multiplicatively in a way that is consistent between each row/column — essential for when there is, for example, a temporal dimension to the data.

The earliest model looked for constant clusters given a binary matrix D. For example, [197] evaluates biclusters with the maximum fraction of 1's inside in the maximum size subspace of D. This can be generalized to tolerate noise [402] according to how homogeneous one desires the clusters to be [24]. In real valued matrices, the clusters would have a similar value across the board. For this, [164] tests the biclusters for unexpectedly low variance and unique mean compared to that of the whole D — usually when modelling a clusters' values as a Gaussian. [235], on the other hand, loosens this definition such that a constant cluster can simply be the same across rows, as in $D_{ij} = c_j + \alpha_i + e_{ij}$ where $\alpha_i = 0$, or columns.

Additive clusters follow a model called plaid [407]. Here we speak to it as if it is defined as a biclustering method, but it can be generalized to any dimension. Regardless of the effects of the objects and features, plaid assumes that each cluster has an overarching additive effect on its values. A value is therefore defined as the 'layers' of these effects added onto it. With this assumption, plaid allows for overlapping clusters in the case that the effect of the two clusters is additive. As such, within our notation, Θ_k can be defined as follows.

$$\Theta_k = \sum_k \Theta_{ijk}\rho_{ik}\kappa_{jk}$$

$$\rho_{ik} = \begin{cases} 1 & \text{if } i \in I_k \\ 0 & otherwise \end{cases}$$

$$\kappa_{jk} = \begin{cases} 1 & \text{if } j \in J_k \\ 0 & otherwise \end{cases}$$

where $\Theta_{ijk} = \mu_k + \alpha_{ik} + \beta_{jk}$ (μ_k is the mean or simply a base value of cluster k) such that the latter two terms are optional. Including the latter two terms will allow users to find biclusters where different rows/columns of a single cluster can have its own effects. This additive model allows users to find objects or genes where their values are either over or underexpressed. Users can also choose to find only either one of those by adding on a constraint such that Θ_{ijk} always has the same sign, either positive or negative. Given its model, this heuristic then lends itself well to be converted into an objective function based

on MSE:

$$min \sum_{i,j} (D_{ij} - \sum_{k} \Theta_{ijk} \rho_{ik} \kappa_{jk})^2$$

Multiplicative models are where $\Theta_k = \alpha_i \beta_j \gamma_h$. Like additive models, it assumes values are effected by a multitude of cluster-based factors, but as a product rather than a sum. Hence, this assumption lends itself well to matrix factorization. Matrix factorization takes as input an $n \times m$ matrix and splits it up into a product of an $n \times m'$ and an $n' \times m$ matrix where $n' < n$ and $m' < m$. The new dimension of the these two matrices are termed factors. As a dimensionality reduction tool, these factors can then be used as new features to cluster the objects or features respectively. In the multidimensional clustering case, each factor can represent a fuzzy cluster label of the rows and columns. Since matrix factorization methods are flexible in the terms that it can include in its objective function, one can consider many aspects of what they want to focus on. For example, one can put in a term to help simultaneously rid of confounding factors. In this case, D would be represented as a sum of factor products: (known sample covariates: object factors $(n \times M) \times (M \times m)$) + (known gene covariates if available: object factors $(n \times L) \times (L \times m)$) + (unobserved effects: $(n \times N) \times (N \times m)$). Under this assumption, the mean expression of row i and column j can be written as

$$ln(\mu_{ij}) = E(D_{ij} | b_{ij} = 0, B, C, D)$$

and the probability of dropouts as

$$logit(\pi_{ij}) = P(b_{ij} = 1 | B, C, D)$$

where D_{ij} is the value in the count matrix, $b_{ij} = \begin{cases} 1, & \text{if } D_{ij} \text{ is a dropout} \\ 0, & \text{otherwise} \end{cases}$. As with other model-based methods, the parameters here can be estimated using optimization procedures such as sampling. Though similar, note that this model is different from remove unwanted variation (RUV) [295] for eliminating confounding factors — because here, the unobserved effects may not necessarily be but can be unwanted confounders.

Order preserving homogeneity assumes that each row/column has the same relative pattern of the increase or decrease of values [235]. The order is important if there is a dimension that makes sense to have order consistency [392], for example, a temporal dimension — as one may be interested in how the expression of cells or patient samples change over time. One possible way this change could occur is through partial orthonormality. Partial orthonormality is when the shift in the scale of values is correlated between time segments in a cluster. For example, if one gene increases in expression over time, the other genes should too. One of the earlier attempts to model this is by [392]. Given an ordered triplet of times $\pi(z_j) = z_{h_1} \prec z_{h_2} \prec z_{h_3}$, [392] selects for objects and features such that the S^2 score is lower than a user-specified threshold. It samples for feature triplets many times and tests

each one until this requirement is satisfied.

$$S^2(I, J, \{h_1, h_2, h_3\}) = max_{i \in I, \pi(y_j) \subseteq J} \frac{D_{ijh_2} - D_{ijh_1}}{D_{ijh_3} - D_{ijh_1}} - min_{i \in I, \pi(y_j) \subseteq J} \frac{D_{ijh_2} - D_{ijh_1}}{D_{ijh_3} - D_{ijh_1}}$$

As one of the earlier methods to explore this field, it does successfully reach its goal of finding ordered clusters. One reason for this is that the time dimension is usually very small, given the cost of producing the data. The difference in the size of each dimension also means that the larger (e.g. feature) dimensions can heavily influence the results. Therefore, enforcing small triplet-based clusters is a good idea to overcome Challenge (a), but this makes the method difficult to extend to other situations. Therefore, other methods use more generalizable metrics to test the quality of the cluster e.g. pairwise row/column/slice Pearson correlation and later the Spearman rank correlation for its trait of being shift and scale-invariant [173].

Nevertheless, this idea to treat different dimensions differently opened a new path in triclustering in that it does not have to follow a single homogeneity criterion for all dimensions. Many methods choose to first find good biclusters on all individual slices, and then connect them using a separate inter-slice homogeneity criterion [392] e.g. Pearson or Spearman correlation [11, 411], or cosine distance [156]. Meanwhile, [158] evaluates whether the difference between the average values of each slice in a tricluster is coherent ensuring that the rate of increase or decrease between slices is consistent. To the same end, another example of an inter-slice homogeneity criterion is a thresholded PMRS (planar mean residue similarity) [11]. Given two biclusters on separate slices (I, J, h_1) and (I, J, h_2),

$$PMRS = \frac{\sum_{i \in I, j \in J} |(D_{ijh_1} - \mu_{IJh_1}) - (D_{ijh_2} - \mu_{IJh_2})|}{2max(\sum_{i \in I, j \in J} |D_{ijh_1} - \mu_{IJh_1}| - \sum_{i \in I, j \in J} |D_{ijh_2} - \mu_{IJh_2}|)}$$

. Not only is this tactic good for when the third dimension is smaller in size, but it also customizes the homogeneity criteria for the different domains the dimensions are in. For example, correlation metrics are good for finding order-preserving relationships (e.g. in a temporal domain) while other metrics may be more suited for finding constant, additive, or multiplicative clusters (e.g. in an object vs feature domain). That said, note that the inter-slice homogeneity criteria need not always be order-preserving.

Other triclustering methods attempt to get rid of the third dimension altogether by collapsing the third dimension using some statistics. However, this means that all the triclusters found would contain all the indices from the collapsed dimension. This is viable for some situations, mostly when the collapsed dimension is small [325].

Algorithms Previously, We talked about different types of heuristics that pertain to different assumptions people have about a good cluster. Now, we go into the algorithms that can find clusters based on the heuristics described.

Considerations one would have when designing these algorithms is the structure of the cluster one is looking for. For example, whether one wants hard or fuzzy clusters [370] and non/overlapping clusters. Non-overlapping clusters are easy to visualize as one would simply need to reorder the indices in each dimension to obtain a good heatmap approximating the clusters on a matrix [313]. However, overlapping clusters are often more flexible and suitable for real-life scenarios. In the latter case may involve algorithms that simultaneously find multiple clusters. Whereas the former case, algorithms tend to remove already found clusters from D before moving on to find new clusters.

This procedure of finding one cluster at a time represents many of the early algorithms made for finding multidimensional clusters: iterative. Iterative algorithms aim to maximize or minimize a heuristic objective function according to an optional threshold. It does so by repeatedly adding or deleting rows/columns/slices to a cluster. A classic example of this is the iterative signature algorithm (ISA) [37, 165] on which many later methods expanded on [36, 25, 401, 12]. ISA is a biclustering algorithm that starts out with a binary matrix. This could be a matrix where a gene is represented as 1 if it is DE in a sample, or 0 otherwise. It then builds a directed graph where the nodes $i \in U$ and $j \in V$ are objects and features within their respective clusters (note, we exclude cluster indices for clarity). ISA ensures that the nodes included in a bicluster keep the variance of values in the bicluster to a minimum i.e. it finds a constant cluster. More specifically:

$$ISA(V') = \{i \in U | \sum_{j \in V'} D_{ij} > t_V \delta_{V'}\}$$

$$ISA(U') = \{j \in V | \sum_{i \in U'} D_{ij} > t_U \delta_{U'}\}$$

where δ is the standard deviation of the new subsets U', V', and t_U, t_V are user specified thresholds. In this case, a perfect bicluster would be where $ISA(U') = V'$ and vice versa. The algorithm initializes J' as a random or known feature set and U' as an empty object set. It then iteratively goes through each feature j and calculates

$$I' = \{i \in I | \mu_{iJ'} > \frac{T_J}{\sqrt{|J'|}}\}$$

For all the i's that do meet the criteria, an edge is placed between them and the said feature. ISA does the same for all is that were included above and commences to add another set of directed edges if any are found. Here, T_I and T_J represent object and feature z-score thresholds based on intra-cluster variance. An edge is added between these nodes if, considering the size of the bicluster, $\frac{|J'\ J''|}{|J' \cup J''|} < \epsilon$ where J'' is the old J' from the previous iteration and ϵ is a user given threshold. Finally, ISA finds communities of nodes that are interconnected with each other. For this, one can simply do a breadth or depth-first search to exhaustively find all communities or more sophisticated processes can be used.

For example, an extensions of ISA, Metafac [219,] finds these communities using non-negative multi-tensor factorization. [165] also extends ISA by initializing only maximal objects and feature sets via a heuristic-based greedy algorithm. [253], on the other hand, uses a sliding window to ensure inter-feature contiguity and uses enrichment constraints to get a biologically significant initial set of features. [178] Also uses enrichment databases on top of customizing the algorithm for use on parallel systems. This process of parallel processing can also be called hierarchical or divide and conquer; where the same iterative method of adding and deleting objects into clusters can simultaneously occur on all objects. Since simultaneous processing means multiple sets are created, Bimax [324], for example, deals with this by merging relevant subsets according to how highly correlated they are. Another extension of iterative methods is one where repeat additions or deletions of objects/features/slices are allowed, as in Cheng and Church [69], BackSPIN [405], along with many more that followed [220, 361, 299, 163, 231].

Additive plaid models [407] were also first optimized using a greedy iterative model. It takes as input, a log-transformed D, such that the Gaussian error term can be a robust distribution for noise [60]. It iteratively adds one layer of the parameters onto the model at a time to create something close to D i.e. minimize MSE. Eventually, methods got around to using a multitude of methods to optimize plaid. For example, [415] uses low-rank factorization optimized via coordinate descent, while [347], again, uses tensor factorization to reduce computational complexity. Other extensions include FLOC [394], which assumes no background layer and enforces that α, β, and γ cannot be 0 such that all elements in D must be clustered. xMotif [363], on the other hand, enforces $\beta = 0$ to find clusters one dimension at a time. Meanwhile, plaid can also be optimized by regressing D as the sum of plaid layers on a two-way ANOVA. Assuming that the MSE error term is a Gaussian distribution, it can also be optimized using the EM algorithm i.e. by optimizing for one of Θ, ρ_{ik}, and κ_{jk} at a time while keeping the other two fixed.

Though a bit more computationally expensive, model-based methods are flexible in that users can integrate prior knowledge into the clustering process to mitigate Challenge (d). Gibbs-plaid [61] is a variant of a model-based algorithm for plaid optimized via Gibbs sampling using the Wang-Landau algorithm. It incorporates a Gibbs field regularizer in the objective function. This Gibbs field comes in the form of a similarity matrix between genes based on the gene ontology such that the resulting clusters are encouraged to contain similar genes. This ability for users to optionally incorporate a prior makes model-based clustering a powerful and flexible tool for multidimensional clustering. These priors can also be for disease progression or subtype of patients [304], survival time [10], drug response, or experimental settings. One can also capture noise with local, as opposed to global, distribution parameters for each cluster. The parameters can even be specified such that certain shapes, sizes, and structures need to be respected.

In addition to data integration, model-based approaches can also incorporate an advantage to Challenge (c). Assuming that each cluster layer in plaid is a distribution, these distributions can be generated, potentially infinitely, by a DP to automatically find the best K. An example of a model-based clustering method that fits within all of these moulds is B2PS [183] and its extension SUBSTRA (supervised Bayesian patient stratification), a method that uses kernelized Bayesian matrix factorization [184]. It allows for the incorporation of prior knowledge on all dimensions and uses DP so that K is not required from users. It also controls for the shape of its biclusters by specifying that all elements in D must be a part of a matrix or have a cluster index latent parameter, making it exhaustive. At the same time, each element on a dimension can only belong to one cluster within that dimension, making the method inclusive. However, unlike many methods, since it separates feature cluster labels from object cluster labels, several object dimension clusters may contain the same features and vice versa. More specifically, SUBSTRA uses the Bernoulli distribution, a common distribution for binary data, to model expression values in D. The parameters of this distribution are then modelled by a Beta distribution.

$$D_{ij} \sim Bern(\theta_{g_i^p, g_j^q})$$

$$\theta_{k,l} \sim beta(G = 1)$$

$$g_i^p \sim CRP(\alpha^p = 1) \quad g_j^q \sim CRP(\alpha^q = 1)$$

where g_i^p and g_j^q represent the cluster indices of the i'th item in the p object dimension and j'th item in the q feature dimension respectively. CRP is the chinese restaurant representation of DP and the two α's are defined by a user reflecting the amount of clusters the method should tend to output. It also incorporates an optional feature weight vector $\mathbf{w}$ whose values reflect how many times each feature should be considered for input into a cluster. Such a prior can also be included for the objects $\mathbf{f}$. Hence the full probabilistic model amounts to

$$P(e, w, f, g_p, g_q | \alpha^p, \alpha^q, \beta, G) = P(g^p|\alpha^p)P(g^q|\alpha^q)P(a|\theta, g^p, g^q)P(\theta|G)P(f|\sigma, g^p)P(\sigma|\beta)$$

whose conditional probabilities of the latent variables (the clusters we are trying to infer), according to the Bayesian rule, is

$$P(p_i = k | \alpha^p, \theta, g^p_{-i}, g^q, w, f_i, \sigma)$$

which is proportional to

$$P(g_i^p = k|\alpha^p, g^p_{-i})P(a_i|\theta, g_i^p = k, g^q, w)P(f_i|g_i^p = k\sigma) = CRP(\alpha^p) \times \sigma_k(f_i) \times \prod_{j=1}^{m} (\theta_{g_i^p, g_j^q}(a_{ij}))^{w_j}$$

$$CRP(\alpha^p) = \begin{cases} \frac{\alpha^p}{n-1+\alpha^p} & \text{if } x \text{ is empty cluster} \\ \frac{|\{d|g_d^p=k \ \wedge \ d\neq i\}|}{n-1+\alpha^p} & \text{otherwise} \end{cases}$$

$$\sigma_k(f_i) = \begin{cases} \sigma_k = \frac{\text{number of ppl in cluster k with phenotype } 1+\beta/2}{\text{number of ppl in cluster } k+\beta} & \text{if } f_i = 1 \\ 1-\sigma_k & \text{if } f_i = 0 \end{cases}$$

$$\theta_{g_i^p,g_j^q}(a_{ij}) = \begin{cases} \theta_{g_i^p,g_j^q} = \frac{\text{no of 1's in cluster } (g_i^p,g_j^q)+\frac{1}{2}G}{\text{no of 1's in cluster } (g_i^p,g_j^q)+G} & \text{if } a_{ij} = 1 \\ 1-\theta_{g_i^p,g_j^q} & \text{otherwise} \end{cases}$$

Finally, SUBSTRA moves onto its inference phase where it initializes random clusters according to the optional priors, and then uses gibbs sampling to infer the latent variables. As seen in plaid, it iteratively infers each of the above latent variables while holding the other two fixed.

Model-based algorithms also lend well to triclustering. For example, [13] first finds biclusters in the object and feature dimensions. These are also represented using the Bernoulli distribution (alternatively, one can use a Gaussian distribution to represent real values). While finding these biclusters, they are simultaneously improved during Gibbs sampling based on how the Gaussian distributed biclusters in one slice correlate with those of another slice. This allows for triclusters that contain a collection of similar biclusters in the third dimension — much like what HDPGMM did for FCM. Meanwhile, [125] uses a three-level hierarchical DP optimized with MCMC such that the same heuristics hold for all dimensions. Therefore, though computationally expensive, model-based algorithms are powerful tools that can incorporate an array of parameters. It is also statistically reliable as long as assumptions on data distributions hold.

Though fundamentally, graphs simply represent distance or similarity matrices, representing data as graphs make it convenient to use already established definitions and algorithms based on the graph literature. For example, maximal cliques are analogous to maximal biclusters. Using this definition, triclustering algorithm [411], later improved by [173], first finds biclusters by taking a Pearson and Spearman similarity matrix based graph as input. It then applies multiple iterations of depth-first searches to find several candidate maximal clique biclusters for each slice. Finally, biclusters are merged between slices if they have a Pearson and Spearman similarity above a certain threshold. Another graph definition one can exploit is based on the graph structure. [133] represents 3D data (or 2D) as a tripartite (or bipartite) graph. Here, each index in each dimension represents a node which connects to all indices in the other dimensions. When it looks for connected components or densely connected nodes representing clusters in this graph, the tripartite graph definition guarantees that it will include elements from all dimensions.

More recently, evolutionary algorithms have taken off due to their ability to refine clusters within each iteration [80, 87, 26, 230, 255]. Based on evolutionary terminologies, it is

like iterative clustering, except each iteration would be analogous to a 'generation'. In each generation, it performs a series of 'selection', 'crossovers', 'mutations', and 'replacements' to build and refine clusters until a stopping criterion is met. As it is modelled after population biology, it is efficient and effective for exploring large data sets [283, 87, 230]. More specifically, the generic framework for evolutionary algorithms is as follows for biclustering — note that the same process simply needs to be repeated for a third dimension to adapt itself to triclustering.

1. Initialize 'populations' Q or biclusters randomly or via an existing bi/triclustering algorithm. For easy visualization, we call these populations.

2. For each iteration, conduct 'selection', 'crossovers', 'mutations', and 'replacements' until a stopping criterion is met, usually a user-specified maximum number of iterations. Afterwards, the final clusters are returned as the results.

    - Selection: this is where low-quality clusters are eliminated. A measure for quality here can be a thresholded heuristic objective function.
    - Crossover: from the remaining clusters, combine them and reproduce new ones. For example, [230] opts to merge two close clusters by their rows and columns. Then, for each feature in this new cluster, it selects which objects, if added to this cluster, will decrease the cluster's standard deviation. Starting from here, it re-evaluates all of its cluster assignments. If two objects are often added to the same clusters, they would be put together as a new cluster.
    - Mutation: these newly reproduced offspring clusters are mutated or modified. This can be done by deleting and adding random elements of D into any cluster. Another option would be to pick elements to add to biclusters if they improve a heuristic objective e.g. the correlation of all pairwise objects in a cluster.
    - Replacement: lastly, these new clusters are filtered yet again, such that low-quality clusters are deleted.

With its pros, evolutionary algorithms require that the user has a clear assumption on cluster quality, as its refinement procedure is highly heuristic.

Integrative clustering

Multidimensional clustering also fits well into situations when we want to incorporate other data into the clustering process, to mitigate Challenge (d). As we have seen before in Gibbs-plaid [127] and other forms of GRNMF (graph-regularized non-negative matrix factorization) [127], data integration may involve incorporating the gene ontology such that we can influence clustering via structural relationships between features. Other data sets that lend well to such regularization methods include drug-target/compound interaction

(to understand how gene expression may affect treatment), or even protein-gene interaction data (to see how downstream effects might change clustering results).

In the case that external data sets have a common dimension to that of the given RNAseq data, one can even incorporate these data sets into D as a third dimension and input the combined data set into a triclustering algorithm.

The authors of [127] combines these strategies by conducting matrix tri-factorization on two data sets ($(patient\ sample \times gene)$, $(drug \times target)$) regularized by a gene-target/protein interaction network to connect the two data sets. Trifactorization breaks up a matrix into three factors indicative of what clusters the samples, genes and target compounds belong to.

While simultaneously clustering everything by triclustering is ideal, this can be computationally expensive. As well, the data sets might be imbalanced in terms of size. For example, RNAseq usually has a large feature space compared to metabolomics where researchers may only test for a few metabolites. Therefore, first biclustering and then connecting biclusters using inter-slice metrics become a more common strategy. One can also settle with only clustering the object dimension over multiple feature sets by using multiple competing objective functions [41, 140, 221] or a single objective function made up of a linear combination of individual weighted objective functions i.e. a linear combination of different objective functions. Taking this further, another approach is to cluster the same objects individually in each data set and then merge the clusters via consensus clustering. Another strategy is to create a distance matrix from each data set and then merge those distance matrices as input into a clustering algorithm [276, 203]. For example, [84] first creates individual distance matrices via the heat kernel. For each distance matrix, it creates a KNN graph Laplacian. Each of these graphs can be embedded into a subspace of graphs U and then merged onto a Grassman manifold. The goal here is to obtain a merged graph on which one can obtain a Laplacian over to create a new balanced feature space for the original objects. This way, the effect of each data sets is balanced and we obtain a single input matrix that can be used with any clustering algorithm. Another simpler strategy would be to simply combine the first few eigenvectors of graph laplacians from the individual data set. [371] employs a different strategy for merging distance matrices. Also modelling these matrices as graphs, it uses the message passing algorithm to iteratively update each network with information from other networks such that each distance matrix slowly becomes more similar to each other. Finally, it conducts spectral clustering on the final distance matrix. In all cases, it is important to ensure that the influence of each data set on the clustering result is balanced in terms of the number of features and their values. Also, note that the same data set merging tactic can be applied to align networks [128] before they are used to regularize a clustering method [155].

Deep learning-based clustering

Deep learning-based "clustering" methods, like their imputation counterparts, are not unsupervised but self-supervised methods that are trained through self-learning. Again, these methods almost exclusively use the autoencoder architecture. For example, for scRNAseq data, scDeepCluster [356], DESC [215], and SAUCIE [15] uses an autoencoder with an objective function that uses KLD to ensure that the output, or the reconstructed input, has a similar distribution across samples. scDeepCluster assumes that its input data conforms to the ZINB distribution and so it uses regularization layers to fit its input on this distribution. In addition, SAUCIE also adds two regularization layers to ensure that the embedding contains near binary values and to minimize the Euclidean intra-cluster distance between points. Knowing their objective functions, these methods only work if the same cell populations exist across all samples, which is not necessarily always the case. However, these methods show that if this assumption holds, they can also handle the denoising, batch effect correction, and imputation in one network. scScope [82] and scGNN [376] also combines these steps together with cell population identification. However, scScope opts to use recurrent network layers in their autoencoder to account for dependency between genes. On the other hand, scGNN assumes that it is important to integrate the cell-cell interaction graph to account for systems biology signals such as cell type-specific regulation. It does so by using three neural networks. The first neural network takes as input the original data and outputs a cell-cell interaction graph. This graph is given to the second autoencoder network which, like SAUCIE, produces a cluster embedding. The reconstructed output is finally given as input into an autoencoder specific to its cluster. The output of these autoencoders is, again, given to the first neural network. This process is repeated until convergence and produces outputs: imputed data, cell-cell graph, and cell clusters.

Despite the creative application of deep learning, these methods are, again, very inefficient. To circumvent this issue, GOAEGONN [280], uses the gene ontology, or the known hierarchical relationship between genes, to reduce the number of connections between fully connected layers in its autoencoder (for dimensionality reduction) and neural network (for clustering). Nevertheless, even the fasted deep learning methods run for tens or hundreds of times longer than classical clustering algorithms such as Louvain. And while none of them outperform classical clustering on all cluster evaluation metrics, they each excel in specific metrics [374, 198] corroborating their immense potential.

3.7.5 Postprocessing

Postprocessing usually consists of checking the quality of clusters to see if any should be deleted. Unlike FCM, RNAseq usually has a small object dimension, therefore, there is not as big of a need to look out for rare clusters. Even if there is, many of the iterative refinement

processes are embedded into the clustering method itself e.g. iterative, evolutionary, and consensus clustering.

In model-based clustering, PACK uses BIC and kurtosis to identify which clustering result out of the multiple rounds of clustering it did, bringing about the best results. Meanwhile, SIBER opts to use the bimodality index (BI) [375] to evaluate pairwise clusters.

$$BI = \sqrt{\pi(1-\pi)}\frac{|\mu_1 - \mu_2|}{\sqrt{(1-\pi)\delta_1^2 + \pi\delta_2^2}}$$

where it compares the mean and variance of the two clusters as marked in the subscript. BI became an ideal choice for its ability to produce a gene ranking — in that, if each feature is tested individually, one can determine which genes separate the two clusters the best. Though BI makes it easier to interpret the results, it can still be influenced by low confidence genes.

One can also opt to directly use their assumption or heuristics to see if a cluster is appropriate. Using statistical distribution tests is applicable for testing constant, additive, and in some cases, multiplicative clusters, and the results can be numeric or visualized via a QQplot. For example, one can test whether the variance is reduced between the whole data set versus the individual variance of the found clusters [149]. Using a more statistical approach, [257, 350] evaluates constant clusters by comparing the values in the cluster to a null Gaussian distribution. Assuming that a unique cluster is different from the rest of D, it can be deemed significant if it deviates from that null distribution. To conduct a valid statistical test, they make sure that the number of objects in a cluster is larger than a threshold $|I|$ before testing the cluster Q: $P(Q \sim Binomial(1/n!, |H|)) \leq p) < \alpha/n!$ where α is a threshold. Also using a statistical test, [243] ensures that its null distribution is robust by basing it on multiple resampling of D.

Meanwhile, statistics like TRIQ [138] have also emerged to evaluate RNAseq-specific triclustering results. TRIQ uses a weighted sum of four terms that reflect on 1) how deep and coregulated each feature in a cluster is according to the gene ontology (see the table of values in [138]), and 2,3,4) correlation between values with the multislope measure (MSL) [139], Pearson, and Spearman correlation metrics. Since these test for scale and shift-invariant correlation between all dimensions, they can evaluate constant, order-preserving, and some additive and multiplicative triclusters. TRIQ's strength lies in the fact that it brings domain knowledge into the picture. This way, it can ask the question of whether the clusters make sense in terms of the relationship between genes. Other ways of utilizing domain knowledge are by enriching the genes from each cluster to see if they have any common downstream effects. One can even test to see if similar results occur in other comparable species for cross-species consistency [175]. However, one needs to take caution when using such metrics, because methods that integrate the same domain knowledge while clustering would naturally yield good results. As well, external data sets are usually compiled through a community

effort. While they include information from lots of experiments, they tend to have a positive bias — in that, a lack of an interaction or a negative result is usually not reported for publication and therefore not easily confirmed.

3.7.6 Interpretation

On top of a numeric statistic for evaluating clusters and their DE, visualization is also key to understanding what the clustering results mean. While one can use classic methods such as PCA, t-SNE, and Umap [249], customized methods for RNAseq have also popped into the scene. Aimed at maximizing inter-cluster distances, [372] attempts to visualize the distance between cells by learning the weights in a linear combination of an optional user-defined set of kernel distance metrics (Gaussian by default).

Other than visualization, it is also important to annotate the results. For example, using multidimensional clustering and integrated data sets can help verify or even discover new interactions between genes/proteins. A common assumption is that if a gene/protein strongly interacts with a group of genes/proteins that affect the same disease, this gene in question may also affect that disease in some way. [270, 315, 264] are a few methods that take the clustered genes and query them on these interaction networks to graph out and find such linkages.

3.7.7 Alternative solutions

In contrast to FCM, clustering is the solution one would look to when grouping RNAseq objects since it is difficult for a person to manually analyze such a high-dimensional data set. On the other hand, if one does have a prior for how the objects or features can relate to each other, integrating these data types, with methods such as Gibbs-plaid, can be seen as a semi-supervised method and therefore often called an alternative to purely unsupervised clustering methods.

On the other end of the spectrum, it is noteworthy to mention that supervised classification methods are used commonly in patient stratification or drug-target interaction. For example, when trying to re-purpose drugs, one might learn what patient group a drug is effective for already. Then, when given a new patient, one can evaluate whether this patient is similar enough to the learned patients per their transcriptomic profile. If they are similar, researchers can hypothesize that the drug may affect the new patient in a similar way [266, 379, 409, 98]. Nevertheless, the same problem can also be solved using clustering-based methods. While researchers have done many experiments to confirm whether drugs affect a multitude of targets, compounds, or patient groups, there are still experiments that have yet to be done. Hence, drug-target interaction data sets are very sparse. To fill these gaps, [413] uses MSCMF (multiple similarities collaborative matrix factorization). This is a semi-supervised approach where it regularizes the drug dimensions with a drug-drug interaction matrix and then factorizes the drug-target similarity matrix using alternating least squares.

It then multiplies these factors back together to create an ideally dense matrix to fill in those gaps. In other words, it finds the clusters each drug and target belong to and then assumes that those in the same cluster should be similar. If any point lacks a value, it fills in those values based on those of its neighbours.

Remarks

Clustering has become one of the most popular methods to analyze RNAseq data. Although many methods started out as being designed for microarray experiments, they have been successfully applied to RNAseq. However, RNAseq data presents new problems such as sparsity. With imputation and data integration, researchers have started to try to mitigate this issue, but it is still one of the largest bottlenecks in information recovery from RNAseq [390]. Another prospect for these methods is how they can be further customized for application to similar data from other sequencing technologies such as ChIPseq and DNAseq, along with the third-generation long-read sequencing technologies PACBIO and Oxford Nanopore. Though scRNAseq took a similar path as RNAseq in terms of clustering, it also presents an opportunity, much like in FCM, on how different samples of cells can be compared with each other post cell population identification. While the samples in scRNAseq can be incorporated as a third dimension, so can many other aspects of an RNAseq experiment. While the most popular additional dimension is time, there are opportunities for further methodology refinement in incorporating spatial data such as RNA and protein structure [258, 272, 157, 232, 225].

Although methodology development is important, another aspect to them is how much impact they are having. With the increase in the number of technologies that can utilize clustering methods, the ratio of application studies to novel methodology papers has increased from 1/9 to 2/3 — with patient stratification and drug re-purposing or drug-target interaction applications in the forefront of this need [390]. Many of these papers start by using classic dimensionality reduction and clustering methods to reach their goals. Fortunately, these methods are still very effective. In fact, using Kmeans on t-SNE processed scRNAseq data has been shown to yield better results than many more sophisticated methods [90]. However, more work is required in making the customized RNAseq clustering methods more accessible to the general user.

Chapter 4

Identifying differential cell populations in flow cytometry data accounting for marker frequency

The contents of this chapter address Problem 2 and have is published in [400]. The method described in this chapter is freely available on Github (https://github.com/aya49/flowGraph) and Bioconductor (https://bioconductor.org/packages/flowGraph).

4.1 Introduction

The last goal (Problem 2) in the FCM analysis pipeline (but first chronologically in this thesis) is the identification of biomarker candidates. One group of candidates is the *differential cell populations (DCPs)*. These are cell populations whose proportional abundances (i.e., the relative quantity of cells in a cell population) differ significantly between samples of different classes (e.g. disease vs healthy). Commonly used metrics for proportional abundance are cells per μL of blood and proportion (i.e. the ratio between the count of cells in a population and some parent population).

We propose the concept of *maximal differential cell populations* (MDCPs). MDCPs are DCPs whose change in proportional abundance is only significantly associated with its sample class and not a result of proportional abundance change in a related DCP. For example, if there is a significant decrease in the proportion of helper T-cells in samples from diseased individuals, then helper T-cells is a DCP. However, if the proportion of all types of T-cells decreases at a similar rate, then we can hypothesize that the disease reduces the proportion of all T-cells. It follows that T-cells and all of its child populations, including helper T-cells, are DCPs but only T-cells is an MDCP. MDCPs are preferable candidate biomarkers because their proportion change is only driven by their association with a sample class. We refer to such cell populations as driver cell populations. To our knowledge, many methods find biomarker candidates by identifying DCPs, but there are no methods that do so by isolating the MDCPs among those DCPs.

Current methods identify DCPs either as a byproduct of another procedure [27, 67, 404, 359, 50] (e.g., CytoDX [159] main goal is to classify FCM samples, but it also tries to find DCPs as a postprocessing step) or compare a limited amount of known prespecified cell populations identified in human-created gating strategies by evaluating whether there is a significant difference in their proportional abundance across samples using some statistical significance test [368, 268, 216]. A summary of related methods can be found in [75, 6]. Though there are methods that attempt to find MDCPs by expanding their DCP candidates to cell populations that are dependent on each other, the statistical tests they use assume independence between cell populations. For example, Cydar [226] uses the spatial false discovery rate, and diffcyt [382] applies statistical tests traditionally used for differential analysis in bulk RNAseq data sets where the transcripts/genes are assumed to be independent of each other. Such statistical tests would only find DCPs despite the large MDCP candidate pool because they do not account for the relationships between cell populations.

To address these shortcomings, we describe a method that identifies MDCPs by comparing the SpecEnr of all possible cell populations across samples modelled on a cell hierarchy taking into account all possible relationship between cell populations. Specifically:

1. To enable analyzing all possible cell populations and their relationship with each other, we use SpecEnr, a novel cell population score (a numerical metric) derived from the proportional abundance metric, proportion.

    (a) Our method considers all possible previously known and new cell populations in each FCM sample as candidates for MDCPs. In contrast:

    - Identifying DCPs manually via a gating strategy means that our candidates would be limited to only those cell populations that are already known to human experts.
    - Identifying DCPs from cell populations found through single-cell clustering (Chapter 3) means that we would only analyze cell populations from a single layer in the cell hierarchy that do not overlap with each other (i.e. all cells are assigned only one cell population label).

    (b) Analyzing all possible cell populations also implies that we consider all possible relationships between cell populations. Each cell population may have several parent and child cell populations. We show the significance and importance of taking these factors into account in our results at the end of this chapter and from the extensive experiments we performed in [398]. In contrast:

    - Cell populations found through manual gating only have zero to one parent cell population greatly limiting the number of relationships tested for finding MDCPs.

- Cell populations found through single-cell clustering have no such parent and child relationships. In this case, one would use traditional statistical tests that assume independence between cell populations. However, this assumption does not allow us to test whether a cell population is a MDCP given its ancestors because they are not identified.

2. Finally, modelling cell populations and their relationship on a cell hierarchy allows us to semantically label cell populations with marker conditions and visualize MDCPs to aid the interpretation of results.

Therefore, we hypothesize that identifying MDCPs will aid the understanding of disease etiology.

In this chapter, we:

1. Define and formulate the problem of finding driver cell populations by identifying MDCPs.

2. Introduce a cell population score SpecEnr (specific enrichment) that accounts for dependencies between parent and child cell populations.

3. Describe a method that harnesses SpecEnr properties to find robust, accurate, and easily interpretable driver cell populations.

4.2 Methods

4.2.1 Preprocessing and cell population identification

To calculate SpecEnr, we can take as input, a vector of cell population proportions for each FCM sample generated using any suitable manual or automated approach. In our experiments, given a preprocessed FCM sample, we identify the cell populations in this sample via flowDensity [240]. When completed, this step outputs L marker thresholds, one for each marker. These thresholds may differ slightly between samples. After obtaining the thresholds, we enumerate all possible cell populations using flowType [268]. flowType takes as input, the L thresholds and the preprocessed FCM sample's $R \times L$ matrix, and enumerates all possible cell populations and their cell count into a length $m = 3^L$ vector. Next, we normalize cell counts relative to the total cell count in each sample. We do so by converting counts into proportions by taking the cell count of each cell population over the total number of cells in the sample.

Users can also choose to identify cell populations via methods other than flowType. Given an FCM sample's cell hierarchy (see Section 2.3), the requirement for calculating the SpecEnr of a cell population in this cell hierarchy is that its and all of its parents' and grandparents' proportions should be available. For example, if the we choose to identify cell populations via clustering, we can treat each cluster as a unique gate; this way, a cell

population's parent cell populations would include all possible pairwise combinations of it and all other cell populations. Its grandparent cell populations would be all possible mergers of it and any two other cell populations.

4.2.2 Notations

To visualize the relationship between cell populations, we use the cell population hierarchy of a sample as described in Section 2.3.

We denote the actual proportion P of any node $v^{1:\ell}$ in layer ℓ by $P(v^{1:\ell})$ such that $1{:}\ell$ $(1, 2, ..., \ell)$ are the indices of the marker conditions its label contains.

We show in Section A.2 that we can derive our method's scores for all cell populations just from those cell populations whose labels only contain positive conditions. Following this reasoning, and to simplify our notation, we assume that the marker conditions used to label our cell populations are all positive. This implies that the markers used must be unique, as they always should be. For example, cell population $A^+B^+C^+$ has three positive marker conditions and can therefore be denoted as $v^{1:3}$; subsequently, we can denote its parents A^+C^+ and A^+B^+ as $v^{\{1:3\}\setminus 2}$ and $v^{\{1:3\}\setminus 3}$ by excluding the second and third marker conditions.

4.2.3 Cell population score: SpecEnr

The assumptions we will introduce for SpecEnr are based on well-established concepts in probability theory [115]. marker conditions are random events and the proportion of each cell population is the probability of jointly occurring random events.

To obtain SpecEnr, we compare the actual proportion of a cell population with its expected proportion: the proportion we expect a cell population to have given the proportion of its ancestors. By doing so, we can evaluate its proportion changes independent of the effects incurred by its ancestors.

Expected proportion

The SpenEnr null hypothesis imagines that each cell population has at least two marker conditions that are independent given the others. Specifically, under the null hypothesis, for a cell population $v^{1:\ell}$ with proportion $P(v^{1:\ell})$, the following holds. Without loss of generality, let us assume that $P(v^1)$ (e.g. A^+) and $P(v^2)$ (e.g. B^+) are independent given $P(v^{3:\ell})$ (e.g. C^+).

$$P(v^1|v^{2:\ell}) = P(v^1|v^{3:\ell}) \tag{4.1}$$

$$P(v^{1:\ell}) = P(v^{2:\ell})\frac{P(v^1, v^{3:\ell})}{P(v^{3:\ell})} \tag{4.2}$$

where $P(v^1|v^{3:\ell})$ indicates the conditional proportion of v^1 given $v^{3:\ell}$.

Generalizing this assumption to any p, q pair, $p \in 1{:}\ell$ and $q \in 1{:}\ell \setminus p$, we get

$$P(v^{1:\ell}) = P(v^{1:\ell\setminus p})\frac{P(v^{1:\ell\setminus q})}{P(v^{1:\ell\setminus\{p,q\}})} \tag{4.3}$$

While this assumption may be applied to most cell populations, there are edge cases. Our assumption requires $P(v^{1:\ell\setminus\{p,q\}})$ to exist. Therefore, expected proportion is only calculated for cell populations in layers $\ell \geq 2$. For the root node, we initialize its expected proportion to 1. For the nodes in layer one, we initialize their expected proportions to 0.5. By initializing their expected proportion to 0.5, we maintain the sum-to-1 rule in probability where, for example, $P(A^+) + P(A^-) = 1$.

To identify differential cell populations, we compare their expected and actual proportion. In Equation 4.3, we assumed all marker condition pairs, with indices $\{q, p\}$, $P(v^p)$ and $P(v^q)$ to be independent give $P(v^{1:\ell\setminus\{p,q\}})$. Now let us assume that this does not hold for $A^+B^+C^+$'s parent cell population A^+C^+. While A^+ and C^+ are dependent on each other, B^+ is independent of both A^+ and C^+. In this case, the assumption we made in Equation 4.2 only holds for cell population $A^+B^+C^+$ when $q \in \{1, 2\}$ and $p = 3$. We do not want to flag $A^+B^+C^+$ as maximally differential as its proportion change is completely dependent on cell populations A^+C^+ and B^+. Therefore, we relax our assumption in Equation 4.3 to: there must be some index pair $\{p, q\}$ such that $P(v^p)$ is independent of $P(v^q)$ given $P(v^{1:\ell\setminus\{p,q\}})$. Then $P(v^{1:\ell})$ can be calculated as follows.

$$P(v^{1:\ell}) = P(v^{1:\ell\setminus p})\frac{P(v^{1:\ell\setminus q})}{P(v^{1:\ell\setminus\{p,q\}})} \tag{4.4}$$

$$p = \arg\max_{p\in 1:\ell} P(v^{1:\ell\setminus p})$$

$$q = \arg\min_{q\in 1:\ell\setminus p} \frac{P(v^{1:\ell\setminus q})}{P(v^{1:\ell\setminus\{p,q\}})}$$

Otherwise, if there is no p, q pair such that $P(v^p)$ is independent of $P(v^q)$, then Equation 4.4 does not hold and $P(v)$'s abundance change cannot be attributed to any of its ancestors' abundance change.

Additional details on proof of correctness for our assumption are in Section A.1.

SpecEnr

In this section, we explain how we calculate our proposed SpecEnr score. Given the expected proportion of cell population v calculated using Equation 4.4, SpecEnr is the natural log of v's actual proportion over its expected proportion calculated using Equation 4.4 which we denote here as $E(v)$.

$$SpecEnr(v) = \ln \frac{P(v)}{E(v)} \tag{4.5}$$

$$p = \arg\max_{p \in 1:\ell} P(v^{1:\ell \setminus p})$$

$$q = \arg\min_{q \in 1:\ell \setminus p} \frac{P(v^{1:\ell \setminus q})}{P(v^{1:\ell \setminus \{p,q\}})}$$

SpecEnr accounts for the dependency of a cell population on its ancestors. For example, if a cell population has a SpecEnr value of 0, then its proportional abundance is completely dependent on that of its ancestors. Otherwise, it contains marker conditions that are all dependent on each other, where $P(v^p)$ is dependent on $P(v^q)$ for all $\{p, q\} \in 1{:}\ell$ (i.e. Equation 4.3 does not hold for any p, q).

The asymptotic runtime and actual runtime to calculate SpecEnr are provided in the Section A.2 and A.7.

Maximal differential cell population (MDCP)

A maximal differential cell population (MDCP) is a cell population that has significantly different abundance across sample groups. In this respect, an MDCP is similar to a differential cell population (DCP). However, in addition to this, MDCP has an additional property where its abundance difference cannot solely be attributed to the abundance difference in its ancestor cell populations. Therefore, an MDCP's abundance change is unique and can help users confirm or reject hypotheses in biological experiments.

SpecEnr is an example of log-probability ratios, which are commonly used in Bayesian hypothesis testing [38]. Following a similar framework, a cell population is not an MDCP if its SpecEnr values across classes are not significantly different. Conversely, in order for a cell population to be an MDCP $v^{1:\ell}$, it must satisfy two conditions.

1. A MDCP SpecEnr must be significantly different between samples according to a filtered adjusted T-test we describe in the next section.

2. A MDCP must be maximal, in that it must not have any direct descendants who meet the first condition above.

The second condition is required because our first is also satisfied by direct ancestors of an MDCP as its ancestor cell populations are defined by a subset of marker conditions defining the MDCP.

Relating our definition back to the difference between MDCP and DCP: if there exists one MDCP in our data set (e.g. A^+B^+), then the DCPs would be the MDCP's ancestors (e.g. A^+ and B^+) and descendants (e.g. $A^+B^+C^+$ and $A^+B^+C^-$), and all cell populations that share at least one marker condition with the MDCP (e.g. B^+C^+ and A^+C^-). This

further demonstrates the difficulty of identifying MDCPs among DCPs; as all DCPs would be a candidate MDCP.

4.2.4 Testing whether the cell population SpecEnr values across sample classes are significantly different

To test if a cell population satisfies our first condition for MDCPs (i.e. its SpecEnr is significantly different across samples), we apply the T-test on SpecEnr values for each cell population across two sets of samples (e.g. a control group and an experiment group). We show that the raw SpecEnr T-test p-values are statistically sound in Section A.5

Given that we are testing multiple hypotheses, we adjust the p-values ρ_v for each cell population v using layer-stratified Bonferroni correction [44] to obtain our final adjusted p-values ρ'_v. We do so by multiplying our p-values with the number of cell populations in the layer on which cell population v resides m_ℓ and the total number of layers $L+1$ (including the layer 0; see Section A.3 for additional details). We use a q-value (i.e., the adjusted p-value) threshold $< .05$ to determine if a cell population q-value is significant and potentially maximally differential.

Avoiding falsely significant q-values with filters

In some cases, the p-value obtained by evaluating SpecEnr may be falsely significant when dealing with small or noisy data sets. As a cell population's proportion gets close to 0, the actual versus expected proportion ratio used to calculate SpecEnr becomes inflated. As well, if we are conducting significance tests on cell populations with SpecEnr values of 0 (i.e. actual and expected proportions are the same) model-based significance tests (e.g. T-test) are highly influenced by outliers and rank-based significant tests (e.g. Wilcoxon) are influenced by the random ordering of 0's. To ensure our SpecEnr p-values are valid, we mark cell populations as insignificant if any of the following apply.

1. They do not have a mean count of a user-specified threshold of events (we use > 50 for our data sets) to prevent inflated ratios,
2. They do not have significantly different actual versus expected proportions for at least one of the sample classes, and
3. They have actual and expected proportions that are significantly different across both sample classes.

In our experiments, we use a standard significance threshold of $< .05$ for all T-test p-values on filter-related significance tests. We show an example of these filters in the Section A.4.

For brevity, we call the p- and q-values obtained using SpecEnr and proportion, SpecEnr p- and q-values, and proportion p- and q-values respectively.

4.2.5 Experiment

To confirm that flowGraph is able to identify known MDCPs we prepared synthetic negative and positive control data sets and used two previously published biological data sets.

Synthetic data

- **neg1** (Negative control): For each cell, we assigned it to be positive$^+$ for each marker with a 50% probability.
  - Samples: 10 control vs 10 experiment (300,000 cells/sample).
  - markers: A, B, C, and D.
- **pos1** (Positive control 1): Same as neg1, except in the experiment samples, cell population A^+ is increased by 50%. More specifically, in each $R \times L$ matrix, we duplicated a random sample of half the cells in A^+.
- **pos2** (Positive control 2): Same as pos1, except instead of A^+, $A^+B^+C^+$ is increased by 50%.
- **pos3** (Positive control 3): Same as pos1, except instead of A^+, a random sample of half of all cells that belong to at least one of A^+B^+ and D^+ are duplicated (i.e. increased by 50%), indirectly causing a unique increase in cell population $A^+B^+D^+$. Note that cells that belong to both A^+B^+ and D^+ are duplicated once instead of twice to ensure both cell populations increase by 50%.

Real data sets

- **flowcap** (FlowCAP-II AML data set): This data set is from the FlowCAP-II [6], AML challenge, panel 6. It is known that AML samples have a larger $CD34^+$ population [6].
  - Samples: 316 healthy vs 43 AML positive subject's blood or bone marrow tissue samples (60,000 cells/sample).
  - markers: *HLA-DR*, *CD117*, *CD45*, *CD34*, and *CD38*.
- **pregnancy** (Immune clock of pregnancy data set): While the previous two data sets are flow cytometry data sets, this is a CyTOF data set. Nevertheless, our method can be used on either type of data set. So far, there have been no experiments that identified ground truth driver cell populations for the pregnancy data set [7]. However, the original authors were able to train classifiers on the same patients using FCM and multi-omics data [281]. Therefore, we hypothesize that we will be able to find MDCPs in this data set that are associated with the sample classes listed below.

- Samples: 28 late-term pregnancy vs 28 6-weeks postpartum human maternal whole-blood samples (approximately 300,000 cells/sample); Samples are taken from each of the 18 and 10 women of the training and validation cohort during late-term pregnancy and 6 weeks postpartum.
- markers: *CD123*, *CD14*, *CD16*, *CD3*, *CD4*, *CD45*, *CD45RA*, *CD56*, *CD66*, *CD7*, *CD8*, *Tbet*, and *TCRgd*.
- To account for possible batch effects associated with the subjects who provided the FCM samples, we used the paired T-test where samples were paired with respect to subject.

4.3 Results

SpecEnr p-values are robust. We hypothesized that theoretically similar data sets yield similar unadjusted p-values across all cell populations. To test this, we split up the samples in data set pos1 in half and compare the samples across these two halves or "theoretically similar data sets". When we compared the unadjusted SpecEnr p-values across these theoretically similar data sets using the Spearman correlation, we obtained a perfect score of 1. We saw the same result with metrics recall, precision, and F measure over the first set. These results indicate that significant cell populations in the first set also show up as significant in the second set. We also show that unadjusted SpecEnr p-values are statistically sound with the following experimental results [380]. Using SpecEnr, we were able to generate a random uniform distribution of unadjusted p-values on our negative control data set neg1. It follows that 5% of the SpecEnr p-values were below our .05 threshold (See Section A.6 for added detail).

SpecEnr q-values help identify accurate driver cell populations in synthetic data sets. flowGraph accurately identified that pos1 and pos2 driver cell populations were A^+ and $A^+B^+C^+$ (Figure 4.1). While both SpecEnr and proportion q-values flagged these cell populations, when we observed SpecEnr q-values, the descendants of these driver cell populations were not flagged as significant. This was also true when multiple driver cell populations were present in lower layers of the cell hierarchy. In pos3, where both A^+B^+ and D^+ were increased to cause a unique change in $A^+B^+D^+$; we saw that SpecEnr q-values were only significant for these three cell populations and their ancestors. Results from our positive control data sets were also similar when the same cell populations decreased instead of increased in proportional abundance (Section A.6).

SpecEnr q-values flag known and novel driver cell populations in real data sets. For the flowcap data set, SpecEnr directs users down a branch of the cell hierarchy from physical properties SS^+ and FS^- to $FS^-SS^+CD117^+45^+$ and $HLA^+CD117^-CD45^+CD34^+$.

While *HLA* and *CD117* are variably expressed on cells in FCM samples from subjects with AML [385, 282], *CD34* and *CD45* are expressed on blast cells [289, 144]. This is important as the abundance of blast cells aid in diagnosis of AML [6].

In the pregnancy data set, the top most significant cell populations displayed by our statistical significance test showed an up-regulation in cell populations containing *CD3*, *CD45*, and *CD45RA* (e.g. $CD3^{+}CD45RA^{+}CD56^{-}Tbet^{-}$). SpecEnr q-values also indicate that cell populations containing markers *CD8* and *CD16* are significantly down-regulated. Meanwhile, proportion q-values flag all DCPs in the cell hierarchy as significant.

4.4 Discussion

In this chapter, we introduced a new cell population score, SpecEnr, and a method, flowGraph, that integrates SpecEnr to identify MDCPs. We showed that the results of flowGraph are statistically sound, accurate, and easily interpretable via the cell hierarchy.

In the FlowCAP-II challenge, the AML data set was used to evaluate how well methods are able to classify samples belonging to healthy and AML-positive subjects. Among the competing methods, those that used cell population proportions for classification were DREAM–D, flowPeakssvm, Kmeanssvm, flowType, FeaLect, PBSC, BCB/SPADE, SWIFT [6]. All of these methods assume that cell count and proportion may be used to differentiate between the two classes of samples. However, cell count and proportion do not account for relations between cell populations, making it difficult to isolate the MDCPs among the DCPs (Figure 4.1). To account for these relationships, one can manually analyze the ratio of the count of cells in a population over all of its direct parent populations. However, given L markers, there are $3^L \cdot \frac{2L}{3}$ such relationships not including the relationship between a cell population and its indirect ancestors [268]. In contrast to comparing 3^L cell population scores, directly comparing cell population relations becomes computationally impractical. SpecEnr mitigates both problems as it is a cell population score that accounts for relations between cell populations. q-values obtained from computed SpecEnr scores isolated only the few ground truth driver cell populations (MDCP e.g. $SS^{-}CD34^{+}$). Our results not only reveal known driver cell population $CD34^{+}$ but also provide visualizations signifying that their change was caused by a change in its descendants exposing novel driver cell populations.

We also observed this contrast in behaviour between SpecEnr and proportion q-values in the pregnancy data set. We hypothesized that flowGraph would be able to find MDCPs because [7] were able to use L_1, L_2, and cell signal pathway regularized regression to classify samples taken from women at different stages of pregnancy. The original authors also assumed that there exist MDCPs in the pregnancy data set [159]. However, because these methods find candidate biomarkers as a byproduct of a sample classification method, there was no way of verifying whether the candidate biomarkers they inferred are simply

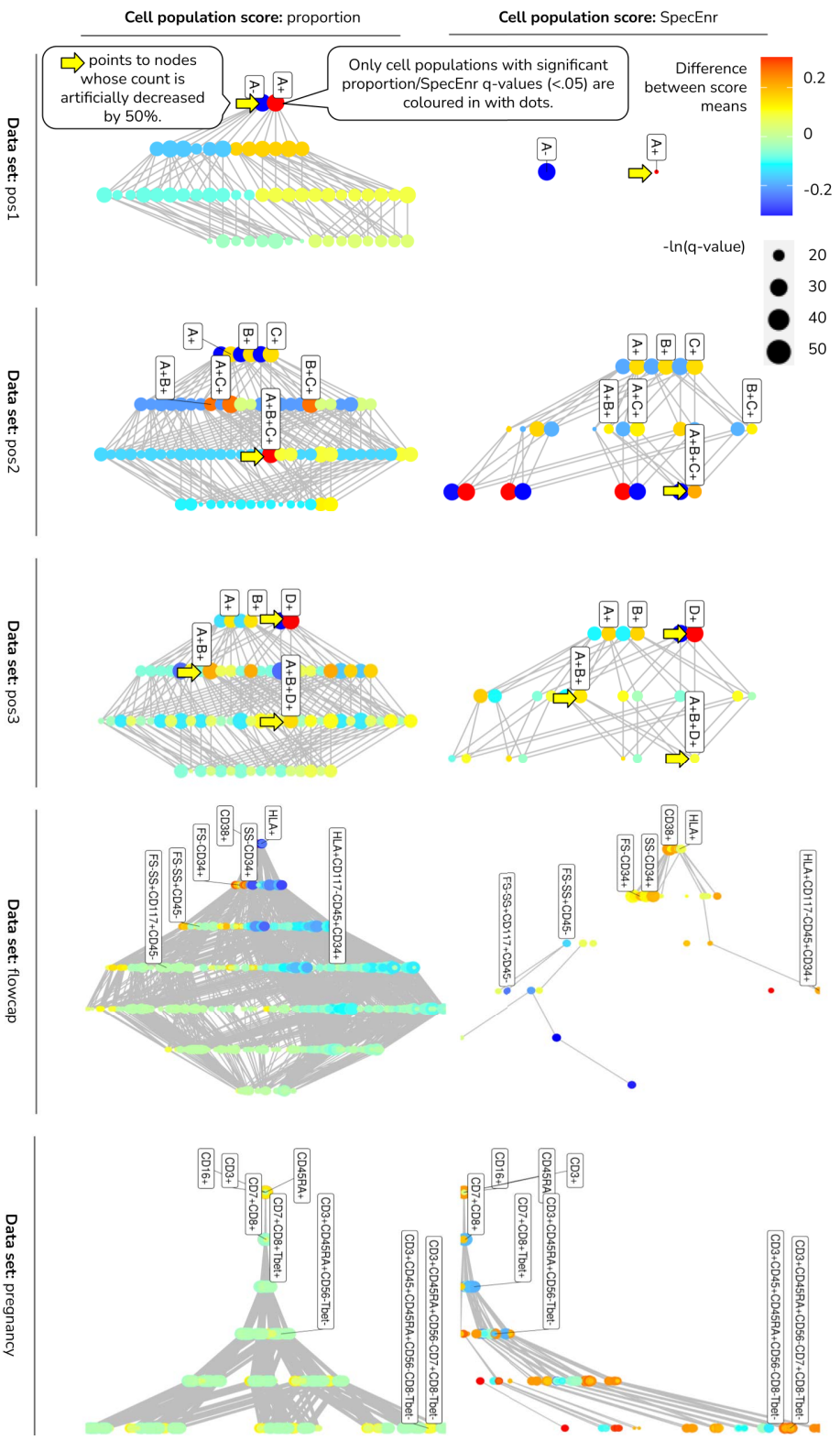

Figure 4.1: Cell hierarchy plots for synthetic data sets pos1-3 and real data sets flowcap and pregnancy show that SpecEnr q-values accurately identify MDCPs while proportion q-values flag all DCPs but do not highlight which of the DCPs are MDCPs.

DCPs or are also MDCPs. FlowGraph answers this question by providing users with a way to differentiate between the two while verifying our hypothesis validating the existence of MDCPs in the pregnancy data set.

Since SpecEnr is calculated using proportions, it is prone to the same issue that occurs when using proportions directly. That is, changes in the proportion of cell populations must sum to 0. For example, in pos1, A^+ abundance doubled, so its proportion increased from .5 to .66; but A^- proportion decreased from .5 to .33. More generally, if a cell population is differential, it will induce a change in the proportion of all cell populations that are labelled using the same set of markers as it; because these cell populations are mutually exclusive. Another example of this are the $\{A^{\{+,-\}}B^{\{+,-\}}C^{\{+,-\}}\}$ cell populations from pos2. If the driver cell population resides in layers > 1, then it is easily identifiable as the cell population with the largest magnitude of change. In the future, we would like to improve on our method such that we only flag the driver cell populations and not the cell populations it affects in the context of proportions.

Finally, we showed that an adjusted and filtered T-test on SpecEnr will yield a significant q-value on driver cell populations and their ancestors. While this makes driver cell populations intuitive to find on a cell hierarchy plot, ideally, we should only flag the driver cell populations as significant and not their ancestors. By preventing excessive flagging of ancestor populations, we enable more expressive and detailed insights into results interpretation.

Chapter 5

Automated 2D gating via motif matching for cell population identification

5.1 Introduction

This chapter aims to identify cell populations in FCM samples in a supervised manner by projecting expert created gates onto ungated samples. We refer readers to Section 2.2 for a primer on manual gating and related work, and Chapter 3 for cell population identification methods via clustering. Provided that current un/supervised cell population identification methods produce results that lack interpretability, we are motivated to explore the following hypothesis: a supervised method that uses 2D visual scatterplot features a human uses while gating should produce accurate and more interpretable results than a method that only uses FI values. Our hypothesis extends that of previous methods [227, 171] which automatically gates in 1D.

To verify this hypothesis, in this chapter we:

1. Describe and extract visual features from each scatterplot.
2. Implement our first method for cell population identification. This method harnesses the above visual features to learn a 2D gate from at least one gated scatterplot and project this gate onto a set of ungated scatterplots.
3. Present results that directly motivate the development of our second method for cell population identification in the next Chapter 6.

Since our method mimics manual gating, our method inherits the two interpretability traits we defined in Section 2.2: 1) we project human-created gates along with their cell population labels and 2) our gates are projected on 2D scatterplots whose dimensions are specified by the user and so these plots can be easily analyzed by the user.

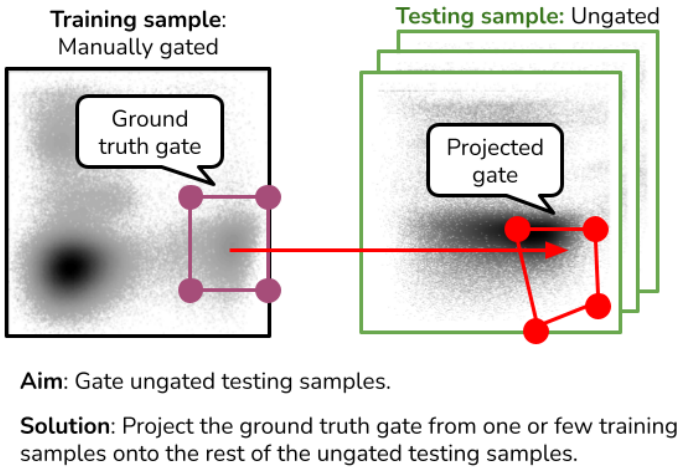

Figure 5.1: The aim of our method is to gate ungated testing FCM samples by projecting the ground truth gate from one or few training samples.

We do not perform an exhaustive evaluation of the method in this chapter in favour of the method in Chapter 6, which also uses visual features. We expect the latter to perform better because instead of manually assigning parameters for extracting visual features, it uses a deep neural network that learns and refines these parameters based on existing data sets.

5.2 Method

To verify whether our visual features can help to gate FCM samples, we describe a method that uses these features to project 2D polygon gates from one or few gated 'training' samples onto ungated 'testing' samples. All of these samples plot cells from the same cell population on scatterplots with the same two markers as the two dimensions. We say these samples are plotted on the same 'scatterplot' for the same 'unique marker pair' for brevity. In this chapter, we assume there is one gate per scatterplot. However, if there are multiple gates on the same scatterplot, we can repeat the procedure in our method for each gate. We also use the terms 'learn' and 'train' as more generic terms to describe the process of projecting gates.

5.2.1 Feature engineering

For all all sample (training and testing), we extract their numeric visual features that will be used in later stages of our method. The signals exposed by these features should correspond with what a human observes from a scatterplot to guide manual gating. Given a preprocessed FCM sample, we extract its scatterplot for a unique marker pair as a scatterplot image. This image is a 400×400 matrix where each of its values represent an image pixel. A scatterplot image's 'image-based' visual features are a set of 400×400 matrices. We say a visual feature

type is 'local' if each value in its matrix summarizes not only a pixel but a voxel of the image. A voxel is the set of pixels surrounding a pixel of interest. For example, if our voxel size is 3, the value at coordinate $(10, 10)$ represents the set of 3×3 pixels centered around coordinate $(10, 10)$, its 'center pixel'. In other words, this voxel contains pixels with coordinates $(9, 9), (9, 10), (9, 11), (10, 9), (10, 10), (10, 11), (11, 9), (11, 10), (11, 11)$ and its center pixel has the coordinate $(10, 10)$. This voxel is not to be confused with the acronym 'voxels' used in medical imaging. In our experiment, the values for each feature type are normalized to the range $[0, 1]$. When we conduct cell population identification on a scatterplot image, we assign each pixel of this image, a cell population label. The label on each pixel is also assigned to all the cells that are plotted on that pixel. The final F1 score we calculate to determine the accuracy of our method is calculated based on the number of cells we labelled correctly.

To summarize, for each FCM sample's 400×400 scatterplot image, we obtain the following additional visual feature matrices of the same resolution that we will be using downstream in our method:

- Image-based visual features:
  - Original density scatterplot image
  - Binned density contour image
- Local visual features:
  - Edge detection filtered image (as calculated by the Scharr filter [88])
  - Edge detection filtered image (as calculated by the Sobel filter [190])
  - Orientation filtered image (as calculated by HOG (histogram of oriented gradients) [113])
  - Orientation filtered image (as calculated by Gabor [252])

Image-based visual features

We take as input, a scatterplot image which is also our first image-based visual feature. The feature or the numeric value assigned to each pixel in this image is the Gaussian kernel 2D density estimation of cells on each pixel. These are numbered from 1 to 0 based on high to low density. Finally, we tighten the axes such that there is no arbitrary white space on the edges of our scatterplot image. To do this, we set the borders at the maximum second-order derivative of the kernel density estimate curve on each axis. The resulting image is also the image that humans use to perform gating.

Our second image-based visual feature takes the scatterplot image as input and is binned, then smoothed. We use the bivariate normal density kernel, and plot these as density contour images. The reason we include the density contours as an input to our method is because we

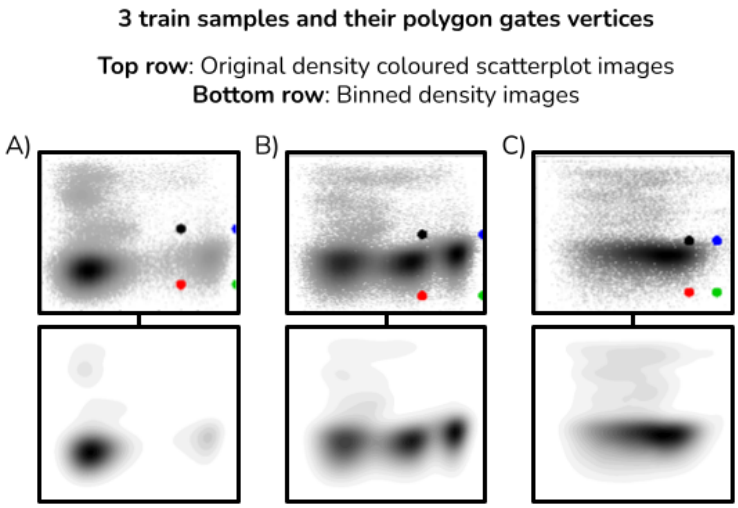

Figure 5.2: A), B), and C) are the images of three training samples. The top row consists of their original density scatterplot image with the vertices of their ground truth gate and the bottom row consists of their binned density contour image.

deduce that these contours guide the users when they manually draw polygon and elliptical gates.

Local visual features

After we obtain our images, we extract additional features that amplify signals of edges and edge orientation on the image. An 'edge' in image processing is a section of the image where there is an abrupt change or, more formally, a discontinuity in the image brightness. First, we reduce noise by applying a Gaussian filter to blur our scatterplot image. In our experiment, we chose the commonly used filter voxel sizes 10, 25, and 40 pixels. This blurred image and the binned density contour image are taken as input to create the features described in the rest of this section. A toy example of these are shown in Figure 5.3.

To detect edges, we apply filters Scharr and Sobel [88, 190]. These approximate the Gussian derivative of our input values resulting in large numbers at places where colours change abruptly and small numbers elsewhere.

We also apply standard orientation feature extraction techniques Gabor [252] and HOG [113] to detect the orientation of pixels (e.g. whether a voxel contains horizontal or vertical lines). HOG calculates a gradient and then bins these gradients into a certain number of directions. Its final output is a histogram of these directions for each voxel. Gabor is a set of predefined filters consisting of a Gaussian multiplied by a sinusoidal wave. For both HOG and Gabor, we use the same voxel sizes we use for the other filters and we set the number of orientations to the default of 6.

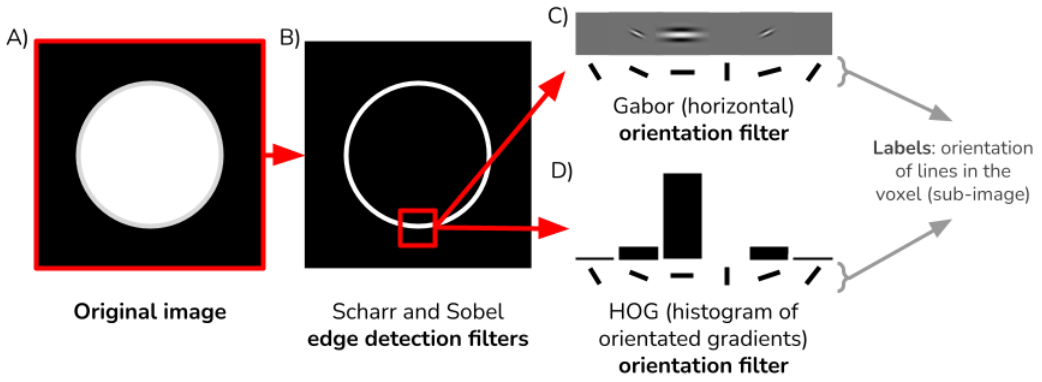

Figure 5.3: Toy example image and a visual representation of its visual features. A) is the original image, B) is the image after an edge detection filter has been applied, C) is a visual representation of the Gabor filter applied on a voxel (sub-image) from B), and D) is a HOG (histogram of gradient orientation) of the same voxel (sub-image) from B).

5.2.2 Gate representation

We represent a polygon gate $\mathbf{g}$ as a set of three or more x, y pixel coordinates $g_i = \{g_i^x, g_i^y\}, g_i \in \mathbf{g}$ corresponding to its vertices on the scatterplot image. Our representation of the gate handles arbitrarily shaped convex and, in the rare case, non-convex gates. In the latter case, $\mathbf{g}$ is an ordered list of three or more coordinates.

We call the set of pixels surrounding each vertex pixel as a 'vertex voxel'. The size s of these vertex voxels is given by the user. For both the training and testing set, we set this value to

$$s = 0.5\sqrt{\frac{f_a(g)}{n_g}}$$

where f_a is the function that, given a gate, outputs the continuous geometric area of the gate. g is the training gate and n_g is the number of vertices in g. For our experiments, this covered a sufficient amount of pixels for gate projection.

5.2.3 Workflow

Our method takes as input, a training sample and its gate, and a testing sample on which we want to project the gate. If not given, the first step in our method is to choose this training sample. After identifying the training sample, we proceed to project its gate onto the testing sample. We summarize this workflow pseudocode in Algorithm 1 and then describe it verbosely below. Its notations are as follows. T_r and T_e are the set of training and testing samples respectively. Each sample $t \in T_r \cup T_e$, contains the values from its images and visual features. In addition to these, each sample $t_r \in T_r$ contains a gate. d is the distance function used to create our distance matrix earlier in this section. θ is a user-specified distance. In our experiment, we use $k' = 3$. If any of the three training samples was an original training sample (i.e., the testing sample is similar to some original training sample), we set $k' = 1$

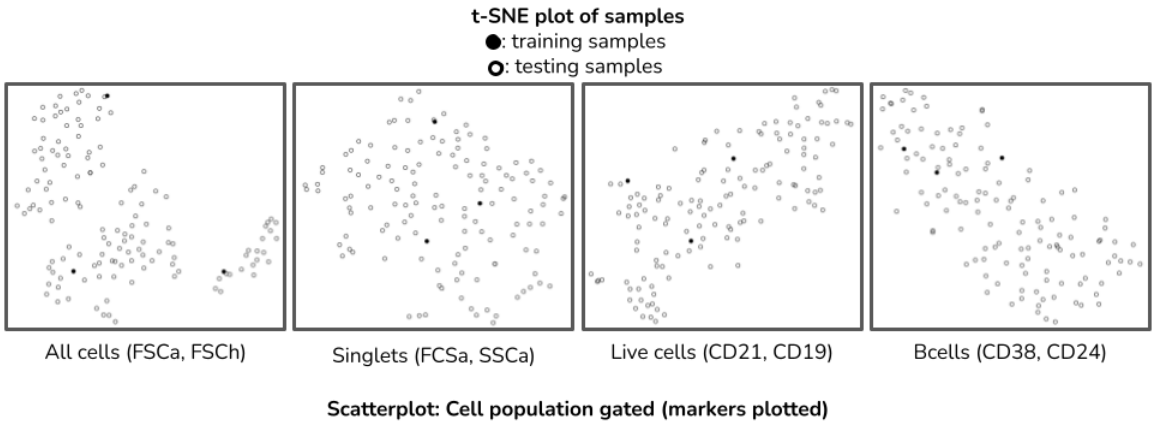

Figure 5.4: t-SNE plot representing the distance between all samples. Axis do not represent meaningful variables, t-SNE only ensures that the Euclidean distance between points on this 2D layout represents those in the given distance matrix.

and only use the closest original training sample. Finally, f_{gp} is a gate projection function that projects the gate from input training sample t''_r to testing sample t'_e and outputs this gate as $\mathbf{g_{t'_e}}$.

Algorithm 1 Pseudocode summarizing the overarching workflow of our method for cell population identification: 1) Training sample selection and embedded in it, 2) gate projection as f_{gp}. The inputs are the training samples T_r and testing samples T_e. The output is the set of all gates we projected for the given testing samples $\mathbf{g_{t'_e}}$.

Require: $|T_r| = k$
  while $|T_r| < |T_r \cup T_e|$ **do**
    $t'_e \leftarrow \arg\min_{t'_e \in T_e} d(t'_e, t_r) \forall t_r \in T_r$.
    $\mathbf{t_r}$ s.t. $\mathbf{t_r} T_r$ and $|\mathbf{t_r}| = k'$ and $\arg\min_{t_r \in \mathbf{t_r}} d(t'_e, t_r)$
    $\mathbf{w} \leftarrow \{\}$
    for $t'_r \in \mathbf{t_r}$ **do**
      $w' \leftarrow \frac{1}{d(t'_e, t'_r)}$
      $\mathbf{w} \leftarrow \mathbf{w} \cup \{w'\}$
    end for
    $\mathbf{w} \leftarrow \frac{\mathbf{w}}{max(\mathbf{w})}$
    $t''_r \leftarrow \{\sum_{w' \in \mathbf{w}, t'_r \in \mathbf{t_r}} w' t'_r, \}$
    $g_{t'_e} \leftarrow f_{gp}(t''_r, t'_e)$
    $T_r \leftarrow T_r \cup \{t'_e, \mathbf{g_{t'_e}}\}$
    $T_e \leftarrow T_e \setminus t'_e$
  end while

1. Calculate a pair-wise Euclidean distance matrix between samples.

This distance is calculated using the flattened vectorized visual features for each sample. We also considered using distance metrics, Manhattan and Normalized Cross-Correlation (NCC), but these methods produced similar (insignificantly different) results and were com-

putationally slower to calculate, respectively. From this step, we obtain a $sample \times sample$ distance matrix. Using this distance matrix, we create a t-SNE plot as illustrated in Figure 5.4.

2. Find the training samples

that we request the ground truth gates for from a human expert. We pick the original training samples by conducting agglomerative clustering using the complete link. If k, the number of training samples desired, is not provided, we conduct a post-processing step. We calculate a median silhouette index for each $k < k_{max}$ cluster where k_{max} is a user-specified maximum number of samples they were willing to gate manually. We then choose the k that yields the highest median silhouette index. Finally, We then select the medoid sample of each of the k clusters and ask the user to provide the ground truth gates for them. These would be our training samples and gates. We set $k = 3$ based on the clustering that obtained the highest median silhouette index.

3. Find the most similar (least distant) training sample for each testing sample to project the gate from.

If there is no original training sample that is similar to our testing sample (see criteria in Algorithm 1), we create a 'hybrid training sample'. This is created by taking the weighted (by distance) mean of the visual features and gate vertex coordinates belonging to the three gated testing samples most similar to our testing sample. We deduce that projecting a gate from a gated testing sample similar to our testing sample would produce more accurate results than projecting a gate from a training sample that is very dissimilar to our testing sample. In other words, we assume that the distance (see Section 5.2.3) between samples is positively correlated with some distance between their associated ground truth gates. This also mitigates situations when k may be too small to handle the heterogeneity between samples. We refer to this procedure as adding gated testing samples to our training sample set to make our training sample set larger.

4. Project the training sample's gate onto the testing sample as an optimization problem

We formally describe the gate projection method f_{gp} by reducing it to a minimum-cost node and edge clique problem. We do so by constructing the following undirected graph $G = V, E$. See Figure 5.5 for an example.

Node Recall that for each training gate vertex (e.g. bottom left vertex of training gate), we want to select the best pixel on the testing sample as the corresponding testing gate vertex (e.g. bottom left vertex of projected gate). As described in Section 5.2.1, each pixel coordinate on a sample can be represented by its surrounding voxel and the corresponding

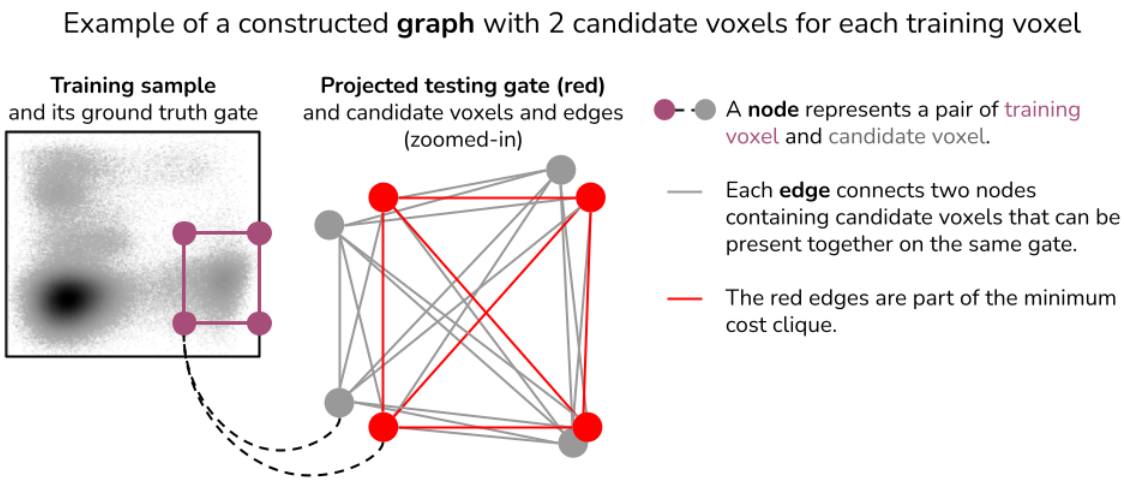

Figure 5.5: An example of a constructed graph constructed.

visual features. Therefore, we refer to gate vertices as the voxels that describe them. For brevity, we shorten the terms: training gate vertex voxels to training voxels and candidate gate vertex voxels to candidate voxels. To increase efficiency, for each training voxel, we limit the search space or the number of candidate voxels to those whose center pixel is within a $s \times s$ pixel grid centred around the training voxel coordinate on the testing sample. s is defined in Section 5.2.2. The center pixel of this grid share the same coordinate as the center pixel of the training voxel.

We define each node $v_i \in V$ as a unique pairing between a training voxel and a candidate voxel from the testing sample. Each node v_i is also assigned a cost c_i representing the visually dissimilar between its training and candidate voxel pair:

$$c_i = 2a_i + p_i$$

where a_i and p_i are the Euclidean distance between sample features and coordinate position of the training and testing candidate voxel, respectively. This follows our assumption that gates across samples are positioned in relatively similar areas of the scatterplot (e.g. bottom right corner).

Edge Each edge connects two nodes i, j that can feasibly occur together. Two nodes can feasibly occur together iff the following are true (i.e. edge constraints):

1. The two nodes (i.e. two pairs of vertex voxels) combined represent exactly four unique vertex voxels.

2. The angle between the two candidate voxels across the two nodes cannot be greater than λ degrees off from the angle between the two training voxels.

1) ensures that each training voxel must be paired with a unique candidate voxel. 2) follows our assumption that each vertex voxel should be in the same relative orientation as each other. For example, the upper-right and bottom-left training voxels should correspond to the upper-right and bottom-left candidate voxels, not the bottom-left and upper-right candidate voxels. Another example is if we have a horizontal edge on our training gate, the corresponding edge should not be vertical on the gate of our testing sample. We set λ to 75°. However, if this is a desired trait, the user should set λ to 90°or more. One of the gates we tested our method on is a perfect rectangle. If we assume we know this, we can set $\lambda = 0$. However, we assume that we do not have this information to account for cases when polygon angles are flexible.

We also assign a cost c_{ij} to the edge connecting nodes v_i and v_j.

$$c_{ij} = q_{ij} + \rho_i j$$

The variables above represent the difference in Euclidean distance (q_{ij}) and angle ($\rho_i j$) between the two training voxels and the two candidate voxels.

5.2.4 Optimization

We aim to find a unique one-to-one pairing of a candidate voxel to each training voxel. We do this by finding the minimum-cost node and edge clique inside the graph constructed in the previous section. A clique is a subgraph where each of its nodes are adjacent or connected by an edge to every other node in the clique. The node and edge cost of a clique is the sum of weights over all of its nodes and edges. It follows that a minimum-cost node and edge clique is the clique with the lowest cost. Notice that any clique in our graph contains at most h nodes and is a valid gate given our edge constraints. h is also the number of vertices in our polygon, and these nodes represent the best set of candidate voxels.

The minimum-cost node and edge clique problem is NP-complete [248]. Since this method verifies the effectiveness of our visual features and since the number of vertices we used is relatively small, we can practically use grid search to find the solution. However, we acknowledge that there are existing more efficient mature solutions that provide approximate solutions [200, 331] which should be applied as the number of vertices increase.

5.3 Experiment data and accuracy metric

We applied our method on a data set containing 3 training and 133 testing samples courtesy of Jessi Tuengel from the BC Children's Hospital and Sybil Drissler from BC Cancer Research Centre [365]. The ground truth gates were created manually and further refined using flowDensity [240], flowPeaks [121], and SamSpectral [404] by a human expert. We experimented on four scatterplots from the B-cell panel. In the context of gates, an 'arbi-

trarily shaped' polygon gate is a gate where the angle between vertices is arbitrary, instead of being the same. For example, every angle in a rectangle is 90 degrees. The gates on scatterplots plotting All cells are arbitrarily shaped polygons with four vertices, the gates on scatterplots plotting Singlets are rectangles, and the gates on scatterplots plotting Live cells and B-cells are arbitrarily shaped polygons with seven and five vertices respectively.

We show the original training samples used, their images, and their gates in Figure 5.2. We also plot these training samples relative to the testing samples in a t-SNE plot created according to the distance matrix described in Section 5.2.3 in Figure 5.4.

We used the F1 score to measure how accurately we gated the cell population of interest. The F1 score is the standard accuracy metric in cell identification papers in FCM. We calculate the F1 score using the ground truth cell population labels versus those produced by the gate our method projected. If we classify each pixel perfectly, we obtain near perfect (> 0.99) F1 scores at an even lower 256×256 pixel resolution (see Figure C.1).

We compare our F1 scores against those produced by flowSOM, a state of the art method for cell population identification. In our data set, there is one gate on each scatterplot that separates the cells into two cell populations (them being inside or outside the gate). Since flowSOM is an unsupervised clustering method, we set the number of meta-clusters in flowSOM to be five and pick the best one-to-one or one-to-many matching between clusters and cell population to get the final F1 score for each sample. The higher the number of meta-clusters, the easier it is to find a more accurate cluster to cell population matching. We choose to be more lenient because our cell population of interest does not naturally separate base on the data distribution in every sample (e.g. see Figure 5.8).

5.4 Results and discussion

In this section, we show that our method was able to obtain higher accuracy than flowSOM. We also show samples where our method scored high and low for to facilitate our discussion on why there is a large variance in our methods' F1 scores.

Our method uses visual features to accurately label cell populations identified by arbitrarily shaped polygon gates. Figure 5.6 shows that our method obtained higher mean F1 scores than flowSOM [367], a state-of-the-art method for cell population identification. This applied across all gates regardless of their shape. Although our method obtained lower scores for a few outlying samples, we observed that the gates obtained were visually reasonable. The term 'outlying' describes samples that have a long distance from the original three training samples we used based on the distance matrix from Section 5.2.3. We will expand on our interpretation of 'visually reasonable' with examples in the following sections.

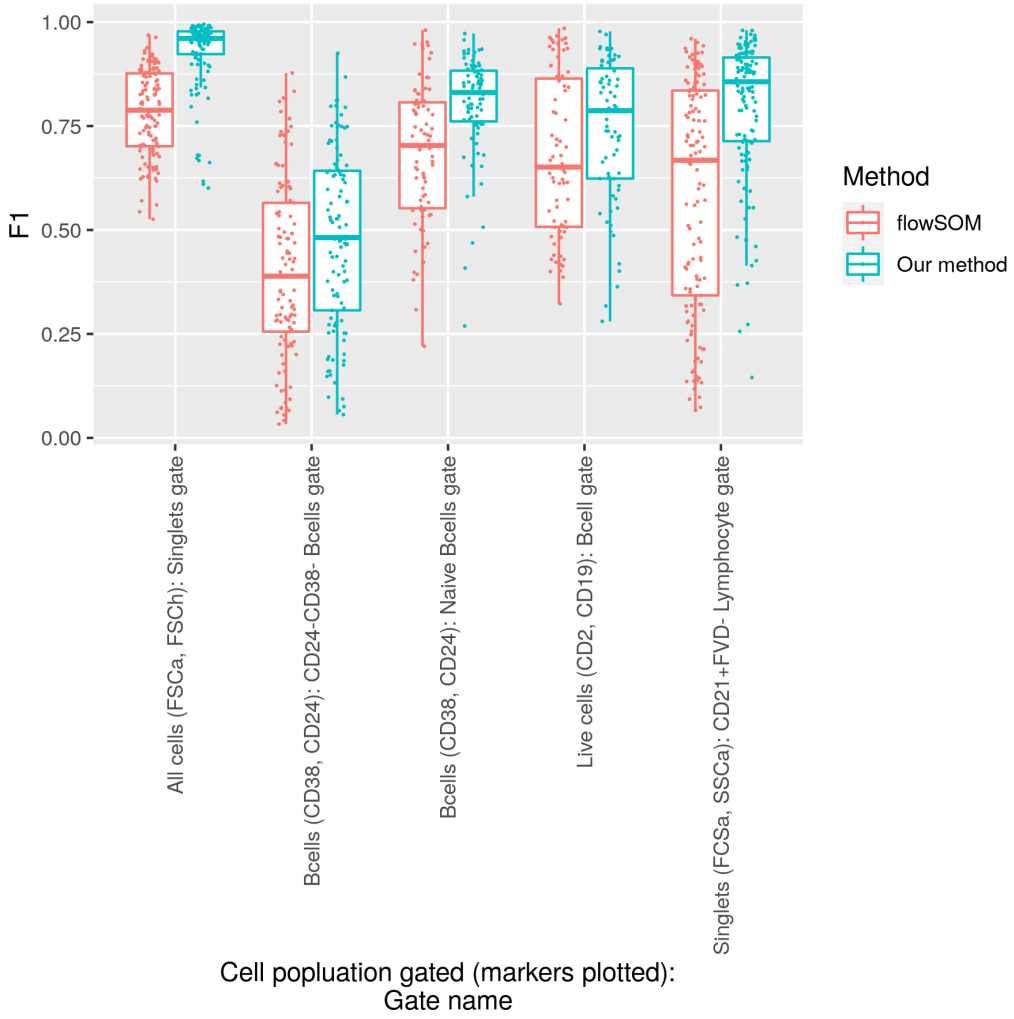

Figure 5.6: Boxplot comparison of our method's F1 accuracy scores for each testing sample and their gated cell population versus those of flowSOM [196] across all gates. F1 scores are calculated based on the cell population labelling we obtained with our method's gate and flowSOM's clustering versus that of the ground truth flowDensity gates.

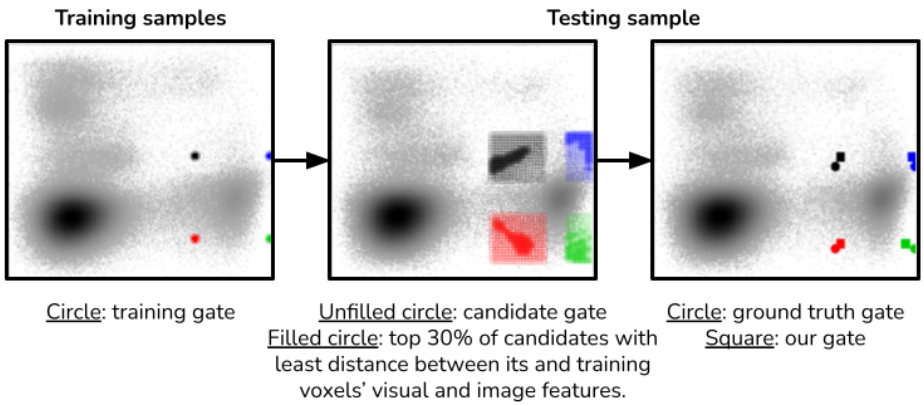

Figure 5.7: The testing sample gating for which we obtained the highest F1 score for the scatterplot plotting Singlets containing a $CD21^{+}FVD^{-}$ Lymphocyte gate; also included are the original training sample and gate used.

Visual features allow our method to project gates such that our training and chosen candidate voxels are on visually similar regions of the scatterplot, regardless of F1 score. Our method mimics the way humans would project gates on visually similar regions between training and testing samples. Therefore, we can easily interpret why our method projected the gates to where it did by looking directly at the plots. Figure 5.7 and 5.8 show the gating for the testing samples with the best and worst F1 scores for the Singlets scatterplot.

The former obtained the highest score as the original scatterplot image is almost visually identical between the training and testing samples. This is verified by analyzing the top 30% candidate voxels with the smallest a_i highlighted in the figure. Our method favoured the top scoring candidate voxels tightly wrapped around the cell population of interest. In addition to the local visual features, we also acknowledge the importance of isolating candidate voxels that are on coordinates close to the training voxels. On the scatterplot, there were many regions that were visually similar to each other. For example, the top-left candidate voxels contained visual features that were also similar to those in the top-left corner and central-left region of the scatterplot.

We observed the same behaviour in the testing sample with the lowest F1 score. Our projected gate closely aligned with the ground truth gate. However, its left side was on a dense region of the scatterplot. A slight difference in the positioning of our chosen candidate voxels would cause a large F1 score decrease. This also occurs in other low scoring samples. See Section B.1 for example plots and discussion around results for other scatterplots.

Using gated testing samples may propagate gating error. We also analyzed one testing sample that obtained a median F1 score in Figure 5.9. The gate projected onto this testing sample was from a hybrid training sample. While the spatial position of the gate vertices differed from the ground truth, the visual features in the local area around

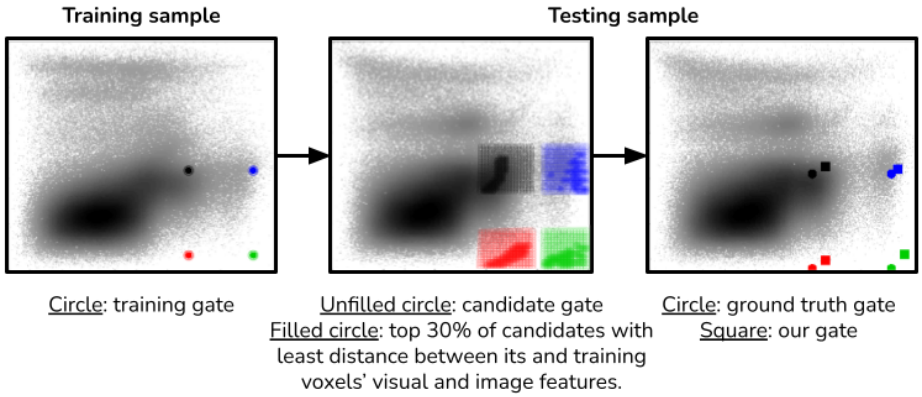

Figure 5.8: The testing sample gating for which we obtained the highest F1 score for the scatterplot plotting Singlets containing a $CD21^{+}FVD^{-}$ Lymphocyte gate; also included are the original training sample and gate used.

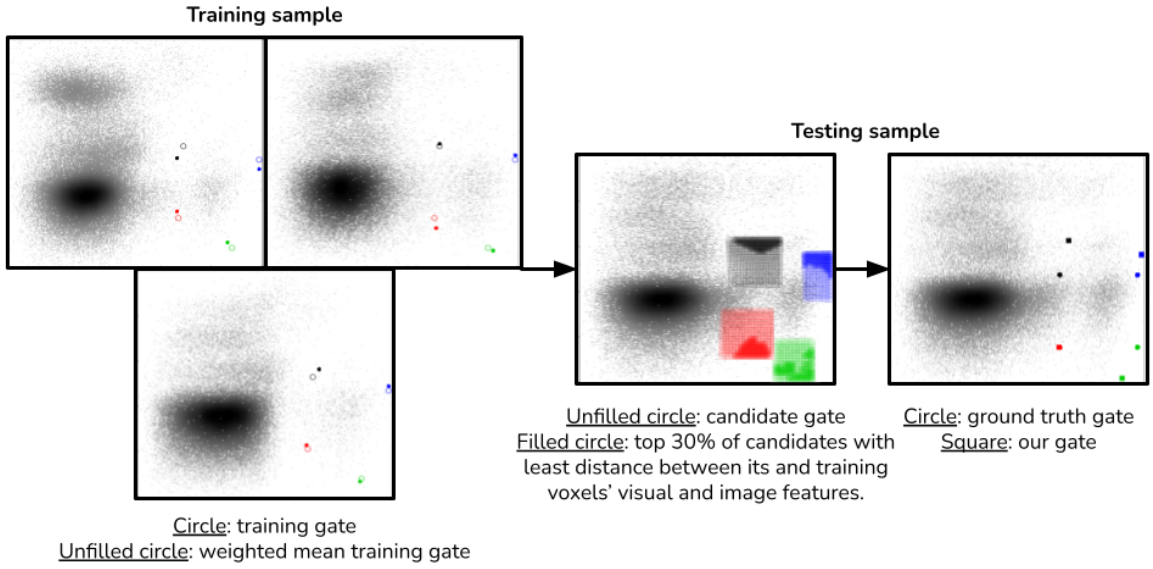

Figure 5.9: An example of gating where we used multiple gated testing samples to create a hybrid training sample that is used to gate a testing sample that obtained a median F1 score.

our vertex voxels remained consistent between training and testing samples. One concern was that our strategy to increase our training set by creating 'hybrid' training samples would propagate errors. In this case, the shape of our polygon was not rectangular as it was in the original training samples. This observation motivates us to examine what happens when we only use original training samples in the next chapter. Alternatively, if we assume that we knew that our gate should be rectangular, we could have set our λ to 0 to enforce rectangular gates; see Section 5.2.3.

5.5 Conclusion

We showed that our method that uses visual features produced interpretable results. As we opted to mimic the gating procedure, we can project a manually created gate onto a testing sample in a visually motivated way.

We also showed that supplementing visual features with spatial constraints ensured gates stayed in relatively similar locations across samples. Efficiency-wise, the optimization step of our method takes one to a few seconds per sample. The mean runtime for the optimization step projecting 4, 5, and 7 vertices polygon gates take 0.7, 0.9, and 10 seconds per sample, respectively, on a single CPU core. While our method is not significantly faster than an experienced human, it is an accurate alternative and can save human labour costs. Even for outlying samples where we obtained a low F1 score, our projected gate is a good starting point for manual modification if any is required. It also serves as a second automated opinion on how to gate outlying samples as this is not a trivial problem even for a human.

Bringing the above ideas into the next chapter, we develop a method that we think can be potentially more accurate. We also experiment on additional data sets to understand the effectiveness of using visual features to gate a larger variety of scatterplots.

Chapter 6

Automated 2D gating via few-shot image segmentation for cell population identification

In continuation of the previous chapter, we solve the same problem (Problem 1) of gating FCM samples on 2D scatterplot images. We develop a method that uses transfer learning and few-shot image segmentation. We harness a large database of human-gated FCM samples to pre-train our model. We propose and validate a training procedure that can directly use ("transfer") this trained model for gating ("segmenting") scatterplot images from FCM samples.

The work presented in this chapter is a work in progress in that our results currently do not yield higher accuracy than all of the state-of-the-art methods. In deep learning experiments, there are many engineering changes we can use to tune and refine our model. While these exemplify the immense potential of deep learning, a trade-off of such experiments is that they can be time-intensive. We believe our approach can yield higher accuracies. However, we also believe that at this stage, we can report its results and our observations as scientific contributions and lessons learned to help inform further method refinements.

6.1 Background: Few-shot image segmentation and meta-learning

Few-shot image segmentation is where we train an image segmentation model using one or 'few' training image 'shots' and their segmentation (e.g. gate) and then we use the trained model to segment previously unseen testing images. We refer to the scatterplot images used for training as few-shot-training samples. This is unlike regular image segmentation where we can train our model on hundreds of thousands of training images and their segmentation then test it on a few testing images.

Starting from [318], earlier models for few-shot segmentation contain variations of a support and query branch [291, 410, 99]. Each branch is a deep learning model. The support branch takes as input, the 'supporting' few-shot-training samples and the query branch takes as input the 'query' testing image we want to segment. The segmentation mask proposed for the testing image is a merger of the results from these two branches. These models are flexible in that they can project any segmentation from the supporting image onto the query image. However, this model requires extensive training using a large database of supporting and query images.

The MAML (model agnostic meta-learning) [106] is a type of transfer learning approach that can be applied to any image segmentation model but requires that the user knows how to add custom auxilliary nodes to augment the model. MAML first trains this augmented model on a large data set for varying tasks (e.g. to segment birds, cars, and humans). It then performs few-shot-training where the model is trained on few-shot-training samples for one specific task (e.g. to segment monkeys only). This two-step process is unlike regular training procedures where one creates and trains a custom image segmentation model for each task in one step [106]. Additional engineering strategies MAML employs are 1) augmenting training data by transforming the training set, 2) limiting hypothesis space by sharing parameters in multitask models (e.g. adding task-specific/invariant feature embedding parameters to the model) , and 3) modifying search strategy via algorithm changes [378]. 2) is the main selling point of MAML and what makes it unique from other transfer learning strategies [343]. However, model augmentation makes the training process less memory efficient and accessible for non-technical users.

In comparison, traditional transfer learning approaches do not require 2). Generic transfer learning also performs the two-step process in MAML but the second step still requires abundant training samples and is not suitable for few-shot training. We retain the same two-step training procedure but, unlike in traditional transfer learning, we do few-shot training in the second step. In addition, unlike in MAML, we do not modify the original segmentation model. Instead, we only perform few-shot training on a few layers in our network. Our training methodology is inspired by [357] who applied this on the simpler problem of image classification.

6.2 Method

6.2.1 Workflow

This section describes the overall workflow of our method including how we preprocess our data sets and perform our two-step transfer learning workflow of pre-training and few-shot-training.

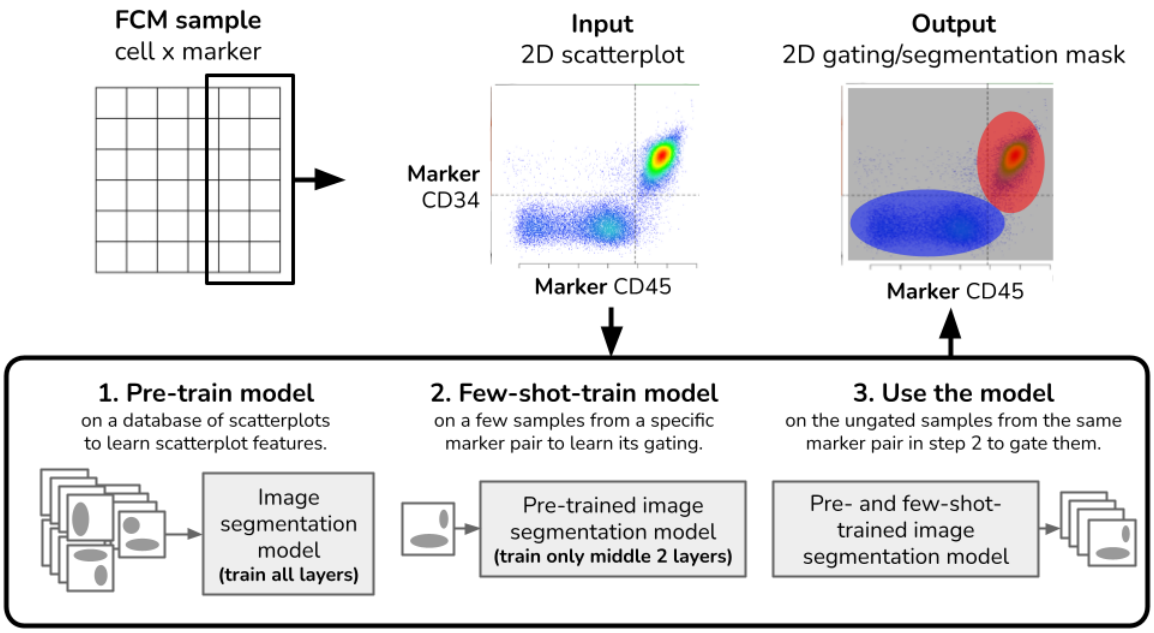

Figure 6.1: Our FCM scatterplot image segmentation workflow.

Preparing FCM data sets. Taking a preprocessed FCM sample as input (see Section 2.1), for a scatterplot or a unique user specified marker pair, we create a 256×256 pixel scatterplot image (see Section 5.4 and Figure C.1 for on image resolution experiments). This image has four channels (i.e. image-based visual features):

1. **Density**: The first channel contains values from 0 to 100 in each pixel based on a Gaussian kernel 2D density estimation of the density of cells on the pixel.

2. **Contours and density thresholds**: The second channel is a black and white image where the white pixels represent the background and the black pixels represent the lines of all naturally occurring 1) 2D density contours based on the density distribution estimated in the first channel and 2) 1D thresholds for both markers identified by flowDensity [240] i.e. horizontal and vertical lines at the density valleys, peaks, and inflection points.

3. **Row-wise position**: In this channel, the values in each row is the row number on which they reside (e.g. all values in row 1 is 1, all values in row 25 is 25). The reason for adding this channel is that purely convolution neural networks value local visual features over position-wise proximity of particular feature types to each other. This channel is used because cell population clusters in scatterplots often look alike. FI is what ultimately decides which cell population a cell belongs to and the location on the scatterplot where a cell population lies.

4. **Column-wise position**: This channel contains data just like the previous channel but for columns.

Unlike in the previous chapter, we represent each gate as a mask. A mask is a matrix whose dimensions are the same as its corresponding scatterplot image. Its background pixels contain the value 0 and the pixels inside of a gate has the value 1. If there are multiple gates per image, then, for example, the pixels inside the first, second, and third gates contain the values 1, 2, and 3 respectively. See Section 5.3 for how we prepared the ground truth gatings.

Pre-training the image segmentation model on a database of scatterplots. In the pre-training phase, we train our entire image segmentation model on large, highly heterogeneous data sets such that our model learns to effectively encode visual features from input and decode the segmentation into a gating mask. As an analogy, in the last chapter, we manually set the parameters for our visual feature filters. The parameters we used were standard for any image. Conversely, this pre-training step refines these parameters specifically scatterplot images.

To prevent over-fitting, we randomly transformed our data sets with cropping (to 80%-100% of the original size) and rotations (from -45 to 45 degrees). For our model, we used a 5 layer encoder and decoder Unet architecture [303] with ResNet18 layers [146]. We trained the model with a batch size of 32, optimized using Adam gradient descent [186], on 100 epochs. Because we have a different number of samples for each data set and their scatterplots, we sample the same number of images from each unique scatterplot i.e. we over-sampled from data sets that have fewer samples. We used the Lovász-Softmax loss which is known to work well for segmentation tasks [228].

Few-shot-training the image segmentation model on one scatterplot. In the few-shot-training phase, we take our pre-trained model and train it further to gate samples plotted on the same scatterplot. The few-shot-training samples used for each scatterplot are selected the same way as in the previous chapter (see Section 5.2.3; we do not use hybrid training samples in this chapter). We take the pre-trained model and we freeze all layers except for the centre-most two layers which are trained using the few-shot training samples. Our few-shot-training specifications follow those of the pre-training phase. We do not manually adjust optimization parameters, such as step size, because Adam gradient descent automatically adjusts those during training. From a use case perspective, end-users are directly given the pre-trained model from the previous step and only need to perform this few-shot-training step for the scatterplot they are interested in gating. Pre-training only needs to be done once while few-shot training needs to be done for each scatterplot.

6.2.2 Experiment data and accuracy

We increased our data set size to accommodate for the massive increase in parameters in deep learning models compared to our method in the previous chapter. We used 4 freely available data sets (see Table C.1) for our experiments.

Typically, for training and testing models, one would conduct k-fold cross-validation [43]. In the context of this chapter, $k = 10$-fold cross-validation would entail combining all the 46 scatterplots and then randomly splitting them up into ten evenly sized sets. A first experiment would be done where we few-shot train and test on the first set using a model pre-trained on the last nine sets of scatterplots. Similarly, this would be done nine more times with few-shot training and testing over each set on models pre-trained using the rest of the nine sets of scatterplots. We can assume that since we are randomly selecting the pre-training and few-shot training scatterplots over a larger set of scatterplots, the heterogeneity within the pre-training and few-shot training scatterplots should be similar (i.e. our model would be able to learn how to encode scatterplots in the few-shot training set because it has seen similar scatterplots in the pre-training set). This statement holds as the number of scatterplots available to us nears infinity [43].

However, in flow cytometry, we deal with sets of scatterplots given to us in gating strategies. In total, we used four sets of scatterplots from four gating strategies, so it would be more realistic to pre-train on three sets of scatterplots and few-shot train on the last set. This is effectively a 4-fold cross-validation except that the scatterplot sets were not selected randomly but via their gating strategies. Since our selection was not random, our assumption that our model would have learned to encode scatterplots in the few-shot training set from scatterplots seen in the pre-training set may not hold. We perform one of those cross-validation experiments by pre-training our model on three freely available data sets: 'sangerP2', 'HIPCB-cell', and 'HIPCMyeloid'. The sangerP2 data set contains 2,328 samples, each plotted on 12 scatterplots from panel 2 of [2]. The HIPCB-cell and HIPCmyeloid data sets (from the Human Immunology Project Consortium [105]) contain 1,350 samples plotted on 12 and 11 scatterplots from the B cell and myeloid panels respectively. Finally, we few-shot-trained and tested on the 'pregnancy' data set [7] which contains 112 samples, each plotted on 10 scatterplots. See Table C.1 complete description of these data sets.

We compared our F1 scores obtained using 10 few-shot-training samples with those of peer-reviewed, state-of-the-art methods in Table 6.1. For unsupervised clustering, we used the gigaSOM implementation of flowSOM [196, 367]. For gigaSOM, we set the number of desired meta-clusters to 6, one more than the maximum number of cell populations in all scatterplots. We then take the F1 score for the best one-to-one or one-to-many cluster to cell population matching. Again, the more meta-clusters specified, the higher gigaSOM should score. We did not use the exact number of cell populations to account for when cell populations do not separate naturally based on their data distribution. For supervised methods, we used deepCyTof [212] and flowLearn [227]. For deepCyTOF, we also used 10 training samples. As a comparison, we circumvent the lack of training samples by pre-training an effective visual feature encoder while deepCyTOF uses a separate autoencoder model to transform the data distribution in testing samples to match with that of the few-shot-training samples. Though we use much more data in the pre-training step, we

consider these strategies as preprocessing steps. Our approaches are comparable because the number of samples we use in the few-shot-training step is the same. For flowLearn, scores are calculated for scatterplots that contain only threshold gates.

We calculated all the F1 scores based on gold standard ground truth gating for each sample created using flowDensity [240] guided by manually prepared gates. These scores are averaged across cell populations for each scatterplot and their respective samples. For flowLearn and deepCyTOF, we also used 10 few-shot-training samples.

6.3 Results and discussion

Table 6.1: F1 scores of our method and three other state-of-the-art methods in cell population identification for the pregnancy data set. Scatterplot labels indicate: the cell population being gated (marker pair the scatterplot was plotted on).

Scatterplot; Method	Leukocyte (CD66, CD45)	Mononuclear (CD3, CD19)	NKLin- (CD14, CD7)	Lin- (CD14, CD16)	Tcell (CD4, CD8)	Tcell (CD4, CD45RA)	CD4+Tcell (FoxP3, CD25)	CD4+Tcell (TCRgd, CD3)	NotCD4+CD8+Tcell (CD8, CD45RA)	CD8+Tcell (Tbet, CD45RA)	Avg rank
deepCyTOF	0.76	0.62	0.98	0.77	0.96	0.86	**0.88**	0.87	0.91	0.81	3.4
gigaSOM	0.93	0.60	**0.99**	0.49	0.73	**0.96**	0.74	**0.98**	0.97	0.71	2.9
flowLearn		**0.88**	0.92	**0.95**	**0.99**	0.92			**0.99**	**0.92**	2.8
Our method	0.92	0.76	0.98	0.81	0.95	0.90	0.68	0.81	0.94	0.85	3.1
Our method (without pre-training)	**0.97**	0.71	0.97	0.84	0.98	0.94	0.73	0.80	0.95	0.80	2.8

Averaging the ranking of the score for each method and scatterplot, our method ranked first (without pre-training), tied with flowLearn, and third. From the scores, we can see that each state-of-the-art method had its own advantages. deepCyTOF is a supervised method based on FI values. It performed best for scatterplots where the FI values of cells from the same cell population across samples did not have large changes. Meanwhile, the gigaSOM implementation of flowSOM obtained high F1 scores for scatterplots where cell populations were well separated in their data distribution. flowLearn was exceptionally accurate for learning threshold gates but it cannot learn polygon gates. Scatterplots that require polygon gates exist in almost, if not all, data sets (hence the missing F1 score). Our method could gate both polygon and threshold gates while accounting for both feature

types: spatial features (FI) and visual features that represent the data distribution. We also acknowledge and investigate further the two scatterplots where our method did worse than the competitors later in this section.

6.3.1 Pre- vs few-shot-training

In the last line of Table 6.1, we show that our training framework produced results comparable to the baseline where we do not pre-train and directly trained on the 10 few-shot training samples. This baseline ranked second overall. If we do not include this result, our method also ranks second compared to the competitors.

While this draws the question of whether we should exclude the pre-training step, we need to verify this further by testing on additional data sets to analyze whether there is a significant increase in accuracy that warrants the longer training time. We hypothesized that pre-training our model allows it to learn encoding for more heterogeneous visual features. This should allow us to use fewer few-shot-train samples to get good few-shot-testing results. However, we did not see this in the pregnancy data set. Regardless of whether or not pre-training makes a difference, this is a positive sign that the heterogeneity within our data set does not cause as many problems for image segmentation models as it does for methods that only take FI as input. Conversely, a pro of pre-training is that the few-shot-training phase converges within 100 epochs across all scatterplots whereas doing it without pre-training takes within 300 epochs; this makes for more time-efficient training.

6.3.2 Few-shot-training samples

To understand how the number of few-shot-training samples affects our results, we experimented with 1, 5, 10, 15, and 20 few-shot-training samples. The mean F1 score obtained across cell populations for each scatterplot is shown in Figure 6.2. F1 scores significantly increased after increasing the few-shot-training set size from 1 to 5; beyond that, the F1 score converged as the few-shot-training sample size continued to increase. This may be because 5 few-shot-training samples were sufficiently representative of the full sample set. However, we acknowledge that our full sample set (112 samples) is much larger than that of a typical FCM experiment (10-20 samples). In the typical case, a few-shot-training sample size under 5 should be sufficient for highly heterogeneous sample sets.

6.3.3 Gate, sample, and feature heterogeneity

In this section, we show and discuss the most representative sample(s) and the samples with the highest and lowest F1 scores from each scatterplot in Figures 6.3 to 6.12. We go over what the model may be doing to learn gates and possible ways to improve our results.

In summary, the main advantage of methods that use visual features is their ability to deal with heterogeneous data while using a small amount of few-shot-train samples. As we

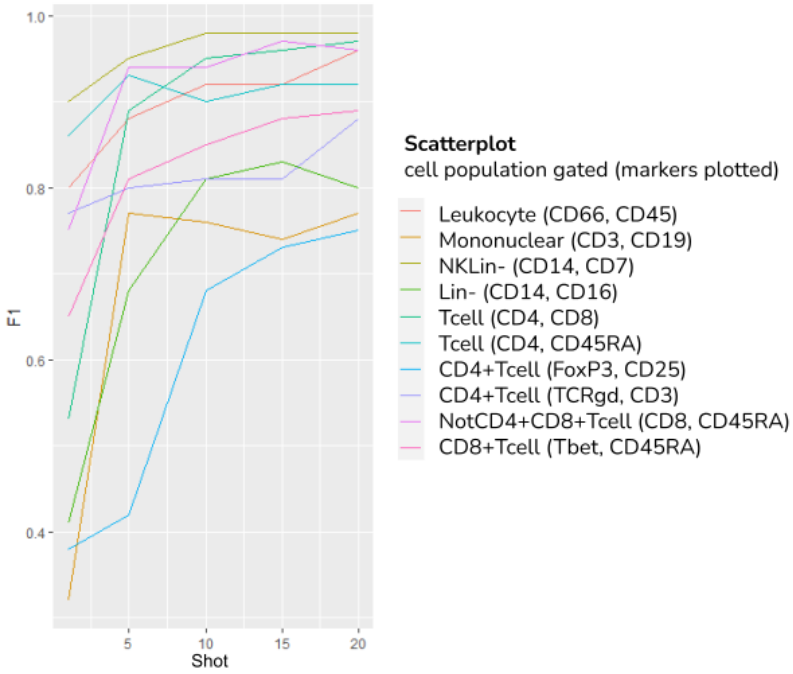

Figure 6.2: Mean F1 score when few-shot-training is done on 1, 5, 10, 15 samples (shots) across cell populations for each scatterplot.

can supply abundant feature types to the more flexible deep learning model, it can take into account features outside of just FI values. The heterogeneity in gates, however, posed a problem. We discuss this in detail below.

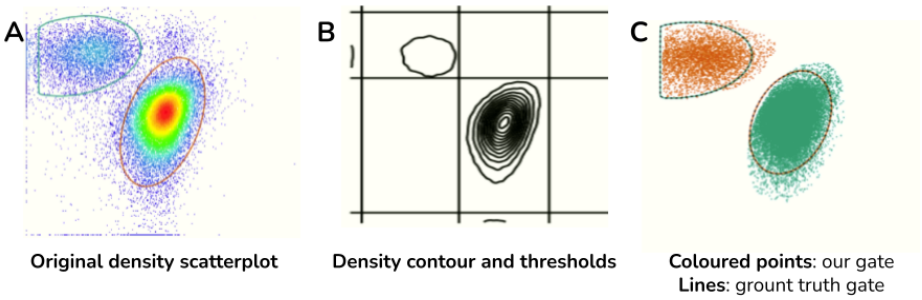

Figure 6.3: Sample scatterplot result for Leukocyte (CD66, CD45) cells from the pregnancy data set. A), B), C) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample.

Our method scores well if the ground truth gate aligns with a density contour or a threshold. Our model yields a higher F1 score when gates conform to density contours in channels 1 and 2. See Figures 6.3 and 6.10. In both cases, there was some indication that they were an isolated group of cells in channel 1 and strong contours for their gates in channel 2. Given the encoder-decoder nature of segmentation models, we also know that

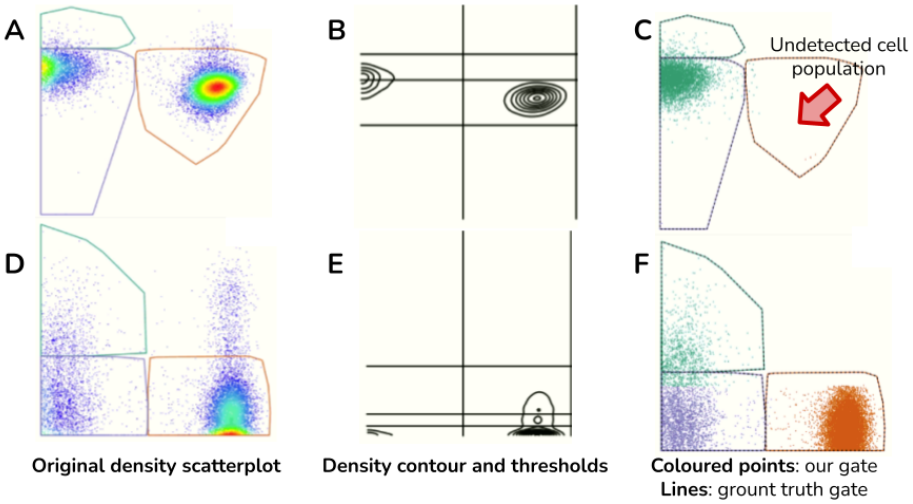

Figure 6.4: Sample scatterplots result for Mononuclear (CD3, CD19) cells from the pregnancy data set. A/D), B/E), C/F) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample. Top and bottom rows show an FCM sample with the best and worst F1 score respectively.

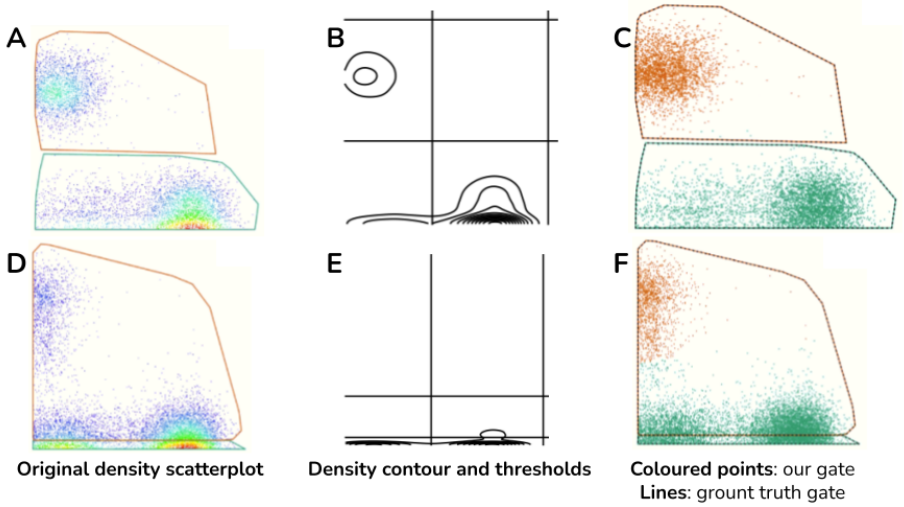

Figure 6.5: Sample scatterplot result for NKLin- (CD14, CD7) cells from the pregnancy data set. A/D), B/E), C/F) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample. Top and bottom rows show an FCM sample with the best and worst F1 score respectively.

the gates they produce were smooth and would not have sharp edges. This is advantageous for gating round gates.

Our model learned to align gate borders with threshold lines from channel 2 of the given scatterplot image. This is a trend we noticed in particular with scatterplots that have gates made of threshold lines. See Figures 6.4, 6.5, 6.8, 6.11, and 6.12. This behaviour also leads to the lower accuracy results because the model followed the lines that

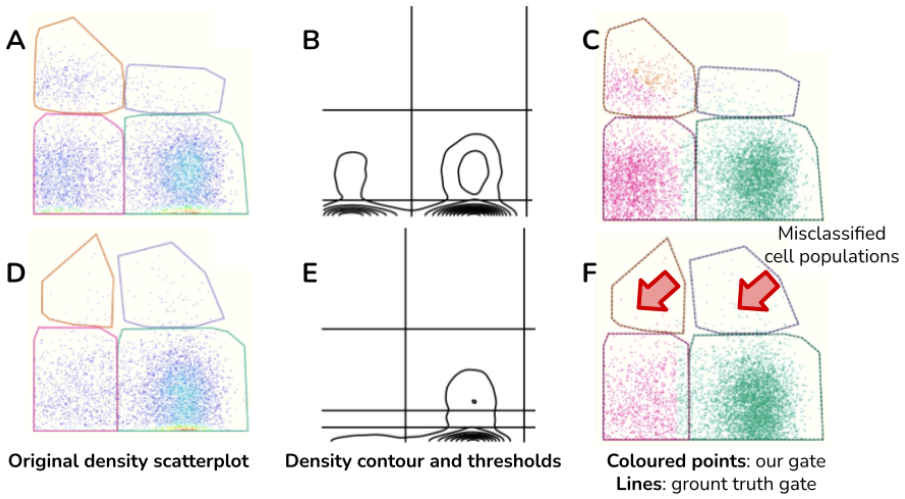

Figure 6.6: Sample scatterplot result for Lin- (CD14, CD16) cells from the pregnancy data set. A/D), B/E), C/F) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample. Top and bottom rows show an FCM sample with the best and worst F1 score respectively.

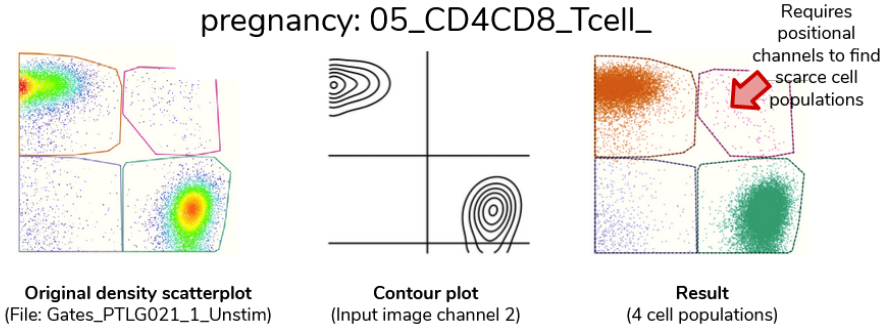

Figure 6.7: Sample scatterplot result for Tcell (CD4, CD8) from the pregnancy data set. A), B), C) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample.

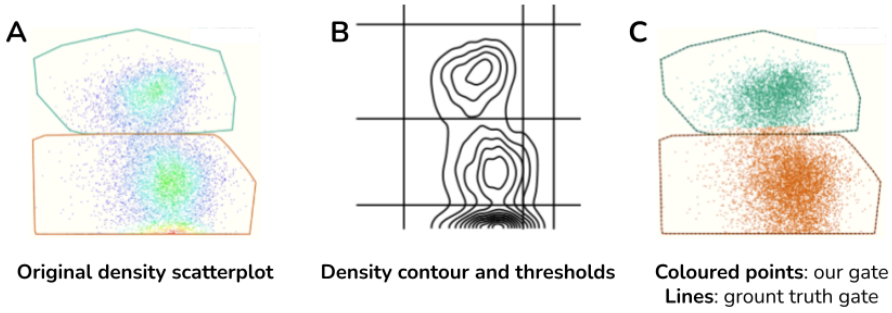

Figure 6.8: Sample scatterplot result for Tcell (CD4, CD45RA) from the pregnancy data set. A), B), C) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample.

do not align with the ground truth gate in scatterplots, as shown in Figures 6.4, 6.5, 6.9 and 6.12. This may be because our model was not penalized heavily when gate borders do

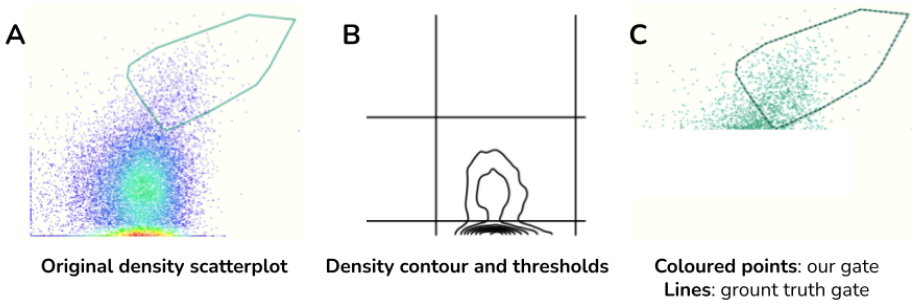

Figure 6.9: Sample scatterplot result for CD4+Tcell (FoxP3, CD25) from the pregnancy data set. A), B), C) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample.

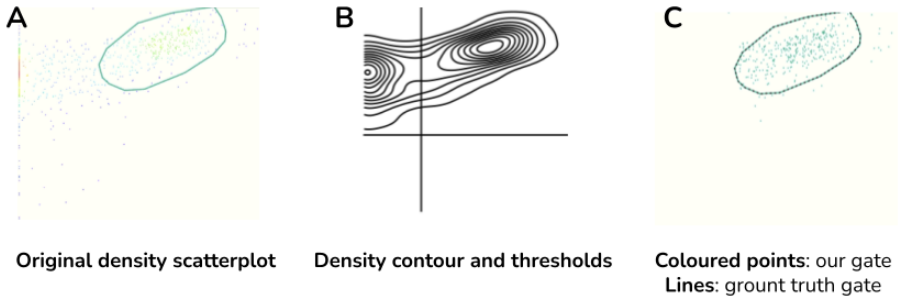

Figure 6.10: Sample scatterplot result for CD4+Tcell (TCRgd, CD3) from the pregnancy data set. A), B), C) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample.

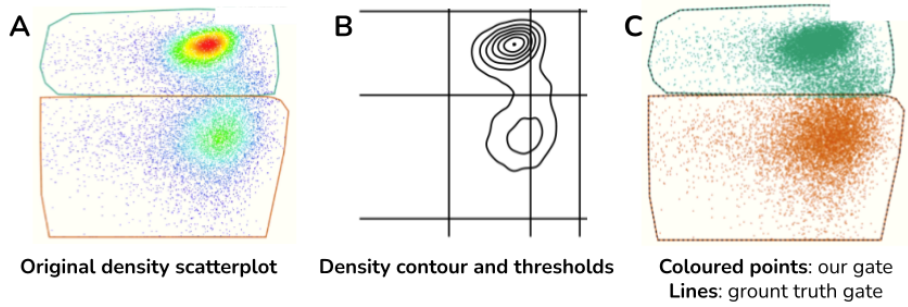

Figure 6.11: Sample scatterplot result for NotCD4+CD8+Tcell (CD8, CD45RA) from the pregnancy data set. A), B), C) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample.

not match with the ground truth exactly. A possible future experiment is to use the Dice coefficient [341] as the loss function as it is calculated based on the distance between actual and predicted gate borders, serving as a penalization [77].

Our method does not score well if 1) the ground truth gate does not align with density contours and thresholds or 2) is on a low-density area of the scatterplot. Another possible reason for the mentioned gating errors is that the ground truth gates do not align with any given density contours and thresholds.

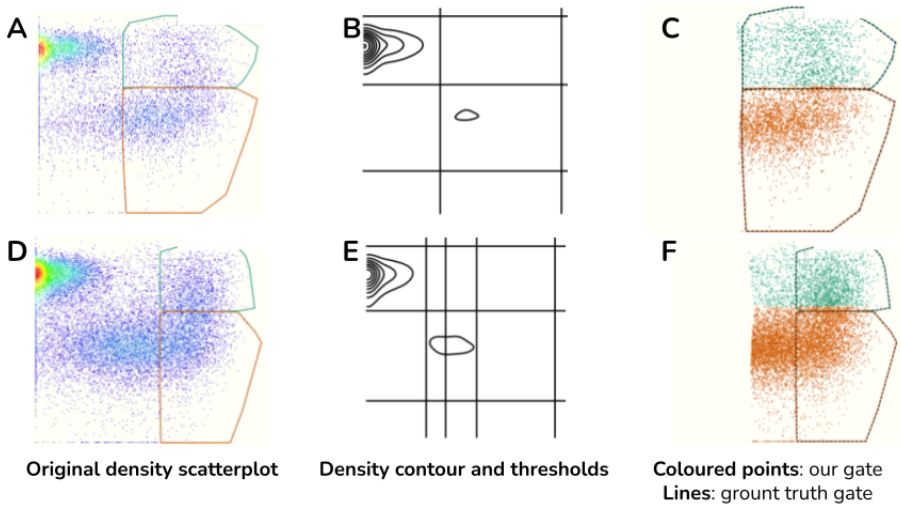

Figure 6.12: Sample scatterplot result for CD8+Tcell (Tbet, CD45RA) from the pregnancy data set. A/D), B/E), C/F) are the original density scatterplot image, density contour and thresholds, and gating result for the FCM sample. Top and bottom rows show an FCM sample with the best and worst F1 score respectively.

While straight gates help immensely, the model still needs the help of density-based visual cues from channel 1 to affirm its straight gates. This can be seen in the scatterplots of Figure 6.6, 6.7 where our cell population of interest resides on pixels with near 0 low-density values and no density contour. Though the scatterplot in Figure 6.9 did the opposite of what was expected, it too lacked the density-based visual cues required. Without visual density cues or contour/lines to guide its gate, the most valuable information we gave the model to gate this scatterplot in Figure 6.9 were the row- and column-wise position values in channels 3 and 4. We also know that it is a V-shaped gate, but there was no indication as to where this V-shape should reside.

One possible solution is to see if we can reveal more signals in the density channel. To do this, one future experiment will be where we transform our kernel density estimate values with log transform to ensure our data is spread out.

6.3.4 Pre-training data set selection

Another question we want to discuss is: how many pre-training scatterplots are required to train an encoder that can effectively encode images from the few-shot training scatterplot set for few-shot segmentation? The answer to this depends on the complexity of the problem and its corresponding model, and the heterogeneity of the few-shot training and testing data set the model will be used for [43]. In terms of complexity, models such as deep convolutional or transformer neural networks have high complexity because they have a relatively high number of parameters. A large number of parameters require more data to train with. For heterogeneity, each gating strategy typically has around ten scatterplots (as do our data sets). So as long as the pre-training data set contains scatterplots with similar visual features

and segmentation as those in the ten scatterplots. Since we assume that we do not know what few-shot training scatterplots our model will have to train on in the future, we assume that the more heterogeneous our pre-training data set, the better.

Heterogeneity is subjective, however, so the most effective way to evaluate this is through empirically evaluating 1) the use of 1, 2, and 3 data sets in the pre-training set and 2) different levels of heterogeneity in the pre-training set, and then comparing the resulting few-shot testing F1 scores. For 2), ideally, we would pre-train with all possible combinations of scatterplots in the three pre-training data sets, measure the heterogeneity in each combination using a heterogeneity metric (e.g. [246]), and then plot the F1 scores for each scatterplot in the fourth few-shot training data set to see how F1 scores scale with heterogeneity. The same relationship can be plotted for 1) but instead of having heterogeneity on one axis, we have the number of scatterplots in the pre-training data set. As each pre-training takes a substantial amount of time, we move these outside the scope of this thesis. However, it is worth examining in the future.

6.4 Conclusion

In this chapter, we presented a work-in-progress method to gate FCM samples using visual features, with only a few training samples. Our method is a few-shot segmentation trained using a unique but simple transfer learning procedure that does not require the user to modify the original image segmentation model architecture. Our approach requires human experts to create gating strategies giving the user more control in exchange for less automation. However, with efforts to standardize gating strategies and gates [105] combined with our approach, we believe a fully automated 2D gating pipeline is possible.

In the future, we want to engineer the ability to recognize and apply standard gating strategies to our framework; but in the near future, we want to continue to improve our method through additional experiments and modifications. These include expanding our experiments to additional data sets and a larger variety of scatterplot types to verify how well our training procedure can generalize to even more heterogeneous data sets. We learned in Chapter 5 that spatial features assist with finding gates in similar regions of the scatterplot. Therefore, on top of inputting spatial data, we want to try our approach on different models. Specifically, models that account for global visual features. An example of this type of model uses visual transformers encoders, a global feature extraction encoder that has been gaining traction in the neural network community [412]. We also mentioned future work to amplify signals in our kernel density estimate feature by performing log transform and using the Dice coefficient loss function to increase the accuracy of gates along its borders.

In summary, for publication, we plan to add to our current contents: 1) engineering refinements to the method to improve accuracy especially for the the scatterplots all methods

perform poorly on and 2) extend current experiments on the additional data sets. For 1), we aim to gain a top mean ranking over the state-of-the-art methods.

Chapter 7

Conclusion and future work

The field of cytometry bioinformatics will continue to grow given the high usage of cytometry in biology. In this thesis, we presented solutions to two of the most common problems in the cytometry data analysis pipeline. With more solutions being created, we hope to increase their accessibility to push them into industrialized environments. In this chapter, we discuss future work that tackle this in two ways: 1) fill in gaps in the data analysis pipelines to make it fully automated and easy to use, and 2) maintaining interpretability of results allowing users to retain control.

7.1 High-dimensional single-cell data registration

On the experiment level, one of the biggest challenges in single-cell data analysis is the lack of comparability between samples and data modalities produced in different experiments and institutions. Here, we define data modalities as data coming from different biotechnologies. For example, we can analyze the same blood sample using FCM and scRNAseq, these are two data modalities. We experienced the challenge of analyzing data from different institutions in our master's thesis working with IMPC data [398]. Although the data set from multiple institutions were available, we were unable to make valid comparisons between these samples. While aligning protocols is one solution, putting all this data together, standardizing meta-data, and making them readily available for public use requires extensive efforts [6]. This process of processing and analyzing data from various sources is called meta-analysis.

Current meta-analysis tools available for single-cell level analysis are directly adapted from bulk data analysis tools [205, 279]. These tools bring with them the assumption that the same cell population exists and can be compared across samples. However, these assumptions are not always true. Extending our current work, we want to explore ways to register single-cell data from different experiments and data modalities into a larger database whose data is readily available for meta-analysis.

Our vision for sample registration is two-fold. First, it allows for standardized cell population identification across experiments and data modalities — while accounting for the fact that cell populations may not match perfectly across samples. Second, our tool should help us visualize and interpret this standardized matching at a granular single-cell level.

Aim 1: Unified cell population identification across data modalities While single-cell data comes naturally from FCM, recent advances in genomics have led to additional single-cell data modalities, such as scRNAseq. Analyzing and integrating multiple data modalities has the potential to explain holistic biological systems. One way to integrate single-cell data is to find common cell populations across data modalities while accounting for possible heterogeneity between samples. We will focus on discrete cell population and cell state modelling in this section.

Like threshold gating on markers in FCM, classification of discrete cell populations in scRNAseq, scATACseq, and other forms of omics data are currently done exclusively via thresholding on a limited set of user-specified genes. This is where certain cell populations express either positively or negatively for some set of genes [207]. However, we know from our work in FCM that not all cell populations conform to strict thresholding. Not only do we have positive and negative marker conditions, we also leave open the option of not including a marker condition altogether (i.e. $+$, $-$, no condition). We also know that cell populations can be enclosed in polygons made up of edges that may not be orthogonal to any dimension. We extend these assumptions to omics data. We propose an automated way to classify cell populations while maintaining the act of thresholding that is common across FCM and other single-cell data modalities. This unifies the definition of cell population across single-cell data modalities making their cell population and motivating features (protein and genes) comparable with each other.

One way to do this is space partitioning, one of the oldest forms of classification. Binary space partitioning (BSP) partitions space using planes and density contours. BSP is more flexible than the Mondrian process as it does not restrict its hyperplanes to be orthogonal to a particular dimension. It is also more efficient and interpretable than the random tessellation process as users can restrict the hyperplanes to be orthogonal to all but two dimensions. Optimization algorithms, however, are extremely slow. This is especially true in high dimensions due to the large search space [120]. This search space can be limited by our assumption that hyperplanes should land on areas with certain density motifs. This not only increases efficiency but ensures our hyperplanes conform with our assumptions for threshold gates.

Aim 2: Single-cell data registration via domain adaptation With a unified definition of cell populations as the unit of analysis, we need to ensure that they are comparable samples of different experiments. In our case, we can easily compare cell populations across

samples by using the cell population hierarchy representation used in this thesis. The label of each cell population would be a combination of marker (or gene) conditions. Currently, researchers have combined the process of cell population identification and cell population matching across samples by developing complex neural networks; but again, these are made based on the assumption that the same cell populations exist across all samples [215, 202]. However, due to their unified approach, they can visualize these results at a single-cell level. Our approach, so far, does not support single-cell level matching. Though our cell population hierarchy provides clear visualizations, we acknowledge that it is also important to show results on a single-cell level.

Domain adaptation provides single-cell level matching across samples of different experiments and data modalities. We show an example of a way we can do this below. The advantage of this strategy is that it is model agnostic; as such, changes in one model can be directly propagated and applied to new data sets. This is an important consideration given the heterogeneity in our data. More specifically, consider the following scenario. Given a gold standard and a new $cell \times feature$ matrix, $\mathcal{X}_\mathcal{T}$ and $\mathcal{X}_\mathcal{S}$ respectively, their cell population labels $\mathcal{Y}_\mathcal{T}$ and $\mathcal{Y}_\mathcal{S}$; there is a function ϕ that transforms our sample such that $\psi_T(\phi(X_S)) = \psi_S(X_S)$ where ψ_T and ψ_S are their ground truth cell population classification functions. If we assume ψ_T can be represented as a set of point clouds (e.g. of a set of convex hulls) enclosing cell populations Y'_T, then $\phi^{-1}(Y'_T) = Y'_S$. Naively, our goal is to find ϕ such that

$$\arg\min_{\phi} \; d_\mathcal{Y}(\psi_T(\phi(X_S)), \psi_S(X_S)) \tag{7.1}$$

where $d_\mathcal{Y}$ is some distance metric in $\mathcal{Y}$ space.

However, we cannot directly solve for this problem. ϕ needs to be a nonlinear transformation that preserve order among cells. As well, we cannot assume that we have a ground truth Y_S. Therefore, we can re-frame our problem as:

$$\arg\min_{\phi} \; d_\mathcal{X}(\varphi(\phi(X_S)), \varphi(X_T)) \tag{7.2}$$

where $d_\mathcal{X}$ is some distance metric in $d_\mathcal{X}$ space.

7.2 Comprehensive reasoning framework for identifying and interpreting biomarkers in multi-modal hierarchical data sets

We showed the importance of incorporating the hierarchical relationship between cell populations when identifying driver cell populations as biomarkers. We want to make this framework of biomarkers identification more accessible in the context of multi-modal data sets. Overly complex models made specifically for a single task lose their utility beyond their first publication because of their inability to generalize to other variations of the task. There-

fore, we aim to compartmentalize this task into two simpler tasks: 1) map out important dependencies between cell population features, and 2) create an interface for interpreting the results.

Aim 3: Dependency-aware feature integration and interpretation Like the way we created the cell population hierarchy, we want to create something similar for cell populations in multi-modal data. Given the increase in data, we want to do this more efficiently. Instead of accounting for all possible relationships between cell populations, we create a dependency network where we only keep significantly dependent relationships. Note that we do assume we can build a strictly causal relationship network. Building a causal relationship network requires us to assume that we have accounted for all possible batch effects and confounding variables. We do not make this assumption. Given the cell population hierarchy for each data modality, we can create this network by iteratively removing independent relationships between cell populations. Then for each outcome variable, we map out the relationship between cell populations of different modalities given an outcome variable. Iteratively adding on only important connections create sparse networks which are easy to interpret. If only a few features are directly related to the outcome variable, we can explain how those features can then affect other features to create a propagated effect via message passing [241, 32, 408].

Aside from the accuracy and interpretability of our results, we want to make our methods accessible to users. The solutions implemented in our thesis require knowledge of computer programming to use — even if usage consists of a one-line command on a terminal. Conversely, everyone who uses a computer is familiar with the visual user interface. Visualizations are often made for specific data formats without the original biological anecdote in mind. For example, networks lend well to visualizations, but the context of these networks and how they connect to other data modalities are not always incorporated. Even when user interfaces and visualizations are created, they are difficult to maintain and are discarded when new methods are created. However, effective interfaces and visualizations take effort to create and verify. These require a combined knowledge of user interface design with user studies, visual arts, and visualization design. Therefore, we see opportunities to create an accessible unified front-end interface for the data analysis pipeline. To make the interface easy to maintain and not a one-time throw-away project, we need to make it easy for researchers to build modules for it. For example, if a new method comes out for the cell population identification step of the pipeline, then it should be easy for the method to accept and give inputs and outputs, respectively, (and specific visualizations if available) to the interface. We see interface and visualization development as an ongoing process done in parallel across projects.

Bibliography

[1] Tamim Abdelaal, Vincent van Unen, Thomas Höllt, Frits Koning, Marcel JT Reinders, and Ahmed Mahfouz. Predicting cell populations in single cell mass cytometry data. *Cytometry Part A*, 95(7):769–781, 2019.

[2] Lucie Abeler-Dörner, Adam G Laing, Anna Lorenc, Dmitry S Ushakov, Simon Clare, Anneliese O Speak, Maria A Duque-Correa, Jacqueline K White, Ramiro Ramirez-Solis, Namita Saran, et al. High-throughput phenotyping reveals expansive genetic and structural underpinnings of immune variation. *Nature immunology*, 21(1):86–100, 2020.

[3] Basel Abu-Jamous and Steven Kelly. Clust: automatic extraction of optimal co-expressed gene clusters from gene expression data. *Genome biology*, 19(1):172, 2018.

[4] Nima Aghaeepour. Flowmeans: non-parametric flow cytometry data gating. *R package version*, 1(0), 2010.

[5] Nima Aghaeepour, Pratip K Chattopadhyay, Maria Chikina, Tom Dhaene, Sofie Van Gassen, Miron Kursa, Bart N Lambrecht, Mehrnoush Malek, GJ McLachlan, Yu Qian, et al. A benchmark for evaluation of algorithms for identification of cellular correlates of clinical outcomes. *Cytometry Part A*, 89(1):16–21, 2016.

[6] Nima Aghaeepour, Greg Finak, Holger Hoos, Tim R Mosmann, Ryan Brinkman, Raphael Gottardo, Richard H Scheuermann, FlowCAP Consortium, DREAM Consortium, et al. Critical assessment of automated flow cytometry data analysis techniques. *Nature methods*, 10(3):228–238, 2013.

[7] Nima Aghaeepour, Edward A Ganio, David Mcilwain, Amy S Tsai, Martha Tingle, Sofie Van Gassen, Dyani K Gaudilliere, Quentin Baca, Leslie McNeil, Robin Okada, et al. An immune clock of human pregnancy. *Science immunology*, 2(15):eaan2946, 2017.

[8] Nima Aghaeepour, Radina Nikolic, Holger H Hoos, and Ryan R Brinkman. Rapid cell population identification in flow cytometry data. *Cytometry Part A*, 79(1):6–13, 2011.

[9] Nima Aghaeepour, Erin F Simonds, David JHF Knapp, Robert Bruggner, Karen Sachs, Anthony Culos, Pier Federico Gherardini, Nikolay Samusik, Gabriela Fragiadakis, Sean Bendall, et al. Gatefinder: Projection-based gating strategy optimization for flow and mass cytometry. *Bioinformatics*, 2018.

[10] Ashar Ahmad and Holger Fröhlich. Towards clinically more relevant dissection of patient heterogeneity via survival-based bayesian clustering. *Bioinformatics*, 33(22):3558–3566, 2017.

[11] H. A. Ahmed, P. Mahanta, D. K. Bhattacharyya, J. K. Kalita, and A. Ghosh. Intersected coexpressed subcube miner: An effective triclustering algorithm. In *2011 World Congress on Information and Communication Technologies*, pages 846–851, 2011.

[12] Hasin Afzal Ahmed, Priyakshi Mahanta, Dhruba Kumar Bhattacharyya, and Jugal Kumar Kalita. Shifting-and-scaling correlation based biclustering algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(6):1239–1252, 2014.

[13] David Amar, Daniel Yekutieli, Adi Maron-Katz, Talma Hendler, and Ron Shamir. A hierarchical bayesian model for flexible module discovery in three-way time-series data. *Bioinformatics*, 31(12):17–26, 2015.

[14] El-ad David Amir, Kara L Davis, Michelle D Tadmor, Erin F Simonds, Jacob H Levine, Sean C Bendall, Daniel K Shenfeld, Smita Krishnaswamy, Garry P Nolan, and Dana Pe'er. visne enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia. *Nature biotechnology*, 31(6):545–552, 2013.

[15] Matthew Amodio, David Van Dijk, Krishnan Srinivasan, William S Chen, Hussein Mohsen, Kevin R Moon, Allison Campbell, Yujiao Zhao, Xiaomei Wang, Manjunatha Venkataswamy, et al. Exploring single-cell data with deep multitasking neural networks. *Nature methods*, pages 1–7, 2019.

[16] Benedict Anchang, Mary T Do, Xi Zhao, and Sylvia K Plevritis. Ccast: a model-based gating strategy to isolate homogeneous subpopulations in a heterogeneous population of single cells. *PLoS computational biology*, 10(7):e1003664, 2014.

[17] Simon Anders and Wolfgang Huber. Differential expression of rna-seq data at the gene level–the deseq package. *Heidelberg, Germany: European Molecular Biology Laboratory (EMBL)*, 2012.

[18] Tallulah S. Andrews and Martin Hemberg. Dropout-based feature selection for scrnaseq. *bioRxiv*, page 65094, 2018.

[19] Tallulah S Andrews and Martin Hemberg. Identifying cell populations with scrnaseq. *Molecular aspects of medicine*, 59:114–122, 2018.

[20] Tallulah S Andrews, Vladimir Yu Kiselev, Davis McCarthy, and Martin Hemberg. Tutorial: guidelines for the computational analysis of single-cell rna sequencing data. *Nature protocols*, 16(1):1–9, 2021.

[21] Cedric Arisdakessian, Olivier Poirion, Breck Yunits, Xun Zhu, and Lana X Garmire. Deepimpute: an accurate, fast, and scalable deep neural network method to impute single-cell rna-seq data. *Genome biology*, 20(1):1–14, 2019.

[22] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[23] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan Peter Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, 25(1):25–29, 2000.

[24] Lo A¯c Cerf, Jeremy Besson, Kim-Ngan T. Nguyen, and Jean-Fran A§ois Boulicaut. Closed and noise-tolerant patterns in n-ary relations. *Data Mining and Knowledge Discovery*, 26(3):574–619, 2013.

[25] Wassim Ayadi, Mourad Elloumi, and Jin-Kao Hao. Bicfinder: a biclustering algorithm for microarray data analysis. *Knowledge and Information Systems*, 30(2):341–358, 2012.

[26] Wassim Ayadi, Ons Maatouk, and Hend Bouziri. Evolutionary biclustering algorithm of gene expression data. In *2012 23rd International Workshop on Database and Expert Systems Applications*, pages 206–210, 2012.

[27] Ariful Azad, Bartek Rajwa, and Alex Pothen. immunophenotype discovery, hierarchical organization, and template-based classification of flow cytometry samples. *Frontiers in Oncology*, 6:188, 2016.

[28] Amos Bairoch, Brigitte Boeckmann, Serenella Ferro, and Elisabeth Gasteiger. Swiss-prot: juggling between evolution and stability. *Briefings in bioinformatics*, 5(1):39–55, 2004.

[29] Kevin P Balanda and HL MacGillivray. Kurtosis: a critical review. *The American Statistician*, 42(2):111–119, 1988.

[30] Yael Baran, Arnau Sebe-Pedros, Yaniv Lubling, Amir Giladi, Elad Chomsky, Zohar Meir, Michael Hoichman, Aviezer Lifshitz, and Amos Tanay. Metacell: analysis of single cell rna-seq data using k-nn graph partitions. *bioRxiv*, page 437665, 2018.

[31] Ali Bashashati and Ryan R Brinkman. A survey of flow cytometry data analysis methods. *Advances in bioinformatics*, 2009, 2009.

[32] Mohsen Bayati, David F Gleich, Amin Saberi, and Ying Wang. Message-passing algorithms for sparse network alignment. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(1):1–31, 2013.

[33] Burkhard Becher, Andreas Schlitzer, Jinmiao Chen, Florian Mair, Hermi R Sumatoh, Karen Wei Weng Teng, Donovan Low, Christiane Ruedl, Paola Riccardi-Castagnoli, Michael Poidinger, et al. High-dimensional analysis of the murine myeloid cell system. *Nature immunology*, 15(12):1181–1189, 2014.

[34] Etienne Becht, Yannick Simoni, Elaine Coustan-Smith, Evrard Maximilien, Yang Cheng, Lai Guan Ng, Dario Campana, Evan Newell, and Jonathan Wren. Reverse-engineering flow-cytometry gating strategies for phenotypic labelling and high-performance cell sorting. *Bioinformatics*, 2018.

[35] Richard Ernest Bellman. Rand corporation (1957). *Dynamic programming*.

[36] Amir Ben-Dor, Benny Chor, Richard M. Karp, and Zohar Yakhini. Discovering local structure in gene expression data: the order-preserving submatrix problem. In *Proceedings of the sixth annual international conference on Computational biology*, pages 49–57, 2002.

[37] Sven Bergmann, Jan Ihmels, and Naama Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical Review E*, 67(3):31902, 2003.

[38] José M Bernardo and Raúl Rueda. Bayesian hypothesis testing: A reference approach. *International Statistical Review*, 70(3):351–372, 2002.

[39] Marina Bessarabova, Eugene Kirillov, Weiwei Shi, Andrej Bugrim, Yuri Nikolsky, and Tatiana Nikolskaya. Bimodal gene expression patterns in breast cancer. *BMC genomics*, 11(1):S8, 2010.

[40] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer, 1999.

[41] Anirban Bhar, Martin Haubrock, Anirban Mukhopadhyay, and Edgar Wingender. Multiobjective triclustering of time-series transcriptome data reveals key genes of biological processes. *BMC bioinformatics*, 16(1):200, 2015.

[42] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE transactions on pattern analysis and machine intelligence*, 22(7):719–725, 2000.

[43] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[44] J Martin Bland and Douglas G Altman. Multiple significance tests: the bonferroni method. *Bmj*, 310(6973):170, 1995.

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

[46] Anthony M Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: a flexible trimmer for illumina sequence data. *Bioinformatics*, 30(15):2114–2120, 2014.

[47] WA Bonner, HR Hulett, RG Sweet, and LA Herzenberg. Fluorescence activated cell sorting. *Review of Scientific Instruments*, 43(3):404–409, 1972.

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

[49] Ryan R Brinkman, Nima Aghaeepour, Greg Finak, Raphael Gottardo, Tim Mosmann, and Richard H Scheuermann. Automated analysis of flow cytometry data comes of age. *Cytometry Part A*, 89(1):13–15, 2016.

[50] Robert V Bruggner, Bernd Bodenmiller, David L Dill, Robert J Tibshirani, and Garry P Nolan. Automated identification of stratifying signatures in cellular subpopulations. *Proceedings of the National Academy of Sciences*, 111(26):E2770–e2777, 2014.

[51] Kenneth Bryan, Marta Terrile, Isabella M. Bray, Raquel Domingo-FernandÃ©z, Karen M. Watters, Jan Koster, Rogier Versteeg, and Raymond L. Stallings. Discovery and visualization of mirna-mrna functional modules within integrated data using bicluster analysis. *Nucleic Acids Research*, 42(3), 2014.

[52] Joseph C Burns, Michael C Kelly, Michael Hoa, Robert J Morell, and Matthew W Kelley. Single-cell rna-seq resolves cellular complexity in sensory organs from the neonatal inner ear. *Nature communications*, 6:8557, 2015.

[53] Maren Buttner, Zhichao Miao, Alexander Wolf, Sarah A Teichmann, and Fabian J Theis. Assessment of batch-correction methods for scrna-seq data with a new test metric. *bioRxiv*, page 200345, 2017.

[54] Francesco Camastra. Data dimensionality estimation methods: a survey. *Pattern recognition*, 36(12):2945–2954, 2003.

[55] John N Campbell, Evan Z Macosko, Henning Fenselau, Tune H Pers, Anna Lyubetskaya, Danielle Tenen, Melissa Goldman, Anne MJ Verstegen, Jon M Resch, Steven A McCarroll, et al. A molecular census of arcuate hypothalamus and median eminence cell types. *Nature neuroscience*, 20(3):484, 2017.

[56] Cristian I. Castillo-Davis and Daniel L. Hartl. Genemerge post-genomic analysis, data mining, and hypothesis testing. *Bioinformatics*, 19(7):891–892, 2003.

[57] Pratip K Chattopadhyay. Toward 40+ parameter fluorescence flow cytometry. In *Cyto2014*. Congress Int. Soc. Advancement Cytom, 2014.

[58] Pratip K Chattopadhyay, Carl-Magnus Hogerkorp, and Mario Roederer. A chromatic explosion: the development and future of multiparameter flow cytometry. *Immunology*, 125(4):441–449, 2008.

[59] Jeff Cheeger. A lower bound for the smallest eigenvalue of the laplacian. In *Proceedings of the Princeton conference in honor of Professor S. Bochner*, 1969.

[60] Thierry Chekouo and Alejandro Murua. The penalized biclustering model and related algorithms. *Journal of Applied Statistics*, 42(6):1255–1277, 2015.

[61] Thierry Chekouo, Alejandro Murua, and Wolfgang Raffelsberger. The gibbs-plaid biclustering model. *The Annals of Applied Statistics*, 9(3):1643–1670, 2015.

[62] Guanhua Chen, Patrick F Sullivan, and Michael Rene Kosorok. Biclustering with heterogeneous variance. *Proceedings of the National Academy of Sciences of the United States of America*, 110(30):12253–12258, 2013.

[63] Hao Chen, Mai Chan Lau, Michael Thomas Wong, Evan W Newell, Michael Poidinger, and Jinmiao Chen. Cytofkit: A bioconductor package for an integrated mass cytometry data analysis pipeline. *PLoS Comput Biol*, 12(9):e1005112, 2016.

[64] Renchao Chen, Xiaoji Wu, Lan Jiang, and Yi Zhang. Single-cell rna-seq reveals hypothalamic cell diversity. *Cell Reports*, 18(13):3227–3241, 2017.

[65] Tiffany J Chen and Nikesh Kotecha. Cytobank: providing an analytics platform for community cytometry data analysis and collaboration. In *High-Dimensional Single Cell Analysis*, pages 127–157. Springer, 2014.

[66] Xiaoyi Chen, Milena Hasan, Valentina Libri, Alejandra Urrutia, Benoît Beitz, Vincent Rouilly, Darragh Duffy, Étienne Patin, Bernard Chalmond, Lars Rogge, et al. Automated flow cytometric analysis across large numbers of samples and cell types. *Clinical Immunology*, 157(2):249–260, 2015.

[67] Ye Chen, Ryan D Calvert, Ariful Azad, Bartek Rajwa, James Fleet, Timothy Ratliff, and Alex Pothen. Phenotyping immune cells in tumor and healthy tissue using flow cytometry data. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 73–78, 2018.

[68] Yang Cheng, Michael T Wong, Laurens van der Maaten, and Evan W Newell. Categorical analysis of human t cell heterogeneity with one-dimensional soli-expression by nonlinear stochastic embedding. *The Journal of Immunology*, 196(2):924–932, 2016.

[69] Yizong Cheng and George M. Church. Biclustering of expression data. In *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, volume 8, pages 93–103, 2000.

[70] Leonardo Collado-Torres, Abhinav Nellore, Alyssa C Frazee, Christopher Wilks, Michael I Love, Ben Langmead, Rafael A Irizarry, Jeffrey T Leek, and Andrew E Jaffe. Flexible expressed region analysis for rna-seq with derfinder. *Nucleic acids research*, 45(2):e9–e9, 2016.

[71] Francis S. Collins and Harold Varmus. A new initiative on precision medicine. *The New England Journal of Medicine*, 372(9):793–795, 2015.

[72] Daniel Commenges, Chariff Alkhassim, Raphael Gottardo, Boris Hejblum, and Rodolphe Thiebaut. cytometree: a binary tree algorithm for automatic gating in cytometry analysis. *bioRxiv*, 93(11):335554, 2018.

[73] Daniel Commenges, A Sayyareh, Luc Letenneur, Jeremie Guedj, Avner Bar-Hen, et al. Estimating a difference of kullback–leibler risks using a normalized difference of aic. *The Annals of Applied Statistics*, 2(3):1123–1142, 2008.

[74] Ana Conesa, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew McPherson, Michał Wojciech Szcześniak, Daniel J Gaffney, Laura L Elo, Xuegong Zhang, et al. A survey of best practices for rna-seq data analysis. *Genome biology*, 17(1):13, 2016.

[75] Andrea Cossarizza, Hyun-Dong Chang, Andreas Radbruch, Mübeccel Akdis, Immanuel Andrä, Francesco Annunziato, Petra Bacher, Vincenzo Barnaba, Luca Battistini, Wolfgang M Bauer, et al. Guidelines for the use of flow cytometry and cell sorting in immunological studies. *European journal of immunology*, 47(10):1584–1797, 2017.

[76] Andrew Cron, Cécile Gouttefangeas, Jacob Frelinger, Lin Lin, Satwinder K Singh, Cedrik M Britten, Marij JP Welters, Sjoerd H van der Burg, Mike West, and Cliburn Chan. Hierarchical modeling for rare event detection and cell subset alignment across flow cytometry samples. *PLoS Comput Biol*, 9(7):e1003130, 2013.

[77] William R Crum, Oscar Camara, and Derek LG Hill. Generalized overlap measures for evaluation and validation in medical image analysis. *IEEE transactions on medical imaging*, 25(11):1451–1461, 2006.

[78] Juan Cui, Yunbo Chen, Wen Chi Chou, Liankun Sun, Li Chen, Jian Suo, Zhaohui Ni, Ming Zhang, Xiaoxia Kong, Lisabeth L. Hoffman, Jinsong Kang, Yingying Su, Victor Olman, Darryl Johnson, Daniel W. Tench, I. Jonathan Amster, Ron Orlando, David Puett, Fan Li, and Ying Xu. An integrated transcriptomic and computational analysis for biomarker identification in gastric cancer. *Nucleic Acids Research*, 39(4):1197–1207, 2011.

[79] Manhong Dai, Robert C Thompson, Christopher Maher, Rafael Contreras-Galindo, Mark H Kaplan, David M Markovitz, Gil Omenn, and Fan Meng. Ngsqc: cross-platform quality analysis pipeline for deep sequencing data. In *BMC genomics*, volume 11, page S7. BioMed Central, 2010.

[80] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[81] Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[82] Yue Deng, Feng Bao, Qionghai Dai, Lani F Wu, and Steven J Altschuler. Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. *Nature methods*, 16(4):311–314, 2019.

[83] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. Acm, 2004.

[84] Hao Ding, Michael Sharpnack, Chao Wang, Kun Huang, and Raghu Machiraju. Integrative cancer patient stratification via subspace merging. *Bioinformatics*, 2018.

[85] Jiarui Ding, Sohrab Shah, and Anne Condon. densitycut: an efficient and versatile topological approach for automatic clustering of biological data. *Bioinformatics*, page btw227, 2016.

[86] Wolfgang Dittrich and Wolfgang Gohde. Flow-through chamber for photometers to measure and count particles in a dispersion medium, September 25 1973. US Patent 3,761,187.

[87] Federico Divina and Jesus S. Aguilar-Ruiz. Biclustering of expression data with evolutionary computation. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):590–602, 2006.

[88] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[89] Murat Dundar, Ferit Akova, Halid Z Yerebakan, and Bartek Rajwa. A non-parametric bayesian model for joint cell clustering and cluster matching: identification of anomalous sample phenotypes with random effects. *BMC bioinformatics*, 15(1):314, 2014.

[90] Angelo Duò, Mark D Robinson, and Charlotte Soneson. A systematic performance evaluation of clustering methods for single-cell rna-seq data. *F1000Research*, 7, 2018.

[91] Tarn Duong, Arianna Cowling, Inge Koch, and Matt P Wand. Feature significance for multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 52(9):4225–4242, 2008.

[92] Damodar Reddy Edla, Prasanta K Jana, and IEEE Senior Member. A prototype-based modified dbscan for gene clustering. *Procedia Technology*, 6:485–492, 2012.

[93] Gökcen Eraslan, Lukas M Simon, Maria Mircea, Nikola S Mueller, and Fabian J Theis. Single-cell rna-seq denoising using a deep count autoencoder. *Nature communications*, 10(1):1–14, 2019.

[94] Adam Ertel and Aydin Tozeren. Human and mouse switch-like genes share common transcriptional regulatory mechanisms for bimodality. *BMC genomics*, 9(1):628, 2008.

[95] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[96] Xiunan Fang and Joshua WK Ho. Flowgrid enables fast clustering of very large single-cell rna-seq data. *Bioinformatics*, 38(1):282–283, 2022.

[97] A FastQC et al. Quality control tool for high throughput sequence data, 2015.

[98] Qingyuan Feng, Evgenia V. Dueva, Artem Cherkasov, and Martin Ester. Padme: A deep learning-based framework for drug-target interaction prediction. *arXiv preprint arXiv:1807.09741*, 2018.

[99] Ruiwei Feng, Xiangshang Zheng, Tianxiang Gao, Jintai Chen, Wenzhe Wang, Danny Z Chen, and Jian Wu. Interactive few-shot learning: Limited supervision, better medical image segmentation. *IEEE Transactions on Medical Imaging*, 2021.

[100] Laura Ferrer-Font, Christophe Pellefigues, Johannes U Mayer, Sam J Small, Maria C Jaimes, and Kylie M Price. Panel design and optimization for high-dimensional immunophenotyping assays using spectral flow cytometry. *Current protocols in cytometry*, 92(1):e70, 2020.

[101] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[102] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern recognition*, 41(1):176–190, 2008.

[103] Greg Finak, Ali Bashashati, Ryan Brinkman, and Raphaël Gottardo. Merging mixture components for cell population identification in flow cytometry. *Advances in Bioinformatics*, 2009, 2009.

[104] Greg Finak, Jacob Frelinger, Wenxin Jiang, Evan W Newell, John Ramey, Mark M Davis, Spyros A Kalams, Stephen C De Rosa, and Raphael Gottardo. Opencyto: an open source infrastructure for scalable, robust, reproducible, and automated, end-to-end flow cytometry data analysis. *PLoS Comput Biol*, 10(8):e1003806, 2014.

[105] Greg Finak, Marc Langweiler, Maria Jaimes, Mehrnoush Malek, Jafar Taghiyar, Yael Korin, Khadir Raddassi, Lesley Devine, Gerlinde Obermoser, Marcin L Pekalski, et al. Standardizing flow cytometry immunophenotyping analysis from the human immunophenotyping consortium. *Scientific reports*, 6(1):1–11, 2016.

[106] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. Pmlr, 2017.

[107] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, et al. Pfam: the protein families database. *Nucleic acids research*, 42(D1):D222–d230, 2013.

[108] Kipper Fletez-Brant, Josef Špidlen, Ryan R Brinkman, Mario Roederer, and Pratip K Chattopadhyay. flow clean: Automated identification and removal of fluorescence anomalies in flow cytometry data. *Cytometry Part A*, 89(5):461–471, 2016.

[109] Chris Fraley, Adrian Raftery, and Ron Wehrens. Incremental model-based clustering for large datasets with small clusters. *Journal of Computational and Graphical Statistics*, 14(3):529–546, 2005.

[110] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[111] Donald S Frankel, Robert J Olson, Sheila L Frankel, and Sallie W Chisholm. Use of a neural net computer system for analysis of flow cytometric data of phytoplankton populations. *Cytometry: The Journal of the International Society for Analytical Cytology*, 10(5):540–550, 1989.

[112] David Freedman and Persi Diaconis. On the histogram as a density estimator: L 2 theory. *Probability theory and related fields*, 57(4):453–476, 1981.

[113] William T Freeman and Michal Roth. Orientation histograms for hand gesture recognition. In *International workshop on automatic face and gesture recognition*, volume 12, pages 296–301. IEEE Computer Society, Washington, DC, 1995.

[114] Paul Freulon, Jérémie Bigot, and Boris P Hejblum. Cytopt: Optimal transport with domain adaptation for interpreting flow cytometry data. *arXiv preprint arXiv:2006.09003*, 2020.

[115] Bert E Fristedt and Lawrence F Gray. *A modern approach to probability theory*. Springer Science & Business Media, 2013.

[116] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[117] Fernando García-Alcalde, Konstantin Okonechnikov, José Carbonell, Luis M Cruz, Stefan Götz, Sonia Tarazona, Joaquín Dopazo, Thomas F Meyer, and Ana Conesa. Qualimap: evaluating next-generation sequencing alignment data. *Bioinformatics*, 28(20):2678–2679, 2012.

[118] Paul P Gardner, Jennifer Daub, John G Tate, Eric P Nawrocki, Diana L Kolbe, Stinus Lindgreen, Adam C Wilkinson, Robert D Finn, Sam Griffiths-Jones, Sean R Eddy, et al. Rfam: updates to the rna families database. *Nucleic acids research*, 37(suppl_1):D136–d140, 2008.

[119] Jeff Gauthier, Antony T Vincent, Steve J Charette, and Nicolas Derome. A brief history of bioinformatics. *Briefings in bioinformatics*, 20(6):1981–1996, 2019.

[120] Shufei Ge, Shijia Wang, Yee Whye Teh, Liangliang Wang, and Lloyd Elliott. Random tessellation forests. In *Advances in Neural Information Processing Systems*, pages 9575–9585, 2019.

[121] Yongchao Ge and Stuart C Sealfon. flowpeaks: a fast unsupervised clustering for flow cytometry data via k-means and density peak finding. *Bioinformatics*, 28(15):2052–2058, 2012.

[122] Thomas A Geddes, Taiyun Kim, Lihao Nan, James G Burchfield, Jean YH Yang, Dacheng Tao, and Pengyi Yang. Autoencoder-based cluster ensembles for single-cell rna-seq data analysis. *BMC bioinformatics*, 20(19):1–11, 2019.

[123] Andrew Gelman, Gareth O Roberts, Walter R Gilks, et al. Efficient metropolis jumping rules. *Bayesian statistics*, 5(599-608):42, 1996.

[124] Robert C Gentleman, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, Laurent Gautier, Yongchao Ge, Jeff Gentry, et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, 5(10):R80, 2004.

[125] Georg Kurt Gerber, Robin D Dowell, Tommi S Jaakkola, and David Kenneth Gifford. Automated discovery of functional generality of human gene expression programs. *PLOS Computational Biology*, 3(8), 2005.

[126] Jesse Gillis and Paul Pavlidis. The impact of multifunctional genes on guilt by association analysis. *Plos One*, 6(2), 2011.

[127] Gligorijevic, Malod-Dognin N, and Przulj N. Patient-specific data fusion for cancer stratification and personalised treatment. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, volume 21, pages 321–332, 2016.

[128] Vladimir Gligorijević, Noël Malod-Dognin, and Nataša Pržulj. Fuse: multiple network alignment via data fusion. *Bioinformatics*, 32(8):1195–1203, 2015.

[129] Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, et al. Trinity: reconstructing a full-length transcriptome without a genome from rna-seq data. *Nature biotechnology*, 29(7):644, 2011.

[130] Harvey J Greenberg. Greedy algorithms for minimum spanning tree. *University of Colorado at Denver*, 1998.

[131] Evan Greene, Greg Finak, Leonard A D'Amico, Nina Bhardwaj, Candice D Church, Chihiro Morishima, Nirasha Ramchurren, Janis M Taube, Paul T Nghiem, Martin A Cheever, et al. New interpretable machine-learning method for single-cell data reveals correlates of clinical response to cancer immunotherapy. *Patterns*, 2(12):100372, 2021.

[132] Dominic Grün, Mauro J Muraro, Jean-Charles Boisset, Kay Wiebrands, Anna Lyubimova, Gitanjali Dharmadhikari, Maaike van den Born, Johan van Es, Erik Jansen, Hans Clevers, et al. De novo prediction of stem cell identity using single-cell transcriptome data. *Cell Stem Cell*, 19(2):266–277, 2016.

[133] Romain GuigourÃ¨s, Marc Boulle, and Fabrice Rossi. Discovering patterns in time-varying graphs: A triclustering approach. *Advanced Data Analysis and Classification*, 12(3):509–536, 2018.

[134] Minzhe Guo, Hui Wang, S Steven Potter, Jeffrey A Whitsett, and Yan Xu. Sincera: a pipeline for single-cell rna-seq profiling analysis. *PLoS computational biology*, 11(11):e1004575, 2015.

[135] Mayetri Gupta, Ching-Lung Cheung, Yi-Hsiang Hsu, Serkalem Demissie, L Adrienne Cupples, Douglas P Kiel, and David Karasik. Identification of homogeneous genetic architecture of multiple genetically correlated traits by block clustering of genome-wide associations. *Journal of Bone and Mineral Research*, 26(6):1261–1271, 2011.

[136] Daniel Gusenleitner, Eleanor A. Howe, Stefan Bentink, John Quackenbush, and AedÃ n C. Culhane. ibbig: iterative binary bi-clustering of gene sets. *Bioinformatics*, 28(19):2484–2492, 2012.

[137] Michael P Gustafson, Yi Lin, Mabel Ryder, and Allan B Dietz. Strategies for improving the reporting of human immunophenotypes by flow cytometry. *Journal for immunotherapy of cancer*, 2(1):18, 2014.

[138] David Gutierrez-Aviles, Raul GiraÃ¡ldez, Francisco Javier Gil-Cumbreras, and Cristina Rubio-Escudero. Triq: a new method to evaluate triclusters. *Biodata Mining*, 11(1):15, 2018.

[139] David Gutiérrez-Avilés and Cristina Rubio-Escudero. Msl: a measure to evaluate three-dimensional patterns in gene expression data. *Evolutionary Bioinformatics*, 11:Ebo–s25822, 2015.

[140] David Gutiérrez-Avilés, Cristina Rubio-Escudero, Francisco Martínez-Álvarez, and José C Riquelme. Trigen: A genetic algorithm to mine triclusters in temporal gene expression data. *Neurocomputing*, 132:42–53, 2014.

[141] Julia Handl and Joshua Knowles. An evolutionary approach to multiobjective clustering. *IEEE transactions on Evolutionary Computation*, 11(1):56–76, 2007.

[142] G Hannon. Fastx-toolkit. *FASTQ/A Short-reads Preprocessing Tools*, 2010.

[143] Thomas J Hardcastle and Krystyna A Kelly. bayseq: empirical bayesian methods for identifying differential expression in sequence count data. *BMC bioinformatics*, 11(1):422, 2010.

[144] Alexandra M Harrington, Horatiu Olteanu, and Steven H Kroft. A dissection of the cd45/side scatter "blast gate". *American journal of clinical pathology*, 137(5):800–804, 2012.

[145] John A Hartigan, Pamela M Hartigan, et al. The dip test of unimodality. *The annals of Statistics*, 13(1):70–84, 1985.

[146] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[147] Donald Hedeker, Stephen HC du Toit, Hakan Demirtas, and Robert D Gibbons. A note on marginalization of regression parameters from mixed models of binary outcomes. *Biometrics*, 74(1):354–361, 2018.

[148] Boris P Hejblum, Chariff Alkhassim, Raphael Gottardo, François Caron, Rodolphe Thiébaut, et al. Sequential dirichlet process mixtures of multivariate skew t-distributions for model-based clustering of flow cytometry data. *The Annals of Applied Statistics*, 13(1):638–660, 2019.

[149] Birte Hellwig, Jan G Hengstler, Marcus Schmidt, Mathias C Gehrmann, Wiebke Schormann, and Jörg Rahnenführer. Comparison of scores for bimodality of gene expression distributions and genome-wide evaluation of the prognostic relevance of high-scoring genes. *BMC bioinformatics*, 11(1):276, 2010.

[150] Rui Henriques and Sara C. Madeira. Bicpam: Pattern-based biclustering for biomedical data analysis. *Algorithms for Molecular Biology*, 9:27, 2014.

[151] Rui Henriques and Sara C. Madeira. Bic2pam: constraint-guided biclustering for biological data analysis with domain knowledge. *Algorithms for Molecular Biology*, 11(1):23, 2016.

[152] Rui Henriques and Sara C Madeira. Bsig: evaluating the statistical significance of biclustering solutions. *Data Mining and Knowledge Discovery*, 32(1):124–161, 2018.

[153] Stephanie C. Hicks and Rafael A. Irizarry. When to use quantile normalization. *bioRxiv*, page 12203, 2014.

[154] Alexander Hinneburg and Hans-Henning Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *International symposium on intelligent data analysis*, pages 70–80. Springer, 2007.

[155] Matan Hofree, John P Shen, Hannah Carter, Andrew Gross, and Trey Ideker. Network-based stratification of tumor mutations. *Nature Methods*, 10(11):1108–1115, 2013.

[156] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101:5249–5253, 2004.

[157] Youjin Hu, Qin An, Katherine Sheu, Brandon Trejo, Shuxin Fan, and Ying Guo. Single cell multi-omics technology: Methodology and application. *Frontiers in Cell and Developmental Biology*, 6, 2018.

[158] Zhen Hu and Raj Bhatnagar. Algorithm for discovering low-variance 3-clusters from real-valued datasets. In *2010 IEEE International Conference on Data Mining*, pages 236–245, 2010.

[159] Zicheng Hu, Benjamin S Glicksberg, and Atul J Butte. Robust prediction of clinical outcomes using cytometry data. *Bioinformatics*, 35(7):1197–1203, 2018.

[160] Zicheng Hu, Chethan Jujjavarapu, Jacob J Hughey, Sandra Andorf, Hao-Chih Lee, Pier Federico Gherardini, Matthew H Spitzer, Cristel G Thomas, John Campbell, Patrick Dunn, et al. Metacyto: a tool for automated meta-analysis of mass and flow cytometry data. *Cell reports*, 24(5):1377–1388, 2018.

[161] Tim Hubbard, Daniel Barker, Ewan Birney, Graham Cameron, Yuan Chen, L Clark, Tony Cox, J Cuff, Val Curwen, Thomas Down, et al. The ensembl genome database project. *Nucleic acids research*, 30(1):38–41, 2002.

[162] Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, et al. Interpro: the integrative protein signature database. *Nucleic acids research*, 37(suppl_1):D211–d215, 2008.

[163] Syed Fawad Hussain and Muhammad Ramazan. Biclustering of human cancer microarray data using co-similarity based co-clustering. *Expert Systems With Applications*, 55:520–531, 2016.

[164] Dmitry I. Ignatov, Dmitry V. Gnatyshak, Sergei O. Kuznetsov, and Boris G. Mirkin. Triadic formal concept analysis and triclustering: searching for optimal patterns. *Machine Learning*, 101(1):271–302, 2015.

[165] Jan Ihmels, Sven Bergmann, and Naama Barkai. Defining transcription modules using large-scale gene expression data. *Bioinformatics*, 20(13):1993–2003, 2004.

[166] Jonathan Michael Irish. Beyond the age of cellular discovery. *Nature immunology*, 15(12):1095–1097, 2014.

[167] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[168] Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *Machine vision*, volume 5. McGraw-Hill New York, 1995.

[169] Charles A Janeway, Paul Travers, Mark Walport, and Mark J Shlomchik. *Immunobiology: the immune system in health and disease*, volume 1. Current Biology Singapore, 1997.

[170] Disi Ji, Eric Nalisnick, and Padhraic Smyth. Mondrian processes for flow cytometry analysis. *arXiv preprint arXiv:1711.07673*, 2017.

[171] Disi Ji, Preston Putzel, Yu Qian, Ivan Chang, Aishwarya Mandava, Richard H Scheuermann, Jack D Bui, Huan-You Wang, and Padhraic Smyth. Machine learning of discriminative gate locations for clinical diagnosis. *Cytometry Part A*, 97(3):296–307, 2020.

[172] Zhicheng Ji and Hong Kai Ji. Tscan: Pseudo-time reconstruction and evaluation in single-cell rna-seq analysis. *Nucleic Acids Research*, 44(13), 2016.

[173] Haoliang Jiang, Shuigeng Zhou, Jihong Guan, and Ying Zheng. gtricluster: A more general and effective 3d clustering algorithm for gene-sample-time microarray data. In *International Workshop on Data Mining for Biomedical Applications*, pages 48–59, 2006.

[174] Lan Jiang, Huidong Chen, Luca Pinello, and Guo-Cheng Yuan. Giniclust: detecting rare cell types from single-cell gene expression data with gini index. *Genome biology*, 17(1):144, 2016.

[175] Matthew B Johnson and Christopher A Walsh. Cerebral cortical neuron diversity and development at single-cell resolution. *Current opinion in neurobiology*, 42:9–16, 2017.

[176] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[177] Kerstin Johnsson, Jonas Wallin, and Magnus Fontes. Bayesflow: latent modeling of flow cytometry cell populations. *BMC bioinformatics*, 17(1):25, 2016.

[178] Tulika Kakati, Hasin A. Ahmed, Dhruba K. Bhattacharyya, and Jugal K. Kalita. A fast gene expression analysis using parallel biclustering and distributed triclustering approach. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, page 122, 2016.

[179] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[180] Donna Karolchik, Robert Baertsch, Mark Diekhans, Terrence S Furey, Angie Hinrichs, YT Lu, Krishna M Roskin, Matthias Schwartz, Charles W Sugnet, Daryl J Thomas, et al. The ucsc genome browser database. *Nucleic acids research*, 31(1):51–54, 2003.

[181] Michael Steinbach George Karypis, Vipin Kumar, and Michael Steinbach. A comparison of document clustering techniques. In *TextMining Workshop at KDD2000 (May 2000)*, 2000.

[182] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[183] Sahand Khakabimamaghani and Martin Ester. Bayesian biclustering for patient stratification. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, volume 21, pages 345–356. World Scientific, 2016.

[184] Sahand Khakabimamaghani, Yogeshwar Kelkar, Bruno Grande, Ryan Morin, Martin Ester, and Daniel Ziemek. Substra: Supervised bayesian patient stratification. *BioRxiv*, page 538512, 2019.

[185] Seyoung Kim and Padhraic Smyth. Hierarchical dirichlet processes with random effects. In *Advances in Neural Information Processing Systems*, pages 697–704, 2007.

[186] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[187] Kristina Kirschner, Tamir Chandra, Vladimir Kiselev, David Flores-Santa Cruz, Iain C Macaulay, Hyun Jun Park, Juan Li, David G Kent, Rupa Kumar, Dean C Pask, et al. Proliferation drives aging-related functional decline in a subpopulation of the hematopoietic stem cell compartment. *Cell reports*, 19(8):1503–1511, 2017.

[188] Vladimir Yu Kiselev, Tallulah S. Andrews, and Martin Hemberg. Challenges in unsupervised clustering of single-cell rna-seq data. *Nature Reviews Genetics*, page 1, 2019.

[189] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, and Martin Hemberg. Sc3: consensus clustering of single-cell rna-seq data. *Nature Methods*, 14(5):483–486, 2017.

[190] Josef Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1(1):37–42, 1983.

[191] Kevin H Knuth. Optimal data-based binning for histograms. *arXiv preprint physics/0605197*, 2006.

[192] Aleksandra A. Kolodziejczyk, Jong Kyoung Kim, Jason C.H. Tsang, Tomislav Ilicic, Johan Henriksson, Kedar N. Natarajan, Alex C. Tuck, Xuefei Gao, Marc BÃ¼hler, Pentao Liu, John C. Marioni, and Sarah A. Teichmann. Single cell rna-sequencing of pluripotent states unlocks modular transcriptional variation. *Cell Stem Cell*, 17(4):471–485, 2015.

[193] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pages 315–322, 2002.

[194] Warren L. G. Koontz, Patrenahalli M. Narendra, and Keinosuke Fukunaga. A graph-theoretic approach to nonparametric cluster analysis. *IEEE Transactions on Computers*, (9):936–944, 1976.

[195] Ana Kozomara and Sam Griffiths-Jones. mirbase: annotating high confidence micrornas using deep sequencing data. *Nucleic acids research*, 42(D1):D68–d73, 2013.

[196] Miroslav Kratochvíl, Oliver Hunewald, Laurent Heirendt, Vasco Verissimo, Jiří Vondrášek, Venkata P Satagopam, Reinhard Schneider, Christophe Trefois, and Markus Ollert. Gigasom. jl: High-performance clustering and visualization of huge cytometry datasets. *GigaScience*, 9(11):giaa127, 2020.

[197] Sabine Krolak-Schwerdt, Peter Orlik, and Bernhard Ganter. Tripat: a model for analyzing three-mode binary data. In *Information Systems and Data Analysis*, pages 298–307. Springer, 1994.

[198] Monika Krzak, Yordan Raykov, Alexis Boukouvalas, Luisa Cutillo, and Claudia Angelini. Benchmark and parameter sensitivity analysis of single-cell rna sequencing clustering methods. *Frontiers in genetics*, page 1253, 2019.

[199] Kimberly R. Kukurba and Stephen B. Montgomery. Rna sequencing and analysis. *CSH Protocols*, 2015(11):951–969, 2015.

[200] Ariel Kulik and Hadas Shachnai. On lagrangian relaxation and subset selection problems. In *International Workshop on Approximation and Online Algorithms*, pages 160–173. Springer, 2008.

[201] Pia Kvistborg, Cécile Gouttefangeas, Nima Aghaeepour, Angelica Cazaly, Pratip K Chattopadhyay, Cliburn Chan, Judith Eckl, Greg Finak, Sine Reker Hadrup, Holden T Maecker, et al. Thinking outside the gate: single-cell assessments in multiple dimensions. *Immunity*, 42(4):591, 2015.

[202] Justin Lakkis, David Wang, Yuanchao Zhang, Gang Hu, Kui Wang, Huize Pan, Lyle Ungar, Muredach P Reilly, Xiangjie Li, and Mingyao Li. A joint deep learning model enables simultaneous batch effect correction, denoising, and clustering in single-cell transcriptomics. *Genome research*, 31(10):1753–1766, 2021.

[203] Gert RG Lanckriet, Tijl De Bie, Nello Cristianini, Michael I Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[204] Pedro Larranaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Inaki Inza, José A Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez, et al. Machine learning in bioinformatics. *Briefings in bioinformatics*, 7(1):86–112, 2006.

[205] Gyemin Lee, William Finn, and Clayton Scott. Statistical file matching of flow cytometry data. *Journal of biomedical informatics*, 44(4):663–676, 2011.

[206] Hao-Chih Lee, Roman Kosoy, Christine E Becker, Joel T Dudley, and Brian A Kidd. Automated cell type discovery and classification through knowledge transfer. *Bioinformatics*, 33(11):1689–1695, 2017.

[207] Jimmy Tsz Hang Lee and Martin Hemberg. Supervised clustering for single-cell analysis. *Nature methods*, 16(10):965–966, 2019.

[208] Brian D. Lehmann, Joshua A. Bauer, Xi Chen, Melinda E. Sanders, A. Bapsi Chakravarthy, Yu Shyr, and Jennifer A. Pietenpol. Identification of human triple-negative breast cancer subtypes and preclinical models for selection of targeted therapies. *Journal of Clinical Investigation*, 121(7):2750–2767, 2011.

[209] Jacob H Levine, Erin F Simonds, Sean C Bendall, Kara L Davis, D Amir El-ad, Michelle D Tadmor, Oren Litvin, Harris G Fienberg, Astraea Jager, Eli R Zunder, et al. Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis. *Cell*, 162(1):184–197, 2015.

[210] Chang-Lin Li, Kai-Cheng Li, Dan Wu, Yan Chen, Hao Luo, Jing-Rong Zhao, Sa-Shuang Wang, Ming-Ming Sun, Ying-Jin Lu, Yan-Qing Zhong, Xu-Ye Hu, Rui Hou, Bei-Bei Zhou, Lan Bao, Hua-Sheng Xiao, and Xu Zhang. Somatosensory neuron types identified by high-coverage single-cell rna-sequencing and functional heterogeneity. *Cell Research*, 26(1):83–102, 2016.

[211] Huamin Li, Uri Shaham, Kelly P Stanton, Yi Yao, Ruth R Montgomery, and Yuval Kluger. Gating mass cytometry data by deep learning. *Bioinformatics*, 33(21):3423–3430, 2017.

[212] Huamin Li, Uri Shaham, Yi Yao, Ruth Montgomery, and Yuval Kluger. Deepcytof: Automated cell classification of mass cytometry data by deep learning and domain adaptation. *bioRxiv*, page 054411, 2016.

[213] Jun Li and Maintainer Jun Li. Package 'poissonseq'. 2012.

[214] Qiyuan Li, Aron C Eklund, Nicolai Juul, Benjamin Haibe-Kains, Christopher T Workman, Andrea L Richardson, Zoltan Szallasi, and Charles Swanton. Minimising immunohistochemical false negative er classification using a complementary 23 gene expression signature of er status. *PloS one*, 5(12):e15031, 2010.

[215] Xiangjie Li, Kui Wang, Yafei Lyu, Huize Pan, Jingxiao Zhang, Dwight Stambolian, Katalin Susztak, Muredach P Reilly, Gang Hu, and Mingyao Li. Deep learning enables accurate clustering with batch effect removal in single-cell rna-seq analysis. *Nature communications*, 11(1):1–14, 2020.

[216] Lin Lin, Greg Finak, Kevin Ushey, Chetan Seshadri, Thomas R Hawn, Nicole Frahm, Thomas J Scriba, Hassan Mahomed, Willem Hanekom, Pierre-Alexandre Bart, et al. Compass identifies t-cell subsets correlated with clinical outcomes. *Nature biotechnology*, 33(6):610–616, 2015.

[217] Peijie Lin, Michael Troup, and Joshua W. K. Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome Biology*, 18(1):59, 2017.

[218] Tsung-I Lin. Robust mixture modeling using multivariate skew t distributions. *Statistics and Computing*, 20(3):343–356, 2010.

[219] Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi B. Konuru, Hari Sundaram, and Aisling Kelliher. Metafac: community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 527–536, 2009.

[220] Jinze Liu and Wei Wang. Op-cluster: clustering by tendency in high dimensional space. In *Third IEEE International Conference on Data Mining*, pages 187–194, 2003.

[221] Junwan Liu, Zhoujun Li, Xiaohua Hu, and Yiming Chen. Multi-objective evolutionary algorithm for mining 3d clusters in gene-sample-time microarray data. In *2008 IEEE International Conference on Granular Computing*, pages 442–447, 2008.

[222] Yiyi Liu, Quanquan Gu, Jack P Hou, Jiawei Han, and Jian Ma. A network-assisted co-clustering algorithm to discover cancer subtypes based on gene expression. *BMC Bioinformatics*, 15(1):37–37, 2014.

[223] Kenneth Lo, Ryan Remy Brinkman, and Raphael Gottardo. Automated gating of flow cytometry data via robust model-based clustering. *Cytometry Part A*, 73(4):321–332, 2008.

[224] Michael I Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550, 2014.

[225] P Y Lum, G Singh, A Lehman, T Ishkanov, Mikael Vejdemo-Johansson, M Alagappan, J Carlsson, and G Carlsson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3(1):1236–1236, 2013.

[226] Aaron TL Lun, Arianne C Richard, and John C Marioni. Testing for differential abundance in mass cytometry data. *Nature methods*, 14(7):707, 2017.

[227] Markus Lux, Ryan Remy Brinkman, Cedric Chauve, Adam Laing, Anna Lorenc, Lucie Abeler-Dörner, Barbara Hammer, and Jonathan Wren. flowlearn: Fast and precise identification and quality checking of cell populations in flow cytometry. *Bioinformatics*, 1(13):9, 2018.

[228] Jun Ma, Jianan Chen, Matthew Ng, Rui Huang, Yu Li, Chen Li, Xiaoping Yang, and Anne L. Martel. Loss odyssey in medical image segmentation. *Medical Image Analysis*, 71:102035, 2021.

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

[230] Ons Maatouk, Wassim Ayadi, Hend Bouziri, and BÃ©atrice Duval. Evolutionary biclustering algorithms: an experimental study on microarray data. *soft computing*, pages 1–27, 2018.

[231] Ons Maatouk, Wassim Ayadi, Hend Bouziri, and Beatrice Duval. Local search method based on biological knowledge for the biclustering of gene expression data. *Advances in Smart Systems Research*, 6(2):65, 2012.

[232] Iain C Macaulay, Christopher Ponting, and Thierry Voet. Single-cell multiomics: Multiple measurements from single cells. *Trends in Genetics*, 33(2):155–168, 2017.

[233] James W Macdonald and Debashis Ghosh. Copa–cancer outlier profile analysis. *Bioinformatics*, 22(23):2950–2951, 2006.

[234] Evan Z. Macosko, Anindita Basu, Rahul Satija, James Nemesh, Karthik Shekhar, Melissa Goldman, Itay Tirosh, Allison R. Bialas, Nolan Kamitaki, Emily M. Martersteck, John J. Trombetta, David A. Weitz, Joshua R. Sanes, Alex K. Shalek, Aviv Regev, and Steven A. McCarroll. Highly parallel genome-wide expression profiling of individual cells using nanoliter droplets. *Cell*, 161(5):1202–1214, 2015.

[235] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 1(1):24–45, 2004.

[236] Sara C. Madeira, Miguel C. Teixeira, Isabel SÃ¡-Correia, and Arlindo L. Oliveira. Identification of regulatory modules in time series gene expression data using a linear time biclustering algorithm. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(1):153–165, 2010.

[237] Holden T Maecker, Aline Rinfret, Patricia D'Souza, Janice Darden, Eva Roig, Claire Landry, Peter Hayes, Josephine Birungi, Omu Anzala, Miguel Garcia, et al. Standardization of cytokine flow cytometry assays. *BMC immunology*, 6(1):13, 2005.

[238] P Mahanta, HA Ahmed, DK Bhattacharyya, and Jugal K Kalita. Triclustering in gene expression data analysis: a selected survey. In *2011 2nd National Conference on Emerging Trends and Applications in Computer Science*, pages 1–6. Ieee, 2011.

[239] Florian Mair, Felix J Hartmann, Dunja Mrdjen, Vinko Tosevski, Carsten Krieg, and Burkhard Becher. The end of gating? an introduction to automated analysis of high dimensional cytometry data. *European journal of immunology*, 46(1):34–43, 2016.

[240] Mehrnoush Malek, Mohammad Jafar Taghiyar, Lauren Chong, Greg Finak, Raphael Gottardo, and Ryan R Brinkman. flowdensity: reproducing manual gating of flow cytometry data by automated density-based cell population identification. *Bioinformatics*, 31(4):606–607, 2015.

[241] Arian Maleki and Andrea Montanari. Analysis of approximate message passing algorithm. In *2010 44th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–7. Ieee, 2010.

[242] Noel Malod-Dognin, Julia Petschnigg, and Natasa Przulj. Precision medicine - a promising, yet challenging road lies ahead. *Current Opinion in Systems Biology*, 7:1–7, 2018.

[243] Shawn Mankad and George Michailidis. Biclustering three-dimensional data arrays with plaid models. *Journal of Computational and Graphical Statistics*, 23(4):943–965, 2014.

[244] Vivien Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–260, 2013.

[245] Hiroko Masuda, Keith A. Baggerly, Ying Wang, Ya Zhang, Ana Maria Gonzalez-Angulo, Funda Meric-Bernstam, Vicente Valero, Brian D. Lehmann, Jennifer A. Pietenpol, Gabriel N. Hortobagyi, W. Fraser Symmans, and Naoto T. Ueno. Differential response to neoadjuvant chemotherapy among 7 triple-negative breast cancer molecular subtypes. *Clinical Cancer Research*, 19(19):5533–5540, 2013.

[246] Maya B Mathur and Tyler J VanderWeele. New metrics for meta-analyses of heterogeneous effects. *Statistics in Medicine*, 38(8):1336–1342, 2019.

[247] Elie Maza, Pierre Frasse, Pavel Senin, Mondher Bouzayen, and Mohamed Zouine. Comparison of normalization methods for differential gene expression analysis in rna-seq experiments: a matter of relative size of studied transcriptomes. *Communicative & integrative biology*, 6(6):e25849, 2013.

[248] Ciaran McCreesh, Patrick Prosser, Kyle Simpson, and James Trimble. On maximum weight clique algorithms, and how they are evaluated. In *International Conference on Principles and Practice of Constraint Programming*, pages 206–225. Springer, 2017.

[249] Leland McInnes and John Healy. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[250] Katherine M McKinnon. Flow cytometry: an overview. *Current protocols in immunology*, 120(1):5–1, 2018.

[251] Ignacio Medina, Jose Carbonell, Luis Pulido, Sara C. Madeira, Stefan Goetz, Ana Conesa, Joaquin TÃ¡rraga, Alberto Pascual-Montano, Ruben Nogales-Cadenas, Javier Santoyo, Francisco Garcia, Martina MarbÃ , David Montaner, and Joaquin Dopazo. Babelomics: an integrative platform for the analysis of transcriptomics, proteomics and genomic data with advanced functional profiling. *Nucleic Acids Research*, 38:210–213, 2010.

[252] Rajiv Mehrotra, Kameswara Rao Namuduri, and Nagarajan Ranganathan. Gabor filter-based edge detection. *Pattern recognition*, 25(12):1479–1494, 1992.

[253] Jia Meng, Shou Jiang Gao, and Yufei Huang. Enrichment constrained time-dependent clustering analysis for finding meaningful temporal transcription modules. *Bioinformatics*, 25(12):1521–1527, 2009.

[254] Vilas Menon. Clustering single cells: a review of approaches on high-and low-depth single-cell rna-seq data. *Briefings in Functional Genomics*, 17(4):240–245, 2018.

[255] Sushmita Mitra and Haider Banka. Multi-objective evolutionary biclustering of gene expression data. *Pattern Recognition*, 39(12):2464–2477, 2006.

[256] Shin-ichi Miyashita, Alexander S Groombridge, Shin-ichiro Fujii, Ayumi Minoda, Akiko Takatsu, Akiharu Hioki, Koichi Chiba, and Kazumi Inagaki. Highly efficient single-cell analysis of microbial cells by time-resolved inductively coupled plasma mass spectrometry. *Journal of Analytical Atomic Spectrometry*, 29(9):1598–1606, 2014.

[257] Gabriela Moise and Jörg Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008, pages 533–541, 2008.

[258] Reuben Moncada, Florian Wagner, Marta Chiodin, Joseph C. Devlin, Maayan Baron, Cristina H. Hajdu, Diane Simeone, and Itai Yanai. Building a tumor atlas: integrating single-cell rna-seq data with spatial transcriptomics in pancreatic ductal adenocarcinoma. *bioRxiv*, page 254375, 2018.

[259] Aanchal Mongia, Debarka Sengupta, and Angshul Majumdar. deepmc: deep matrix completion for imputation of single-cell rna-seq data. *Journal of Computational Biology*, 27(7):1011–1019, 2020.

[260] Laura Moody, Suparna Mantha, Hong Chen, and Yuan-Xiang Pan. Computational methods to identify bimodal gene expression and facilitate personalized treatment in cancer patients. *Journal of Biomedical Informatics: X*, page 100001, 2018.

[261] Sara Mostafavi, Alexis Battle, Xiaowei Zhu, Alexander E. Urban, Douglas Levinson, Stephen B. Montgomery, and Daphne Koller. Normalizing rna-sequencing data by modeling hidden covariates with prior knowledge. *Plos One*, 8(7), 2013.

[262] Mauro J Muraro, Gitanjali Dharmadhikari, Dominic Grün, Nathalie Groen, Tim Dielen, Erik Jansen, Leon van Gurp, Marten A Engelse, Francoise Carlotti, Eelco JP de Koning, et al. A single-cell transcriptome atlas of the human pancreas. *Cell systems*, 3(4):385–394, 2016.

[263] Robert F Murphy. Automated identification of subpopulations in flow cytometric list mode data using cluster analysis. *Cytometry: The Journal of the International Society for Analytical Cytology*, 6(4):302–309, 1985.

[264] Elena Nabieva, Kam Jim, Amit Agarwal, Bernard Chazelle, and Mona Singh. Whole-proteome prediction of protein function via graph-theoretic analysis of interaction maps. *intelligent systems in molecular biology*, 21(1):302–310, 2005.

[265] Iftekhar Naim, Suprakash Datta, Jonathan Rebhahn, James S Cavenaugh, Tim R Mosmann, and Gaurav Sharma. Swift–scalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, part 1: Algorithm design. *Cytometry Part A*, 85(5):408–421, 2014.

[266] Francesco Napolitano, Yan Zhao, Vania M Moreira, Roberto Tagliaferri, Juha Kere, Mauro D'Amato, and Dario Greco. Drug repositioning: a machine-learning approach through data integration. *Journal of Cheminformatics*, 5(1):30–30, 2013.

[267] Kieran O'Neill, Nima Aghaeepour, Josef Špidlen, and Ryan Brinkman. Flow cytometry bioinformatics. *PLoS Comput Biol*, 9(12):e1003365, 2013.

[268] Kieran O'Neill, Adrin Jalali, Nima Aghaeepour, Holger Hoos, and Ryan R Brinkman. Enhanced flowtype/rchyoptimyx: a bioconductor pipeline for discovery in high-dimensional cytometry data. *Bioinformatics*, 30(9):1329–1330, 2014.

[269] Darya Y Orlova, Leonore A Herzenberg, and Guenther Walther. Science not art: statistically sound methods for identifying subsets in multi-dimensional flow and mass cytometry data sets. *Nature Reviews Immunology*, 18(1):77, 2018.

[270] Martin Oti, Berend Snel, Martijn A Huynen, and Han G Brunner. Predicting disease genes using proteinâ€“protein interactions. *Journal of Medical Genetics*, 43(8):691–698, 2006.

[271] Fang Ou, Cushla McGoverin, Simon Swift, and Frederique Vanholsbeeck. Absolute bacterial cell enumeration using flow cytometry. *Journal of applied microbiology*, 123(2):464–477, 2017.

[272] Shristi Pandey, Karthik Shekhar, Aviv Regev, and Alexander F. Schier. Comprehensive identification and spatial mapping of habenular neuronal types using single-cell rna-seq. *Current Biology*, 28(7), 2018.

[273] Enea Parimbelli, Simone Marini, Lucia Sacchi, and Riccardo Bellazzi. Patient similarity for precision medicine: A systematic review. *Journal of Biomedical Informatics*, 83:87–96, 2018.

[274] Lily M Park, Joanne Lannigan, and Maria C Jaimes. Omip-069: forty-color full spectrum flow cytometry panel for deep immunophenotyping of major cell subsets in human peripheral blood. *Cytometry Part A*, 97(10):1044–1051, 2020.

[275] David R Parks, Mario Roederer, and Wayne A Moore. A new "logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry Part A*, 69(6):541–551, 2006.

[276] Paul Pavlidis, Jason Weston, Jinsong Cai, and William Stafford Noble. Learning gene functional classifications from multiple data types. *Journal of computational biology*, 9(2):401–411, 2002.

[277] Iwona Pawlikowska, Gang Wu, Michael Edmonson, Zhifa Liu, Tanja Gruber, Jinghui Zhang, and Stan Pounds. The most informative spacing test effectively discovers biologically relevant outliers or multiple modes in expression. *Bioinformatics*, 30(10):1400–1408, 2014.

[278] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[279] Carlos E Pedreira, Elaine S Costa, Susana Barrena, Quentin Lecrevisse, Julia Almeida, Jacques JM van Dongen, and Alberto Orfao. Generation of flow cytometry data files with a potentially infinite number of dimensions. *Cytometry Part A: The Journal of the International Society for Analytical Cytology*, 73(9):834–846, 2008.

[280] Jiajie Peng, Xiaoyu Wang, and Xuequn Shang. Combining gene ontology with deep neural networks to enhance the clustering of single cell rna-seq data. *BMC bioinformatics*, 20(8):1–12, 2019.

[281] Laura S Peterson, Ina A Stelzer, Amy S Tsai, Mohammad S Ghaemi, Xiaoyuan Han, Kazuo Ando, Virginia D Winn, Nadine R Martinez, Kevin Contrepois, Mira N Moufarrej, et al. Multiomic immune clockworks of pregnancy. In *Seminars in Immunopathology*, pages 1–16. Springer, 2020.

[282] Alan Pomerantz, Sergio Rodríguez-Rodríguez, Roberta Demichelis-Gómez, Georgina Barrera-Lumbreras, Olga V Barrales-Benítez, María José Díaz-Huízar, Monica Goldberg-Murow, Xavier López-Karpovitch, and Álvaro Aguayo. Importance of cd117 in the assignation of a myeloid lineage in acute leukemias. *Archives of Medical Research*, 48(2):212–215, 2017.

[283] Beatriz Pontes, Raul GirÃ¡ldez, and Jesus S Aguilar-Ruiz. Configurable pattern-based evolutionary biclustering of gene expression data. *Algorithms for Molecular Biology*, 8(1):4–4, 2013.

[284] Saumyadipta Pyne, Xinli Hu, Kui Wang, Elizabeth Rossin, Tsung-I Lin, Lisa M Maier, Clare Baecher-Allan, Geoffrey J McLachlan, Pablo Tamayo, David A Hafler, et al. Automated high-dimensional flow cytometric data analysis. *Proceedings of the National Academy of Sciences*, 106(21):8519–8524, 2009.

[285] Yu Qian, Chungwen Wei, F Eun-Hyung Lee, John Campbell, Jessica Halliley, Jamie A Lee, Jennifer Cai, Y Megan Kong, Eva Sadat, Elizabeth Thomson, et al. Elucidation of seventeen human peripheral blood b-cell subsets and quantification of the tetanus response using a density-based method for the automated identification of cell populations in multidimensional flow cytometry data. *Cytometry Part B: Clinical Cytometry*, 78(S1):S69–s82, 2010.

[286] Peng Qiu, Erin F Simonds, Sean C Bendall, Kenneth D Gibbs Jr, Robert V Bruggner, Michael D Linderman, Karen Sachs, Garry P Nolan, and Sylvia K Plevritis. Extracting a cellular hierarchy from high-dimensional cytometry data with spade. *Nature biotechnology*, 29(10):886–891, 2011.

[287] Xiaojie Qiu, Andrew Hill, Jonathan Packer, Dejun Lin, Yi-An Ma, and Cole Trapnell. Single-cell mrna quantification and differential analysis with census. *Nature methods*, 14(3):309, 2017.

[288] Xiaojie Qiu, Qi Mao, Ying Tang, Li Wang, Raghav Chawla, Hannah A Pliner, and Cole Trapnell. Reversed graph embedding resolves complex single-cell trajectories. *Nature methods*, 14(10):979, 2017.

[289] Lynn Quek, Georg W Otto, Catherine Garnett, Ludovic Lhermitte, Dimitris Karamitros, Bilyana Stoilova, I-Jun Lau, Jessica Doondeea, Batchimeg Usukhbayar, Alison Kennedy, et al. Genetically distinct leukemic stem cells in human cd34- acute myeloid leukemia are arrested at a hemopoietic precursor-like stage. *Journal of Experimental Medicine*, 213(8):1513–1535, 2016.

[290] Albina Rahim, Justin Meskas, Sibyl Drissler, Alice Yue, Anna Lorenc, Adam Laing, Namita Saran, Jacqui White, Lucie Abeler-Dörner, Adrian Hayday, et al. High throughput automated analysis of big flow cytometry data. *Methods*, 134:164–176, 2017.

[291] Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alyosha Efros, and Sergey Levine. Conditional networks for few-shot semantic segmentation. 2018.

[292] Jiahua Rao, Xiang Zhou, Yutong Lu, Huiying Zhao, and Yuedong Yang. Imputing single-cell rna-seq data by combining graph convolution and autoencoder neural networks. *Iscience*, 24(5):102393, 2021.

[293] Franck Rapaport, Raya Khanin, Yupu Liang, Mono Pirun, Azra Krek, Paul Zumbo, Christopher E Mason, Nicholas D Socci, and Doron Betel. Comprehensive evaluation of differential gene expression analysis methods for rna-seq data. *Genome biology*, 14(9):3158, 2013.

[294] Carl Edward Rasmussen. The infinite gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

[295] Davide Risso, John Ngai, Terence P Speed, and Sandrine Dudoit. Normalization of rna-seq data using factor analysis of control genes or samples. *Nature Biotechnology*, 32(9):896–902, 2014.

[296] Marylyn D Ritchie. Bioinformatics approaches for detecting gene–gene and gene–environment interactions in studies of human disease. *Neurosurgical focus*, 19(4):1–4, 2005.

[297] Mark D Robinson, Davis J McCarthy, and Gordon K Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010.

[298] Mark D Robinson and Alicia Oshlack. A scaling normalization method for differential expression analysis of rna-seq data. *Genome biology*, 11(3):R25, 2010.

[299] Domingo S. Rodriguez-Baena, Antonio J. Perez-Pulido, and Jesus S. Aguilar-Ruiz. A biclustering algorithm for extracting bit-patterns from binary datasets. *Bioinformatics*, 27(19):2738–2745, 2011.

[300] Mario Roederer. Compensation in flow cytometry. *Current protocols in cytometry*, 22(1):1–14, 2002.

[301] Wade T Rogers, Allan R Moser, Herbert A Holyst, Andrew Bantly, Emile R Mohler III, George Scangas, and Jonni S Moore. Cytometric fingerprinting: quantitative characterization of multivariate distributions. *Cytometry Part A: the journal of the International Society for Analytical Cytology*, 73(5):430–441, 2008.

[302] Jonathan Ronen and Altuna Akalin. netsmooth: Network-smoothing based imputation for single cell rna-seq. *F1000Research*, 7, 2018.

[303] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[304] James C Ross, Peter J Castaldi, Michael H Cho, Junxiang Chen, Yale Chang, Jennifer G Dy, Edwin K Silverman, George R Washko, and Raúl San José Estépar. A bayesian nonparametric model for disease subtyping: application to emphysema phenotypes. *IEEE transactions on medical imaging*, 36(1):343–354, 2017.

[305] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[306] Peter Rubbens, Ruben Props, Frederiek-Maarten Kerckhof, Nico Boon, and Willem Waegeman. Phenogmm: Gaussian mixture modeling of cytometry data quantifies changes in microbial community structure. *MSphere*, 6(1):e00530–20, 2021.

[307] Yvan Saeys, Sofie Van Gassen, and Bart Lambrecht. Response to orlova et al."science not art: statistically sound methods for identifying subsets in multi-dimensional flow and mass cytometry data sets". *Nature Reviews Immunology*, 18(1):78, 2018.

[308] Yvan Saeys, Sofie Van Gassen, and Bart N Lambrecht. Computational flow cytometry: helping to make sense of high-dimensional immunology data. *Nature Reviews Immunology*, 16(7):449–462, 2016.

[309] RM Sakia. The box-cox transformation technique: a review. *The statistician*, pages 169–178, 1992.

[310] Nikolay Samusik, Zinaida Good, Matthew H Spitzer, Kara L Davis, and Garry P Nolan. Automated mapping of phenotype space with single-cell data. *Nature methods*, 2016.

[311] Rahul Satija, Jeffrey A Farrell, David Gennert, Alexander F Schier, and Aviv Regev. Spatial reconstruction of single-cell gene expression data. *Nature Biotechnology*, 33(5):495–502, 2015.

[312] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[313] Jan Schepers, Iven Van Mechelen, and Eva Ceulemans. Three-mode partitioning. *Computational Statistics & Data Analysis*, 51(3):1623–1642, 2006.

[314] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[315] Benno Schwikowski, Peter Uetz, and Stanley Fields. A network of proteinâ€"protein interactions in yeast. *Nature Biotechnology*, 18(12):1257–1261, 2000.

[316] Anne Senabouth, Samuel Lukowski, Jose Alquicira, Stacey Andersen, Xin Mei, Quan Nguyen, and Joseph Powell. ascend: R package for analysis of single cell rna-seq data. *BioRxiv*, page 207704, 2017.

[317] Fatemeh Seyednasrollah, Asta Laiho, and Laura L Elo. Comparison of software packages for detecting differential expression in rna-seq studies. *Briefings in bioinformatics*, 16(1):59–70, 2013.

[318] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017.

[319] Howard M Shapiro. *Practical flow cytometry*. John Wiley & Sons, 2005.

[320] Karthik Shekhar, Petter Brodin, Mark M Davis, and Arup K Chakraborty. Automatic classification of cellular expression by nonlinear stochastic embedding (accense). *Proceedings of the National Academy of Sciences*, 111(1):202–207, 2014.

[321] Brad T Sherman, Da Wei Huang, Qina Tan, Yongjian Guo, Stephan Bour, David Liu, Robert Stephens, Michael W Baseler, H Clifford Lane, and Richard A Lempicki. David knowledgebase: a gene-centered database integrating heterogeneous gene annotation resources to facilitate high-throughput gene functional analysis. *BMC bioinformatics*, 8(1):426, 2007.

[322] ShiFunan and HuangHaiyan. Identifying cell subpopulations and their genetic drivers from single-cell rna-seq data using a biclustering approach. *Journal of Computational Biology*, 24(7):663–674, 2017.

[323] Robin Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The computer journal*, 16(1):30–34, 1973.

[324] Kelvin Sim, Zeyar Aung, and Vivekanand Gopalkrishnan. Discovering correlated subspace clusters in 3d continuous-valued data. In *2010 IEEE International Conference on Data Mining*, pages 471–480, 2010.

[325] Kelvin Sim, Ghim-Eng Yap, David R Hardoon, Vivekanand Gopalkrishnan, Gao Cong, and Suryani Lukman. Centroid-based actionable 3d subspace clustering. *IEEE transactions on knowledge and data engineering*, 25(6):1213–1226, 2013.

[326] David Sims, Ian Sudbery, Nicholas E Ilott, Andreas Heger, and Chris P Ponting. Sequencing depth and coverage: key considerations in genomic analyses. *Nature Reviews Genetics*, 15(2):121, 2014.

[327] Gordon K Smyth. Limma: linear models for microarray data. In *Bioinformatics and computational biology solutions using R and Bioconductor*, pages 397–420. Springer, 2005.

[328] Robert R Sokal. A statistical method for evaluating systematic relationships. *Univ. Kansas, Sci. Bull.*, 38:1409–1438, 1958.

[329] Charlotte Soneson and Mauro Delorenzi. A comparison of methods for differential expression analysis of rna-seq data. *BMC bioinformatics*, 14(1):91, 2013.

[330] Li Song, Hongbin Ma, Mei Wu, Zilong Zhou, and Mengyin Fu. A brief survey of dimension reduction. In *International Conference on Intelligent Science and Big Data Engineering*, pages 189–200. Springer, 2018.

[331] Michael M Sørensen. New facets and a branch-and-cut algorithm for the weighted clique problem. *European Journal of Operational Research*, 154(1):57–70, 2004.

[332] Till Sörensen, Sabine Baumgart, Pawel Durek, Andreas Grützkau, and Thomas Häupl. immunoclust–an automated analysis pipeline for the identification of immunophenotypic signatures in high-dimensional cytometric datasets. *Cytometry Part A*, 87(7):603–615, 2015.

[333] Josef Spidlen, Aaron Barsky, Karin Breuer, Peter Carr, Marc-Danie Nazaire, Barbara Allen Hill, Yu Qian, Ted Liefeld, Michael Reich, Jill P Mesirov, et al. Genepattern flow cytometry suite. *Source code for biology and medicine*, 8(1):14, 2013.

[334] Josef Spidlen, Karin Breuer, Chad Rosenberg, Nikesh Kotecha, and Ryan R Brinkman. Flowrepository: A resource of annotated flow cytometry datasets associated with peer-reviewed publications. *Cytometry Part A*, 81(9):727–731, 2012.

[335] Josef Spidlen, Wayne Moore, David Parks, Michael Goldberg, Chris Bray, Pierre Bierre, Peter Gorombey, Bill Hyun, Mark Hubbard, Simon Lange, et al. Data file standard for flow cytometry, version fcs 3.1. *Cytometry Part A*, 77(1):97–100, 2010.

[336] Matthew H Spitzer, Pier Federico Gherardini, Gabriela K Fragiadakis, Nupur Bhattacharya, Robert T Yuan, Andrew N Hotson, Rachel Finck, Yaron Carmi, Eli R Zunder, Wendy J Fantl, et al. An interactive reference framework for modeling a dynamic immune system. *Science*, 349(6244):1259425, 2015.

[337] Matthew H Spitzer and Garry P Nolan. Mass cytometry: Single cells, many features. *Cell*, 165(4):780–791, 2016.

[338] Wiwat Sriphum, Gary B Wills, and Nicolas G Green. Floptics: A novel automated gating technique for flow cytometry data. *International Journal of Organizational and Collective Intelligence (IJOCI)*, 12(1):1–21, 2022.

[339] Oliver Stegle, Leopold Parts, Matias Piipari, John Winn, and Richard Durbin. Using probabilistic estimation of expression residuals (peer) to obtain increased power and interpretability of gene expression analyses. *Nature Protocols*, 7(3):500–507, 2012.

[340] Caleb K Stein, Pingping Qu, Joshua Epstein, Amy Buros, Adam Rosenthal, John Crowley, Gareth Morgan, and Bart Barlogie. Removing batch effects from purified plasma cell gene expression microarrays with modified combat. *BMC bioinformatics*, 16(1):63, 2015.

[341] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 240–248. Springer, 2017.

[342] Istvan P Sugar and Stuart C Sealfon. Misty mountain clustering: application to fast unsupervised flow cytometry gating. *BMC bioinformatics*, 11(1):502, 2010.

[343] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412, 2019.

[344] Kim T, Chen Ir, Lin Y, Wang Ay, Yang Jyh, and Yang P. Impact of similarity metrics on single-cell rna-seq data clustering. *Briefings in Bioinformatics*, 2018.

[345] Divyanshu Talwar, Aanchal Mongia, Debarka Sengupta, and Angshul Majumdar. Autoimpute: Autoencoder based imputation of single-cell rna-seq data. *Scientific reports*, 8(1):16329, 2018.

[346] Amos Tanay, Roded Sharan, Martin Kupiec, and Ron Shamir. Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2981–2986, 2004.

[347] Jinhui Tang, Xiangbo Shu, Guo-Jun Qi, Zechao Li, Meng Wang, Shuicheng Yan, and Ramesh Jain. Tri-clustered tensor completion for social-aware image tag refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1662–1674, 2017.

[348] Sonia Tarazona, Fernando García, Alberto Ferrer, Joaquín Dopazo, and Ana Conesa. Noiseq: a rna-seq differential expression method robust for sequencing depth biases. *EMBnet. journal*, 17(B):18–19, 2011.

[349] Sonia Tarazona, Fernando García-Alcalde, Joaquín Dopazo, Alberto Ferrer, and Ana Conesa. Differential expression in rna-seq: a matter of depth. *Genome research*, 21(12):2213–2223, 2011.

[350] Alain B. Tchagang, Sieu Phan, Fazel Famili, Heather Shearer, Pierre R. Fobert, Yi Huang, Jitao Zou, Daiqing Huang, Adrian Cutler, Ziying Liu, and Youlian Pan. Mining biological information from 3d short time-series gene expression data: the optricluster algorithm. *BMC Bioinformatics*, 13(1):54–54, 2012.

[351] Yee W Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392, 2005.

[352] Ben Teng, Can Yang, Jiming Liu, Zhipeng Cai, and Xiang Wan. Exploring the genetic patterns of complex diseases via the integrative genome-wide approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 13(3):557–564, 2016.

[353] Andrew E Teschendorff, Ahmad Miremadi, Sarah E Pinder, Ian O Ellis, and Carlos Caldas. An immune response gene expression module identifies a good prognosis subtype in estrogen receptor negative breast cancer. *Genome biology*, 8(8):R157, 2007.

[354] Andrew E Teschendorff, Ali Naderi, Nuno L Barbosa-Morais, and Carlos Caldas. Pack: Profile analysis using clustering and kurtosis to find molecular classifiers in cancer. *Bioinformatics*, 22(18):2269–2275, 2006.

[355] Paul W Tetteh, Onur Basak, Henner F Farin, Kay Wiebrands, Kai Kretzschmar, Harry Begthel, Maaike van den Born, Jeroen Korving, Frederic De Sauvage, Johan H Van Es, et al. Replacement of lost lgr5-positive stem cells through plasticity of their enterocyte-lineage daughters. *Cell stem cell*, 18(2):203–213, 2016.

[356] Tian Tian, Ji Wan, Qi Song, and Zhi Wei. Clustering single-cell rna-seq data with a model-based deep learning approach. *Nature Machine Intelligence*, 1(4):191–198, 2019.

[357] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

[358] Scott A Tomlins, Daniel R Rhodes, Sven Perner, Saravana M Dhanasekaran, Rohit Mehra, Xiao-Wei Sun, Sooryanarayana Varambally, Xuhong Cao, Joelle Tchinda, Rainer Kuefer, et al. Recurrent fusion of tmprss2 and ets transcription factor genes in prostate cancer. *science*, 310(5748):644–648, 2005.

[359] Dong Ling Tong, Graham R Ball, and A Graham Pockley. gem/gann: A multivariate computational strategy for auto-characterizing relationships between cellular and clinical phenotypes and predicting disease progression time using high-dimensional flow cytometry data. *Cytometry Part A*, 87(7):616–623, 2015.

[360] Pan Tong, Yong Chen, Xiao Su, and Kevin R Coombes. Siber: systematic identification of bimodally expressed genes using rnaseq data. *Bioinformatics*, 29(5):605–613, 2013.

[361] Trang Tran, Cam Chi Nguyen, and Ngoc Minh Hoang. Management and analysis of dna microarray data by using weighted trees. *Journal of Global Optimization*, 39(4):623–645, 2007.

[362] Barbara Treutlein, Doug G. Brownfield, Angela Ruohao Wu, Norma F. Neff, Gary L. Mantalas, F. Hernan Espinoza, Tushar J. Desai, Mark A. Krasnow, and Stephen R. Quake. Reconstructing lineage hierarchies of the distal lung epithelium using single-cell rna-seq. *Nature*, 509(7500):371–375, 2014.

[363] Ewoud De Troyer, Dan Lin, Ziv Shkedy, and Sebastian Kaiser. The xmotif algorithm. 2016.

[364] Jason CH Tsang, Yong Yu, Shannon Burke, Florian Buettner, Cui Wang, Aleksandra A Kolodziejczyk, Sarah A Teichmann, Liming Lu, and Pentao Liu. Single-cell transcriptomic reconstruction reveals cell cycle and multi-lineage differentiation defects in bcl11a-deficient hematopoietic stem cells. *Genome biology*, 16(1):178, 2015.

[365] Jessica Tuengel, Sanya Ranchal, Alexandra Maslova, Gurpreet Aulakh, Maria Papadopoulou, Sibyl Drissler, Bing Cai, Cetare Mohsenzadeh-Green, Hugo Soudeyns, Sara Mostafavi, et al. Characterization of adaptive-like gamma delta t cells in ugandan infants during primary cytomegalovirus infection. *Viruses*, 13(10):1987, 2021.

[366] JJM Van Dongen, L Lhermitte, Stephan Böttcher, Julia Almeida, VHJ Van der Velden, Juan Flores-Montero, A Rawstron, Vahid Asnafi, Quentin Lecrevisse, Paulo Lucio, et al. Euroflow antibody panels for standardized n-dimensional flow cytometric immunophenotyping of normal, reactive and malignant leukocytes. *Leukemia*, 26(9):1908–1975, 2012.

[367] Sofie Van Gassen, Britt Callebaut, Mary J Van Helden, Bart N Lambrecht, Piet Demeester, Tom Dhaene, and Yvan Saeys. Flowsom: Using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A*, 87(7):636–645, 2015.

[368] Sofie Van Gassen, Celine Vens, Tom Dhaene, Bart N Lambrecht, and Yvan Saeys. Floremi: Flow density survival regression using minimal feature redundancy. *Cytometry Part A*, 89(1):22–29, 2016.

[369] Chris P Verschoor, Alina Lelic, Jonathan L Bramson, and Dawn ME Bowdish. An introduction to automated flow cytometry gating tools and their implementation. *Frontiers in immunology*, 6:380, 2015.

[370] Peter Waltman, Thadeous Kacmarczyk, Ashley R Bate, Daniel B Kearns, David J Reiss, Patrick Eichenberger, and Richard Bonneau. Multi-species integrative biclustering. *Genome biology*, 11(9):R96, 2010.

[371] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods*, 11(3):333–337, 2014.

[372] Bo Wang, Junjie Zhu, Emma Pierson, Daniele Ramazzotti, and Serafim Batzoglou. Visualization and analysis of single-cell rna-seq data by kernel-based similarity learning. *Nature Methods*, 14(4):414–416, 2017.

[373] Fei Wang and Jimeng Sun. Survey on distance metric learning and dimensionality reduction in data mining. *Data Mining and Knowledge Discovery*, 29(2):534–564, 2015.

[374] Jiacheng Wang, Quan Zou, and Chen Lin. A comparison of deep learning-based preprocessing and clustering approaches for single-cell rna sequencing data. *Briefings in Bioinformatics*, 23(1):bbab345, 2022.

[375] Jing Wang, Sijin Wen, W Fraser Symmans, Lajos Pusztai, and Kevin R Coombes. The bimodality index: a criterion for discovering and ranking bimodal signatures from cancer gene expression profiling data. *Cancer informatics*, 7:Cin–s2846, 2009.

[376] Juexin Wang, Anjun Ma, Yuzhou Chang, Jianting Gong, Yuexu Jiang, Ren Qi, Cankun Wang, Hongjun Fu, Qin Ma, and Dong Xu. scgnn is a novel graph neural network framework for single-cell rna-seq analyses. *Nature communications*, 12(1):1–11, 2021.

[377] Liguo Wang, Shengqin Wang, and Wei Li. Rseqc: quality control of rna-seq experiments. *Bioinformatics*, 28(16):2184–2185, 2012.

[378] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[379] Yongcui Wang, Shilong Chen, Naiyang Deng, and Yong Wang. Drug repositioning by kernel-based integration of molecular structure, molecular activity, and phenotype data. *Plos One*, 8(11), 2013.

[380] James H Ware, Frederick Mosteller, Fernando Delgado, Christl Donnelly, and Joseph A Ingelfinger. P values. *Medical uses of statistics*, 2:181–200, 1986.

[381] Lukas M Weber, Malgorzata Nowicka, Charlotte Soneson, and Mark D Robinson. diffcyt: Differential discovery in high-dimensional cytometry via high-resolution clustering. *BioRxiv*, page 349738, 2018.

[382] Lukas M Weber, Malgorzata Nowicka, Charlotte Soneson, and Mark D Robinson. diffcyt: Differential discovery in high-dimensional cytometry via high-resolution clustering. *Communications biology*, 2(1):1–11, 2019.

[383] Lukas M Weber and Mark D Robinson. Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. *Cytometry Part A*, 89(12):1084–1096, 2016.

[384] SR Weijers, J De Jonge, O Van Zanten, L Benedetti, J Langeveld, HW Menkveld, and AF Van Nieuwenhuijzen. Kallisto: cost effective and integrated optimization of the urban wastewater system eindhoven. *Water Practice and Technology*, 7(2), 2012.

[385] M Wetzler, BK McElwain, CC Stewart, L Blumenson, A Mortazavi, LA Ford, James L Slack, M Barcos, S Ferrone, and MR Baer. Hla-dr antigen-negative acute myeloid leukemia. *Leukemia*, 17(4):707–715, 2003.

[386] Malcolm F Wilkins, Colin Morris, and Lynne Boddy. A comparison of radial basis function and backpropagation neural networks for identification of marine phytoplankton from multivariate flow cytometry data. *Bioinformatics*, 10(3):285–294, 1994.

[387] Christian Wiwie, Jan Baumbach, and Richard Röttger. Comparing the performance of biomedical clustering methods. *Nature methods*, 12(11):1033, 2015.

[388] Matthias Wödlinger, Michael Reiter, Lisa Weijler, Margarita Maurer-Granofszky, Angela Schumich, and Michael Dworzak. Automated identification of cell populations in flow cytometry data with transformers. *arXiv preprint arXiv:2108.10072*, 2021.

[389] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[390] Juan Xie, Anjun Ma, Anne Fennell, Qin Ma, and Jing Zhao. It is time to apply biclustering: a comprehensive review of biclustering applications in biological and biomedical data. *Briefings in Bioinformatics*, 2018.

[391] Yinlong Xie, Gengxiong Wu, Jingbo Tang, Ruibang Luo, Jordan Patterson, Shanlin Liu, Weihua Huang, Guangzhu He, Shengchang Gu, Shengkang Li, et al. Soapdenovo-trans: de novo transcriptome assembly with short rna-seq reads. *Bioinformatics*, 30(12):1660–1666, 2014.

[392] Xin Xu, Ying Lu, Kian-Lee Tan, and Anthony KH Tung. Finding time-lagged 3d clusters. In *2009 IEEE 25th International Conference on Data Engineering*, pages 445–456. Ieee, 2009.

[393] Yungang Xu, Zhigang Zhang, Lei You, Jiajia Liu, Zhiwei Fan, and Xiaobo Zhou. scigans: single-cell rna-seq imputation using generative adversarial networks. *Nucleic acids research*, 48(15):e85–e85, 2020.

[394] Jiong Yang, Haixun Wang, Wei Wang, and Philip S. Yu. Enhanced biclustering on expression data. In *Third IEEE Symposium on Bioinformatics and Bioengineering, 2003. Proceedings.*, pages 321–327, 2003.

[395] Xingyu Yang and Peng Qiu. Automatically generate two-dimensional gating hierarchy from clustered cytometry data. *Cytometry Part A*, 93(10):1039–1050, 2018.

[396] Yuchen Yang, Ruth Huh, Houston W. Culpepper, Yuan Lin, Michael I. Love, and Yun Li. Safe-clustering: Single-cell aggregated (from ensemble) clustering for single-cell rna-seq data. *Bioinformatics*, page 215723, 2018.

[397] Christopher Yau et al. pcareduce: hierarchical clustering of single cell transcriptional profiles. *BMC bioinformatics*, 17(1):140, 2016.

[398] Alice Yue. *Feature-based Comparison of Flow Cytometry Data.* PhD thesis, Applied Sciences: School of Computing Science, 2017.

[399] Alice Yue and Ryan Brinkman. Automated flow cytometry cell population identification and visualization. *European Journal of Immunology*, 2017.

[400] Alice Yue, Maxwell W Libbrecht, Cedric Chauve, and Ryan R Brinkman. Automated identification of maximal differential cell populations in flow cytometry data. *Cytometry Part A*, 2021.

[401] Taegyun Yun and Gwan-Su Yi. Biclustering for the comprehensive search of correlated gene expression patterns using clustered seed expansion. *BMC Genomics*, 14(1):144–144, 2013.

[402] Maxim Yurov and Dmitry I. Ignatov. Turning krimp into a triclustering technique on sets of attribute-condition pairs that compress. In *International Joint Conference on Rough Sets*, pages 558–569, 2017.

[403] Luke Zappia and Fabian J Theis. Over 1000 tools reveal trends in the single-cell rna-seq analysis landscape. *Genome biology*, 22(1):1–18, 2021.

[404] Habil Zare, Parisa Shooshtari, Arvind Gupta, and Ryan R Brinkman. Data reduction for spectral clustering to analyze high throughput flow cytometry data. *BMC bioinformatics*, 11(1):403, 2010.

[405] Amit Zeisel, Ana B. MuÃ±oz−Manchado, SimoneCodeluppi, PeterLÃűnnerberg, Gioele LaManno, An *cellrna − seq. Science*, 347(6226) : 1138 − −1142, 2015.

[406] Jesse M. Zhang, Jue Fan, H. Christina Fan, David Rosenfeld, and David N. Tse. An interpretable framework for clustering single-cell rna-seq datasets. *BMC Bioinformatics*, 19(1):93, 2018.

[407] Jian Zhang. Generalized plaid models. *Neurocomputing*, 79:95–104, 2012.

[408] Li Zhang, Dan Xu, Anurag Arnab, and Philip HS Torr. Dynamic graph message passing networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3726–3735, 2020.

[409] Ping Zhang, Fei Wang, and Jianying Hu. Towards drug repositioning: a unified computational framework for integrating multiple aspects of drug similarity and disease similarity. In *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, volume 2014, pages 1258–1267, 2014.

[410] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas S Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *IEEE Transactions on Cybernetics*, 50(9):3855–3865, 2020.

[411] Lizhuang Zhao and Mohammed Javeed Zaki. Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 694–705, 2005.

[412] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6881–6890, 2021.

[413] Xiaodong Zheng, Hao Ding, Hiroshi Mamitsuka, and Shanfeng Zhu. Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1025–1033, 2013.

[414] Jiayu Zhou, Fei Wang, Jianying Hu, and Jieping Ye. From micro to macro: data driven phenotyping by densification of longitudinal electronic medical records. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 135–144, 2014.

[415] Jiayu Zhou, Fei Wang, Jianying Hu, and Jieping Ye. From micro to macro: data driven phenotyping by densification of longitudinal electronic medical records. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 135–144, 2014.

Appendix A

Identifying differential cell populations in flow cytometry data accounting for marker frequency

A.1 Proof of correctness for Equation 4.

In this section, we work under the assumption that there exists some pair of marker conditions that are independent of each other.

Theorem A.1.1. *Assuming that there exists some pair of marker condition indices* $\{p, q\}$ *s.t.* $P(v^p)$ *and* $P(v^q)$ *are independent given* $P(v^{1:\ell\setminus\{p,q\}})$*:*

$$P(v^{1:\ell}) = P(v^{1:\ell\setminus p})\frac{P(v^{1:\ell\setminus q})}{P(v^{1:\ell\setminus\{p,q\}})}$$

where P *is the actual proportion of cell population* v *defined by* ℓ *marker conditions, then we can identify such an index pair using the following.*

$$p = \arg\max_{p\in 1:\ell} P(v^{1:\ell\setminus p})$$

$$q = \arg\min_{q\in 1:\ell\setminus p} \frac{P(v^{1:\ell\setminus q})}{P(v^{1:\ell\setminus\{p,q\}})}$$

Proof. In our method, we derived equation 4 by assuming there is some $\{p, q\}$ index pair s.t. $P(v^p)$ is independent of $P(v^q)$ given $P(v^{1:\ell\setminus\{p,q\}})$. For purposes of this proof, we will use an alternative set of notations:

$$\sum_z p(z) = P(v^{1:\ell\setminus\{p,q\}}) + (1 - P(v^{1:\ell\setminus\{p,q\}})) = 1$$

$$\sum_x p(x) = P(v^{1:\ell\setminus p}) + (1 - P(v^{1:\ell\setminus p})) = 1$$

$$\sum_y p(y) = P(v^{1:\ell \setminus q}) + (1 - P(v^{1:\ell \setminus q})) = 1$$

where p is the discrete probability mass function. In our application, each cell is either a part of the our cell population v, or it is not. This means that our probability distributions p are all binary and add up to 1.

First we show the relationship between conditional mutual information and Equation 4.

$$\begin{aligned}
I(X;Y|Z) &= \sum_{x,y,z} p(x,y,z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)} \\
&= \sum_{x,y,z} p(x,y,z) \log \frac{p(x,y,z)p(z)}{p(x,z)p(y,z)} \\
&= \sum_{x,y,z} p(x,y,z) \left(\log p(x,y,z) + \log \frac{p(z)}{p(x,z)p(y,z)} \right) \\
&= \sum_{x,y,z} p(x,y,z) \log p(x,y,z) + \sum_{x,y,z} p(x,y,z) \log \frac{p(z)}{p(x,z)p(y,z)}
\end{aligned}$$

Conditional mutual information $I(X;Y|Z) = 0$ iff X is independent of Y given Z. We assumed such a pair exists. However, even if we are not able to find such X and Y, we also know that conditional mutual information must always non-negative $I(x;Y|Z) \geq 0$. Given these two properties, the following is true.

$$\begin{aligned}
0 &\leq \sum_{x,y,z} p(x,y,z) \log p(x,y,z) + \sum_{x,y,z} p(x,y,z) \log \frac{p(z)}{p(x,z)p(y,z)} \\
-\sum_{x,y,z} p(x,y,z) \log p(x,y,z) &\geq \sum_{x,y,z} p(x,y,z) \log \frac{p(z)}{p(x,z)p(y,z)} \\
\sum_{x,y,z} p(x,y,z) \log p(x,y,z) &\leq \sum_{x,y,z} p(x,y,z) \log \frac{p(x,z)p(y,z)}{p(z)}
\end{aligned}$$

In order to find a X and Y such that they are independent given Z, we must minimize the difference, to 0, between two sides of this inequality. In order to do so, we must choose {p,q} according to Equation 4, concluding our proof.

□

A.2 Algorithm: calculating SpecEnr values for each cell population given proportions.

To reduce runtime, we only directly calculate this expected proportion for cell populations with only positive marker conditions (e.g. A^+B^+, C^+D^+); the expected proportion for the rest of the cell populations can be directly inferred from these cell populations in a later step. This runtime includes operations to first calculate all the arc values in our cell hierarchy. Arc values can be interpreted as the proportion of the cell population an arc points to over the proportion of the cell population an arc originates from. Since there are $\sum_{\ell=0}^{L} \left\{ l \cdot \begin{pmatrix} L \\ \ell \end{pmatrix} \right\} = 2^{L-1}L$ edges, this step takes $O(2^L)$ operations. Next, to find the maximum parent proportion and minimum value on edges pointing to the non-maximum parent, we do ℓ and $\ell(\ell-1)$ operations for each cell population. Given that there are $\begin{pmatrix} L \\ l \end{pmatrix}$ cell populations with only positive marker conditions on each layer, this part requires $\sum_{\ell=2}^{L} \ell^2 \begin{pmatrix} L \\ l \end{pmatrix}$ or $O(\ell^3)$ operations. If we were to calculate the above for all cell populations, the first part would require $O(3^L)$ as opposed to $O(2^L)$ operations and we would have to consider $2^\ell \begin{pmatrix} L \\ l \end{pmatrix}$ instead of $\begin{pmatrix} L \\ l \end{pmatrix}$ cell populations per layer for the second part.

To follow up on all other cell populations, we observe that the score values for these cell populations collectively implicitly contain information about all nodes in the full hierarchy. As such, we can deduce the expected proportion of all other cell populations with at least one negative marker condition. For example, the first layer of a cell hierarchy can be specified with L cell populations. The count of these cell populations with a negative marker condition (e.g. A^-) can be deduced by the difference between the total proportion and the proportion of the corresponding cell population expressing positively for that marker (e.g. $P(A^-) = P(root) - P(A^+)$). The L'th layer can be specified with one cell population i.e. a cell population with all markers. The total number of cell populations needed to specify all cell populations is 2^L or the number of cell populations with only positive marker conditions.

Algorithm 2 Calculating the expected proportion of cell populations with negative marker conditions

---

$V \leftarrow$ cell populations with all positive marker conditions whose expected proportion is already calculated. $V^- \leftarrow$ all cell populations with ≥ 1 negative marker condition(s).

for $\ell := 2 \rightarrow L$ **do** $\mathbf{v}_\ell \leftarrow \{v | v \in V, |v| = \ell\}$

  while $v_\ell \neq \{\}$ **do** $\mathbf{v}^*_\ell \leftarrow \{\}$

    for $v_\ell := \mathbf{v}_\ell$ **do**

      for $p^+ :=$ a marker in v_ℓ with a positive$^+$ condition **do** $P(v^{p^-,1:\ell\setminus p^+}) = P(v^{1:\ell\setminus p^+}) - P'(v_\ell)$

$\mathbf{v}^*_\ell \leftarrow \{\mathbf{v}^*_\ell, v^{p^-,1:\ell\setminus p^+}\}$

$V = \{V, v^{p^-,1:\ell\setminus p^+}\}$

$V^- = \{V^- \setminus v^{p^-,1:\ell\setminus p^+}\}$ $\mathbf{v}_\ell = \mathbf{v}^*_\ell$

From this, we use Algorithm 2 to infer the expected proportion of all other cell populations with at least one negative marker condition. This algorithm takes one difference operation per cell population amounting to $O(3^L)$ operations total. Since we only calculate expected proportions for $O(2^L)$ cell populations and since the total number of other cell populations far out-scales $O(2^L)$, we achieve an overall runtime of $O(3^L)$.

In the case that there is more than one threshold per marker (e.g. on top of A^- and A^-, there also exists A^{++}, A^{+++}, and so on), the proof of correctness for expected proportions still holds as a cell population in layer ℓ must have ℓ parent nodes. The correctness of Algorithm 2 also still holds except the proportions represented by index $p+$ is now not a single proportion value but a sum of all proportions with a positive marker condition for the marker on index p. Note that this is contingent on a characteristic of flowType which labels markers with additional thresholds as positive marker conditions. Conversely, there can be only one negative marker condition per marker (i.e. there can be no A^{--}). As we still only calculate the expected proportion for cell populations containing exclusively positive marker conditions, the runtime of expected proportion calculation increases with the number of thresholds. If each marker is assigned $k \geq 1$ thresholds, then the number of cell population containing only positive marker conditions is $O((k+1)^L)$ while the total number of cell populations is $O((k+2)^L)$. The overall runtime then becomes $O((k+2)^L)$.

A.3 Layer-stratified Bonferroni correction

We define layer-stratified Bonferroni correction as follows. Let $p_{1:m}$ be p-values associated with a set of hypotheses $H_{1:m}$. For $\ell \in 1 \dots L$, let σ_ℓ be a subset of hypotheses $\{1 \dots m\}$ (e.g. cell populations in particular layer of a hierarchy). For $i \in \sigma_\ell$, we define the adjusted p-value $p'_i = p_i|\sigma_\ell|L$. We accept a hypothesis H_i if $p'_i < \alpha$.

Theorem A.3.1. *The layer-stratified Bonferroni correction has a family-wise error rate (FWER) of at most α.*

Proof.

$$
\begin{aligned}
& P\left(\bigcup_{i=1}^{m}(p'_i \le \alpha)\right) \\
&= P\left(\bigcup_{i=1}^{m}(p_i|\sigma_\ell|L \le \alpha\right) \\
&= P\left(\bigcup_{\ell=1}^{L}\bigcup_{i\in\sigma_\ell}(p_i|\sigma_\ell|L \le \alpha)\right) \\
&\le \sum_{\ell=1}^{L} P\left(\bigcup_{i\in\sigma_\ell}(p_i|\sigma_\ell|L \le \alpha)\right) \\
&\le \sum_{\ell=1}^{L}\sum_{i\in\sigma_\ell} P\left(p_i|\sigma_\ell|L \le \alpha\right) \\
&\le \sum_{\ell=1}^{L}\sum_{i\in\sigma_\ell} \alpha/|\sigma_\ell|L \\
&= \alpha
\end{aligned}
$$

□

When there is a single layer, or when all layers have the same size, the layer-stratified Bonferroni correction is identical to a standard Bonferroni correction.

A.4 An example of the filters used to determine whether a cell population has a true positive significant p-value based on SpecEnr

A significant cell population should: 1) have a mean count of > 50 events to prevent inflated ratios (this threshold can be adjusted based on data set; empirically, this value worked well for our experiments), 2) have significantly different actual vs expected proportions for at least one of the sample classes. (if both classes contain actual vs expected proportions that are significantly different, then SpecEnr can be used without filtering), and 3) contains actual and expected proportions that are different at the same rate across both sample classes. For filter 1), we tested thresholds 25, 50, 100, and 200. 50 worked well in practice as we ended up with just as many MDCPs as when we used higher thresholds. However, this threshold may differ based on the data set. For example, if one hypothesizes that their MDCP may be a rare cell population with cell counts of < 50 events, then this threshold should be decreased. Note we used a significance threshold of $< .05$ for all T-test p-values.

An example of Filter 2 is shown in Figure A.1B where samples of both classes of the flowcap data set have similar actual vs expected proportion values. However, the proportion values from the control class are relatively smaller compare to those of samples from the AML class. To prevent this , we have filter 3), illustrated in Figure A.1C. We take the difference between the actual proportion of a random set of 43 control samples and all 43 AML samples. We do the same for expected proportions and we compare these two sets of differences using a T-test.

In Figure A.1, cell population $SS^{-}CD45^{-}CD34^{+}$ from the flowcap data set has 1) a mean cell count > 50, 2) is significant for the second condition (Figure A.1B shows the significant and insignificant difference between actual and expected proportions in the control and AML class respectively), and 3) is insignificant for the third condition (Figure A.1C shows that the difference between actual and expected proportions across sample classes are not significantly different). Therefore, it is insignificant overall.

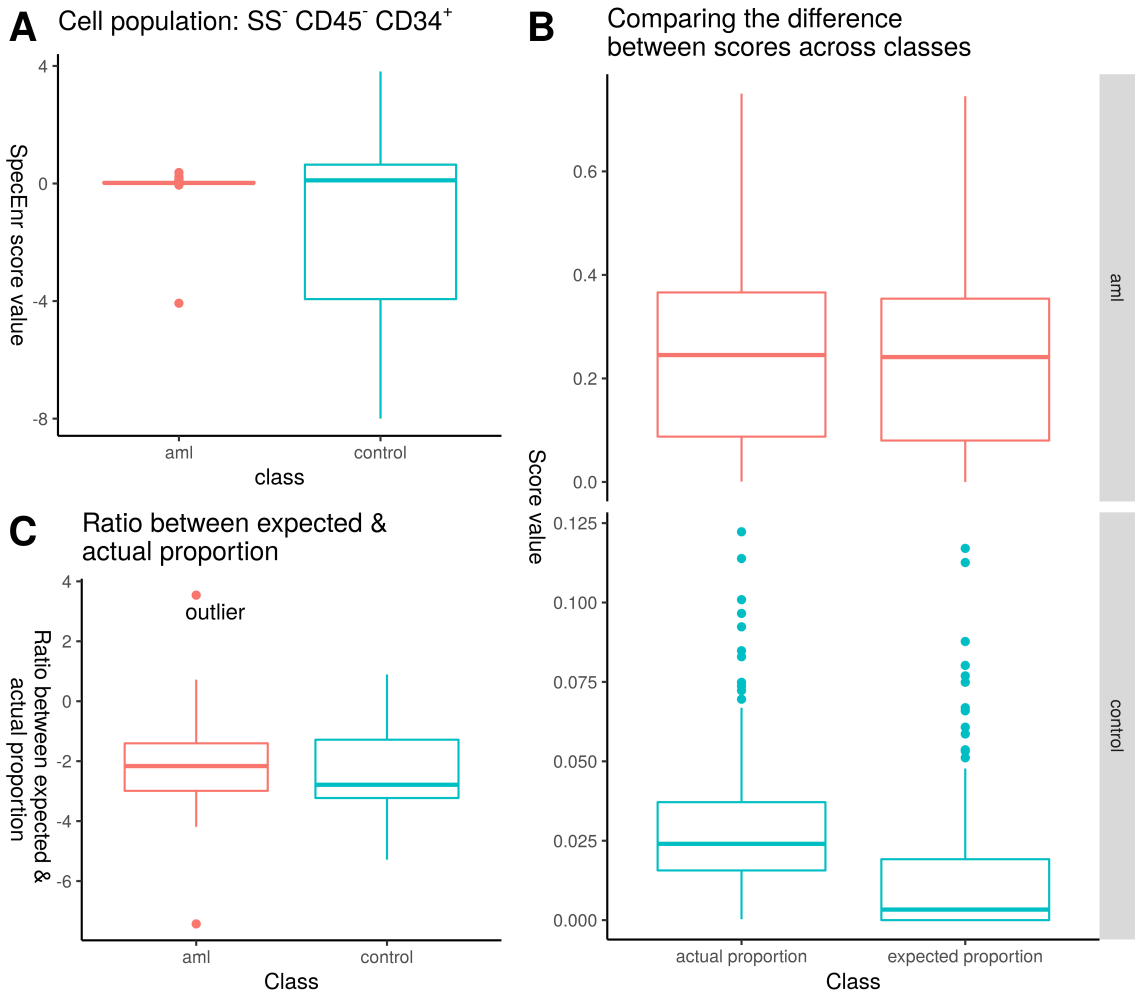

Figure A.1: Filters applied to SpecEnr p-values shown on the flowcap data set's $SS^{-}CD45^{-}CD34^{+}$ cell population. A) compares the SpecEnr values between the two classes. B) compares the difference between the actual and expected proportions within sample classes, and C) compares the difference between the actual and expected proportions across sample classes. The comparisons in B) and C) correspond to our three filters.

A.5 SpecEnr produces robust p-values and q-values

Figure A.2 shows that SpecEnr is just as robust as other cell population scores because A) F-measure, recall, precision, and Spearman correlation scores measure high consistency between two theoretically identical positive control data sets (pos1) and for the negative control data set (neg1), B) the proportion of cell populations with significant ($< .05$) p-values is the expected .05, and C) the QQ-plot shows that unadjusted T-test p-values for all scores align well with a theoretically uniform distribution indicative of our SpecEnr score's statistical robustness.

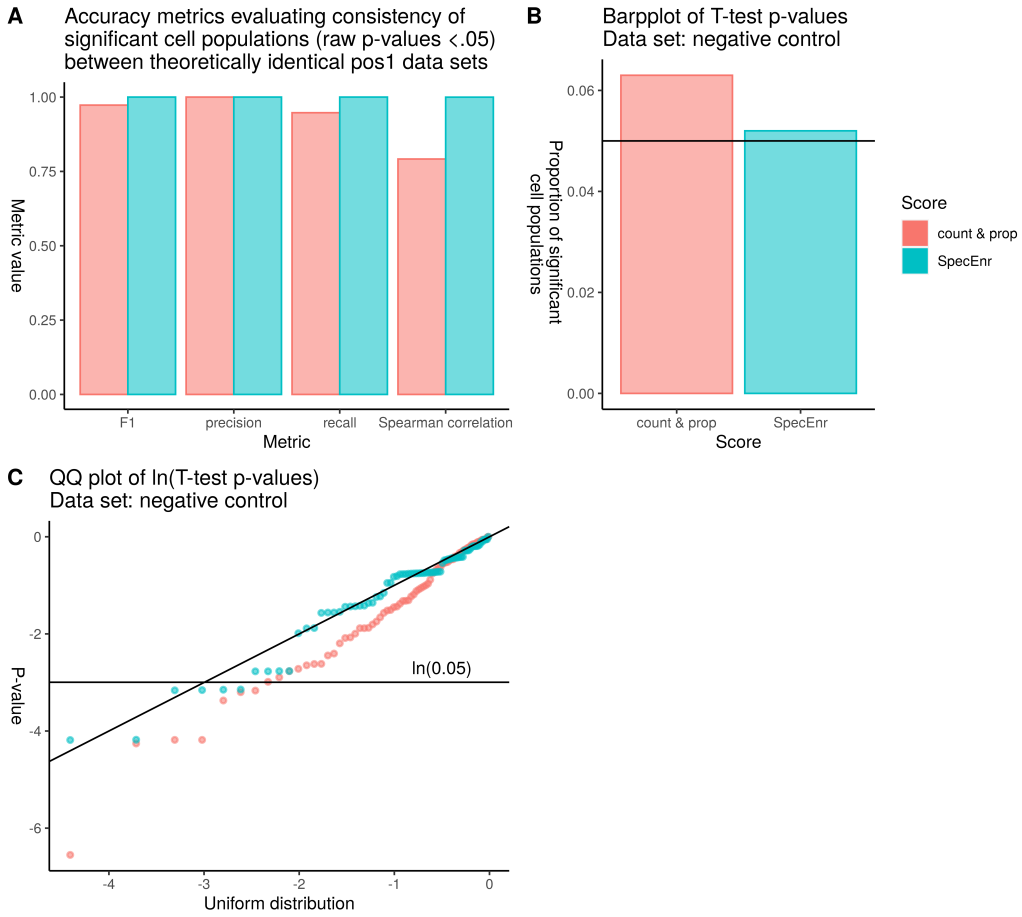

Figure A.2: A) F-measure, recall, precision, and Spearman correlation scores for two theoretically identical positive control data sets (pos1) and for the negative control data set (neg1), B) proportion of cell populations with significant ($< .05$) p-values, and C) -plot for unadjusted T-test p-values for all scores.

A.6 Proportion vs SpecEnr plots for data sets pos1-3 where the abundances of the same cell populations are decreased by 50%.

Figure A.3 shows cell hierarchy plots and cell populations with significant flowGraph SpecEnr q-values for the data sets pos1-3, where cell population abundances are decreased instead of increased. SpecEnr highlights only the MDCPs and their ancestors as opposed to all DCPs as with prop q-values.

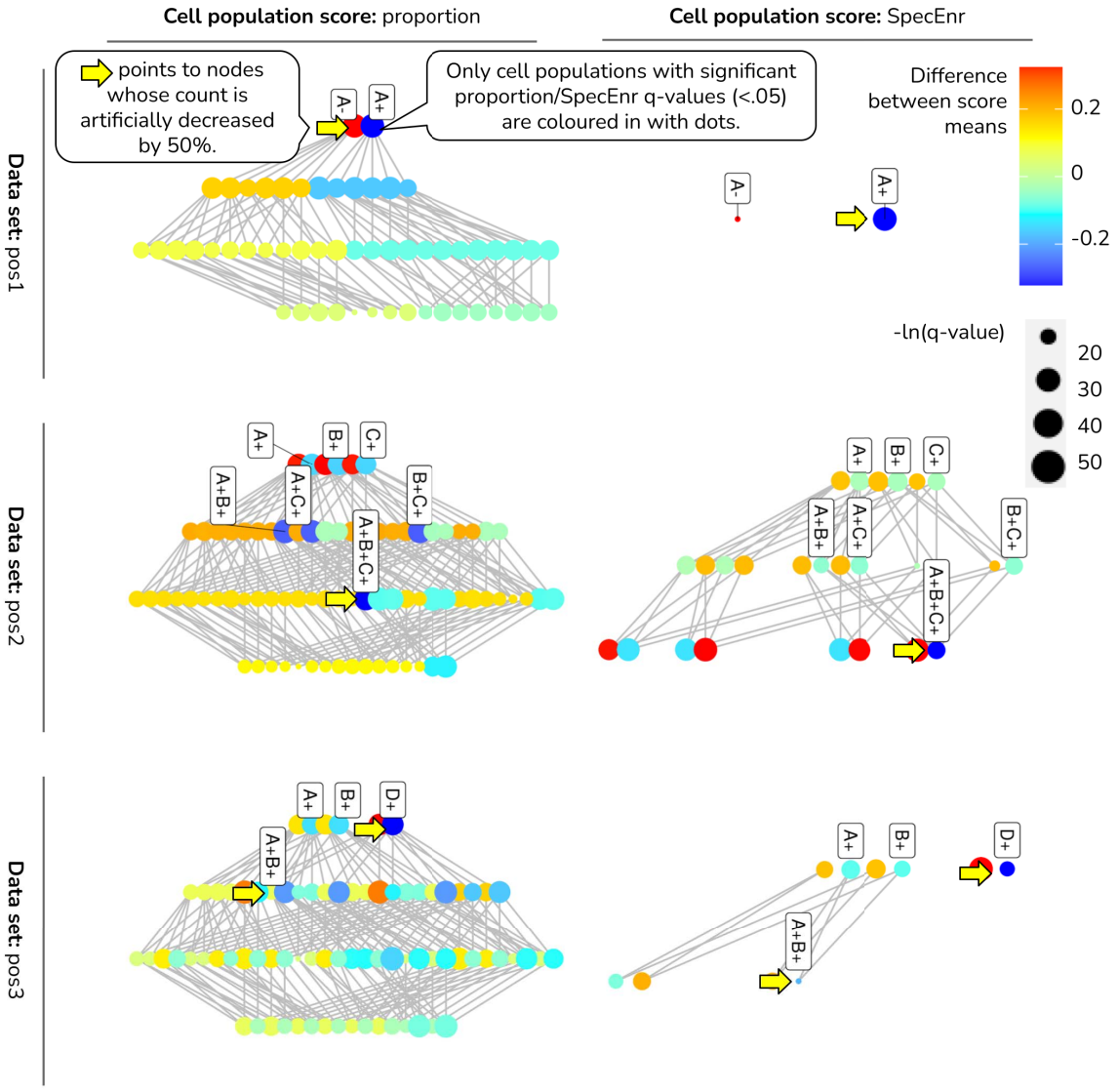

Figure A.3: Cell hierarchy plots showing cell populations with significant flowGraph SpecEnr q-values for the data sets pos1-3, where cell population abundances are decreased instead of increased.

A.7 Runtime experiments

Table A.1 shows the runtime in seconds (hours and minutes if the time surpasses a minute) of flowGraph for calculating the edge list of the cell population hierarchy, SpecEnr scores, and T-test p-values (including the filters). We also show runtimes when we use 15 cores vs just 1 core for the pregnancy data set to gauge potential runtime decreases with hardware improvements. Note that we did not do this for the other data sets as setting up the workers for parallel processing would take a few seconds, which would already surpass the runtime for the actual SpecEnr calculation.

Table A.1: flowGraph runtime in seconds for calculating the edge list of the cell population hierarchy, SpecEnr scores, and T-test.

data set	flowGraph (1 core) (seconds)	flowGraph (15 cores) (seconds)	no. of markers	no. of cell populations
pos1	0.318	NA	3	27
pos2	0.353	NA	3	27
pos3	0.296	NA	3	27
flowcap	12.659	NA	7	2,193
pregnancy	7,978	2,307	13	109,192

To see if we can decrease runtime, we also list runtimes for flowGraphSubset in Table A.2. flowGraphSubset is an alternative mode of flowGraph that calculates the edge list, SpecEnr, and T-test q-values (including the filters) for only the cell populations that have a parent population with a significant SpecEnr q-value. This way, we skip calculating everything for cell populations that do not meet this criterion, thereby saving runtime. The assumption that important MDCPs always have at least one parent population with a significant SpecEnr q-value applies to almost all cases. However, if the user wants to test multiple class label sets on the same samples (e.g. control vs experiment, age, etc.), we still recommend users to use the basic flowGraph constructor to calculate the SpecEnr score for all cell populations. So, we recommend users to only use flowGraphSubset IF: 1) the user's data set has more than 10,000 cell populations and you want to speed up your calculation time AND 2) you only have one set of classes you want to test on the same set of samples (e.g. control vs experiment).

Table A.2: flowGraphSubset runtime in seconds for calculating the edge list of the cell population hierarchy, SpecEnr scores, and T-test.

data set	flowGraphSubset (1 core) (seconds)	flowGraphSubset (15 cores) (seconds)	no. of markers	no. of cell populations
pos1	0.135	NA	3	27
pos2	0.155	NA	3	27
pos3	0.154	NA	3	27
flowcap	11.371	NA	7	688
pregnancy	6,190	958	13	17,991

Appendix B

Automated 2D gating via motif matching for cell population identification

B.1 Example scatterplots

This section lists example plots for all the scatterplots whose F1 scores are plotted in Figure 5.6 except for the Singlet cells scatterplot — which is shown in Section 5.4. These examples serve to support our statement: Our method uses visual features to accurately identify cell populations with arbitrarily shaped polygon gates. Since our method projects gates on visually similar regions of the scatterplot as that of the training sample, the resulting is easy to interpret — users can easily understand why our method projected its gate where it did based on visual cues from the scatterplot. We will not repeat this statement below as it applies to all scatterplots.

B.1.1 Scatterplot: All cells

To extract the Singlets from All cells, we projected a five vertices polygon gate from three training samples onto the rest of our testing samples. This gate is an example of an arbitrarily shaped polygon with four vertices and we hypothesize that this is an easy gate to project as it isolates a naturally formed cluster of cells. From Figure 5.6, we see that this was indeed the case. Our method obtained the highest F1 scores for this scatterplot.

While we obtained lower scores on a few samples, our previous statement still holds. No matter the F1 score, the polygon gate our method projected was in visually similar regions as that of the training sample, making it easier to understand why our method projected the gate where it did. In Figure B.1, we obtained the lowest accuracy when the ground truth testing sample gate was on visually dissimilar regions compared to that of the training sample. In this case, the ground truth gate of the testing sample included additional cells plotted towards the bottom right region of the scatterplot.

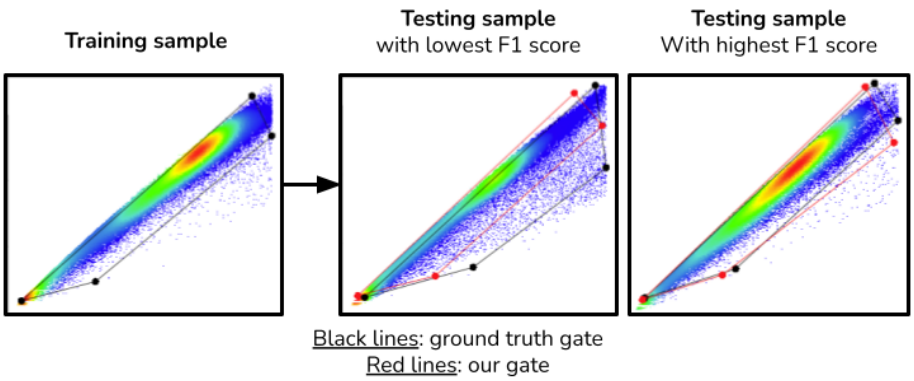

Figure B.1: The testing sample gatings for which we obtained the lowest and highest F1 scores for the scatterplot plotting All cells containing a Singlets gate; also included are the original training sample and gate used.

B.1.2 Scatterplot: B-cells

We separately projected two gates (gating $CD24^{-}CD8^{-}$ B-cells and Naive B-cells) from the scatterplot containing B-cells. Both gates had five vertices. We chose to experiment on these gates because they gated cells that do not separate well from other cells. The vertices also belonged on areas of the scatterplot that were not necessarily visually distinct from other areas. In addition, for the $CD24^{-}CD8^{-}$ B-cells gate, there were relatively fewer cells inside these gates so a small error in gating would cause a large drop in the F1 score.

In the end, our method still obtained higher mean F1 scores than flowSOM, we obtained relatively low scores for the $CD24^{-}CD8^{-}$ B-cells gate, and we surprisingly got the third-highest F1 score on the Memory B-cells gate.

As shown in Figures B.2,B.3, the testing samples corresponding to the lowest F1 scores were outlying samples that look distinct from the chosen training samples. However, the vertices that our method chose for its polygon gates still resided on visually similar areas as those in the training sample. Ironically, the testing samples with the highest F1 scores also looked distinct from the training sample; but our methods' strategy luckily obtained an appropriate gate that largely overlaps with the ground truth gate. These examples show a problem that our method and all supervised methods — the inability to project gates onto testing samples that do not look similar to any training sample.

B.1.3 Scatterplot: Live cells

The scatterplot plotting Live cells contained a non-convex arbitrarily shaped seven vertices polygon B-cell gate. Given that this gate required that we accurately project seven vertices as opposed to five (i.e. larger degree of freedom), we hypothesized that we would obtain lower F1 scores. This proved to be true. We also hypothesized that flowSOM would receive higher F1 scores than our method since the gate isolates a cell population whose data distribution was naturally separated from the other cells. Fortunately, we obtained a higher mean F1 score. Another observation is that for our method, this scatterplot obtained the

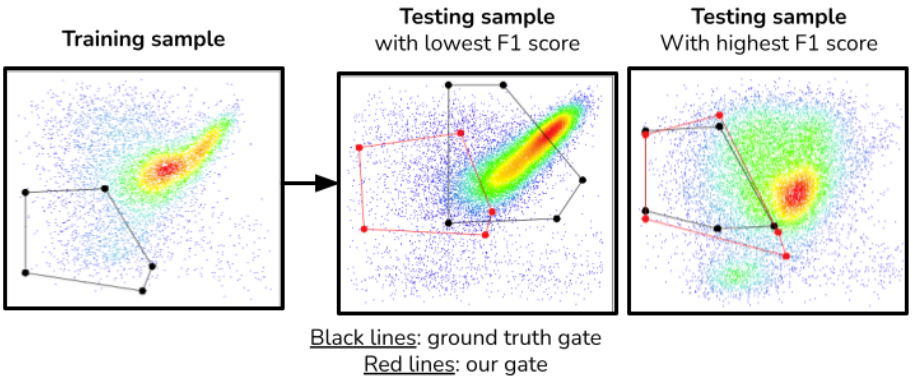

Figure B.2: The testing sample gatings for which we obtained the lowest and highest F1 scores for the scatterplot plotting B-cells containing a $CD24^{-}CD8^{-}$ B-cells gate; also included are the original training sample and gate used.

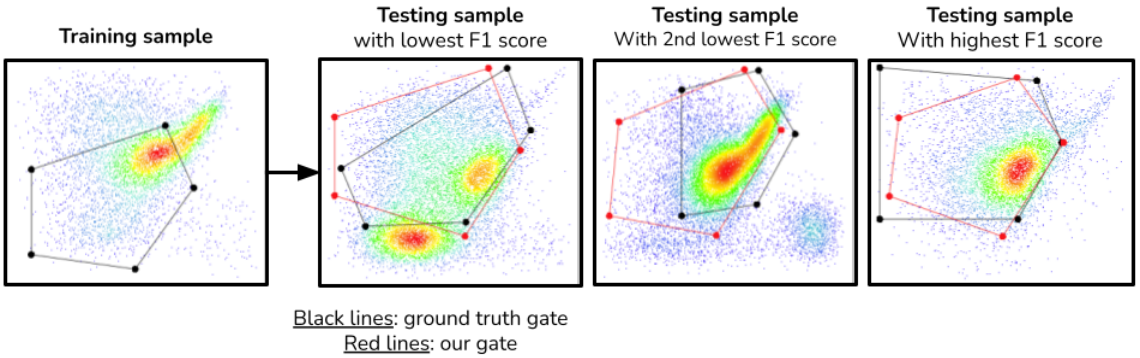

Figure B.3: The testing sample gatings for which we obtained the lowest and highest F1 scores for the scatterplot plotting B-cells containing a Naive B-cells gate; also included are the original training sample and gate used.

second-lowest mean F1 score out of all the scatterplots i.e. it was a difficult scatterplot to gate.

In Figure B.4, the gate we obtained in the testing sample that corresponded to the lowest F1 score included cells from the bottom left cluster of cells. While not dramatically different from the original ground truth gate, this was an example of when our method chose vertices that were in similar coordinates as that of the training sample over similar visual regions. This occurs when cells in a scatterplot are significantly shifted. This issue can be solved by increasing weights for visual similarity (a_i) over coordinate similarity (p_i) in the node objective function (Section 5.2.3). In addition, one can also increase the number of candidate voxels for each vertex. This will increase computational complexity in favour of accuracy.

Interestingly, we also see that the testing sample corresponding to the highest F1 score contains a gate that does not have a perfect area overlap with the gate our method obtained. However, since the area that did not overlap did not contain many cells, this did not affect accuracy.

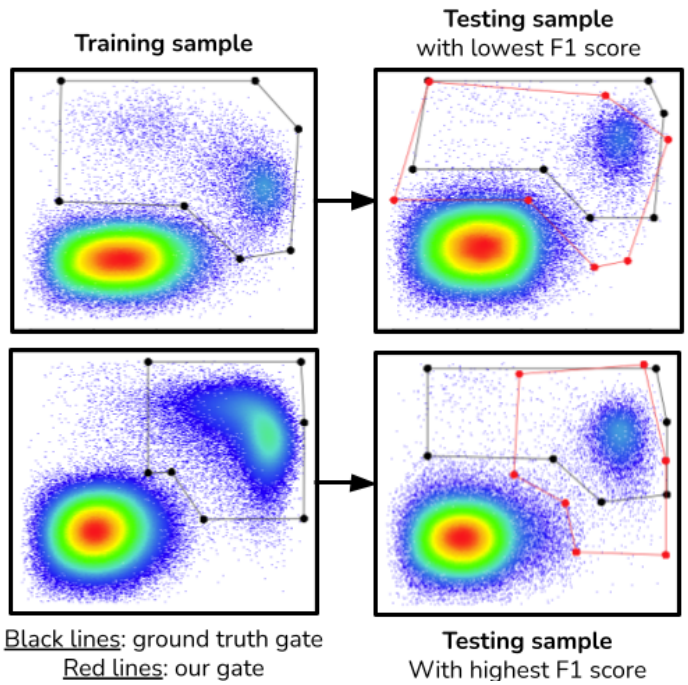

Figure B.4: The testing sample gatings for which we obtained the lowest and highest F1 scores for the scatterplot plotting Live cells containing a B-cell gate; also included are the original training sample and gate used.

Appendix C

Automated 2D gating via few-shot image segmentation for cell population identification

Table C.1: We used 4 freely available data sets, each with samples plotted on several pairs of markers to create our input scatterplot images. Sample size within each data set varies based on the ground truth gates available.

Data set	Samples	Cells per sample	Scatterplot name: cell population gated (plotted marker pair)	Cell populations contained
HIPCB-cell	1,349	4,000	CD19+CD20+ (CD10, CD27)	2: CD10+, CD10-
HIPCB-cell	1,344	4,000	IGM (CD10, CD38)	1+: immature transition B-cell, other
HIPCB-cell	1,350	4,000	CD19+B-cell (CD19, CD20)	2: CD19+CD20+, CD19+CD20-
HIPCB-cell	1,350	50,000	Not granulocyte (CD19, SSCa)	1+: CD19+B-cell, other
HIPCB-cell	1,350	4,000	CD10- (CD27, IgD)	4: atypical B-cell, unswitched memory B-cell, naïve B-cell, switched memory B-cell
HIPCB-cell	1,328	100,000	Live cell (CD34, SSCa)	1+: blast, other
HIPCB-cell	1,350	4,000	CD19+B-cell (CD38, CD138)	1+: plasma cell, other
HIPCB-cell	1,337	4,000	CD19+B-cell (CD38, CD27)	1+: plasmablast, other
HIPCB-cell	1,350	100,000	Live cell (CD66, CD14)	2: granulocyte, not granulocyte
HIPCB-cell	1,350	200,000	Singlets (FSCa, SSCa)	2+: beads, cells, other
HIPCB-cell	1,350	4,000	CD19+B-cell (IgD, IgM)	4: IGD-IGM- B-cell, IGD+IGM- B-cell, IGD-IGM+ B-cell, IGD+IGM+ B-cell
HIPCB-cell	1,342	125,000	All cells (viability dye, SSCA)	1+: live cell, other
HIPCmyeloid	1,379	50,000	Granulocytes (CD11b, CD16.FITCA)	4: CD11b+CD16+ mature neutrophil, CD11b-CD16- immature neutrophil 1, CD11b+CD16- granulocyte, CD11b-CD16+ immature neutrophil
HIPCmyeloid	1,379	5,000	HLADR+CD14- (CD123, CD11c)	3: mDC, pDC, B-cell
HIPCmyeloid	1,379	2,500	CD56-CD16-cells (CD123, HLADR)	1+: basophil, other
HIPCmyeloid	1,379	8,000	HLADR+CD14+ Monocytes (CD14, CD16.FTICA)	2: Non/classical monocyte
HIPCmyeloid	1,379	5,000	CD3- (CD16.FITCA, CD56)	5: CD56Hi NK, CD56-CD16-, CD56-CD16+ NK, CD56dimCD16- NK, CD56dimCD16+ NK
HIPCmyeloid	1,379	20,000	CD3+ Tcells (CD16.FITCA, CD56)	1+: CD56+CD16+ NK Tcell, other
HIPCmyeloid	1,379	23,000	CD3+Tcells (CD3, gd)	2: gd Tcell, gd- Tcell
HIPCmyeloid	1,379	28,000	HLADR-CD14- (CD3, SSCa)	2: CD3+ Tcell, CD3-
HIPCmyeloid	1,379	40,000	CD11b+CD16+ Mature neutrophils (CD64, CD11b)	2: CD64+, CD64-
HIPCmyeloid	1,379	100,000	Live cells (CD66, CD16)	2+: granulocyte, CD45+CD66- not granulocyte, other
HIPCmyeloid	1,379	60,000	CD45+CD66- Not granulocyte (HLADR, CD14)	3+: HLADR+CD14+ Monocyte, HLADR+CD14-, HLADR-CD14-, other
pregnancy	112	300,000	Leukocyte (CD66, CD45)	2+: granulocyte, mononuclear
pregnancy	112	75,000	Mononuclear (CD3, CD19)	3+: B-cell, Tcell, NK lin-, other
pregnancy	112	15,000	NKLin- (CD14, CD7)	2: lin-, NK
pregnancy	112	12,000	Lin- (CD14, CD16)	4: cMC, ncMC, intMC, not MC
pregnancy	112	50,000	Tcell (CD4, CD8)	4: CD4+ Tcell, CD8+ Tcell, CD4-CD8- Tcell, CD4+CD8+ Tcell
pregnancy	112	30,000	Tcell (CD4, CD45RA)	2: CD4+ naïve Tcell, CD4+ memory Tcell
pregnancy	112	13,000	CD4+Tcell (FoxP3, CD25)	1+: Tregs naïve Tcell, CD4+ naïve Tcell other
pregnancy	112	3,000	CD4+Tcell (TCRgd, CD3)	1+: gamma-delta Tcell, CD4-CD8- Tcell other
pregnancy	112	17,000	NotCD4+CD8+Tcell (CD8, CD45RA)	2: CD8+ naïve Tcell, CD8+ memory Tcell
pregnancy	112	17,000	CD8+Tcell (Tbet, CD45RA)	2+: CD25+CD8+ naïve Tcell, CD25+CD8+ memory Tcell, CD25+CD8+ Tcell other
sangerP2	2,348	11,000	CD11b+lymphocyte (CD5, CD11b)	2: granulocyte, not granulocyte
sangerP2	2,341	200,000	Not granulocyte (Ly6C, CD11b)	2: monocyte, not monocyte
sangerP2	2,329	200,000	Not monocyte (CD11b, SSCh)	2: eosinophil, not eosinophil
sangerP2	2,328	200,000	Not eosinophils (CD161, CD19)	2: CD161+, CD161-
sangerP2	2,328	10,000	CD161+ (CD5, CD11b)	2: NKT, NK
sangerP2	2,328	7,000	NK (CD11b, Ly6C)	4: NK immature Ly6C+, NK mature Ly6C+, NK immature Ly6C-, NK mature Ly6C-
sangerP2	2,328	3,000	NKTcell (CD11b, Ly6C)	4: NKT CD11b-Ly6C+, NKT CD11b+Ly6C+, NKT CD11b-Ly6C-, NKT CD11b+Ly6C-
sangerP2	2,328	200,000	CD161- (MHCII, CD5)	2: Tcell, not Tcell
sangerP2	2,328	134,000	Not Tcell (CD19 , CD11c)	2: B-cell, not B-cell
sangerP2	2,328	3,600	cDC (CD11b, MHCII)	3: cDC CD8+, cDC CD11b+, cDC
sangerP2	2,328	117,000	B-cell (CD5, CD21)	2: B1B-cell, B2B-cell
sangerP2	2,328	115,000	B2B-cell (CD23, CD21)	3+: preB, MZB, folB, other

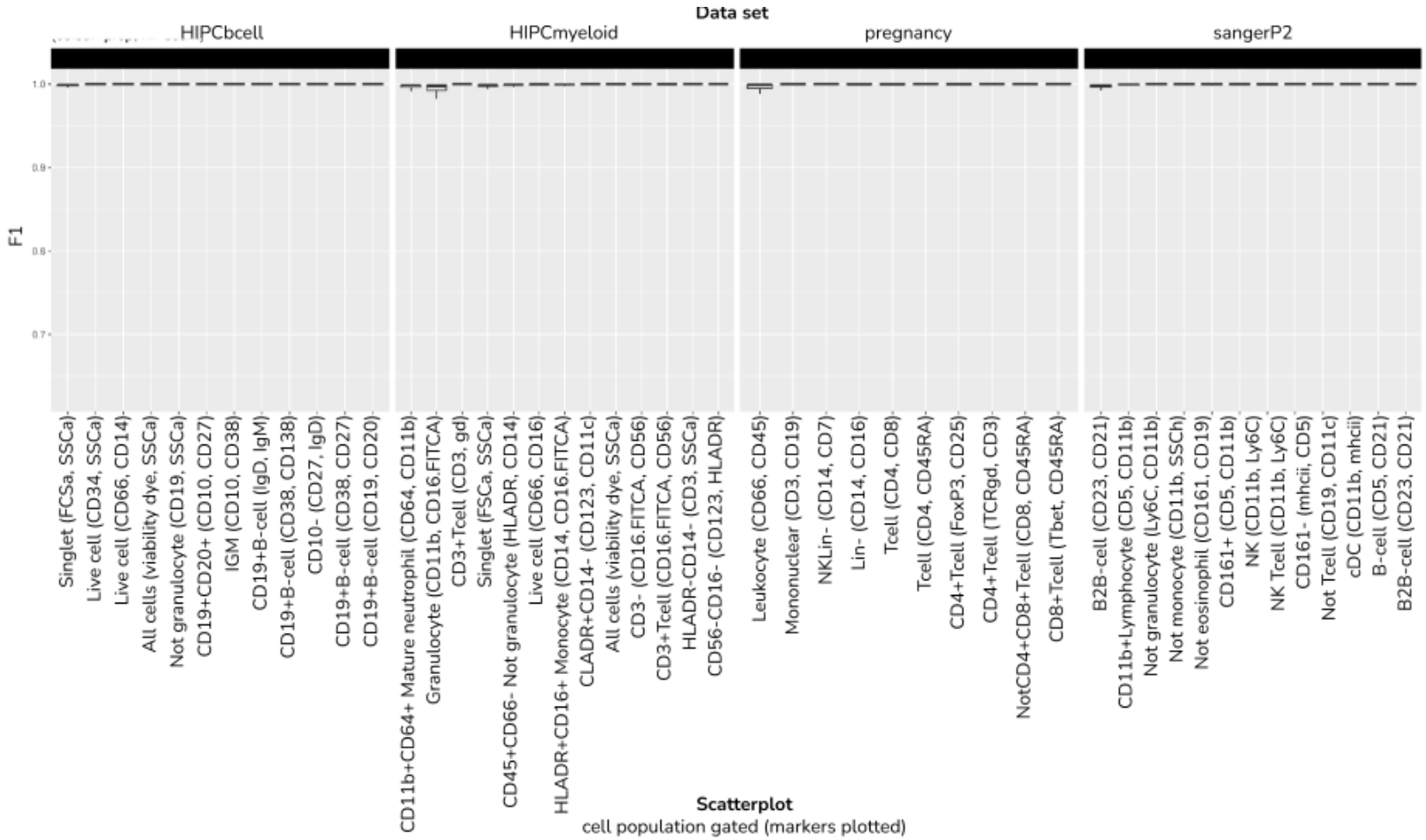

Figure C.1: F1 scores for each cell population if all pixels are classified correctly at resolution 256 × 256 pixels. See data sets used to create this figure in Table C.1