LUDWIG-MAXIMILIANS-UNIVERSITÄT
TECHNISCHE UNIVERSITÄT MÜNCHEN

Helmholtz Center Munich
Computational Modeling in Biology

Masterarbeit
in Bioinformatik

Data Mining Approaches for Analyzing Single Cell Stem Cell Data

Bernd Streppel

Aufgabensteller:	Prof. Fabian Theis
Betreuer:	Florian Buettner
Abgabedatum:	15. Dezember 2012

Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

15. Dezember 2012 ____________________________

Bernd Streppel

Abstract

In stem cell research current developments go towards single cell data acquisition and analysis. Two frequently used acquiring approaches are fluorescence activated cell sorting (FACS) and continuous time-lapse movies. Both produce complex, large and therefore not easy to analyse and interpret datasets. Here we develop new data mining approaches allowing for an adequate analysis and visualization of these types of data.

Correlations of markers in FACS data can reveal new biological insights. To detect theses redundancies we designed a feature selection algorithm based on an existing non-deterministic algorithm for hierarchical clustering of FACS data (SPADE). It is able to find the smallest subset of markers which is able to preserve the (cluster) structure of the data. A distance measure is defined to quantify the similarity between different marker combinations. We tested the algorithm on a hematopoietic as well as a mesenchymal stem cell dataset.

Current visualization techniques for FACS data have several drawbacks like not using all dimensions or not preserving distances. To overcome this we combined and optimized SPADE with the visualization method tree preserving embeddings (TPE). Our visualization guarantees the hierarchical structure of the data to be preserved in the visualization. We showed the suitability of our approach through an evaluation with expert knowledge. In addition we present an extended approach which allows the intuitive visualization of different timepoints, which we demonstrate on a dataset with four timepoints of mesenchymal stem cells.

Manual inspection of stem cell trees - which are the result of processed time-lapse movies - results easily in the oversight of simple frequent pattern. We present a new mining algorithm to find the most interesting patterns which are differentialy expressed in two treesets. For this we define possible patterns and their matching to trees. On two biological datasets we were able to show the applicability to real data.

Abstract

Experimente auf Einzelzell-Ebene sind in der aktuellen Stammzellforschung unabdingbar. Die zwei am häufigsten verwendeten Techniken sind "fluorescence activated cell sorting" (FACS) und Zeitrafferaufnahme von Stammzellen. Die resultierenden Daten sind hoch-dimensional, komplex und deshalb schwierig mit Standard Techniken zu analysieren. In dieser Arbeit stellen wir daher neue Data Mining Ansätze vor, die bei der Analyse und Visualisierung dieser Daten helfen sollen.

Korrelation zwischen verschiedenen FACS Markern erlauben interessante Rückschlüsse auf das untersuchte biologische System. Um diese Korrelationen zu finden haben wir, basierend auf einer existierenden hierarchischen Clustering Methode (SPADE), einen neuen Marker-Selektions-Algorithms entworfen. Dieser sucht eine Teilmenge von Markern, die ausreichend ist um die intrinsische Clusterstruktur der Daten zu erklären. Dazu definieren wir ein neues Distanzmaß um die Unterschiede zwischen zwei hierarchischen Clusterings zu quantifizieren. Zur Evaluation werden zwei Datensätze mit hematopoetischen bzw. mesenchymalen Stammzellen verwendet.

Aktuelle Visualisierungsverfahren für FACS Daten stellen teilweise nicht alle verfügbaren Dimensionen gleichzeitig dar und haben Probleme mit dem Erhalt von Strukturen in der Visualisierung. Deshalb verbinden wir das hierarchische Clusteringverfahren von SPADE mit der Visualisierung "tree preserving embedding" (TPE) um diese Probleme zu umgehen. Diese Kombination garantiert - im Gegensatz zu anderen Verfahren - den Erhalt der Struktur durch die Bewahrung aller Abstände eines Dendrograms. Wir demonstrieren die Funktion und Nützlichkeit unseres Ansatzes mit Hilfe eines Datensatzes mit Expertenwissen. Zusätzlich ist das Verfahren in der Lage Zellpopulationen aus verschiedenen Zeitpunkten an einer einheitlichen Struktur darzustellen was mit einem Datensatz aus mesenchymalen Stammzellen gezeigt wird.

Die manuelle Suche nach häufigen Mustern in Stammzellbäumen (den Ergebnissen der Zeitrafferaufnahme) ist schwierig, zeitaufwändig und fehleranfällig. Wir präsentieren einen effizienten Muster-Erkennungs-Algorithmus der Muster findet, die differenziell in zwei Datensätzen vorkommen. Dazu Definieren wir Muster, ihre Kanonisierung und Spezialisierung, sowie ihre Beschreibung der Stammzellbäume. An zwei Stammzelldatensätzen zeigen wir die Funktionalität sowie Nützlichkeit des Algorithmus zur biologischen Analyse.

Contents

Chapter 1

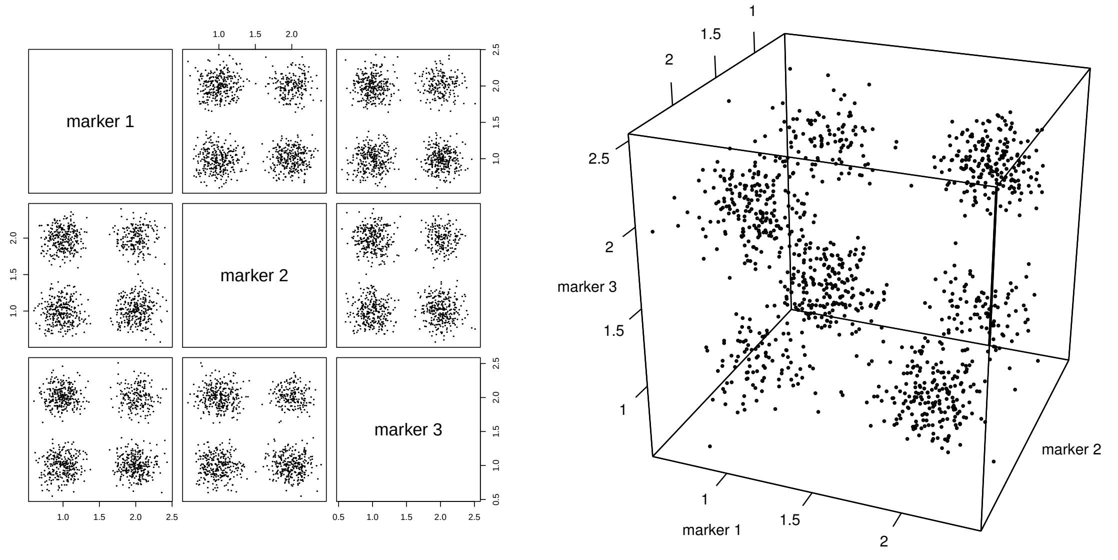
Introduction

Stem cells are characterised by two main properties: first, they are able to maintain themselves and second, they are able to *differentiate* into multiple possible cell types (multipotency) [1]. They are of major interest for regenerative medicine which utilises this multipotency for repairing damaged tissue [2]. This requires a good understanding of the underlying processes and one major goal of stem cell research is to reveal the intrinsic control mechanisms of the differentiation process. Through the integration of fusion proteins with fluorescence proteins into the genome and the availability of antibodies for surface antigens in the cell membrane, techniques are available to observe intra-cellular processes from the outside.

Population average analyses allow to make statements about the whole observed cell populations but cannot reveal differences between single cells in the same population. In order to resolve such heterogeneities observations on the single cell level are required [3]. The obtained data has a complex structure and requires special approaches to be analyzed, but allows for the modelling and prediction of cell behaviours [4][5]. *Data Mining* is the process of discovering patterns in large datasets and allows for deeper insights into a system [6]. So data mining methods are perfectly suitable to analyse the single cell data.

The first type of single cell data we use in this thesis is *fluorescence activated cell sorting* (FACS) data, which consists of fluorescence marker intensities measured for each single cell in a probe [7]. The resulting large (up to millions of cells) dataset is hard to analyze because each of the up to 18 ([8]) measured markers represents a new dimension. The standard way to look at these high-dimensional datasets is a series of scatter plots. The drawback is that only two markers can be displayed at the same time and even when looking at several scatter plots simultaneously it is difficult to determine the underlying high dimensional structure (see figure 1.1). We present a new method of visualizing FACS data in 2D which is able to preserve the internal structure in the visualization and allows intuitive interpretation of the data.

A different way of reducing complexity is to omit some markers of the dataset. We present a method to find marker subsets which are sufficient to explain the whole dataset,



(a) Scatter plots (b) New visualization technique

Figure 1.1: It is hard to determine the underlying high dimensional structure only from scatter plots. (a) For an artificial dataset with three markers all scatter plots are shown. (b) The real three dimensional structure with 7 - not as expected with 8 - populations is shown.

so the remaining markers can be ignored. This gives interesting biological insight into the analyzed system and also frees wavelengths in FACS experiments which can then be used with new interesting markers.

One general problem with standard FACS analysis is that only one timepoint is available and even with several measurements at different timepoints there is no easy connection between the cells in each timepoint. To determine the dynamics of stem cell differentiation several timepoints are required together with the identification of the same cell at these timepoints. Continuous time-lapse movies of stem cells can fulfill these requirements and are increasingly used in stem cell research. After post-processing, the movies result in stem cell trees which represent the life of a stem cell and its progeny from the experiment start to fully differentiated cells.

To analyze these trees simple approaches such as calculating average cell lifetime or number of generations per tree can be used [9], but are limited and not able to include the tree structure into the analysis. However, the tree structure is the most valuable information and should be used in such analyses. More sophisticated analysis methods are required to find complex reoccurring patterns in the trees. Such a pattern can be for example a stem cell divides into two daughters, one daughter differentiates into a specific cell type while the other dies. These patterns are hard to identify manually (see figure 1.2), so we developed an algorithm which is able to find frequent patterns on stem cell

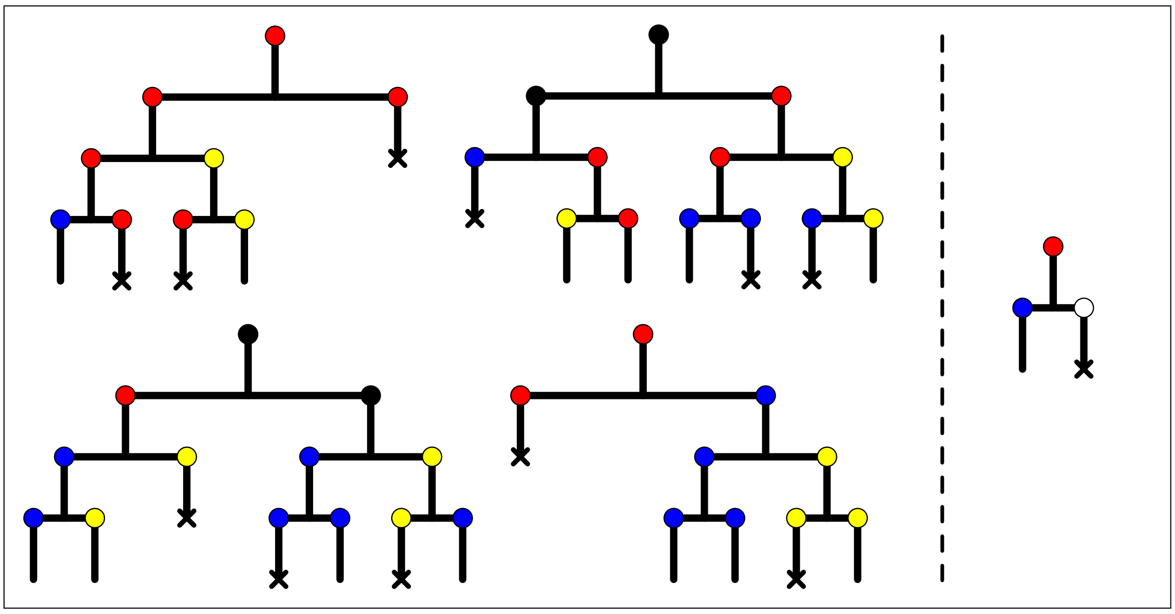


Figure 1.2: It is difficult to find common patterns in stem cells trees manually. Four artificial annotated stem cell trees (left) together with one patten (right) occurring in all four trees are shown. The colours correspond to different cell fates; a cross indicates apoptosis.

trees efficiently.

Structure of this Thesis

This thesis consists of six chapters. In chapter 2 we introduce hematopoiesis as the best understood stem cell system and two experimental techniques to obtain single cell data which are analyzed in later chapters. The question which markers are sufficient to define cell populations is addressed in chapter 3. Chapter 4 discusses a new approach to visualize FACS data which allows an easier interpretation of high dimensional data. For stem cell trees we present a pattern discovery algorithm in chapter 5 which is able to detect frequent and interesting patterns. The last chapter summarises this thesis and gives an outlook on extensions and future research of the presented methods.

Chapter 2

Introduction to Hematopoiesis

Blood is one of the major organs of higher organisms and responsible for the transport of many important molecules like oxygen and carbon dioxide. It consists of multiple different cell types like red blood cells or T-cells which have to be kept on appropriate numbers. This process of life-long production of blood cells from stem cells is called *hematopoiesis* [10].

Hematopoiesis is the best understood adult stem cell system [3] and preserved through the vertebrate evolution [11]. Hence there is large research in mouse hematopoiesis, which is considered to be transferable to the human system. Although this is fundamental research, the future goal is to support the regenerative medicine to treat blood diseases like leukemia [3].

Hematopoietic stem cells are characterised by two major properties: their multipotency and their ability for a long-time self-renewal. They can differentiate into *all* different blood cell types and are able to reproduce and maintain themselves.

2.1 The Hematopoietic Hierarchy

Different hematopoietic cell types and their progenitors are organized as hierarchy. The current state of scientific knowledge of this hierarchy is shown in figure 2.1. The hierarchy is not a real tree because there are multiple paths from stem cells at the top to a specific committed cell at the bottom. An example are *granulocyte-macrophage progenitors* (GMPs) which can develop either from *common myeloid progenitors* (CMPs) or *granulocyte-macrophage-lymphocyte progenitors* (GMLPs). This yields to the hypothesis that there are no distinct cell types but a continuum defined by the bias of cells towards a specific cell type [12].

At the top of the hierarchy there are hematopoietic stem cells (HSCs) while at the bottom there are fully differentiated cell types like erythrocytes or megakaryocytes. HSCs give rise to *multipotent progenitor cells* (MPPs) which are still multipotent and differenti-

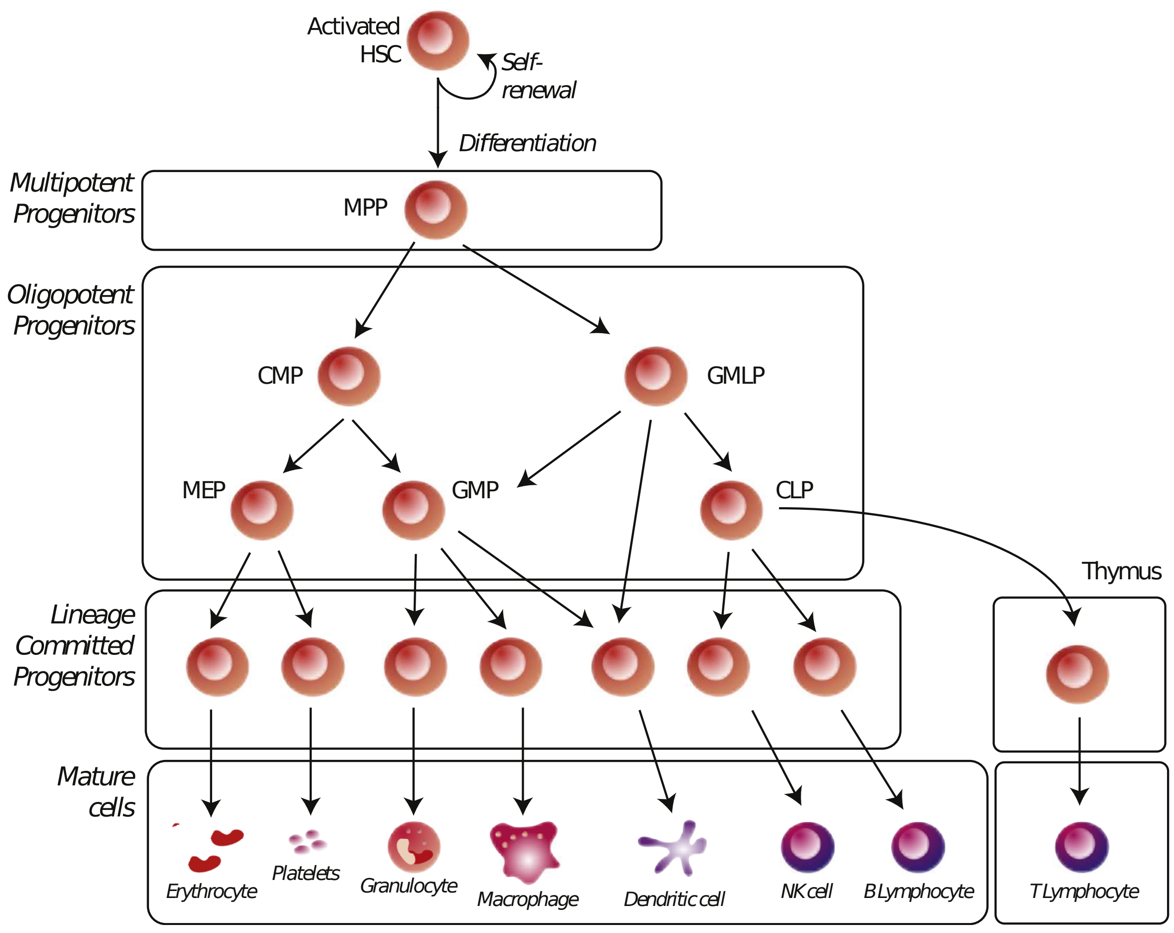


Figure 2.1: The hematopoetic hierarchy. Based on [10] figure 1.

ate into oligopotent progenitors. These can only develop into one cell type, but they can still proliferate to generate numerous cells of one lineage.

MPPs split in myeloid and lymphoid cells which are called *common myeloid progenitors* (CMPs) and *common lymphoid progenitors* (CLPs). The CMPs can differentiate into *megakaryocyte-erythrocyte progenitors* (MEPs) and *granulocyte-macrophage progenitors* (GMPs). On the other hand CLPs give rise to *natural killer cells* and *B-* and *T-lymphocytes*. On the bottom of the hierarchy *erythrocytes* (red blood cells) and *platelets* can be found, which are the progeny of MEPs. In contrast the GMPs develop into *myeloid cells* and *dendritic cells*.

The whole system is highly regulated to maintain the appropriate mixture of cell types in blood. The main work is done by the lineage restricted progenitors which are able to divide fast and produce much progeny.

2.1.1 Markers for Cell Populations

Stem cells are defined functionally through their multipotency and self-renewal which can be shown by transplantation experiments [13]. With these it was possible to identify surface markers which allow for prospective identification of HSCs in a heterogeneous cell mixture. These markers lead to a proper reliable and reproducible extraction of stem cells from bone marrow. However the purity is only at 50% [14].

There are many surface markers which can be used to distinguish between different cell types. They are collected as *clusters of differentiation*, numbered and abbreviated as e.g. *CD150*. *Lineage markers* are markers on cells which are already differentiated. Therefore HSCs are negative for these markers. Another marker is called *c-kit* which is in fact CD117 and is mainly expressed in HSCs. The third important marker to mention here is *Sca1* (stem cell antigen-1) which exists in immature progenitor cells. Also many other CD markers like CD34, CD48, CD105 and CD150 can be used to identify cell types in the hematopoietic hierarchy [15].

2.2 Fluorescence Activated Cell Sorting

To detect cell markers in a high-throughput way *fluorescence activated cell sorting* (FACS) is often applied. Machines using this technique are able to measure up to 18 markers for each cell very fast. Cell numbers of up to millions are not uncommon. In principle it is a flow cytometry with the ability to sort afterwards [16].

Cells have to be marked with some fluorescence markers, e.g. antibodies for cell surface antigens or fusion proteins inside the cell. The fluorochromes are then activated by lasers at a specific *excitation wavelength*. The activated molecules emit light at their specific *emission wavelength* which is captured by photo detectors and the measured intensity is sent to a computer. Each detector is also called *channel*. In addition the forward scatter and side scatter are measured and recorded. Forward scatter correlates to size and side scatter to granularity. In the end a dataset which contains an entry with the measured channel intensities is obtained for each cell [8]. A schematic design of a FACS machine can be found in figure 2.2.

2.2.1 Compensation

As the emission spectrum of fluorochromes is not a sharp peak at the given emission wavelength but a spectrum around that peak there is an overlap between nearby channels. This effect is called *spill-over*. To remove that effect the acquired data has to be *compensated* [8].

Therefore each fluorochrome is measured separately - without being influenced by others - on all channels to determine the influence on that channel. With the assumption

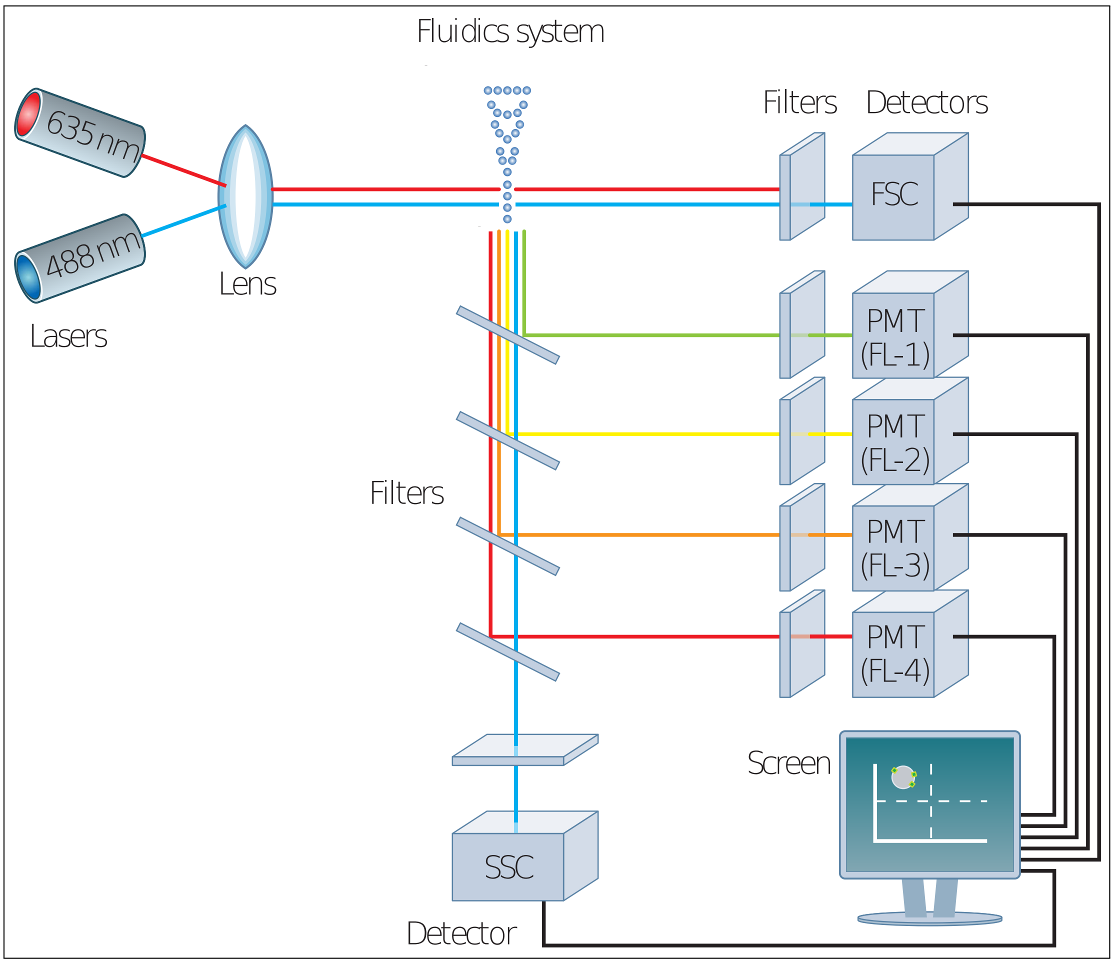


Figure 2.2: General scheme of a FACS machine. The fluorochromes are activated by different lasers, the emitted signal is then routed through several filters before it is measured by detectors. (From [16] figure 3))

that the real intensities are linear combinations of measured intensities the reference measurements can be combined into a *compensation matrix*. With this matrix the actual compensation is a matrix multiplication with a datapoint.

This can result in negative values which seems counter-intuitive but can be explained by the configuration of the FACS machine. For each experiment it is adjusted to record 0 intensity on an unlabeled sample. This is required because of auto-fluorescence of the analyzed cells. So negative values correspond to less fluorescence than the auto-fluorescence of the control.

Today it is common to store the raw data before compensation in data files. It is also possible to save the compensation matrix itself in those files, but this is not always the case. So when working with FACS data the first step after loading from files is to apply the compensation [17].

2.2.2 Visualization

The compensation step is followed by the visualization for which the data has to be adjusted to allow for an optimal visualization. This process is also called *transformation*. The standard way is to show scatter plots with two channels on the axes. There are two challenges which have to be solved in order to view the data in an appropriate way. First, the data range is very high and spans several orders of magnitude. Second, many values can be close to 0 or even negative (due to the compensation).

A logarithmic scale in plots can solve the first problem of the large range (see figure 2.3(b)). The large range is compressed into a smaller one which is easier to interpret for humans. However, this scale is not suitable for values near 0 and not applicable for negative values, because the logarithm of 0 and negative values is not defined. To overcome this problem one can try to use linear scales, which can handle negative values and show values near zero well (see figure 2.3(a)) but the large range is not covered in this approach.

To overcome both problems a new scaling method called *logicle* scaling was developed [18]. It is based on the reverse hyperbolic sine function (*arcsinh*) with an additional *cofactor* for adjustments. The arcsinh function combines the advantages of linear and logarithmic scaling. It is symmetric, linear around zero and logarithmic for larger values (see figure 2.3(c)). Therefore it can handle negative values as well as positives.

The actual transformation formula T for a value x is

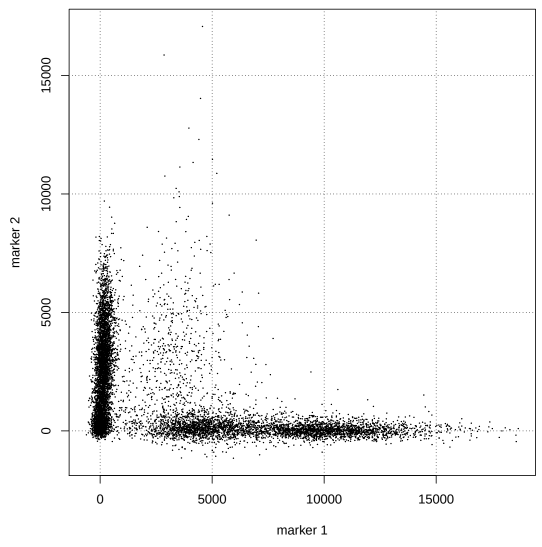
$$T(x) = \log\left(\frac{x}{C} + \sqrt{\left(\frac{x}{C}\right)^2 + 1}\right) \tag{2.1}$$

with C as the transformation cofactor. The cofactor controls the size of the linear range around 0 (see figure 2.3(d)). A further generalization of the hyperbolic sine function with five instead of one parameter yields in the so called *biexponential* transformation.

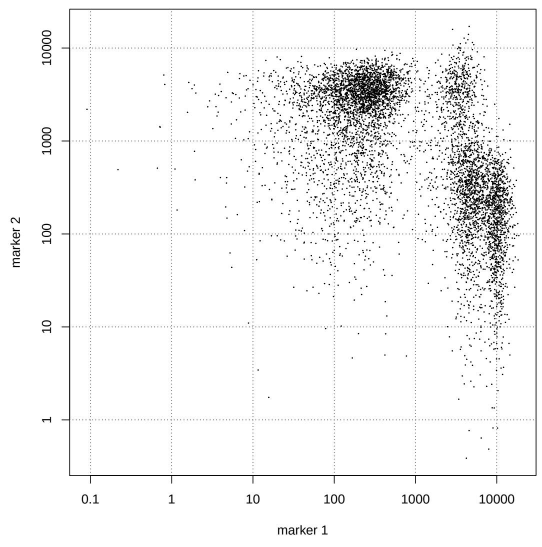
2.2.3 Gating Analysis

The main goal of FACS analysis is to identify different cell populations and sort them to make experiments with clearly defined populations. The normal way is *gating* which is a repeated selection of different parts of the data based on visual filters drawn to scatter plots.

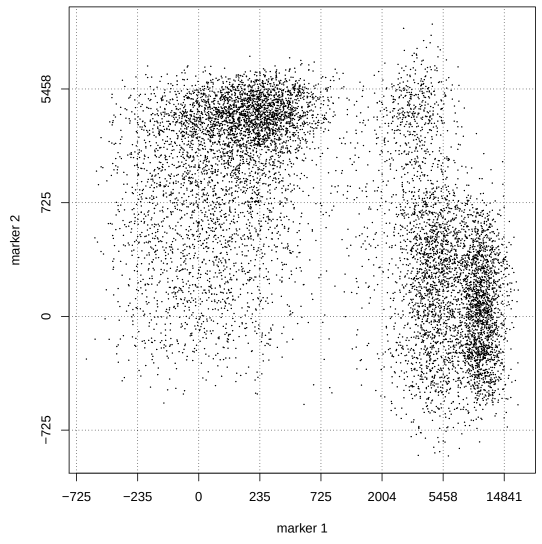
The user starts with all cells, looks at a scatter plot of the two most interesting markers and selects visually a population to continue with. The visual selection is done most of the time through a square drawn to the plot, but there are other possibilities like polygons or hand drawn lines. This selection defines a first population which is then visualized in a



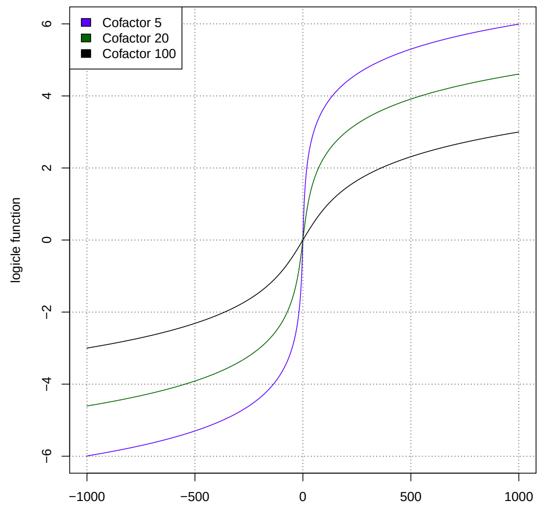
(a) Linear scale



(b) Logarithmic scale without values ≤ 0.



(c) Logicle scale with cofactor 100.



(d) Logicle scaling function with different cofactors.

Figure 2.3: Logicle scale is required for an appropriate display of FACS data. (a)-(c) Scatter plots with two markers of a FACS measurement are displayed. (d) Logicle scaling function with different cofactors.

new scatter plot with different markers on the axes than before. This procedure resulting in a *gating hierarchy* is repeated until the desired populations are found.

2.3 Time-Lapse Movies of Hematopoietic Cells

While FACS analysis is a useful experimental technique for identifying and sorting cell populations, it has some major drawbacks when it comes to analysing dynamic processes such as differentiation. Only single timepoints can be obtained through FACS analysis and repeated analysis at different timepoints does not allow for the correct identification of individual cells in both datasets. There is no link between the two measurements which allows for a tracing of individual cells.

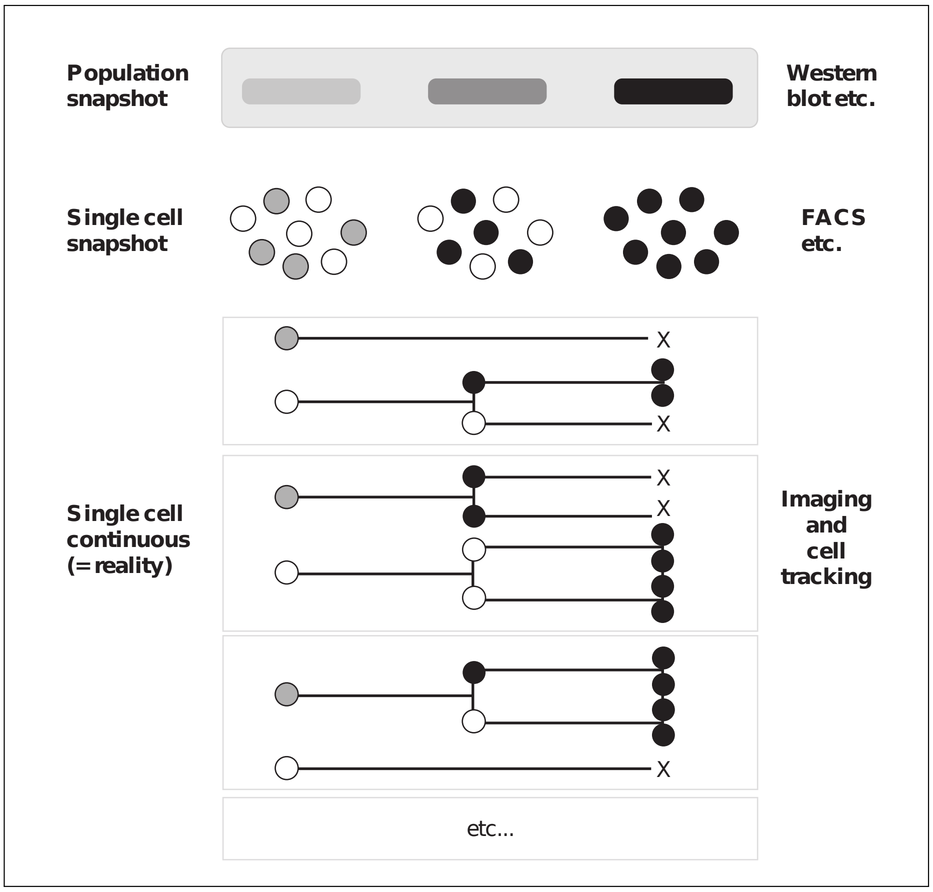


Figure 2.4: Only continuous observation of single cells reveals the reality. Population snapshots do not discover the heterogeneity. Single cell snapshots can not connect cells from different timepoints. (From [3] figure 1)

To overcome this problem movies on the single cell level have to be acquired [19]. This continuous observation of cells allows a considerable better analysis (see figure 2.4). One starting cell, e.g. an HSC, can be tracked through all its daughters until the cells are fully differentiated. In between fluorescence markers are measured which allows for better insights into the decision-making process.

Nevertheless it is a complex technology and requires a good interaction of several different components like microscope, light sources and motorized stages. For a seamless integration of all steps custom made software is needed since no commercial offers are available [3].

The post-processing step following the recording of the movie requires efficiently designed custom software. Cells on different frames of the movie have to be connected if they represent the same cell (*tracking*). Although there are attempts to automate this process it is currently too error-prone to be used without human interaction [3]. So the main work has still to be done manually which is the major bottleneck in this approach. The result of the tracking process is a cell tree which represents the relationships (mother, daughter) between the different cells.

As the last step of the data acquisition the data has to be annotated manually. This can be for example different cell types or the expression of different fluorescence markers. Also quantification of these markers is possible and leads to time courses for each cell. We call the final result *annotated stem cell trees*.

The usage of this movie approach increased over the past few years in hematopoiesis research. Despite many challenges movies helped already answering decade old questions in hematopoieses. For example it was found that cytokines can instruct lineage choice [20].

Chapter 3

Identification of Redundant Markers in FACS Data

As described in section 2.2, FACS analysis is today a standard tool for analysing cells. Many different markers are used to find cells of interest and investigate their development. The question which may arise is which markers are really required and which markers are redundant. While the answer to this question is easy if the dependency is linear and can therefore be seen in a simple scatter plot it is hard to determine an answer if the redundant information is spread in a non-linear way over different other markers. An artificial dataset with two obvious redundant markers is shown in figure 3.1. The goal of this chapter is to construct and implement a procedure to find out which markers are relevant.

The knowledge about redundant markers can be helpful in several ways. It saves money, because the markers can be omitted in experiments and have not to be purchased. Also the now unused colors or wavelengths can be used by different and maybe more interesting markers in FACS measurements. Besides these practical advantages it can give new biological insights into the investigated system. For example it is possible that the markers are co-regulated and activated by other analysed markers.

Our general idea is to use a clustering algorithm which was specifically developed for FACS data (SPADE) [21], run it with different marker combinations and determine which of them is the best. Going through a clustering to make a statement about the markers themselves can first seem not straight-forward but has the large advantage to find also non-linear dependencies opposed to methods which use the data directly. The comparison of these clusterings checks if the global hierarchical structure of the data is preserved and therefore we can state that the used markers are enough to explain the data.

Thus our contribution in this chapter is two-fold. First we derive a distance measure to quantify the similarity between the results of two hierarchical clusterings. This is an extension to standard metrics such as the F-score [22] which quantifies similarities

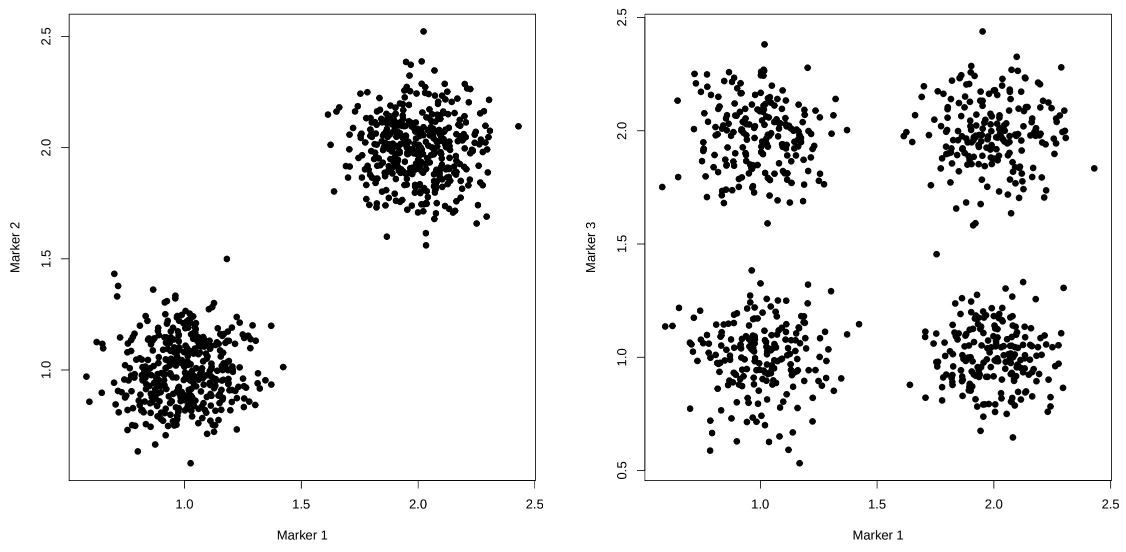


Figure 3.1: Artificial dataset with four cell populations and two redundant markers "Marker 1" and "Marker 2" and a non-redundant marker "Marker 3".

between results of two strict partitioning clusterings (e.g. k-means) where each datapoint belongs to one cluster only. Second, we propose a marker selection algorithm based on this measure to identify redundant markers in a FACS analysis.

3.1 The SPADE Algorithm

The goal of SPADE [21] is the identification and visualization of cell populations in FACS data. In contrast to other cluster algorithms which are density based, it is also able to discover small populations (with a low density) which will be easily overlooked in standard analysis methods such as scatter plots and density maps.

3.1.1 Algorithm

SPADE treats the cells as points in a high dimensional space. It always uses the *L1-norm* (also called *manhattan distance*) as distance measure d_{L1} between cells. That means for two datapoints $p_1 = (x_{1,1}, x_{1,2}, \ldots)$ and $p_2 = (x_{2,1}, x_{2,2}, \ldots)$ with values x for each marker the distance d_{L1} is defined as

$$d_{\mathrm{L1}}(p_1, p_2) = \sum_i |x_{1,i} - x_{2,i}| \,. \tag{3.1}$$

The usage reflects the fact that FACS data have no geometrical representation and therefore the normal Euclidean distance is not suitable here.

Before the algorithm starts all data is transformed to be in logicle scale (see section 2.2.2) to take the special properties of FACS data (exponential scale, negative values and values near zero) into account. All following steps including the distance measure use only the transformed data.

The main algorithm consists of five main steps visualized in figure 3.2. First a down-sampling is done to get a similar density over the whole dataset. Then the dataset is clustered into a user-defined number of clusters and a minimum spanning tree with the clusters as nodes is constructed. As last steps the original dataset is integrated into that tree and visualised in a 2D plane. So the result is more than just a clustering, it is a clustering visualized for human inspection in a minimum spanning tree

Density-Dependent Down-Sampling

The first step equalizes the density over the whole dataset $D = \{p_1, p_2, \dots, p_n\}$ with n datapoints. Dense regions which corresponds to frequent populations are highly down-sampled and sparse regions are not touched at all. For this the user has to specify a *target density* TD to which the dataset will be down-sampled. A common value is to choose the 5% percentile of the density distribution, which allows an automated adjustment to the specific dataset. In addition an *outlier density* OD is defined to remove outliers. A typical value here is the 1% percentile. All cells with a lower density are regarded as outliers and removed before the next step.

The *local density* LD_i for a cell i is defined as

$$LD_i = \sum_{j=1}^{n} I(d_{\mathrm{L1}}(p_i, p_j) \leq T) \tag{3.2}$$

with I being the indicator function and T the size of the neighbourhood. The density is then the number of cells in the neighbourhood. The range T is approximated at the beginning so that on average each cell has at least five neighbour cells. After the density calculation the density value is used for the actual down-sampling. Each cell with a lower density than the outlier density is removed, each cell with a lower density than the target density is not touched. Each other cell is kept in a way that the expected density is TD. So the keep probability kp_i for a cell i is

$$kp_i = \begin{cases} 0 & 0 \leq LD_i < OD \\ 1 & OD \leq LD_i < TD \\ TD/LD_i & TD \leq LD_i \end{cases} . \tag{3.3}$$

Round-Based Agglomerative Clustering

The second step clusters the cells into a user defined number of clusters. The algorithm is a modified version of *agglomerative clustering* with *single linkage* [6]. This means the

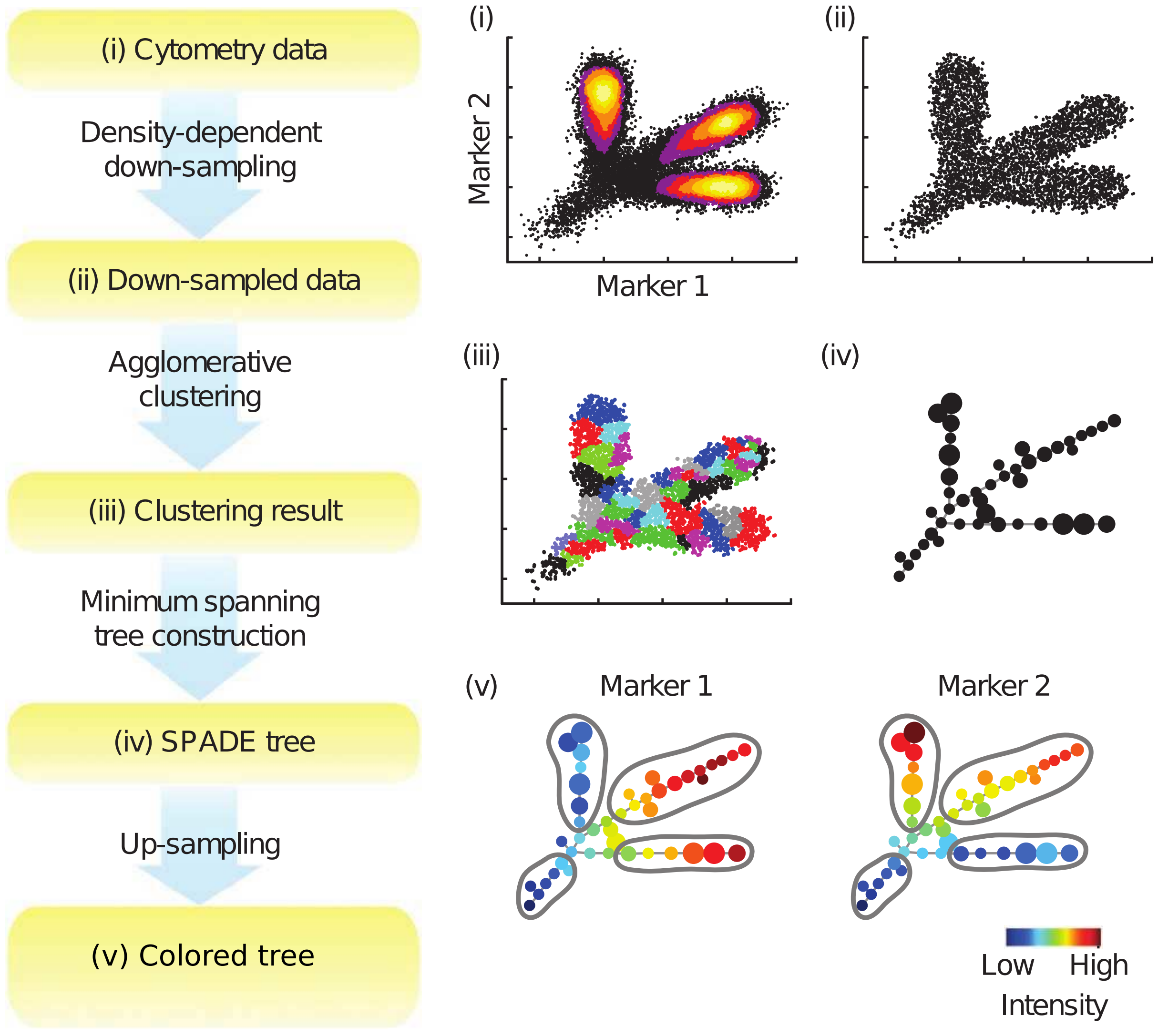


Figure 3.2: Workflow of SPADE visualizing the main steps and their results. (Taken from [23] figure 1)

distance D between two clusters c_1 and c_2 is

$$D(c_1, c_2) = \min_{p \in c_1, p' \in c_2} d_{\mathrm{L1}}(p, p'). \tag{3.4}$$

The clustering algorithm has a new constraint which restricts the merging in each round to only clusters which have not been merged in that round before. This results approximately in same-sized clusters and prevent problems like the single-link effect which occurs often when single linkage is used and creates only one large cluster which is extended by one new cell each round.

At the beginning each cell forms its own cluster. Then a cluster is randomly chosen and merged with its nearest neighbour cluster (single linkage). The merge is only allowed if the other cluster has not been merged before in the current round. This is repeated until all clusters have been examined. After that the current round ends and a new round starts. The procedure stops when a user defined number of clusters is reached. There is an additional step to remove outlier cells which are not contained in any cluster after five rounds. They will not be part of the clustering at all.

Minimum Spanning Tree Construction

Then the clusters are put into a *minimum spanning tree* (MST), which means the clusters are connected such that a tree arises and there is no other way to connect them which results in a smaller sum over all used distances. This is done by the *Boruvka's Algorithm* [24]. Each cluster represents a node in that tree and has the medium marker expression levels of all contained cells. This step tries to recover the overall structure of the data cloud and put it into the tree.

Up-Sampling

To visualise the clusters accurately all cells from the original (not down-sampled) dataset are assigned to that cluster which is assigned to their nearest neighbour in the down-sampled set. So for each cluster the number of cells and the median marker expression levels can be calculated.

Visualization

In the last step the constructed minimum spanning tree is laid out in a 2D plane. For this a modified *Fruchterman-Reingold Algorithm* [25] is used. It first determines the longest path through the tree and displays it on a parable. Then all remaining branches are put there in a way that the distances between the branches are maximised. In addition nodes are sized corresponded to the number of cells belonging to that cluster and coloured according to the median of a user selected marker.

3.1.2 Discussion of Important SPADE Properties

In the following a number of important properties of the SPADE algorithm will be discussed.

Clustering and Spanning Tree Construction are Closely Related

There is a non-trivial large similarity between the two main steps of SPADE, the hierarchical clustering and the minimum spanning tree construction. Both use the L1-norm together with single linkage as similarity measure between clusters. We show that both algorithms use the same edges to construct their result trees and therefore are equal.

For a minimum spanning tree the so called *cut property* [26] holds. If we look at all edges connecting two disjunct subgraphs (which together result in the complete graph) the edge with the lowest weight has to be in the minimum spanning tree. The reason is that if the edge is not in the MST, it can be replaced with the minimal edge connecting both subgraphs. This results in a tree with a lower total weight which is a contradiction to the fact that the tree was previously a MST.

In the clustering step a cluster can only be merged with its nearest neighbour. That means a cluster C is only merged with its neighbour if the edge connecting both is the shortest edge starting at cluster C. As we have seen above that edge is part of the minimum spanning tree. So both algorithms are using exactly the same edges and therefore construct the same result tree although in a different order. Of course if there are two nearest neighbours at the exact same distance the results can vary.

Nevertheless there are two slight differences in both algorithms. The dendrogram, which is the result tree in the clustering step, has a root and therefore a direction while the minimum spanning tree does not have this information. The second point is that "singletons" are removed by the clustering algorithm in round five. This is not considered in the argumentation above and therefore both algorithms are not exactly the same.

Taken together SPADE can be viewed as an agglomerative clustering which replaces clusters at a user-defined point with their current median values and visualizes only the dendrogram above this threshold.

Results are Non-Deterministic

The down-sampling and the clustering step use randomness, so if *SPADE* is run twice with the same data and the same parameter set the result will be different (see figure 3.3). This is especially true for the minimum spanning tree as a main SPADE result. It is not really clear how stable it is and whether small changes in the input can change the shape of the spanning tree considerably.

The supplementary material of [23] section S3 shows a stability analysis performed through the addition of noise to the data. It is stated that there are only little changes in the results but are in general still good. But it can still confuse the user and lead to irritation how to analyse and how to interpret the result tree.

Visualisation is Arbitrary

In a simplified view the used visualisation maximises just the distance between nodes. That means the position and orientation of clusters on the plane have no real information. The only information given are the edges, which are part of the minimum spanning tree,

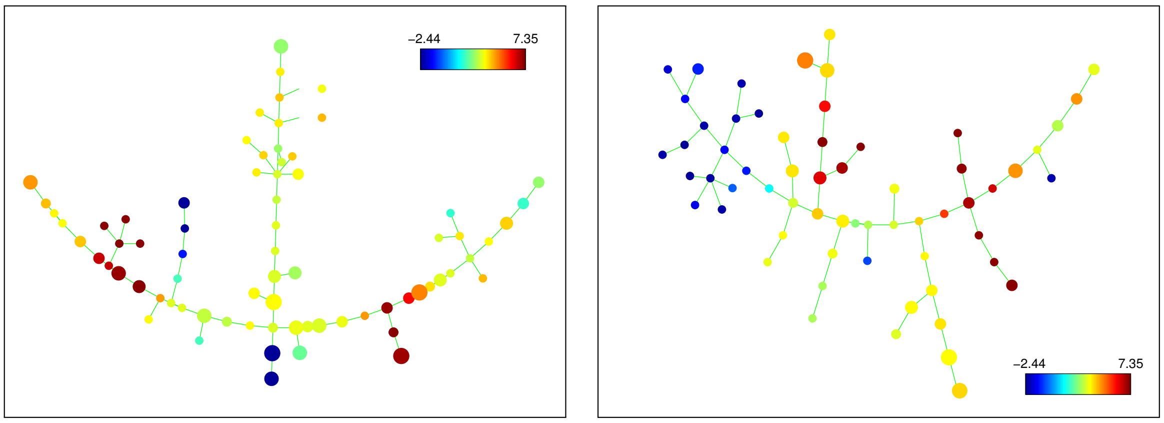


Figure 3.3: SPADE results vary. Two SPADE result trees resulting from two independent SPADE runs on exactly the same data and parameters are shown. The nodes are coloured by the same marker.

and the colour and size of nodes. There is no representation or visualization of distances other then these edges.

3.2 Comparison of Cluster Trees

To be able to test which markers are redundant we need to compare two hierarchical clusterings against each other and test if they are *similar*. We use the minimum spanning tree of clusters created by SPADE for comparison. It is equivalent to the upper part of a dendrogram cut at a specific height but without a defined root. For the actual comparison we define a *distance measure*.

3.2.1 Distance Measure

The measure should include both, the clusters itself and also the tree structure to compare the results. In addition SPADE does not always produce the same number of clusters. The cluster parameter is rather a hint when to stop than a hard limit. This has to be taken into account when constructing the measure. In classical approaches one compares either trees or clusters, but not both at the same time. So classical performance measures like *purity* [22] or *F-measure* [22] can not be directly applied here.

A tree of clusters is at tree $T = (C, E)$ of clusters $C = \{c_1, \ldots, c_{n_T}\}$. The n cells are distributed over all clusters $c_i \subset [1; n]$. To be a clustering each cell has to be in exactly one cluster:

$$\bigcup_{i=1}^{n_T} c_i = [1; n] \quad \text{and} \quad \forall i \neq j : c_i \cap c_j = \emptyset \tag{3.5}$$

Mapping of Trees

The idea is to map the clusters from one tree T' to the other tree T and calculate how far away the cells are in those mapped clusters from their real position in T.

The first tree $T = (C, E)$ is used as "reference tree" while the other tree $T' = (C', E')$ is compared against that reference. Each cluster c' in T' is mapped to some clusters in the reference tree T. Let M be the matrix containing the number of cells which are both in c_i and c'_j

$$M = (m_{ij}) \quad \text{with} \quad m_{ij} = \left| c_i \cap c'_j \right|. \tag{3.6}$$

The mapping has not to be one to one. One cluster can be mapped to different clusters in the reference. In the following we define two mappings, the first mapping M_1 maps a cluster c'_j to exactly that single cluster c_i which contains the most of its cells:

$$M_1(j) = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n_T} \end{pmatrix} \quad \text{with} \quad x_k = \begin{cases} 1 & \text{if } k = \arg\max_i M_{ij} \\ 0 & \text{else} \end{cases} \tag{3.7}$$

The second mapping M_2 maps one cluster c'_j to all clusters c_i which contain cells of c'_j. The mapping is proportional to the number of cells which are shared between both clusters.

$$M_2(j) = \frac{1}{\left|c'_j\right|} \begin{pmatrix} m_{1j} \\ m_{2j} \\ \vdots \\ m_{n_T j} \end{pmatrix} \tag{3.8}$$

After the mapping the shortest path between the mapped location and the original location of each cell is calculated. We define the shortest path vector $S_T(c_j) = (s_{ij})$ for a cluster c_j with s_{ij} to be the length of the path to cluster c_i in T (there is only one path because T is a tree). We use the average of all shortest paths between the original and mapped clusters as distance measure for cluster trees. So we define the distance d_1 between T and T' as

$$d_1(T, T') = a \sum_{i=1}^{n_T} b_i \sum_{j \in c_i} M_1(j) \cdot S_T(c_T(j)). \tag{3.9}$$

with a and b_i as weight factors, $\cdot$ the dot product and $c_T(j)$ the cluster of datapoint j in T. In the same way d_2 is defined with M_2 instead of M_1. The first sum goes over all clusters in T, the second one over the cells in the current cluster of T.

The weight factor a and b_i control if the averaging is done per cluster or only once for the complete set. For now we leave the weight factors at $a = n^{-1}$ and $b_i = 1$. This transforms d_1 into the average shortest path of cells to their mapped partners.

The measure can also be interpreted as the number of “moves” of a cell from cluster to cluster which has to be done to turn T' into T. Although it is not clear that it is the smallest possible number for that.

Both definitions d_1 and d_2 are not symmetric, so $d(A, B)$ is not the same as $d(B, A)$, which means they are not a metric. This can lead to some problems if used in a different context but here it does not make any problems because we will use it actually in a way that there is a reference. Another thing to mention here is that the structure of the second tree is not included in this measure. So it can happen that the structure of the second tree is very different from the reference tree while the clusters themselves are not. In that case the measure will report a small distance instead of a large one.

Symmetric Measure

To overcome both problems we extend the measure to make it symmetric and also to take the structure of the second tree into account. In contrast to the measures described above, this version calculates the mapping not only from T' to T but also in the opposite direction. Then both mappings are used as edges in a combined graph of both trees T and T' which connect the mapped nodes in both trees. Note that this only makes sense for mapping M_1 where each node has exactly one partner which can be translated into an edge.

So we look at the graph $G = (C \cup C', E \cup E' \cup E_m)$ which consists of the two original trees T and T' and their connecting edges

$$E_m = \left\{ (c_i, c'_j) \in C \times C' : (i = \arg\max_k M_{kj}) \vee (j = \arg\max_k M_{ik}) \right\}. \tag{3.10}$$

Again the shortest path length between nodes in T and T' is computed and for each cluster the shortest paths for its cells are calculated. So we get

$$d'_3(T, T') = a \sum_{i=1}^{n_T} b_i \sum_{j \in c_i} S_G(i, c_{T'}(j)) \tag{3.11}$$

with $S_G(i, j)$ being the shortest path between c_i and c'_j in the combined graph G. For now this definition is symmetric because b_i is always 1 and therefore the two sums run exactly over all cells. This will change later if we choose different b_i values. To be prepared for that and always work on the same formula we define d_3 here as

$$d_3(T, T') = \frac{d'_3(T, T') + d'_3(T', T)}{2}. \tag{3.12}$$

While all measures are non-negative ($d(x, y) > 0$) and fulfill the identity of indiscernibles ($d(x, y) = 0 \Leftrightarrow x = y$) the first and second measure are not symmetric

$(d(x,y) = d(y,x)))$ and are therefore not a metric. Also the d_3 measure is not a metric because it was found empirical not to satisfy the triangle equation $(d(x,y) \leq d(x,z)+d(z,y))$. This restricts the usage of our measure in other areas but is no problem here because we do not require a metric at all.

Adjustments

SPADE results do not always have the same number of clusters. As described earlier, the correspondent parameter is rather a hint than a strict setting. But the length of a shortest path correlates with the size of a graph. In general the height of a random tree is proportional to the harmonic numbers and therefore proportional to $\sqrt{2\pi n_T}$ [27]. So we adjust all measures to a reference tree with n_{ref} nodes by multiplying a with $\sqrt{n_{ref}/n_T}$.

To prevent different cluster sizes from having an effect on the measures we adjusted for that, too. This can be done simply through setting $a = n_T^{-1}$ and $b = |c_i|^{-1}$.

Further analysis of the shortest paths in the combined graph used in d_3 revealed that some paths "jump" several times between both subgraphs. This is not really intended and therefore we restricted the paths to only "jump" once between both trees. To implement this the weight of an edge connecting both parts is set to a very high value. A shortest path has to use one of these edges exactly once, because the start and end point are on the different parts of the graph. Afterwards the resulting length is readjusted by subtracting the very high value.

3.2.2 Reference Distribution

Due to the fact that SPADE is non-deterministic, it produces different results although it was run with the same parameter set and same data. However the reference is very crucial for the selection process as each new tree is compared against that reference tree. So it is better to use a distribution of trees instead of a single one. This allows us to distinguish between the inherent variations of SPADE and real variation caused by different markers.

The distribution is created by running SPADE r times with the same parameters on the same data. Each created result tree is compared against all others, so we get a total of $r(r-1)$ distance values. This is the empirical approximation of the real measure distribution which represents here the *null-distribution* F_0.

To compare a new tree T' against that distribution, T' is compared against all r reference trees and the resulting distance values are used as *sample distribution* F_S. Now we are able to test if the sample distribution and the null-distribution are drawn from the same distribution (null hypothesis). This is done by the *Two-Sample Kolmogorov-Smirnoff-Test* [28]. It is a frequently used non-parametric test to check if two samples are drawn from the same distribution. We test if the distributions are equal so a high p-value is considered as a positive result. The test statistic is

$$D = \sup_x |F_0(x) - F_S(x)| \tag{3.13}$$

with F_0 and F_S as the empirical distribution functions. The null hypothesis is rejected if

$$rD > K_\alpha \tag{3.14}$$

with K_α being the critical value from a table.

Besides the advantage of catching the SPADE inherent variation this approach allows for the calculation of p-values. These values are much easier to interpret than the raw distance values. They allows for a standardised comparison of results. Also the threshold used in the selection process can be set much more reliably. We use the widely used threshold of 5%.

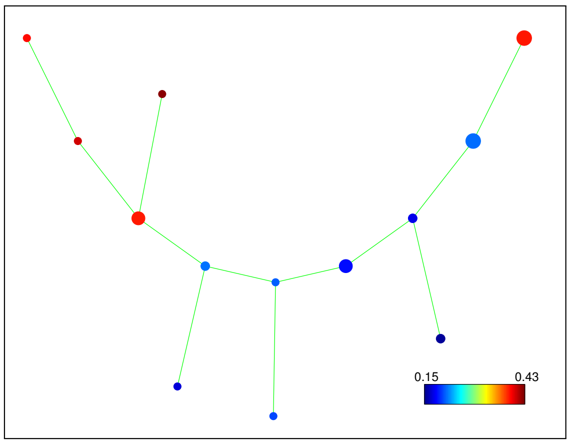
An example of different marker combinations compared to a reference distribution based on an artificial dataset is shown in figure 3.4.

3.3 Identifying Informative Marker Subsets

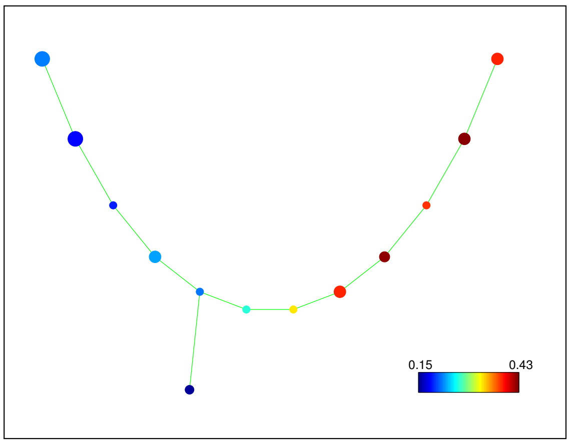
We are now able to compare cluster trees against each other. With this we can address the problem of finding redundant markers. The idea is to run the hierarchical clustering several times with different marker sets and compare the results to find the smallest still sufficient marker subset. The first run is with the full set of possible markers to create a reference distribution. All subsequent runs have then a reduced number of markers. In each step the decision if a marker is removed or not is based on the distance measure between the reference and the one created by the current run.

This approach assumes two things about SPADE and the distance measure. The first is that all markers together produce, in terms of biological meaning, the *best* possible result. But that is not really clear, e.g. if a marker only contains noise then a result constructed with all but that marker is probable better. So we consider that all markers which are used here for feature selection make sense and have a biological meaning. In addition the SPADE authors write that the algorithm is robust against some meaningless markers. Also the more markers are there the more information is available and the algorithm has a chance to create a good result.

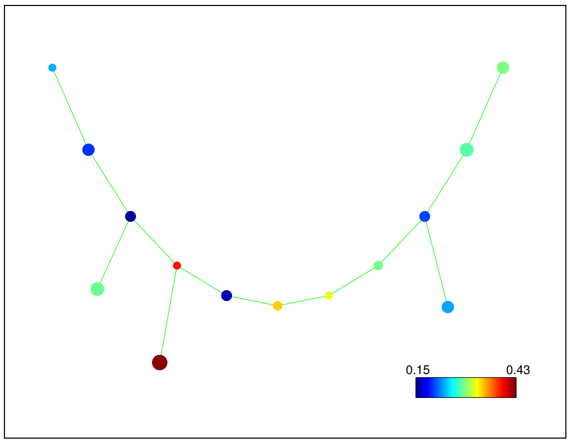
The second assumption is that the removal of one marker always increases the distance to the reference (*monotonic assumption*). That means marker combinations of less markers are worse than combinations with more markers. The assumption can be justified with the information which is available. The more markers are there the more information is available for the algorithm. But there is a problem here: as seen above SPADE results are not deterministic and vary so it can happen just by chance that a smaller set of markers is closer to the reference than the larger set. With this assumptions we can be sure that one of our proposed methods finds the best marker set.



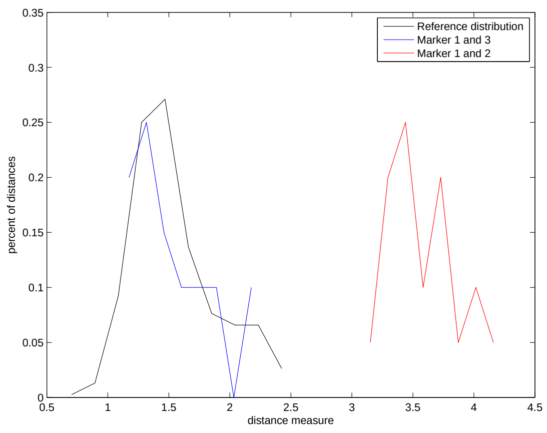
(a) Example of reference tree.



(b) Tree created with markers 1 and 3.



(c) Tree created with markers 1 and 2.



(d) Distance distributions of tree (b) and (c) against the reference.

Figure 3.4: Usage of the distance measure to determine the quality of marker combinations. (a)-(c) Clustering results with different marker sets on artificial dataset (fig. 3.1) coloured by marker 3. (d) Histogram of distance distributions.)

3.3.1 Lattice of Marker Combinations

To be able to explain different methods for the feature selection on the same structure we define here the *lattice* of marker combinations. It is a directed graph which contains all marker combinations and their relations. Let $X = (x_1, x_2, \ldots, x_n)$ be the set of available markers. Then all marker combinations $\mathcal{P}(X)$ (power set of X) are the nodes of the lattice. A combination has an outgoing edge to another node if it is a direct superset of that node. That means an ingoing edge comes in if the node is a subset of the starting node of the edge. The whole lattice L is defined as:

$$L = (\mathcal{P}(X), E) \quad \text{with} \quad E = \big\{(a, b) \in \mathcal{P}(X)^2 : (b \subset a) \wedge (|a| = |b| + 1)\big\} \tag{3.15}$$

So the graph constructed here has exactly one node with only outgoing connections. It is the node with all markers X. On the other side of the graph there is the only node with no outgoing connections which is the node representing the empty set.

3.3.2 Methods for Feature Selection

We present three different methods for marker selection. All are using the lattice and start with X as start node. They differ in the way they decide which node to check next and which nodes will never be checked. This general approach is also known as *backward feature selection* [6].

Obviously the best method would be to check all combinations and simply choose the smallest set which is still similar enough to the reference, but the computational effort is too high. There are exponential many combinations and one SPADE run lasts at least 10 minutes, but can get up to several hours depending on the dataset and computation power available.

Greedy Backward Elimination

This simple method removes each marker once and measures the remaining similarity. The marker which has the largest distance to the reference is considered the most useless marker and is therefore removed. This procedure is repeated with the remaining markers until the set becomes too different compared to the reference. The best marker set which is still good enough is the result of this method. The method is called "greedy" because it never looks back and tests if a once remove marker is maybe better combined with another set.

This is also the main criticism for this method. It assumes that the information of each marker is independent of all others. E.g. a marker is favoured for removal over two others with the same information content although it is better to remove the two ones.

Exhaustive Search

The second method is more "intelligent" and able to use markers which were previously removed from the set. It checks a marker combination for its distance to the reference if and only if all predecessors in the lattice have been already positively (small distance) tested. If just one superset is considered not similar enough the whole combination is not tested. This method is guaranteed to find the best marker combination if the monotonic assumption is true.

Many more combinations are tested here compared to the first method, because several paths in the lattice are tried and not the first “good looking” path is taken without considering others.

A variant of this is the third method which has a softened requirement for checking a new node. It is enough if only one predecessor is good to check the new node. This can be viewed as a breath-first search for good nodes in the lattice. The algorithm starts at X and checks all successors; if they are good it checks their successors and so on. Of course there are several paths to the same node and the algorithm has to prevent that one combination is tested several times.

If the assumption that the removal of a marker only increases the distance was true, which is not correct, then this variant would not be required and the normal exhaustive search method would able to find exactly the same marker subsets. If at least one superset is too distant the new set cannot be similar again because the distance has to increase.

3.4 Implementation

We have implemented the presented methods and measures in MATLAB. There are several SPADE versions available and we decided to take SPADE2, which is a newer version than introduced in [23] and has more features, better performance (some critical parts are now written directly in C) and a nicer user interface. The main part is still MATLAB code so all extensions and modifications are made in MATLAB, too. There is also a large library of ready to use functions e.g. for statistical tests and shortest path calculation.

3.4.1 Problems with the SPADE Batch Script

The batch script supplied with SPADE2 has some bugs which are significant when working with different marker sets. In the first step SPADE estimates the median distance between two cells and then calculates the size of the neighbourhood for each cell. In both steps the complete dataset with all markers is used and not only the data for selected markers. Both mistakes are corrected and only the corrected version is used in the following sections.

3.4.2 Extensions

We extend SPADE such that already computed data is automatically cached. The cache handles different marker sets and some of the parameters like number of clusters. For each marker combination the cache saves the density, the down-sampled data and the result into different files with canonical filenames. The first run of SPADE creates these files and each subsequent call loads the already computed results.

Obviously the cache speeds up the calculations considerably, because SPADE is only run if necessary. The cached results are independent of the distance measure and the feature selection method, so for different combinations of these methods the same cached data can be used.

Another extension we made is the ability to process not only *.fcs* files but also *.csv* files. The user interface was extended to load the different trees resulting from different marker combinations.

3.5 Results and Discussion

3.5.1 Dataset Description

We use three different datasets for analysis and evaluation of our described methods. The first is an artificial set of randomly generated data. The other two sets are FACS measurements of actual biological experiments.

Evaluation Dataset

The evaluation dataset consists of 10 markers and 50.000 cells. For each cell-marker combination an intensity value is chosen uniformly distributed between 0 and 1. With this clearly defined dataset it is possible to check if the expectations we have about our methods are true. E.g. a test if the adjustment for cluster counts works can be done.

Hematopoietic Stem Cell Dataset[1]

This dataset was obtained by a FACS analysis of isolated mouse bone marrow. Nine markers, forward and side scatter were measured for each cell. Two of the nine markers are fusion proteins (*PU.1eYFP* and *GATA1mCHERRY*) which consists of a target protein and an attached fluorescence protein which allows the detection. The other seven markers are antibodies against different cell surface antigens.

After compensation a preliminary gating was done on the dataset to gate out mature hematopoietic cells and to obtain the so called *progenitor gate* which only contains non differentiated cells and contains here 190.000 cells. It is defined as lineage marker negative, c-kit positive and Sca1 negative. In addition the progenitor gate is further subdivided into six subpopulations corresponding to different intermediate cell types in the hematopoietic hierarchy. For this gating CD150, CD41, CD105 and FcgR markers were used. These six gates represents the expert knowledge which we can use to test our algorithms.

[1]Provided by Philipp Hoppe, *Research Unit Stem Cell Dynamics* of Timm Schroeder at Helmholtz Center Munich

Mesenchymal Stem Cell Dataset[2]

In contrast to the blood stem cells in the previous dataset, this third dataset consists of *mesenchymal stem cells* (MSCs) which are responsible for the creation of bone. Here CD105, CD140a, Sca1 and the *fibroblast growth factor receptor* 1 and 2 are measured in addition to forward and side scatter. As the dataset before, this set is not raw data but pre-gated to exclude cells unrelated to MSCs and contains about 50.000 cells.

There is still no functional definition for MSCs meaning it is not really clear what they are. The existing studies are hard to compare because of different definitions of MSCs and different protocols used for isolation [29] [30].

The dataset does not only contain one single timepoint but several. For each timepoint the same experiment was repeated and the actual measurements were taken after different time periods. In the beginning of each experiment some cells are marked with GFP and can therefore be refound at all timepoints.

3.5.2 Cluster Count Adjustment

The first thing we will check is if our adjustment for different cluster counts is working as expected. Due to the fact that the reference distribution consists of distances between trees which have different numbers of clusters it is important to check this adjustment first.

The evaluation dataset is used here to create three reference distributions each with a different desired cluster number of 50, 100 and 200, and based on 20 reference trees. The created trees and the distance measure d_1 are used to create 6 reference distributions by comparing all against all per set once with and once without cluster count adjustment to 50 clusters.

The resulting cumulative distribution functions are shown in figure 3.5. The first three lines correspond to the adjusted distributions while the lines on the right are unadjusted. The unadjusted distributions are clearly different and can be simply distinguished by a threshold. On the other hand the adjusted distributions cannot be separated easily and the adjustment works as expected.

The differences in the adjusted and not adjusted curve for 50 clusters can be explained by SPADE. It uses the cluster count parameter as hint and stops always above this number. Therefore there are more than 50 clusters and the paths between nodes are longer than in a graph with exactly 50 clusters.

A statistical test (*Two-Sample Kolmogorov-Smirnoff-Test*) on the three adjusted distributions to check if they are drawn from the same distribution yields in a negative result and states all distributions as pairwise different. You see even by eye that especially the

[2]Provided by Dr. Daniel Coutu, *Research Unit Stem Cell Dynamics* of Timm Schroeder at Helmholtz Center Munich

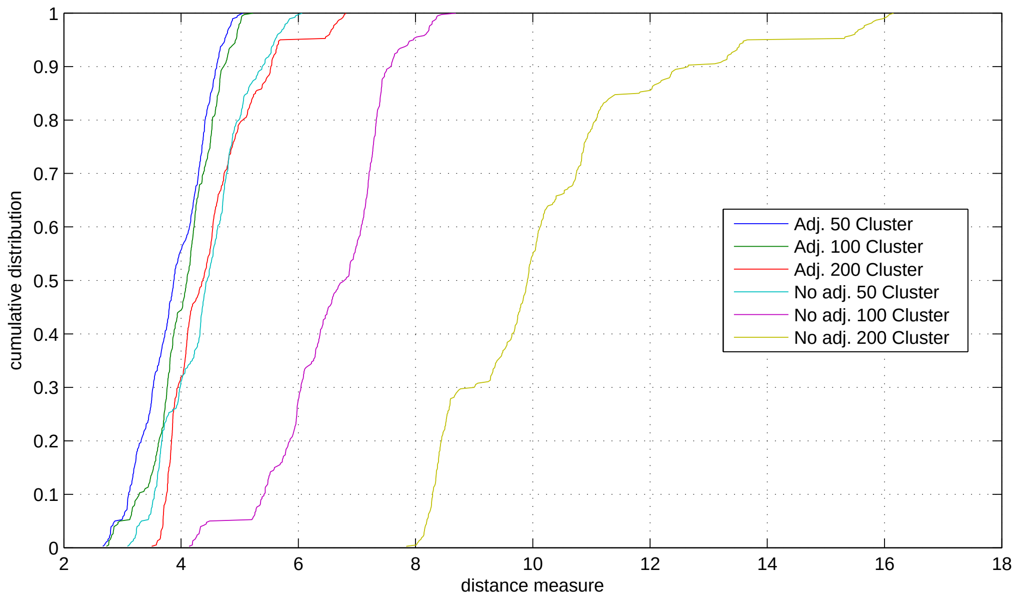


Figure 3.5: Adjustment for cluster count works. The cumulative distributions of three reference distributions (for 50, 100 and 200 clusters) once with and once without cluster count adjustment based on an evaluation dataset and distance measure d_1 are plotted.

adjusted results for 100 and 200 clusters are shifted by approximately 0.33. These remaining differences after the adjustment can be explained by the adjustment itself. We use the average *height* of trees to adjust the values but we are working with *path lengths* on trees for the distance measure. Both are similar and related but not equal.

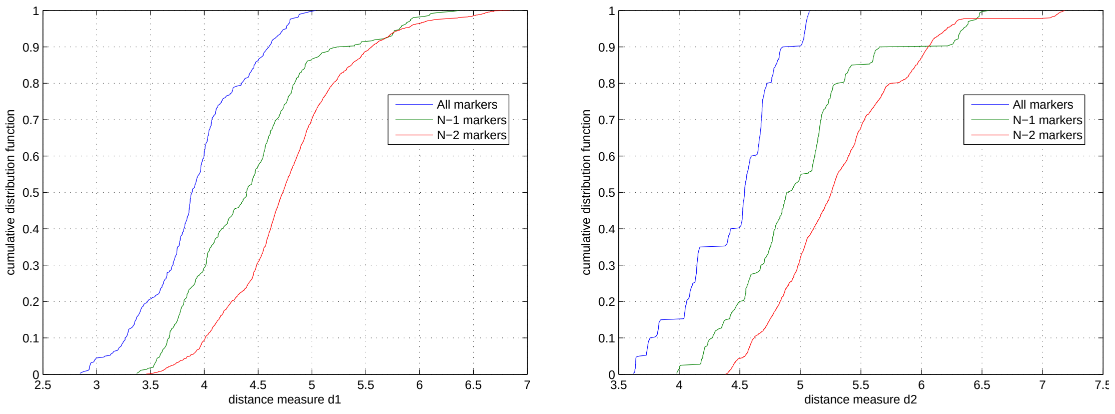
For our usage it is enough that the adjusted distributions are in the same range, because there are other random effects which make the remaining differences unimportant compared to them.

You also see the variation between different SPADE runs. Each distance in the distribution is between two result trees which are created by the SPADE runs on the same data with the same parameters. So the usage of a reference distribution and the comparison with the sample distribution is justified and required.

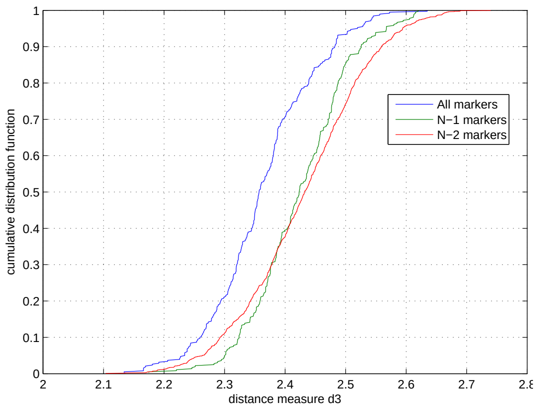
3.5.3 Distance Measure Analysis

With the same evaluation dataset as before we compare the three different distance measures. For each measure we create three different distributions with cluster count adjustment. The first with all markers which is equivalent to the reference distribution. For the second SPADE run ten times each time with one marker left out. The pairwise distances of the resulting trees are then used to create the distribution. The third one is created in the same way as the second one but with two markers removed instead of just one.

The expectation in this evaluation dataset is that SPADE finds a random structure if used with all markers. Due to the uniform creation of the set we assume that the information of the random structure is equally distributed between the ten markers. With this we expect the distance of the second distribution to the first (reference) distribution to be the same as between the second and third one.



(a) distance measure d_1 (b) distance measure d_2



(c) distance measure d_3

Figure 3.6: Distance measure d_1 is best. For a random dataset the cumulative distributions of distances between SPADE results from runs with 10, 9 and 8 markers are plotted.

The cumulative distribution functions of the distances in the three sets are plotted in figure 3.6. With d_1 as distance measure the three distributions can be distinguished well. Only for large values the green (second set) line is very near to the red one.

The plot with d_2 looks similar but the first two lines have steps. Further analysis revealed that each step belongs to one tree which is used as reference. Because d_2 does not make use of the tree structure in the second tree and the clusters in both trees are very similar the resulting distance only depends on the structure of the reference tree. This distance is represented just by the shortest paths in that reference structure. This problem also occurs with d_1 but there is more variation due to the other mapping method and is therefore not as problematic as with d_2.

With d_3 the distances are not distinguishable. In the lower distances the all marker set is equivalent with the set with two markers less but in the end it is higher than the set with one marker less. One reason for this behaviour can be that a random fluctuation in the clusters leads to a (random) new mapping which introduces a new edge in the combined graph used by d_3 and results in a shorter shortest path.

Overall we decided to take d_1 as our default distance measure and all following analysis are performed with it.

3.5.4 Marker Selection Method

To test which selection method is the best one and what are the advantages and disadvantages of each method we tested each method with the d_1 distance measure on all three datasets and compared the number of required SPADE runs and the size of the best found marker combination. The more runs are required the more time is needed and the less useful it is for usage on a daily basis. The number of markers in the best set represents how good the method is, because all methods search on the same search space and we defined the best set as the one with the least markers which is still good.

We expect that the greedy backward elimination requires the fewest runs, followed by the exhaustive search. The exhaustive search with softened requirements should require the most SPADE runs because it has the lowest barrier to test new combinations. However, the high number of tested marker combinations should result in the best subset of markers being found.

On the evaluation dataset all methods are able to find a set of only one marker which seems to be enough to explain the random structure if the data. A closer look at the mean and standard deviation of the reference distribution reveals that yet in the reference distribution there is a large dissimilarity which is caused by the down-sampling step. It randomly chooses points out of all uniform distributed points. These small variations results in different cluster structures of the remaining points. Compared with the two biological datasets with a mean of 1.0 and 1.3, the mean of 3.9 is three fold higher and the reason for finding one marker. Therefore the completely random dataset is not suitable to reveal differences in the selection methods.

The greedy method requires only 55 runs of SPADE which confirms the expectation that this method needs the fewest runs befor finishing. The two other methods require

	greedy	exhaustive	exhaustive softened
Evaluation Dataset (Mean: 3.9, Stddev.: 0.5)			
# of SPADE runs	75	1043	1043
# of markers in best set	1	1	1
Hematopoietic Stem Cells (Mean: 1.3, Stddev.: 0.3)			
# of SPADE runs	50	38	184
# of markers in best set	9	9	7
Mesenchymal Stem Cells (Mean: 1.0, Stddev.: 0.2)			
# of SPADE runs	27	27	27
# of markers in best set	7	7	7

Table 3.1: Exhaustive softened method finds the best marker set. The three marker selection methods are tested on two biological datasets. The number of required SPADE runs and size of the best marker set are reported here. For each dataset also the mean and standard deviation (stddev) for the reference distribution are given.

all 1043 runs which are 20 runs for the reference distribution and 1023 for all possible combinations. With this dataset there is no difference in both methods.

In the hematopoietic stem cell dataset the greedy method is only able to find a set with 9 out of 11 markers which is still sufficiently similar. This also shows that the order of marker removals in the greedy method is important. It is not able to find the best set with seven markers, because one important marker (Sca1) is already removed in the second round. The exhaustive method finds the same set as the greedy method but needs less SPADE runs for that. The reason is the very strict criterion for a combination to be tested. The exhaustive method with softened requirements checks the most combinations compared to the two others. The reason is that each path which can lead to a solution is tested and followed. The same reason is responsible for the good result of a marker set only consisting of 7 markers.

For the MSC dataset all methods perform the same. Each requires 20 runs to create the reference distribution and then seven runs to test the removal of each marker separately. The result is that all markers are required to maintain the SPADE tree.

3.5.5 Biological Interpretation

Hematopoietic Stem Cells

The exhaustive method with softened requirements finds a set of seven markers to be enough to explain the SPADE result tree. On the other hand the definition of the six subpopulations by an expert only requires four markers. This suggests that there is more information and structure in the data than captured through gating.

The exhaustive softened method finds three minimal sets or markers each with seven markers. Each of the three sets contains PU.1eYFP, CD41, CD150, FcgR and GATA1-mCHERRY. The remaining two markers are either side scatter and CD105, forward scatter and Sca1 or CD105 and Sca1.

Mesenchymal Stem Cells

The marker selection process does not find any redundant marker in the mesenchymal stem cell dataset. This means all markers are required to maintain the cluster structure found by the hierarchical clustering. So each marker itself has information which is not contained in other markers. As there is no definition of MSCs by marker expression all used markers can be potentially used in such a definition.

3.6 Conclusion

The marker selection algorithm uses the SPADE algorithm as basis to find marker subsets which are able to explain the whole dataset. We defined three different distance measures to compare cluster trees and determine their similarity. Through the non-deterministic property of SPADE the use of a reference distribution instead of a simple reference tree was required. Therefore the actual comparison is a comparison of distributions done by a statistical test. Using this test three search methods were proposed to find the best marker subset out of all possible marker combinations.

The evaluation reveals one distance measure and the exhaustive search method with softened constraint to be better than other combinations. It was tested on two biological datasets and not able to find a marker set as small as expected on the first set. A reason might be that there is more structure in the data than expected. On the mesenchymal stem cell (MSC) dataset we judged all markers as required to explain the internal structure, which is important because there is still no definition (in terms of marker expressions) for MSCs.

Chapter 4

Structure Preserving Visualization of FACS Data

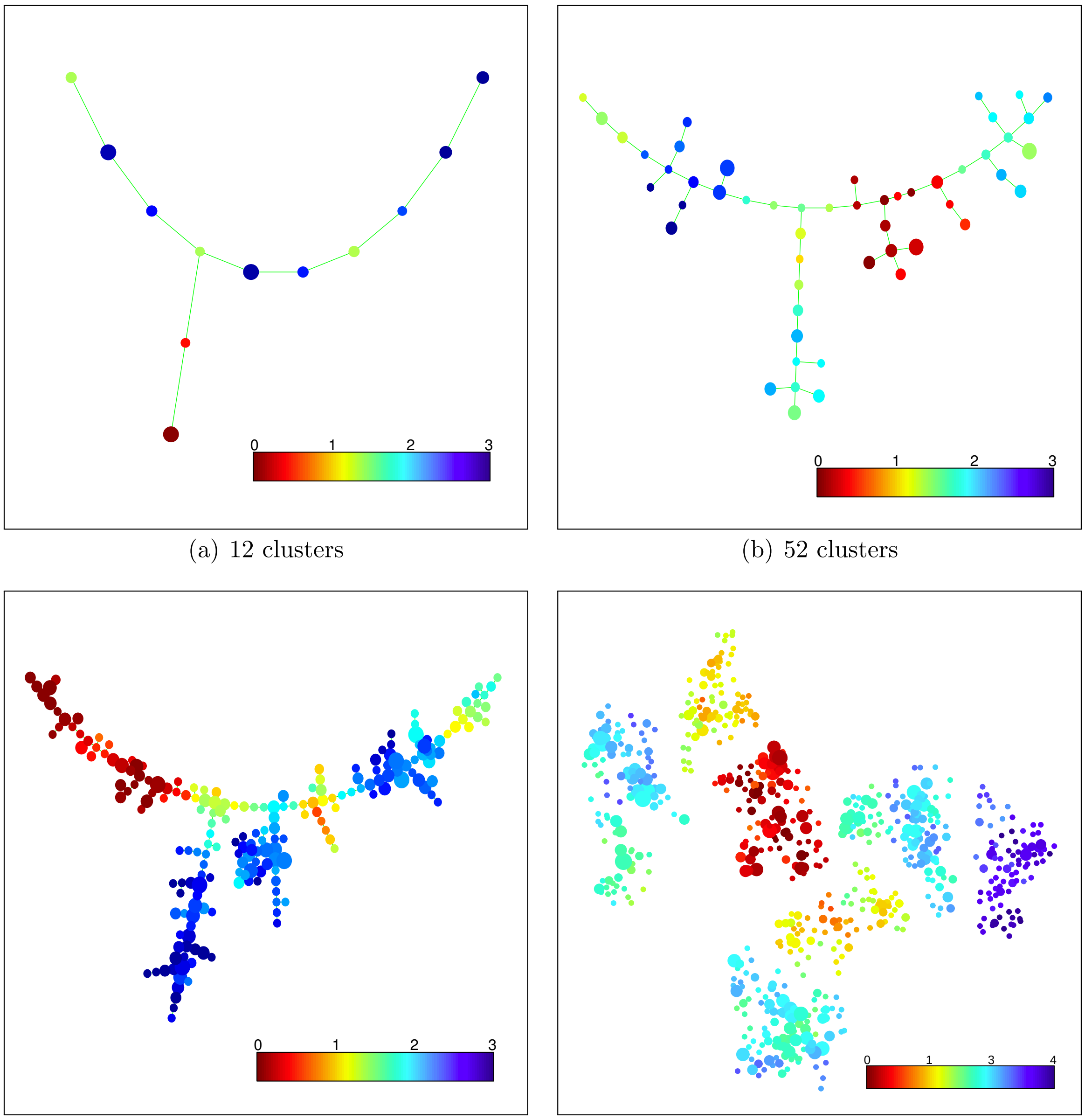
As seen in the previous chapter SPADE has some problems with its visualization while the hierarchical clustering itself is working well. The layout of the clusters is arbitrary and only the direct connections between clusters have an actual meaning of being nearest neighbours. In figure 4.1 it we shot that SPADE is not able to maintain the original distances in the data. Also it is not able to detect the seven contained clusters visually and requires the user to set the cluster count parameter manually. Therefore we implemented a new visualization for the hierarchical clustering results.

Also the default visualization of FACS data with scatter and density plots has some disadvantages. It is only possible to visualize two dimensions at the same time while the structure which has to be analysed can have many more dimensions. These dimensions cannot be displayed all together on the same plot, because there is no plot type which can visualize more than four (e.g. position in space and colour) dimensions and is easy to interpret at the same time.

So a technique which is able to project the high dimensional data into a lower e.g. two dimensional space while preserving the inherent structure of the data would increase the interpretability of the plot much. Here we choose to project the data into two dimensions, because it is a very natural choice and yet a three dimensional plot can be hard to analyse.

We use the same approach as SPADE for down-sampling and clustering of the data. The result is a dendrogram, which reflects the internal structure of the whole data cloud. Afterwards we apply an algorithm called *tree preserving embeddings* (TPE) [31], which is in contrast to e.g. PCA able to project data points into a 2D plane while preserving the dendrogram structure and guarantees that a clustering on the 2D plane produces exactly the same dendrogram as on the original high dimensional dataset. So we are able to obtain a 2D projection of the data which represents the internal structure of the dataset.

In this application we use the cluster algorithm without the cluster count parameter,



(a) 12 clusters

(b) 52 clusters

(c) 214 clusters

(d) Our visualization

Figure 4.1: SPADE has issues with visualization. Colors corresponds to distances between the cluster and a reference point located at the center of one cluster. SPADE was run on the artificial dataset (see section 4.4.1) with seven clusters. (a) with 10 desired clusters, (b) with 50 clusters, (c) with 200 clusters. (d) Result of our new visualization method on this dataset.))

because the clustering and minimum spanning tree construction are essentially the same.

We simply combine both steps and just run the algorithm until one large cluster with all cells is created.

In this chapter we show how TPE optimizes the visualization and how it guarantees the dendrogram to be preserved. Then we present how we optimized an existing TPE package to allow a faster analysis and evaluate the improved runtimes. We determine the quality of the visualization with a dataset including expert knowledge. Finally, a new approach able to visualize dynamics between different timepoints is presented.

4.1 Tree Preserving Embeddings

The input of TPE [31] is a hierarchical clustering of n objects, a distance matrix $D = (D_{ij})$ with distances between the objects and a dendrogram C which represents the clustering. $C \subseteq \mathcal{P}([1;n])^2$ ($\mathcal{P}$ means powerset) contains pairs of clusters (c_i, c_j) which are merged at some point in the clustering algorithm and therefore are children of the same parent in the dendrogram.

The goal of TPE is to find a p-dimensional Euclidean embedding $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ of the n objects which minimizes the penalty function *stress*

$$\sigma(X) = \sum_{x_i, x_j \in X} (d_{ij}(X) - D_{ij})^2 \tag{4.1}$$

with $d_{ij}(X)$ as the Euclidean distance between two objects in the embedding X and D_{ij} as the same distance but in the original high dimensional space. Without any further constraints just minimizing $\sigma(X)$ leads to the so called *crowding problem* [32] meaning there is not enough space in the embedding to preserve the intrinsic dimensionality. It is the main reason why it is hard to preserve clusters in a dimension reduction method. So TPE uses an additional constraint

$$\min_{k \in c_i, l \in c_j} d_{kl}(X) = \min_{k \in c_i, l \in c_j} D_{kl} \qquad \forall (c_i, c_j) \in C \tag{4.2}$$

to prevent that problem. The (single linkage) distances between two clusters c_i and c_j in the real data have to be the same as in the embedding. This guarantees that a re-clustering on the embedding results in exactly the same dendrogram as before, because the distances between all cluster pairs required for the dendrogram are the same as in the original data.

4.1.1 Greedy Optimization

To simplify the optimization TPE does not optimize the global problem with all cluster pairs at once, but uses the dendrogram to do this in a hierarchical fashion. So starting at the ground with the singleton clusters, each cluster pair (c_i, c_j), with the previous embeddings X_i and X_j, is embedded by solving the optimization problem

$$\begin{aligned} T* = \underset{T \in E(p)}{\arg\min} & \sum_{k \in c_i, l \in c_j} \left(d_{kl}(T(X_i), X_j) - D_{kl}\right)^2 \\ \text{s.t.} \quad \min_{k \in c_i, l \in c_j} d_{kl}(X) &= \min_{k \in c_i, l \in c_j} D_{kl} \quad \forall (c_i, c_j) \in C \end{aligned} \tag{4.3}$$

where $d_{kl}(X_i, X_j)$ is the distance of cell k and l in the embedding of X_i together with X_j, $T(X)$ applies the transformation T to the embedding X and $E(P)$ is the set of all $p-$dimensional rigid transformations. The optimization looks for transformations for c_i which minimize the stress between c_i and c_j while preserving the minimal distance between both clusters. The use of rigid transformations (rotations, translations and reflections) guarantees that the already embedded internal structure of the clusters is maintained by the new embedding. The resulting embedding for the root node of the dendrogram is the final result of TPE and plotted on a 2D plane. Because it is not guaranteed to find the same optimum as when looking at all cluster pairs at the same time, the authors of TPE call this approach "greedy optimization".

This approach allows TPE to preserve clusters at all resolutions and not only at an (arbitrarily) chosen one. TPE is more than just a new visualization technique for dendrograms. Through the use of the penalty function $\sigma(X)$ not only the minimal distances between clusters, but also the internal structure of the clusters is considered for the embedding.

Please note that we use SPADE for clustering and therefore the distances D in high dimensional space are the L1-norm while in the embedding the L2-norm is used. This seems maybe arbitrary but is justified here. SPADE is optimized for usage with FACS data and uses the L1-norm, so we can judge it is a valid choice there. On the other hand the visualization is made for humans and humans naturally think with the Euclidean norm as distance.

4.2 Visualization Method

The whole process of visualization FACS data is as follows. The FACS data is compensated and transformed as described in section 2.2.1 and 2.2.2. Then the SPADE down-sampling (section 3.1.1) is performed to reduce the number of objects to handle. The remaining cells are clustered with SPADE and a dendrogram is constructed of all cells. There is no stop if the "number of clusters" parameter is reached. With that dendrogram and the L1-distance matrix TPE is called to create a 2D embedding. The cells of the whole dataset are assigned to their nearest neighbour in the down-sampled set, so that information of the complete set can be visualized in the plot.

The visualization itself is done in the following way. The embedding is plotted into a 2D plane, each point corresponds to a cell in the down-sampled set and its size corresponds to the logarithm of number of cells for which it is the nearest neighbour. This allows the user to distinguish between high and low density regions in the plot. The colour of a point corresponds either to the median value of a marker or some other external data, e.g. gating information. An example visualization is shown in figure 4.2.

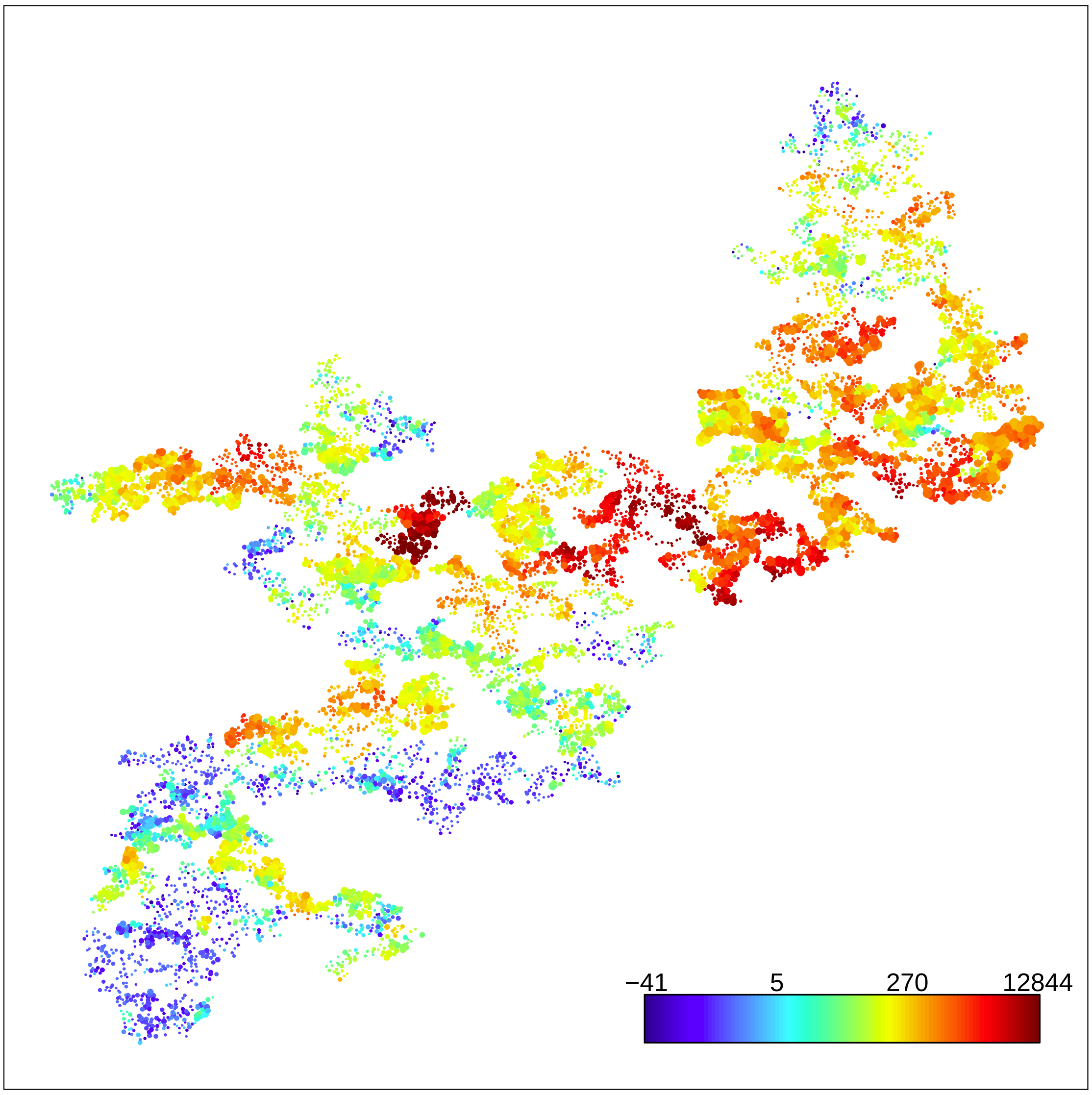


Figure 4.2: Example visualization of the hematopoietic stem cell dataset of section 3.5.1. 10000 clusters are visualized coloured by the CD41 marker. Size of the points corresponds to the number of cells in their neighbourhood.

4.2.1 Visualization of FACS Timepoints

We can modify our approach to allow different timepoints of similar data to be visualized on the same structure. The first step is to construct a visualization of all timepoints together, which can be seen as “cell space” of possible locations of cells. Afterwards each datapoint can than be visualized on this space independently.

Due to different sample sizes at different timepoints we down-sample each set separately and put it afterwards together to one set. This set is then visualized with the normal approach described above. The important thing of the result are the downsampled points and their location in the visualization.

After that each timepoint is visualized separately on this structure. Each cell in the timepoint is assigned to its nearest neighbour in the previously created structure. These points in the visualization are then visualized using the number of assigned cells and their average marker expression.

The markers used for the construction of “cell space” are considered to be constant for cell populations between different timepoints. Only with this assumption the mapping to the structure makes sense.

4.3 Implementation

We used R for the implementation in this chapter, because an R package for TPE was already available online. Indeed there is an R implementation of SPADE, too, but it is very user-oriented and therefore hard to change internals, which is required here (e.g the minimum spanning tree construction is not required). So we implemented the SPADE workflow without visualization again in R.

4.3.1 SPADE Implementation in R

Nevertheless we use the SPADE R package for the density calculation, because it is available as a simple method call. It is multi-threaded and uses many cores simultaneously to speed up the calculation. The cluster algorithm itself was implemented completely by ourselves. To assign the cells from the complete dataset to their nearest neighbour in the down-sampled set (up-sampling step cf. 3.1.1) we use a modified (L1 instead of L2-norm) version of the *RANN* [1] R package which implements the nearest neighbour search using a *kd-tree* [33] in C. However a use of this kd-tree for the density calculation in the first step is slower than the SPADE package. The reason is probably that the neighbourhood is too large and the overhead of traversing the tree is higher than the brute-force overhead in the naive version. That is why we use the SPADE package for this task.

[1]The original version is available through the *The Comprehensive R Archive Network* under http://cran.r-project.org/

4.3.2 TPE Package Optimization

In the beginning the original TPE package has long runtimes which makes the visualization with more than 1000 datapoints unfeasible. So we applied the following optimizations to the package:

- Implemented the stress and constraint function in C
- Removed wrapper methods for C code
- Prepare parameters for the C code outside of the methods
- Removed root calculation when looking only for minimal distances
- Removed some internal optimizations of the original TPE package which turned out to be slower than using the new C code without these optimizations.

4.4 Results and Discussion

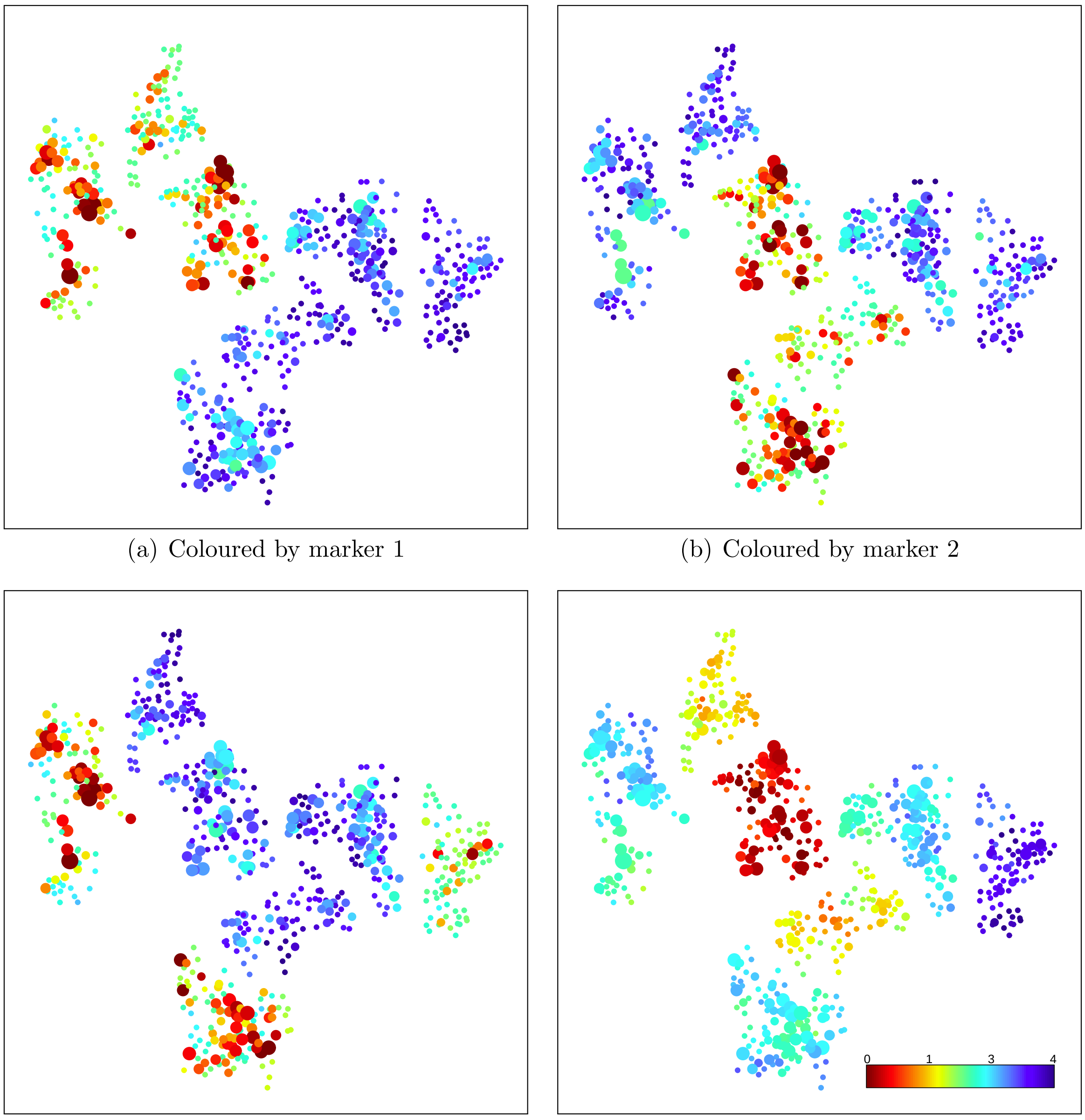
4.4.1 Evaluation on Artificial Dataset

We test our visualization on a small artificial dataset to show its advantages (cf. figure 1.1). The dataset consists of 1100 datapoints with three dimensions. There are four large clusters of datapoints (each 200 points) and three clusters with only 100 points. The clusters represent different cell populations in FACS data. The points for each cluster are drawn from a normal distribution with standard deviation of 0.15 and are positioned in the corners of a cube. There are only seven clusters for the eight corners of the cube, so one corner is left empty.

In figure 1.1(a) all scatter plots for this dataset are shown. Without further investigation (e.g. gating) eight clusters are expected because the empty corner is hidden in all dimensions by another cluster.

We run our visualization method on that dataset. The downsampled set contains approximately 600 datapoints and its visualization is shown in figure 4.3. With the coloured versions (a)-(c) it is easy to distinguish the different clusters.

To check if the distances between the different datapoints are preserved, we calculated the L1 distances from an arbitrarily chosen datapoint to all others. The result coloured by these distances is shown in figure 4.3(d). As you can see short distances in the plot corresponds to short distances in the real data and far away points correspond to long distances in the data. So the visualization is in fact able to preserve real distances while showing the whole data structure in one two dimensional plot.



(a) Coloured by marker 1

(b) Coloured by marker 2

(c) Coloured by marker 3

(d) Coloured by distance

Figure 4.3: With our new visualization technique all seven clusters are separately displayed while the distances are preserved. Our method was run on the artificial dataset. (a)-(c) The results coloured by the different markers are shown here (red=high, blue=low). (d) The result is coloured by the (manhattan) distance to an arbitrarily chosen datapoint.

4.4.2 Runtime Analysis of Optimized Code

We re-implemented and optimized large parts of the existing code so it is important to keep track of the required runtimes used by the different steps. Here we compare

(1) the density calculation with the R SPADE package versus the original code, (2) the reimplemented SPADE cluster algorithm in R versus the original in MATLAB, (3) the modified RANN package versus the original upsampling code and (4) the optimized TPE package versus the original TPE package.

To be comparable all tests were run on the same notebook with four processor cores and 4 GB of ram. As dataset we choose a randomly generated set with 10 markers and 100000 cells.

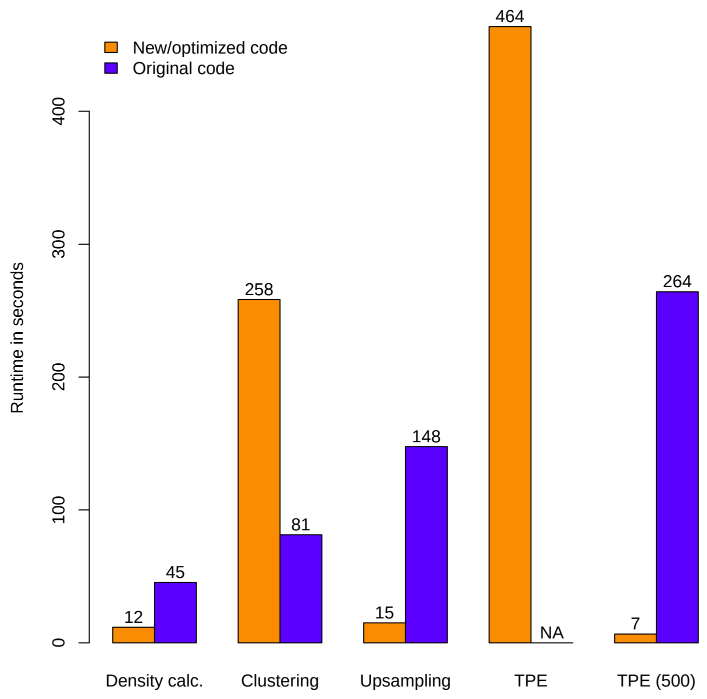


Figure 4.4: New code is substantially faster in most steps. New runtimes of the different steps in the visualization are compared against the original algorithms and implementations.

The first two bars in figure 4.4 show the times required for the density calculation. The multi-threaded version is four times faster than the original version, because it can use all four available processor cores at the same time.

The runtimes for the clustering step are surprising. The new implemented version in R is approximately three times slower than the original MATLAB code. A closer look into the SPADE source code reveals at least one reason for that. The clustering algorithm does not use single linkage for the clustering but just uses distances between the median centers of two clusters. So there is no need to consider and calculate the pairwise distances between datapoints in two clusters, which requires much more runtime.

The usage of a kd-tree in the upsampling step is a substantial improvement. The new code only takes 15 seconds while the original code takes 2.5 minutes. This is a speed up of factor of ten compared to the original method using a brute force approach.

The new optimized TPE package needs about eight minutes to finish, while we were not able to complete the run with the original TPE because of long runtimes. Instead we used a smaller dataset of only 500 instead of 100.000 data points to compare both versions against each other. In that comparison the new version is factor 35 faster than the old one. Here the optimizations discussed in section 4.3.2 perform well.

Taken together we were able to speed up the TPE package by a factor of 30 which allows the processing of thousands of datapoints in minutes instead of hours. For our application this was enough, so we did not investigate further optimizations like a relaxed stress function.

4.4.3 Biological Analysis with Expert Knowledge

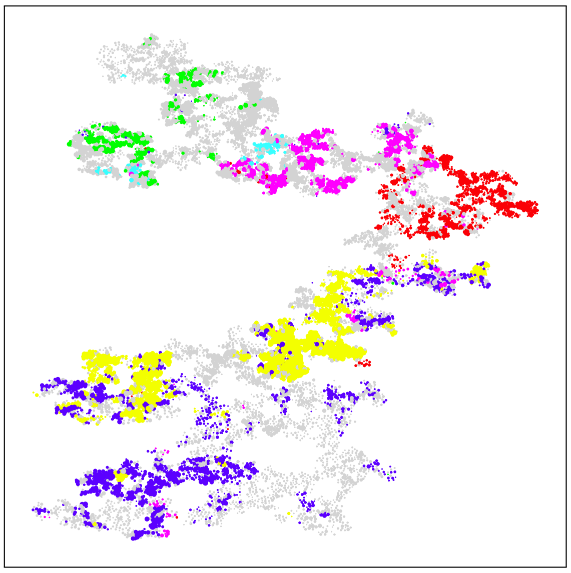
We tested our visualization technique with the dataset from section 3.5.1, because there is expert knowledge about the different populations available and therefore a visual analysis of the quality can be done. In addition we tested how different selected marker sets from the previous chapter change the visualization.

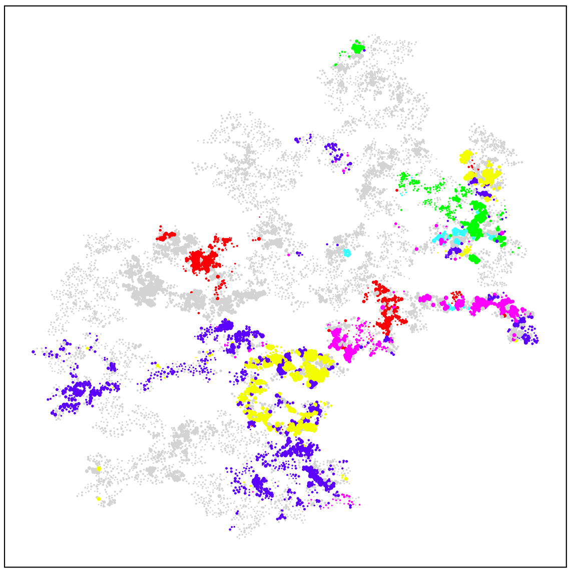
The results are shown in figure 4.5. Each point is assigned to the majority gate in its neighbourhood and is coloured according to that gate. A light grey point indicates that most cells in the neighbourhood are not contained in any gate.

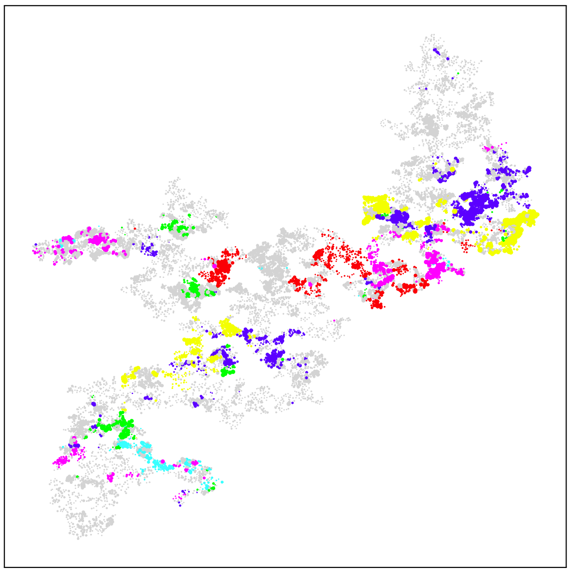
The frequency differences of different gates between the three visualizations are due to different shapes in high dimensional space which change if new dimensions (markers) are added or removed. So a cell population which uses much space in figure 4.5(a) with 11 markers can be compressed to a very small part of the total space in the third plot with only four markers. The down-sampling step then selects many cells in the first and only a few in the second case.

The visualization with all available markers shows a good separation of all six supplied gates. Only the *preGMs* are in parts scattered around in the lower part of the plot and mixed up with the *GMPs*. This is not surprising, because the preGMs give rise to the GMPs, so it can be expected that both cell types are similar and are not easy to distinguish. Also the lineage tree (see figure 4.5(d)) is mostly maintained. The pre megE population splits into the CFU-E to the left and to the right into the MkPs. Furthermore the CFU-E develop into the erythrocytic progenitors again to the left.

The result gets worse if the seven markers of the previous chapter are used to visualize the data. The megakaryocytic progenitors are split up into two subpopulations, the same is true for the pre MegE population. GMPs show up in a well defined population in the middle of the plot, still partly mixed up with the preGMs. The marker selection only take a truncated dendrogram into account for comparisons while here we use the complete one.


(a) All eleven markers.


(b) Seven selected markers.


(c) Four markers used for gating.

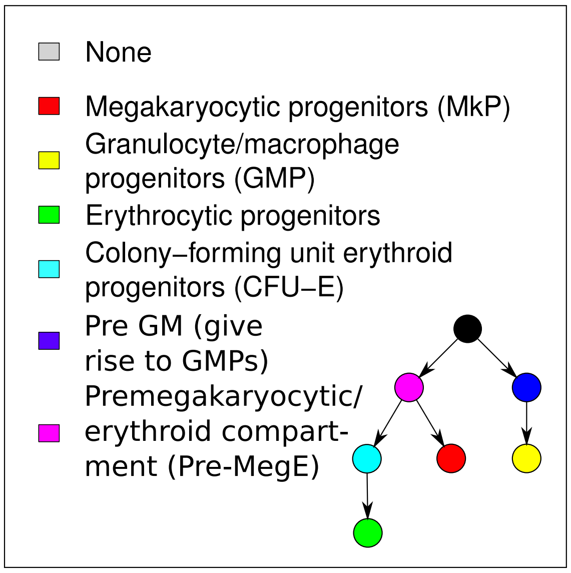

None
Megakaryocytic progenitors (MkP)
Granulocyte/macrophage progenitors (GMP)
Erythrocytic progenitors
Colony–forming unit erythroid progenitors (CFU–E)
Pre GM (give rise to GMPs)
Premegakaryocytic/ erythroid compart- ment (Pre-MegE)
(d) Legend and lineage tree

Figure 4.5: Visualization with all available markers leads to the best visualization. On a dataset with six subpopulations our method run three times with different marker sets. The resulting datapoints are coloured according to the majority subpopulation in their neighbourhood. (d) Shows the colour legend for the six cell populations together with lineage tree for the six cell populations based on [34] and [35].

Therefore we assume that the shown differences are caused by the structure of the lower dendrogram levels.

The expert uses only four markers for the gating. So the expectation is that the visualization with only those four markers should be as good or even better (if some noisy markers are removed) than with all markers. But the opposite is the case. The result with less markers is worse than with all markers. In figure 4.5(c) most gates are locally well preserved but globally spread over the whole plot. E.g. the erythrocytic progenitors are split up into four populations. That means that the remaining seven markers contain information which is not used by the manual gating scheme.

4.4.4 Comparison with other Dimension Reduction Methods

We compare our visualization to other dimension reduction methods to evaluate the performance. We selected the standard method *principal component analysis* (PCA) [22], a recent and more sophisticated method *t-SNE* [32] which is based on *stochastic neighbor embedding* [36] and SPADE (see section 3.1) for comparison.

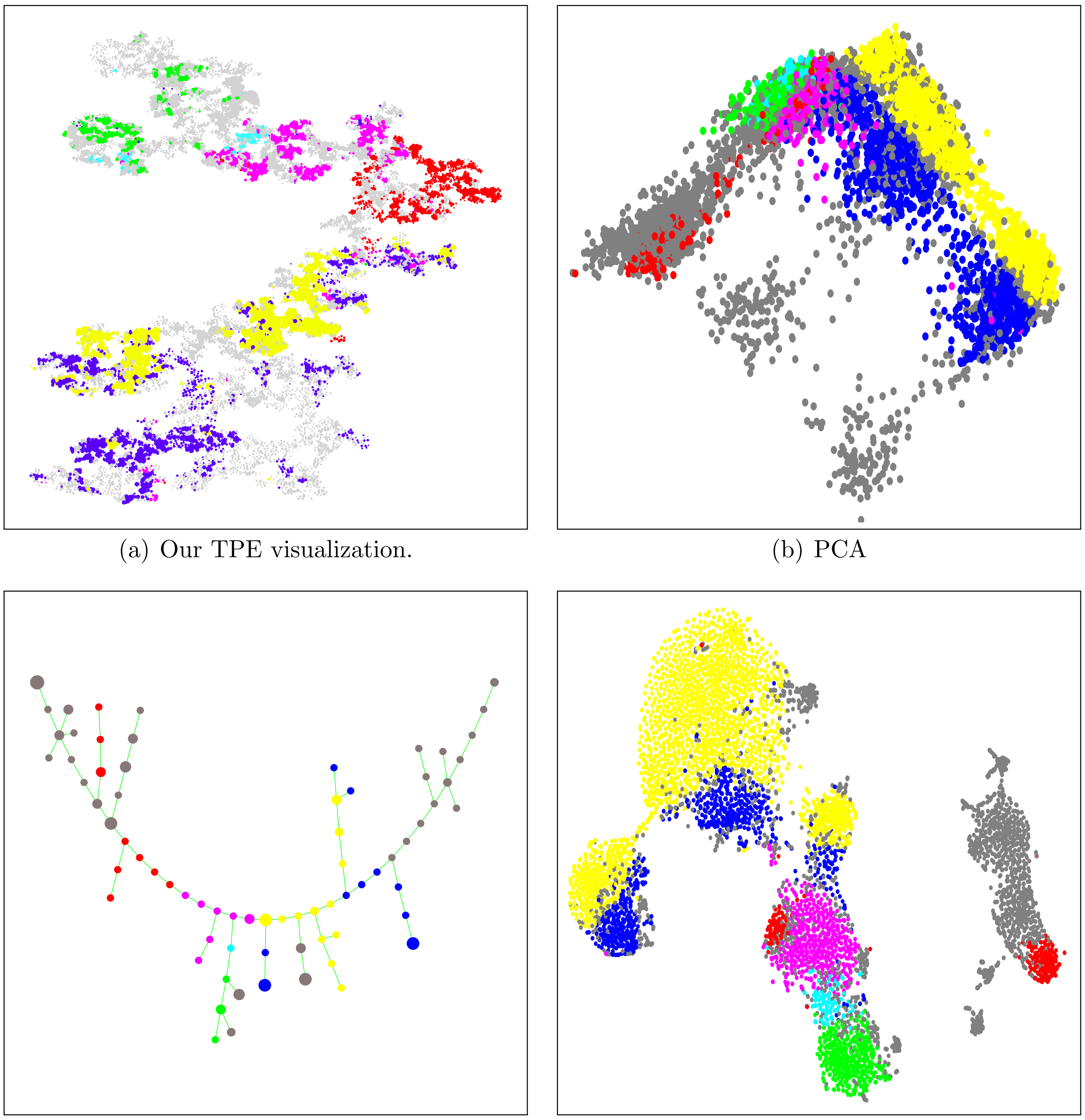
Principle component analysis looks for that linear combinations of the input markers which allow the explanation of the most variance in the data. For t-SNE we use the landmark approach for large datasets, which selects randomly landmark points and uses random walks between these points in a neighbourhood graph of the whole dataset to determine the probability to reach another landmark point through this random walk. Then the 2D visualization is optimized so that the distance in 2D is equal to the calculated probability.

The same dataset and expert knowledge as in the previous section is used for this analysis. All available eleven markers are used as input for the different methods. The resulting 2D visualisations are shown in figure 4.6.

PCA finds a view of the data which separates pre GMs (blue) well from GMPs (yellow) (figure 4.6(b)). PCA is a linear transformation and therefore the linear gates (which are used to define pre GMs and GMPs) are also transformed and lead to the good separation. The megakaryocytic progenitors are spread across the upper left part of the projection and also the CFU-E and erythroytic progenitors are piled and not separable. For all population PCA is not able to separate cells in the background from the foreground cells.

SPADE is able to find the different populations and group them together. Only the megakaryocytic progenitors are separated by two clusters representing no gate. As already seen in our visualization the GMPs and pre GMs are mixed, which can be explained by their direct relation. Despite this good separation and visualization the distances in the plot have no meaning and cannot be used for further interpretation.

T-SNE is able to distinguish very well between different populations. As with SPADE and TPE GMPs and pre GMs are close together. Also the very small population of preCFU is found and visualised very near together. The megakaryocytic progenitors are split up into two populations which was already seen with SPADE. T-SNE does not use a



(a) Our TPE visualization. (b) PCA

(c) SPADE (d) t-SNE

Figure 4.6: Our visualization method separates the subpopulation. Comparison with three different methods. For PCA the first and second principle component is shown. SPADE was run with the default parameters. T-SNE was run in *landmark mode*. Colors and legend are the same as in figure 4.5.

downsampling step therefore frequent populations are also frequent in the visualization. This can be seen in figure 4.6(d) with a lot of yellow datapoints, while in our approach the down-sampling allows also small population to be visualized well.

The simple PCA method is not suitable to separate different cell populations. T-SNE does a very good separation but do not preserve the original hierarchy of data at all resolutions which may be interesting for biological interpretation. SPADE also makes a good separation, but has the same problem. Our visualization is based on SPADE so we are able to detect different cell populations and also to preserve original distances in the visualization at appropriate levels.

4.4.5 Visualization of Different Timepoints

We use the mesenchymal stem cell dataset from section 3.5.1 with four different timepoints (day 2, 14, 28 and 42). Some cells are marked with the *green fluorescent protein* (GFP) so they can be "traced" where they move to and how they differentiate.

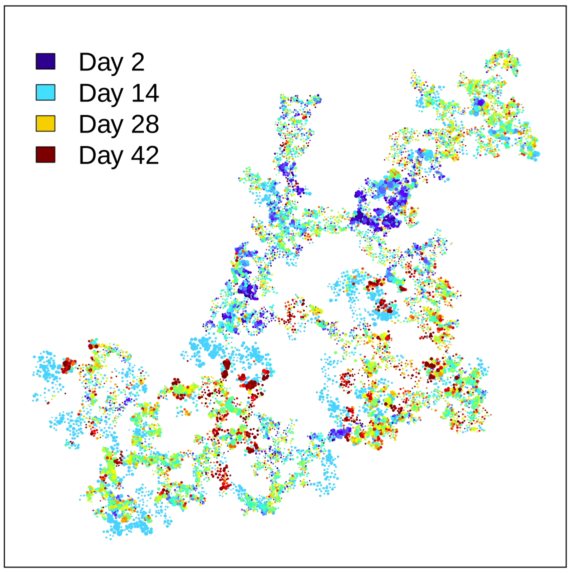
The modified visualization approach of section 4.2.1 is used here. We do not use GFP for the construction of the structure because it changes over time and cannot be considered to be constant in subpopulations between different timepoints.

The visualization shown in figure 4.7(a) is coloured by the origin of each datapoint. In most parts there is a mixture of at least three colours, which means the cells of the different datasets are located at similar points and belong to the same subpopulations. Only for day 14 there are a lot of cells which are at the border and only show up in that dataset. The reason for this is maybe the usage of a different FACS machine. Therefore we excluded day 14 from further analysis.

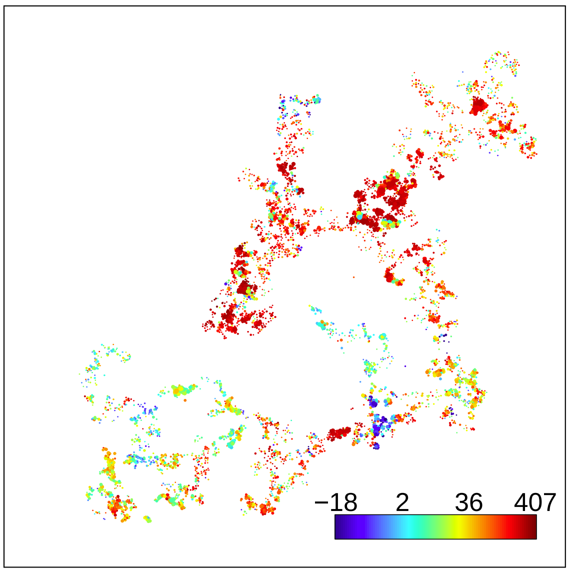
The results are shown in figure 4.7. Between day 2 and day 28 there is not much change except for a cluster of cells on the right bottom which changes its intensity from very low (dark blue) to high (red). Between day 28 and day 42 there is more change. Cells are moving from top to bottom as indicated by the circle sizes. Also in the lower left part of the plot new populations with high GFP levels show up.

4.5 Conclusion

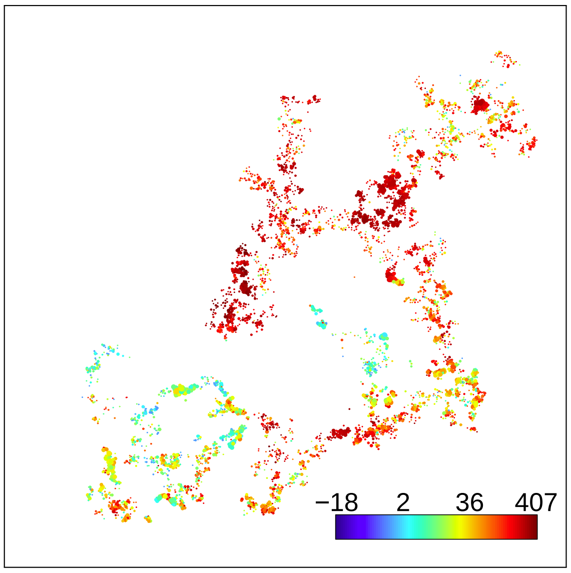
To improve the visualization of FACS data we used a modified basis of SPADE to create a hierarchical single-linkage clustering on the whole dataset together with an optimized version of TPE for visualization. SPADE was modified to create a complete dendrogram instead of stopping at a defined level. The resulting dendrogram is then visualized using TPE which is able to preserve distances between clusters at each level of the dendrogram and guarantees a clustering on the visualization to find the same dendrogram as before again. For the visualization the dot size shows the density of cells in that region and the colour indicates the average marker expression of these cells. In addition we developed a variant of this technique to visualize different timepoints on an "average structure" which allows to trace cells through different timepoints.



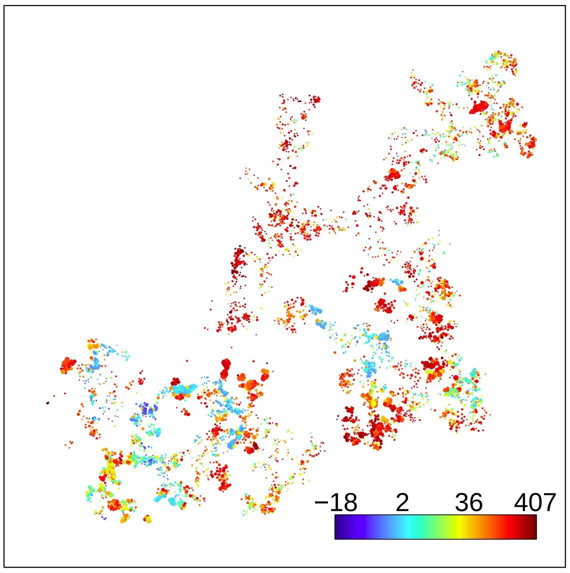
(a) Average structure. Coloured by the origin of the datapoints.



(b) GFP expression on day 2.



(c) GFP expression on day 28.



(d) GFP expression on day 42.

Figure 4.7: A different application of our visualization is tracing of cell populations through different timepoints. Similar cells are measured at different timepoints. (a) all timepoints together are used to construct the “average structure”. (b)-(d) timepoints visualized on the “average structure” coloured by the GFP level.

The algorithm was tested with an artificial dataset to proof its proposed properties and also with different marker sets on a dataset with expert knowledge. The result was better the more markers are available for visualization. With all markers we were able to separate the expert-given subpopulations in the visualization. In comparison with three other visualization methods (PCA, t-SNE and SPADE) we found our method to be suitable due to the ability to separate expert subpopulations while preserving real distances in the visualization which allows a better interpretation of the dataset. With the timepoint approach we were able to trace mesenchymal stem cell through different populations but the data has to few timepoints and was to noisy to reveal new biological insights.

Chapter 5

Pattern Discovery in Stem Cell Trees

In contrast to the two preceding chapters we work in this chapter with annotated stem cell trees as introduced in section 2.3. The goal of this part of the thesis is to find frequent substructures in these trees. More specifically we are looking for patterns which appear differentially in two datasets. That means a pattern occurs often in one dataset, but is very infrequent in the another one. This is especially useful if there is a known qualitative difference between the datasets, e.g. two different culture conditions one with and one without a specific substance. Then the interesting patterns can give insights into biological mechanisms and role of that substance.

For trivial patterns, like one cell with one marker on, this can be done manually but for more complicated patterns this is not possible (cf. figure 1.2). Additionally it is not feasible to detect complicated patterns just by looking by hand at the trees. So a systematic approach with an appropriate search algorithm is required.

This task is related to *differential network analysis* which tries to identify differences in biological interaction maps [37]. In both cases two graphs are compared but we have sets of graphs while in differential network analysis you look at two large networks. A related approach was discussed in [38] which aims to the construction of a *sub group tree*. The tree represents all possible subgroups which are defined through different combination of attribute filters. Each subgroup is then coloured according to a p-value.

In this chapter we will first discuss how the datasets look like. Secondly, we will define patterns, their canonical representation and how they match to the trees. Finally we describe our algorithm to efficiently identify the top k patterns.

5.1 Data Definition

For our algorithm we require two datasets. The first dataset is the so called control or *background dataset* D_B, meaning there is nothing special done during the experiment. In the second or *main dataset* D something is changed compared to the background dataset, e.g. the addition or removal of some substances in the growth medium.

Both datasets D and D_B consist of annotated stem cell trees as described in section 2.3. On a more general and abstract level these trees are *unordered rooted binary trees*. Each tree t has a defined root, no order between and exactly two children per node. There is no need that the trees are balanced or complete. The nodes correspond to cells and each edge corresponds to a *mother-daughter-relation*. The concepts presented here can be applied to any tree set which satisfies these properties.

This definition alone represents only the structure and no additional "annotations". In practice each node can have additional attributes, e.g. cell lifetime, intensity of some markers or manually annotated cell fates. Attributes other than scalar or boolean can be used in principal, the only required property is a pattern definition and a matching method which returns *true* or *false*.

5.2 Patterns and Matching

5.2.1 Pattern Definition

Patterns consist of two parts, a structural and an attribute part. The structural part is responsible that the structure of the trees is taken into account while the attribute part only uses the additional attributes to determine a match.

To represent the structural part we use the same structure as the trees themselves, unordered rooted binary trees. Each node in the structural pattern can have additional constraints using the cell attributes, so called *attribute patterns*, which represent the attribute part. They consist of a range of allowed values, either inclusive or exclusive the borders. Each node can only have zero or one constraint per attribute, so multiple patterns for the same attribute are not allowed. This simplifies the pattern generation and maintenance because the frequency of a pattern remains monotonic (see equation 5.2 below).

5.2.2 Matching to Trees

We define four ways of matching a pattern p against a tree t. In general a pattern matches a tree if and only if there exists a mapping of all nodes in p to t. Three constraints have to hold for the mapping in order to be a valid match with two possibilities for 1 and 2:

1 a. The roots of p and t have to be mapped onto each other (*root matching*).

b. The root of p can be mapped on any node in t (*non-root-matching*).

2 a. The mother-daughter relation of p has to be preserved in t (*strict matching*).

b. The mother-daughter relation (n, n') in p can skip generations in t, meaning the mapped n' does not need to be a direct child of n but has to be in its progeny (*non-strict matching*).

3 All attribute patterns of nodes in p have to match their mapped nodes in t.

So in total there are four possible matching methods (see figure 5.1). Constraint 1 restricts the matching of the roots, both roots have to be mapped onto each other. Constraint 2 determines the meaning of the child relation (n, n') in p. Either n' has to be a direct child of n or n' has to be in the progeny of n. The third point ensures that the attribute patterns are not violated.

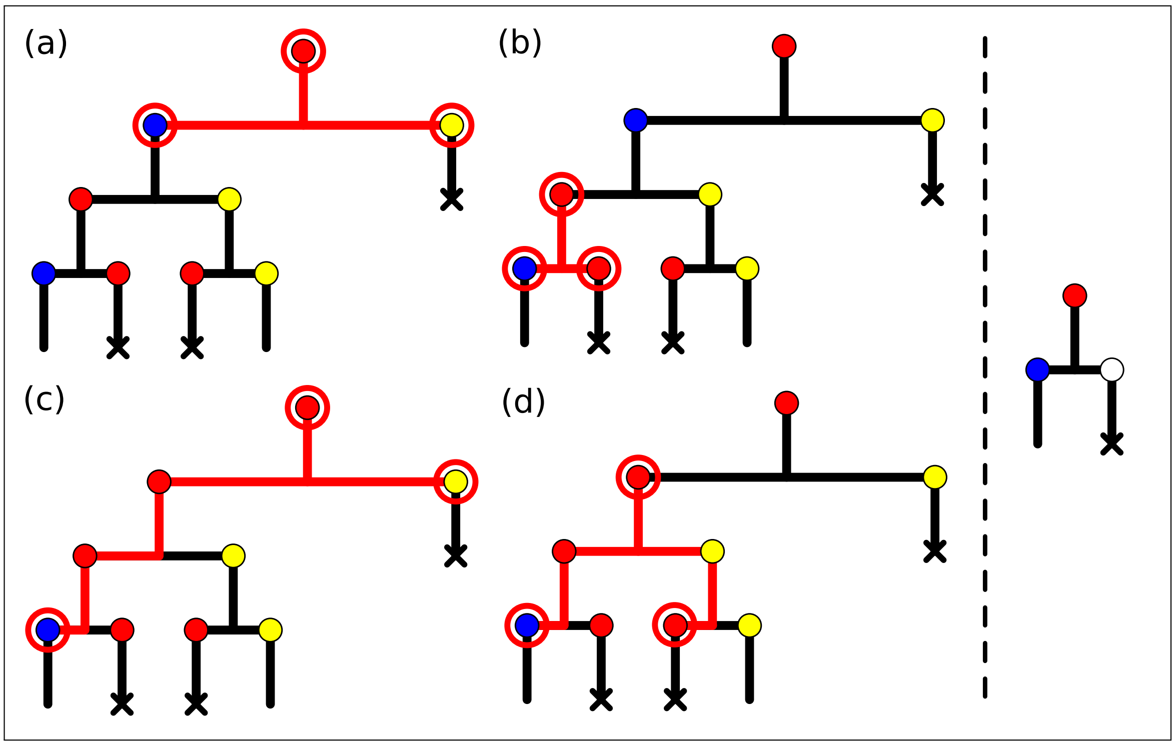


Figure 5.1: Different matching methods for patterns. The same pattern (right) is matched with the four different methods to similar trees. Only one possible matching is shown if several are possible. (a) Root and strict matching. (b) Non-root and strict matching. (c) Root and non-strict matching. (d) Non-root and non-strict matching.

We will score a pattern p according to its *frequency* in dataset D which is defined as

$$freq(p, D) = \frac{1}{|D|} \sum_{t \in D} I(M(p, t)) \tag{5.1}$$

with $M(p, t)$ denoting a match and I being the indicator function. The frequency is simply the percentage of trees in D which the pattern matches.

5.2.3 Canonicalization of Patterns

Although trees representing patterns are unordered the visualization of a tree has in fact an order on the children of a node. There are several *isomorphisms* which all represent exactly the same tree. To avoid duplicates and simplify the generation and maintenance of patterns we will define one of these isomorphic trees as our canonical form. The order of children is not part of the matching process so this canonicalization does not change the trees which a pattern matches. It also prevents the search algorithm from evaluating the same patterns several times and also simplifies the generation of new patterns.

We utilize an extended form of *level sequences* [39] to achieve this task. The depth of a node in the pattern (the length of the path to the root plus one) is called level of that node. To obtain the extended level sequence we consider a depth-first traversal of the now ordered pattern tree. At each node we write down the level of the node and the name of each attribute pattern in alphabetic order. For simplicity we assume the attribute pattern names are just A, B, C, ... For example *1B233A2AB* is an extended level sequence. The *canonical form* of a pattern is defined as that representation which has the greatest level sequence in a lexicographical sense. We define that characters are always greater than numbers. This implies that the sequence above is not canonical because *1B2AB23A3* is larger and represents only a structural isomorphism. A visualization of the two sequences can be found in figure 5.2.

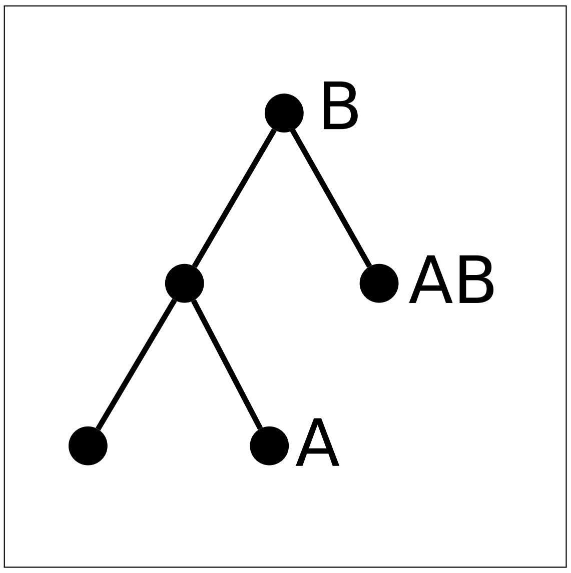
If we look at the tree structure of the pattern this canonicalization means that the tree is *left-heavy*. For each node the sequence of the left child is greater than the right one. This can be either because the attribute patterns are larger or there are more children.

There is no need to include more information about the pattern in the sequence, because we restricted the number of attribute patterns per node per attribute to one. There is simply one or none.

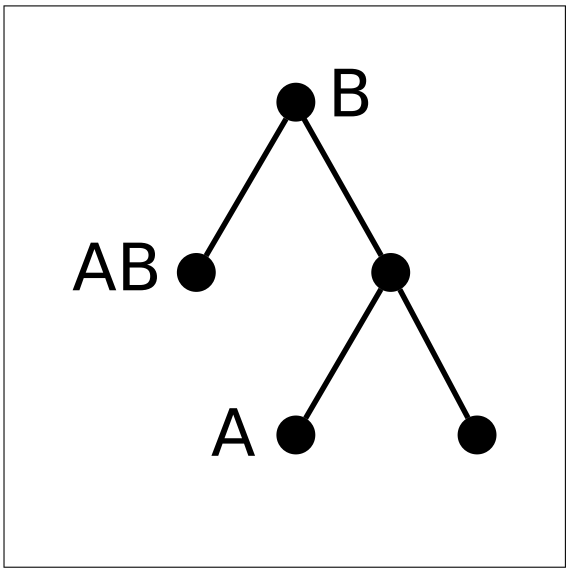
Note that for simplicity reasons we used level numbers and attribute names with only one digit or character, so the lexicographical comparison is easy. In reality a level number of 10 is of course larger than a 9 and if there are more than 26 attributes you have to do the comparison more carefully.

5.3 Pattern Tree

In order to be able to describe the search algorithm in an easy fashion we define the *pattern tree*. This tree contains all possible canonical patterns and their relation in terms of “is specialization of”. Therefore we define how the specialization works and how it can be used by the algorithm to be efficient.



(a) Isomorphism with level sequence *1B233A2AB*. Not in canonical form.



(b) Isomorphism with level sequence *1B2AB23A3*. In canonical form.

Figure 5.2: Isomorphisms of patterns are a problem. The same pattern drawn in different isomorphisms.

5.3.1 Specialization of Patterns

We define specialization as the creation of several new patterns which are more specialized than the original pattern. In general you expect the new patterns to not match as many trees as the original pattern.

The extended level sequence facilitates an easy definition of specialisation. We can specialize a pattern p by extending its sequence by one position. The sequence of pattern p is extended by one new position, if the newly obtained sequence is canonical the new pattern q is a specialization of p, which is denoted by $p \prec q$. The extension can be made with either a new number, which corresponds to a new child on the right most branch, or a new character, which corresponds to a new attribute pattern. If we look in the structure the specialization means that on the right-most node (the last in depth-first traversal) we add either a new attribute pattern or a new child node. Because canonical means left-heavy and here we add "weight" to the right it is clear why the new pattern has not to be canonical anymore although the initial pattern is. Therefore the canonicality has to be tested afterwards.

This specialization is *monotonic*, which means a specialization of a pattern can not have more matched trees than the original pattern. So for a dataset D and two patterns p and q it holds that

$$p \prec q \Rightarrow freq(p, D) \geq freq(q, D). \tag{5.2}$$

The very first pattern, which cannot be constructed through specialization is the *empty pattern* p_0 with the empty level sequence. It does not contain any nodes or attribute patterns and matches every tree regardless of its structure or attributes.

This method of specialization presented here requires more than constant time for each pattern to be constructed, because for each candidate it has to be tested if the pattern is canonical. There are more sophisticated algorithms which are able to enumerate all the trees in constant time per pattern [40]. Because speed was not an issue here we have not implemented such an algorithm but this may be required in the future.

5.3.2 Definition and Completeness

Now we define the pattern tree PT with $\mathfrak{P}$ being the set of all canonical patterns as

$$PT = (\mathfrak{P}, E) \quad \text{with} \quad E = \left\{(p, q) \in \mathfrak{P}^2 \,|\, p \prec q\right\}. \tag{5.3}$$

The patterns which are specializations of a different pattern are children of that pattern in the tree.

Given a canonical pattern q with extended level sequence $s = (c_1, \dots, c_n)$ we invert the specialization step and remove the last item c_n of the sequence. This results in a sequence which belongs to the parent p of pattern q in the tree. This way each pattern can have only one parent and after n iterations of removal we reach the root node p_0 with the empty level sequence.

We now show that each intermediate sequence s_i in this process is canonical which implies that the corresponding patterns p_i is part of $\mathfrak{P}$ and therefore part of the tree. The removal of the last item c_n in the sequence of p always works on the right most node of p. Either an attribute pattern is removed (if c_n is a character) or the whole node is removed (if c_n is a number). A pattern is canonical if for each node the children are ordered descending by their extended level sequences. Because we work on the right most node we only change the sequence of the rightest node, which is also the lightest one. Through the removal of an item the sequence only gets lighter and therefore the order of sequences is not changed. So proof through induction: the parent pattern is guaranteed to be canonical if p was canonical.

Each canonical pattern is part of PT and can be reached through a path starting at the root and using the specialization to go deeper into the tree.

5.3.3 Validation

To verify the correctness of our implementation of the pattern tree, we generate all patterns with a specific number of nodes and compare the number to theoretically computed numbers.

We consider patterns with n nodes and a possible attributes. (The number of different attribute patterns per attribute is not considered here. We assume there is one possible attribute pattern per attribute.)

Calculating the number of possible unordered rooted binary trees with n nodes can be split up into the smaller problems of finding the numbers of possible left and right subtrees with together $n-1$ nodes. So let $N(n)$ be the function calculating the number of trees with n nodes. With the assumption that n is even and an odd number of nodes have to be distributed to left and right we can recursively define it as

$$N(n) = \sum_{i=1}^{\lfloor n/2 \rfloor} \left(N(i-1)N(n-i)\right). \tag{5.4}$$

We can stop at $n/2$, because then the same trees are counted only with left and right subtree exchanged and we consider here only unordered trees. If n is odd the new case of having $(n-1)/2$ nodes on the left and right side occur. For this you have to choose 2 out of $N\left((n-1)/2\right)$ possibilities with replacement which are $\binom{N((n-1)/2)+1}{2}$ possibilities.

Each node can have each possible combination of attribute patterns and there are 2^a combinations for the a patterns. So in each step these pattern can be assigned to the root, which leads to the final function $N_a(n)$. The number of possible unordered rooted binary trees with n nodes and 2^a possible labels for each node is

$$N_a(n) = 2^a \sum_{i=1}^{\lfloor n/2 \rfloor} \left(N(i-1)N(n-i)\right) + 2^a \begin{cases} \binom{N((n-1)/2)+1}{2} & n \text{ is odd} \\ 0 & \text{else} \end{cases} \tag{5.5}$$

with the recursion start at

$$N_a(0) = 1 \quad \text{and} \quad N_a(1) = 2^a. \tag{5.6}$$

The values of this function for different parameters for n and a are shown in table 5.1. The results of our implementation of the pattern tree are compared against these numbers to ensure a correct and error-free implementation.

5.4 Mining Algorithm

5.4.1 Interesting Measure for Patterns

We need a way to quantify how "interesting" a pattern is. We defined interesting as occurring frequently in dataset D while occurring infrequently in D_B therefore we use the difference of frequencies as interesting measure and the score or interestingness $I(p)$ of pattern p is

$$I(p) = freq(p, D) - freq(p, D_B). \tag{5.7}$$

n	a				
	0	1	2	3	4
0	1	1	1	1	1
1	1	2	4	8	16
2	1	4	16	64	256
3	2	14	104	800	6272
4	3	44	672	10496	165888
5	6	164	4896	151808	4786176
6	11	616	36992	2295808	144736256
7	23	2450	291152	36019328	4535272448
8	46	9908	2349376	579986432	145868210176
9	98	41116	19364672	9532859392	4.788215×10^{12}
10	207	173144	162231552	159264088064	1.597583×10^{14}

Table 5.1: Values of the function $N_a(n)$ with different parameters for n (in rows) and a (in columns).

This is a very simple measure. The fact that the frequency is a percentage and therefore is adjusted to the dataset size makes this measure independent of different dataset sizes and is comparable also between different datasets. Other measures which are used frequently in similar cases and are eventually suitable here often require knowledge of the combined distribution of both datasets (i.e. $P(A \wedge B)$), which is not available here.

One advantage of this differential approach is that we are not required in any way to quantify the complexity of a pattern. If you consider the task of finding interesting patterns in one dataset, the most frequent ones are the simplest ones at the same time. So the pattern with one cell has certainly a frequency of 100%. In general to find interesting patterns a good balance between complexity and frequency has to be found. In our case this is done automatically through the usage of the background set. Everything that is not in there is interesting and the simplest interesting patterns are found, because they are more frequent than complex ones.

Upper Bound for Interestingness Measure

This measure allows for an upper bound on the measure for specialized patterns. Due to the observation in equation 5.2 we can derive the upper bound $B(p)$ for patterns p and all its specializations q

$$I(q) \leq freq(q, D) \leq freq(p, D) = B(p) \tag{5.8}$$

for the measure I. Through the transitivity of the specialization $\prec$ this bound holds for complete branches in the pattern tree. Given a pattern we can determine for the whole branch starting at that pattern an upper bound for the interesting measure. This will allow us an efficient way of pruning the tree and building a fast mining algorithm.

5.4.2 Algorithm

The goal of the algorithm is to find the top k patterns according to the measure I. It is basically a version of the $A*$ algorithm [41] which is able to use external information (such as the upper bound) to perform an efficient search through all possible patterns. It starts at the root of the pattern tree with the empty pattern and traverses the tree while maintaining a list of the current top k patterns and another list with candidates which can lead to better patterns in the future.

The first of the two main data structures with patterns is a sorted result list R of maximum size k with the current top k patterns sorted by their interesting measure. The value of the last pattern (lowest $I(p)$) is called $I_{\min}$. The second is a priority queue Q which contains patterns with an upper bound $B(p)$ higher than I_{min}. So it is possible for specializations of patterns in Q to become a top k pattern in the future. This queue contains initially only the empty pattern p_0.

With this data structures the algorithm runs as follows:

- Take the first pattern p in Q and do for each specialization q of p
  - If $B(q) < I_{\min}$ ignore q and go on with the next pattern.
  - If $I(q) > I_{\min}$ add q to R. Prune R if it is now larger than k and update $I_{\min}$ accordingly.
  - If $B(q) > I_{\min}$ add q to Q.
- Repeat until Q is empty.

In the end R contains the top k patterns found. The use of the pattern tree ensures that the upper bound holds and also that no pattern is checked twice or is omitted, which is in general not easy to achieve.

Generation of Attribute Pattern

We have not yet spent much time explaining how the attribute patterns are constructed and handled during the mining process. All attribute patterns are constructed before the actual mining starts. Because each node in a pattern can only have one pattern per attribute, we have to build many patterns per attribute in the beginning.

To determine which patterns we will use the distribution of values over all trees (in D and D_B) for each attribute is considered. If there are less than five different values for an attribute, e.g. because the attribute is an ordinal or boolean one, we build an attribute pattern for each distinct value itself. In all other cases we use quantiles to determine the ranges for the patterns. We construct five *base patterns* which range from the 0th to 20th, 20th to 40th, ..., quantile of the attribute value distribution. The use of quantiles here ensures that we take the internal structure of the data into account.

Because we want also patterns which range e.g. from the 0th to 40th quantile to be constructed and used during the algorithm we combine several base patterns to a new attribute pattern. We do this for all possible combinations, but only if all combined base patterns are side-by-side. This ensures that there is no gap in the range covered by a pattern.

5.4.3 Application with only One Dataset

Sometime it is interesting to look at only one dataset and not at the differences between two sets. Our algorithm can handle this as special case with an empty background set. That means the interesting measure can be simplified to

$$I(p) = freq(p, D) \tag{5.9}$$

because D_B is empty and therefore $freq(p, D_B)$ is 0. The problem which arises then is that very much pattern are found, e.g. the pattern with only one cell is found in every tree so it is the best found pattern. Through usage of additional pattern filters it is nevertheless possible to check the found patterns.

5.4.4 Implementation

The whole subproject was implemented in Java. It has the major advantage over MATLAB that there are references available. This is especially useful when working with trees, so each node can have e.g. a reference to the root node of the tree. It is also quite fast compared to MATLAB although this was not a major point here. Because MATLAB is partly based on Java it is possible to use the code directly in MATLAB where e.g. loading routines for trees were already available.

5.5 Results and Discussion

5.5.1 Dataset Description

TGFβ Change[1]

For the following analysis we used a dataset of annotated stem cell trees which consists of two parts. HSCs were imaged under two different conditions, once with the *transforming*

[1]Provided by Dirk Loeffler, *Research Unit Stem Cell Dynamics* of Timm Schroeder at Helmholtz Center Munich

growth factor beta (TGFβ) and once without. TGFβ is a cytokine and plays a role in cell differentiation and is therefore interesting in the stem cell research [9]. In addition to the structure of the trees three markers are quantified over time and available as additional attributes. They are *CD*48, *Sca1* and *VWF*.

HSCs versus MPPs[2]

In this dataset hematopoietic stem cells (HSCs) are compared against multipotent progenitor cells (MPPs). Interestingly, the current methods to isolate HSCs with flow cytometry (through different marker combinations) result in approximately 50% purity of this rare population of cells [14]. The notion in the field is that the remaining cells could be cells that are closer to MPPs. Our goal here is to identify patterns which are unique for HSCs or MPPs. With that the purity problem can be addressed through matching the trees afterwards against the two pattern and decide if the first cell in the tree was an HSC or a MPP.

Apart from the structure information of the trees only the annotated end fates (dividing, apoptosis or movie end) per cell are used for the pattern discovery.

5.5.2 Comparison of Different Matching Methods

The different matching methods have a biological background. Allowing the root of a pattern to match any cell in the tree reflects the fact that e.g. a differentiation can start later in the tree and only the event itself should be covered by the pattern and not the previous history. Matching directly at the root means that the patterns focus on processes occurring directly after the experiment start.

We tested the four available matching methods on both datasets and show the interestingness measure and size of the best found pattern in table 5.2. For each dataset four runs of the algorithm are performed, two with only one of the available set to detect frequent patterns only in the single sets. And two comparing the two sets in both directions so pattern frequent in set 1 and infrequent in set 2 are found as well as patterns frequent in set 2 and infrequent in set 1.

In the TGFβ dataset there is not much difference between the different matching methods. The found patterns for each direction are nearly identical.

For the HSC vs. MPP dataset all resulting best patterns are just one cell with either the attribute pattern of dividing or apoptotic. Therefore the results of the different methods are very similar. Also the non-strict mapping of mother-daughter relations has no effect here, because the patterns only consists of one cell and do not have such a relation.

[2]Provided by Konstantinos Kokkaliaris, *Research Unit Stem Cell Dynamics* of Timm Schroeder at Helmholtz Center Munich

	n.-root, n.-strict	n.-root, strict	root, n.-strict	root, strict
TGFβ change				
With vs. w/o	96%/33%	96%/33%	96%/33%	68%/25%
W/o vs. with	79%/5%	79%/5%	79%/5%	78%/5%
HSC's vs. MPP's				
HSC's vs MPP's	75%/34%	75%/34%	75%/34%	75%/34%
MPP's vs HCS's	86%/76%	86%/76%	66%/19%	66%/19%

Table 5.2: Comparison of the different matching methods on two datasets. The frequency in both sets of the best found pattern are shown in the table.

Comparing one direction to the back direction the results make sense. One direction finds a pattern matching cells which divide. The other direction cells which die. In the case of root-matching this behaviour can be seen in the result table: The frequencies sum up to 100%, e.g. for MPP's the values are 34% and 66%. In the HSC case there are (in addition to division and apoptosis) also cells which cannot be tracked so the frequencies of 75% and 19% do not sum up to 100%.

Overall the datasets are too small to find differences between the matching methods. It depends also on the biological goal which one to use in the algorithm.

5.5.3 Biological Analysis

For this analysis we used the root matching with strict mother-daughter relation. Also we do not focus only on the best pattern as before but on a larger set of patterns.

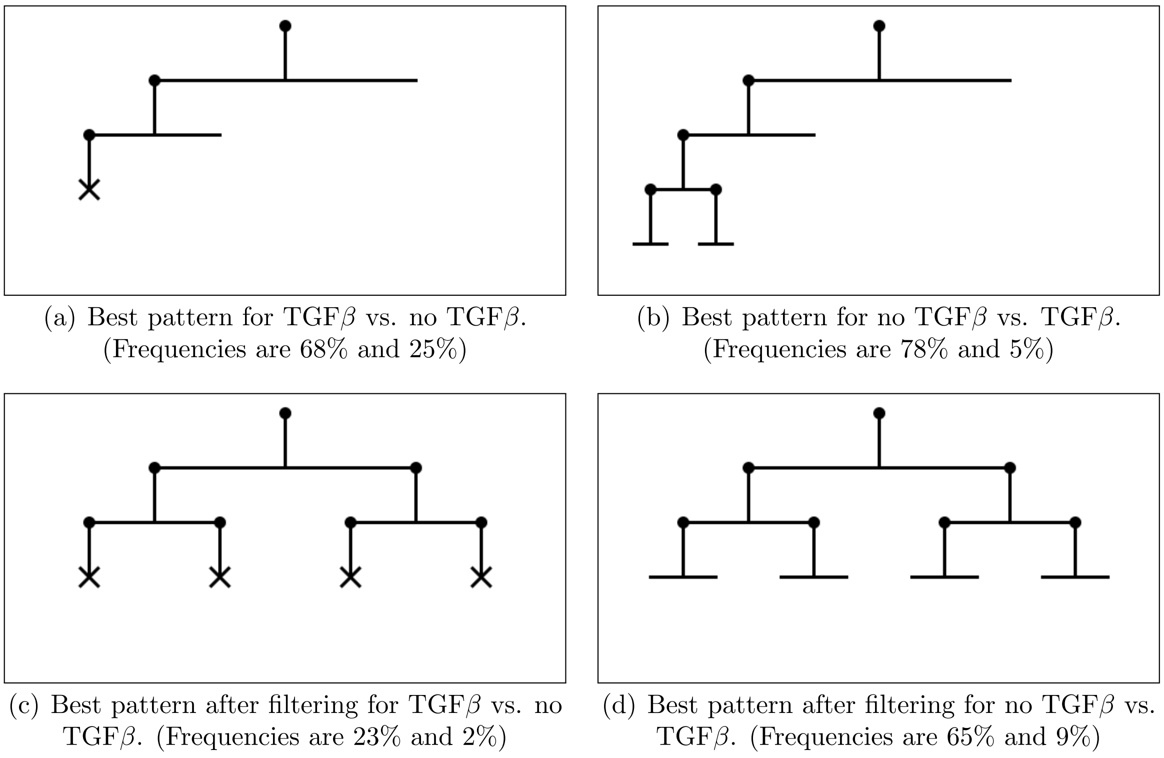
To be able to handle this large set we implemented a lot of pattern filters. The first filter restricts the number of nodes and generations in a pattern. This allows us to control how deep we go in the pattern tree. A normal setting was to only allow patterns to cover three generations. The second very important filter allows only patterns with each node containing an attribute pattern of either dividing, apoptotic or movie end.

The third filter passes cells up to generation 2 which are either apoptotic or are dividing and have two daughters. This filter together with the second filter results in a set of found pattern which is complete. This means each tree matches exactly one of the found patterns.

With this third filter we can also run the algorithm with only one dataset. The filter ensures that only complete pattern are found.

TGFβ Change

The best patterns found for this dataset are shown in figure 5.3(a) and 5.3(b). The first one consists only of three cells in a row of which the last one is apoptotic. This result


(a) Best pattern for TGFβ vs. no TGFβ. (Frequencies are 68% and 25%)

(b) Best pattern for no TGFβ vs. TGFβ. (Frequencies are 78% and 5%)

(c) Best pattern after filtering for TGFβ vs. no TGFβ. (Frequencies are 23% and 2%)

(d) Best pattern after filtering for no TGFβ vs. TGFβ. (Frequencies are 65% and 9%)

Figure 5.3: Best patterns for the TGFβ change dataset. Cross indicate apoptosis and horizontal lines indicate a cell division.

verifies that the algorithm is working well, because it was previously known that cells with TGFβ die earlier [9]. So the apoptotic pattern makes sense. Also the other way around the pattern in figure 5.3(b) shows two cells in generation 3 which divide matches this observation.

The results after filtering are shown in figure 5.3(c) and 5.3(d). They are complete until generation 2 and allow a better view on the whole tree.

HSCs versus MPPs

With the three discussed filters we were able to identify three patterns which are statistically more frequent in the HSC dataset. The treeset of MPPs has the problem of half of the cells dying in the first generation due to the used culture conditions. This is a problem because it leads to high frequency differences for more complex pattern. In this analysis we used *fisher's exact test* [28] to determine the significance of different patterns.

If we only consider the first two generations the best pattern has a frequency of 46% in the HSC dataset but only 22% in MPPs. So if we assume the purity of HSCs to be around 50% this can be an interesting starting point for further analyses. Later, we extended our analysis to trees with 3 available generations. We identified a pattern with frequency of 13% on HSCs that was completely absent from the MPP trees. Despite the fact that this preliminary analysis of HSC and MPP data needs higher tree numbers to yield statistically significant results, the algorithm already enables biologists to analyse complex tree data in a fast and automated manner.

5.5.4 Discussion

In general the quality of the analysed dataset is important. Without filters and only looking at the best patterns the found patterns are very obvious patterns with only one cell. This means either it is the biological interpretation that one cell type dies often while the other divides often or there is a biased tracking. That means the user does not track all cells equal likely but selects cells to track based on an expectation, e.g. because they divide.

5.6 Conclusion

For the discovery of differential occurring patterns in stem cell trees we developed an efficient algorithm. Given two tree datasets this algorithm is able to find the top k patterns which maximize the difference between frequencies in the two sets. We defined our patterns to consist of unordered rooted binary trees with additional attribute constrains attached to the nodes of the tree. We also defined four ways of matching a pattern to a given stem cell tree which corresponds to different biological assumptions. Due to several isomorphisms representing the same actual pattern we defined a canonical form of patterns based on extended level sequences. Using this sequences we defined also a way to specialize patterns which results in the definition of the pattern tree. This tree contains all possible patterns and specialization can be used to enumerate them. Additionally we proofed the completeness of this tree. As last step we presented a mining algorithm which is able to use an upper bound for the frequency difference to only test appropriate parts of the pattern tree to find the top k patterns.

We validated our implementation with theoretical results on the number of possible patterns which ensures a correct implementation. With two biological datasets we tested the different pattern matching methods which reveals that there is little difference between them. The reason is that the trees in the dataset are too small and too different to uncover differences between the four methods. In general there are many frequent and very similar patterns which led to the development of several pattern filters which restrict the algorithm to find only important patterns. With two HSC datasets - one with and one without TGFβ - we were able to recover the previously known fact that cells with TGFβ die earlier. In the second test we looked for patterns distinguishing between HSCs and MPPs and were able identify patterns unique to either one of those data sets.

Chapter 6

Conclusion

6.1 Summary of this Thesis

In this thesis we presented three different methods to analyze and visualize single cell stem cell data. First, we developed a new algorithm to detect redundant markers in FACS data based the comparison of SPADE result trees. Second, we presented a new visualization approach for FACS data which is able to preserve the internal cluster structure in the visualization. Finally we introduce a fast algorithm to discover differentialy occurring patterns in stem cell trees.

The marker selection algorithm is based on the hierarchical clustering of SPADE and is able to distinguish between the intrinsic variation of SPADE and changes which occur due to the removal of one marker. We evaluated our proposed three distance measures together with the cluster count adjustment as well as the feature selection methods. The algorithm was evaluated on two biological datasets.

To overcome several problems with current visualizations for FACS data we introduced a new approach combining SPADE clustering together with tree preserving embeddings. We compared our optimizations with the original code and measured a speed up of factor 30 for the TPE part. In the analysis with expert knowledge the visualization was able to recover the underlying linage tree. The comparison to other methods showed the large advantage of preserving hierarchical structure during the visualization. As last point we demonstrated the timepoint visualization on biological data to trace MSCs.

With the pattern discovery algorithm we address the problem of finding differentialy occurring patterns. We validated the algorithm with theoretical results to ensure a error free implementation. On an HSC dataset with two conditions we were able to rediscover the fact that HSCs with TGFβ die earlier than without. Through the application of pattern filters we were able to find some interesting patterns which may help in the discrimination between HSCs and MPPs.

6.2 Outlook

Here we present several ideas and new approaches which can be tested in the future to improve the presented methods.

To be applicable by biologists there needs to be an easy to use graphical user interface (GUI) to the presented algorithm. This is definitely one of the next steps to go, at least for the visualization and pattern discovery algorithm. The broader application allows the evaluation on a variety of different datasets and further improvement and adjustments.

Besides the GUI one further idea is to use the same analysis tools as for FACS scatter plots on the new visualization. This means to draw gates on the visualization to select different cell populations and maybe leads to a higher purity of selected populations.

The main problem with the pattern mining algorithm is the large number of found and very similar patterns. We reduced this problems by implementing several filters which selects which patterns can be found. Here further research is needed, e.g. one idea is to remove patterns which have a successor in the pattern tree which has a better score. So the pattern itself is of less value because there is a new pattern which is more specialized and has a better score.

Probabilistic matching is also a point which is a possible topic for further research. Until now matches are binary either there is a match or not. With probabilistic matching a pattern can match a tree e.g. with 80% certainty. In nature there are a lot of processes which are stochastic so the new matching would lead maybe to a better way of detecting often occurring patterns in stem cell trees.

List of Figures

Bibliography

[1] What are the unique properties of all stem cells? [Stem cell information]. http://stemcells.nih.gov/info/basics/basics2.asp, December 2012.

[2] J. M. Gimble, A. J. Katz, and B. A. Bunnell. Adipose-derived stem cells for regenerative medicine. *Circulation Research*, 100(9):1249–1260, November 2007.

[3] K. Kokkaliaris, D. Loeffler, and T. Schroeder. Advances in tracking hematopoiesis at the single-cell level. *Curr. Opin. Hematol.*, 19(4):243–249, July 2012.

[4] C. Marr, M. Strasser, M. Schwarzfischer, T. Schroeder, and F. J. Theis. Multi-scale modeling of GMP differentiation based on single-cell genealogies. *The FEBS Journal*, 279(18):3488–3500, June 2012.

[5] J. Krumsiek, C. Marr, T. Schroeder, and F. J. Theis. Hierarchical differentiation of myeloid progenitors is encoded in the transcription factor network. *PLoS ONE*, 6(8), August 2011.

[6] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2006.

[7] M. H. Julius, T. Masuda, and L. A. Herzenberg. Demonstration that antigen-binding cells are precursors of antibody-producing cells after purification with a fluorescence-activated cell sorter. *Proc Natl Acad Sci USA*, 69(7):1934–1938, July 1972.

[8] J. W. Tung, K. Heydari, R. Tirouvanziam, B. Sahaf, D. R. Parks, and L. A. Herzenberg. Modern flow cytometry: A practical approach. *Clin Lab Med*, 27(3), September 2007.

[9] S. Yamazaki, A. Iwama, S. Takayanagi, K. Eto, H. Ema, and H. Nakauchi. TGF-β as a candidate bone marrow niche signal to induce hematopoietic stem cell hibernation. *Blood*, 113(6):1250–1256, May 2009.

[10] L. Rossi, K.K. Lin, N.C. Boles, L. Yang, K.Y. King, M. Jeong, A. Mayle, and M.A. Goodell. Less is more: Unveiling the functional core of hematopoietic stem cells through knockout mice. *Cell Stem Cell*, 11(3):302–317, September 2012.

[11] S. H. Orkin and L. I. Zon. Hematopoiesis: An evolving paradigm for stem cell biology. *Cell*, 132(4):631–644, February 2008.

[12] P. J. Quesenberry, G. A. Colvin, and M. S. Dooner. The stem cell continuum: A new model of stem cell regulation. In A. M. Wobus and K. R. Boheler, editors, *Stem Cells*, pages 169–183. Springer Berlin Heidelberg, January 2006.

[13] L. E. Purton and D. T. Scadden. Limiting factors in murine hematopoietic stem cell assays. *Cell Stem Cell*, 1(3):263–270, September 2007.

[14] M. J. Kiel, O. H. Yilmaz, T. Iwashita, O. H. Yilmaz, C. Terhorst, and S. J. Morrison. SLAM family receptors distinguish hematopoietic stem and progenitor cells and reveal endothelial niches for stem cells. *Cell*, 121(7):1109–1121, July 2005.

[15] S. L. McKinney-Freeman, O. Naveiras, F. Yates, S. Loewer, M. Philitas, M. Curran, P. J. Park, and G. Q. Daley. Surface antigen phenotypes of hematopoietic stem cells from embryos and murine embryonic stem cells. *Blood*, 114(2):268–278, September 2009.

[16] M. Rahman. Introduction to flow cytometry. Technical report, 2006.

[17] L. A. Herzenberg, J. Tung, W. A. Moore, L. A. Herzenberg, and D. R. Parks. Interpreting flow cytometry data: a guide for the perplexed. *Nature Immunology*, 7(7):681–685, 2006.

[18] D. R. Parks, M. Roederer, and W. A. Moore. A new "Logicle" display method avoids deceptive effects of logarithmic scaling for low signals and compensated data. *Cytometry Part A*, 69A(6):541–551, 2006.

[19] T. Schroeder. Long-term single-cell imaging of mammalian stem cells. *Nature Methods*, 8:S30–S35, April 2011.

[20] M. A. Rieger, P. S. Hoppe, B. M. Smejkal, A. C. Eitelhuber, and T. Schroeder. Hematopoietic cytokines can instruct lineage choice. *Science*, 325(5937):217–218, October 2009.

[21] P. Qiu, E. F. Simonds, S. C. Bendall, K. D. Gibbs, R. V. Bruggner, M. D. Linderman, K. Sachs, G. P. Nolan, and S. K. Plevritis. Extracting a cellular hierarchy from high-dimensional cytometry data with SPADE. *Nature Biotechnology*, 29(10):886–891, October 2011.

[22] N. Ye. *The Handbook of Data Mining.* Routledge, November 2004.

[23] S. C. Bendall, E. F. Simonds, P. Qiu, E. D. Amir, P. O. Krutzik, R. Finck, R. V. Bruggner, R. Melamed, A. Trejo, O. I. Ornatsky, R. S. Balderas, S. K. Plevritis, K. Sachs, D. Pe'er, S. D. Tanner, and G. P. Nolan. Single-cell mass cytometry of differential immune and drug responses across a human hematopoietic continuum. *Science*, 332(6030):687–696, May 2011.

[24] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *Journal of the ACM*, 49:16–34, January 2002.

[25] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw: Pract. Exper.*, 21(10):1129–1164, November 1991.

[26] M.-Y. Kao. *Encyclopedia of Algorithms*. Springer, August 2008.

[27] A. Rényi and G. Szekeres. On the height of trees. *Journal of the Australian Mathematical Society*, 7(04):497–507, October 1967.

[28] D. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures: Third Edition*. CRC Press, 2004.

[29] M. Crisan, C.-W. Chen, M. Corselli, G. Andriolo, L. Lazzari, and B. Péault. Perivascular multipotent progenitor cells in human organs. *Ann. N. Y. Acad. Sci.*, 1176:118–123, September 2009.

[30] S. Méndez-Ferrer, T. V. Michurina, F. Ferraro, A. R. Mazloom, B. D. Macarthur, S. A. Lira, D. T. Scadden, A. Ma'ayan, G. N. Enikolopov, and P. S. Frenette. Mesenchymal and haematopoietic stem cells form a unique bone marrow niche. *Nature*, 466(7308):829–834, August 2010.

[31] A. D. Shieh, T. B. Hashimoto, and E. M. Airoldi. Tree preserving embedding. *PNAS*, 108(41):16916–16921, November 2011.

[32] L. Van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(2579-2605):85, August 2008.

[33] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Comm. of the ACM*, 18(9):509–517, September 1975.

[34] C. L. Semerad, E. M. Mercer, M. A. Inlay, I. L. Weissman, and C. Murre. E2A proteins maintain the hematopoietic stem cell pool and promote the maturation of myelolymphoid and myeloerythroid progenitors. *PNAS*, 106(6):1930–1935, October 2009.

[35] C. Murre. Developmental trajectories in early hematopoiesis. *Genes Dev.*, 23(20):2366–2370, October 2009.

[36] G. Hinton and S. Roweis. Stochastic neighbor embedding. 15, 2002.

[37] T. Ideker and N. J. Krogan. Differential network biology. *Molecular Systems Biology*, 8(1), January 2012.

[38] K. Azadov. Anfrageoptimierung, visualisierung und automatisierte auswertung von annotierten stammzell-differenzierungsbäumen. Diplomarbeit, February 2010.

[39] H. S. Wilf. *Combinatorial Algorithms: An Update.* SIAM, 1989.

[40] R. A. Wright, B. Richmond, A. Odlyzko, and B. D. McKay. Constant time generation of free trees. *SIAM J. Comput.*, 15(2):540–548, May 1986.

[41] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *System Science and Cybernetics*, 4(2):100–107, July 1968.