# Ontwikkeling van technieken op basis van machinaal leren voor flowcytometriedata

# Development of Machine Learning Techniques for Flow Cytometry Data

Sofie Van Gassen

Promotoren: prof. dr. ir. T. Dhaene, prof. dr. Y. Saeys
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

Vakgroep Informatietechnologie
Voorzitter: prof. dr. ir. B. Dhoedt
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2016 - 2017

UNIVERSITEIT
GENT

ISBN 978-90-8578-995-6
NUR 980
Wettelijk depot: D/2017/10.500/30

Universiteit Gent
Faculteit Ingenieurswetenschappen en Architectuur
Vakgroep Informatietechnologie

| | |
|---|---|
| Promotoren: | prof. dr. Yvan Saeys |
| | prof. dr. ir. Tom Dhaene |

| | |
|---|---|
| Examencommissie: | prof. dr. ir. Gert de Cooman (voorzitter) |
| | prof. dr. ir. Tijl De Bie (secretaris) |
| | prof. dr. Sophie Janssens |
| | prof. dr. Peter Dawyndt |
| | dr. Emmanuel Gustin |
| | prof. dr. Yvan Saeys |
| | prof. dr. ir. Tom Dhaene |

Dit werk kwam tot stand in het kader van een specialisatiebeurs van het IWT-Vlaanderen (Instituut voor de aanmoediging van Innovatie door Wetenschap en Technologie in Vlaanderen).

Dit werk werd uitgevoerd in samenwerking met het Vlaams Instituut voor Biotechnologie.

Proefschrift tot het behalen van de graad van
Doctor in de ingenieurswetenschappen:
computerwetenschappen
Academiejaar 2016-2017

# Dankwoord

Na vier jaar onderzoek sta ik nu op het punt om mijn doctoraat over de analyse van flowcytometriedata af te ronden. Ik heb tijdens deze vier jaar niet alleen de kans gekregen om bijzonder veel bij te leren van mijn collega's, maar ook om samen vele gezellige momenten te beleven. Ik ben er zeker van dat ik hieronder nog namen vergeet expliciet te noemen, maar ik wil jullie allemaal oprecht bedanken voor jullie interesse in mijn onderzoek en jullie vriendschap!

In de eerste plaats wil ik mijn promotoren, Yvan en Tom, bedanken. Na mijn masterthesis overtuigden jullie mij om een doctoraat te starten, iets wat ik daarvoor nog helemaal niet in mijn mogelijke toekomstplannen had voorzien. Ik had een beetje schrik dat een doctoraat erg theoretisch zou zijn, en de algoritmes die ik ontwikkelde misschien helemaal niet in de praktijk toegepast zouden worden. Maar jullie zorgden voor goede contacten met zowel mensen in de machine learning community als met de mensen in het labo en gaven me de kans om heel praktijkgericht te werken. Bart, Martin, Hamida and Sophie, you all played an important role by allowing me to apply my algorithms on your data. Your never-ending enthusiasm about our new techniques really convinced me of the usefulness of our work and kept me motivated during my PhD.

Joeri, je begeleidde mijn masterthesis al vol overgave, maakte mij wegwijs in de IBCN-groep en stond steeds klaar om mij te helpen met allerlei praktische zaken. Leen, jammer genoeg werkte je maar korte tijd op flowcytometriedata, maar ik vond het superfijn om met jou te kunnen samenwerken. Joachim, ook jij kon steeds tijd maken voor een babbel als ik toevallig eens in de Zuiderpoort of het iGent-gebouw passeerde.

Met verloop van tijd werd niet de IBCN-groep, maar de DaMBi-groep mijn vaste werkomgeving, waar ik steeds bij iedereen terecht kon voor wetenschappelijk advies, een ontspannende babbel, of het halen van een soepje in de resto. Robrecht, we hadden al samen onze bachelor informatica en onze masterthesissen tot een goed einde gebracht en ook de voorbije vier jaar stond je altijd voor me klaar. We gingen samen naar IJsland en Seattle en besproken allerlei algoritmische problemen. Hopelijk kunnen we nog een tijdje collega's blijven en nog veel langer goede vrienden! Wouter, jij bent altijd bereid om tijd voor me maken, of ik nu een vraag heb over het ontwerpen van een wetenschappelijke figuur of over de etymologie van een random woord. Je bent voor mij het absolute voorbeeld van hoe gedreven een doctoraatsstudent kan zijn met interesse in allerlei soorten wetenschap, waar ik me zo goed mogelijk aan probeer te spiegelen. Liesbet, we zijn ongeveer samen gestart in de DaMBi groep en jij zorgde ervoor dat ik me van

in het begin welkom voelde in de groep. In Barts lab konden we steeds samen oplossingen zoeken voor allerlei vreemde R-problemen en jij vormde ook de schakel om langzamerhand de andere Bio-IT'ers beter te leren kennen. Arne, bij jou kon ik steeds terecht met al mijn IT-gerelateerde vragen. Bovendien vind ik het fantastisch dat jij een webtool hebt ontwikkeld voor mijn algoritme, en het zo een stuk toegankelijker gaat worden voor de mensen in het lab. Dani, you are increasing the usefulness of FlowSOM by incorporating it into a bigger pipeline, and I'm very happy with the opportunity to follow along with your work. I also always enjoy the company on the train rides to Leuven! Celine, ook al was je maar korte tijd in onze groep, ik vond het toch superfijn om met jou samengewerkt te hebben en dankzij jou een hele nieuwe soort algoritmes te ontdekken. Pieter, Paco, Sarah en Joris, jullie zorgen ook mee voor de gezellige DaMBi-sfeer, of het nu is tijdens een lunch in de cafetaria of in het UZ, tijdens de TeamStorm of gewoon in onze wekelijkse DaMBi-meetings.

Naast de DaMBi-groep, was het vooral de U_Bla-groep die mij wegwijs maakte in de wondere wereld van de flowcytometrie. Ik kwam in jullie groep zonder enige biologische achtergrondkennis, en jullie stonden steeds klaar om op al mijn vragen te antwoorden. Gert, van jou heb ik niet alleen superveel geleerd over de technische werking van de machine, je stond ook steeds klaar om allerlei vreemde data-artefacten te proberen verklaren en maakte de CYTO-conferenties een fantastische ervaring. Mary, jij genereerde niet alleen data voor mij, maar nam ook de tijd om mij experimenten mee te laten volgen en mij zelfs even zelf de tubes onder de flowcytometer te laten steken, een ervaring die mij absoluut extra inzicht heeft gegeven in waar al die data nu eigenlijk vandaan komt. Charlie, your enthusiasm in using the FlowSOM algorithm on your data was really motivating during my PhD! Simon, Dorine, Martijn, Nincy, Marieke, Melissa, Delfien, en vele anderen, jullie bezorgden me ook data om op te werken, zo ervoor zorgend dat ik nuttige algoritmes kon ontwikkelen en alle kleine eigenaardigheden van de data leerde kennen!

I also want to thank everyone from my temporary lab in Stanford. Nima, thank you so much for giving me the opportunity to visit your lab, to provide an interesting project and to make me feel very welcome! Brice and Dyani, I really enjoyed sharing the office with you. Ed and Hope, thanks for taking me along when you were doing a CYTOF experiment! Julie and Dave, you welcomed me in your home during my stay in Stanford and I really enjoyed all the artistic inspiration I got from you!

Britt, Pieter, Elissa en Annelies, ik ben heel blij dat ik jullie masterthesissen heb mogen begeleiden, ik denk dat ik daar minstens evenveel van heb bijgeleerd als jullie!

Barbara, Bernd, Clément, George, Heleen, Kinga, Myrthe, Pauline, Rachel and Sébastien, you all travelled to Ghent to learn about computational flow cytometry, bringing your own data and being brave enough to start writing scripts yourselves. I hope I convinced you all to keep exploring your data in new ways, even though these R scripts might seem a bit cryptic when first looking at them.

Minerva en Manon, bij jullie kan ik steeds terecht als alle wetenschappelijke dingen me even teveel worden. Ik kan jullie niet genoeg bedanken voor alle steun en vriendschap. Daarnaast wil ik uiteraard ook mijn familie bedanken, en in het bijzonder mijn mama, voor alle goede zorgen en fijne momenten. Toen ik besloot informatica te gaan studeren had ik nooit verwacht om uiteindelijk toch in een omgeving terecht te komen die heel dicht aansluit bij jouw en papa's werk, maar het is dankzij jullie dat ik altijd al geboeid ben geweest door wetenschap en dat hopelijk nog lang ga blijven! Natuurlijk mag ook Nicolas' familie niet ontbreken: Marijke, Dirk en Pauline, ook bij jullie kan ik steeds terecht als ik eens nood heb aan ontspanning.

En dan uiteraard, als allerlaatste en allerbelangrijkste, Nicolas. Zonder jou had ik dit nooit gekund, en elke avond bij jou thuiskomen is het fijnste gevoel ter wereld. Je staat altijd voor me klaar, ook op de moeilijke momenten vol stress voor deadlines en zelfs toen ik besloot plots twee maand naar Amerika te vertrekken heb je me altijd volop gesteund. Samen kunnen we de wereld aan. ♡!

*Gent, mei 2017*
*Sofie Van Gassen*

# Table of Contents

# List of Abbreviations

| | |
|---|---|
| **2D** | Two-dimensional |
| **AIDS** | Acquired immune deficiency syndrome |
| **BAL** | Bronchoalveolar lavage |
| **CD** | Cluster of differentiation |
| **DC** | Dendritic cell |
| **FACS** | Fluorescence-activated cell sorting |
| **FlowCAP** | Flow Cytometry: Critical Assessment of Population Identification Methods project |
| **FloReMi** | Flow density survival regression using minimal feature redundancy |
| **FMO** | Fluorescence-minus-one |
| **FSC** | Forward scatter |
| **GB** | Gigabytes |
| **GUI** | Graphical user interface |
| **HDM** | House dust mite |
| **HIV** | Human immunodeficiency virus |
| **IRF8** | Interferon regulatory factor-8 |
| **KNN** | *k*-Nearest neighbors |
| **KO** | Knock-out |
| **MF** | Macrophage |
| **MFI** | Mean or median fluorescence intensity |
| **MST** | Minimal spanning tree |
| **NK cells** | Natural killer cells |
| **PBMCs** | Peripheral blood mononuclear cells |
| **PBS** | Phosphate-buffered saline |
| **PMT** | Photomultiplying tube |
| **RNA** | Ribonucleic acid |
| **SOM** | Self-organizing map |

| | |
|---|---|
| **SPADE** | Spanning tree progression of density normalized events |
| **SSC** | Side scatter |
| **TCR** | T cell receptor |
| **t-SNE** | t-Distributed stochastic neighbor embedding |
| **WT** | Wild type |

# Samenvatting
# – Summary in Dutch –

Het immuunsysteem, een complex systeem dat uit veel verschillende celtypes bestaat, verdedigt ons lichaam tegen allerlei indringers en speelt een grote rol in de meeste ziekten. Bij infectieziekten, zoals bijvoorbeeld griep, bestrijdt het de ziektekiem, maar soms kan het ook ziekten veroorzaken als er iets misloopt bij de werking, zoals bijvoorbeeld bij allergieën. Het immuunprofiel van een patiënt opstellen kan helpen om een diagnose te stellen of om een behandeling op te volgen, terwijl het bestuderen van *in vitro* immuuncellen of het immuunsysteem van proefdieren cruciaal is bij de ontwikkeling van geneesmiddelen. Om een immuunprofiel te bepalen wordt vaak flowcytometrie gebruikt, een *high-throughput* techniek waarbij biologische stalen worden gekleurd met antilichamen die aan fluorochromen gebonden zijn en waarbij de cellen passeren door een vloeistofsysteem. Met behulp van een optische installatie met lasers en bandfilters wordt de fluorescentie-emissie van elke individuele cel gemeten, waardoor de aanwezigheid van specifieke eiwitten of 'merkers' op het celoppervlak gedetecteerd wordt. Zo kunnen verschillende celtypes geïdentificeerd worden en krijgt men inzicht in het immuunprofiel van de patiënt. Deze techniek kan informatie opmeten van duizenden individuele cellen per seconde.

De analyse van flowcytometriedata bestaat typisch uit meerdere onderdelen. Eerst moeten een aantal stappen voor kwaliteitscontrole doorlopen worden, zoals het verwijderen van verkeerde metingen veroorzaakt door obstructies in de machine, dode cellen of doubletten. Sommige artefacten van het optisch systeem moeten ook gecorrigeerd worden door de data te compenseren en te transformeren. Vervolgens worden de verschillende celtypes geïdentificeerd. Traditioneel wordt dit gedaan door de data te '*gaten*', een procedure waarbij kleinere groepen van cellen herhaaldelijk geselecteerd worden door veelhoeken over tweedimensionale spreidingsdiagrammen te tekenen. Het detecteren van de verschillende celtypes is zelden het uiteindelijke doel van het onderzoek. Vaak wordt een bijkomende analyse uitgevoerd op de aantallen of percentages van de celtypes, om verschillen tussen patiëntengroepen of proefdieren te bepalen.

In de voorbije jaren is het aantal merkers dat gelijktijdig kan opgemeten worden sterk toegenomen. Waar de originele machines in de jaren '70 slechts twee kleuren konden meten en dit geleidelijk aangroeide tot 12 in de jaren '90, steeg het in de laatste tien jaar tot 30 en meer door de ontdekking van fluorochromen met kleinere emmissiespectra en de ontwikkeling van massacytometrie. Hierdoor is de traditionele manier om deze data te analyseren niet langer toereikend. Bij kleinere datasets was het observeren van twee parameters per keer voldoende om de celpopulaties te identificeren, maar dit beeld is te gelimiteerd voor hoogdimensionale datasets. Het is niet alleen tijdrovend, maar ook zeer bevooroordeeld naar de verwachte populaties toe. Veel cellen worden weggelaten en nooit geanalyseerd en het gebeurt zelden dat alle merkers bestudeerd worden voor één cel. Daarbij komt ook nog dat naarmate meer celpopulaties gedetecteerd worden, het moeilijker wordt om te identificeren welke (combinaties van) celpopulaties voorspellend kunnen zijn voor een klinisch resultaat.

Machinaal leren, een onderzoeksveld in de computerwetenschappen waar modellen worden geleerd uit data, zou kunnen helpen om deze problemen aan te pakken. Het heeft algoritmes die kunnen omgaan met hoogdimensionale data (zoals dimensionaliteitsreductie en kenmerkselectie), algoritmes om subpopulaties te selecteren in de data ('clustering') en algoritmes om waarden te voorspellen, zoals een groepslabel of een overlevingstijd, vanuit een beschrijving van een patiënt (classificatie en regressie). De meeste van deze technieken kunnen nuttige toepassingen vinden in flowcytometrieonderzoek.

In deze thesis evalueren we welke algoritmes het best geschikt zijn voor dit type data en ontwikkelen we een aantal specifieke oplossingen voor verschillende situaties. Het eerste hoofdstuk bevat een algemene inleiding van de flowcytometrietechniek, illustreert het gebruik ervan in immunologisch onderzoek en toont een kort overzicht van algoritmes voor machinaal leren.

In het tweede hoofdstuk ontwikkelen we een beter visualisatiealgoritme voor flowcytometriedata, omdat de traditionele 2D-spreidingsdiagrammen incompleet waren en alternatieve technieken zoals SPADE en viSNE de miljoenen cellen die in flowcytometriestalen gemeten worden niet konden verwerken. FlowSOM gebruikt een *self-organizing map*, die het computationeel goed schaalbaar maakt, en bevat een extra metaclusteringsstap, die clusters met verschillende maten en vormen toelaat. De clusters van de *self-organizing map* worden in een minimaal opspannende boom gevisualiseerd. Dit beeld heeft een zeer intuïtieve interpretatie waarin de verschillende takken diverse celtypes voorstellen en de verschillende nodes in een tak kleine variaties binnen een specifieke celpopulatie weergeven.

Terwijl de eerste versie van het FlowSOM-algoritme een volledig overzicht van een dataset kon geven, merkten we al snel dat het zonder extra moeite nog geen antwoord kon geven op verscheidene vragen van de immunologen, zoals 'Wat is het immunofenotypisch verschil tussen deze twee groepen patiënten?' en 'Welke tak stelt de dendritische cellen voor?'. In het derde hoofdstuk beschrijven we extra mogelijkheden die geïmplementeerd werden in het FlowSOM R-pakket dat beschikbaar is op Bioconductor en gebruikers toestaat een meer volledige analyse van hun data te doen zonder veel extra werk.

Het vierde hoofdstuk beschrijft onze deelname aan de FlowCAP IV-wedstrijd. Het FlowCAP-consortium leverde een flowcytometriedataset van hiv-patiënten met een gekende progressietijd tot aids en was op zoek naar celpopulaties die deze progressiesnelheid konden voorspellen. We bouwden een pijplijn genaamd FloReMi, die eerst extensieve voorbewerking toepast om foute metingen in de bestanden te verwijderen en dan een combinatie van de bestaande flowDensity- en flowType-algoritmes gebruikt om automatisch zeer veel mogelijke populaties te detecteren. We pasten een kenmerkselectieprocedure toe om interessante populaties te vinden met minimale redundantie en de progressietijd werd voorspeld met behulp van een *random survival forest*. Onze uiteindelijke resultaten waren beter dan de inzendingen van de andere acht teams die aan deze wedstrijd deelnamen.

In het vijfde hoofdstuk wordt een overzicht van computationele flowcytometrietechnieken gegeven, inclusief de twee technieken uit de vorige hoofdstukken en verschillende technieken die in andere onderzoeksgroepen werden ontwikkeld. Hoewel er veel algoritmes bestaan, gebruiken de meeste mensen in het labo nog steeds de traditionele *gating*-methode om hun data te analyseren. Het is noodzakelijk om deze nieuwe technieken aan immunologen te introduceren en hen een overzicht te geven van alle verschillende methodes en hun voordelen. Zo kunnen ze een geïnformeerde beslissing nemen over wat voor hen het leren waard is om hun onderzoek vooruit te helpen.

Waar de focus van alle vorige hoofdstukken op flowcytometriedata lag, kunnen de algoritmes ook gebruikt worden in een context van massacytometrie. Dit is een variatie op flowcytometrie waar de cellen gelabeld worden met zeldzame aardmetalen in plaats van fluorochromen. Dit omzeilt de beperkingen van het optisch systeem en laat toe dat het aantal merkers stijgt tot 50 en meer. Aangezien massacytometrie steeds vaker in klinische studies wordt gebruikt, is het zeer belangrijk dat de waarden vergelijkbaar zijn tussen de verschillende stalen. Om dit te verzekeren worden stalen vaak per plaat verwerkt, maar zelfs dan kunnen batcheffecten optreden tussen verschillende platen. In het zesde hoofdstuk stellen we een nieuw algoritme voor, gebaseerd op normalisatie aan de hand van kwantielen, dat rekening houdt met de celtype-specifieke effecten die kunnen voorkomen door het incorporeren van het FlowSOM-algoritme.

Kort samengevat ontwikkelden we nieuwe algoritmes voor alle stappen van een flowcytometrieanalyse, gaande van voorbewerking over celtype-identificatie tot prognosevoorspelling. Het gebruik van machinaal leren liet ons toe om betere resultaten te behalen in vergelijking met bestaande technieken en verscheidene van onze methodes zijn in gebruik genomen door andere onderzoeksgroepen. Toch is het programmeren van scripts voor de meeste mensen in het labo nog net een stap te ver. Het zal wat tijd kosten tot deze nieuwe methodes geïmplementeerd worden in de commerciële oplossingen die geen programmeervaardigheden vragen en die door de meeste immunologen gebruikt worden. Ondertussen zullen sterke samenwerkingen tussen *wet lab*-teams en bio-informatici de computationele flowcytometrie naar een hoger niveau blijven tillen.

# Summary

The immune system, a complex system consisting of many different cell types, is our body's main defense mechanism against all kinds of intruders. It plays a huge role in most diseases, either by battling the culprit in infectious diseases such as flu, or because something goes wrong with its functioning in immune diseases such as asthma. Determining the immune profile of patients can help to diagnose them or to follow their treatment, whereas studying in vitro immune cells or the immune system of laboratory animals is crucial in medicine development. To determine an immune profile, a high-throughput technology called flow cytometry is often used. Biological samples are stained with antibodies bound to fluorochromes and the cells in solution are passed through a fluidics system. Using an optics system including lasers and bandpass filters, the fluorescence emission of every individual cell is measured, which reports the presence of specific proteins or 'markers' on the cell surface, allowing to identify different cell types and giving insight in the immune profile of a patient. This technology can capture information of thousands of individual cells per second.

The analysis of flow cytometry data will typically consist of multiple steps. First, some quality control steps should be executed, such as removing erroneous measurements caused by obstructions in the machine, dead cells or doublets. Some artifacts from the optics system need to be corrected as well, by compensating and transforming the data. Next, the different cell types can be distinguished. Traditionally, this is done by 'gating' the data, a procedure where subsets of cells are repetitively selected by drawing polygon shapes on two-dimensional scatter plots. Detecting the different cell types is rarely the final goal of the research. Often an additional analysis is executed on the cell type counts or percentages, to determine differences between patient groups or lab animals.

In recent years, the number of markers that can be measured simultaneously has strongly increased. Whereas the original machine design in the seventies was only able to measure two colors and this increased gradually to twelve in the nineties, with the discovery of fluorochromes with smaller emission spectra and the advent of mass cytometry, this number increased to thirty and more in the last ten years. This causes the traditional way of analyzing this data to fall short. While for smaller datasets observing two parameters at a time was enough to identify the cell populations, this view is just too limited for high-dimensional datasets. It is not only time-consuming, but also very biased towards the expected populations. Many cells are 'gated out' and never analyzed, and rarely all markers are studied for a single cell. Additionally, as more and more cell populations can be detected, it becomes harder to identify which (combinations of) cell populations can be predictive for a clinical outcome.

Machine learning, a branch of computer science in which models are learned from data, might help to tackle these problems. It has algorithms to handle high-dimensional data (such as dimensionality reduction and feature selection), algorithms to select subpopulations in the data (called clustering) and algorithms to predict values, such as a group label or a survival time, from a description of a patient (classification and regression). Most of these machine learning techniques can find useful applications in flow cytometry research.

In this work, we evaluate which algorithms are best suited for this type of data and develop several specific solutions for different use cases. The first chapter is a general introduction of the flow cytometry technique, showing its uses in immunology research and a short overview of machine learning approaches.

In the second chapter, we develop a comprehensive visualization tool for flow cytometry data, as the traditional 2D scatter plots were incomplete, and alternative techniques such as SPADE and viSNE were not able to handle the millions of cells processed from flow cytometry samples. FlowSOM uses a self-organizing map, making it computationally very scalable, and includes an additional meta-clustering step, allowing clusters in strongly varying sizes and shapes. The clusters from the self-organizing map are visualized in a minimal spanning tree, a view which has a very intuitive interpretation with the separate branches representing different cell types and the separate nodes in the branches representing small variations in a specific cell population.

While the first version of the FlowSOM algorithm managed to give a comprehensive overview of a dataset, we soon noticed that it was not yet able to answer many questions of the immunologists without effort, such as ‘What is the immunophenotypic difference between these two groups of patients?’ and ‘Which branch represents the dendritic cells?’. In the third chapter, we describe some additional functionality that was implemented into the FlowSOM R package that is available on Bioconductor and that allows users to do a more complete analysis of their data without much extra work.

The fourth chapter describes our participation in the FlowCAP IV challenge. The FlowCAP consortium provided a flow cytometry dataset of HIV patients with known progression time to AIDS, and were looking for cell populations which could predict this progression rate. We built a pipeline called FloReMi, which first applied extensive preprocessing to clean the files and then used a combination of the existing flowDensity and flowType algorithms to automatically detect many possible populations. We applied a supervised feature selection procedure to find populations of interest with minimal redundancy and the progression time was predicted using a random survival forest. Our final results outperformed the submissions of the other eight teams participating in this challenge.

In the fifth chapter, a review of computational flow cytometry techniques is given, including the two techniques from the previous chapters and many other techniques developed in different research groups. Even though several algorithms exist, most people in the lab still use the traditional gating approach to analyze their data. It is necessary to introduce these new techniques to immunologists and to give them an overview of all the different approaches and their advantages, so they can make an informed decision about what could be worth learning to advance their research.

Where all previous chapters focused on flow cytometry data, the algorithms could also be used in mass cytometry settings. Mass cytometry is a variation on flow cytometry where the cells are labeled with rare earth metals instead of fluorochromes. This circumvents the limitations of the optics system and allows the number of markers to go up to fifty and more. As mass cytometry is used more and more often in clinical studies, it is of utmost importance that the values are comparable between the different samples. To ensure this, samples are often processed per plate, but even then, batch effects might pop up between different plates. In the sixth chapter, we propose a new normalization procedure based on quantile normalization, which takes into account the cell type specific effects that might be occurring by incorporating the FlowSOM algorithm.

In conclusion, we developed new tools for all steps of a flow cytometry analysis, going from preprocessing, over cell type identification up until prognosis prediction. Making use of machine learning techniques allowed us to improve compared to existing analysis tools and several of our methods have been adopted by other research groups. However, most people in the lab will not yet take the leap to start programming scripts. It will take some time until these new analysis tools are implemented in the commercial point-and-click solutions used by most immunologists. In the meantime, strong collaborations between wet lab teams and bioinformatics teams will keep pushing computational flow cytometry to a new level.

# 1 Introduction

In this chapter, we present some basic concepts to situate the conducted research. The main focus of this work lies on flow cytometry data, so we introduce flow cytometry and its specific terminology. We also describe its use in immunology research and present a selection of cell types to familiarize the reader with the interpretation of the data. Technical improvements to flow cytometry necessitated a different approach to the analysis, and we outline some different types of machine learning techniques which have proven useful. However, a number of challenges still remain, which are subsequently discussed. Finally, we summarize the main contributions of the presented work and outline the structure of this dissertation. At the end of this chapter, we provide an overview of the publications that were authored during this research period.

## 1.1 Flow Cytometry

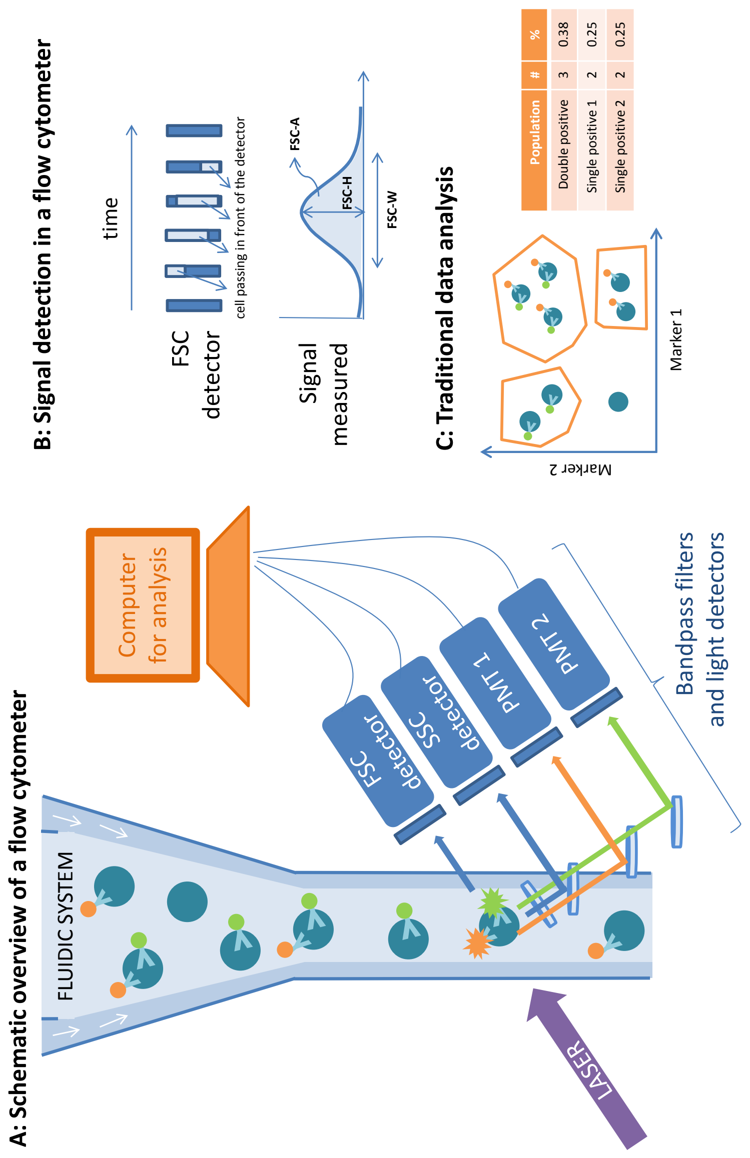
Flow cytometry literally means 'Measuring cells in a stream'. It is a technology which can measure specific cell properties for thousands of individual cells per second, allowing to detect different cell types. It can be used instead of microscopy when many cells need to be studied. Flow cytometry datasets can easily contain millions of cells. The properties measured consist of two light scatter values, which correlate to cell size and complexity, and a selection of fluorescently labeled **markers**: extra- or intracellular properties of the cells, such as DNA, RNA or proteins which allow to identify cell types of interest.

The first flow cytometers were developed in the seventies [1] and the current machines are still built by the same principles. A flow cytometer typically consists of four main parts:

1. A fluidic system, to enable the cells to pass by the measuring point one by one
2. Laser(s), to provide light scatter and excite the fluorochromes attached to the cells
3. Light detectors (PMTs) with analog-to-digital conversion, to detect the light emitted
4. A computer, for further analysis of the signals

A schematic overview is shown in Figure 1.1. The cells are suspended in a stream with hydrodynamic focussing and pass by a detection apparatus one by one. One or multiple lasers illuminate the cell, and the light that gets scattered by the cell is measured. We distinguish two kinds of **scatter**: forward scatter (FSC), which correlates with the cell size, and side scatter (SSC), which correlates with the cell complexity. Additionally, the colors emitted can be detected by passing the light through multiple bandpass filters, so that each photomultiplying tube (PMT) can measure the signal in a certain wavelength range or **channel**. Although some cells are autofluorescent and emit a color signal by themselves (e.g. macrophages), most do not emit much useful information in different colors. Rather, the cells are **stained** using monoclonal antibodies bound to fluorochromes. The fluorochromes are available in multiple colors and by attaching them to antibodies which are uniquely binding to known proteins, the color signal becomes a reporter for that specific marker. **Panel design**, the process of selecting which antibodies to include in the experiment and which fluorochromes to combine them with is one of the main tasks of a flow cytometrist. In recent years, initiatives have been taken to work towards standardized panels which can be reused throughout the community [2], but especially in research environments, experimentation in panel design is still key.

When processing the data, some artifacts caused by the optics system need to be taken into account. For example, a fluorochrome typically emits signal in a wavelength range, having a peak at a certain wavelength but also emitting some signal in the adjacent wavelengths. Bandpass filters then filter out the peak of this range to measure the emission. However, the edges of the range of one color might overlap with the peak region of another one, meaning that some signal might be detected in a neighboring channel. This phenomenon is called **spillover**, and should be corrected for by applying **compensation**. To estimate how much false signal is detected in the other channels, single-stained samples or beads can be used. The


A: Schematic overview of a flow cytometer
FLUIDIC SYSTEM
LASER
FSC detector
SSC detector
PMT 1
PMT 2
Computer for analysis
Bandpass filters and light detectors
B: Signal detection in a flow cytometer
time
FSC detector
cell passing in front of the detector
Signal measured
FSC-A
FSC-H
FSC-W
C: Traditional data analysis
Marker 1
Marker 2

| Population | # | % |
|---|---|---|
| Double positive | 3 | 0.38 |
| Single positive 1 | 2 | 0.25 |
| Single positive 2 | 2 | 0.25 |

***Figure 1.1:*** *Schematic overview of the main principles of a flow cytometry experiment. (A) The four main parts of a flow cytometer: a fluidic system, a laser, light detectors and a computer. (B) How a cell gets measured by the light detectors. For every marker, height (-H), width (-W) and area (-A) of the signal can be recorded. (C) In a traditional analysis, 2 markers are plotted against each other and populations of interest are defined by drawing polygons.*

smaller the emission range of the fluorochrome, the better, as less overlap will lead to purer results and allow to include more colors in the panel. The development of fluorochromes with a very narrow emission range is one of the technological advances made in the last decade which allows to measure more markers simultaneously.

Another artifact is caused by the analog-to-digital conversion. The signal is typically measured on an exponential scale, but in most cases the actual numeric value is not really of interest. It is enough to know whether the value is high, meaning the marker is present, or the value is low, meaning the marker is not present. It is rarely necesary to distinguish cell types by having a dimmer positive signal than others. To make it easier to distinguish the positive and negative cells, researchers used to apply a logarithmic **transformation** to the data. However, this leads to artifacts such as many events being piled up on the axis at zero, giving a skewed view on the data. Nowadays, mostly a biexponential or **logicle** transformation is used [3], as illustrated in Figure 1.2. This transformation is linear around zero and becomes logarithmic for bigger values, resulting typically in positive and negative populations with a close to Gaussian distribution. In many cases, the split between the positive and negative population can be determined quite easily. However, it is not always that simple when the positive population is either very small or having a high variance, resulting in a continuous smear. In those cases, a negative control or an **FMO** (fluorescence-minus-one) sample can help. By staining the sample with the whole panel except the marker of interest, the researcher can determine the highest numeric values that still belong to the negative population.

Once the data is compensated and transformed, the researcher will typically **gate** the data. This is an interative process of inspecting 2D scatter plots representing 2 markers for all individual cells, and drawing polygons (or 'gates') to identify subsets of interest. New scatter plots of these subsets can then be analyzed, building a hierarchical decision process to identify the cell populations of interest. An example of such a gating strategy is given in Figure 1.3.

Some artifacts might be caused by the sample itself and are typically removed in the first gating steps. Cells might stick together, resulting in markers on two different cells seemingly reported on one cell. However, as these clumps go through the stream, they will appear to have an elongated shape compared to the typically round cells. This information can be derived from comparing the total signal measured (FSC-A, area) to the maximum signal caused by the cell (FSC-H, height). For a doublet, the height will stay the same, but the area will double, because it takes twice as long to pass by the measuring point. By gating out cells with an unexpected ratio, **doublets** can be removed.

Dead cells are another reason for false signal. Because of the changed cell structure, non-specific binding of antibodies might happen. Typically, a live/dead marker is included in the panel to allow gating on live cells.

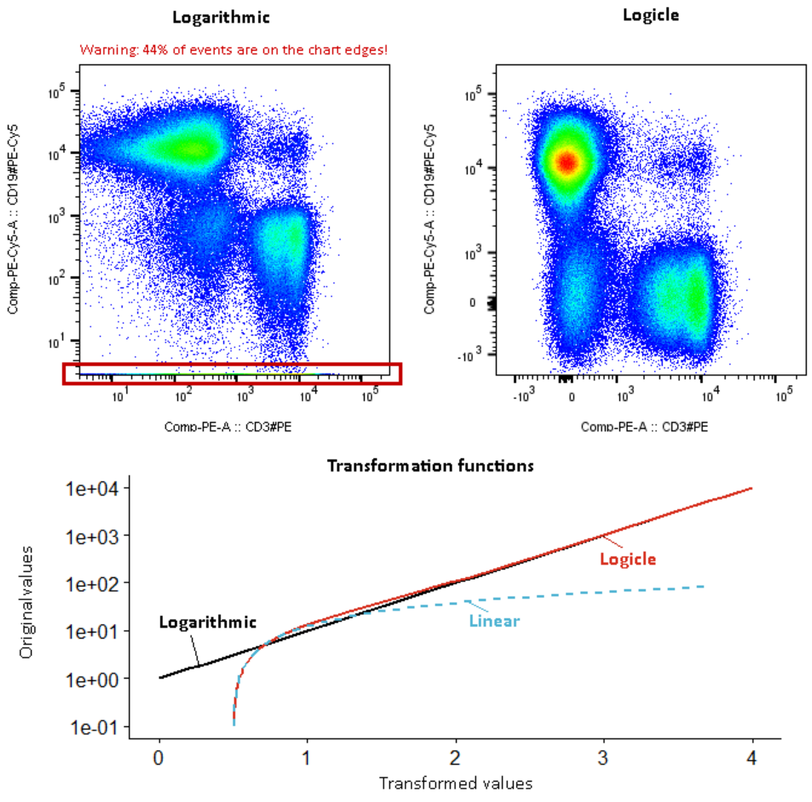


***Figure 1.2:*** *A logarithmic data transform results in many cells piled up on the axis. The logicle transforms stays linear for small values and becomes logarithmic for larger values, giving a clearer view of the data.*

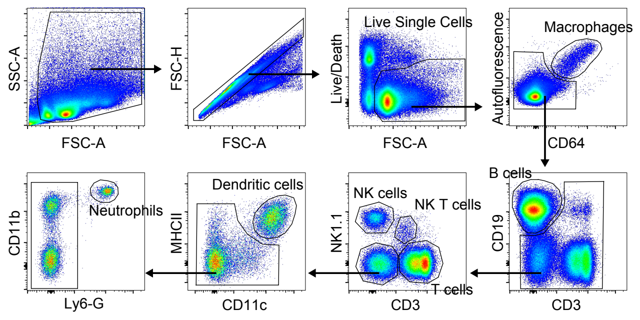


***Figure 1.3:*** *An example of a manual gating strategy. A scatterplot is shown of two markers, on which the researcher draws a polygon to select cell subsets of interest. For those subsets, they can again pick two different markers and select smaller subsets. This process is repeated until all cell populations of interest are identified.*

### 1.1.1 Uses of Flow Cytometry in Immunology

Although flow cytometry can be used in many fields, this work focuses on immunological data. Our immune system plays a crucial part in many diseases, either as a defense mechanism once a pathogen is detected or as the cause of the disease when something goes wrong with the immune system. Also in battling cancer, the immune system can play an important role. Therefore, studying the immune system is of utmost importance for medical research and immunophenotyping is very valuable in diagnosing patients.

The immune system is a complex system in which many different cell types can react to pathogens, such as viruses and bacteria, in different ways. The immune cells can also interact with each other, leading to a cascade of reactions once a pathogen is detected. To gain insight into diseases, distinguishing these different cell types is the first step. They can be identified by detecting proteins such as immunoglobulins or cluster of differentiation (CD) markers on their cell surface. Additionally, intracellular cytokine staining can give insight into the activation state of the cells.

Although the immune cells can reside in most tissues of the body, they travel through the blood to reach the places where they are needed. Immune cells are therefore also known as white blood cells. The most common option to collect immune cells are **peripheral blood mononuclear cells** (PBMCs). These can be collected by lysing red blood cells and serum from a blood sample, so that only the white blood cells remain. As the blood travels throughout the whole body, this can give an insight into the general immune state of the patient.

By analyzing a heterogeneous cell sample to identify the presence and proportions of various cell types of interest, the **imunophenotype** of a patient is determined. This gives insight into the disease state. For example, asthma, a disease caused by an allergic reaction in the airways, typically results in a strong increase in eosinophils. On the other hand, in patients with HIV (Human Immunodeficiency Virus), the CD4$^+$ T cell count drops strongly. These parameters can be used to follow up treatment or to prognose the further course of the disease.

In medical research, immunophenotyping is also often applied on **mice models**. By comparing wild-type mice with mice in which one specific gene is knocked out (KO mice), insights into the function of this gene can be gained. Advanced mouse models allow to knock out a gene only in specific cell types, allowing a more detailed inspection of the gene's function. By comparing mice with different treatments, effects of compounds can be studied for drug development.

Some flow cytometers also have the option to sort different cell populations into different vials, a technology called **fluorescence-activated cell sorting** (FACS). This enables researchers to harvest specific cells for further research, such as for example RNA sequencing.

### 1.1.2 A Selection of Cell Types

In this section, we shortly introduce a selection of immune cells and give an example of their function, just to familiarize the reader with the interpretation of the data. As the immune system is a very complex system, a full overview of all cell types and functionalities is beyond the scope of this work.

The development of red and white blood cells is called **hematopoiesis**. These cells have varying life spans, ranging from about 24 hours (e.g. neutrophils) to multiple years (e.g. memory B and T cells). Therefore, hematopoiesis is a continually ongoing process, depending on the demand for the different cell types. This is influenced by infections, allergic reactions, low oxygen levels etc. The development starts with stem cells in the red bone marrow which can differentiate into multiple more specific precursor cell types, to finally mature into functional red blood cells and immune cells ready to travel throughout the body to where they are needed.

We distinguish two big branches in this developmental tree: the **lymphoid** and the **myeloid** cells. The lymphoid cells all have one common progenitor and are the main cell types found in the lymphatic system: B cells, T cells and NK cells. Red blood cells, neutrophils, eosinophils, macrophages and dendritic cells all develop from the myeloid precursor cells. In Table 1.1 an overview of some of their functions and markers to identify them is given. As the size of a flow cytometry panel is limited, typically including ten to twenty markers, often multiple panels are used on aliquots of the samples, to separately study the lymphoid and myeloid populations in more detail.

A cartoon sketch of how these cells can interact is shown in Figure 1.4, a drawing made to reach out to a broader audience and explain the immune system to kids in primary schools.

***Table 1.1:*** *Overview of a selection of cell types, some of their functions and some of the markers that can be used to identify them.*

| CELL TYPE | EXAMPLE OF FUNCTION | MARKERS |
|---|---|---|
| B cells | Secrete antibodies and cytokines | CD19+ CD20+ CD21+ MHCII+ |
| T cells | Many subsets with a variety of functions | CD3+ |
| T helper cells | Help the maturation of other immune cells | CD3+ TCRb+ CD4+ CD8- |
| Cytotoxic T cells | Destroy virus-infected cells and tumor cells | CD3+ TCRb+ CD8+ CD4- |
| $\gamma\delta$ T cells | Present antigens to other T cells | CD3+ TCR$\gamma\delta$+ |
| Natural Killer cells (NKs) | Destroy virus-infected cells | NK1.1+ CD3- |
| Natural killer T cells | Share properties from T cells and NK cells | NK1.1+ CD3+ TCRb+ |
| Neutrophils | Destroy microbes and activate other immune cells | Ly6G+ CD11b+ |
| Eosinophils | Attack parasites | SiglecF+ CD11b+ CD24+ |
| Macrophages (MF) | Eat old cells and pathogens covered in antibodies | F4/80+ Autofluorescent |
| Conventional dendritic cells (cDCs) | Present antigens to other immune cells | CD11c+ MHCII+ |

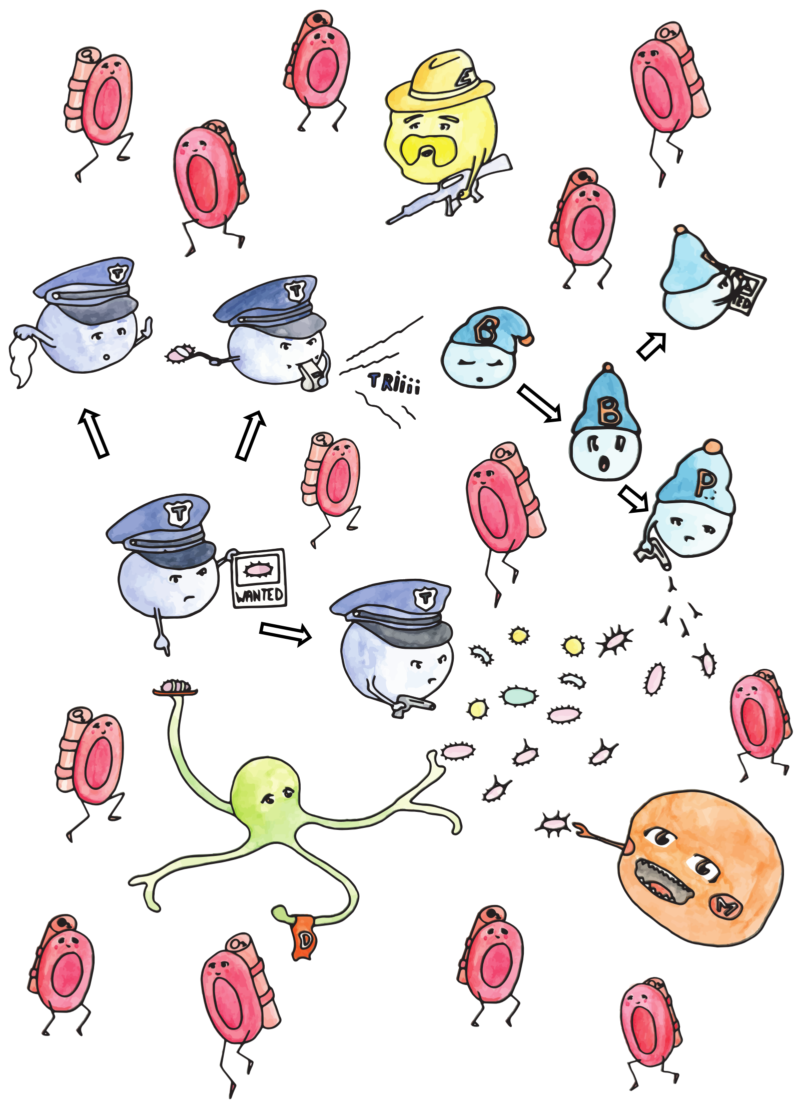


***Figure 1.4:*** *Cartoon sketch made for visiting elementary schools in the VIB project 'Wetenschap op stap'. It illustrates different immune cells and their interactions, including a dendritic cell (green), T cells (dark blue), B cells and a plasma cell (light blue), an eosinophil (yellow) and a macrophage (orange). The dendritic cell captures the pathogen and presents it to a T cell. The T cell, who was specifically looking for this intruder, can either go and attack the pathogen, make peace or alert other cells, such as the B cell, of its presence. The B cell can become a memory cell, also specifically looking for this pathogen, or can become a plasma cell, shooting the pathogen with antibodies. The macrophage will eat old cells or pathogens covered in antibody, and the eosinophil is a hunter for other types of pathogens, such as parasites.*

## 1.2 Machine Learning Techniques

To automate the analysis of cytometry data, we will make use of machine learning techniques. Machine learning is a branch of computer science where mathematical models are learned from data. The ultimate goal is to learn models that generalize beyond the example data that is used to train the model and can also predict results for new data points.

A dataset is typically described as a $n \times m$ numeric matrix $X$, where the $n$ rows represent the data points or 'instances' and the $m$ columns describe multiple properties of these instances, also called 'features'. One element of the matrix is denoted as $x_{ij}$ with $i$ indicating the row and $j$ indicating the column. The notation $x_i$ is used to refer to a full row, i.e. the description of one instance.

Machine learning techniques can be split in two main categories: supervised and unsupervised techniques, based on whether the algorithm is trained using a known outcome for the instances. For supervised techniques, an additional vector $y$ of length $n$ is given, describing the outcome $y_i$ for every instance $x_i$.

### 1.2.1 Supervised Machine Learning

For supervised techniques, a certain outcome $y_i$ is known for every training instance $x_i$ and the goal of the model is to predict this outcome for new instances. When this outcome is chosen from a discrete, finite set of values, we call this process **classification**. If the outcome is a continuous variable, it is called **regression**. While some algorithms can only be used for classification or for regression, many techniques can be used for both with small adaptations. **Survival time prediction** is a special case of regression analysis in which the continuous variable to predict is the time until a certain event (for example death, hence the name). In this setting, information might be missing for instances where the event did not happen in the time they were studied. However, some knowledge can be gained from the fact that the event did not happen up until a certain point, even if it is not known what happens from that point on. This type of data is called right-censored data and $y_i$ indicates either the time of the event or the time of the censoring. Another binary vector describes whether events happened or not. Modeling this data can benefit from an adapted approach instead of just removing the instances for which the event did not happen.

A well-known supervised machine learning technique is the **decision tree** [4], a hierarchical model presenting a sequence of simple decision steps. This model can be represented as a binary tree in which each internal node contains a test based on a specific feature and each leaf node contains a predictive value $\hat{y}$. This model has the advantage that it is easy to interpret and that it makes no strong assumptions about the distribution of $y$. To learn the model, the whole dataset is presented as input to the root node. An optimal split of the instances is decided

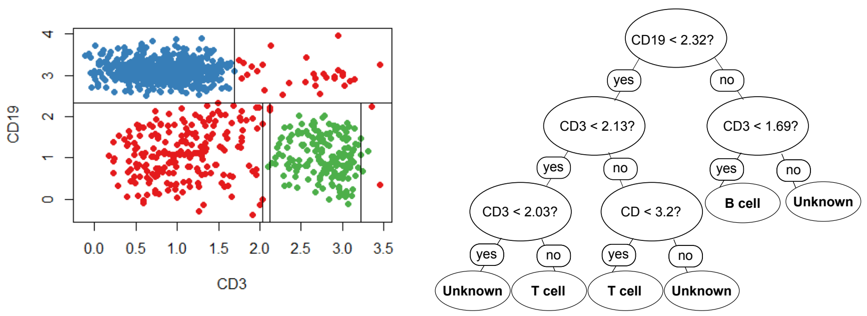


***Figure 1.5:*** *Toy example of a decision tree on a dataset with only two markers. The data is shown on the left and colored by manual labeling. The straight lines in the left image illustrate the decision boundaries as described by the decision tree on the right.*

based on one of the features. What is optimal depends on the setting and multiple evaluation criteria exist. For a classification problem, the two resulting subsets should each be as pure as possible, while for regression the two groups should for example have minimal variance. Once a certain split value $\theta$ is chosen for a feature $j$, all instances $x_i$ with $x_{ij} \leq \theta$ go as input to the left child of the root node, while all the others with $x_{ij} > \theta$ follow the right branch. This process is repeated until only a specified number of data points remain in the node, which is then labeled as a leaf node and given a predictive value. The outcome for new instances can be predicted by following a specific path through the tree depending on the evaluation of the tests in each internal node and returning the value assigned to the leaf node. A small toy example is shown in Figure 1.5.

An extension of the decision tree algorithm is the **random forest** [5]. As the name suggests, this is an ensemble of decision trees, in which each tree is trained on a random subset of the data by drawing $n$ instances with replacement from the original dataset. This excludes on average 37% of the data. Additional randomization is added by only evaluating a random selection of $m'$ features to find the optimal split in each node, with $m' < m$, often $m' = \sqrt{m}$. The trees are built until only one instance remains in every leaf, and their value $y_i$ is set as the predictive value. By using many different trees which were all trained on slightly different features and instance sets, a consensus result is returned for new instances (majority voting in the case of classification or the average value in case of regression). This results in a more robust model which is less prone to overfitting: the problem of modelling the training data so specifically that the model does not generalize anymore to new instances. The **random survival forest** [6], used in Chapter 4, is a variation on the traditional random forest, in which the split evaluation can take censored data into account. Instead of evaluating the variance of the child nodes, the survival difference between the child nodes is maximized.

Another algorithm for survival time prediction is the **Cox Proportional Hazard** model [7]. In contrast to the random survival forest, this is a linear model. As the name suggests, it is assumed that the effects of a unit increase in a feature is multiplicative with respect to the hazard rate, the event rate at time $t$ for all instances with survival time $y_i > t$. It returns a p-value on how strongly a feature relates to the outcome, but also builds a linear model with one or multiple features to estimate the hazard rate over time, which can be used to predict the survival time of a new instance.

### 1.2.2 Unsupervised Machine Learning

Unsupervised techniques do not make use of any known outcome, but look for structure in the data. **Clustering** techniques split the dataset in multiple groups of similar instances. A very well known clustering technique is **hierarchical clustering** [8], where initially every instance is defined as its own cluster. The two most similar instances or clusters are iteratively merged together until a certain number of clusters is reached. Several variations exist which use different similarity measures. As all data points need to be compared against each other to determine which pair is the most similar, hierarchical clustering has a running time which scales quadratically with the number of instances ($O(n^2)$).

A **self-organizing map** (SOM) [9] is another clustering algorithm, which compares the data points to a limited set of $k$ cluster centers, connected in a grid. It starts with a random initialization of the grid, and then iteratively picks an instance from the dataset, identifies the node representing the most similar cluster center and updates this node and the neighboring ones in the grid to become more similar to the datapoint. The learning rate and the neigborhood size are decreasing while looping over the dataset $R$ times. This algorithm results in a clustering for which the running time scales linearly with the number of instances ($O(Rkn)$), making it more suitable for large datasets. This algorithm is further explored in Chapter 2, a toy example is shown in Figure 1.6.

**Dimensionality reduction** techniques are another class of unsupervised algorithms, which try to describe the instances as well as possible while using a smaller number of features. The best known dimensionality reduction algorithm is **principal component analysis** (PCA) [10]. By computing the principal components, it becomes possible to select only those which explain the most variance in the dataset.

Another dimensionality reduction algorithm which is gaining popularity is **t-distributed Stochastic Neighbor Embedding** (t-SNE) [11]. In contrast to PCA, it is a non-linear method and focuses on preserving local structures instead of the whole global structure of the dataset. It does this by translating Euclidean distances

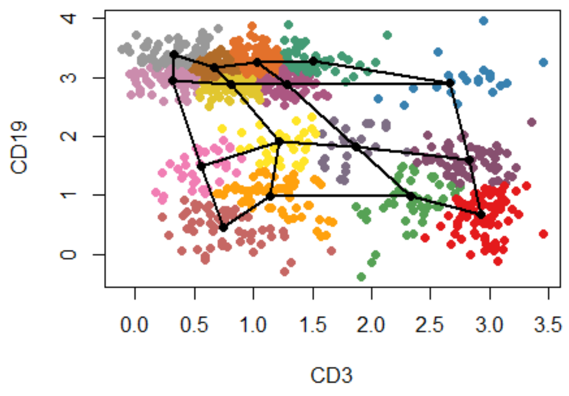


***Figure 1.6:** Self-organizing map clustering on the same toy dataset. The datapoints are colored by the final clustering result. The black lines and dots illustrate the grid which is stretched to fit the data.*

into conditional probabilities that represent similarities. It first embeds the dataset in a lower-dimensional space (randomly or by PCA) and then iteratively updates the placement of the data points to minimize the mismatch between the probabilities in the high- and low-dimensional space. By using an asymmetric scoring function, different types of errors in the pairwise distances in the low-dimensional space are not weighted equally, with a large cost for similar data points being far away in the new space, but only a small cost for asimilar data points being close in the new space. This ensures the focus on preserving local structure. An example can be seen in Figure 1.7.

Another way to gain insight into the structure of a dataset are **graph-based approaches**. A dataset can be transformed in a graph by regarding each instance as a node in the graph, and computing the distances between instances to assign as edge weights. Depending on the situation, the distance measure can be adapted, for example from euclidean distance to pearson correlation. Where a fully connected graph will result in a hair-ball structure and will provide limited information, less connected graphs can capture the main structure of the dataset. To plot the structure found by these graphs in a 2D space, force based algorithms can be used. These force the nodes apart and attempt to visualize the graph with as few overlap as possible between the edges.

One option to build such a graph is the **Minimal Spanning Tree** (MST), which will build a graph structure without any loops (a tree) which connects all nodes (spanning). It is minimal in the sense that those edges are chosen which have

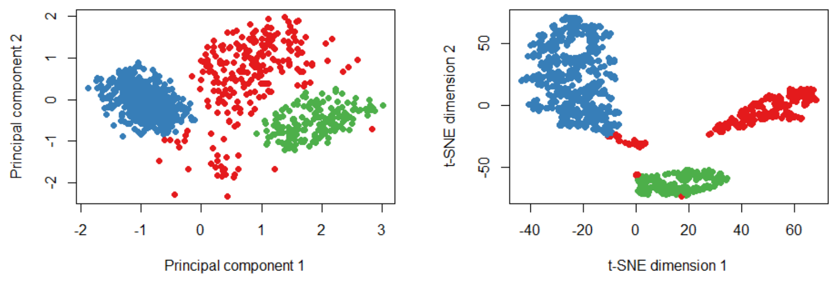


***Figure 1.7:*** *PCA and t-SNE executed on the same toy dataset. In this toy dataset, we go from 2 dimensions to 2 dimensions, so no real dimensionality reduction is executed. This allows to visually compare the behavior of the two methods. Whereas PCA (on the left), basically rotates the axes to make them correspond with maximal variance, t-SNE (on the right) captures the clusters present in the data and spreads them further apart, without taking the global structure into account.*

minimal weight, resulting in the most similar datapoints being connected.

Another variant is the $k$**-Nearest Neighbor graph** (KNN graph). For this approach, the $k$ most similar datapoints are determined for each instance, and the corresponding nodes are connected in the graph. This does not necessarily result in a spanning structure, but does allow for loops, which are not suppported by a MST.

### 1.2.3 Feature Selection

**Feature selection** is a different machine learning technique, which can be used in both supervised and unsupervised settings. Several issues can occur when the number of features is too large, such as unique features being ignored in favor of redundant features, or overfitting, modeling the data in such a detailed way that the model does not generalize anymore. Therefore, it might be necessary to select a subset of features to process in the subsequent algorithms.

In bioinformatics feature selection plays an important role, as the goal is often not just to build a diagnostic model, but also to gain further insight into the biological system. By selecting the features which are most informative about the patient state, new potential biomarkers can be identified.

In a supervised setting, the feature selection can be either done as a filter approach, where some score for each feature is computed upfront (for example based on **correlation** or mutual information with the outcome) or as a wrapper approach. In the wrapper approach, a subset is used to build a prediction model and then the subset is iteratively altered to improve the prediction model. This approach is not often used for bigger datasets, as the repeated training of prediction models becomes very time consuming. Instead, an embedded approach is used more often, where the supervised algorithm itself will try to determine the most important features and only one prediction model is needed.

For example, **LASSO** [12] is an adaptation of the least squares regression algorithm, which adds an extra regularization parameter to the objective function. Instead of just finding a linear function with minimal square errors, a balance needs to be found to also minimize the value of the coeficients in the model. This forces the model to discard as many features as possible, by putting their coefficient to zero.

A **random forest** can be used as an embedded feature selection method as well. Even though all features are used in the training of the model, an importance score is derived from how often a feature is chosen as a split criterium and how well it reduces the impurity or variance of the child nodes in those splits. This results in a feature ranking, from which the top features of interest can be selected.

In an unsupervised setting, the selection cannot be made based on which features have the most predictive value, but the goal is rather to select those features which contain as much information about the data as possible. This can be defined in multiple ways. Approaches to filter the most interesting features might select features with the largest variance or with a non-gaussian distribution. Other algorithms, such as **Dense Feature Groups** [13], try to remove redundant features, by clustering the features and keeping only one exemplar for each group. This can allow further insights in which features are highly correlated and describing similar information. This might be useful if some features are easier to collect for new data instances than others.

## 1.3 Computational Flow Cytometry

Due to technical improvements in the flow cytometry field, the need for automation and computational flow cytometry increased. As both the number of cells measured and the number of markers in the panel increase, a manual analysis becomes too laborious and biased to apply.

The problem of cell type identification corresponds to a clustering problem. The data is described as a matrix in which the cells are represented by rows and the markers measured by the flow cytometer as the columns. The goal is to find populations of cells which have similar marker expressions in the high-dimensional space.

The first clustering algorithms that have been applied to flow cytometry data were based on gaussian mixture models, such as FlowClust [14]. They make use of probabilistic distributions to build a model. The disadvantage of these techniques is that they make strong assumptions about the distribution of the data. FLAME

[15] also allows assymetric distributions, but still will not be able to model any convex clusters. Algorithms as FlowMerge [16] and SWIFT [17] try to solve this problem by aggregating clusters again at the end.

K-means is a well known clustering algorithm, but has the disadvantage that the number of clusters has to be known up front and the result strongly depends on the initialization of the algorithm. Density-based algorithms such as flowMeans [18] and FLOCK [19] try to estimate this parameter in an automated way.

A third option is based on graph clustering algorithms. However, due to the high-troughput nature of flow cytometry, it is impractical to represent the whole dataset as a graph. The SamSPECTRAL [20] algorithm applies an instance selection procedure first to make computations feasible, but this might lead to some information loss.

Next to cell type identification, patient classification is another task for which machine learning techniques might prove useful. This corresponds with traditional classification algorithms, but an extra step is necessary because flow cytometry data does not correspond to the typical matrix shape of machine learning datasets. Instead of one vector describing a patient, a whole matrix is available, which describes thousands to millions of individual cells from the patient.

In practice, most often a clustering algorithm is executed first, either on an aggregate of all the files or on the seperate files. The patient can then be described by the distribution of their cells over the different clusters, resulting in a traditional input vector for classification or regression algorithms. If the samples were clustered separately, the clusters first need to be matched across the files, a task which is not straightforward because clusters might shift or disappear between patients. On the other hand, these differences might cause too much noise to get a good clustering from the aggregated file.

### 1.3.1 The FlowCAP Challenges

To evaluate whether these machine learning approaches are useful in practice, the Flow Cytometry: Critical Assessment of Population Identification Methods (FlowCAP) project was initiated. The goal of FlowCAP is to advance the development of computational methods for the identification of cell populations of interest in flow cytometry data. Multiple algorithms were evaluated in the first two FlowCAP challenges [21], which provided benchmark datasets for cell type identification and patient classification.

The results of the first challenge indicated that most clustering algorithms show good correspondance with manual gating results. The differences between the automated results and the manual gating were of the same order as the differences between multiple researchers manually analysing the data.

In the second FlowCAP challenge several combinations of different clustering and classification algorithms were compared. However, the datasets used were either quite straightforward (almost all algorithms got a great score) or too difficult (no algorithms got a good result). Therefore, further comparisons should be made for practical use cases.

The third FlowCAP challenge included a dataset which was collected across multiple centers, to investigate a standardized approach to data collection and analysis [22]. Again it was concluded that the variability of automated gating procedures is as low or even lower than a manual gating approach, demonstrating that clustering approaches are suited to be incorporated in a flow cytometry analysis pipeline.

The fourth FlowCAP challenge [23] posed the problem of progression time prediction for HIV patients. Our participation in this challenge is described in Chapter 4. Although the results were not strong enough to build a prognostic model for use in clinical settings, interesting potential biomarkers were identified to aid further HIV research.

## 1.4 Challenges

When analyzing flow cytometry data, we can identify three subtasks: data preprocessing, detecting cell populations and diagnosing patients. All three tasks pose challenges for which machine learning approaches might prove useful.

The data preprocessing includes compensation and transformation, two steps for which algorithms are available and have been proven succesful. However, steps as important as quality control are mostly executed by hand or not at all. Ensuring stability of the flow stream over time is very important, as obstructions or some system failure might skew the data, shown in Figure 1.8. As datasets can differ a lot, learning a 'normal' pattern from the data itself might help to remove these issues.

The identification of cell populations can still cause problems too. While many clustering algorithms exist, most of them make assumptions about the data structure. This is not favorable for flow cytometry data, as different shapes and sizes might occur. Additionaly, it is often hard to identify which clustering level is of interest. In a manual analysis, the researcher will typically identify some big populations and then a few very detailed ones, describing small populations of interest. This makes it hard to use the manual gating result as a golden standard and evalute the clustering algorithms. Especially when rare populations of interest exist, most algorithms fail to separate them correctly. Additional knowledge in the form of biological control samples might need to be incorporated to fix these issues.

One option to test if the biologically relevant populations are found, is to combine it with the next step: modelling a clinical outcome. The challenge here lies in doing a good feature selection. When $m$ markers are measured and only one split

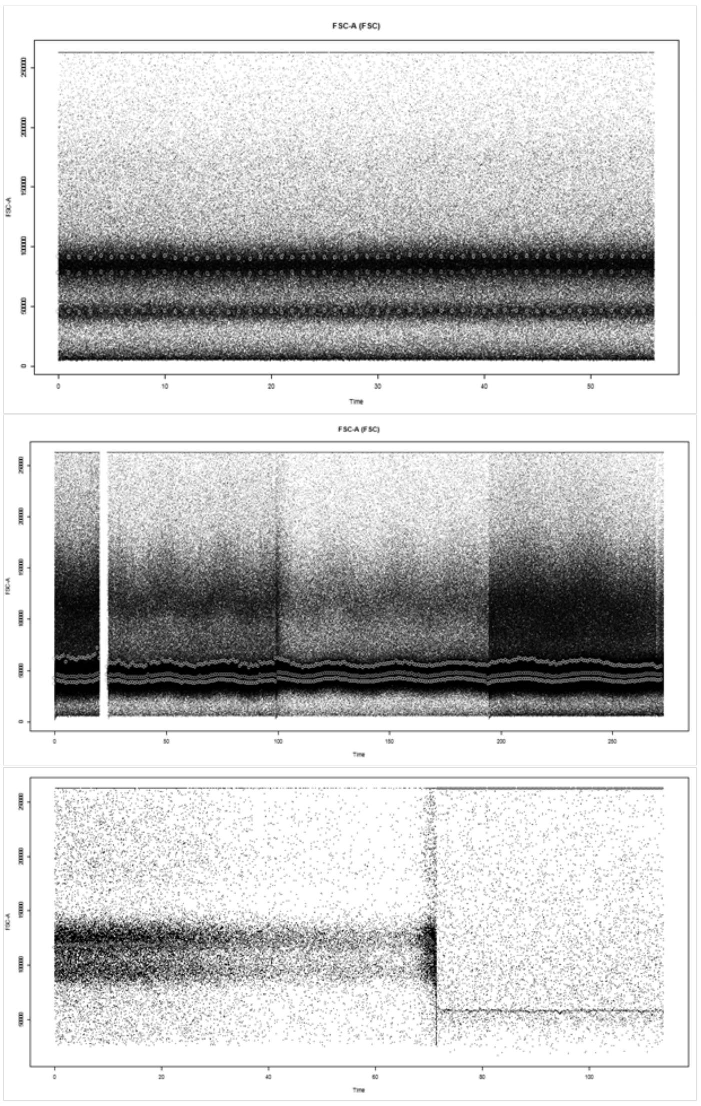


***Figure 1.8:*** *Examples of measurement values over time from three samples. In the first sample, everything stays stable over time, as would be expected. In the second panel, measurement artifacts are influencing the signal, with the measuring speed slowing down in the middle. In the third panel, the tube ran dry and air was measured at the end, an artifact that needs complete removal.*

is made in each marker to distinguish positive and negative cells, $2^m$ cell populations can be defined. This number easily becomes much larger than the number of patients in the study, increasing the risks of overfitting.

In general, two important challenges apply to all these subtasks. The first is the need for a clear visualization of any results. This is necessary for interpretation by the immunologists and crucial to gain new biological insights. The other challenge lies in the scalability of the methods. As big datasets can be generated by this high-throughput technique, a linear scalability in respect to the number of cells is strongly preferable.

## 1.5 Outline

This dissertation is composed of a number of publications that were realized within the scope of this PhD. The selected publications provide an integral and consistent overview of the work performed. The different research contributions are detailed in Section 1.6 and the complete list of publications that resulted from this work is presented in Section 1.7. Within this section we give an overview of the remainder of this dissertation and explain how the different chapters are linked together.

We start with a new visualization tool, called FlowSOM, in Chapter 2. Seeing the data is extremely important to interpret any results, especially when immunologists have been used to analyse this data in a very visual manner. A black box which just returns some numbers will be very hard for them to interpret and believe. Our algorithm is both computationally efficient and able to visualize large datasets, but in its first version, it was still limited to giving one simple overview. In Chapter 3, we describe some extensions to our FlowSOM algorithm, such as visualizing statistical differences between groups of patients or mice and automatic labelling of the clusters. In the fourth chapter, we describe our participation to the FlowCAP IV challenge. In this challenge, flow data of more than 200 HIV patients was available, for which their progression time to AIDS should be predicted. We built an entire pipeline, called FloReMi, starting from preprocessing and quality control, over feature extraction and selection, up until the final progression time prediction. In the fifth chapter, we give an overview of all current tools for flow cytometry. This includes our own tools, but also many others, to introduce immunolgists to the field of computational flow cytmetry. Finally, in the sixth chapter, we report about my stay at Stanford University, where I worked on a normalization algorithm for mass cytometry data in collaboration with dr. Nima Aghaeepour from the Nolan group.

Table 1.2 shows the main tasks that were highlighted in Section 1.4 and indicates which were targeted per chapter.

***Table 1.2:*** *An overview of the tasks discussed per chapter in this dissertation.*

| | Ch.2 | Ch.3 | Ch.4 | Ch.5 | Ch.6 |
|---|---|---|---|---|---|
| Preprocessing | | | ● | ● | ● |
| Population identification | ● | ● | ● | ● | |
| Patient diagnosis | | ● | ● | ● | |

## 1.6 Research Contributions

In Section 1.4, the problems and challenges for analyzing flow cytometry data are formulated. They are tackled in the remainder of this PhD dissertation for which the outline is given in Section 1.5. To conclude, we present an elaborated list of the research contributions within this dissertation:

- Design of a new visualization tool.
  - Implementation of the FlowSOM algorithm, making use of two-level clustering.
  - Implementation of extensions on the FlowSOM algorithm for easier use in the lab.
- Development of a survival prediction pipeline based on flow cytometry data.
  - Design of several preprocessing steps, such as automated singlet selection and time quality control.
  - Implementation of a feature extraction and feature selection pipeline.
  - Top ranked algorithm in the FlowCAP IV challenge.
- A review to introduce the state-of-the-art tools to immunologists.
- Design of a normalization algorithm for mass cytometry data
  - Demonstration of the need for a cell type based approach.
  - Implementation of a normalization algorithm, with clear improvements over the current practice.

## 1.7 Publications

The research results presented in this thesis have been published in or are submitted to scientific journals and have been presented on both national and international conferences. The following list provides an overview of the output during my PhD research.

### 1.7.1 Publications in international journals (listed in the Science Citation Index)

Published:

1. **Van Gassen, S.**, Callebaut, B., van Helden, M., Lambrecht, B.N., Demeester, P., Dhaene, T., and Saeys, Y. (2015) FlowSOM: using self-organizing maps for visualization and interpretation of cytometry data. *Cytometry Part A*, 87A(7), pages 636–645.

2. **Van Gassen, S.**, Vens, C., Dhaene, T., Lambrecht, B.N., and Saeys, Y. (2016) FloReMi: flow density survival regression using minimal feature redundancy. *Cytometry Part A*, 89A(1), 22–29.

3. Aghaeepour, N., Chattopadhyay, P., Chikina, M., Dhaene, T., **Van Gassen, S.**, Kursa, M., Lambrecht, B.N., et al. (2016) A benchmark for evaluation of algorithms for identification of cellular correlates of clinical outcomes. *Cytometry Part A*, 89A(1), 1621.

4. Saeys, Y., **Van Gassen, S.**, and Lambrecht, B.N. (2016) Computational flow cytometry: helping to make sense of high-dimensional immunology data. *Nature Reviews Immunology*, 16(7), 449462.

5. Guilliams, M., Dutertre, C.-A., Scott, C., McGovern, N., Sichien, D., Chakarov, S., **Van Gassen, S.**, et al. (2016) Unsupervised high-dimensional analysis aligns dendritic cells across tissues and species. *Immunity*, 45(3), 669684.

6. Sichien, D., Scott, C., Martens, L., Vanderkerken, M., **Van Gassen, S.**, Plantinga, M., Joeris, T., et al. (2016) IRF8 transcription factor controls survival and function of terminally differentiated conventional and plasmacytoid dendritic cells, respectively. *Immunity*, 45(3), 626640.

Submitted:

7. **Van Gassen, S.**, Gaudilliere, B., Dhaene, T., Angst, M., Nolan, G.P., Saeys, Y. and Aghaeepour, N. (2017) A Cross-Sample Cell-Type Specific Normalization Algorithm for Clinical Mass Cytometry Datasets. *Submitted to Cytometry Part A*

8. Govindarajan, S., Van Der Cruyssen, R., Verheugen, E., **Van Gassen, S.**, Saeys, Y., Iwawaki, T., Elewaut, D., Drennan, M.B. (2017) Post-transcriptional stabilization of cytokine mRNAs within invariant NKT cells requires the serine-threonine kinase IRE1a. *Submitted to Nature Immunology*

### 1.7.2 Conference Proceedings

1. Vens, C., **Van Gassen, S.**, Dhaene, T., and Saeys, Y. (2015). Complex aggregates over clusters of elements. In J. Davis and J. Ramon (Eds.), *Lecture Notes in Artificial Intelligence* (Vol. 9046, pp. 181193). Presented at the 24th International conference on Inductive Logic Programming (ILP), Berlin, Germany: Springer.

2. De Baets, L., **Van Gassen, S.**, Dhaene, T., and Saeys, Y. (2015). Unsupervised trajectory inference using graph mining. *Computational Intelligence Methods for Bioinformatics and Biostatistics, Lecture Notes in Bioinformatics* (Vol. 9874, pp. 8497). Presented at the Computational Intelligence Methods for Bioinformatics and Biostatistics, Lecture Notes in Bioinformatics.

3. **Van Gassen, S.**, Dhaene, T., and Saeys, Y. (2016). Machine learning challenges for single cell data. *Machine Learning and Knowledge Discovery in Databases, ECML PKDD 2016, PT III* (Vol. 9853, pp. 275279). Presented at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD) , Berlin: Springer-verlag Berlin.

### 1.7.3 Contributions to International Symposia (posters and oral presentations)

1. **Van Gassen, S.**, Dhaene, T., and Saeys, Y. (2014) Machine learning techniques for flow cytometry. *Machine Learning Summer School Iceland 2014, colocated with the 17th International Conference on Artificial Intelligence and Statistics* (Poster)

2. **Van Gassen, S.**, Callebaut, B., Vens, C., Dhaene, T., Lambrecht, B.N., Saeys, Y. (2014) Enhancing flow cytometry data visualization using FlowSOM. *29th Congress of the International Society for Advancement of Cytometry* (Poster)

3. Vens, C., **Van Gassen, S.**, Dhaene, T., and Saeys, Y. (2014) Complex aggregates over subsets of elements. *24th International Conference on Inductive Logic Programming* (Presentation)

4. **Van Gassen, S.**, Vens, C., Dhaene, T., Lambrecht, B.N., and Saeys, Y. (2015) FloReMi: Survival time prediction based on flow cytometry data. *2015 annual workshop on Statistical Methods for Post Genomic Data* (Presentation)

5. **Van Gassen, S.**, Dhaene, T., Lambrecht, B.N., Saeys, Y. (2015) Differential Population Identification Using FlowSOM *30th Congress of the International Society for Advancement of Cytometry* (Poster)

6. **Van Gassen, S.**, Van Helden, M., Guilliams, M., Dhaene, T., Saeys, Y. (2016) Automated Cell Type Annotation *31th Congress of the International Society for Advancement of Cytometry* (Poster)

7. **Van Gassen, S.**, Dhaene, T., and Saeys, Y. (2016) Machine learning challenges for single cell data *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases* (Nectar track, Poster and presentation)

8. **Van Gassen, S.**, Gaudilliere, B., Dhaene, T., Angst, M., Nolan, G., Saeys, Y. and Aghaeepour, N. (2017) A Cross-Sample Cell-Type Specific Normalization Algorithm for Clinical Mass Cytometry Datasets. *32th Congress of the International Society for Advancement of Cytometry* (Accepted for presentation)

### 1.7.4 Other Publications

1. **Van Gassen, S.**, Ruyssinck, J., Saeys, Y., and Dhaene, T. (2013) Stable Feature Selection Techniques for Microarray Data *8th BeNeLux Bioinformatics Conference* (Poster)

2. **Van Gassen, S.**, Saeys, Y., and Dhaene, T. (2014) Mining Flow Cytometry Data *24th Belgian-Dutch Conference on Machine Learning* (Poster)

3. **Van Gassen, S.**, Callebaut, B., Van Helden, M.J., Lambrecht, B.N., Demeester, P., Dhaene, T., and Saeys, Y. (2014) FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data *9th BeNeLux Bioinformatics Conference* (Poster)

4. **Van Gassen, S.**, Vens, C., Dhaene, T., Lambrecht, B.N., and Saeys, Y. (2015) FloReMi: Survival Time Prediction based on Flow Cytometry Data *10th BeNeLux Bioinformatics Conference* (Poster)

5. **Van Gassen, S.**, Vens, C., Dhaene, T., Lambrecht, B.N., and Saeys, Y. (2016) FloReMi: Survival Time Prediction based on Flow Cytometry Data *25th Belgian-Dutch Conference on Machine Learning* (Poster and short talk)

## References

[1] J. W. Gray, A. V. Carrano, L. L. Steinmetz, M. A. Van Dilla, D. H. Moore II, B. H. Mayall, and M. L. Mendelsohn. *Chromosome measurement and sorting by flow systems*. Proceedings of the National Academy of Sciences of the United States of America, 72(4):1231–1234, 1975.

[2] Y. Mahnke, P. Chattopadhyay, and M. Roederer. *Publication of optimized multicolor immunofluorescence panels*. Cytometry Part A, 77(9):814–818, 2010.

[3] 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, 69(6):541–551, 2006.

[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[5] L. Breiman. *Random forests*. Machine learning, 45(1):5–32, 2001.

[6] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. *Random survival forests*. The annals of applied statistics, pages 841–860, 2008.

[7] D. R. Cox. *Regression models and life-tables*. In Breakthroughs in statistics, pages 527–541. Springer, 1992.

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

[9] T. Kohonen. *The self-organizing map*. Proceedings of the IEEE, 78(9):1464–1480, 1990.

[10] H. Hotelling. *Analysis of a complex of statistical variables into principal components*. Journal of educational psychology, 24(6):417, 1933.

[11] L. van der Maaten and G. Hinton. *Visualizing data using t-SNE*. Journal of Machine Learning Research, 9(Nov):2579–2605, 2008.

[12] R. Tibshirani. *Regression shrinkage and selection via the lasso*. Journal of the Royal Statistical Society. Series B (Methodological), pages 267–288, 1996.

[13] L. Yu, C. Ding, and S. Loscalzo. *Stable feature selection via dense feature groups*. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 803–811. ACM, 2008.

[14] K. Lo, F. Hahne, R. R. Brinkman, and R. Gottardo. *flowClust: a Bioconductor package for automated gating of flow cytometry data.* BMC bioinformatics, 10:145, 2009.

[15] S. Pyne, X. Hu, K. Wang, E. Rossin, T.-I. T.-I. Lin, L. M. Maier, C. Baecher-Allan, G. J. McLachlan, P. Tamayo, D. A. Hafler, P. L. De Jager, and J. P. Mesirov. *Automated high-dimensional flow cytometric data analysis.* Proceedings of the National Academy of Sciences of the United States of America, 106(21):8519–8524, 2009.

[16] G. Finak, A. Bashashati, R. Brinkman, and R. Gottardo. *Merging mixture components for cell population identification in flow cytometry.* Advances in bioinformatics, 2009:247646, 2009.

[17] I. Naim, S. Datta, J. Rebhahn, J. S. Cavenaugh, T. R. Mosmann, and G. Sharma. *SWIFTscalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, Part 1: Algorithm design*. Cytometry Part A, 85(5):408–421, 2014.

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

[19] Y. Qian, C. Wei, F. Eun-Hyung Lee, J. Campbell, J. Halliley, J. A. Lee, J. Cai, Y. M. Kong, E. Sadat, E. Thomson, et al. *Elucidation of seventeen human peripheral blood B-cell subsets and quantification of the tetanus response using a density-based method for the automated identification of cell populations in multidimensional flow cytometry data*. Cytometry Part B: Clinical Cytometry, 78(S1):S69–S82, 2010.

[20] H. Zare, P. Shooshtari, A. Gupta, and R. R. Brinkman. *Data reduction for spectral clustering to analyze high throughput flow cytometry data*. BMC bioinformatics, 11(1):403, 2010.

[21] N. Aghaeepour, G. Finak, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann. *Critical assessment of automated flow cytometry data analysis techniques.* Nature methods, 10(3):228–38, 2013.

[22] R. Scheuermann, G. Finak, J. Ramey, J. Taghiyar, R. Stanton, A. Brandes, P. De Jager, P. Qiu, J. McCoy, D. Hafler, et al. *FlowCAP: comparison of automated and manual gating of standardized lyoplate flow cytometry data (P3374).* The Journal of Immunology, 190(1 Supplement):135–13, 2013.

[23] N. Aghaeepour, P. Chattopadhyay, M. Chikina, T. Dhaene, S. Van Gassen, M. Kursa, B. N. Lambrecht, M. Malek, G. McLachlan, Y. Qian, et al. *A*

*benchmark for evaluation of algorithms for identification of cellular correlates of clinical outcomes*. Cytometry Part A, 89(1):16–21, 2016.

# 2 FlowSOM: Using Self-Organizing Maps for Visualization and Interpretation of Cytometry Data

*"Seeing is believing."*

*To start exploring flow cytometry data, we will first develop a better visualization of the data. We reasoned that just extracting numbers would not convince the immunologists to switch from their traditional analysis tools, as it is often hard to interprete results from a black box. However, an improved visualization tool to explore the data can demonstrate that useful alternatives to gating exist, while still being quite straightforward to interprete. A few new visualizations tools have been proposed, but they are rather slow and can only show one marker at a time (even though they do use the high-dimensional information in training). There was a clear need for an algorithm that could visualize multiple markers in a quick way. In this chapter, we present FlowSOM, the algorithm we developed. On top of the visualization result, a meta-clustering is applied. This allows identifying groups of cells which correspond with the traditional cell types, without a huge time investment like most other algorithms need. This is confirmed in an independent review by Weber and Robinson, published in Cytometry Part A, December 2016.*

**Sofie Van Gassen, Britt Callebaut, Mary J. Van Helden, Bart N. Lambrecht, Piet Demeester, Tom Dhaene and Yvan Saeys.**

**Published in Cytometry Part A, 2015; 87A(7), 636-645.**

**Abstract** The number of markers measured in both flow and mass cytometry keeps increasing steadily. Although this provides a wealth of information, it becomes infeasible to analyze these datasets manually. When using 2D scatter plots, the number of possible plots increases exponentially with the number of markers and therefore, relevant information that is present in the data might be missed. In this article, we introduce a new visualization technique, called FlowSOM, which analyzes flow or mass cytometry data using a Self-Organizing Map. Using a two-level clustering and star charts, our algorithm helps to obtain a clear overview of how all markers are behaving on all cells, and to detect subsets that might be missed otherwise. R code is available at https://github.com/SofieVG/FlowSOM and will be made available at Bioconductor.

## 2.1 Introduction

At the moment, many flow cytometry experiments are performed with seven colors or more. For mass cytometry experiments, this number is even higher. Analyzing these high-dimensional datasets is not always easy, as traditional gating relies on selection of defined cell populations. It is difficult and time-consuming to keep an overview of how markers are behaving for all these defined cell types. In practice, not all combinations of markers are examined and therefore, valuable information can remain unexamined and unnoticed.

A solution to this problem is the use of advanced visualization techniques in which more information is provided than in the traditionally used scatter plots.

Examples of new visualization techniques developed specifically for this purpose are viSNE [1] and SPADE [2]. Whereas viSNE will plot all cells in a transformed twodimensional space, SPADE will cluster cells in many groups and visualize the results in a minimal spanning tree. SPADE is, however, quite slow, especially for larger datasets. For both viSNE and SPADE, many plots need to be investigated to get a correct annotation of cluster boundaries and cell types.

Completely automatic clustering algorithms like flowMeans, SWIFT and others [3–10] are another solution that might be considered. Yet, even when using these algorithms, it is necessary to visualize the results clearly to interpret them correctly. The problems we described before are intrinsic to using scatter plots, so the same problems remain as with traditional gating if these automatic techniques are not combined with new visualization algorithms.

A self-organizing map (SOM) is an unsupervised technique for clustering and dimensionality reduction, in which a discretized representation of the input space is trained. This technique has already been used on flow cytometry data by the FlowKOH algorithm [11]. FlowKOH tries to obtain a one-to-one mapping between the nodes of the SOM-grid and the identified cell types.

In this article, we propose a new method to analyze flow or mass cytometry data using a self-organizing map: FlowSOM. FlowSOM does not only focus on clustering, but is also a visualization aid. Therefore, we use a much larger amount of clusters than the expected number of cell types. FlowSOM can either be used as a starting point for an analysis or it can be used after a manual gating has been performed, just as a way to easily visualize the results. This way, it also gives information about subpopulations that might have been missed in the original manual gating. We obtain results very similar to SPADE, which are calculated one to two orders faster. We also propose a new way of plotting the data using star charts and calculate a meta-clustering of the data, which gives a great starting point for manual annotation of the results. Using pie charts, our algorithm can help in the visualization of both manual gatings and automated clusterings.

In the following sections, we will present our algorithm, explain each step in detail and demonstrate both a flow cytometry and a mass cytometry use case.

## 2.2 Problem Formulation

In this section, we shortly introduce a formal notation for the data we are working with. This will enable us to explain our algorithm clearly in the next section.

Most experiments consist of multiple samples $s_1, ..., s_s$, for example blood samples of several patients. Each sample is processed by the cytometer and hundreds of thousands up to millions of cells can be measured.

In total, we get measurements from $n$ cells $c_1, ..., c_n$, with $d$ measurements per cell $c_i : \{c_{i_1}, ..., c_{i_d}\}$. This results in an $n \times d$ input matrix for our algorithm. When processing the data, we will consider each cell $c_i$ as a point in a $d$-dimensional space.

We assume that each cell belongs to an *a priori* unknown class or cell type $C_1, ..., C_m$. Our goal is to assign the cells to $k$ clusters $K_1, ..., K_k$, in such a way that the clusters correspond to the underlying true cell types.

Typically, we choose $k$ much larger than $m$ for visualization opportunities. In this case, the goal is to obtain a good purity for each of the clusters: i.e., all cells in a cluster $K_i$ should have the same true cell type. To measure how good a certain result is, we use the purity measure. For each cluster, the percentages of cell types that are present are computed and the maximum is determined. The weighted average of these maxima over all clusters is the final purity value.

As our algorithm also provides a meta-clustering step, we can also strive for a good agreement between the metaclustering result and the true cell types. For this purpose, we use the F-measure: the geometric mean of precision and recall. We compute this in the same way as is explained in the FlowCAP I article [11]. For a clustering $K$ and true cell types $C$, the exact definitions of purity and F-measure can be found in Formulas 2.1 and 2.2.

$$\text{Purity}(C, K) = \sum_{j=1}^{k} \frac{K_j}{n} \max_{i=1..m} \left( \frac{K_j \cap C_i}{K_j} \right) \tag{2.1}$$

$$F(C, K) = \sum_{i=1}^{m} \frac{C_i}{n} \max_{j=1..k} \left( \frac{2\text{Pr}(C_i, K_j)\text{Re}(C_i, K_j)}{\text{Pr}(C_i, K_j) + \text{Re}(C_i, K_j)} \right)$$

$$\text{with } \text{Pr}(C_i, K_j) = \frac{|K_j \cap C_i|}{|K_j|} \text{ and } \text{Re}(C_i, K_j) = \frac{|K_j \cap C_i|}{|C_i|} \tag{2.2}$$

## 2.3 FlowSOM Algorithm

The complete workflow of FlowSOM consists of four steps: reading the data, building a self-organizing map, building a minimal spanning tree and computing a meta-clustering result. An overview can be seen in Figure 2.1. In this section, we will explain each step in detail.

### 2.3.1 Reading the Data

The first step of our algorithm is actually a preprocessing step, illustrated in the red part of Figure 2.1. Although it is possible to process each file by itself, it is often advantageous to combine several files from an experiment. This way, one model is trained for the whole experiment and each cell type can be represented in it. This enables easy comparisons between several samples in the experiment.

To start, the fcs-files produced by the cytometer are read and, if necessary, steps like compensation and logicle transformation can be performed. Next, if samples $s_1, ..., s_s$ have respectively $n_1, ..., n_s$ cells measured, they are all brought together in one big $n \times d$ matrix, with $n = \sum_{i=1}^{s} n_i$. Once all files are read and bundled together, we also pre-process the data by scaling it: $c_{i_j} = \frac{c_{i_j} - \text{mean}(c_{1_j}, \ldots, c_{n_j})}{\text{stdev}(c_{1_j}, \ldots, c_{n_j})}$. This means each column gets a mean value of 0 and a standard deviation of 1, and ensures that each marker gets the same importance in the further processing of the data. If expert knowledge indicates that one marker should get a higher importance than another one, specific scaling parameters can be set to reflect this in the further course of the algorithm.

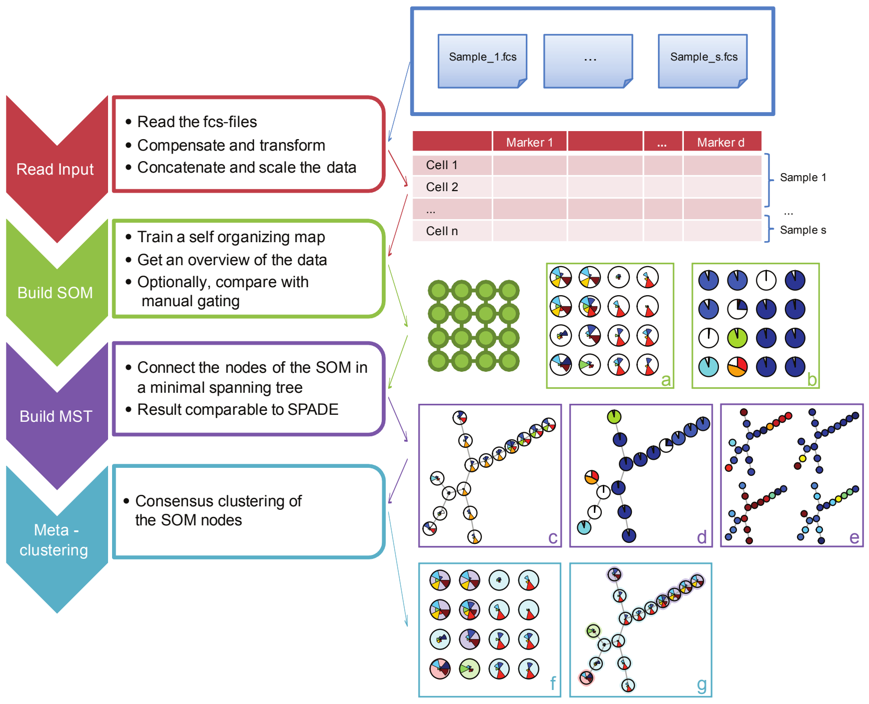

Read Input
• Read the fcs-files
• Compensate and transform
• Concatenate and scale the data
Build SOM
• Train a self organizing map
• Get an overview of the data
• Optionally, compare with manual gating
Build MST
• Connect the nodes of the SOM in a minimal spanning tree
• Result comparable to SPADE
Meta - clustering
• Consensus clustering of the SOM nodes
Sample_1.fcs
...
Sample_s.fcs
Marker 1
...
Marker d
Cell 1
Cell 2
...
Cell n
Sample 1
...
Sample s
a
b
c
d
e
f
g

***Figure 2.1:*** *Overview of the four steps of the FlowSOM algorithm. (i) The fcs-files are read, compensated, transformed, concatenated and scaled, resulting in a matrix with a row for every cell, describing the measured marker values. (ii) A self-organizing map is trained on the matrix. The result is a grid of nodes, corresponding to cell clusters. Visualizations can be made showing the mean marker values of each node in star charts (a) or the concordance with a manual gating in pie charts (b). (iii) A minimal spanning tree is built, on which the same information can be visualized (c,d). We can also show mean marker values for specifically chosen markers, resulting in figures very similar to SPADE trees (e). (iv) A meta-clustering of the nodes is calculated, corresponding to an automatic gating procedure. This is indicated by the background color of the nodes, and can be both visualized in the grid (f) or the minimal spanning tree (g).*

### 2.3.2 Building a Self-Organizing Map

In this section, we will present the self-organizing map algorithm (SOM), as introduced by Kohonen [12]. Our own implementation is strongly based on [13] and is represented by the green section in Figure 2.1.

A SOM is a specific type of artificial neural network, used for clustering. It consists of a grid of nodes, in which each node represents a point in the multidimensional input space. When clustering, a new point is classified with the node that is its nearest neighbor. The grid is trained in such a way that the nodes closely connected to each other resemble each other more than nodes that are only connected through a long path. As such, the grid contains topological information and a single training point can influence multiple nodes.

More formally explained, we have $k$ nodes, each defined as a $d$-dimensional point. We start by initializing the nodes with random points of the dataset. We define a neighborhood function as the Chebyshev distance in the two-dimensional grid of nodes. The self-organizing map is trained by repeatedly picking a point from the dataset, finding the node nearest to it and updating all the nodes in the neighborhood of that node. During the algorithm, the size of the neighborhood $\varepsilon$ and the learning factor $\alpha$ are decreased. In the end, after iterating over $rlen * n$ points, all $n$ points of the dataset are assigned to the node that resembles it the best, resulting in the final clustering.

***algorithm 1:** Training a self-organizing map*

```
procedure SOM
 1: $nodes \leftarrow k$ randomly chosen points of the dataset
 2: for $run = 1$ to $rlen * n$ do
 3:     $c_i \leftarrow$ randomly chosen cell
 4:     $nn \leftarrow$ nearestNode($nodes$,$c_i$)
 5:     for $j = 1$ to length($nodes$) do
 6:         if distance($nodes[j]$,$nn$) $< \varepsilon$ then
 7:             $nodes[j] \leftarrow nodes[j] + \alpha(c_i - nodes[j])$
 8:     $\varepsilon \leftarrow \varepsilon - \varepsilon$_change
 9:     $\alpha \leftarrow \alpha - \alpha$_change
10: return $nodes$
```

### 2.3.3 Building a Minimal Spanning Tree

The resulting clustering of the SOM can be visualized in a minimal spanning tree (MST) [14, 15], as shown in the purple part of Figure 2.1. This technique is also used for SPADE visualizations. An MST connects the nodes of a graph in such a way that the sum of the weights of the branches is minimal. By doing this, nodes will get connected to the ones they are the most similar to, taking the multidimensional topology of the data in account. The result is a connected acyclic graph, which can be visualized using the algorithm proposed by Kamada and Kawai [16].

### 2.3.4 Meta-Clustering

SOMs can be used to get an immediate clustering, where the number of nodes is set to the expected number of cell types. However, for visualization purposes, it is advantageous to include more nodes than the expected number of clusters. By doing this, cells that are in between cell types can also get a place in the grid and smaller changes in the cell types can be noticed.

To help the user get a final clustering of the data, we cluster the node centers in a next step. The meta-clustering is indicated by the background color of the nodes in the blue part of Figure 2.1. To choose the number of meta-clusters, one can either use prior knowledge about the number of expected cell types, or one can use the so called elbow-criterion.

To use this criterion, several values for k are tried out and for each clustering the variance in the clusters is calculated. If the number of clusters is very low, the variance will be high. The variance will decrease strongly if the number of clusters is increased. If the number of clusters is correct, the variance will be relatively low. If the number of clusters is increased further, the variance will still decrease, but much more slowly, the extra clusters only making a minimal difference. The goal of the elbow criterion is to detect the point where the variance stops decreasing sharply, and only decreases slowly from that point on. This point can be found by fitting two linear regression lines on the measured variances. The break point will be the point with the minimal residual error.

The clustering method we use is consensus hierarchical clustering, as implemented in the ConsensusClusterPlus R package [17]. This method works by subsampling the points several times, and making a hierarchical clustering for each subsampling. Based on how often the same points are clustered together or not, a final clustering is made. By testing the stability of the clustering, this method gets better results than applying the basic hierarchical clustering algorithm.

## 2.4 Results and Discussion

In this section, we present several results which were obtained with our algorithm. To evaluate our algorithm, we compared the results of FlowSOM with the traditional gating results of researchers. Evaluating our algorithm by comparing with traditional gating is not optimal, because the manual analysis of cytometry data can be quite subjective: different researchers will get different results and they often gate only the cell types that are of interest to them. However, we aim for a visualization method that will give researchers a clear overview of their data,

***Table 2.1:*** *Overview of datasets used in the result section. The number of events indicates the number of events measured by the cytometer. The number of cells indicates the number of cells used for further analysis after a lymphocyte gate for the BAL staining, and a single cell gate for the bone marrow dataset.*

| DATASET | MARKERS | FILES | NUMBER OF EVENTS | NUMBER OF CELLS |
|---|---|---|---|---|
| FlowCAP I | | | | |
| Diffuse large B cell lymphoma | 3 | 30 | - | 308,676 |
| Graft versus host disease | 4 | 12 | - | 207,171 |
| Hematopoietic stem cell transplant | 4 | 30 | - | 278,005 |
| Normal donor | 10 | 30 | - | 1,778,883 |
| Symptomatic West Nile virus | 6 | 13 | - | 1,214,373 |
| BAL staining | 8 | 12 | 3,635,910 | 411,143 |
| Bone marrow | 31 | 3 | 1,264,755 | 660,084 |

and the least they will expect is an acceptable correspondence with their manual results.

## 2.4.1 Overview of the Datasets

We tested our algorithm on several real cytometry datasets. An overview of their properties can be found in Table 2.1. First, we optimized the parameter choices we made for our algorithm. For this purpose, we used the benchmark datasets from the FlowCAP I challenge. These consist of five flow cytometry datasets, on which several state-of-the-art algorithms have been tested. More information about these datasets can be found in [11].

Second, we applied our algorithm on an in-house dataset in which the bronchoalveolar lavage (BAL) of 12 mice were stained for 8 different markers. These 12 samples come from 6 wild type mice and 6 CXCR6-eGFP mice. The goal of the experiment was to study the influence of CXCR6 on house dust mite induced asthma.

Finally, we also wanted to show the results of our algorithm on a mass cytometry dataset. Therefore, we used a previously described human bone marrow dataset [18]. Mass cytometers can measure even more markers simultaneously than the current state-of-the-art flow cytometers, and this number will only increase.

## 2.4.2 Choosing the Optimal Parameters

The application of FlowSOM requires the selection of several parameter settings. We used a rectangular nontoroidal grid. In a toroidal setting, the clusters can wrap around the borders, which makes it harder to distinguish them visually. We used a learning rate $\alpha$ starting from 0.05 decreasing to 0.01 and a neighborhood size $\varepsilon$ starting from the 67% quantile of all neighborhood distances decreasing to 0.

In addition, we optimized several other settings using the FlowCAP I benchmark data files.

First, we varied the distance measure to find the nearest neighbor of a new point. We tried Manhattan distance, Euclidean distance and Chebyshev distance. For all five datasets, Euclidean distance gave the best results. This corresponded to the manual way of gating, in which Euclidean distance is also intrinsically used on 2D scatter plots.

We also evaluated various grid sizes. Obviously, when comparing a $5 \times 5$, $10 \times 10$, and $15 \times 15$ grid, a higher number of nodes corresponds to a higher purity. To determine the optimal grid size, it might be necessary to visually inspect the results. Too many nodes will give a better purity, but also a more cluttered view. In our experience the shape of the grid  square ($10 \times 10$) or a long rectangle ($4 \times 25$) did not seem to have much influence on the quality of the results. By default, we use a $10 \times 10$ grid.

We varied the number of times the training set is presented from 1 to 200 times. Remarkably, it was not advantageous to use many repetitions. This can be explained by the fact that these datasets already contain a relatively high number of cells, in which redundancy is already present. Using 10 repetitions gave us good results, and this did not improve much with more repetitions.

We also compared several clustering algorithms for the meta-clustering. We tested another SOM, k-means, hierarchical clustering and consensus hierarchical clustering. Consensus hierarchical clustering clearly gave the best results, while hierarchical clustering was the runner-up. Both k-means or a SOM with a small number of nodes did not perform very well.

### 2.4.3 Use Case 1: A BAL Staining

To evaluate the performance of FlowSOM, we used a flow cytometry dataset with seven surface markers (CD19, CD3, TCR$\gamma\delta$, TCR$\beta$, CD4, CD8, and NK1.1) and GFP (transgenic mice; knockout) or not (WT mice).

Before presenting this dataset to FlowSOM, a quite strict lymphocyte gate was manually set on the FSC-SSC scatter plot. We took only the cells that were inside the lymphocyte gate along for further analysis by FlowSOM. The data was compensated, transformed with a logicle transformation and scaled. We used the seven surface markers in our analysis.

We applied FlowSOM to this dataset with all the default parameter settings. Figure 2.2 shows the result of this analysis in a SOM grid, where each node is represented by a star chart. These star charts indicate the mean intensities of the markers for all cells in the dataset assigned to that node.

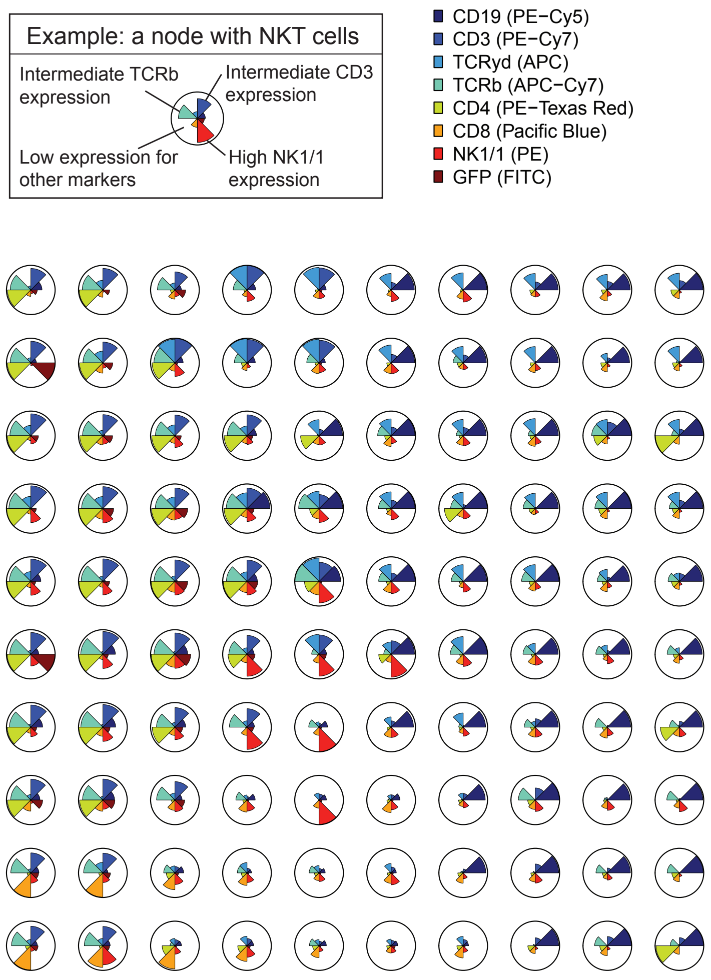


***Figure 2.2:*** *SOM result for the BAL staining. The mean marker values are visualized for each node, using star charts. The height of each part indicates the intensity: if the part reaches the border of the circle, the cells have a high expression for that marker. By looking at the specific colors (starting from the right, going counter-clock-wise) information about the cells represented by each node can be extracted. Differences between the stars can be quite easily distinguished, and give an indication about which 2D plots might be interesting to examine in more detail.*

To analyze the performance of FlowSOM, we subsequently compared these results with a classical manual gating strategy. From the dataset, we manually gated 6 different populations of cells: B cells, CD4+ T cells, CD8+ T cells, $\gamma\delta$T cells, NK cells, and NKT cells. These results are depicted by pie charts, indicating the presence of the defined cell types in each node (Fig. 2.3A). We obtain a weighted mean purity of 0.95.

All B cells are assigned to the right side of the grid and all T cells to the left side of the grid. Using the star charts, it is easy to distinguish that all nodes on the right side of the grid are quite similar and high in CD19. On the right side of the grid, we see that all nodes are high in CD3 and TCR$\beta$, but it is clear that the four nodes at the bottom are different from the others: they are high in CD8, whereas the others are high in CD4.

In Figure 2.3B, we demonstrate some other visualization opportunities of the SOM grid. We split the dataset in two groups: the wild type mice and the knockout mice. The size of the nodes indicates the percentage of cells represented. The color of the nodes indicates the GFP intensity. By comparing the two groups, the results show that the KO mice lacked most $\gamma\delta$ T cells and had a higher percentage of B cells. In addition, our results clearly indicate that the $\alpha\beta$ T cells of KO mice express high levels of GFP.

In some cases, a tree view can give a clearer overview of the data than the grid structure and we therefore also applied such a visualization to the manual gated data (Fig. 2.3A) and to a meta-clustering of the SOM nodes with 8 clusters (Fig. 2.3C).

The meta-clusters correspond very well to the manual gated cell types and we obtain an F-measure of 0.93. This shows that FlowSOM would have given a great starting point to analyze this data even if the manual analysis was not available. With the manual analysis next to it, we noted that the B cell group is split in two clusters. The small cluster that is separated from the B cells, are B cells with a high CD4 intensity. This way, the FlowSOM visualization can help to detect interesting subsets in the dataset that might be overlooked otherwise.

### 2.4.4 Use Case 2: Human Bone Marrow

When analyzing mass cytometry datasets, the amount of markers to take into account is often overwhelming. We applied the FlowSOM algorithm to a human bone marrow mass cytometry dataset to illustrate the usefulness of our approach.

This dataset was stained with 31 markers, of which 13 were surface markers (CD3, CD4, CD8, CD11b, CD19, CD20, CD33, CD34, CD38, CD45, CD45RA, CD90, and CD123) and 18 intracellular markers (pPLC$\gamma$2, pSTAT5, pERK1/2, Ki67, pMAPKAPK2, pSHP2, pZAP70/Syk, pSTAT3, pSLP-76, pNFkB, total lkB$\alpha$, pH3, p-p38, pBT/ltk, pS6, pSrcFK, pCrkL, pCREB). For demonstration purposes,

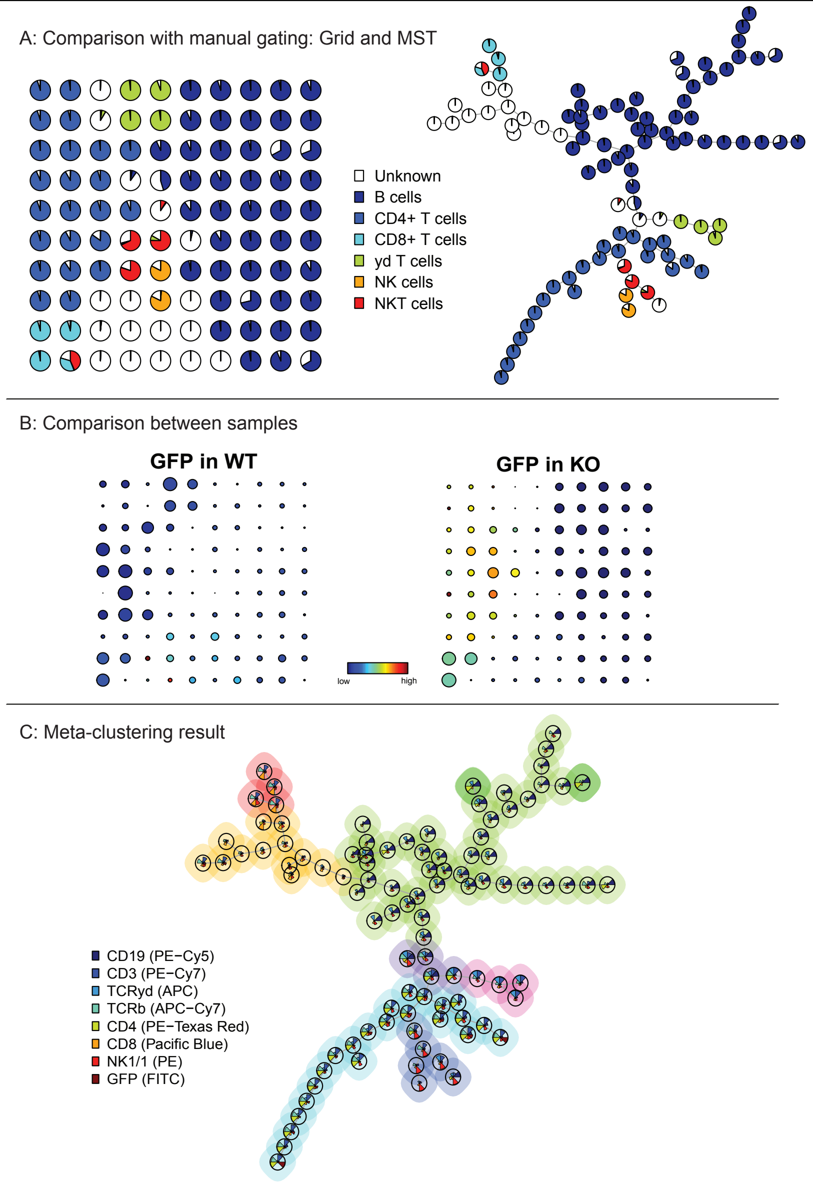


***Figure 2.3:*** *Further visualization options of the FlowSOM algorithm for the BAL staining. (a) Pie charts indicate the percentage of cells represented by the node falling in the original manual gates. This can be both visualized in the grid and the minimal spanning tree. A strong correspondence between the branches of the tree and the manually defined cell types can be seen. (B) The mean marker intensity of the GFP value is shown for two groups of mice. The size of the nodes corresponds with the number of cells represented by the node. Differences in color and size between the two grids can be easily distinguished. (C) An automatic meta-clustering of the FlowSOM nodes is indicated by the background color of the nodes. This corresponds well with the manual gating result from part A.*

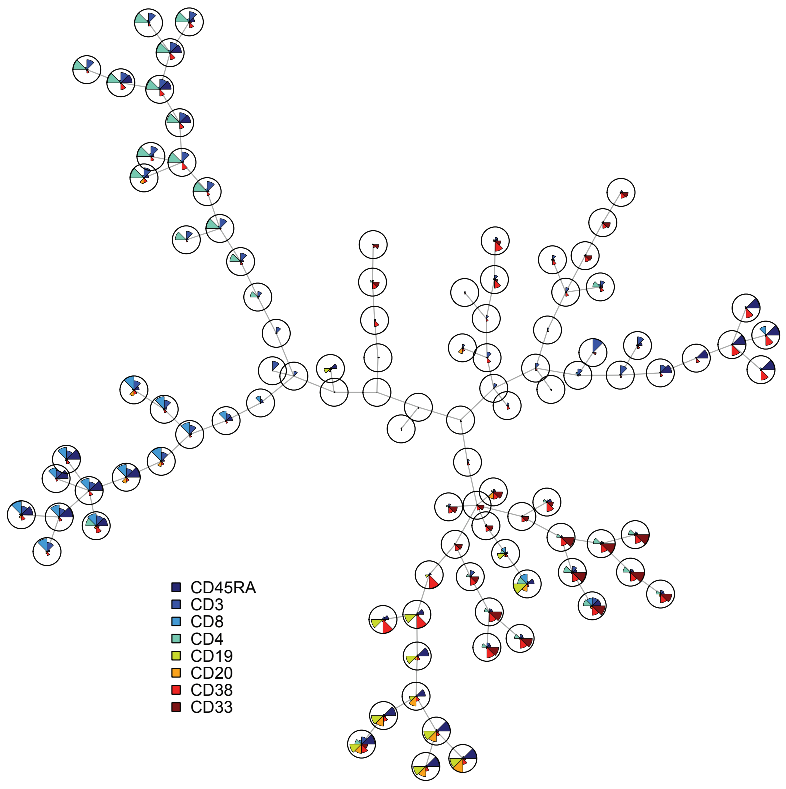


***Figure 2.4:*** *Result for the human bone marrow dataset. Only a selected number of markers are shown in the star charts for clarity. By comparing these markers, we can e.g., identify CD8 and CD4 T cells and B cells.*

we only used three sample files: an unstimulated sample, a sample simulated with IL-3 and a sample stimulated with LPS.

First a singlet gate was manually applied on a scatter plot with DNA content and Cell length. Only the single cells were used in our analysis. The data was transformed with an arcsinh transformation and scaled. We used the 13 surface markers to build the SOM.

To identify naive and memory T and B cells, we can either use the star charts (Fig. 2.4) or compare with the manual gating (Fig. 2.5A). In comparison to SPADE, more information is visible on one star chart figure. We did not have to compare eight separate figures to find the cells that have the right combination of marker intensities.

In Figure 2.5B, we investigate the intracellular signaling status of some mark-

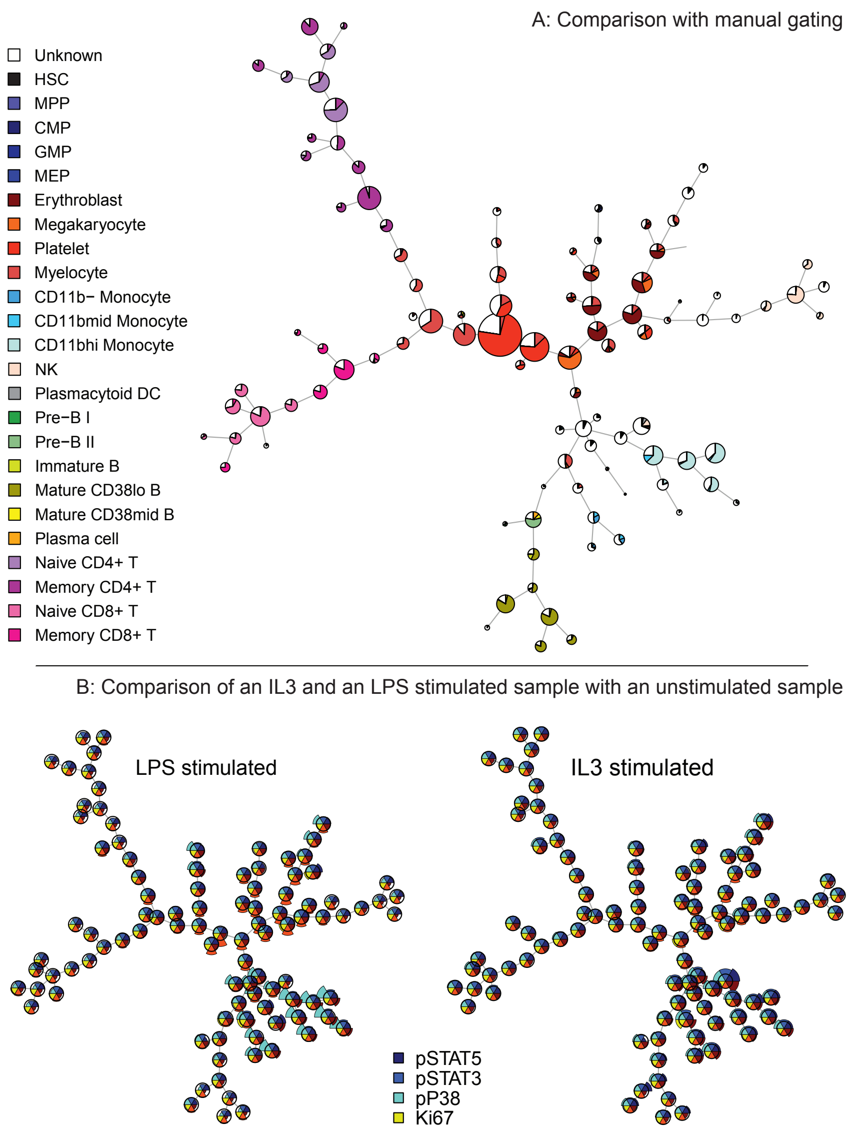


***Figure 2.5:*** *Further visualization options for the human bone marrow dataset. (A) Comparison with manual gating. The size of the nodes indicates the number of cells assigned to each node. (B) Comparison of two stimulated samples with an unstimulated sample. The star charts indicate several intracellular marker values for the different cell types. If the part reaches the circle exactly, the two samples have the same expression. If the part does not reach the circle, there is a lower expression in the simulated sample. If the part goes beyond the circle, the expression is higher in the simulated sample.*

ers. Instead of showing the values themselves, we show the difference of the IL-3 and LPS stimulated samples with the unstimulated sample. If the pieces of the star chart reach the enveloping circle exactly, the stimulated sample has the same level as the unstimulated sample. If the pieces go outside the circle, there is a stronger induction. If the pieces are smaller than the circle, the induction has diminished.

When observing the results after stimulation with LPS, the monocytes have an induced phosphorylation of P38 MAPK (cyan in the figure) and the megakaryocytes an induced phosphorylation of Btk/ltk (orange in the figure).

The node containing 98% of the plasmacytoid DCs had a strong induction of several intracellular markers after stimulation with IL3. This is the cell type indicated in the original article to strongly increase expression of pSTAT5 and pSTAT3 on IL3 stimulation. In comparison, B cells only have an induced phosphorylation of STAT5 and not of STAT3.

### 2.4.5 Comparison with SPADE

The FlowSOM minimal spanning tree result is very similar to a SPADE result. The key difference between FlowSOM and SPADE is the clustering algorithm. A self-organizing map, the clustering algorithm used by FlowSOM, works very differently from hierarchical clustering, as proposed in the SPADE article. More specifically, it does not tend to make each cluster approximately the same size. This way, rare cell types can still be detected without the need for any density-based subsampling. Furthermore, by removing the subsampling stage, a significant time improvement can be achieved.

We ran some tests on the human bone marrow mass cytometry dataset, in which several cell types are present as $<1\%$ of the data. These tests indicated that with the same number of nodes, results will be very similar between FlowSOM (without subsampling) and SPADE (with subsampling). Unfortunately, both algorithms use stochastic processes, so the results may vary slightly per run. By using many more clusters than the expected number of cell types, both algorithms are able to assign nodes to the rare cell types. However, the number of nodes has an important role: it might be necessary to increase the number of nodes to 200 or 300 to find these rare cell types. It can also be advantageous to do some manual gating steps to focus on the cell types of interest and remove debris and doublets.

In the BAL dataset, which is pre-gated on lymphocytes, both NK and NKT cells are present only in 1% of the data. Both cell types are assigned to multiple nodes in the SOM grid (Fig. 2.3A). For the bone marrow dataset, when using 100 nodes, NK cells (2.8%) are assigned to multiple nodes, while Pre-B II cells (0.9%) are represented by one node (Fig. 2.5A). However, when increasing the number of nodes to 200, we also get pure nodes for plasma cells (0.2%) and mature CD38 mid B cells (0.4%), even though we do not manage to catch the CD11bmid mono-

***Table 2.2:*** *Results on the benchmark datafiles from the FlowCAP I challenge*

| DATASET | SPADE PURITY | FLOWSOM PURITY | FLOWSOM F-MEASURE |
|---|---|---|---|
| Diffuse large B cell lymphoma | 0.9367 | 0.9494 | 0.8370 |
| Graft versus host disease | 0.9335 | 0.9439 | 0.8546 |
| Hematopoietic stem cell transplant | 0.9787 | 0.9759 | 0.7449 |
| Normal donor | 0.8173 | 0.8228 | 0.6775 |
| Symptomatic West Nile virus | 0.9102 | 0.9234 | 0.8292 |

cytes (0.8%) separately. This indicates that not only the percentage of the rare cell type is important, but also how strongly it differs from other cell types. The SPADE algorithm with 200 nodes also did not assign a pure node to the CD11bmid monocytes.

#### 2.4.5.1 Quality

To evaluate the quality of our algorithm with the default parameter settings, we used the FlowCAP I benchmark files.

First, we evaluated the purity of the nodes, as defined in Formula 2.1 and compared this with the purity of SPADE nodes. We calculated the mean purity of all samples for five runs of the algorithms. As shown in Table 2.2, we obtain a slightly better purity than SPADE. Second, we evaluated the results of the meta-clustering with the F-measure defined in Formula 2.2. The number of meta-clusters was determined automatically by the elbow method. These results are in line with the results obtained in the FlowCAP I challenge, which indicates that the metaclustering is a good starting point for further analysis.

#### 2.4.5.2 Running Time and Memory Usage

All measurements for running time and memory usage are made on a cluster with eight computing nodes, each having 24 cores and 256 GB RAM. We first evaluate the running time of our algorithm in respect to the number of cells in the dataset. An overview is shown in Figure 2.6. Reading the data and training the SOM will take longer if more data is available. However, if many cells are available, it might be possible to decrease the number of times the training set is presented to the algorithm. On the other hand, if time constraints are not a problem, the number of times the dataset is presented in training can be increased for better results. For building the MST and meta-clustering the data, only the grid nodes are used, so the running time of these steps is not influenced by the amount of data we start with. We observe that if we double the amount of data, the computation will need

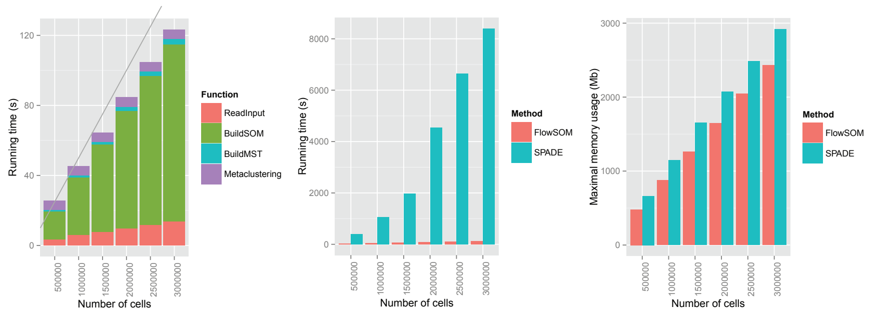


***Figure 2.6:*** *(A) Running time of each step of the flowSOM algorithm, for a varying number of cells. (B,C) Comparison of running time and memory usage with SPADE, for a varying number of cells.*

a bit less than double the amount of time. Because our algorithm does not need to subsample the data, we can obtain a huge time gain in comparison to SPADE. This can be seen in Figure 2.6B. Our algorithm is not parallelized at the moment and runs on a single core. We compared with the SPADE algorithm which is optimized to run in parallel and used eight cores. Still, we are ten to fifty times faster. We use about the same amount of memory as SPADE (Fig. 2.6C). For both FlowSOM and SPADE, running time and memory usage are not strongly influenced by varying the number of markers in the analysis.

## 2.5 Conclusion

FlowSOM offers new ways to visualize and analyze cytometry data. The algorithm consists of four steps: reading the data, building a self-organizing map, building a minimal spanning tree and computing a meta-clustering. We proposed several visualization options: star charts to inspect several markers, pie charts to compare with manual gating results, variable node sizes dependent on the amount of cells assigned to the node and a grid or a tree structure which both give topological information.

The purity of the FlowSOM nodes and the F-measures of the meta-clustering indicate a strong correspondence with manual gating results.

The FlowSOM algorithm has a purity comparable to the SPADE algorithm, but results can be obtained up to 50 times faster.

Traditional gating uses a hierarchy of gates. As a consequence, many cells are gated out: they are not examined any further. Using a visualization method in which all cells are represented minimizes the risk of missing interesting cell populations. By using star charts, several markers can be examined at once and annotation becomes easier.

In the future, we will investigate more optimizations to the FlowSOM algorithm. Many variants of the traditional SOM algorithm exist. One possibility is to define the neighborhood function by using a minimal spanning tree instead of a grid. This might enable the algorithm to better separate differentiated cell types and is in accordance with the minimal spanning tree visualization we use afterwards.

At the moment, the algorithm is not optimized to run in parallel. This is also a possible improvement that can be added to FlowSOM.

## Acknowledgment

S.V.G. is funded by the Flanders Agency for Innovation by Science and Technology (IWT). This research was conducted in the context of the Ghent University Multidisciplinary Research Partnership Bioinformatics.

## References

[1] E.-a. D. Amir, K. L. Davis, M. D. Tadmor, E. F. Simonds, J. H. Levine, S. C. Bendall, D. K. Shenfeld, S. Krishnaswamy, G. P. Nolan, and D. Pe'er. *viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia.* Nature biotechnology, 31(6):545–52, 2013.

[2] 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):886–91, 2011.

[3] K. Lo, R. R. Brinkman, and R. Gottardo. *Automated gating of flow cytometry data via robust model-based clustering*. In Cytometry Part A, volume 73, pages 321–332, 2008.

[4] S. Pyne, X. Hu, K. Wang, E. Rossin, T.-I. T.-I. Lin, L. M. Maier, C. Baecher-Allan, G. J. McLachlan, P. Tamayo, D. A. Hafler, P. L. De Jager, and J. P. Mesirov. *Automated high-dimensional flow cytometric data analysis.* Proceedings of the National Academy of Sciences of the United States of America, 106(21):8519–8524, 2009.

[5] K. Lo, F. Hahne, R. R. Brinkman, and R. Gottardo. *flowClust: a Bioconductor package for automated gating of flow cytometry data.* BMC bioinformatics, 10:145, 2009.

[6] G. Finak, A. Bashashati, R. Brinkman, and R. Gottardo. *Merging mixture components for cell population identification in flow cytometry.* Advances in bioinformatics, 2009:247646, 2009.

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

[8] I. Naim, S. Datta, J. Rebhahn, J. S. Cavenaugh, T. R. Mosmann, and G. Sharma. *SWIFTscalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, Part 1: Algorithm design*. Cytometry Part A, 85(5):408–421, 2014.

[9] A. Cron, C. Gouttefangeas, J. Frelinger, L. Lin, S. K. Singh, C. M. Britten, M. J. P. Welters, S. H. van der Burg, M. West, and C. Chan. *Hierarchical Modeling for Rare Event Detection and Cell Subset Alignment across Flow Cytometry Samples*. PLoS Computational Biology, 9(7), 2013.

[10] S. Pyne, S. X. Lee, K. Wang, J. Irish, P. Tamayo, M. D. Nazaire, T. Duong, S. K. Ng, D. Hafler, R. Levy, G. P. Nolan, J. Mesirov, and G. J. McLachlan. *Joint modeling and registration of cell populations in cohorts of high-dimensional flow cytometric data*. PLoS ONE, 9(7), 2014.

[11] N. Aghaeepour, G. Finak, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann. *Critical assessment of automated flow cytometry data analysis techniques.* Nature methods, 10(3):228–38, 2013.

[12] T. Kohonen. *The Self-Organizing Map*. Proceedings of the IEEE, 78(9):1464–1480, 1990.

[13] R. Wehrens and L. M. C. Buydens. *Self- and super-organizing maps in R: The kohonen package*. Journal of Statistical Software, 21(5):1–19, 2007.

[14] V. K. M. Whitney. *Algorithm 422: minimal spanning tree [H].* Communications of the {ACM}, 15(4):273–274, 1972.

[15] G. Csardi and T. Nepusz. *The igraph software package for complex network research*. InterJournal Complex Systems, page 1695, 2006.

[16] T. Kamada and S. Kawai. *An algorithm for drawing general undirected graphs*. Information Processing Letters, 31(1):7–15, 1989.

[17] M. D. Wilkerson and D. N. Hayes. *ConsensusClusterPlus: A class discovery tool with confidence assessments and item tracking*. Bioinformatics, 26(12):1572–1573, 2010.

[18] S. C. Bendall, E. F. Simonds, P. Qiu, E.-a. 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, S. D. Tanner, and G. P. Nolan. *Single-Cell Mass Cytometry of Differential Immune and Drug Responses Across a Human Hematopoietic Continuum*. Science, 332(May):687–695, 2011.

# 3

# Applications of the FlowSOM algorithm

*The FlowSOM algorithm is provided as an R package, which is easy to use by bioinformaticians but not straightforward for people in the lab. Once the FlowSOM algorithm was developed, we therefore started collaborations with multiple researchers to apply the algorithm to their data. Such collaborations were also very fruitful to us, to gain further insight in the problems posed and to improve our algorithm for easier use. Based on this feedback, we added three additional functionalities to the R package. First we explored multiple alternative layout functions in addition to the minimal spanning tree (MST). Although several seemed to have potential, none clearly stood out as the perfect replacement for the MST. Next, we added a statistical test to compare groups of samples. By visualizing which nodes increase or decrease between samples, we could ensure that no important changes are overlooked when interpreting the tree. Finally, we also added an automated annotation of the FlowSOM nodes with cell type labels. This allows the researcher to go without a manual gating and immediately start exploring the FlowSOM tree. The approaches described in this chapter have not been published as papers, but were presented as poster presentations at the CYTO conferences 2015 and 2016.*

## 3.1 Implementation in R

The original FlowSOM algorithm and the extensions described in this chapter are bundled into an R package, which is available on the github website of our group (`https://github.com/saeyslab/FlowSOM`). Additionaly, it is also available on Bioconductor, a platform collecting open source R packages for bioinformatics (`https://www.bioconductor.org/packages`). This allows easy discovery and installation by bioinformaticians. All functions are well documented and a *vignette* is available, which guides a new user through the typical workflow when using this package (see Appendix A).

## 3.2 Exploring Different Layouts

A self-organizing map is a clustering algorithm which is often used for visualization, because it results in clusters positioned in a grid. Clusters which are closer together in the grid typically contain more similar datapoints. However, when analyzing flow cytometry data, most cell types are spread over multiple nodes and it takes some time to identify which neighboring clusters are really similar and which ones are actually borders between the different cell types. This might be hard to distinguish in a grid.

Therefore, in the previous chapter, we also introduced the Minimal Spanning Tree (MST) visualization. Inspired by the SPADE algorithm, this visualization has an easier interpretation, as most branches correspond to the separate cell types. However, a minimal spanning tree also has some limitations. For example, when a loop exists in the data, it will be cut at some random point, which might result in similar clusters being on different sides of the tree. The tree should be interpreted as 'connected clusters are certainly similar, but disconnected clusters are not necessarily dissimilar'. This does not correspond with the intuition of the researchers analyzing the trees.

To solve this problem, we experimented with some alternative layout options, visualized in Figures 3.1 and 3.2. The six layout functions described below are all applied on the same FlowSOM grid and colored by the manual gating results. About half of the nodes represent B cells, while some other cell types, such as the NK cells and macrophages are only represented by one node. A good layout should enable us to identify seperate cell populations when the coloring of the nodes would not be available. However, it is important to keep in mind that the manual gating of the nodes is not perfect either, e.g. there exists a node with CD3+,CD19+ cells, which should be classified as 'Unknown' because CD3 is a T cell marker while CD19 is a B cell marker and these are not expected to occur together. However, it is annotated as a B cell node in this dataset.

The first alternative we tried is the t-SNE algorithm. This is a dimensionality reduction algorithm which is also sometimes applied on flow cytometry data (e.g. the viSNE algorithm), but which is rather slow for large datasets. However, we only need to apply it on the cluster centers, commonly 49 (for a 7x7 grid) or 100 (for a 10x10 grid), which computes reasonably fast. Because the t-SNE algorithm uses a non-symmetric distance mapping, it focuses on local similarities, typically resulting in clearly separated clusters. However, it is not always straightforward to define what is 'local', given as the *perplexity* parameter of the algorithm, and some manual fine tuning is often necessary to get a clear representation of the FlowSOM nodes.

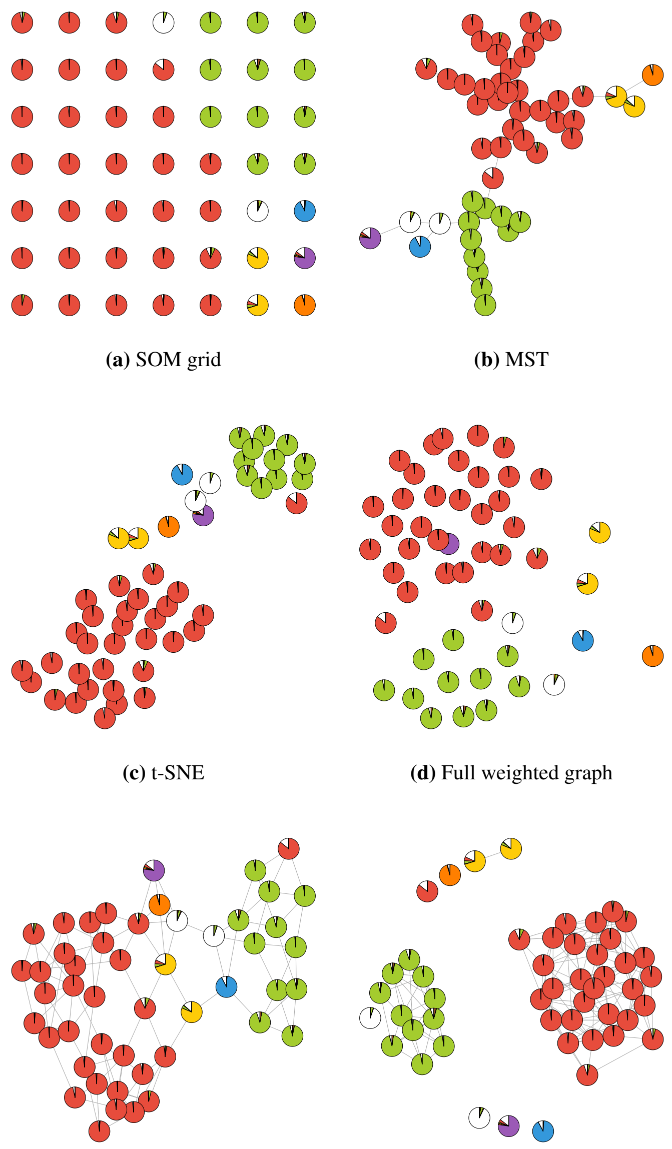
The other alternatives we explored are graph-based, because these structures can contain loops, avoiding the over-simplification that might happen in the MST. One option is using a fully connected graph, where the edges are weighted by the euclidean distance between the node centers. Plotting all edges would result in a hairball graph, but using a force-based layout algorithm like Kamada-Kawai can result in a 2D mapping similar to the t-SNE result, while taking the global structure into account. An alternative option is plotting a nearest-neighbor graph, although it is difficult to pick an optimal number of neighbours. Some clusters might represent a cell type on their own, while others are strongly connected with other clusters, resulting in strange artifacts when using a fixed number of nearest neighbors. An alternative option is to use a cut-off, showing only edges for which the distance between the nodes is lower than a fixed threshold. However, defining this threshold is again not trivial.

As all these alternative options have parameter tuning issues, we stick to the MST visualization as the default option for now, although they are interesting to explore in specific cases where the limitations of the MST are an issue.

## 3.3 Comparing Groups

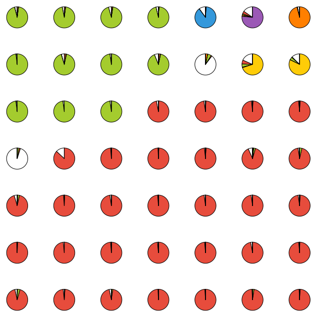
In the previous chapter, we focussed on giving a comprehensive overview of a dataset. While it is indeed important to be aware of all the different cell types present in the data, in most experiment settings the researcher is especially interested in a comparison between different samples.

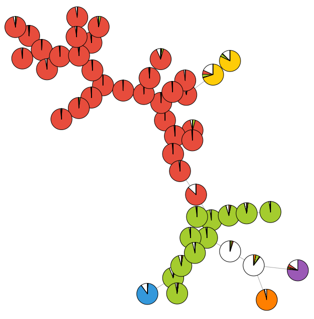
To enable this, we adapted our code in such a way that the background color of the nodes, previously used to show the metaclustering result, could represent any kind of variable with a value for each node. This way, we can define one group of samples as the baseline setting and color nodes which are over- or underrepresented in the other group. Typically, the overrepresented ones are shown in red, while the underrepresented ones are blue. To determine which nodes are significantly different between groups of samples, we made use of a wilcoxon rank-sum test. This statistical test indicates whether two sets of samples are likely to be taken from the same distribution or not, without assuming a normal distribution.

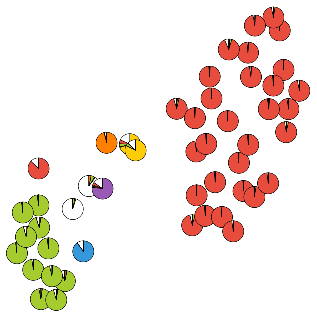


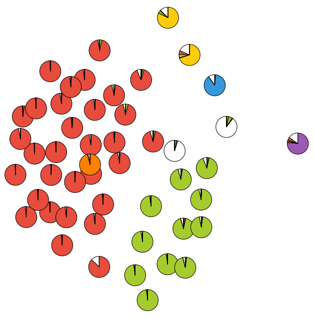
**(a)** SOM grid **(b)** MST **(c)** t-SNE **(d)** Full weighted graph **(e)** KNN graph **(f)** Thresholded graph

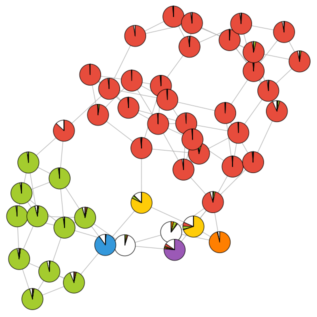
***Figure 3.1:*** *Different layout options for the FlowSOM nodes. The nodes are colored by manual mapping, red: B cells, green: T cells, blue: NK cells, orange: macrophages, yellow: dendritic cells, purple: neutrophils. For the t-SNE algorithm, the perplexity was set to 10. The kNN graph uses 3 neighbors and the cutoff on the weights was set to 2.3. All graphs are mapped to 2D using the Kamada-Kawai layout algorithm.*


**(a)** SOM grid


**(b)** MST


**(c)** t-SNE


**(d)** Full weighted graph


**(e)** KNN graph

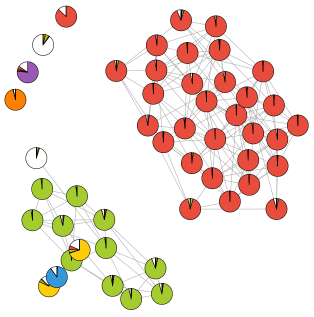

**(f)** Thresholded graph

***Figure 3.2:*** *Another run of the FlowSOM algorithm and all visualization options, using the same parameters. Although the actual layout differs due to the stochasticity of the algorithms, the same populations can be distinguished.*

We color the nodes if they have a p-value lower than 0.05.

We present two case studies in which we used this approach. The first experiment explored an asthma protocol for mice. Allergic asthma is characterized by eosinophilic airway inflammation, goblet cell metaplasia and bronchial hyperreactivity. By sensitizing and challenging mice with allergens such as house dust mites (HDMs), the mice develop all of these canonical asthma features. When using phosphate-buffered saline (PBS) as a negative control in the sensitization phase, the mice do not develop these features. We compared bronchoalveolar lavage samples (BAL) from 12 wild type C57BL/6 mice of which 6 were sensitized with HDM, while the other half were sensitized with PBS. The samples were stained with a general panel which allows the identification of the main immune cell types of interest.

The FlowSOM tree is trained on an aggregated fcs file from all mice together and shown in Figure 3.3. A mapping of the manual gating is shown on the right. The comparison between the groups is shown in Figure 3.4. A tree is drawn for each group, in which the node size corresponds with the average percentage of cells assigned to that node. Instead of comparing the sizes of all these nodes visually, the background color in the second tree indicates all significant increases and decreases in the HDM sensitized mice. This ensures that the changes are representative for all the individual samples, instead of on the averages per group. The eosinophilic inflammation is clearly noticable. Additionally, these mice also show a light increase in B-cells, while all other cell types decrease relatively. This overview enables the researcher to confirm that no unsuspected changes are happening in the other cell types due to the treatment. For a BAL sample, an absolute cell count might also offer an interesting perspective next to percentages. These values can be computed by multiplying the percentages per sample by the number of live cells counted for each of these samples and might tell a slightly different story. For example, in this case, all cell counts increase in the samples treated with HDM, even though the ratios differ strongly. An additional colouring, with the opacity of the colour corresponding to the magnitude of the change can be useful in some cases.

The second experiment we present focused on the importance of interferon regulatory factor-8 (IRF8) in conventional dendritic cells (cDCs) type 1 and 2 [1]. To address the role of this gene, wildtype mice, heterozygous knock-out mice and homozygous knock-out mice were studied. Splenocytes of 15 mice were stained with a panel focused on myeloid cell identification.

Similar to the previous experiment, a FlowSOM tree was trained on an aggregate of all these samples, presented in Figure 3.5. While the manually gated cell types map nicely to separate branches, the FlowSOM tree also indicates that many cells are not manually labeled (white pie-pieces in the tree on the right). In Figure 3.6 the differences between the three groups of mice are visualized. Although

***Table 3.1:*** *Cell type definitions for a lymphocyte staining on splenocytes.*

| | NK1.1 | CD3 | TCR$\gamma\delta$ | TCRb | CD4 | CD8 | CD19 |
|---|---|---|---|---|---|---|---|
| NK cells | + | - | | | | | |
| NKT cells | + | | | + | | | |
| $\gamma\delta$ T cells | - | + | + | - | | | |
| CD4 T cells | - | + | - | + | + | - | |
| CD8 T cells | - | + | - | + | - | + | |
| B cells | | | | | | | + |

there is a strong neutrophilic increase in the homozygous knock-out, the heterozygous knock-out is much more similar to the wildtype phenotype, with only a small increase in neutrophils and a decrease in cDC1s, the cell type of interest in this experiment. This overview allowed the researcher to estimate if any other changes where happening, even the cells that were not manually gated. This is the case in the homozygous knock-out: while the manual gating only notices a strong increase in neuthrophils, the FlowSOM tree shows that there is a whole group of cells which are never labeled in the manual gating but which also increase. This can be a starting point for further data exploration. In this case, these cells might be neutrophil precursors, giving more insight in the effects of the knock-out gene. In the heterozygous mice, indeed only the cDC1s were strongly decreasing, with limited changes to the rest of the immune system. Whereas in the homozygous model so many things are changing that it is hard to make any conclusions from phenotypic observations, the heterozygous model is well suited to investigate the importance of IRF8 for cDC1s.

## 3.4 Labeling Populations

More and more clustering and visualization algorithms are proposed to detect the cell types that are present in flow cytometry samples. However, the interpretation of the resulting clusters can be challenging. For example, in the SPADE algorithm, you need to look at each node and compare plots for all the different markers to determine the marker presence on the cells represented by the node and convert this to a cell type. In FlowSOM, we combine information about the different markers in a star-glyph for an easier overview, but you still need to interpret the marker combinations for each node when you do not have a manual gating available.

However, when conducting an experiment, the researcher often has already some cell types in mind, defined by marker properties such as “CD4 T cells are CD3 positive, CD4 positive and CD8 negative”. Some more examples are given in Table 3.1. Instead of letting the researchers look at each node and match it to these patterns, we propose to let the researcher define these cell type definitions in

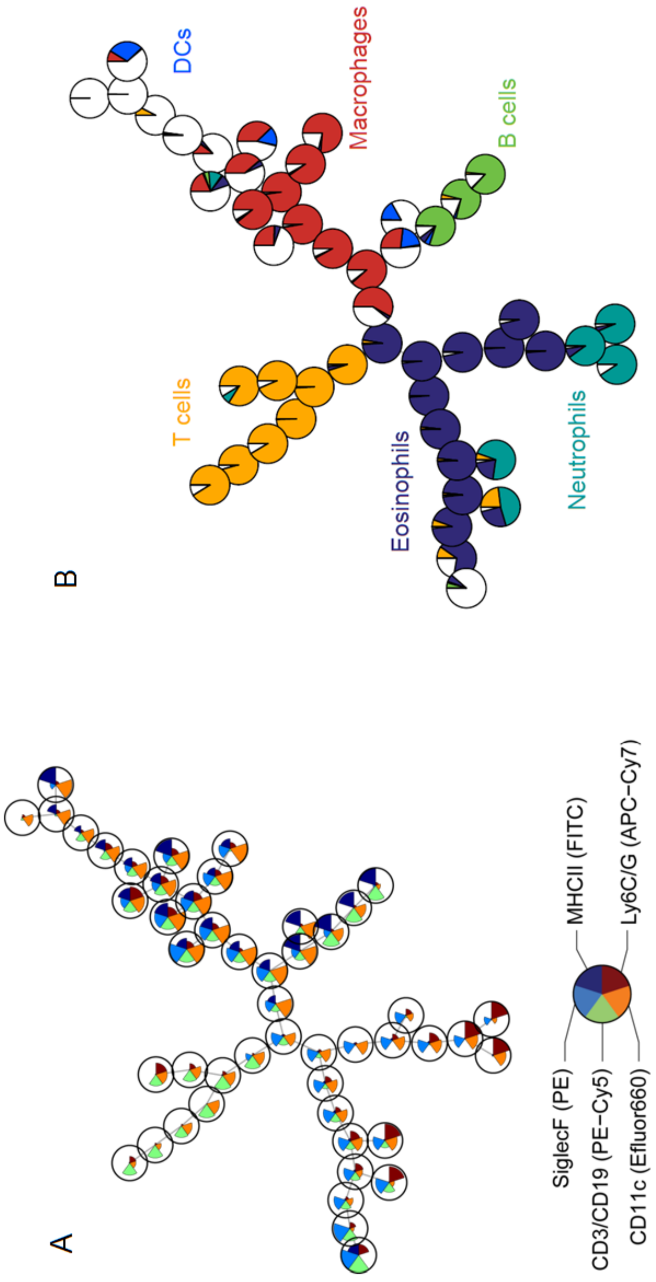


***Figure 3.3:*** *FlowSOM tree trained on fcs files from BAL samples. (A) Overview of the median marker expressions in the clusters. (B) Mapping of the manual gating.*

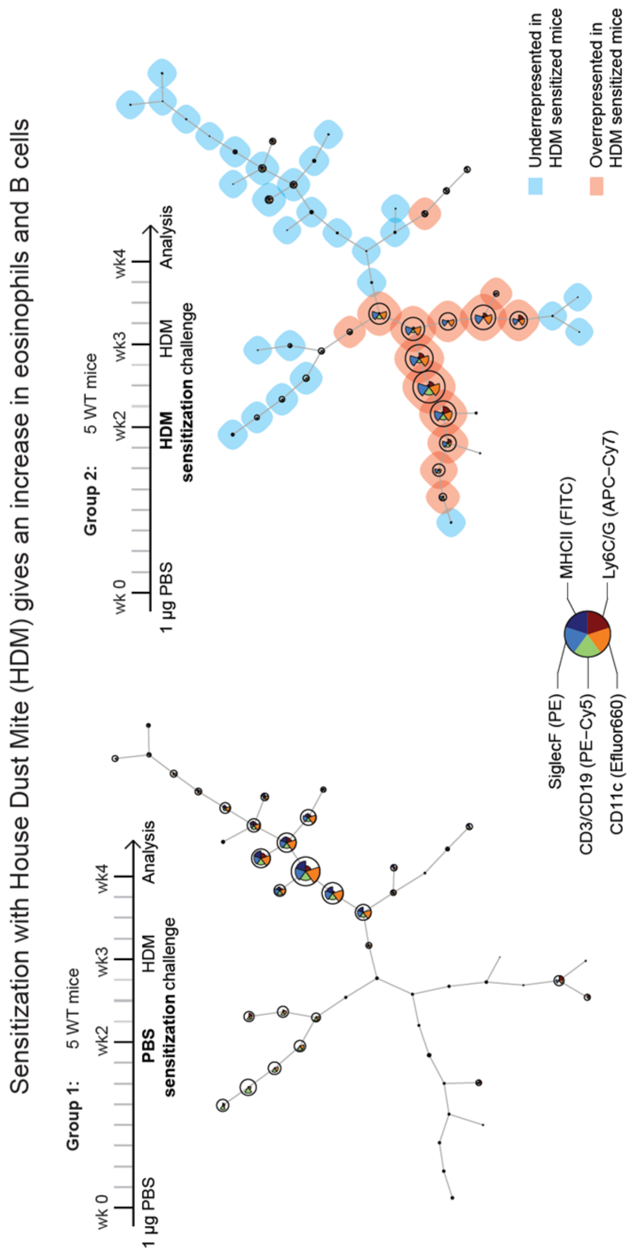


***Figure 3.4:*** *Comparison between two groups of mice with different treatment. The group sensitized with HDM have an asthma phenotype, with a huge increase in eosinophils and a little increase in B cells.*

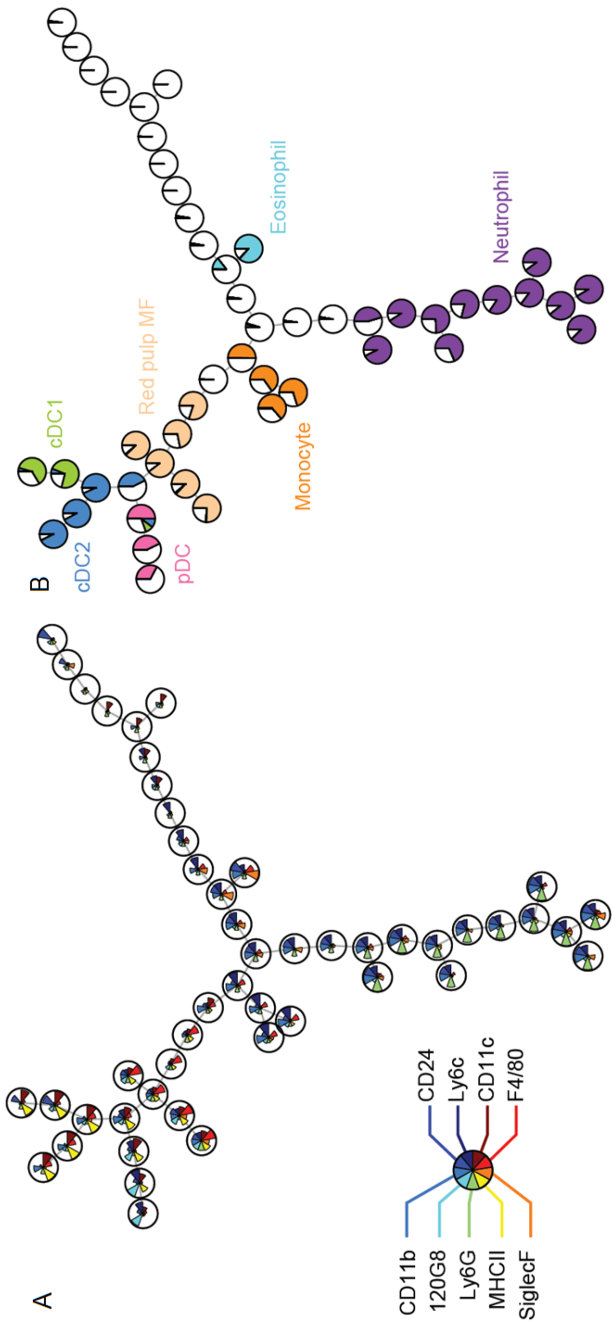


***Figure 3.5:*** *FlowSOM analysis of splenocytes of 15 mice, stained with a myeloid panel.*

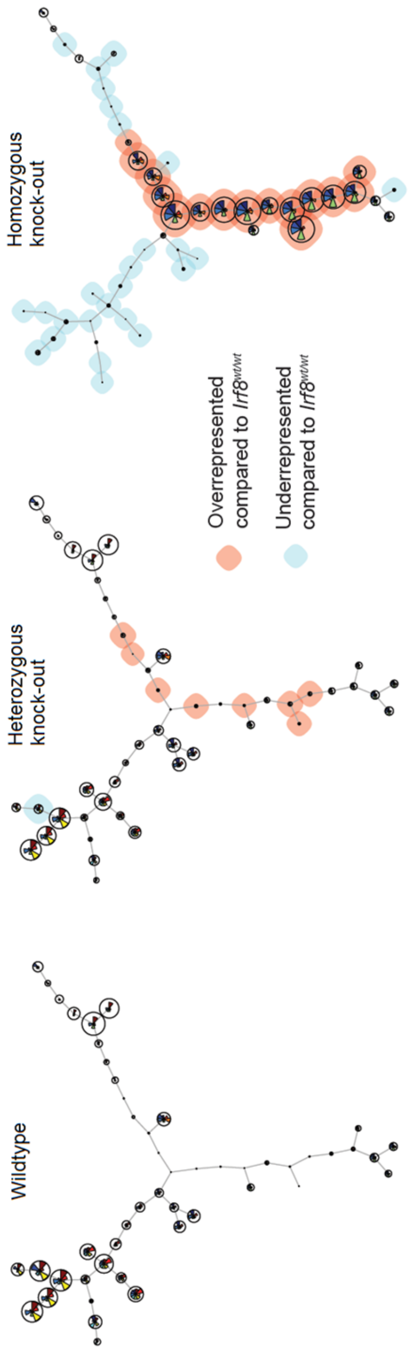

Homozygous knock-out
Heterozygous knock-out
Wildtype
Overrepresented compared to $Irf8^{wt/wt}$
Underrepresented compared to $Irf8^{wt/wt}$

***Figure 3.6:*** *Comparison of the 3 groups of mice. A strong neutrophilic increase is shown in the homozygous knock-out, which is much more tempered in the heterozygous knock-out.*

advance. By automatically matching nodes to these definitions, it becomes much quicker to interpret the FlowSOM tree, while you can still easily detect unexpected marker combinations and cell types.

We developed a scoring function which can automatically test for each node how well it corresponds with a given cell type definition, by comparing the node's MFIs with the values in the other nodes. The researcher first needs to define for which markers a certain cell type is positive or negative. Intermediate expression is not yet supported and it is important to pick only those markers with a clear pattern, as others will add too much noise. Then we compute for every node and every marker of interest, the difference with the maximal or minimal MFI present in any of the nodes (depending on whether the marker should be positive or negative). This value indicates how strong this node deviates from the expected pattern. As a small difference is not as bad as a big difference (a node where one marker does not correspond to the definition at all means the node should not be picked, while some small variations between the nodes are expected), we use the square of the difference to enlarge these differences. Finally, all these scores are scaled between 0 and 1 (1 meaning it corresponds perfectly with the wanted pattern, 0 meaning it is the furthest away from it) and then averaged over all markers in the cell type definition. This way, every node has a score for every cell type. To make a final selection of nodes which correspond with a cell type, a threshold is put a 0.95 * the highest score. Pseudocode is provided in Algorithm 2. By applying this algorithm to all cell types of interest and selecting only the nodes which get the highest scores, an annotated tree can be returned.

We tested this method on a flow cytometry dataset stained with a lymphocyte panel. B cells, CD4 T cells, CD8 T cells, NK cells and gamma delta T cells were successfully automatically identified, using the definitions in Table 3.1. The results are shown in Figure 3.7. This allows for an easier interpretation of the FlowSOM tree and a more straightforward workflow when analyzing new datasets.

Additionally, we also used this method on a separate experiment to detect conventional dendritic cells type 1 and 2 in multiple tissues, as published in [2]. The panel design proposed by the researchers was strong enough to automatically detect the cDC1 and cDC2 cell types across multiple tissues. Figure 3.8 shows the results for multiple runs of the FlowSOM algorithm in comparison to the manual gating results. While the stochasticity of the algorithm results in slightly varying results, they always show a very similar pattern when comparing the mice, which also corresponds nicely to the pattern manually identified.

***algorithm 2:** Automated labeling of clusters*

```
procedure Annotation
1:  nodes ← $k \times m$ matrix, with $m$ markers describing $k$ cluster centers
2:  cellTypeDefinitions ← a list in which every cell type defines a list of markers
    which should be positive or negative
3:  selection ← new list
4:  for cellType in cellTypeDefinitions do
5:      scores ← repeat(0,$k$)
6:      for marker in cellTypeDefinitions[cellType] do
7:          if cellTypeDefinitions[cellType][marker] == positive then
8:              scoresMarker ← (nodes[,marker] - max(nodes[,marker]))$^2$
9:          else if cellTypeDefinitions[cellType][marker] == negative then
10:             scoresMarker ← (nodes[,marker] - min(nodes[,marker]))$^2$
11:         scores ← scores + 1 - $\frac{\text{(scoresMarker - min(scoresMarker))}}{\text{(max(scoresMarker) - min(scoresMarker))}}$
12:     selection[cellType] ← which(scores $\geq$ 0.95*max(scores))
13: return selection
```

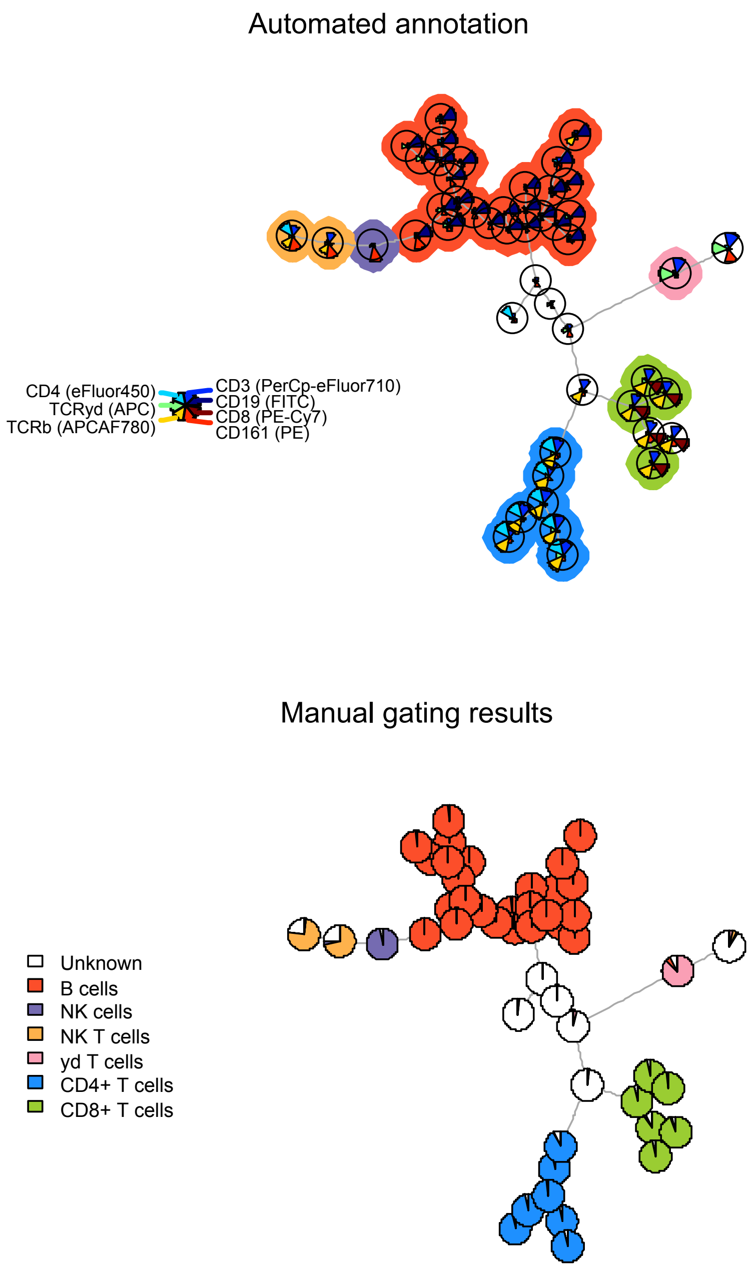


***Figure 3.7:*** *Automated annotation of the FlowSOM tree. A very strong correspondence between the automated results and the manual gating results can be seen, allowing an easier interpretation of the tree if the manual gating is not available.*

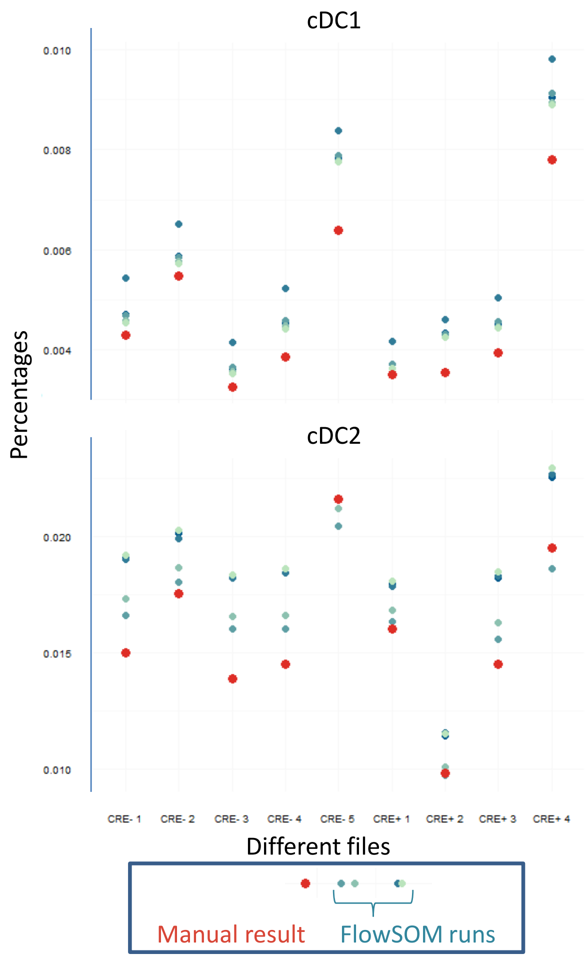


*Figure 3.8:* *Comparison between automated and manual labeling. Although the percentage of cells automatically annotated is always slightly larger, the general pattern of the manual gating can still be clearly determined. The slight differences can be explained because the manual gates are often put quite strict, while in FlowSOM all cells need to be assigned to a node.*

## References

[1] D. Sichien, C. L. Scott, L. Martens, M. Vanderkerken, S. Van Gassen, M. Plantinga, T. Joeris, S. De Prijck, L. Vanhoutte, M. Vanheerswynghels, et al. *IRF8 Transcription Factor Controls Survival and Function of Terminally Differentiated Conventional and Plasmacytoid Dendritic Cells, Respectively*. Immunity, 45(3):626–640, 2016.

[2] M. Guilliams, C.-A. Dutertre, C. L. Scott, N. McGovern, D. Sichien, S. Chakarov, S. Van Gassen, J. Chen, M. Poidinger, S. De Prijck, et al. *Unsupervised high-dimensional analysis aligns dendritic cells across tissues and species*. Immunity, 45(3):669–684, 2016.

# 4 FloReMi: Flow Density Survival Regression Using Minimal Feature Redundancy

*In this chapter, we describe our participation to the FlowCAP IV challenge. The goal of the challenge was to identify cell populations that can prognose the progression time of HIV patients to AIDS. To tackle this challenge, we developed FloReMi, a full pipeline consisting of multiple building blocks fitting after each other: several preprocessing steps, feature extraction, feature selection and survival time prediction. The feature extraction resulted in almost two and a half million features per patient, necessitating the feature selection step. We used a supervised selection process, picking features which correlated well with the progression time, while having minimal correlation with the other features picked. We tried multiple models for the final prognosis prediction, but the random survival forest was the one which ended top-ranked in the FlowCAP IV evaluation. While this analysis is not as visual as the FlowSOM approach described in the previous chapters, it allows the evaluation of millions of features. In this setting the goal was very clear (correlation with clinical outcome), thus it was not necessary to visualize the whole distribution of the cells. Where FlowSOM is more useful for exploratory data analysis, the FloReMi pipeline can be used to generate a list of potential biomarkers, which can then be validated in further experiments.*

**Sofie Van Gassen, Celine Vens, Tom Dhaene, Bart N. Lambrecht and Yvan Saeys.**

**Published in Cytometry Part A, 2016; 89(1), 22-29.**

**Abstract** Advances in flow cytometry bioinformatics have resulted in a wide variety of clustering, classification and visualization techniques. To objectively evaluate the performance of such methods, common benchmarks such as the FlowCAP initiative have proven to be of great value. In this work, we report on a novel method, FloReMi, which was developed to tackle the most recent FlowCAP IV challenge. This challenge was formulated as a survival modeling problem, where participants were expected to design a model to predict the time until progression to AIDS for HIV patients. It is known that variability in progression rate cannot be fully predicted by simple CD4$^+$ T cell counts. However, it is hypothesized that the immunopathogenesis established early in HIV already indicates the course of future disease. Adequately estimating the progression rate of HIV patients is crucial in their treatment. Using an automated pipeline to preprocess the data, and subsequently identify and select informative cell subsets, a survival regression method based on random survival forests was built, which obtained the best results of all submitted approaches to the FlowCAP IV challenge.

## 4.1 Introduction

Current cytometry techniques enable researchers to examine many markers at the same time. This gives an unprecedented view on single cells, but also introduces several challenges. When many markers are measured simultaneously, manual analysis becomes very time-consuming and subjective. As it is infeasible to manually analyze every possible cell population, only a subset of them is examined based on previous experience. Several research groups have developed automatic clustering techniques to assist the manual analysis, based on K-means, mixture models, density estimation and more (e.g. [1–8]). However, the goal of an experiment is often not to identify all the cell types that are present, but to identify those cell types that are indicative of some phenotype. In this case, machine learning techniques can be used to identify subpopulations of cells in the dataset which can be used to predict the phenotype of a patient. A number of techniques have already been developed with this goal in mind. The FlowCAP II challenge [9] was created to compare those techniques and provide some benchmark data. Several algorithms were proposed, most of which combine an automatic clustering algorithm with a traditional classification algorithm. Once the clusters are formed, features can be constructed and selected, which can be used by traditional classification algorithms. One of the proposed algorithms, flowType [10, 11], computes a threshold for every marker to express every possible subpopulation as a certain combination of high/low marker values. Once all these subtypes are defined, statistical tests are performed to identify those subtypes which contain information with regard to the class of the patients. The Citrus algorithm [12] uses a hierarchical clustering algorithm to split the dataset into many possible subdivisions. Afterwards, statistical tests are used to find significant differences between the data of two classes. Classification algorithms trained with those features can be used to diagnose patients. The majority of algorithms that were proposed in this field focus on classification tasks, where a class, e.g., a disease status, is assigned to each patient. Less attention has been given to predict continuous output variables based on flow cytometry data. To address this issue, the FlowCAP IV challenge was proposed. The goal of the FlowCAP IV challenge was to predict the time until progression to AIDS for HIV patients, a task which was manually studied in [13]. In this article, we present our approach for the FlowCAP IV challenge, combining the flowType algorithm with a feature selection algorithm to identify informative, non-redundant features. We evaluated three survival time prediction algorithms using the selected features, of which the random survival forest approach was the most successful, and obtained the best predictive performance of all methods submitted to the challenge.

## 4.2 Problem Definition

The goal of the FlowCAP IV challenge was to predict the time until progression to AIDS for a test set of 192 HIV patients, based on a training set of 191 patients. Patients were described by flow cytometry data, from which features needed to be extracted to reach this goal. Identifying features that correlate with the time until progression to AIDS, e.g. the size of a specific cell population in the dataset, was also a goal of this challenge.

For each patient, a PBMC sample stimulated with HIVGag peptides and an unstimulated control were provided. The unstimulated sample gives an indication of the baseline state of the patient, whereas in the stimulated sample immune response effects to the antigens might be observed. For each sample, FSC-A, FSC-H, SSC-A and 13 fluorescence channels were measured (indicating values for IFN$\gamma$, TNF$\alpha$, CD4, CD27, CD107-A, CD154, CD3, CCR7, IL2, CD8, CD57, CD45RO and V-Amine/CD14). Each patient of the training set was also assigned a label indicating the observed clinical status (1 = progression to AIDS or death, 0 = no progression to AIDS or death) and, if there was progression to AIDS or death, the survival time until the onset of AIDS. If there was no progression to AIDS, the time to the last evaluation was given. It is important to notice that a label of zero does not mean that the patient did not develop AIDS, but only that the patient at least did not develop AIDS until this last observation. This kind of data is called censored information, because what happens after the last evaluation is unknown. From the 191 patients in the training set, only 34 had actual events and 157 were censored. In the test set, 45 patients had actual events and 147 were censored. This strongly complicated the computational framework we had to use to build a model for this data.

The evaluation criterion for the challenge was based on the Cox proportional-hazards model [14]. This model uses both event data and censored data to study the dependency of the survival times on predictor variables and results in a P value indicating how well the variables fit the survival times. During evaluation, the predicted survival times are used as predictor variable.

## 4.3 The FloReMi Algorithm

The FloReMi approach consists of four steps. First the data is preprocessed, in order to remove noise. Subsequently many features (i.e., properties pertaining to certain cell types) are extracted, after which a selection of these features is made. Finally, we use the selected features in a regression model to predict the time until the detection of AIDS for the patients. A schematic overview of our approach is given in Figure 4.1. Our method was developed specifically for the FlowCAP IV challenge, but our scripts are available at www.github.com/ SofieVG/FloReMi and can be adapted for other datasets.

### 4.3.1 Preprocessing

The preprocessing step was applied to each sample separately and consisted of six parts.

We started with a quality control step to detect problems during the acquisition of the sample by the cytometer. This can be done by inspecting uniformity of the data with respect to the time parameter [15]. Therefore, we split the dataset in 100 equally sized intervals for the Time parameter. We calculated the median FSC-A value and the number of cells for each interval. Intervals were removed completely if either their median FSC-A value differed $>10{,}000$ from the interval right before or after it, or if the number of cells in the interval was less than the median number of cells per interval minus two standard deviations. These thresholds were defined after inspecting several problematic sample files. By removing these intervals, we removed measurements with inconsistent values, caused by e.g. disturbances of the flow stream, air bubbles or clogging of the flow cell. A similar technique has been proposed by FletezBrant, which is released in the flowClean R bioconductor library [16]. On average, 5.30% ($\pm$3.60%) of the original cells were removed by this step.

In the next preprocessing part, we removed all margin events. We defined a

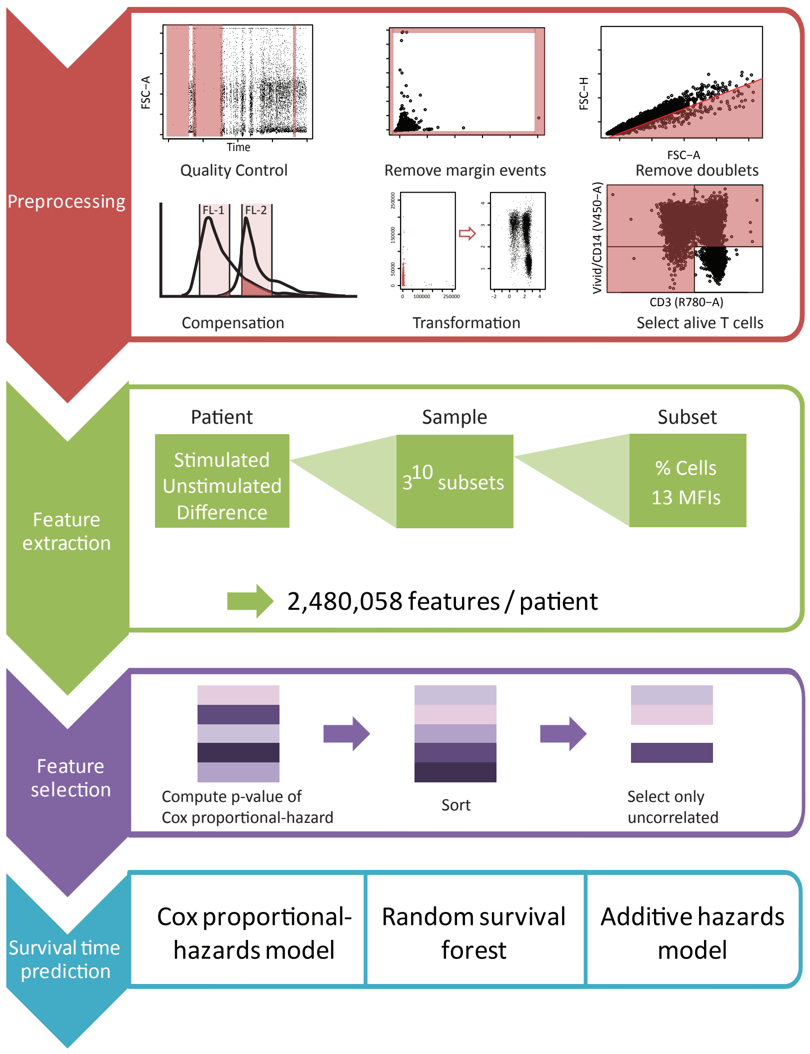


***Figure 4.1:*** *Overview of the four steps of the FloReMi algorithm: preprocessing, feature extraction, feature selection and survival time prediction. (i) During the preprocessing, six steps are executed: problems during measurement are detected, margin events and doublets are removed, the data is compensated and logicle-transformed and alive T cells are selected. (ii) During feature extraction, $3^{10}$ subsets are identified. Each subset can be described by the percentage of cells present in the subset and 13 MFI values. All these features are computed for the stimulated and the unstimulated data, and then also the differences between the features for the two data sets are added. This results in 2,480,058 features per patient. (iii) During feature selection, a Cox proportional-hazards model is built for each feature separately, and the features are sorted by P value (lowest first). For the actual selection, we start with the two first features and only add those which have pairwise correlation lower than 0.2 to all other selected features. (iv) In the final step, we evaluated three different survival time prediction models: the Cox proportional-hazards model, the random survival forest and the additive hazards model.*

margin event as a measurement which had either the minimum or maximum value in any dimension, or a measurement which exceeded the ranges given in the description of the fcs-files. Measurements which exceeded the possible ranges were erroneous, and measurements which had the minimum or maximum value might be saturated and out of the detection range of the cytometer, thus not representing the actual value that should be measured. A similar technique is available in the flow cytometry module of Gene- Pattern: RemoveSaturatedFCSEvents [17]. On average, 2.72%($\pm$3.83%) of the cells remaining after the first preprocessing step were removed by this step.

Our third preprocessing step consisted of removing doublets. We computed the ratio $r$ between FSC-A and FSC-H, since for doublets, the area measured is larger in proportion to the height, because the signal has the same strength but a longer duration in comparison with a single cell [18]. We removed all cells for which the ratio was larger than the median ratio of all cells plus two standard deviations, on average 4.45% ($\pm$1.25%) of the remaining cells after the first two preprocessing steps.

$$r_{\text{cell to keep}} \leq \text{median}(r_{\text{all cells}}) \; + \; 2 \; \text{stdev}(r_{\text{all cells}})$$

The fourth and fifth steps in our preprocessing procedure were the traditional flow cytometry preprocessing steps: compensation and transformation. We compensated the data using the spillover matrix provided in the fcs-files. We transformed the SSC channel and all color channels with the `logicleTransform()` function from the R flowCore package [19], using all default settings.

Our final preprocessing step helped us to zoom in on the area of interest. We used an automatic gating step to select only the alive T-cells for further analysis. To do this, we used the R flowDensity package [20]. This package can automatically determine an optimal split in a single dimension of the dataset. We used this to determine thresholds for both the V450-A (Vivid/CD14) and R780-A (CD3) channels. We selected those cells that were low for V450-A (alive, no macrophages or monocytes) and high for R780-A (T cells).

### 4.3.2 Feature Extraction

Once the dataset was clean, we extracted features from it. By using automatic, unsupervised techniques, we were able to examine a much larger scope of features than would be possible in any manual analysis. The feature extraction part of the pipeline was executed on each sample separately.

First, we used the flowDensity algorithm [20] again to determine splits on ten dimensions: FSC-A, SSC-A, G710-A (CD4), G660-A (CD27), G610-A (CD107-A), G560-A, (CD154), R710-A (CCR7), V800-A (CD8), V585-A (CD57), V545-A (CD45RO). This algorithm uses the density distribution of the cells to determine the best possible split. If two peaks are detected, the minimum intersection point between the two peaks is used. If there are more peaks, it takes into account the distance between the peaks and the height of the valleys to determine which split gives the clearest cut. If no peaks are detected, it will use the 95th percentile to split on. We excluded the FSC-H, V450-A, and R780-A channels because they were dealt with in the preprocessing step. We also excluded the intracellular markers IFN$\gamma$, IL2, and TNF$\alpha$, in order to reduce computation time, even though this might lead to a loss of information. These intracellular markers do not have two clear peaks typically, which makes it harder to get a good split automatically.

The second step in the feature extraction process was to define subsets, groups of cells which have the same annotation of high/low marker intensities, based on the thresholds determined by the flowDensity algorithm. We did this by using the flowType algorithm [10], examining every possible marker combination for which the values are either high, low, or neutral. By exploring all cell subsets with combinations of high and low expression of the markers, we included many possible cell types that might not be identified in a manual analysis. The flowType bioconductor package provides an efficient implementation to assign cells to the subpopulations they belong to, using dynamic programming as an optimization for the combinatorial problem [11]. This led to $3^{10}$ possibilities or 59,049 different subsets.

Once each subset was defined, we extracted features for each sample. For each subset, we computed the percentage of cells and the mean fluorescence intensity for 13 markers (FSC-A, SSC-A, B515-A (IFN$\gamma$), G780-A (TNF$\alpha$), G710-A (CD4), G660-A (CD27), G610-A (CD107-A), G560-A (CD154), R710-A (CCR7), R660-A (IL2), V800-A (CD8), V585-A (CD57), V545-A (CD45RO)). Because we included the intracellular markers IFN$\gamma$, TNF$\alpha$ and IL2, their information might still be used even though they were not included in the subset definitions. This leads to 14 features per subset. Because there was both stimulated and unstimulated data present for each patient, we computed those 14 features for all subsets for both samples, and also the difference between the corresponding features from each sample. This resulted in $(3^{10} \times 14) \times 3$ or 2,480,058 features per patient.

### 4.3.3 Feature Selection

Regression techniques are not able to efficiently handle such a huge amount of features. To solve this problem, the third step of our pipeline was a feature selection step. In this step, we wanted to select those features which have a high correlation with the survival time. However, because of the high percentage of censored values, we could not simply use the Pearson correlation to measure the importance of a feature. Therefore, we made use of the Cox proportional-hazards model.

We used the whole training dataset to compute a Cox proportional-hazards model for each feature separately. This model returned a P value and a concordance index, which indicated how strongly the feature itself corresponded with the actual survival times of the patients. We ranked the features by the P values from their corresponding models.

Strongly correlated features will have a negative impact on a Cox proportional-hazards model. To minimize the redundancy between the selected features, we therefore did not simply pick the first k features from the resulting list. Instead, we started with the two features with the lowest P values, and then iteratively added new features when the pairwise Pearson correlation between the previously selected features and the new candidate feature was low enough. We chose a threshold of 0.2, to make sure that no strong correlations exist between the selected features. When using different thresholds, such as 0.15 or 0.25, different features were selected (Supporting Information Tables 4.4 and 4.5). However, final prediction results were very similar (Supporting Information Table 4.6).

### 4.3.4 Survival Time Prediction

The selected features were used to predict the actual survival time of the patients: we converted the original training and test datasets consisting of all fcs files to two matrices in which the rows represented the patients and the columns the selected features. We evaluated three different regression techniques to reach this goal: Cox proportional-hazards regression [14], random survival forests [21] and additive hazards regression [22]. For each technique, we performed leave-one-out cross validation on the training dataset to evaluate our results, and built a final model using the whole training dataset to make predictions for the test set.

Next to the P value of the Cox proportional-hazards model, we also used the concordance index [23] to evaluate our results. To compute the concordance index, all pairs of comparable patients are checked. A pair is comparable when either both patients have progressed to AIDS, or one patient has progressed to AIDS in a shorter time than the time to the last evaluation of the censored patient. The concordance index is the percentage of pairs for which the predicted survival time order corresponds to the actual order. This means a random assignment will lead to a score of about 0.5, whereas a perfect assignment will give a score of 1 and a reverse assignment will give a score of 0.

The first technique we executed was the Cox proportional-hazards regression. We built this model with an increasing number of uncorrelated features until the corresponding concordance index did no longer improve, as illustrated in Figure 4.2. This resulted in a feature set of 13 features, as presented in Table 4.3. It might be surprising that no TNF$\alpha$ related features are included in this set. However, notice that we start with the best scored features and only add those with

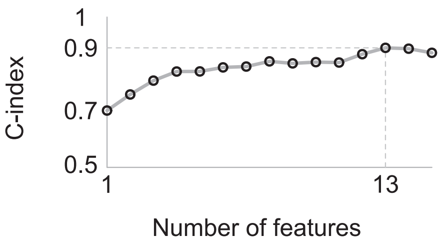


***Figure 4.2:*** *Concordance indices used to choose our feature selection cut-off. Using 13 features, the concordance index was optimal for the training set.*

almost no correlation (<0.2) with the features already selected. Some other features with better p-values and a correlation greater than 0.2 with the TNF$\alpha$ related features are added first to the selection, which results in no TNF$\alpha$ related features in our end selection. E.g. our third selected feature, the difference of the CD107a MFI values of the SSC- CD27+ CD107a+ CD154- CD8+ CD45RO- cell subset, has a correlation of 0.25 with the best ranking TNF$\alpha$ related feature (the TNF$\alpha$ MFI of the stimulated sample of the SSC+ CD4- CD27+ CD107a- CCR7- CD57- CD45RO- cell subset).

The second technique we evaluated was the random survival forest, as implemented by the randomForestSRC package [24]. The random survival forest uses survival trees, in which each split is made in such a way that it maximizes the survival difference between the daughter nodes. It explicitly takes censored data into account. We used the same 13 features as we had used with the Cox proportional-hazards regression and trained a forest with 500 regression trees. This model returns the mortality, rather than the survival time, of patients. To report our results, we scaled the values between 0 and 1 and reversed them, because a higher mortality corresponds with a shorter time until progression to AIDS or death.

Finally, we also used regularization for semiparametric additive hazards regression, as implemented by the ahaz package [25]. This method is a regularized version of the standard hazards model, and thus should inherently perform feature selection (similar to standard Lasso or Elastic net for the traditional regression setting). With the best 100 features from the feature selection step, a model was trained by performing a fivefold internal cross validation on the training set to find the optimal parameter settings. These parameter settings were then used to train a final model. We also tested the model with the best 80 or 200 features, but the results were very similar (Supporting Information Table 4.7).

***Table 4.1:*** *Results of the three regression models on leave-one-out cross validation. All methods performed well during cross validation.*

| CROSS-VALIDATION RESULT | COX PROPORTIONAL HAZARDS | RANDOM SURVIVAL FOREST | ADDITIVE HAZARDS |
|---|---|---|---|
| P value | 0.000 | 0.000 | 0.000 |
| Concordance index | 0.891 | 0.852 | 0.815 |

***Table 4.2:*** *Final results of the three regression models on the FlowCAP IV challenge. Only the random survival forest obtains good results for the test set, which might indicate over-fitting for the other models.*

| FINAL RESULTS | ALL DATA | TRAIN | TEST |
|---|---|---|---|
| | COX PROPORTIONAL HAZARDS | | |
| P value | 0.000 | 0.000 | **0.733** |
| Concordance index | 0.662 | 0.932 | **0.459** |
| | RANDOM SURVIVAL FOREST | | |
| P value | 0.000 | 0.000 | **0.002** |
| Concordance index | 0.813 | 0.976 | **0.672** |
| | ADDITIVE HAZARDS | | |
| P value | 0.000 | 0.000 | **0.782** |
| Concordance index | 0.635 | 0.875 | **0.527** |

## 4.4 Results and Discussion

During the challenge, the correct results for the test dataset were obviously hidden for the participants. Therefore, we performed leave-one-out cross validation to evaluate our techniques. We present the results of our cross-validation in Table 4.1. Because all three algorithms performed well for crossvalidation, we submitted all three versions to the FlowCAP IV challenge.

Once the challenge was finished, the correct results for the test dataset were communicated. We present our results in Table 4.2. Surprisingly, the results of the three algorithms differed greatly on the test dataset. Both the Cox proportional-hazards regression and the additive hazards regression algorithm performed very badly. They have concordance scores around 0.5, which corresponds with a random assignment. However, the random survival forest algorithm did perform quite well on the test set, and actually obtained the best result of all participants of the FlowCAP IV challenge. Seven other groups participated, using a whole range of methods: one based on Boruta and FlowFP [26], one based on SPADE [27], one using a regression tree on a target vector combining clinical diagnosis and survival time, gEM/GANN [28], another also based on flowDensity and flowType, but combined with RchyOptimyx [11] and two other methods of which the strategy is not known to us at the moment.

Our results are illustrated in Figure 4.3, where the real survival times are compared to the predicted scores. For the training dataset, a clear correlation between the predictions and the real survival times is present for all algorithms: the line has an upward slope and if it takes more time for an event to occur, the patient will get a higher score. Notice that the steepness of the curve is not informative because we rescaled all our results between 0 and 1 at the end of the prediction process. However, on the test set, we see the same results as described above: there is no correlation at all for the results of the Cox proportional-hazards model and the additive hazards model. For the random survival forest, there is some correlation, but the data spread is much wider: many patients get a score corresponding to their survival time, but several patients get a score that is too high. We want to stress the importance of our feature selection step here. If we pick the 1000 features with the best scores, without our redundancy-based selection process, the concordance index of the Random Survival Forest model drops to 0.512 for the test set.

Because the Cox proportional-hazards regression and the additive hazards regression performed well on the training data, but badly on the test data, we suspect overfitting might be the problem. The models do not generalize to new data, which indicates they are capturing too much specific details of the training set. When a model is used for diagnosis, the main goal is to make predictions for new samples, which are not available at the time of building the model. However, when overfitting on the training dataset, the model uses very specific details of the training set which are not applicable to the whole population of interest. As such, it will fail to make good predictions for new samples, which is the case here in the validation on the test set. In the prediction step we performed leave-one-out cross-validation, but in the feature selection step, the whole training dataset was used to score the features. The random forest approach uses the same features, but might be more robust against overfitting because of the ensemble approach. Intuitively, random forests may also perform better because they are a non-linear model and, as such, they are better able to model interactions between features.

All our results were generated on a single computing node. The feature extraction step took about 3 days, while all other parts took only a couple of minutes, so this part is clearly the bottleneck for a faster workflow. However, it could be easily parallelized, because the preprocessing and feature extraction happens independently for each sample. When a computing cluster is available, this could strongly reduce running time.

The goal of the challenge was not only to predict the survival times, but also to identify features that might be useful for the diagnosis of HIV patients. In Table 4.3, we present the 13 features that were used for the Cox proportional-hazards regression model and the random survival forest. We notice that features from the unstimulated sample, from the stimulated sample and from the differences between the two are selected, indicating that it was essential to use both samples in

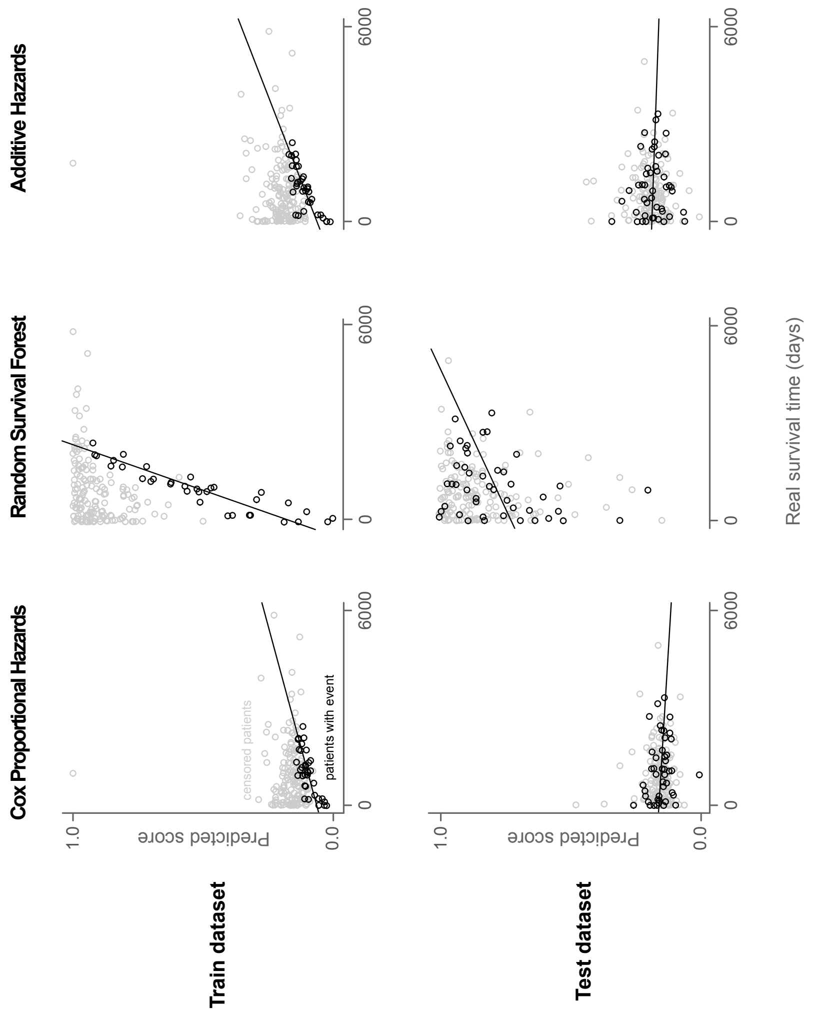


***Figure 4.3:*** *Results of the different models on training and test set. The prediction scores of the models have all been rescaled between 0 and 1 to provide a ranking. On the training set, all models have a correlation with the real survival times, as indicated by the regression line through the patients with events. However, on the test set, only the random survival forest has a correlation. This indicates that the other models overfit on the training set.*

*Table 4.3: Overview of the top 13 features of the feature selection step, which were used by the Cox proportional-hazards model and the random survival forest. Both features from the stimulated and the unstimulated samples are used, as well as comparisons between the two. Both percentages of cells and mean fluorescence intensities (MFI) of specific markers are used. It is remarkable that mainly a negative selection of markers is used in the selected subsets*

| FEATURE | SAMPLE | SUBSET |
|---|---|---|
| Percentage of cells | Unstimulated | CD4- CD27- CD107a- CD154- CD45RO- |
| Percentage of cells | Unstimulated | CD4- CD27- CD154- CD8+ |
| CD107a MFI | Difference | SSC- CD27+ CD107a+ CD154- CD8+ CD45RO- |
| Percentage of cells | Difference | FSC- CD4+ CD107a- CD154- CCR7+ CD8+ CD57- CD45RO- |
| CD4 MFI | Unstimulated | FSC- CCR7- CD57- CD54RO- |
| IL2 MFI | Difference | FSC+ SSC+ CD107a- CCR7- CD8- |
| IL2 MFI | Unstimulated | FSC- SSC- CD4+ CD27- CD107a+ CD8- CD57- |
| IFN$\gamma$ MFI | Difference | CD27- CD107a- CD154+ CCR7- CD57+ CD45RO+ |
| CD57 MFI | Unstimulated | SSC- CD4+ CD27- CD107a- CCR7- CD8+ |
| Percentage of cells | Stimulated | FSC- CD4- CD27- CD154+ CCR7- CD57+ CD45RO+ |
| Percentage of cells | Stimulated | FSC+ SSC+ CD4- CD154- CCR7+ CD8- CD57- |
| CD8 MFI | Difference | CD4+ CD27- CD107a+ CCR7- CD8- CD57- |
| CD8 MFI | Difference | SSC- CD154+ CD57+ CD45RO- |

the analysis. The chosen features contain not only the percentage of cells for specific subpopulations, but also the mean fluorescence intensities for several markers. These features imply that a shift in abundance of markers might happen for certain cell types.

It is important to keep in mind that all population boundaries are automatically determined and might not exactly correspond with a manual gating of the data. Even if this would be the case, it is still quite hard to interpret the features in a biological way. For example, the best feature (see top row in Table 4.3) is negative for all five specified markers and neutral for the others, while traditionally cell types are defined in a positive way, by having a certain marker present. In general, every marker except CD4 is used as least as much as a negative marker than as a positive marker. This might indicate that a certain cell type which does not express the stained markers could correspond well with the progression rate. It is likely that our best feature corresponds with effector CD8$^+$ cells [29], but future research will be necessary to interpret all the features correctly and gain more insight in the process from HIV to AIDS.

## 4.5 Conclusion

In this article we presented the FloReMi approach for the FlowCAP IV challenge, in which we analyzed flow cytometry data to predict survival times of HIV patients. We first thoroughly cleaned the data and extracted more than 2.4 million features for each patient. A feature selection step selected relevant features with minimal redundancy.

We evaluated three different survival time prediction methods, of which only the random survival forest method performed well. This method obtained the best results in the FlowCAP IV challenge by using a selection of 13 features.

It is interesting to notice that the four steps of the FloReMi pipeline work independently of each other. A new preprocessing step, feature extraction method or feature selection method could be plugged in without any problems, and several prediction algorithms can be used, as we did in this article. This leaves much room for improvement for each of the steps separately and poses finding an optimal combination as a new goal.

In future work, we will investigate the optimization of each step of the pipeline. Other score metrics could be used to rank the features, although they do have to take censored data into account. The Random Survival Forest computes a mortality score instead of a survival time, which are closely related but not exactly the same. Other regression techniques which can handle censored data might work better.

Once the algorithm is optimized, more research could be done to interpret the resulting features. Biological validation might be necessary and the results we present can be seen as a starting point for other research projects to unravel the details of HIV.

## Acknowledgment

The authors thank the organizers of the FlowCAP challenge for offering a platform to compare the state-of-the-art results and providing data that stimulates the development of new algorithms.

## 4.6 Supplementary tables

***Table 4.4:*** *Overview of the top features selected with a correlation threshold of 0.15*

| FEATURE | SAMPLE | SUBSET |
|---|---|---|
| Percentage of cells | Unstimulated | CD4- CD27- CD107a- CD154- CD45RO- |
| Percentage of cells | Unstimulated | CD4- CD27- CD154- CD8+ |
| Percentage of cells | Difference | FSC- CD4+ CD107a- CD154- CCR7+ CD8+ CD57- CD45RO- |
| IFN MFI | Stimulated | FSC- CD4+ CD107a+ CD45RO- |
| IL2 MFI | Stimulated | SSC- CD107+ CCR7- CD8- CD45RO- |
| IL2 MFI | Difference | FSC+ SSC+ CD107a- CCR7- CD8- |
| Percentage of cells | Stimulated | SSC+ CD4- CD27- CD107a+ CD154- CCR7+ CD57- |
| CD8 MFI | Difference | CD4+ CD27- CD107a+ CCR7- CD8- CD57- |
| CD8 MFI | Difference | SSC- CD154+ CD57+ CD45RO- |

***Table 4.5:*** *Overview of the top features selected with a correlation threshold of 0.25*

| FEATURE | SAMPLE | SUBSET |
|---|---|---|
| Percentage of cells | Unstimulated | CD4- CD27- CD107a- CD154- CD45RO- |
| Percentage of cells | Unstimulated | CD4- CD27- CD154- CD8+ |
| CD4 MFI | Difference | CD4+ CD27+ CCR7- CD8- CD45RO+ |
| CD107a MFI | Difference | SSC- CD27+ CD107a+ CD154- CD8+ CD45RO- |
| CCR7 MFI | Unstimulated | FSC- CD4- CD57- CD45RO- |
| CD4 MFI | Difference | FSC+ CD4- CD27- CD57+ |
| IL2 MFI | Difference | SSC- CD27+ CCR7- CD57+ CD45RO+ |
| IL2 MFI | Difference | FSC+ SSC+ CD107a- CCR7- CD8- |
| Percentage of cells | Stimulated | FSC- SSC- CD4- CD107a- CCR7- CD57- |
| Percentage of cells | Difference | FSC- SSC+ CD4- CD27- CD154- CD57- |
| IFN MFI | Difference | CD27- CD107a- CD154+ CCR7- CD57+ CD45RO+ |
| CD8 MFI | Difference | SSC- CD4+ CD107a+ CD8- CD57- |
| Percentage of cells | Stimulated | FSC- CD4- CD27- CD107a+ CD154- CCR7+ CD8- CD57- |

***Table 4.6:*** *The final prediction results are very similar for slight variations in the correlation feature selection threshold, where different features are used instead of the 13 reported ones.*

| THRESHOLD 0.15 | | | |
|---|---|---|---|
| | ALL DATA | TRAIN | TEST |
| | COX PROPORTIONAL HAZARDS | | |
| P value | 0.000 | 0.000 | **0.737** |
| Concordance index | 0.632 | 0.906 | **0.429** |
| | RANDOM SURVIVAL FOREST | | |
| P value | 0.000 | 0.000 | **0.042** |
| Concordance index | 0.763 | 0.973 | **0.572** |

| THRESHOLD 0.25 | | | |
|---|---|---|---|
| | ALL DATA | TRAIN | TEST |
| | COX PROPORTIONAL HAZARDS | | |
| P value | 0.000 | 0.000 | **0.634** |
| Concordance index | 0.676 | 0.899 | **0.517** |
| | RANDOM SURVIVAL FOREST | | |
| P value | 0.000 | 0.000 | **0.015** |
| Concordance index | 0.828 | 0.973 | **0.707** |

***Table 4.7:*** *The additive hazards method is quite robust to changes in the number of input features, because it includes regularization in the model building.*

| | ALL DATA | TRAIN | TEST |
|---|---|---|---|
| | ADDITIVE HAZARDS 80 FEATURES | | |
| P value | 0.000 | 0.000 | **0.832** |
| Concordance index | 0.629 | 0.871 | **0.540** |
| | ADDITIVE HAZARDS 100 FEATURES | | |
| P value | 0.000 | 0.000 | **0.782** |
| Concordance index | 0.635 | 0.875 | **0.527** |
| | ADDITIVE HAZARDS 200 FEATURES | | |
| P value | 0.000 | 0.000 | **0.578** |
| Concordance index | 0.645 | 0.878 | **0.478** |

## References

[1] K. Lo, R. R. Brinkman, and R. Gottardo. *Automated gating of flow cytometry data via robust model-based clustering*. In Cytometry Part A, volume 73, pages 321–332, 2008.

[2] I. P. Sugár and S. C. Sealfon. *Misty Mountain clustering: application to fast unsupervised flow cytometry gating.* BMC bioinformatics, 11(1):502, 2010.

[3] H. Zare, P. Shooshtari, A. Gupta, and R. R. Brinkman. *Data reduction for spectral clustering to analyze high throughput flow cytometry data*. BMC Bioinformatics, 11(403):1–16, 2010.

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

[5] A. Cron, C. Gouttefangeas, J. Frelinger, L. Lin, S. K. Singh, C. M. Britten, M. J. P. Welters, S. H. van der Burg, M. West, and C. Chan. *Hierarchical Modeling for Rare Event Detection and Cell Subset Alignment across Flow Cytometry Samples*. PLoS Computational Biology, 9(7), 2013.

[6] Y. Ge and S. C. Sealfon. *Flowpeaks: A fast unsupervised clustering for flow cytometry data via K-means and density peak finding*. Bioinformatics, 28(15):2052–2058, 2012.

[7] I. Naim, S. Datta, J. Rebhahn, J. S. Cavenaugh, T. R. Mosmann, and G. Sharma. *SWIFTscalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, Part 1: Algorithm design*. Cytometry Part A, 85(5):408–421, 2014.

[8] S. Pyne, S. X. Lee, K. Wang, J. Irish, P. Tamayo, M. D. Nazaire, T. Duong, S. K. Ng, D. Hafler, R. Levy, G. P. Nolan, J. Mesirov, and G. J. McLachlan. *Joint modeling and registration of cell populations in cohorts of high-dimensional flow cytometric data*. PLoS ONE, 9(7), 2014.

[9] N. Aghaeepour, G. Finak, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann. *Critical assessment of automated flow cytometry data analysis techniques.* Nature methods, 10(3):228–38, 2013.

[10] N. Aghaeepour, P. K. Chattopadhyay, A. Ganesan, K. O'neill, H. Zare, A. Jalali, H. H. Hoos, M. Roederer, and R. R. Brinkman. *Early immunologic correlates of HIV protection can be identified from computational analysis of complex multivariate T-cell flow cytometry assays*. Bioinformatics, 28(7):1009–1016, 2012.

[11] K. O'Neill, A. Jalali, N. Aghaeepour, H. Hoos, and R. R. Brinkman. *Enhanced flowType/RchyOptimyx: A Bioconductor pipeline for discovery in high-dimensional cytometry data*. Bioinformatics, 30(9):1329–1330, 2014.

[12] R. V. Bruggner, B. Bodenmiller, D. L. Dill, R. J. Tibshirani, and G. P. Nolan. *Automated identification of stratifying signatures in cellular subpopulations.* Proceedings of the National Academy of Sciences of the United States of America, 111(26):E2770–7, 2014.

[13] A. Ganesan, P. K. Chattopadhyay, T. M. Brodie, J. Qin, W. Gu, J. R. Mascola, N. L. Michael, D. A. Follmann, and M. Roederer. *Immunologic and virologic events in early HIV infection predict subsequent rate of progression.* The Journal of infectious diseases, 201(2):272–84, 2010.

[14] J. Fox and S. Weisberg. *An $\{R\}$ Companion to Applied Regression, Second Edition*. Sage, (September 2012):2016, 2011.

[15] J. V. Watson. *Time, a quality???control parameter in flow cytometry*. Cytometry, 8(6):646–649, 1987.

[16] K. Fletez-Brant, J. Špidlen, R. R. Brinkman, M. Roederer, and P. K. Chattopadhyay. *flowClean: Automated identification and removal of fluorescence anomalies in flow cytometry data*. Cytometry Part A, 89(5):461–471, 2016.

[17] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, and J. P. Mesirov. *GenePattern 2.0.* Nature genetics, 38(5):500–1, 2006.

[18] Y. D. Mahnke and M. Roederer. *OMIP-001: Quality and phenotype of Ag-responsive human T-cells*. Cytometry Part A, 77(9):819–820, 2010.

[19] N. L. Meur, F. Hahne, and B. Ellis. *FlowCore: data structures package for flow cytometry data*. 2007.

[20] M. Malek, M. J. Taghiyar, L. Chong, G. Finak, R. Gottardo, and R. R. Brinkman. *FlowDensity: Reproducing manual gating of flow cytometry data by automated density-based cell population identification*. Bioinformatics, 31(4):606–607, 2015.

[21] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. *Random survival forests*. Annals of Applied Statistics, 2(3):841–860, 2008.

[22] D. Y. Lin and Z. Ying. *Additive Hazards Regression Models for Survival Data*. In Proceedings of the First Seattle Symposium in Biostatistics: Survival Analysis, pages 185–198. 1997.

[23] M. D. B. Harrell FE Lee KL. *TUTORIAL IN BIOSTATISTICS - MULTIVARIABLE PROGNOSTIC MODELS: ISSUES IN DEVELOPING MODELS, EVALUATING ASSUMPTIONS AND ADEQUACY, AND MEASURING AND REDUCING ERRORS*. Statistics in Medicine, 1996.

[24] H. Ishwaran and U. Kogalur. *Random Forests for Survival, Regression and Classification (RF-SRC), R package version 1.5.5*. URL http://CRAN. R-project. org/package= randomForestSRC, 2014.

[25] A. Gorst-Rasmussen and T. H. Scheike. *Coordinate Descent Methods for the Penalized Semiparametric Additive Hazards Model*. Journal of Statistical Software, 47(9):??–??, 2012.

[26] W. T. Rogers and H. A. Holyst. *FlowFP: A Bioconductor Package for Fingerprinting Flow Cytometric Data*. Advances in Bioinformatics, 2009:1–11, 2009.

[27] 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):886–91, 2011.

[28] D. L. Tong, G. R. Ball, and A. G. Pockley. *gEM/GANN: A multivariate computational strategy for auto-characterizing relationships between cellular and clinical phenotypes and predicting disease progression time using high-dimensional flow cytometry data*. Cytometry Part A, 87(7):616–623, 2015.

[29] P. K. Chattopadhyay and M. Roederer. *Good cell, bad cell: Flow cytometry reveals T-cell subsets important in HIV disease*, 2010.

# 5
# Computational Flow Cytometry: Helping to Make Sense of High-Dimensional Immunology Data

*In the previous chapters, FlowSOM and FloReMi, two algorithms for analyzing flow cytometry data, were developed. Other research groups have also been developing several alternatives to the traditional gating approach. However, for immunologists, it is not straightforward to learn about new algorithmic development and to see how all these algorithms relate to each other. Therefore, this chapter gives an overview of the current state of the art in computational flow cytometry. This can help the immunologist to determine which approach might be most suited for their specific data and the computer scientist to see where there might be still missing parts that could use further development.*

⋆ ⋆ ⋆

**Yvan Saeys, Sofie Van Gassen and Bart N. Lambrecht.**

**Published in Nature Reviews Immunology, 2016; 16(7), 449-462.**

**Abstract** Recent advances in flow cytometry allow scientists to measure an increasing number of parameters per cell, generating huge and high-dimensional datasets. To analyse, visualize and interpret these data, newly available computa-

tional techniques should be adopted, evaluated and improved upon by the immunological community. Computational flow cytometry is emerging as an important new field at the intersection of immunology and computational biology; it allows new biological knowledge to be extracted from high-throughput single-cell data. This Review provides non-experts with a broad and practical overview of the many recent developments in computational flow cytometry.

## 5.1 Introduction

Flow cytometry measures multiple parameters of cells that rapidly flow in a stream through a system of photonic detectors. During the past 50 years, flow cytometry has established itself as the workhorse for high-throughput quantitative analysis of cells and other particles, making it one of the earliest 'omics' technologies in retrospect. Since its inception in 1965 [1] and its first use in the 1970s [2], the basic design of flow cytometers has remained almost unchanged, highlighting the robust design of the technology [3]. Several other factors have contributed to the success and widespread use of flow cytometry. These include the speed at which cells are analysed (allowing large numbers of cells to be measured), the high accuracy and resolution of the technology, and the low operating costs per sample. In addition, flow cytometry is regarded as a non-destructive technology, being able to viably sort cell populations for further subsequent analyses.

The ability to analyse and sort single cells has resulted in a wide range of biological and medical applications. In immunology, flow cytometry is used to identify and quantify populations of immune cells, which allows the immune status of patients to be monitored over time, and biomarkers can be detected by comparing various patient groups.

The past decades have marked a number of key technological advances in cytometry, all of which increase the number of parameters that can be measured simultaneously from single particles. For conventional flow cytometry, the most notable advance has been the extension of the classical design with additional and more powerful lasers. Together with advances in fluorophore design, 18-parameter flow cytometry is now routinely used [4], 30-parameter flow cytometers have recently become commercially available and 50-parameter flow cytometry is projected to be available soon [5]. This will allow the user to perform analyses comparable to those using another recent innovation in the field, mass cytometry [6], which is more widely known as cytometry by time-of-flight (CyTOF). In this system, antibodies are labelled with metal-conjugated probes instead of fluorophores, and a time-of-flight detector is used to quantify the signal, avoiding the problem of spectral overlap associated with classical fluorophore-based flow cytometry. Theoretically, mass cytometry allows the detection of up to 100 parameters per cell, but the throughput is lower than classical flow cytometry, and cells are destroyed during the process, precluding cell-sorting applications.

Another advance in the field is the recent introduction of spectral flow cytometry [7], in which the classical optics and detectors are replaced by dispersive optics and a linear array of detectors that measures the full emission spectrum. Spectral unmixing algorithms are subsequently used to deconvolve the signal, similar to the classical compensation process in conventional flow cytometry. As more spectral information is used to reconstruct the intensity signal, spectral flow cytometry seems a promising technique to alleviate the problems with compensation and autofluorescence, as typically encountered in classical multiparameter flow cytometry.

Technologies that combine cytometry and imaging have also recently emerged in the field. Imaging flow cytometry [8] combines flow cytometry with a high-resolution multispectral imaging system, acquiring several images per cell, at current rates of 5,000 cells per second. Hundreds of image-based features can subsequently be extracted, giving additional information regarding, for example, morphology, colocalization and cell signalling. Imaging mass cytometry uses a different approach, using mass cytometry to reconstruct tissue images [9]. By scanning a tissue section, vaporizing the tissue spot by spot and subsequently feeding it into a mass cytometer, information for 32 markers is obtained for every spot. Imaging software is subsequently used to combine all spot information and extract information at the single cell level.

The high-throughput nature of flow cytometry, combined with the increasing capacity to measure more cell parameters at once, is generating massive and high-dimensional datasets on a routine daily basis. These data can no longer be adequately analysed using the classical, mostly manual, analysis techniques and therefore require the development of novel computational techniques, as well as their adoption by the broad community. Computational flow cytometry is a new discipline that straddles the fields of immunology and computational biology and provides a set of tools to analyse, visualize and interpret large amounts of cell data in a more automated and unbiased way.

This Review summarizes the potential of computational flow cytometry methods by providing a broad overview of the types of analyses that can be performed using these new tools and by highlighting their advantages over the traditional, manual analysis of the data. We start by explaining the need for computational flow cytometry, highlighting the benefits of standardization and reproducibility and the application towards pre-processing and quality control. We cover two of the most widespread types of computational flow cytometry analysis: visualization of high-dimensional cytometry data and the automated identification of cell populations. These techniques provide the basics for some more advanced types of analyses, including the use of data mining techniques for predictive modelling and biomarker identification, and a recent novel class of techniques for modelling cellular differentiation processes. We focus on conventional flow and mass cytometry data and do not cover in detail other single-cell screening technologies, such as single-cell RNA-sequencing and high-content imaging, although some of the techniques described in this Review could be easily applied to such contexts. By formulating a very practical set of guidelines and making all code available to reproduce the analyses presented in this Review, we make it easier for immunologists to adopt computational flow cytometry in practice. All raw data files are available at http://flowrepository.org/id/FR-FCM-ZZQY, and the source code to reproduce all figures is available at https://github.com/saeyslab/FlowCytometryScripts.

## 5.2 Computational Flow Cytometry on the Rise

The most widespread use of flow cytometry by immunologists is the identification and quantification of cell populations. For conventional flow cytometry, this process is mainly still carried out manually, iteratively plotting cells as two-dimensional scatterplots on a chosen subset of markers, and at each step selecting a subset of cells on which to further focus in the next iteration. This process is called gating and aims to identify cell populations by gradually zooming in on them until a predefined number of marker characteristics is fulfilled. Although many immunologists still rely on manual gating, it is important to realize that such an analysis has several serious drawbacks.

### 5.2.1 Manual analysis is hard to reproduce

Manual analysis of flow cytometry data has been shown to constitute one of the major sources of variability in flow cytometry analyses [10–13]. Manual gating is often hard to reproduce mainly owing to two reasons. The first concerns the order in which pairs of markers are explored. This order often allows for some degree of freedom in the gating process, but might lead to different cells being selected when following an alternative gating strategy. The second reason concerns the shape and boundaries of the gates being used. There is great diversity between researchers regarding the gating strategy, with some experts being stricter with their gates and others being more liberal. Computational flow cytometry techniques can be used to greatly enhance the reproducibility of an analysis, as they are based on sound mathematical principles to define cluster boundaries, and the similarities between cells are directly assessed on all markers simultaneously, compared to the approximated way similarity is assessed in manual gating using subsequent scatterplots on pairs of markers. By sharing raw data, as well as analysis workflows with detailed parameter specifications, reproducibility in the broad immunology community can thus be substantially improved, allowing others to repeat and validate analyses.

### 5.2.2 Manual analysis is subjective and biased

When analysing flow cytometry data by hand, experts are typically only looking at their favourite cell types, and in doing so many other cells are discarded from the analysis [14]. In addition, strong assumptions are made regarding cells being negative or positive for a certain marker at an intermediate gating step, and these markers are often not reconsidered at later stages. As mentioned earlier, subjectivity also occurs both at the level of choosing the order in which to consider parameter combinations, as well as in the shape and boundary of each gate specified in the analysis. Computational flow cytometry techniques are more objective and unbiased towards known cell types. By taking a data-driven approach, they assess similarity on all the cells in the sample, considering all markers simultaneously. This enables the finding of novel populations or biomarkers that would be simply discarded in a manual analysis.

### 5.2.3 Manual analysis is inefficient when exploring large marker panels

Humans are not efficient in exploring data spaces described by more than four dimensions, and the current way of iteratively analysing subspaces defined by two markers only provides a very limited view of the data, obfuscating many other patterns being present in the data. This is especially difficult when dealing with mass cytometry data, for which data is described by more than 30 markers, and manual analysis is no longer feasible. Again, computational techniques can help. By using computational approaches that combine all marker information at the same time, automated gating and visualization techniques can highlight the main patterns in the data. These patterns need subsequent interpretation and validation, allowing for an iterative and interactive process that allows immunologists to interpret high-dimensional flow cytometry data in a more efficient way.

### 5.2.4 Manual analysis is time-consuming for large experiments

When analysing large datasets (containing, for example, different stainings for hundreds or thousands of patients), manual analysis often becomes very time-consuming and sometimes infeasible. Even if a fixed gating strategy is used for a large-scale study, it would still require looping over all populations of interest, making small adjustments to the gates, a process that would have to be repeated for every tube of every donor. In addition, such large-scale experiments are often confronted with many sources of variation, including laboratory and instrument effects, different batches of reagents used at different times and batch effects resulting from data being collected at different time points. Computational flow cytometry techniques can be used to render such an analysis more efficient. First, they can be used to automatically perform data pre-processing and quality control, one of the most important steps in large-scale analyses, allowing the detection of potential problems or biases in the study. Second, automated analyses can be performed to identify, for example, novel biomarkers. Tables 5.1 and 5.2 give an overview of the main steps that can be identified when applying computational flow cytometry tools, highlighting important practical guidelines and tools that can be used to assist in the different steps in the process. We discuss each of these steps in more detail below.

## 5.3 Standardization and Reproducibility

Unlike other fields such as genomics and transcriptomics, for which there are standard procedures to analyse data and deposit it in public repositories, standardized data deposition and analysis in the field of cytometry is still in its infancy. Nevertheless, there is a great need for both sharing of raw data, as well as analysis workflows to enable reproducible research in immunology. Computational approaches towards flow cytometry depend on highly standardized data formats and workflows, and since 1984 the Flow Cytometry Standard (FCS) file format has emerged as a standard for the exchange of raw data [15]. Since 2010, the International Society for Advancement of Cytometry (ISAC) has developed the Archival Cytometry Standard (ACS) file format as a container that stores all files relevant to an experiment, including support for audit trails, versioning and digital signatures [16]. This enables researchers to share their experimental data and analysis in a self-contained way, making it easier to check the reproducibility of their results. To exchange analysis reports, the GatingML [17] and Classification Results (CLR) [18] formats were recently developed. Similar standardization efforts have emerged for novel advances, such as ICEFormat, a standardized format for imaging cytometry experiments [19].

### 5.3.1 Towards Reproducible Flow Cytometry

Reproducible flow cytometry starts at the sample preparation stage, with clearly defined standard operating procedures (SOPs) for sample handling and staining, followed by an appropriate setup of the flow cytometer [20]. The use of setup beads, a proper panel design and standardized nomenclature for markers and channels [21] all contribute to increasing the quality and reproducibility of flow cytometry experiments. Appropriate controls (for example, biological controls, compensation controls and fluorescence minus one (FMO) controls [22]) are also crucial for obtaining high quality data. Subsequently, samples can be acquired and subject to data pre-processing and quality control, and this is the first stage at which computational methods enter the workflow. Finally, automated approaches such

***Table 5.1:*** *Steps for analysing flow cytometry data using computational flow cytometry tools. Continued in Table 5.2*

| GENERATE THE DATA |
|---|
| • Consult a statistician or bioinformatician to help in experimental design, sample size estimation and guidelines regarding the type of data analysis required for your research question<br>• Make sure your sample size is big enough; account for the loss of a small percentage of samples owing to quality control and make sure it does not affect your statistical power<br>• Perform your experiment and sample staining using standardized procedures and, if possible, using standardized panels<br>• Use appropriate machine calibration<br>• Include controls for compensation (for example, fluorescence minus one controls) and normalization (for example, beads)<br>• Deposit your data in flowRepository, to enable easy sharing |
| CLEAN THE DATA<br>(using flowCore, flowQ, flowClean, flowStats, openCyto or flowDensity) |
| • Compensate and transform the data; check for any strange compensation artefacts<br>• Plot the scatter and marker values over time; filter out any regions with abnormal behaviour<br>• Compare samples to detect batch effects and normalize the data if necessary<br>• Correct for internal (for example, smoking status) or external confounding (for example, laboratory) effects<br>• Remove debris, dead cells and doublets<br>• Check boundary effects<br>• Consider pre-gating on a cell population of interest |

***Table 5.2:*** *Steps for analysing flow cytometry data using computational flow cytometry tools. Continued from Table 5.1*

| |
|---|
| VISUALIZE THE DATA<br>(using SPADE, viSNE, FlowSOM, PhenoGraph or Scaffold maps) |
| • Get an overview of the cells present in the samples<br>• If you also do a manual gating, map it to the alternative visualizations to see whether there are any problems with either the manual gating or the parameters of the visualization<br>• It might be useful to try different techniques to establish which one most clearly represents your data<br>• Look for any unexpected populations that might contain information you would not examine otherwise |
| ANALYSE THE DATA<br>(see Table 5.3) |
| • Choose the type of analysis to be performed: automated population identification, biomarker discovery, a predictive model or cell development modelling<br>• Run the technique; if the algorithm is stochastic, it might be useful to repeat the algorithm several times to check the stability of the results<br>• Visualize the results and perform quality control on all steps (for example, check whether automated splits between high and low populations make sense)<br>• Adapt parameters of the algorithm where necessary |

as population identification or biomarker selection can be applied to perform data analysis. When the analysis is finished, both raw data and the analysis workflow can be shared.

As researchers in immunology are well accustomed to depositing their expression data online in databases such as Gene Expression Omnibus (GEO) or Array-Express, we highly recommend extending this good habit to cytometry data sets in the future. Two initiatives currently support the deposition of flow cytometry data. FlowRepository is a public repository that supports deposition, querying and downloading of data that is annotated according to the MIFlowCyt (Minimal Information about a Flow Cytometry experiment) [23] compliance format. MI-FlowCyt facilitates metadata storage in a similar way as the well-known MIAME (Minimum Information About a Microarray Experiment) standard for microarray data. Cytobank is a commercial solution for storage, visualization and analysis of cytometry data, and the basic solution allows sharing data at no cost. FlowRepository has been recommended by the ISAC and by many leading journals as the preferred repository for MIFlowCyt-compliant data.

### 5.3.2 The Importance of Standardized Panels

To fully harness the power of computational flow cytometry, immunologists should focus efforts towards the creation of more standardized marker panels. This will facilitate a better integration of data within and between laboratories and also lead to more reproducible flow cytometry experiments. A good example of the recent interest in designing standardized marker panels has been put forward by the EuroFlow consortium, in which a set of panels to screen for leukaemic disorders was agreed in a community-wide effort [24]. A reference dataset is constructed and new samples, analysed with the same panels, can be mapped onto this reference to classify a patient. Also in larger community-wide efforts, such as the Human Immunology Project Consortium (HIPC) [25] or the Milieu Interieur project [26], the design of standardized panels is of key importance. To facilitate sharing and dissemination of optimized panels, the cytometry community has recently introduced a new series of protocols to improve reproducibility, referred to as optimized multicolour immunofluorescence panels (OMIPs) [27].

### 5.3.3 Algorithmic benchmarking and software availability

Despite its recent emergence as a sub-discipline of computational biology, researchers in flow cytometry bioinformatics have quickly realized the need for objective benchmarks and challenges to move the field forward, resulting in the Flow-CAP (Flow Cytometry: Critical Assessment of Population identification methods) initiative [28, 29]. This initiative serves as a forum to bring together researchers working in flow cytometry bioinformatics and to stimulate novel research directions. FlowCAP not only provides a direct comparison of existing methods, but also a series of public benchmark data sets. For the types of methods already surveyed by FlowCAP, such as automated population identification, this enables easy, direct comparison to the state of the art. To stimulate application, evaluation and extension of recent advances in computational flow cytometry, software developers should make their source code and data available to the broad community. Many flow cytometry tools are available as open source packages for R, which is an open-source environment for statistical analysis, computation and visualization. Within the R environment, Bioconductor [30] represents a group of packages for bioinformatics analysis, including many tools for computational flow cytometry. These packages allow those with a basic level of programming expertise to combine flow cytometry data analysis with more general support for data mining and visualization. Alternatively, code can be made available on GitHub, a platform for code sharing, but without the documentation and testing that are obligatory for Bioconductor packages. Several platforms also offer graphical user interfaces that do not require any programming expertise to start exploring computational flow cytometry tools. These include the freely available platforms GenePattern [31] and Cyt (Matlab), as well as the commercial solutions offered by Cytobank and Gemstone (Verity Software House).

## 5.4 Data Pre-processing and Quality Control

Starting with the raw data files, the first step is to compensate the data for spectral overlap, checking for potential artefacts that arise from improper compensation [32]. Subsequently, data is transformed using an appropriate scaling (for example, biexponential or arcsine) and quality control is performed. When using a Bioconductor pipeline, the flowCore package [33] can be used for these first steps. A first quality control step is to visualize scatter and marker values over time, and filter out regions that show abnormal behaviour (for example, due to clogging, speed change or air measurements when the tube is empty). Both the flowQ [34] and flowClean [35] packages provide this functionality. A second control step is to check for batch effects, which gives an idea of the between-sample variation. If batch effects are present, a normalization step is required (for example using the per-channel basis normalization provided in the flowStats package [36]). Finally, samples that do not pass quality control should be removed.

For small numbers of samples, a careful inspection of all artefacts is possible, but larger studies (for example, a cohort of a few hundred patients) require a fully automated quality control pipeline. In such a case, it is nevertheless advisable to randomly select some samples and manually check for artefacts to get an idea of the specific dataset characteristics, noise and artefacts, as quality control is even more important for subsequent automatic analysis than for manual analysis. Following quality control, samples will be further pre-processed by removing debris, dead cells, doublets, either manually or automatically (using for example the openCyto [37] or flowDensity [38] packages to approximate a manual gating). Also boundary effects (data points at the margin of the data range) should be removed, and possibly a pre-gating step on a specific population of interest for further analysis can be performed (for example, using the flowQ [34] and flowStats [36] packages). Several of these Bioconductor tools are also available through the graphical interface of the GenePattern flow cytometry suite, and a more extensive list of pre-processing software can be found in Kvistborg et al. [39].

## 5.5 Visualization of Cytometry Data

Following pre-processing and quality control, it is a good idea to inspect the data by eye, and for this purpose many visualization techniques have been developed as alternatives to the traditional, two-dimensional scatterplots [40, 41] (Table 5.3). Many of these visualization tools apply dimensionality reduction techniques to represent the original, high-dimensional data into a two-dimensional space that gives a better overview of the data. This allows the discovery of hidden population structures in the data, but also inherently presents a simplification of the data set, as not all details that are present in the original, high-dimensional space, can be preserved in the lower-dimensional projection. Other techniques use clustering to first group all data points into clusters (cell types), and subsequently visualize these clusters in a two-dimensional representation. Clustering methods belong to a class of data-mining techniques that are called unsupervised learning techniques (Figure 5.1). These methods aim to find, in an automated way, groups of similar objects, assigning cells with similar marker profiles to similar clusters, which can subsequently be interpreted as cell types.

Both approaches to visualization can also be combined, for example by first running a dimensionality reduction and using this as input for a clustering algorithm [42]. As cytometry datasets contain thousands to millions of cells, many visualization techniques often resort to downsampling, thereby using only a subset of all cells, to keep running times acceptable. Alongside these methods, new algorithms are also being developed to highlight other patterns in flow cytometry data. DREMIDREVI (density resampled estimate of mutual information-density rescaled visualization) [43] visualizes dependencies between markers, whereas Wanderlust [44] visualizes developmental changes in the cells, discussed further below.

***Table 5.3:*** *Overview of visualization techniques. Abbreviations used: DREMIDREVI, density resampled estimate of mutual information-density rescaled visualization; FlowSOM, flow cytometry data analysis using self-organizing maps; SPADE, spanning tree progression of density normalized events; t-SNE, t-stochastic neighbour embedding.*

| TECHNIQUE | AVAILABILITY | METHOD |
|---|---|---|
| SPADE | Cytobank, Matlab and Bioconductor | The cells are clustered hierarchically and shown in a minimal spanning tree; the number of clusters needs to be provided. |
| FlowMAP | No code available | Similar to SPADE, but uses a stronger connected graph structure instead of a tree, which makes it more robust. |
| FlowSOM | Bioconductor | Similar to SPADE, but using a self-organizing map instead of hierarchical clustering, which makes it faster; the dimensions of the grid to be used need to be provided. |
| viSNE | Cytobank and Matlab | No clustering; all cells are plotted on the basis of the t-SNE dimensionality reduction technique. |
| PhenoGraph | Matlab | No clustering; all cells are plotted in a graph structure. |
| Scaffold map | R package | The cells are clustered and shown in a graph structure, with additional manually labelled landmarks. The library provides a graphical user interface that can be started with one line of code, but this only allows viewing of their data, not mapping one's own. |
| DREMI-DREVI | Matlab | Does not plot the individual cells or clusters but offers an alternative view on marker dependencies. |

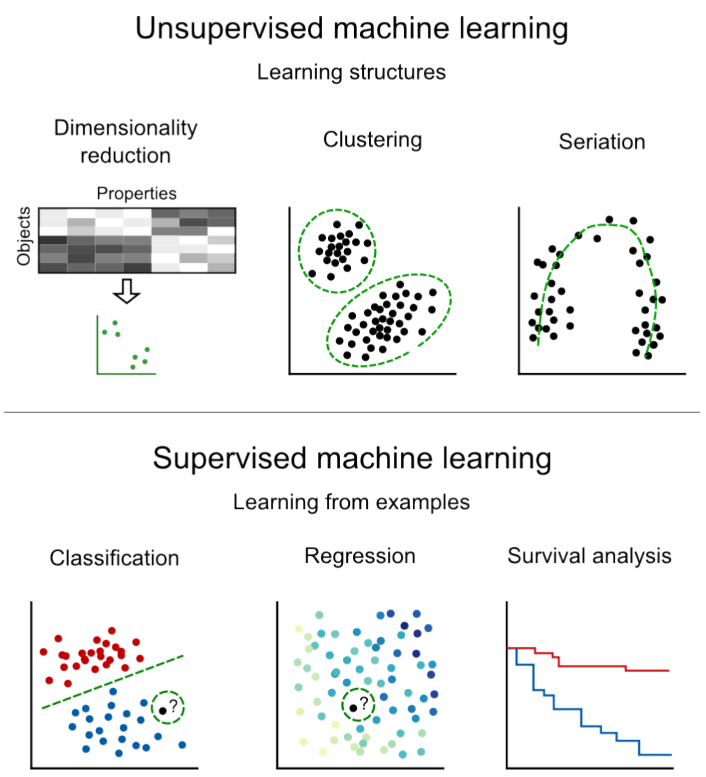


***Figure 5.1:*** *Machine learning techniques are a class of data mining techniques that automatically learn a model from examples. The ultimate goal of machine learning techniques is to produce models that make good generalizations beyond the example data [45]. Two major classes of models can be distinguished: unsupervised learning and supervised learning (see the figure). Unsupervised learning methods aim to recover a hidden structure from unlabelled data. In this case, only descriptive variables for the data points are known. The most well-known type of unsupervised learning is clustering analysis, in which data points are grouped into distinct or overlapping clusters of similar points. Dimensionality reduction techniques are another class of unsupervised learning technique. They project high-dimensional datasets to a lower-dimensional (often two-dimensional) space, aiming to preserve the main structure present in the data. Seriation methods aim to find an optimal ordering of data points, ensuring a smooth transition of properties. Supervised learning methods construct a model to learn the mapping between the variables describing the data and an externally provided variable (denoted using different colours of the data points in the figure). Depending on the nature of the external variable, three types of approaches can be distinguished. Classification methods aim to learn a relationship between the descriptive variables and a finite set of discrete class labels. A common application is to distinguish between different phenotypic variables (for example, healthy versus diseased patients). Regression models learn a mapping between the descriptive variables and a continuous variable (as is used, for example, when modelling dose-response curves). Finally, survival analysis refers to a group of methods for analysing data in which the outcome variable is the time until the occurrence of an event of interest (for example, the occurrence of a disease or death)*

### 5.5.1 Methods Based on Dimensionality Reduction Techniques

Dimensionality reduction techniques aim to preserve the main structure in the data while reducing a high-dimensional data description to a lower-dimensional projection. Traditional dimensionality reduction techniques, such as principal component analysis (PCA), project the data such that the new coordinate system best preserves the variance in the data [46]. More recent dimensionality reduction techniques, including t-stochastic neighbour embedding (t-SNE) [47], apply another principle, aiming to find a lower-dimensional projection that best preserves the similarity in the original, high-dimensional space. This method has been shown to work very well with flow and mass cytometry data; it is used by the viSNE [48] technique, after subsampling the data, and by One-SENSE [49] in combination with heatmaps to visualize marker expression in cell subsets. Graph-based approaches are also used, connecting similar cells in a graph and subsequently applying specific graph layout algorithms. For example, PhenoGraph [50] builds a graph with weighted edges based on the number of nearest neighbours shared by the cells, which is displayed using a force-directed algorithm.

### 5.5.2 Clustering Based Techniques

Whereas dimensionality reduction techniques aim to visualize all data points in a lower-dimensional space, clustering based techniques first group cells into cell type clusters in the original, high-dimensional space and subsequently use visualization algorithms to represent these cell type clusters in a lower-dimensional space. One of the most popular clustering based techniques to visualize flow and mass cytometry data is the SPADE (spanning tree progression of density normalized events) algorithm [51]. SPADE starts by applying a density-dependent downsampling step to the original dataset to obtain a subset of cells, and applies hierarchical clustering to group these cells into cell type clusters. These cell type clusters are subsequently visualized by using a minimal spanning tree (MST) layout algorithm that highlights the relationships between the most closely related cell type clusters. This approach can be extended to other visualization methods, such as the density-based graph structure used by FlowMap [52]. Another clustering based technique to visualize cytometry data is the FlowSOM (flow cytometry data analysis using self-organizing maps) algorithm [53]. FlowSOM uses self-organizing maps (SOM) to simultaneously cluster and visualize cytometry data in a two-dimensional grid of cell type clusters. In a subsequent step, either an MST or t-SNE-based algorithm can be used to better highlight the relationships between the cell type clusters. A comparative evaluation of three visualization techniques (SPADE, t-SNE and FlowSOM) is presented in Figure 5.2. When using these algorithms, results are interpreted by visualizing the marker values, an approach illustrated in detail in Figure 5.3. The scaffold map algorithm [54], however, uses the CLARA (clustering for large applications) algorithm to cluster the cells and then uses a force-directed algorithm to show a weighted graph, in which the edge weights represent the cosine similarity between the median marker values. Next to the clusters, scaffold maps also include manually gated populations as landmarks on the graph (Figure 5.3), facilitating easier interpretation.

### 5.5.3 Strengths and Weaknesses of the Various Approaches

Whereas visualization techniques based on dimensionality reduction aim to plot each cell as a data point, clustering based approaches aggregate information over many cells, thus showing an average picture for each cluster (cell type). Dimensionality reduction based techniques can thus visualize data at the single cell level but, to keep running times acceptable, it is often necessary to select only a subsample of the data. In addition, clustering based techniques already provide groups of cells that can be interpreted as cell types, whereas methods based on dimensionality reduction still need a subsequent clustering step to assign each cell to a cell type.

An important aspect of almost all flow cytometry visualization techniques concerns the stochastic nature of the methods. To keep running times acceptable, approximative methods that use an element of randomness can be used. As a result, running the same visualization algorithm on the same dataset might produce different results in each run. As a user of these methods, it is therefore important to run an algorithm multiple times on the same dataset to get an idea of the variability as well as identify common trends over all runs. Ideally, support for running these algorithms several times and aggregating the results should be built into the algorithm, which is an important challenge remaining for algorithm developers.

The algorithmic properties underlying both types of visualization methods strongly influence their interpretation, which is an important aspect for immunologists to realize. Methods based on t-SNE project data onto a lower-dimensional space, aiming to preserve cell similarities from the original, high-dimensional space. As a result, cells that are similar in the original space will be close in the two-dimensional map, but the converse is not necessarily true: cells close in the two-dimensional map may not necessarily be similar in the original marker space. A similar word of caution applies to the MSTs that are used by SPADE and FlowSOM. These spanning trees cannot be directly interpreted as developmental hierarchies. Although cells that are close in the tree will be similar, it is not necessarily the case that cells that are similar end up close by in the tree. As an MST does not allow for cyclic paths, such paths will be artificially split, resulting in separate branches that may end up in a totally different part of the visualization.

Finally, it should be noted that the computational complexity, and thus the running time, differs greatly between algorithms. While t-SNE based methods and

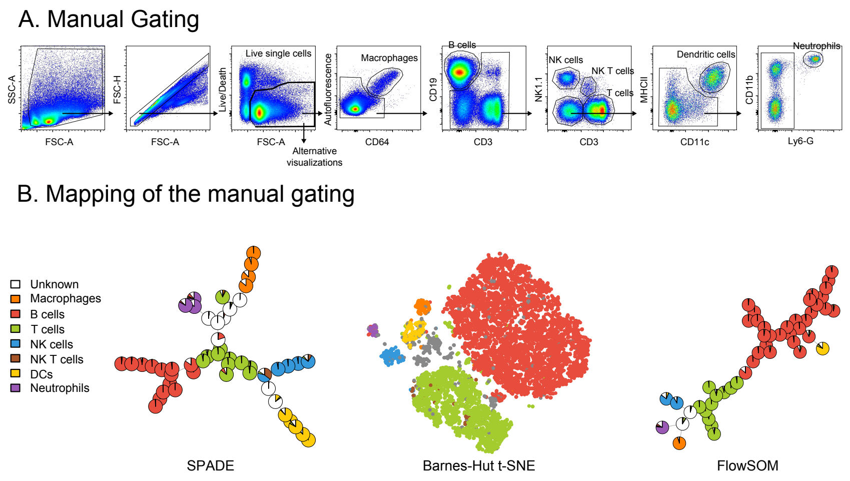


***Figure 5.2:*** *Evaluation of three alternative visualization techniques using a manually gated dataset. Splenocytes of a wild-type C57BL/6J mouse were stained with 11 markers. (a) An example of the traditional, manual gating analysis used to obtain populations of the most common immune cells is shown. The subset of these live, single cells was used as input for three different visualization algorithms. (b) The populations from the manual gating are mapped onto the different representations used by SPADE (spanning tree progression of density normalized events), t-SNE (t-stochastic neighbour embedding)-based methods and FlowSOM (flow cytometry data analysis using self-organizing maps). Colours correspond to the manual gating results and show that most immune cell types are clearly distinguished in all three visualization methods. Only very rare cell types, such as natural killer T (NKT) cells (which constitute only 0.3% of the dataset), are not distinguished, whereas rare cell types such as neutrophils (which constitute 0.7% of the dataset) are distinguished by all three methods. In the SPADE and FlowSOM maps, circles represent cell type clusters, and pie charts indicate the number of each manually defined population present in the automatically determined cell type clusters. In the t-SNE-based maps, each dot represents a single cell, coloured according to the manual gating. Cells that are outgated in the manual analysis appear either as white pie charts (for SPADE and FlowSOM) or as grey cells (for t-SNE). All raw data files are available at http://flowrepository.org/id/FR-FCM-ZZQY and source code to reproduce all figures is available at https://github.com/saeyslab/FlowCytometryScripts. DC, dendritic cell; FSC-A, forward scatter area; FSC-H, forward scatter height; SSC-A, side scatter area.*

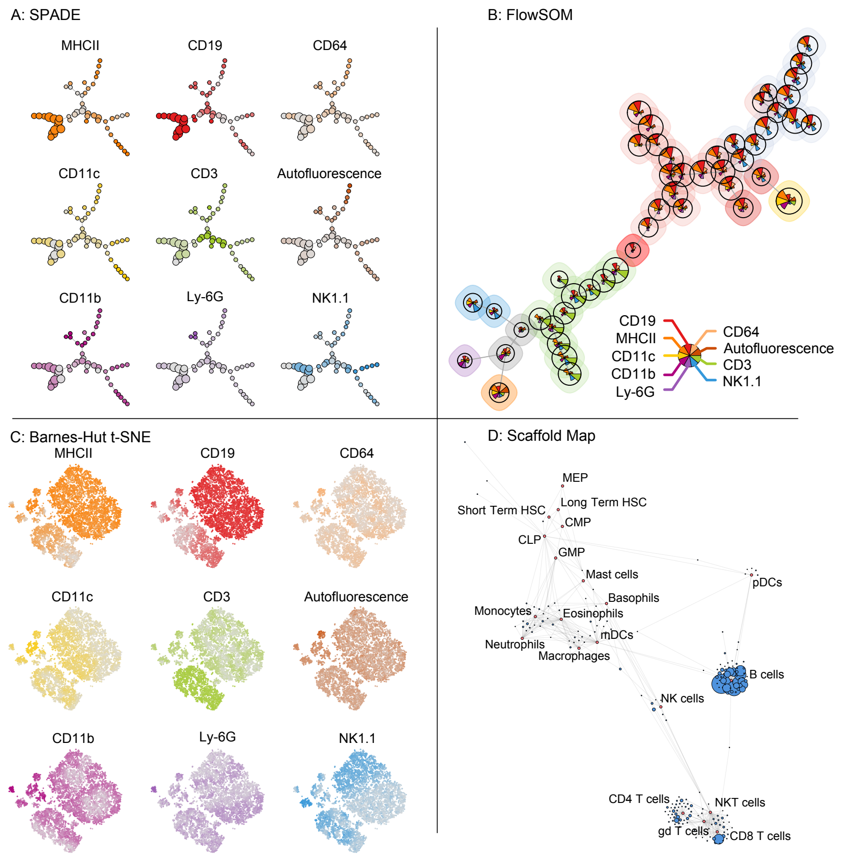


***Figure 5.3:*** *Marker visualization of mouse splenocytes. (ac) Visualization of mouse splenocytes using SPADE, FlowSOM and t-SNE. SPADE uses density-based downsampling and hierarchical clustering to group similar cells, which are visualized in a minimal spanning tree. FlowSOM also uses a minimal spanning tree but does not use subsampling, and it clusters the cells using a self-organizing map. By contrast, methods based on t-SNE (such as viSNE and ACCENSE) do not cluster the cells but show each cell individually in two new dimensions that take similarities in all the original dimensions into account. For SPADE and t-SNE, a subplot is shown for each individual marker, in which the colour is more saturated for higher expression levels. By comparing the different subplots, the cell type can be determined. FlowSOM uses pie charts, combining all markers in a single plot. The height of each part indicates the expression level. Owing to the density based subsampling, SPADE analysis will even out the distribution of the different cell types. Although FlowSOM does not do this, it is still able to distinguish populations as small as 0.7% (such as neutrophils (which are CD11b+LY6G+) in this dataset), while at the same time running almost two orders of magnitude faster (9 seconds versus 700 seconds on a single-threaded processor). FlowSOM also offers additional visualization options, such as the original self-organizing map grid or a t-SNE mapping of the nodes. All cells were used by SPADE and FlowSOM, but owing to computational limitations only 10,000 cells were processed using t-SNE. (d) Visualization of scaffold maps for the mouse immune reference data set from [54].*

SPADE are typically only able to process a few tens of thousands of cells, methods such as FlowSOM scale to much larger datasets than the other visualization algorithms, allowing the algorithm to efficiently process millions of cells.

## 5.6 Automated Population Identification

To overcome the limitations of manual gating, several techniques for automated population identification have been developed during the past decade. Technically, the problem of assigning each cell to a cell type can be described as a clustering problem. As there are many different clustering techniques, it comes as no surprise that there are equally as many approaches to automated gating.

An important aspect of clustering techniques in general is determining the number of clusters (that is, cell types in cytometry) that is anticipated to be present in the data. Whereas some methods explicitly need such a parameter being set by the user, others try to identify the number of cell types in an automated way by trying out different options or have other parameters that influence the number of clusters. In general, it is advisable to set the number of anticipated clusters higher than the anticipated number of cell types – that is, perform over-clustering – to ensure the relevant cell types can be found even if an expected population is split in subpopulations.

### 5.6.1 Automated Gating Techniques

Automated gating techniques can be categorized on the basis of the clustering principle they are building upon. Model based techniques, such as FLAME [45], flowClust [55], flowMerge [56], flowGM [57], immunoClust [58] and SWIFT [59], assume each cell type can be modelled as a multivariate statistical distribution and then focus on fitting these distributions as well as possible to the data. Other methods, such as flowMeans [60], use an approach that is based on cluster representatives, such as the popular K-means clustering algorithm. Alternative approaches that use spectral clustering techniques (for example, SamSPECTRAL [61]) rely on density-based clustering techniques (FLOCK [62] and flowPeaks [63]) or on Bayesian methods (BayesFlow [64]). Although all these techniques are typically applied to high-dimensional data, it is also possible to apply these techniques to reduced data, such as that generated by the viSNE algorithm, for example. This approach is used by ACCENSE [65] and DensVM [66].

Table 5.4 shows an overview of the most commonly used automated gating techniques, describing their major characteristics and a runtime estimate of these methods applied to the data described in Figure 5.2. We also indicate which methods provide a graphical user interface, for easy use by scientists with limited programming experience, and whether the method automatically identifies the number of clusters or whether this should be given as input. The running times differ strongly between the algorithms, as their computational complexity is very diverse. While the clustering results also differ, it is hard to score these objectively. A systematic comparison of the performance of automated gating techniques can be found in Aghaeepour et al. [28].

Automated gating has several advantages compared with manual gating. Computational approaches are more objective; they look at all cells in the sample and are not biased towards known cell types. This allows them to identify novel populations that may or may not correspond to known cell types. Their results are also easier to reproduce and they better scale with the increasing dimensionality of the data (that is, size of the marker panels) while still combining all marker information simultaneously. In addition, they can be fully automated to process experiments consisting of a large number of samples.

### 5.6.2 Challenges for Automated Gating

Automated gating techniques have been shown to greatly overcome some of the major limitations of manual analysis, but several challenges remain. A first challenge is the consistent labelling of cell populations across multiple samples in the presence of biological variability and heterogeneity. If the variability is small, cells from different samples can be clustered as one dataset. However, when the variability increases, it becomes necessary to map corresponding clusters from different samples. While some techniques, such as FLAME [45], HDPGMM [67] and ASPIRE [68], already incorporate such a mapping, many other techniques do not. The FlowMapFriedmanRafsky [69] algorithm was developed as a post-processing step to fill this gap, but mapping cell types across samples still remains a challenging issue. Again here standardization will be crucial, especially when comparing data between multiple centres or over time.

A second challenge regards the fact that many current tools are not yet able to deal with appropriate negative staining controls, such as FMO controls. This may result in negative populations being split further into negative and very negative populations owing purely to data spread. Methods based on one-class classifiers have been explored to include FMOs as reference controls [70], and both the flowDensity [38] and openCyto [37] frameworks allow dealing with FMO controls.

A third challenge concerns the identification of very rare cell types, which are easily mistaken for noise by many clustering algorithms. To identify all relevant populations, it might be necessary to do an exhaustive gating, resulting in a strong over-clustering, and then select only those features related to a phenotype (an approach detailed in the next section). With the traditional clustering algorithms, it is recommended to ensure that only relevant markers are used for the clustering. Markers that vary little or that indicate properties not relevant for cell type

***Table 5.4:*** *Overview of automated gating techniques. Autodetection refers to whether the number of clusters is automatically detected. In the case of No, the number of clusters was set explicitly to 10. All R code was run on a DELL Latitude e5530 laptop with 8 GB of RAM and 4 cores. The Python code is executed on a laptop with 32 GB of RAM and 8 cores. ‡ Graphical user interface. § Implementation used for timings.*

| | AVAILABILITY | METHOD | AUTO-DETECTION? | TIME TO RUN 322,890 cells | TIME TO RUN 10,000 cells |
|---|---|---|---|---|---|
| FLAME | GenePattern‡§ | Model-based clustering using skewed t-distributions on the individual samples, with an additional step to model mapping between samples. Needs a given number of clusters but can try a range of values and automatically determines the best. | No | 7 h | 10 min |
| FLOCK | GenePattern‡, Immport ‡, C code§ | Clustering based on centroids derived from density estimation in a hypergrid. | Yes | 2 min | <1 sec |
| ACCENSE | Stand alone executable‡§ | Clustering on t-SNE result, has both representative based (K-means) and density based (DB-SCAN) options. | Yes | Not possible | 2 min |
| flowClust and flowMerge | GenePattern‡, Bioconductor§ | Clustering based on t-mixture model. flowClust can use either a specified number of clusters or several options to try. flowMerge picks the best possible merge option, without specifying the expected number of clusters. | No | 22 min | 30 sec |
| flowMeans | GenePattern‡, Bioconductor§ | Clustering based on the K-means algorithm, which is representative-based. Either needs an explicit number of clusters or a maximum number. | No | 3 min | 5 sec |
| Sam-SPECTRAL | GenePattern‡, Bioconductor§ | Graph-based clustering. Some extra parameters need to be tuned for optimal results, which will influence the number of clusters. | Yes | 2 h | 15 sec |
| immuno-Clust | Bioconductor§ | Model based clustering of the individual samples, with an additional step to model a mapping between samples. | Yes | 1 h | 1 min |
| flowPeaks | Bioconductor§ | Uses density-based peak finding combined with K-means clustering. | Yes | 13 sec | <1 sec |
| FlowSOM meta | Bioconductor§ | Consensus clustering of the FlowSOM clusters (two step clustering). | No | 15 sec | 4 sec |
| HDPGMM | Python library§ | Generates an aligned data model by analysing several samples simultaneously. Uses a parallelized implementation for quick computations. | No | 12 min | 30 sec |
| SWIFT | Matlab‡§ | Clustering based on splitting and merging of Gaussian mixture models. Specifically developed to identify rare populations. | Yes | 2 h | 6 min |
| ASPIRE | Matlab and C++ § | Similar to HDPGMM but optimized for mapping samples with higher technical or biological variations. | Yes | >4 h | 2 min |

identification (for example, activation markers) are best left out, as these will only contribute noise to the similarity calculation. An adapted distance measure, which assigns different importance scores to different markers, can in some cases also be helpful.

## 5.7 Biomarker Identification

The automated identification of cell populations that are predictive of a phenotypic outcome makes computational flow cytometry techniques an ideal tool for biomarker discovery. These biomarkers can serve various purposes, including diagnosis, prognosis or stratification of patient cohorts, patient monitoring to study treatment response, drug or vaccine safety and efficacy studies, as well as risk profiling. As the increasing size of marker panels leads to a combinatorial increase in the number of potential cell types that can be measured in a sample, machine learning-based techniques can be used to identify the most important cell types (biomarkers) that can be used to link the sample to a clinical outcome of interest. To this end, several supervised machine learning techniques (Figure 5.1) can be used. These techniques typically build on automated gating techniques, using information on the automatically identified populations as input variables for a subsequent mathematical model that associates these variables to an outcome of interest. Common applications include: classification, in which different sample groups are compared (for example, healthy versus diseased patients, knockout mice versus wild-type mice); regression, in which an association with a continuous outcome variable is modelled (for example, efficacy of a drug); and survival analysis, in which the time until the occurrence of an event of interest is modelled (for example, time until progression to a disease or death of a patient). When using such biomarker detection techniques in practice, it is crucial to use an appropriate evaluation scheme, typically performing biomarker selection on a training cohort and keeping an independent validation cohort for testing.

Several computational flow cytometry approaches to biomarker discovery have been proposed. A first option is to cluster the data using a population-identification method and subsequently compare the cluster cell counts with statistical tests. Such an approach was used, for example, by Zare et al. [71] to distinguish between mantle cell lymphoma and small lymphocytic lymphoma; SamSPECTRAL was combined with a feature selection algorithm, identifying the ratio of the mean fluorescence intensities between CD20 and CD23 as the most important diagnostic marker. In a similar manner, Bashashati et al. [72] used a combination of flowClust and survival modelling to identify a group of B cells with high side scatter that correlates well with inferior survival in diffuse large B cell lymphoma. This population could thus be used as a biomarker to identify patients at high risk of relapse.

***Table 5.5:*** *Computational techniques for biomarker identification. Abbreviations used: GUI, graphical user interface.*

| TECHNIQUE | AVAILABILITY | METHOD |
|---|---|---|
| FlowMapFR | Bioconductor | Uses the FriedmanRafsky (FR) statistic to map populations between clustering results. Can be combined with most automated gating methods and the results can be used as input for any predictive algorithm |
| Citrus | R package with Shiny GUI | Makes use of hierarchical clustering to extract information about cell populations, which are then used in a regularized regression model to make predictions. The GUI can be started with one line of code |
| flowType / RchyOptimyx | Bioconductor | Makes use of exhaustive splitting in positive and negative populations to extract the information. RchyOptimyx will select the most informative populations for biomarker detection |
| FloReMi | R script (github) | Uses flowType to extract features but filters redundant features and uses a Random Forest as prediction model |
| COMPASS | Bioconductor | Investigates the differences between stimulated and unstimulated samples for all combinatorial functional cell subsets, given a gating result |
| Competitive SWIFT | Matlab | Builds separate SWIFT models for the separate groups and uses competitive assignment of new samples to these models as predictive algorithm |

An alternative option for biomarker identification consists of using dedicated computational pipelines, which use a more exhaustive clustering, followed by a feature selection step to select the most relevant populations (Table 5.5). Several approaches exist, such as flowTypeRchyOptimyx [73] (which use density base clustering), Citrus [74] (which uses hierarchical clustering) or COMPASS [75] (which uses model based clustering). They all strongly over-cluster the data and include clusters on various levels (more general versus very specific cell population definitions) and apply statistical tests on these clusters to select the ones that correlate with the property of interest. Competitive SWIFT [76] uses another approach: it builds separate models for the different groups and uses a competition framework to determine which model fits best for the new sample.

To objectively benchmark the predictive performance of computational flow cytometry models, several challenges organized by the FlowCAP initiative were launched. The FlowCAP II challenge [28] focused on disease classification from cytometry data, with sub-challenges aiming to detect acute myeloid leukaemia, discriminate between two antigen stimulation groups of individuals after HIV vaccination and predict HIV exposure in African infants. Although highly predictive models could be constructed using the first two applications, the last application turned out to be unsuccessful, probably owing to limited biological background knowledge to design an appropriate marker panel. Together, these results show that there is a great need for more benchmark datasets on a wider variety of problems, as many of the current FlowCAP II classification problems are relatively easy to tackle. The FlowCAP IV challenge aimed to predict the time to progression to AIDS among a cohort of HIV positive subjects, using antigen-stimulated peripheral blood mononuclear cell (PBMC) samples [29]. The goal of this challenge was to identify novel cell populations that correlate well with progression and thus could be used as biomarkers. This challenge proved to be quite difficult, with only two algorithms  FloReMi [77] and the pipeline flowDensityflowType-RchyOptimyx  attaining statistically significant predictive values. Moreover, the latter challenge demonstrated some of the intricacies of biologically interpreting some of the new cell populations that were identified as biomarkers by the predictive models.

## 5.8 Cell Development Modelling

Although the conventional analysis of flow cytometry data aims to find well-delineated populations of cells, the single-cell nature of this data allows us to model gradients or transitions between stable cell states as well. This is, however, not possible using the classical (manual) analysis owing to the current high dimensionality of flow cytometry data, and recently a new class of computational techniques has been introduced to model cellular process in a much more continuous way (Figure 5.4). The central idea is to model cellular development processes by following gradients in the data, thereby inferring cell developmental trajectories using unsupervised seriation techniques. Starting from a snapshot cytometry dataset, which is assumed to be a mixture of cells in different stages of a developmental process, seriation algorithms aim to automatically reconstruct the underlying developmental trajectory that cells are following.

The Wanderlust algorithm [44] orders all cells in a sample with regard to a user-defined starting cell, which represents either the most immature or most mature cell in the sample. Subsequently, a K-nearest neighbour graph (KNN graph) is used to model local similarities between cells, and every cell is ordered with respect to the starting cell by its shortest path in the KNN graph. A bootstrap-like procedure is then applied, averaging over slight variations of the KNN graph to find a robust solution. Wanderlust was shown to reconstruct B cell development in the bone marrow in an almost unsupervised way (the starting cell has to be provided by the user), and it can be assumed to be generally applicable to any linear gradual process measured by cytometry data. An example of the output of Wanderlust is shown in Figure 5.4h. Similarly, probability state modelling [78], as implemented in the commercial software Gemstone, has been shown to successfully model cell developmental processes, although it requires more supervision than the Wanderlust algorithm. Probability state modelling has been applied for modelling CD8+

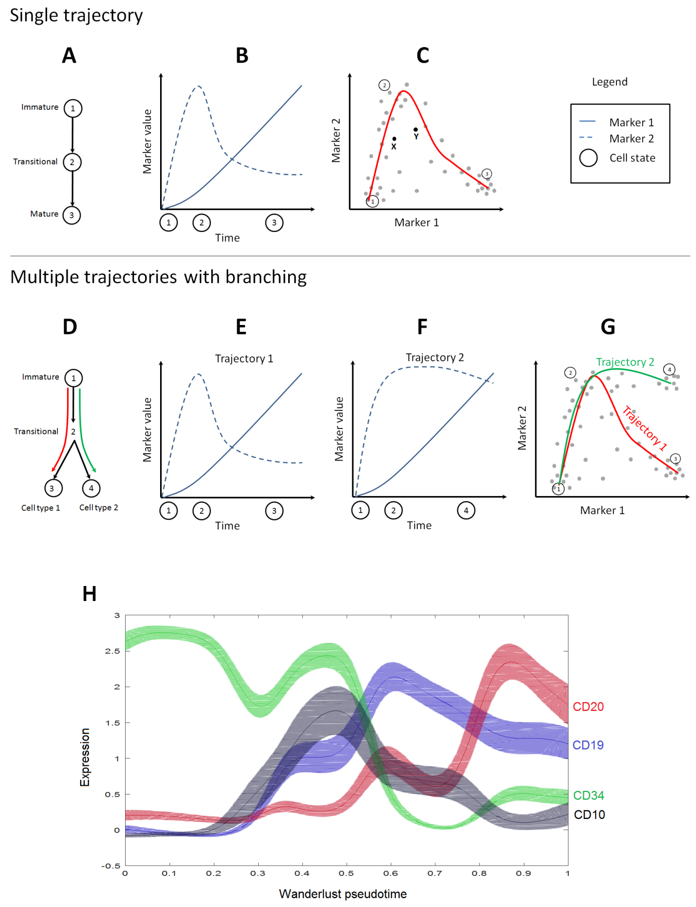


*Figure 5.4: Cell development modelling. Unsupervised seriation techniques can be used to infer pseudo-temporal trajectories that correspond with cellular development processes. The upper part of the figure shows a single trajectory consisting of three states (part a). Cells are starting from an immature state (state 1) and proceed through a transitional state (state 2) to a mature end state (state 3). A simplified dataset characterized by only two markers is shown(parts b, c ). Marker values change with development (part b), and the inferred trajectory (red line) that aims to reconstruct the sequence of cell states from the marker descriptions of the cell is shown (part c). Note that the Euclidean distance between cells X and Y might lead to the misleading interpretation that they are closely related. Cell X might be on the path between state 1 and 2, whereas cell Y might be situated between state 2 and 3. The lower part of the figure shows a more complex, but realistic scenario where two different mature end states originate from a common progenitor (parts dg). In this case, cells follow a common path from the immature state to the transitional state and subsequently branch to result in two end states; this results in two different cell types. Inferring the correct branching topology will thus imply reconstructing two trajectories that share a common path between state 1 and state 2. This situation cannot be dealt with adequately by traditional seriation techniques, as these often only allow to reconstruct a single ordering of all cells. These more complex scenarios therefore call for more advanced modelling techniques, as typically used in the domain of dynamical systems theory. An example of Wanderlust output on a dataset of B cell development [44] (part h), which was produced using the Cyt toolbox.*

T cell differentiation [79], as well as B cell differentiation [80].

When applying these algorithms in practice, it is important to note the following requirements. Wanderlust assumes the developmental process to be gradual, thus requiring all intermediate stages to be present in the sample. Associated to this, it is important that the marker panel is adequately chosen, as leaving out important markers may result in mixing of several stages, and unnecessary markers should be removed, as these may contribute noise. A pre-filtering step may be necessary to focus only on the cells in the developmental process to be modelled, leaving out all other cells in the sample. Otherwise these cells will be forced into the developmental process, obfuscating patterns. Finally, Wanderlust can only model linear (that is, non-branching) trajectories.

To overcome these limitations, recent algorithms, such as SCUBA (single-cell clustering using bifurcation analysis) [81] and Wishbone [82], combine trajectory inference with branch point detection, allowing the generalization of seriation methods to real developmental hierarchies that include branching. Whereas SCUBA combines trajectory inference with bifurcation analysis, Wishbone further extends the approach of Wanderlust. Currently, the development of both linear and branching seriation algorithms is a growing topic in the novel field of single-cell transcriptomics [83–85], but it can be expected that such techniques will become broadly applicable to any high-throughput, single-cell technology.

## 5.9 Conclusions and Future Perspectives

Methods for computational flow and mass cytometry have emerged during the past decade as a novel toolbox to assist immunologists in interpreting their data. These tools have now reached a level of maturity that not only allows bypassing of manual gating, but also opens up new and more unbiased avenues for visualization, interpretation and modelling of cytometry data. Currently, only a limited number of tools are supported by easy-to-use graphical interfaces, such as GenePattern and Cytobank. In addition, many tools are available as open source packages in programming frameworks such as R and MatLab. Together with the growing number of modelling approaches, in particular to better understand cellular differentiation processes, these recent advances thus provide evidence that basic programming and modelling skills will be increasingly required to get the most out of high-dimensional cytometry data in the field of immunology. As cytometric equipment that measures dozens of parameters might soon be available to most research laboratories, it is now time to start training the next generation of flow cytometry users. Reduced costs of flow cytometers that were once too expensive in the clinical setting will ensure that these techniques will hit the clinic sooner rather than later, and clinical biologists will also be faced with the complexity of high-dimensional data sets. We believe that computational flow cytometry also has great potential for improving the early diagnosis and close follow up of patients enrolled in clinical trials or undergoing treatments that perturb the immune and haematopoietic systems.

## Acknowledgment

S.V.G. is funded by the Flanders Agency for Innovation by Science and Technology (IWT). Y.S. is an ISAC Marylou Ingram Scholar. B.N.L. is funded by a European Research Council (ERC) Consolidator grant and several FWO (Research Foundation Flanders) grants.

## References

[1] M. J. Fulwyler. *Electronic separation of biological cells by volume.* Science (New York, N.Y.), 150(3698):910–1, 1965.

[2] J. W. Gray, A. V. Carrano, L. L. Steinmetz, M. A. Van Dilla, D. H. Moore II, B. H. Mayall, and M. L. Mendelsohn. *Chromosome measurement and sorting by flow systems*. Proceedings of the National Academy of Sciences of the United States of America, 72(4):1231–1234, 1975.

[3] J. P. Robinson and M. Roederer. *Flow cytometry strikes gold*. Science, 350(6262):739–740, 2015.

[4] S. P. Perfetto, P. K. Chattopadhyay, and M. Roederer. *Seventeen-colour flow cytometry: unravelling the immune system.* Nature reviews. Immunology, 4(8):648–655, 2004.

[5] P. Chattopadhyay, S. Perfetto, B. Gaylord, A. Stall, L. Duckett, J. Hill, R. Nguyen, D. Ambrozak, R. Balderas, and M. Roederer. *Toward 40+ Parameter Fluorescence Flow Cytometry*. In XXIX Congress of the International Society for Advancement of Cytometry, page 215, 2014.

[6] D. R. Bandura, V. I. Baranov, O. I. Ornatsky, A. Antonov, R. Kinach, X. Lou, S. Pavlov, S. Vorobiev, J. E. Dick, and S. D. Tanner. *Mass cytometry: Technique for real time single cell multitarget immunoassay based on inductively coupled plasma time-of-flight mass spectrometry*. Analytical Chemistry, 81(16):6813–6822, 2009.

[7] J. P. Nolan and D. Condello. *Spectral flow cytometry*. Current Protocols in Cytometry, (SUPPL.63), 2013.

[8] K. E. McGrath, T. P. Bushnell, and J. Palis. *Multispectral imaging of hematopoietic cells: Where flow meets morphology*, 2008.

[9] C. Giesen, H. A. O. Wang, D. Schapiro, N. Zivanovic, A. Jacobs, B. Hattendorf, P. J. Schüffler, D. Grolimund, J. M. Buhmann, S. Brandt, Z. Varga, P. J. Wild, D. Günther, and B. Bodenmiller. *Highly multiplexed imaging of tumor tissues with subcellular resolution by mass cytometry*. Nature Methods, 11(4):417–422, 2014.

[10] L. Nomura, V. C. Maino, and H. T. Maecker. *Standardization and optimization of multiparameter intracellular cytokine staining*. Cytometry Part A, 73(11):984–991, 2008.

[11] H. T. Maecker. *Standardizing immunophenotyping for the Human Immunology*. Nat Rev Immunol, 12(3):191–200, 2012.

[12] G. Pachón, I. Caragol, and J. Petriz. *Subjectivity and flow cytometric variability*. Nature Reviews Immunology, 12(5):1, 2012.

[13] C. Gouttefangeas, C. Chan, S. Attig, T. T. Køllgaard, H. G. Rammensee, S. Stevanović, D. Wernet, P. thor Straten, M. J. P. Welters, C. Ottensmeier, S. H. van der Burg, and C. M. Britten. *Data analysis as a source of variability of the HLA-peptide multimer assay: From manual gating to automated recognition of cell clusters*. Cancer Immunology, Immunotherapy, 64(5):585–598, 2015.

[14] J. M. Irish. *Beyond the age of cellular discovery.* Nature immunology, 15(12):1095–7, 2014.

[15] J. Spidlen, W. Moore, D. Parks, M. Goldberg, C. Bray, P. Bierre, P. Gorombey, B. Hyun, M. Hubbard, S. Lange, R. Lefebvre, R. Leif, D. Novo, L. Ostruszka, A. Treister, J. Wood, R. F. Murphy, M. Roederer, D. Sudar, R. Zigon, and R. R. Brinkman. *Data file standard for flow cytometry, version FCS 3.1*. Cytometry Part A, 77(1):97–100, 2010.

[16] J. Spidlen, P. Shooshtari, T. R. Kollmann, and R. R. Brinkman. *Flow cytometry data standards*. BMC Research Notes, 4:50, 2011.

[17] J. Spidlen, W. Moore, and R. R. Brinkman. *ISAC's Gating-ML 2.0 data exchange standard for gating description*. Cytometry Part A, 87(7):683–687, 2015.

[18] J. Spidlen, C. Bray, and R. R. Brinkman. *ISAC's classification results file format*. Cytometry Part A, 87(1):86–88, 2015.

[19] J. Spidlen and D. Novo. *ICEFormat - the image cytometry experiment format*. Cytometry Part A, 81(12):1015–1018, 2012.

[20] S. Schlickeiser, M. Streitz, and B. Sawitzki. *Standardized multi-color flow cytometry and computational biomarker discovery*. In Methods in Molecular Biology, volume 1371, pages 225–238. 2015.

[21] M. Roederer. *A proposal for unified flow cytometer parameter naming*. Cytometry Part A, 87(8):689–691, 2015.

[22] J. W. Tung, D. R. Parks, W. A. Moore, L. A. Herzenberg, and L. A. Herzenberg. *New approaches to fluorescence compensation and visualization of FACS data*. Clinical Immunology, 110(3):277–283, 2004.

[23] J. A. Lee, J. Spidlen, K. Boyce, J. Cai, N. Crosbie, M. Dalphin, J. Furlong, M. Gasparetto, M. Goldberg, E. M. Goralczyk, B. Hyun, K. Jansen, T. Kollmann, M. Kong, R. Leif, S. McWeeney, T. D. Moloshok, W. Moore,

G. Nolan, J. Nolan, J. Nikolich-Zugich, D. Parrish, B. Purcell, Y. Qian, B. Selvaraj, C. Smith, O. Tchuvatkina, A. Wertheimer, P. Wilkinson, C. Wilson, J. Wood, R. Zigon, R. H. Scheuermann, and R. R. Brinkman. *MIFlow-Cyt: The minimum information about a flow cytometry experiment*. Cytometry Part A, 73(10):926–930, 2008.

[24] J. J. M. van Dongen, L. Lhermitte, S. Böttcher, J. Almeida, V. H. J. van der Velden, J. Flores-Montero, a. Rawstron, V. Asnafi, Q. Lécrevisse, P. Lucio, E. Mejstrikova, T. Szczepański, T. Kalina, R. de Tute, M. Brüggemann, L. Sedek, M. Cullen, a. W. Langerak, a. Mendonça, E. Macintyre, M. Martin-Ayuso, O. Hrusak, M. B. Vidriales, and a. Orfao. *EuroFlow antibody panels for standardized n-dimensional flow cytometric immunophenotyping of normal, reactive and malignant leukocytes*. Leukemia, 26(February):1908–1975, 2012.

[25] G. Finak, M. Langweiler, M. Jaimes, M. Malek, J. Taghiyar, Y. Korin, K. Raddassi, L. Devine, G. Obermoser, M. L. Pekalski, N. Pontikos, A. Diaz, S. Heck, F. Villanova, N. Terrazzini, F. Kern, Y. Qian, R. Stanton, K. Wang, A. Brandes, J. Ramey, N. Aghaeepour, T. Mosmann, R. H. Scheuermann, E. Reed, K. Palucka, V. Pascual, B. B. Blomberg, F. Nestle, R. B. Nussenblatt, R. R. Brinkman, R. Gottardo, H. Maecker, and J. P. McCoy. *Standardizing Flow Cytometry Immunophenotyping Analysis from the Human ImmunoPhenotyping Consortium.* Scientific reports, 6(August 2015):20686, 2016.

[26] M. Hasan, B. Beitz, V. Rouilly, V. Libri, A. Urrutia, D. Duffy, L. Cassard, J. P. Di Santo, E. Mottez, L. Quintana-Murci, M. L. Albert, and L. Rogge. *Semi-automated and standardized cytometric procedures for multi-panel and multi-parametric whole blood immunophenotyping*. Clinical Immunology, 157(2):261–276, 2015.

[27] Y. Mahnke, P. Chattopadhyay, and M. Roederer. *Publication of optimized multicolor immunofluorescence panels*. Cytometry Part A, 77(9):814–818, 2010.

[28] N. Aghaeepour, G. Finak, H. Hoos, T. R. Mosmann, R. Brinkman, R. Gottardo, and R. H. Scheuermann. *Critical assessment of automated flow cytometry data analysis techniques.* Nature methods, 10(3):228–38, 2013.

[29] N. Aghaeepour, P. Chattopadhyay, M. Chikina, T. Dhaene, S. Van Gassen, M. Kursa, B. N. Lambrecht, M. Malek, G. J. Mclachlan, Y. Qian, P. Qiu, Y. Saeys, R. Stanton, D. Tong, C. Vens, S. Walkowiak, K. Wang, G. Finak, R. Gottardo, T. Mosmann, G. P. Nolan, R. H. Scheuermann, and R. R. Brinkman. *A benchmark for evaluation of algorithms for identification of*

*cellular correlates of clinical outcomes*. Cytometry Part A, 89(1):16–21, 2016.

[30] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. H. Yang, and J. Zhang. *Bioconductor: open software development for computational biology and bioinformatics.* Genome biology, 5(10):R80, 2004.

[31] M. Reich, T. Liefeld, J. Gould, J. Lerner, P. Tamayo, and J. P. Mesirov. *GenePattern 2.0.* Nature genetics, 38(5):500–1, 2006.

[32] S. P. Perfetto, D. Ambrozak, R. Nguyen, P. Chattopadhyay, and M. Roederer. *Quality assurance for polychromatic flow cytometry.* Nature protocols, 1(3):1522–30, 2006.

[33] F. Hahne, N. LeMeur, R. R. Brinkman, B. Ellis, P. Haaland, D. Sarkar, J. Spidlen, E. Strain, and R. Gentleman. *flowCore: a Bioconductor package for high throughput flow cytometry.* BMC bioinformatics, 10:106, 2009.

[34] N. Le Meur, A. Rossini, M. Gasparetto, C. Smith, R. Brinkman, and R. Gentleman. *Data Quality Assessment of Ungated Flow Cytometry Data in High Throughput Experiments*. Cytometry Part A, 71(A):393–403, 2007.

[35] K. Fletez-Brant, J. Špidlen, R. R. Brinkman, M. Roederer, and P. K. Chattopadhyay. *flowClean: Automated identification and removal of fluorescence anomalies in flow cytometry data*. Cytometry Part A, 89(5):461–471, 2016.

[36] F. Hahne, A. H. Khodabakhshi, A. Bashashati, C. J. Wong, R. D. Gascoyne, A. P. Weng, V. Seyfert-Margolis, K. Bourcier, A. Asare, T. Lumley, R. Gentleman, and R. R. Brinkman. *Per-channel basis normalization methods for flow cytometry data*. Cytometry Part A, 77(2):121–131, 2010.

[37] G. Finak, J. Frelinger, W. Jiang, E. W. Newell, J. Ramey, M. M. Davis, S. A. Kalams, S. C. De Rosa, and R. Gottardo. *OpenCyto: An Open Source Infrastructure for Scalable, Robust, Reproducible, and Automated, End-to-End Flow Cytometry Data Analysis*. PLoS Computational Biology, 10(8), 2014.

[38] M. Malek, M. J. Taghiyar, L. Chong, G. Finak, R. Gottardo, and R. R. Brinkman. *FlowDensity: Reproducing manual gating of flow cytometry data by automated density-based cell population identification*. Bioinformatics, 31(4):606–607, 2015.

[39] P. Kvistborg, C. Gouttefangeas, N. Aghaeepour, A. Cazaly, P. K. Chattopadhyay, C. Chan, J. Eckl, G. Finak, S. R. Hadrup, H. T. Maecker, et al. *Thinking outside the gate: single-cell assessments in multiple dimensions*. Immunity, 42(4):591, 2015.

[40] C. Chester and H. T. Maecker. *Algorithmic tools for mining high-dimensional cytometry data*. The Journal of Immunology, 195(3):773–779, 2015.

[41] F. Mair, F. J. Hartmann, D. Mrdjen, V. Tosevski, C. Krieg, and B. Becher. *The end of gating? An introduction to automated analysis of high dimensional cytometry data*. European Journal of Immunology, 46(1):34–43, 2016.

[42] K. E. Diggins, P. B. Ferrell, and J. M. Irish. *Methods for discovery and characterization of cell subsets in high dimensional mass cytometry data*. Methods (San Diego, Calif.), 82:55–63, 2015.

[43] S. Krishnaswamy. *Conditional density-based analysis of T cell signaling in single-cell data*. Science, (October):1–20, 2015.

[44] S. C. Bendall, K. L. Davis, E. A. D. Amir, M. D. Tadmor, E. F. Simonds, T. J. Chen, D. K. Shenfeld, G. P. Nolan, and D. Pe'Er. *Single-cell trajectory detection uncovers progression and regulatory coordination in human b cell development*. Cell, 157(3):714–725, 2014.

[45] S. Pyne, X. Hu, K. Wang, E. Rossin, T.-I. T.-I. Lin, L. M. Maier, C. Baecher-Allan, G. J. McLachlan, P. Tamayo, D. A. Hafler, P. L. De Jager, and J. P. Mesirov. *Automated high-dimensional flow cytometric data analysis.* Proceedings of the National Academy of Sciences of the United States of America, 106(21):8519–8524, 2009.

[46] E. Lugli, M. Pinti, M. Nasi, L. Troiano, R. Ferraresi, C. Mussi, G. Salvioli, V. Patsekin, J. P. Robinson, C. Durante, M. Cocchi, and A. Cossarizza. *Subject classification obtained by cluster analysis and principal component analysis applied to flow cytometric data*. Cytometry Part A, 71(5):334–344, 2007.

[47] L. J. P. Van Der Maaten and G. E. Hinton. *Visualizing high-dimensional data using t-sne*. Journal of Machine Learning Research, 9:2579–2605, 2008.

[48] E.-a. D. Amir, K. L. Davis, M. D. Tadmor, E. F. Simonds, J. H. Levine, S. C. Bendall, D. K. Shenfeld, S. Krishnaswamy, G. P. Nolan, and D. Pe'er. *viSNE enables visualization of high dimensional single-cell data and reveals phenotypic heterogeneity of leukemia.* Nature biotechnology, 31(6):545–52, 2013.

[49] Y. Cheng, M. T. Wong, L. van der Maaten, and E. W. Newell. *Categorical Analysis of Human T Cell Heterogeneity with One-Dimensional Soli-Expression by Nonlinear Stochastic Embedding*. J Immunol, 196:924–932, 2016.

[50] J. H. Levine, E. F. Simonds, S. C. Bendall, K. L. Davis, E. A. D. Amir, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, R. Finck, A. L. Gedman, I. Radtke, J. R. Downing, D. Pe'er, and G. P. Nolan. *Data-Driven Phenotypic Dissection of AML Reveals Progenitor-like Cells that Correlate with Prognosis*. Cell, 162(1):184–197, 2015.

[51] 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):886–91, 2011.

[52] E. R. Zunder, E. Lujan, Y. Goltsev, M. Wernig, and G. P. Nolan. *A continuous molecular roadmap to iPSC reprogramming through progression analysis of single-cell mass cytometry*. Cell Stem Cell, 16(3):323–337, 2015.

[53] S. Van Gassen, B. Callebaut, M. J. Van Helden, B. N. Lambrecht, P. Demeester, T. Dhaene, and Y. Saeys. *FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data*. Cytometry Part A, 87(7):636–645, 2015.

[54] M. H. Spitzer, P. F. Gherardini, G. K. Fragiadakis, N. Bhattacharya, R. T. Yuan, A. N. Hotson, R. Finck, Y. Carmi, E. R. Zunder, W. J. Fantl, S. C. Bendall, E. G. Engleman, and G. P. Nolan. *An interactive reference framework for modeling a dynamic immune system*. Science, 349(6244):1259425–1259425, 2015.

[55] K. Lo, R. R. Brinkman, and R. Gottardo. *Automated gating of flow cytometry data via robust model-based clustering*. In Cytometry Part A, volume 73, pages 321–332, 2008.

[56] G. Finak, A. Bashashati, R. Brinkman, and R. Gottardo. *Merging mixture components for cell population identification in flow cytometry.* Advances in bioinformatics, 2009:247646, 2009.

[57] X. Chen, M. Hasan, V. Libri, A. Urrutia, B. Beitz, V. Rouilly, D. Duffy, É. Patin, B. Chalmond, L. Rogge, L. Quintana-Murci, M. L. Albert, and B. Schwikowski. *Automated flow cytometric analysis across large numbers of samples and cell types*. Clinical Immunology, 157(2):249–260, 2015.

[58] T. Sorensen, S. Baumgart, P. Durek, A. Grutzkau, and T. Haupl. *immunoClust-An automated analysis pipeline for the identification of immunophenotypic signatures in high-dimensional cytometric datasets*. Cytometry Part A, 87(7):603–615, 2015.

[59] I. Naim, S. Datta, J. Rebhahn, J. S. Cavenaugh, T. R. Mosmann, and G. Sharma. *SWIFTscalable clustering for automated identification of rare cell populations in large, high-dimensional flow cytometry datasets, Part 1: Algorithm design*. Cytometry Part A, 85(5):408–421, 2014.

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

[61] H. Zare, P. Shooshtari, A. Gupta, and R. R. Brinkman. *Data reduction for spectral clustering to analyze high throughput flow cytometry data*. BMC Bioinformatics, 11(403):1–16, 2010.

[62] Y. Qian, C. Wei, F. Eun-Hyung Lee, J. Campbell, J. Halliley, J. A. Lee, J. Cai, Y. M. Kong, E. Sadat, E. Thomson, et al. *Elucidation of seventeen human peripheral blood B-cell subsets and quantification of the tetanus response using a density-based method for the automated identification of cell populations in multidimensional flow cytometry data*. Cytometry Part B: Clinical Cytometry, 78(S1):S69–S82, 2010.

[63] Y. Ge and S. C. Sealfon. *Flowpeaks: A fast unsupervised clustering for flow cytometry data via K-means and density peak finding*. Bioinformatics, 28(15):2052–2058, 2012.

[64] K. Johnsson, J. Wallin, and M. Fontes. *BayesFlow: latent modeling of flow cytometry cell populations.* BMC bioinformatics, 17(1):25, 2016.

[65] K. Shekhar, P. Brodin, M. M. Davis, and A. K. Chakraborty. *Automatic Classification of Cellular Expression by Nonlinear Stochastic Embedding (ACCENSE).*, volume 111. 2014.

[66] B. Becher, A. Schlitzer, J. Chen, F. Mair, H. R. Sumatoh, K. W. W. Teng, D. Low, C. Ruedl, P. Riccardi-Castagnoli, M. Poidinger, M. Greter, F. Ginhoux, and E. W. Newell. *High-dimensional analysis of the murine myeloid cell system.* Nature immunology, 15(12):1181–9, 2014.

[67] A. Cron, C. Gouttefangeas, J. Frelinger, L. Lin, S. K. Singh, C. M. Britten, M. J. P. Welters, S. H. van der Burg, M. West, and C. Chan. *Hierarchical Modeling for Rare Event Detection and Cell Subset Alignment across Flow Cytometry Samples*. PLoS Computational Biology, 9(7), 2013.

[68] M. Dundar, F. Akova, H. Z. Yerebakan, and B. Rajwa. *A non-parametric Bayesian model for joint cell clustering and cluster matching: identification of anomalous sample phenotypes with random effects.* BMC bioinformatics, 15:314, 2014.

[69] C. Hsiao, M. Liu, R. Stanton, M. Mcgee, Y. Qian, and R. H. Scheuermann. *Mapping cell populations in flow cytometry data for cross-sample comparison using the Friedman-Rafsky test statistic as a distance measure*. Cytometry Part A, 89(1):71–88, 2016.

[70] K. Feher, J. Kirsch, A. Radbruch, H. D. Chang, and T. Kaiser. *Cell population identification using fluorescence-minus-one controls with a one-class classifying algorithm.* Bioinformatics, 30(23):3372–3378, 2014.

[71] H. Zare, A. Bashashati, R. Kridel, N. Aghaeepour, G. Haffari, J. M. Connors, R. D. Gascoyne, A. Gupta, R. R. Brinkman, and A. P. Weng. *Automated analysis of multidimensional flow cytometry data improves diagnostic accuracy between mantle cell lymphoma and small lymphocytic lymphoma*. American Journal of Clinical Pathology, 137(1):75–85, 2012.

[72] A. Bashashati, N. A. Johnson, A. H. Khodabakhshi, M. D. Whiteside, H. Zare, D. W. Scott, K. Lo, R. Gottardo, F. S. L. Brinkman, J. M. Connors, G. W. Slack, R. D. Gascoyne, A. P. Weng, and R. R. Brinkman. *B cells with high side scatter parameter by flow cytometry correlate with inferior survival in diffuse large B-cell lymphoma*. American Journal of Clinical Pathology, 137(5):805–814, 2012.

[73] K. O'Neill, A. Jalali, N. Aghaeepour, H. Hoos, and R. R. Brinkman. *Enhanced flowType/RchyOptimyx: A Bioconductor pipeline for discovery in high-dimensional cytometry data*. Bioinformatics, 30(9):1329–1330, 2014.

[74] R. V. Bruggner, B. Bodenmiller, D. L. Dill, R. J. Tibshirani, and G. P. Nolan. *Automated identification of stratifying signatures in cellular subpopulations.* Proceedings of the National Academy of Sciences of the United States of America, 111(26):E2770–7, 2014.

[75] L. Lin, G. Finak, K. Ushey, C. Seshadri, T. R. Hawn, N. Frahm, T. J. Scriba, H. Mahomed, W. Hanekom, P.-A. Bart, G. Pantaleo, G. D. Tomaras, S. Rerks-Ngarm, J. Kaewkungwal, S. Nitayaphan, P. Pitisuttithum, N. L. Michael, J. H. Kim, M. L. Robb, R. J. O'Connell, N. Karasavvas, P. Gilbert, S. C De Rosa, M. J. McElrath, and R. Gottardo. *COMPASS identifies T-cell subsets correlated with clinical outcomes.* Nature biotechnology, 33(6):610–616, 2015.

[76] J. A. Rebhahn, D. R. Roumanes, Y. Qi, A. Khan, J. Thakar, A. Rosenberg, F. E. H. Lee, S. A. Quataert, G. Sharma, and T. R. Mosmann. *Competitive SWIFT cluster templates enhance detection of aging changes*. Cytometry Part A, 89(1):59–70, 2016.

[77] S. Van Gassen, C. Vens, T. Dhaene, B. N. Lambrecht, and Y. Saeys. *FloReMi: Flow density survival regression using minimal feature redundancy*, 2016.

[78] C. B. Bagwell, B. C. Hunsberger, D. J. Herbert, M. E. Munson, B. L. Hill, C. M. Bray, and F. I. Preffer. *Probability state modeling theory*. Cytometry Part A, 87(7):646–660, 2015.

[79] M. S. Inokuma, V. C. Maino, and C. B. Bagwell. *Probability state modeling of memory CD8+ T-cell differentiation*. Journal of immunological methods, 397(1):8–17, 2013.

[80] C. B. Bagwell, B. L. Hill, B. L. Wood, P. K. Wallace, M. Alrazzak, A. S. Kelliher, and F. I. Preffer. *Human B-cell and progenitor stages as determined by probability state modeling of multidimensional cytometry data*. Cytometry Part B: Clinical Cytometry, 88(4):214–226, 2015.

[81] E. Marco, R. L. Karp, G. Guo, P. Robson, A. H. Hart, L. Trippa, and G.-c. Yuan. *Bifurcation analysis of single-cell gene expression data reveals epigenetic landscape*. Proc Natl Acad Sci U S A, 111(52):5643–5650, 2014.

[82] M. Setty, M. D. Tadmor, S. Reich-Zeliger, O. Angel, T. M. Salame, P. Kathail, K. Choi, S. Bendall, N. Friedman, and D. Pe'er. *Wishbone identifies bifurcating developmental trajectories from single-cell data*. Nature Biotechnology, 34(April):1–14, 2016.

[83] C. Trapnell, D. Cacchiarelli, J. Grimsby, P. Pokharel, S. Li, M. Morse, N. J. Lennon, K. J. Livak, T. S. Mikkelsen, and J. L. Rinn. *The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells.* Nature biotechnology, 32(4):381–6, 2014.

[84] J. Shin, D. A. Berg, Y. Zhu, J. Y. Shin, J. Song, M. A. Bonaguidi, G. Enikolopov, D. W. Nauen, K. M. Christian, G. L. Ming, and H. Song. *Single-Cell RNA-Seq with Waterfall Reveals Molecular Cascades underlying Adult Neurogenesis*. Cell Stem Cell, 17(3):360–372, 2015.

[85] I. C. Macaulay, V. Svensson, C. Labalette, L. Ferreira, F. Hamey, T. Voet, S. A. Teichmann, and A. Cvejic. *Single-Cell RNA-Sequencing Reveals a Continuous Spectrum of Differentiation in Hematopoietic Cells*. Cell Reports, 14(4):966–977, 2016.

# 6 CytofNorm: A Cross-Sample Cell-Type Specific Normalization Algorithm for Clinical Mass Cytometry Datasets

*In this chapter, we propose a normalization method for clinical mass cytometry data. As machine learning algorithms learn from the data presented to them, it is important that this data is of good quality. Otherwise all results learned might be skewed towards the errors in the data. One issue that often occurs with clinical cytometry data is the presence of batch effects when samples are processed over a longer time period. The processing procedure might vary slightly from the beginning towards the end. When comparing the samples, it might be difficult to distinguish the real biological differences from the differences caused by the time difference. Our algorithm will first identify separate cell types, because these might behave differently under the batch effects. For each of the cell populations, a quantile normalization procedure is executed.*

**Sofie Van Gassen*, Brice Gaudilliere*, Tom Dhaene, Martin Angst, Garry Nolan, Yvan Saeys and Nima Aghaeepour**

**Submitted to Cytometry Part A.**
* Equally contributed to the work

**Abstract** High-dimensional flow and mass cytometry have become technologies able to do deep phenotyping of cellular systems at a clinical scale. The resulting high-content datasets allow characterizing the human immune system with unprecedented single cell resolution. However, the results are highly dependent on sample preparation and measurements might drift over time, and while plenty of controls exist for assessment and improvement of data quality in a single sample, the challenges of cross-sample normalization attempts have been limited to aligning the distribution of subjects. These approaches, inspired by bulk genomics and proteomics assays, ignore the single-cell nature of the data and risk the removal of biologically relevant signals.

We propose a new normalization algorithm to ensure internal consistency between clinical samples. In this approach a shared control is included in each batch of samples, from which a machine learning algorithm will then learn a transformation for each control (e.g. from each analysis day). Importantly, some sources of technical variation are strongly influenced by the amount of protein expressed on each cell, which means that a simple transformation for all cell types would never sufficiently adjust all cells from a heterogeneous sample. To address this, our approach first identifies the overall cellular distribution before calculating a transformation for the different control samples. This transformation is then applied to all other clinical samples in the batch to remove all batch-specific variations.

To evaluate our algorithm, an extra control was taken along on all the plates, which should converge if the normalization algorithm works correctly. We make use of Earth Mover's Distance to determine the remaining differences between the plates after normalization, and are able to gain a strong reduction compared to a quantile normalization without cell population identification. This stresses the need for a cell-type specific approach.

## 6.1 Introduction

Mass cytometry is a variant on the traditional flow cytometry technique, in which antibodies are bound to metal tags instead of fluorochromes and time-of-flight measurements of cell particles in a spectrometer are used to identify labeled proteins [1]. The main advantage of mass cytometry over traditional flow cytometry is the increase in the amount of proteins that can be measured simultaneously. Whereas with flow cytometry the current limit is about twenty [2], mass cytometry can easily reach forty and promises to go up even further.

This technique can be used to investigate many different kinds of cell populations and tissues, one example being the follow-up of the immune system, giving insight in infection, inflammation and cancer [3]. However, to make scientific sound observations when following a patients development or comparing between different patients, data quality is crucial. One of the known properties of the mass cytometer is signal variation over time, due to changes in instrument performance and sample preparation [4].

The slight drift over time that happens while measuring one sample, is typically corrected for by including beads in the sample [5]. However, these beads will not capture any differences in sample preparation or channel-dependent changes. As such, additional corrections need to be applied to solve the technical variability between samples. The state-of-the-art technique to remove technical variability and differences in sample preparation is called barcoding [6]. This approach allows comparability between samples by first staining each of the individual samples with a specific set of tags, allowing unique identification of the samples. Afterwards, the samples can be merged together and further processed as one batch, removing all variability in the sample handling from that point on, including the staining with the panel of interest and the time of measurement in the machine.

Due to the limited amount of tags that can be measured simultaneously in a traditional flow cytometry setting, the barcoding technique is mainly used for mass cytometry experiments. This is a great solution if your amount of samples is limited and all samples are available at the same time. However, for practical reasons, samples are often split in multiple groups, each of which is processed on one plate and barcoded together. While comparisons between samples on the same plate are then feasible, comparisons between the plates will still suffer from variability in sample handling.

We propose to include an identical control sample on each of the plates, which can be aliquoted from a healthy control patient. Using the information from this control sample, our algorithm can remove batch effects between different plates which are each barcoded separately. This results in normalized data which can be compared between plates and used for further analysis.

## 6.2 Problem Statement: Characterizing Batch Effects

When a sample is split over multiple plates and each plate is analyzed separately, the measurements will differ slightly. To investigate if this is really due to the sample handling or whether it is caused by the small differences in the aliquots, we took two control samples along over multiple plates.

From Figure 6.1, we can conclude that the main differences are really caused by batch effects: both samples on the same plate have always undergone a similar change in comparison to the other plates. From this observation, we can assume that all samples on one barcoded plate will have similar changes happening. If we can learn these shifts based on one control sample taken along on all the plates, this will give us sufficient information to correct for the technical variability caused by the batch effects.

We can also identify another important characteristic of the batch effects in Figure 6.1. The size of the shift seems dependent on the intensity value: often the line indicating the 25% quantiles follows a different pattern than the line indicating the 75% quantiles. Just applying a linear shift to all cells will not remove the batch effect correctly, so we need to take this into account in our solution. We will do this by using spline functions. However, to estimate these splines, the control sample should contain sufficient information about all possible intensity ranges. Therefore, it can be necessary to include both stimulated and unstimulated control samples in your experimental setup.

Another characteristic of the batch effects is not visible in Figure 6.1. Because we show the quantiles per marker separately, we cannot identify dependencies between markers. However, cell type dependent effects can occur. An example is shown in Figure 6.2, where cells have about the same intensity values for the HLADR marker, but they still show a very different pattern depending on their other markers (gated as B cells or macrophages). To handle these cell type specific batch effects, we will first apply a clustering on the cytometry data, using the FlowSOM algorithm.

## 6.3 Proposed Method

A schematic overview of the proposed method is given in Figure 6.3. The method consists of two main parts: the batch effects are first modeled using the control samples and after that, all files are normalized using the resulting models.

### 6.3.1 Modeling the Batch Effects

To model the batch effects, we use a pipeline consisting of multiple steps. First we cluster the cells of the control files to capture the different cell types present in the data. Next we compute quantiles, to capture the distribution of the cells over the marker values. We determine a goal distribution based on the means of the quantiles, and finally compute splines to translate the original values to new values that confirm to the goal distribution.

To cluster the cells, we use the FlowSOM algorithm [7]. Because we want to capture all cell types present in the result, the control files are first aggregated

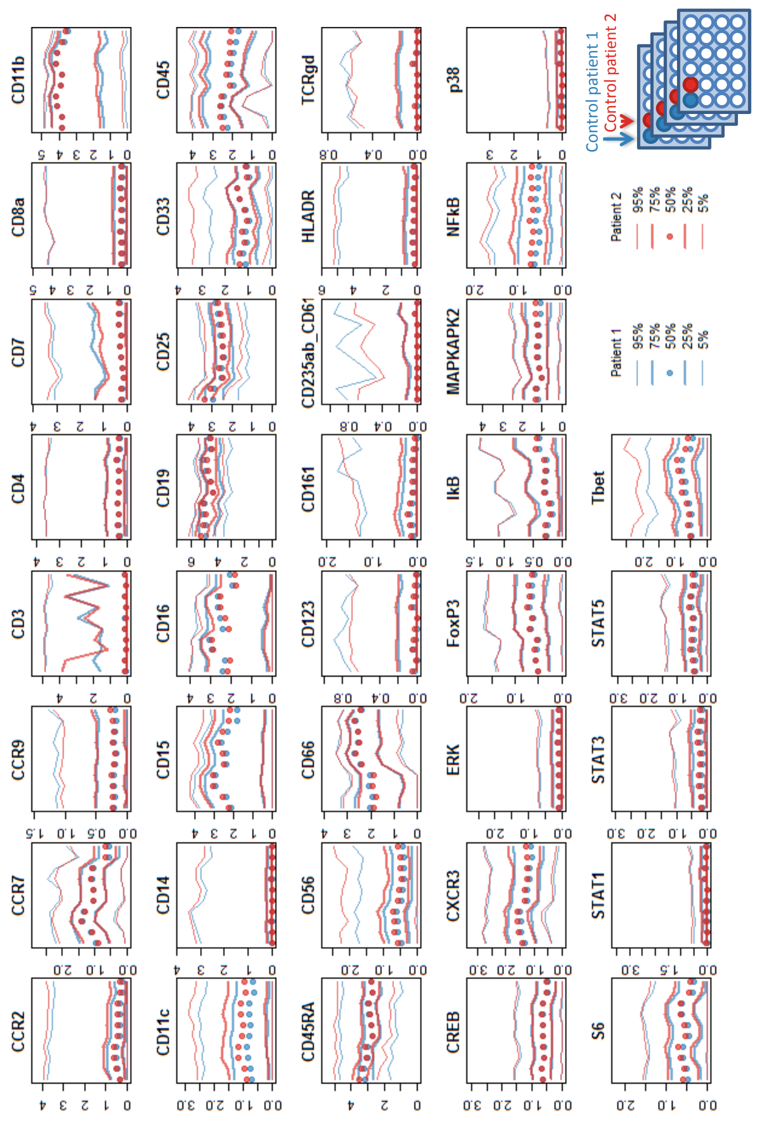


*Figure 6.1:* *Aliquots of blood samples from two healthy volunteers were measured on ten separately barcoded plates. The ten plates are represented on the x-axis, where the y-axis indicates the median and quantiles of the arcsinh-transformed intensity values. The first three rows contain surface markers, the last two intracellular markers. If the measurements had been executed perfectly, all measurements for a patient would be exactly the same, resulting in flat lines. However, the measurements differ slightly for the separate plates. It is notable that the patterns of these shifts are very similar for both patients, which indicates that these shifts are caused by batch effects: samples from the same plate are affected similarly.*

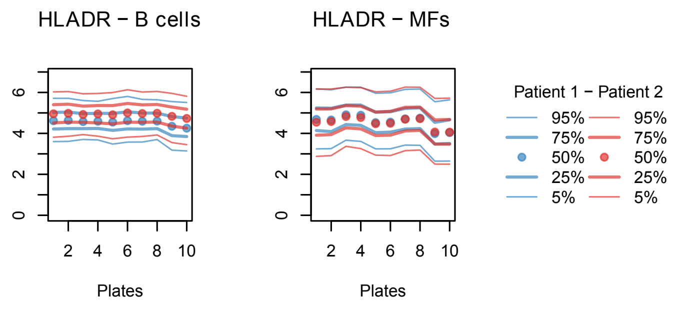


***Figure 6.2:*** *Aliquots of blood samples from two healthy control patients were measured on ten separately barcoded plates. We focus on the median HLADR intensities for manually gated B cells and macrophages (gated as resp. CD45+CD66-CD3-CD7-CD11c-CD123- and CD45+CD66-CD3-CD7-CD11c+CD123-CD11b+CD33+CD14+). Even on the same intensity level, these cells undergo different variations, while the same effects are still occurring on the samples of both patients. This indicates that the batch effects can be cell type specific and a one-dimensional approach will not be sufficient to remove them.*

and the FlowSOM algorithm is applied on all of them together. For computational reasons, it is possible to select a random subset of cells to build the clustering, and then map all the original cells on the clusters for further analysis. The FlowSOM algorithm uses a two-step clustering: the data is first overclustered using a self-organizing map and additionally the resulting cluster centers are clustered themselves using a consensus hierarchical clustering. By using this two-step clustering, the FlowSOM algorithm can detect clusters of varying sizes and shapes, without the computational overhead of density-based clustering algorithms. When applying clustering on the data first, we make the assumption that while the measurements might have shifted between the different samples, the differences between the cell types are still bigger than the shifts caused by the batch effects. A certain cell type might end up in different clusters of the self organizing map for the different files, but will still be in the same meta-cluster detected by the consensus clustering. Each meta-cluster will typically correspond with a different cell type.

Once the clusters are detected, we will apply normalization per cluster. This means that all control files get separately mapped to the FlowSOM clustering, and the cells belonging to each cluster are used for further analysis. Because we are using only one cluster at a time, we assume no further dependencies between the markers and will normalize each marker separately. To describe the distribution of the cells for a specific marker, we make use of quantiles. A quantile indicates a boundary value for which a certain percentage of cells is lower, for example the 0.5 quantile corresponds with the median value, whereas the 0 quantile corresponds with the minimum value and the 1 quantile with the maximum. A 0.25

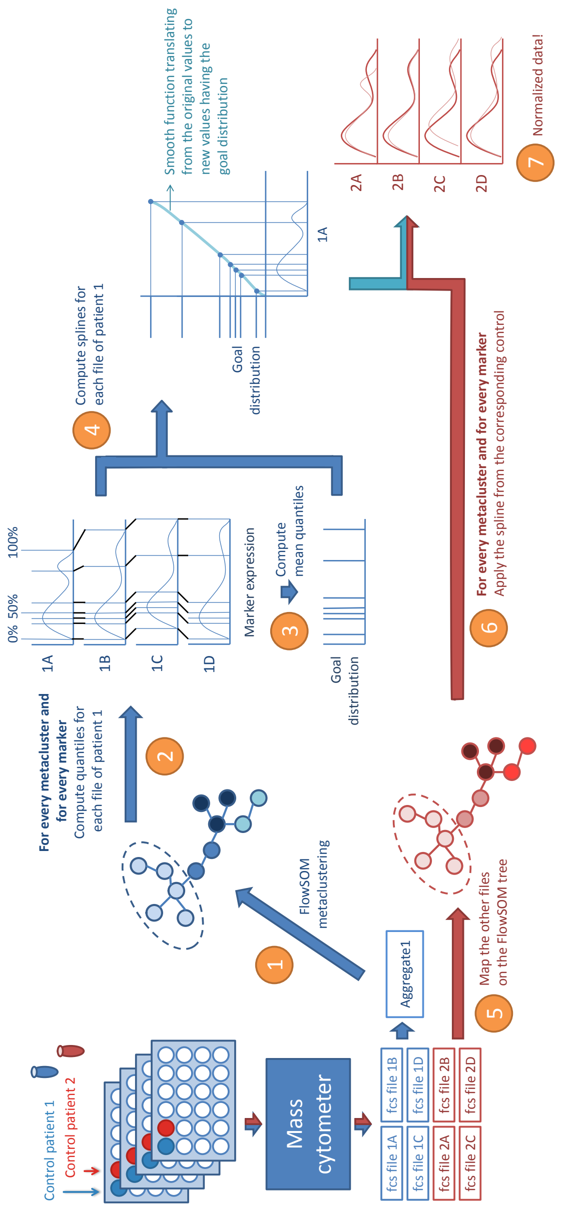

Control patient 1
Control patient 2
Mass cytometer
fcs file 1A
fcs file 1B
fcs file 1C
fcs file 1D
fcs file 2A
fcs file 2B
fcs file 2C
fcs file 2D
Aggregate1
1
FlowSOM metaclustering
2
For every metacluster and for every marker
Compute quantiles for each file of patient 1
1A
1B
1C
1D
0%
50%
100%
Marker expression
3
Compute mean quantiles
Goal distribution
4
Compute splines for each file of patient 1
Goal distribution
1A
Smooth function translating from the original values to new values having the goal distribution
5
Map the other files on the FlowSOM tree
6
For every metacluster and for every marker
Apply the spline from the corresponding control
2A
2B
2C
2D
7
Normalized data!

***Figure 6.3:*** *Schematic overview of the proposed workflow. First the batch effects are learned on the control sample (blue part). Afterwards, the other samples can be normalized (red part). Because an extra control sample was included on the plates, we can evaluate whether these samples become more similar as well.*

quantile will indicate the boundary for which one fourth of the cells are lower. By computing 21 quantiles, going from 0 to 1 in steps of 0.05, we get an overview of the distribution for each control file separately.

Because we are working with control files, the distribution of the cells should be very similar. If the values for the quantiles do not correspond, we can assume a technical issue is causing these shifts. We will build a model that updates the data in such a way that all control files will have the same quantiles in the end. Therefore, the mean value is computed for each of the quantiles over the different control files and these boundaries are set as the goal distribution.

To model the shifts, we use spline functions. A spline is a piece-wise defined function, used to interpolate between given points, while still keeping a certain smoothness at the transitions between the piece-wise defined functions. For each control file, we use the original quantiles as the x values and the corresponding quantiles from the goal distribution as the y values to define the interpolation points. The resulting spline function can then be used to translate all original marker values to the new values, in such a way that the resulting data will be close to the goal distribution.

Running this whole procedure will result in one spline per marker per cluster per control file.

### 6.3.2 Normalizing the Data

To normalize the other files on the plates, we will use the FlowSOM clustering and the splines computed in the previous part.

First the new files are mapped on the given FlowSOM clustering, where every cell gets assigned to a cluster. Then, for every cell in the dataset, we know the cluster, the markers and the control file of relevance. By applying the corresponding spline function on each of the measured values, we will apply the same shifts to the cells from the new files as we have learned from the control files. That way, we end up with data from which we can assume all batch effects, as detected in the control files, are removed and for which all variation left is representing real biological variation.

This whole pipeline is implemented as an R package and will be submitted to Bioconductor. As input, the user needs to define the fcs files from the control samples, the fcs files that need to be normalized and a labeling indicating from which batch each of these files come. In the end, a new set of fcs files is generated with normalized values.

## 6.4 Results

### 6.4.1 Unstimulated Samples from Two Healthy Controls

We will first demonstrate our method on a dataset for which each plate contains two unstimulated control samples, both from healthy controls. This way, we can use the controls from the first patient to learn the batch effects and normalize the control samples from the second patient. Then we check if the normalization is working as expected: all aliquots from the second patient should become very similar. This evaluation is done per cell type, using a manual gating of the dataset (see Figure S1). After all, small differences in the aliquots might be present, and the goal is to align the different cell types over the plates.

To compute how well the samples correspond with each other, we make use of the Earth Movers Distance (EMD). This is a distance measurement developed specifically to compare distributions. To describe the distributions, we bin the data in bins of size 0.1 (on transformed data). For every cell type and every normalized marker, we compute the pairwise EMDs for all the files and take the maximum value, indicating how much two cell populations which should actually be the same can differ from each other. The lower this value is, the better. We apply this measurement on both the original and the normalized files and plot those values against each other, resulting in Figure 6.4. We can conclude that most distances decrease, except a few that were already very small in the first place. We ignore all distances smaller than 2 (as those represent the marker - cell type combinations which have not really undergone any batch effects) and compute the mean percentage of decrease in distance, resulting in a reduction of 0.636.

Additionally, we also applied quantile normalization on the whole dataset, without clustering first. We compare the results from this approach to the normalized files as described before. This comparison is shown in Figure 6.5 and indicates that most marker distributions per cell type become more similar when using the clustering approach. When computing the reduction compared to the original distances as before, this algorithm only results in a 0.335 reduction.

### 6.4.2 Additional Stimulated Samples from Two Healthy Controls

To show the importance of representative control files, we extend our dataset with stimulated control samples that were also taken along on the plates. The cells in these stimulated files express some cytokines that were negative in the unstimulated files. We explore three different settings: training on only the unstimulated files of patient 1, training on only the stimulated files of patient 1, or training on both. The results on the files from patient 2 are shown in Figure 6.6.

When we train only on the unstimulated file from patient 1, we estimate the

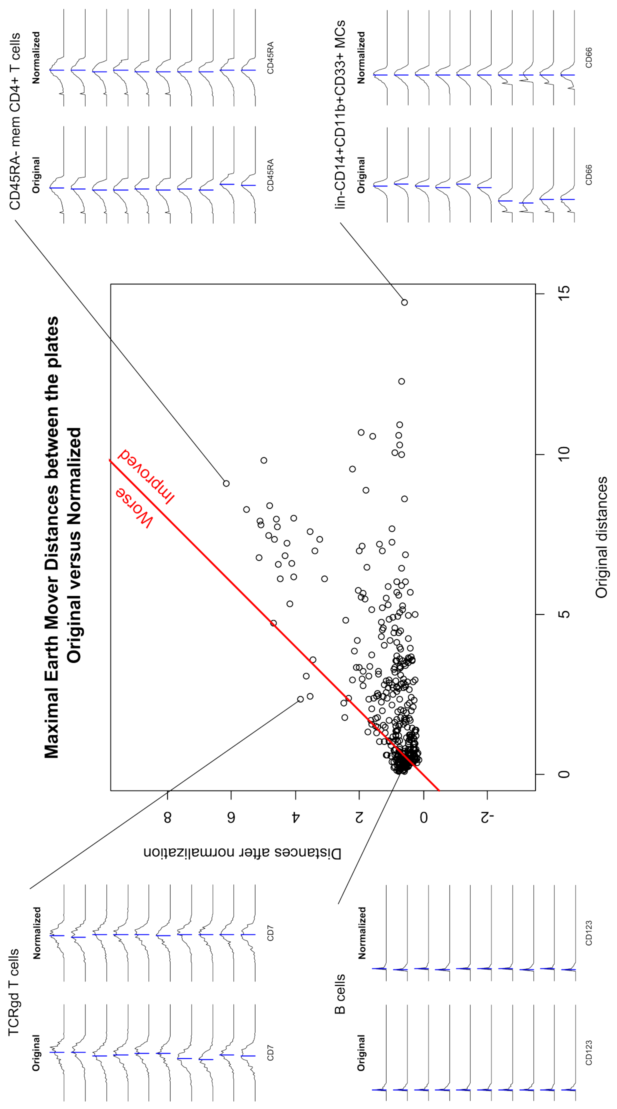


***Figure 6.4:*** *Overview of the improvements made by the normalization for patient 2, files not used to learn the batch effects. In the main panel (middle) every dot corresponds with the marker distribution for a specific cell type (37 markers x 10 cell types = 370 dots). The values indicate the maximal EMD between plates, for the original files (x-axis) and the normalized files (y-axis). The lower this distance, the stronger all plates resemble each other, which should be the case as all samples are aliquots from the same patient. All dots below the red line have a lower maximal distance after normalization and are thus improved. A few dots lay above the line, which means their distributions are less similar after normalization, but this is mainly the case for very small values: they were already almost identical to begin with and they are still very similar afterwards. On the sides, four specific cases are shown in more detail. The marker distribution is shown as a histogram for the ten plates underneath each other, and a blue line indicates the median marker value.*

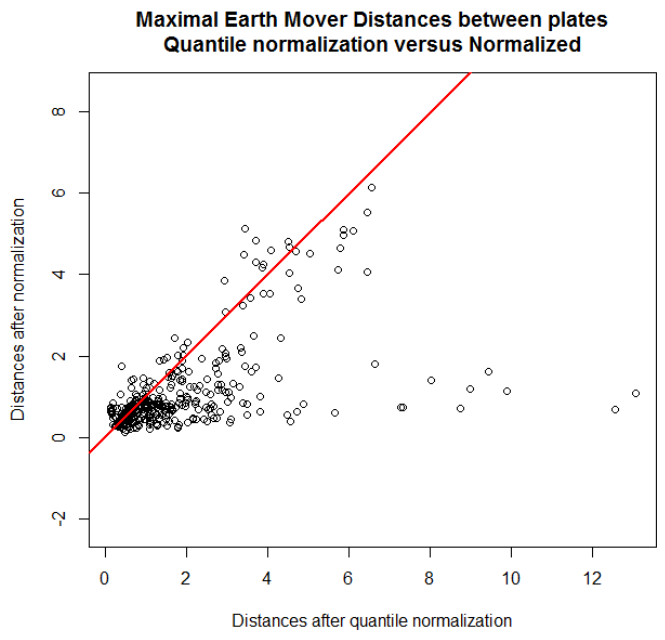


***Figure 6.5:*** *Overview of the improvements made by the normalization compared to normalization without clustering first. Figure interpretation similar to Figure 6.4, every dot represents a specific marker distribution for a specific cell type, and the values indicate the maximum EMD between two plates. Everything below the red line has been improved by using the clustering approach.*

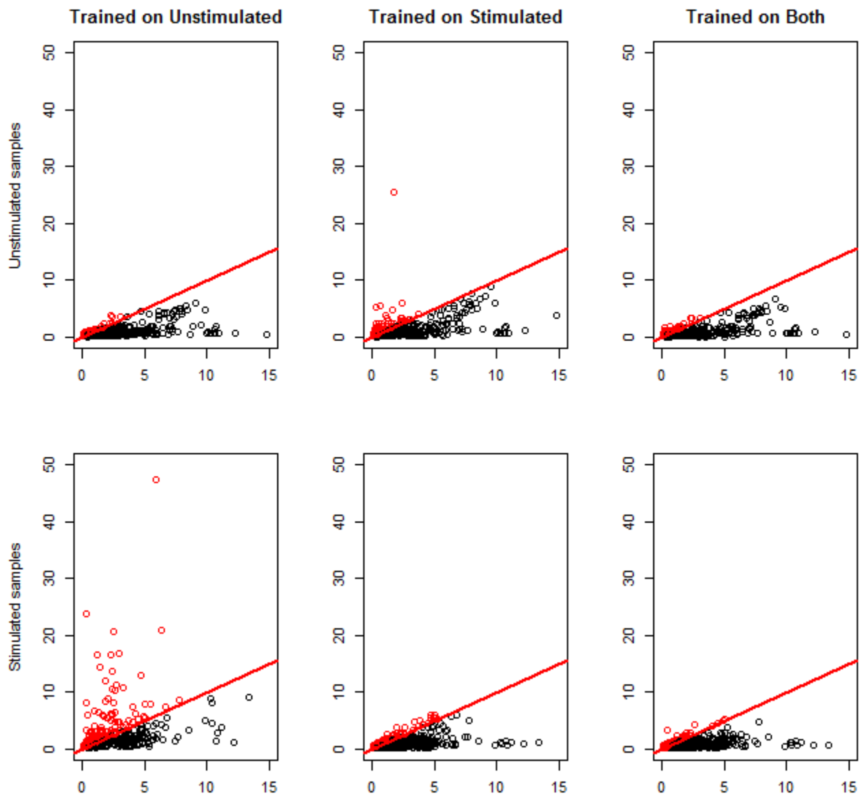


***Figure 6.6:*** *Maximal EMDs between plates for the samples of patient 2, computed on the original files (x-axis) and the normalized files (y-axis). Every dot represents the distribution for one specific marker for one specific cell type. Everything above the red diagonal has become worse after normalization. This is especially present when we normalize the stimulated files after having only trained on the unstimulated files or vice versa, because the cells of the training files do not span the full expression range of the cells in the files to normalize. When we use the correct training set we have little problems, only for populations which were not moving much in the first place. When we use both training sets together, all files get normalized well, indicating that this is a good strategy if you are not sure what will be present in the other wells on the plate.*

batch effects correctly for the unstimulated files of patient 2: almost all distances become smaller, as shown in the previous part as well. However, when we use this model on the stimulated files, many distances are becoming larger instead of smaller. This can be easily explained: the batch effects are not linear over the range of the expression values and we could only train on cells where the expression values were low. The model has no information about the batch effects for high expression and makes mistakes when extrapolating. When we train on the stimulated files, this problem is solved and the stimulated files are correctly normalized. However, now the unstimulated files can pose a problem because the negative populations are not always represented in the stimulated files.

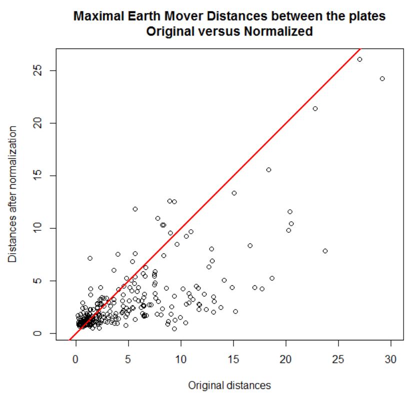
The solution is straightforward: we use both the stimulated and the unstimulated samples for training. This way, the model is built on information over the whole range of expression and works well in all situations. Therefore, it might be advisable to include two or more control samples in your setup if you are not sure what the range of the cells in the other wells on the plate might be.

### 6.4.3 Other Datasets

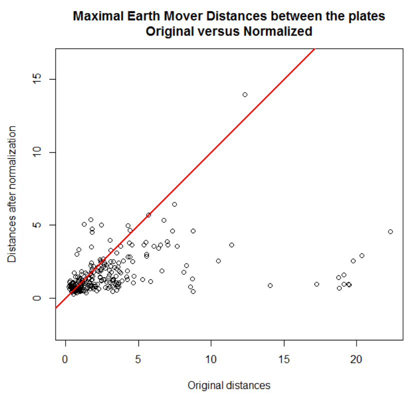
Finally, we show that the method is generalizable by applying it to two clinical datasets for mass cytometry analysis of: 1) normal pregnancy and 2) a immune-modulating nutritional supplement as an intervention post-surgery. Both datasets include longitudinal samples collected clinically over the course of several months and include respectively a total of 19 and 11 plates. For these datasets, only one control patient was taken along, from which both a stimulated and an unstimulated sample were present on all the plates. We use the stimulated samples to train the algorithm and the unstimulated ones to normalize. As shown in the previous paragraph, this might give issues, especially for the intracellular markers which are strongly influenced by the stimulation. Therefore, we only apply the normalization on the surface markers in these datasets. On these datasets, our algorithm produced reductions of 35.2 and 35.9 percent. See Figure 6.7.

## 6.5 Discussion

We propose a normalization strategy that enables mass cytometry analysis of large clinical cohorts. Importantly, to avoid accidental removal of biologically relevant signals, the algorithm makes no assumptions about the distributions of the clinical samples and relies on the internal consistency of cellular barcoding and externally spiked control samples. Also, the algorithm uses a multi-layer learning strategy to account for cell type specific technical variations. The results demonstrate significant improvements in the quality of primary clinical samples based on both automated and manual analysis of data. This strategy has already been implemented in



(a) Pregnancy dataset:
19 plates, reduction: 0.245.



(b) Post-surgery dataset:
11 plates, reduction: 0.318.

***Figure 6.7:*** *Results on two additional datasets. For both datasets, the model is trained on the stimulated control samples and the results are computed on the unstimulated control samples. Only surface markers were taken into account.*

several clinical studies at Stanford. An open source R package is available upon request (and soon through Bioconductor).

## Acknowledgment

S.V.G. is funded by the Flanders Agency for Innovation by Science and Technology (IWT). Y.S. is an ISAC Marylou Ingram Scholar.

## References

[1] D. R. Bandura, V. I. Baranov, O. I. Ornatsky, A. Antonov, R. Kinach, X. Lou, S. Pavlov, S. Vorobiev, J. E. Dick, and S. D. Tanner. *Mass cytometry: Technique for real time single cell multitarget immunoassay based on inductively coupled plasma time-of-flight mass spectrometry*. Analytical Chemistry, 81(16):6813–6822, 2009.

[2] S. P. Perfetto, P. K. Chattopadhyay, and M. Roederer. *Seventeen-colour flow cytometry: unravelling the immune system.* Nature reviews. Immunology, 4(8):648–655, 2004.

[3] Z. B. Bjornson, G. P. Nolan, and W. J. Fantl. *Single-cell mass cytometry for analysis of immune system functional states*, 2013.

[4] K. Kleinsteuber, B. Corleis, N. Rashidi, N. Nchinda, A. Lisanti, J. L. Cho, B. D. Medoff, D. Kwon, and B. D. Walker. *Standardization and quality control for high-dimensional mass cytometry studies of human samples*. Cytometry Part A, 89(10):903–913, 2016.

[5] R. Finck, E. F. Simonds, A. Jager, S. Krishnaswamy, K. Sachs, W. Fantl, D. Pe'er, G. P. Nolan, and S. C. Bendall. *Normalization of mass cytometry data with bead standards*. Cytometry Part A, 83 A(5):483–494, 2013.

[6] E. R. Zunder, R. Finck, G. K. Behbehani, E.-A. D. Amir, S. Krishnaswamy, V. D. Gonzalez, C. G. Lorang, Z. Bjornson, M. H. Spitzer, B. Bodenmiller, W. J. Fantl, D. Pe'er, and G. P. Nolan. *Palladium-based mass tag cell barcoding with a doublet-filtering scheme and single-cell deconvolution algorithm.* Nature protocols, 10(2):316–33, 2015.

[7] S. Van Gassen, B. Callebaut, M. J. Van Helden, B. N. Lambrecht, P. Demeester, T. Dhaene, and Y. Saeys. *FlowSOM: Using self-organizing maps for visualization and interpretation of cytometry data*. Cytometry Part A, 87(7):636–645, 2015.

# 7
# Conclusion

In the last decade, technological advances in the cytometry field resulted in an increasing number of cells, markers and patients measured. Manual analysis cannot keep up with these changes and the need for computational flow cytometry techniques emerged for cell population detection and patient diagnostics.

By using machine learning techniques, multiple research groups developed algorithms to detect cell populations. However, many algorithms either make strong assumptions on the distribution of the data or have a very high execution time to cluster the data. Additionally, there was a clear need for a more visual approach compared to just returning cell counts from the clusters. Therefore, we developed the FlowSOM algorithm. The innovative aspect of this algorithm is the two-step clustering. First, the data is overclustered, which is useful to give a detailed view of the data. By using a metaclustering step afterwards, the assumptions made by the SOM are lifted, resulting in a fast algorithm without strong assumptions on the data. This technique seems very promising and allowed us to collaborate with multiple wet lab teams, resulting in two co-publications and improved functionality of our available software. FlowSOM's combination of a good accuracy with fast running times was also confirmed in an independent review by Weber and Robinson, published in Cytometry Part A, December 2016.

In our next work, FloReMi, we did not just focus on cell population detection, but rather on a whole pipeline, analysing a dataset from start to finish. This included multiple preprocessing steps to clean the data. The implementation of functions for automated quality control could be very helpful to flag problems for the researcher, who might not always do an extensive check of possible issues. We did not use FlowSOM to extract features in this approach, but rather the flowDensity/flowType combination. Where FlowSOM will identify the main populations, flowType will really identify all possible populations. While this is not optimal to get a clear general overview of the data, in this case we wanted to specifically identify which populations showed any correlation with the clinical outcome (HIV progression time). This made it possible to first evaluate many possible populations and then only focus on those with a good correlation with the outcome and minimal redundancy between themselves. Due to the time limit of the challenge, our cross-validation strategy was limited. This showed in the final results, as we submitted three variations of our algorithm of which two overfitted strongly. This stresses the importance of a good cross-validation practice when building new models, especially because the potential number of features is huge. However, our third variation used random survival forests. This non-linear technique has more power and random forests are also known for being more robust against overfitting. This resulted in statistical significance on the test set as well, giving a starting point for further research towards the selected populations.

Although it might seem less like a traditional machine learning task, practice taught us that preprocessing is also a crucial step in any flow cytometry analysis. After all, your results can only be as good as the input you give to the algorithm. With CytofNorm, we focused on the normalization of mass cytometry data. We initially tried several approaches to solve this problem, but in the end a cell type specific approached proved necessary. While our algorithm was developed with barcoded mass cytometry plates in mind, it can also be applied to any other setting in which batches of samples are processed, as long as a control is taken along which is comparable between all batches. This will enable researchers to have more trust in the comparison between samples from different batches.

While the main focus was on algorithm development for flow cytometry data, it is also important that these algorithms find their way to the people in the lab. Because of this, we spent some effort in reaching a broader audience than just the computational flow cytometry community. Part of this were collaborations with wet lab teams, but the Nature Review paper is also an important element in introducing these new techniques to immunologists. This way, computational flow cytometry can really have an impact on immunologic research and drug development.

## 7.1 Future perspectives

While our algorithms were well perceived by the community, there is of course still room for further development of computational flow cytometry techniques. Both the FlowSOM and FloReMi approach could be improved upon themselves, while complementary techniques will also enhance cytometry analyses.

One of the possible improvements on FlowSOM is purely technical: the development of a parallelized version with better memory handling. While FlowSOM scales linear with respect to the number of cells and strongly improves the running time over SPADE, the R implementation runs on a single core and keeps the whole training dataset in memory. This will cause issues for datasets which are combining many samples, resulting in high memory usage and waiting time. As parallelized versions of the SOM algorithm exist, a new implementation which uses these variations will enable further speed-up without changing the accuracy of the clustering.

As was already explored in Chapter 3, the minimal spanning tree visualization also has its limitations. If any loops would be present in the real cluster structure, the MST will just cut them at a random point, resulting in a tree that is difficult to interpret. While we explored some other layout options, none of them seemed optimal yet. A graph-based approach can overcome the topological limitations, but often still needs manual parameter tuning to obtain good results. Algorithms which could learn the optimal parameter settings from the data might alleviate this issue.

Additionally, estimating the amount of clusters in a dataset is a well-known problem in machine learning. While several approaches exist, they do not always correspond with the expert's biological knowledge. In FlowSOM, this issue can be regarded on two levels: determining the size of the SOM grid and determining the number of meta-clusters. Currently, both need to be defined up front. Several SOM-inspired algorithms exist which can extend the grid when it does not seem sufficient to explain the data, such as growing neural gas [1] and the growing hierarchical self-organizing map [2], which avoid determining the grid size. However, as the goal of the first clustering is to overcluster the data, these algorithms might need some adaptations to work in this setting. But as the data is overclustered, it also means the grid size is not that crucial, as long as it is big enough. The second level of clustering is harder, as the actual cell types need to be distinguished at that point. Where clustering the FlowSOM nodes allows using slower algorithms because the number of nodes is limited, it might be necessary to use more information than just the cluster centers to estimate the optimal number of cell populations automatically.

This brings us to the biggest remaining problem: detecting rare cell types. As the number of markers is limited, the flow cytometrist often picks a set of markers which can give detailed information about a small subpopulation. However, this also means that many other cell populations cannot be distinguished, resulting in some big clusters and then a few very small ones which might be of clinical relevance. While the two-level clustering of FlowSOM allows a wide range of cluster sizes, very rare cell types will already be overshadowed when building the grid. Incorporation of negative controls and FMO's in the clustering algorithms might help to reveal more of the biological interpretation, where the raw measurements might suffer from too much noise to clearly distinguish these populations.

New techniques also bring new challenges. For example, an imaging flow cytometer does not just measure the fluorescent light scattered by the cell, but takes an actual photograph of the cell measured. This allows information about the location of the proteins, which can give unprecedented insights, but also complicates the computational aspect of the analysis. Image processing algorithms can extract many features, but they are often highly correlated and it is uncertain whether they contain relevant information or not. This is very different from the typical cytometry features, which are expert-picked to contain maximal information. Feature selection algorithms will need to be incorporated in the workflow to handle this type of data. A similar argument can be made for single-cell RNAseq, where gene expression is measured for individual cells. While the traditional RNAseq algorithms will need adaptations due to the single-cell nature of the data, single-cell techniques as described in this thesis will also need adaptations to work with the increased number of features which might be highly correlated.

Whereas feature selection is not yet necessary for cell population identification in most cases, it is already crucial for building diagnostic and prognostic models on the patient level. This was incorporated in our FloReMi pipeline, but as it was build against a sharp deadline for participation in the FlowCAP IV challenge, further exploration of the possibilities would be valuable. Both supervised and unsupervised feature selection techniques might be of interest. Where supervised techniques will enable the researcher to concentrate on the cell populations correlated with the clinical outcome, they will often need to be applied in a filter approach, as thousands of features can be extracted from flow cytometry data, too many to explore all multivariate possibilities. Using unsupervised feature selection might allow the researcher to first build a good fingerprint of the patient's immune system, in which multivariate models might pick up unexpected multivariate signals of interest.

The other building blocks in the FloReMi pipeline could also be further optimized. While the preprocessing steps showed good results on this dataset, the parameter tuning was done by visually inspecting many samples. New algorithms have been published in the meantime, but they also are dependent on correct parameter tuning. Automated flagging of samples in which measurement issues occur would be very valuable in any flow cytometry experiment, even when a traditional analysis is used further on. The feature extraction step could also use further optimization. Where the flowType/flowDensity seems very promising and exhaustive, it only builds populations on binary splits: every cell is labeled as positive or negative for a marker. Dim populations will not be identified. Furthermore, some markers might need different cutoffs depending on which cell type they are present. Further exhaustive clustering algorithms could be explored in combination with multiple diagnostic models. A more extensive benchmark on multiple datasets is necessary to either determine the optimal combination or to determine the specific cases in which one approach works better than another, if there is no one-fits-all solution.

Additional attention will need to be paid to batch effects when analyzing clinical data. When data is collected over a longer time period, small technical changes are unavoidable. When comparing patients, these technical variations should be removed before making any biological conclusions. With the CytofNorm algorithm, we showed that a cell-type-specific approach is necessary. However, we focused on applying this technique on barcoded mass cytometry data. This approach should be further evaluated on non-barcoded batches to evaluate the performance on flow cytometry data and its general usefulness in the clinic.

All computational flow cytometry algorithms are currently mainly evaluated on accuracy. However, while accuracy is of utmost importance, stability is another aspect of machine learning techniques that should be further investigated. If a dataset differs slightly (a very reasonable assumption, imagine the researcher measuring a few more or a few less cells), the main conclusions, such as the clusters which are detected and the biomarkers which are identified for clinical prediction, should stay the same. However, several techniques might end up returning very different results if the dataset differs slightly. This complicates the interpretation of the resulting models and more stable techniques will allow more insights into the underlying biological systems. The stability of a technique can be investigated by artificially creating slightly different datasets, e.g. by randomly changing the order of the cells or by taking multiple subsets containing 90% of the data etc. By evaluating whether the conclusions hold true for these modified datasets as well, more trust can be put in the results.

Finally, other types of analyses might be explored on flow cytometry data. Trajectory inference is one example, where the data is not divided in separate clusters, but rather one path through the data is inferred, describing a cellular differentiation process. Current algorithms are able to detect linear paths, but further research is applied to detect more complex structures, such as bifurcating or merging paths and loops. These paths are typically derived from snapshot data in which all states through the differentiation process are present, but this approach can also be extended to time series data, where the goal is to identify which cell types are being replaced by others during the course of a treatment and how the presence of these different cell types influence each other.

While many algorithms are being developed, one of the main challenges remains the easy incorporation of these new techniques in the lab. Currently, the field is still relatively new and strongly developing, with many alternative approaches being proposed and compared against each other. It is important that these techniques are tested on multiple real-life datasets, as they might work nicely on the dataset they were originally developed for, but might not always generalize very well. Therefore, good benchmark datasets will play an important role in the development of the algorithms. Additionally, good annotation tools will be necessary to allow the interpretation of these new results. As the field matures further, a selection of techniques will be incorporated in the traditional point-and-click analysis tools used by the immunologists, allowing them to profit from all the benefits provided by these automated techniques.

## References

[1] B. Fritzke et al. *A growing neural gas network learns topologies*. Advances in neural information processing systems, 7:625–632, 1995.

[2] M. Dittenbach, D. Merkl, and A. Rauber. *The growing hierarchical self-organizing map*. In Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on, volume 6, pages 15–19. IEEE, 2000.

# A
# External Validation

In this appendix, we present some evaluation results as published by other research groups.

## A.1 FlowSOM

In this section, we include some figures as published by Weber and Robinson: Comparison of clustering methods for high-dimensional single-cell flow and mass cytometry data. Cytometry Part A 89(12), 1084-1096, December 2016.

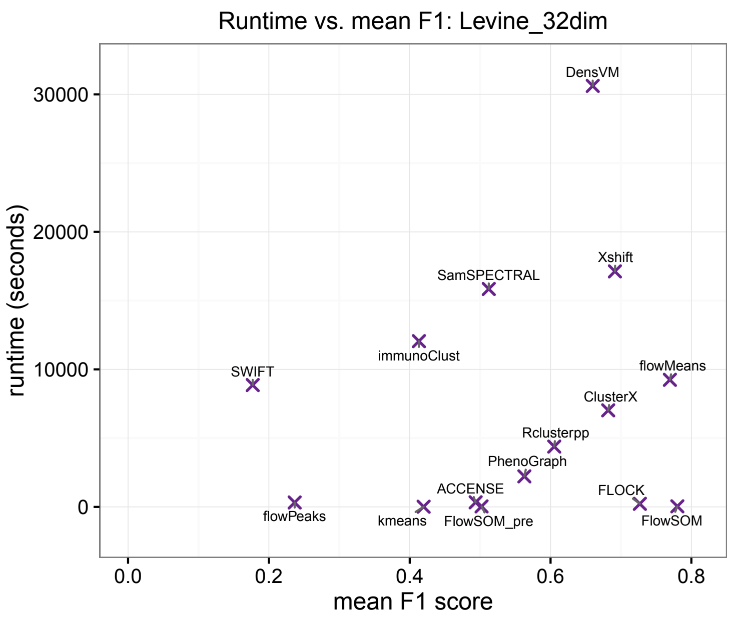


***Figure A.1:*** *Evaluation of multiple clustering algorithms. This dataset consists of 265,627 bone marrow cells from 2 healthy donors, stained for 32 surface markers. The F1-score is computed based on 14 manually gated populations (consisting of 104,184 cells (39%)). FlowSOM, with a fixed number of metaclusters, has both the best F1-score and the fastest runtime (41s).*

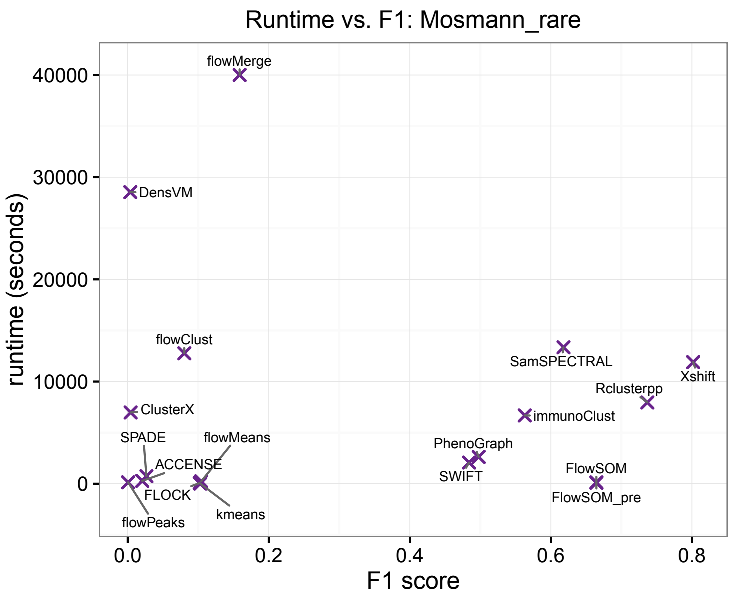


***Figure A.2:*** *Evalution on an additional dataset consisting of 396,460 peripheral blood cells from a healthy donor, stained for 14 markers. The F1-score evaluates the detection of a rare population consisting of 109 cells (0.03%). FlowSOM does not have the optimal result (it has a good recall, but only 50% accuracy, which means some other cells are clustered together with this small population). However, most other clustering algorithms perform worse, or make a big trade-off in computation time (e.g. X-shift takes more than 3 hours, where FlowSOM takes less than 3 minutes.*

## A.2 FloReMi

In this section, we include the results from the FlowCAP IV challenge, as published by Aghaeepour et al.:
A benchmark for evaluation of algorithms for identification of cellular correlates of clinical outcomes. Cytometry Part A 89(1), 16-21, January 2016.

***Table A.1:*** *Overview of the other methods submitted to the challenge.*

| METHOD | SHORT DESCRIPTION |
|---|---|
| BorFlowFP | Exhaustive all-relevant feature selection with the Boruta method over the flow cytometry fingerprints data followed by modeling with Survival Random Forest algorithm. |
| FloReMi.1 | Partitioning of cells (e.g., into positive or negative) to identify cell populations, followed by feature extraction of both cell population size and MFI values. Features with minimal redundancy were selected as input for a random survival forest for survival time prediction. |
| FloReMi.2 | As FloReMi.1, but using a Cox-proportional hazards model for survival time prediction. |
| FloReMi.3 | As FloReMi.1, but using an additive hazards model for survival time prediction. |
| flowDensity/ flowType/ 6RchyOptimyx | Density-based getting and partitioning of cells (e.g., into positive or negative) followed by dynamic programming to ID k-shortest paths to important cell populations. |
| GANN | Identifying profiles of individual bin channels of fluorescence for the different cell markers which can be used for distinguishing different groups / categories. |
| RTOMT | Partitioning of cells (e.g., into positive or negative). Combining clinical diagnosis and survival time into a single target vector. Regression tree. |
| SPADE.SNR | SPADE was used to derive cell clusters in high-dimensional space. Important clusters were selected by a signal-to-noise ratio based on cell frequency of subjects who showed disease progression. |
| UQS | Clustering using expectation maximization fitting of skew-t mixture models was used for feature extraction. A random survival forest was used for building the predictive model. |

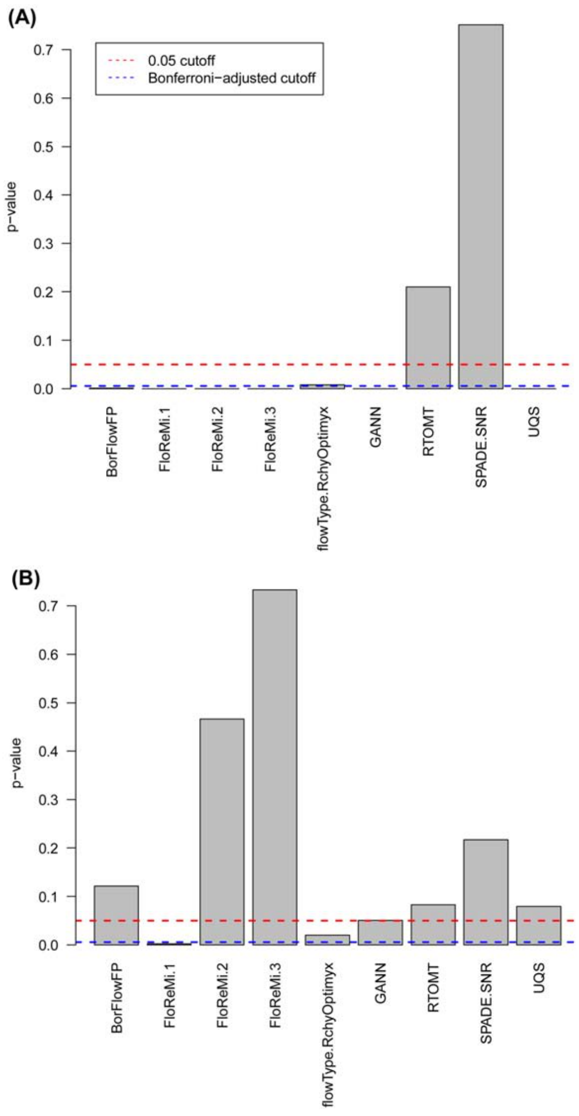


***Figure A.3:*** *Evalution of all submitted algorithms to the FlowCAP IV challenge on the training set (A) and independent test set (B). FloReMi 1, the version which uses Random Survival Forest as the prediction model, is the best performing algorithm on the test set, whereas the other two variations overfit.*

# B
# FlowSOM R Vignette

In this Appendix, the FlowSOM vignette is given, a tutorial for use of the FlowSOM R package. All these examples can be replicated by executing the given scripts after installing the FlowSOM package.

The FlowSOM package can be installed by running

```
devtools::install_github("saeyslab/FlowSOM",
build_vignettes = TRUE)
```

Sofie Van Gassen, Britt Callebaut, Tom Dhaene and Yvan Saeys

Ghent University

March, 2017

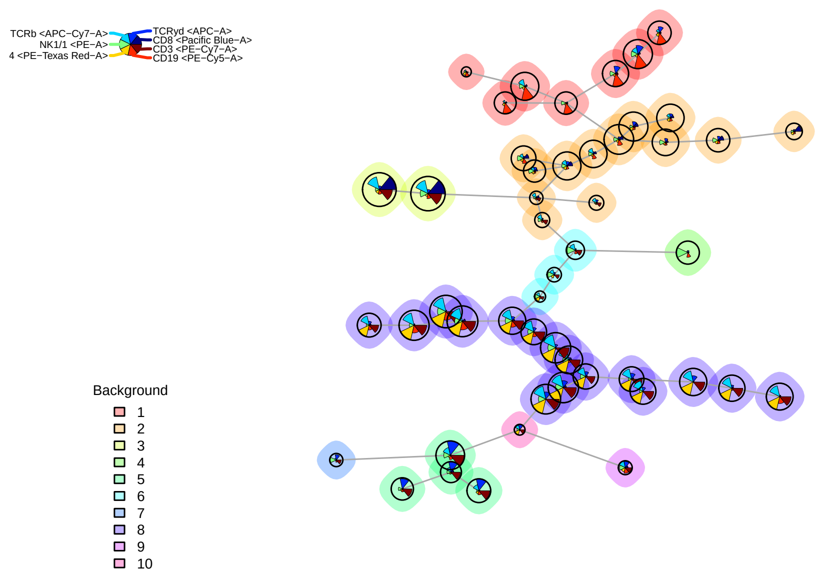
**Abstract**

The *FlowSOM* package provides new visualization opportunities for cytometry data. A four-step algorithm is provided: first, the data is read and preprocessed, then a self-organizing map is trained and a minimal spanning tree is built, and finally, a meta-clustering is computed. Several plotting options are available, using star charts to visualize marker intensities and pie charts to visualize correspondence with manual gating results or other automatic clustering results.

## 1 The easy way

The easiest way to use this package is using the wrapper function `FlowSOM`. It has less options than using the separate functions, but in general it has enough power. It returns a list, of which the first item is the FlowSOM object (as required as input by many functions in this package) and the second item is the result of the metaclustering.

```
> library(FlowSOM)
> fileName <- system.file("extdata","lymphocytes.fcs",
+                         package="FlowSOM")
> fSOM <- FlowSOM(fileName,
+                 # Input options:
+                 compensate = TRUE,
+                 transform = TRUE,toTransform=c(8:18),
+                 scale = TRUE,
+                 # SOM options:
+                 colsToUse = c(9,12,14:18), xdim = 7, ydim = 7,
+                 # Metaclustering options:
+                 nClus = 10,
+                 # Seed for reproducible results:
+                 seed = 42)
> PlotStars(fSOM$FlowSOM,
+             backgroundValues = as.factor(fSOM$metaclustering))
```



## 2 Reading the data

The FlowSOM package has several input options.

The first possibility is to use an array of character strings, specifying paths to files or directories. When given a path to a directory, all files in the directory will be considered. This process does not happen recursively. You can specify a pattern to use only a selection of the files. The default pattern is `".fcs"`, making sure that only fcs-files are selected. When you are already working with your data in *R*, it might be easier to use a *flowFrame* or *flowSet* from the *flowCore* package as input. This is also supported. If multiple paths or a *flowSet* are provided, all data will be concatenated. You should check and apply normalization if needed using other packages.

When reading the data, several preprocessing options are available. The data can be automatically compensated using a specified matrix, or using the `$SPILL` variable from the fcs-file (when `compensate` is `TRUE` but no value is given for `spillover`). The data can be transformed for specified columns. If no columns are provided, all columns from the spillover matrix will be transformed. Finally, the data can be scaled. By default, it will scale to a mean of zero and standard deviation of one. However, specific scaling parameters can be set (see the base *R* `scale` function for more detail).

```
> set.seed(42)
> library(flowCore)
> library(FlowSOM)
> fileName <- system.file("extdata","lymphocytes.fcs",
+                         package="FlowSOM")
> fSOM <- ReadInput(fileName,compensate = TRUE,
+                   transform = TRUE, toTransform=c(8:18),
+                   scale = TRUE)
> ff <- suppressWarnings(flowCore::read.FCS(fileName))
> fSOM <- ReadInput(ff,compensate = TRUE,transform = TRUE,
+                   scale = TRUE)
```

This function returns a FlowSOM object, which is actually a *list* containing several parameters. The data is stored as a matrix in `$data`, and all parameter settings to read the data are also stored. The begin and end indices of the subsets from the different files can be found in `$metadata`.

```
> str(fSOM,max.level = 2)
```

```
List of 12
 $ pattern          : chr ".fcs"
 $ compensate       : logi TRUE
 $ spillover        : num [1:11, 1:11] 1.00 4.84e-04 ...
  ..- attr(*, "dimnames")=List of 2
 $ transform        : logi TRUE
 $ toTransform      : chr [1:11] "FITC-A" "Pacific Blue-A" ...
 $ transformFunction:Formal class 'transform'
                        [package "flowCore"] with 2 slots
 $ scale            : logi TRUE
 $ prettyColnames   : Named chr [1:18] "Time <Time>" ...
  ..- attr(*, "names")= chr [1:18] "Time" "FSC-A" "FSC-H" ...
 $ data             : num [1:19225, 1:18] -1.65 -1.65 -1.65 ...
  ..- attr(*, "dimnames")=List of 2
 $ metaData         :List of 1
  ..$ C:/Users/.../lymphocytes.fcs: num [1:2] 1 19225
 $ scaled.center    : Named num [1:18] 3356 88594 68698 84405 ...
```

```
  ..- attr(*, "names")= chr [1:18] "Time" "FSC-A" "FSC-H" ...
 $ scaled.scale     : Named num [1:18] 2038 15064 3236 ...
  ..- attr(*, "names")= chr [1:18] "Time" "FSC-A" "FSC-H" ...
 - attr(*, "class")= chr "FlowSOM"
```

## 3 Building the self-organizing map

The next step in the algorithm is to build a self-organizing map. Several parameters for the self-organizing map algorithm can be provided, such as the dimensions of the grid, the learning rate, the number of times the dataset has to be presented. However, the most important parameter to decide is on which columns the self-organizing map should be trained. This should contain all the parameters that are useful to identify cell types, and exclude parameters of which you want to study the behavior on all cell types such as activation markers.

The `BuildSOM` function expects a FlowSOM object as input, and will return a FlowSOM object with all information about the self organizing map added in the $map parameter of the FlowSOM object.

```
> fSOM <- BuildSOM(fSOM,colsToUse = c(9,12,14:18))
> str(fSOM$map,max.level = 2)
```

```
List of 15
 $ xdim        : num 10
 $ ydim        : num 10
 $ rlen        : num 10
 $ mst         : num 1
 $ alpha       :List of 1
  ..$ : num [1:2] 0.05 0.01
 $ radius      :List of 1
  ..$ : num [1:2] 6 0
 $ init        : logi FALSE
 $ distf       : num 2
 $ grid        :'data.frame':       100 obs. of  2 variables:
  ..$ Var1: int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ Var2: int [1:100] 1 1 1 1 1 1 1 1 1 1 ...
  ..- attr(*, "out.attrs")=List of 2
 $ codes       : num [1:100, 1:7] -0.226 -0.397 -0.164 ...
  ..- attr(*, "dimnames")=List of 2
 $ mapping     : num [1:19225, 1:2] 10 99 2 96 42 92 91 ...
 $ nNodes      : int 100
 $ colsUsed    : num [1:7] 9 12 14 15 16 17 18
 $ medianValues: num [1:100, 1:18] 0.138 -0.643 0.256 0.18 ...
```

```
  ..- attr(*, "dimnames")=List of 2
 $ sdValues    : num [1:100, 1:18] 0.966 0.839 0.965 0.971 ...
  ..- attr(*, "dimnames")=List of 2
```

## 4 Building the minimal spanning tree

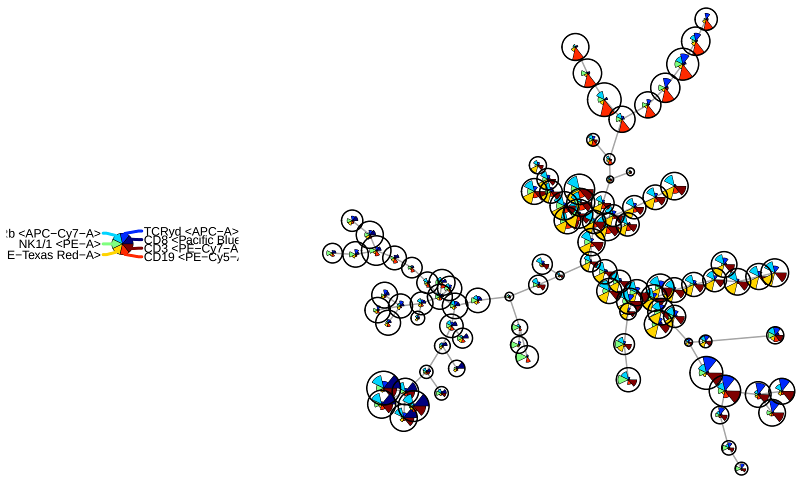
The third step of FlowSOM is to build the minimal spanning tree. This will again return a FlowSOM object, with extra information contained in the `$MST` parameter.

```
> fSOM <- BuildMST(fSOM,tSNE=TRUE)
> str(fSOM$MST)
```

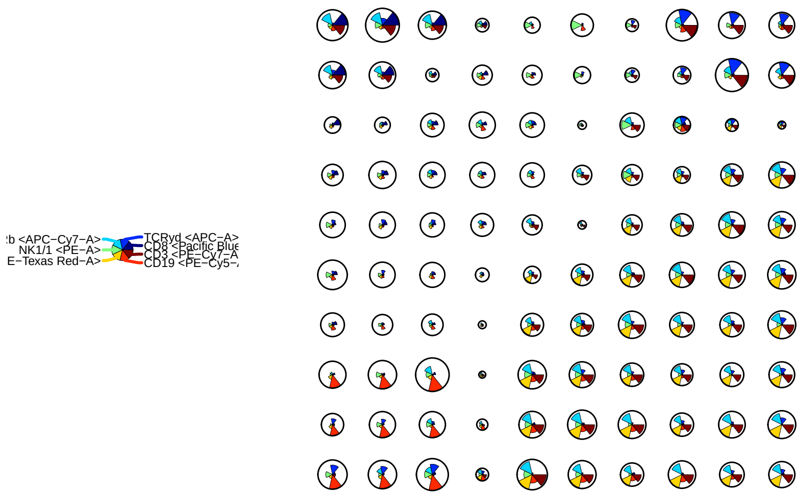
```
List of 4
 $ graph:IGRAPH UNW- 100 99 --
+ attr: name (v/c), weight (e/n)
+ edges (vertex names):
 [1] 1 --3   1 --12  2 --3   2 --11  4 --14  5 --6   ...
 $ l    : num [1:100, 1:2] 1.68495 2.27903 2.02504 ...
 $ l2   : num [1:100, 1:2] -129.5 -175.6 -177.3 106.6 41.5 ...
 $ size : num [1:100] 13 12.62 14.19 5.63 13.52 ...
```

Once this step is finished, the FlowSOM object can be used for visualization. You can plot the nodes in several layouts ("MST": Minimal spanning tree (default),"grid": SOM grid, "tSNE": alternative layout, only possible when tSNE was TRUE in `BuildMST`)

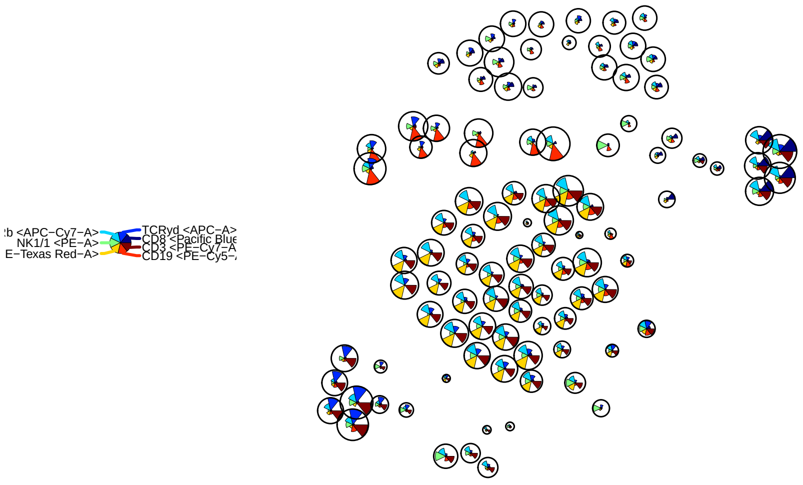
```
> PlotStars(fSOM)
```



```
> PlotStars(fSOM,view="grid")
```

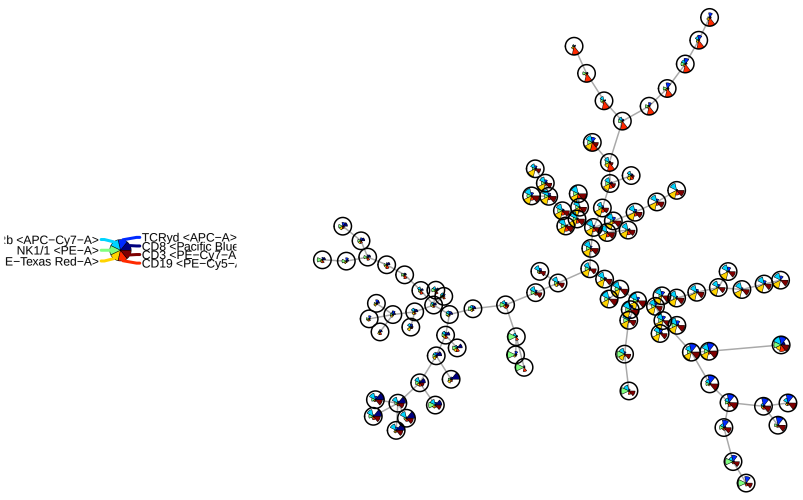


```
> PlotStars(fSOM,view="tSNE")
```



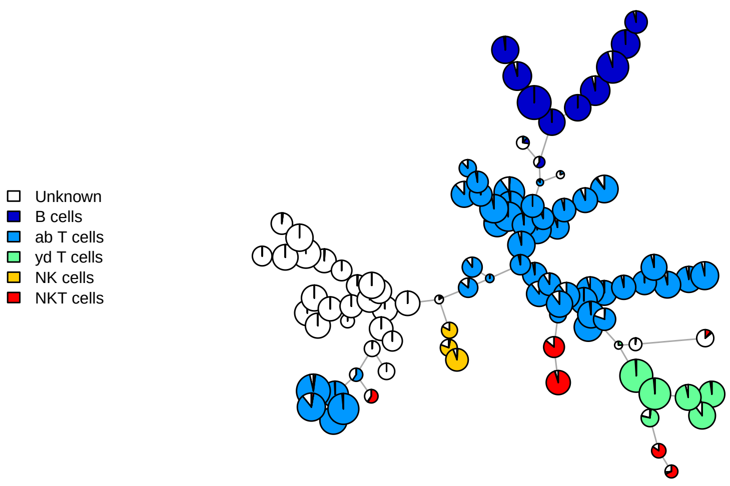
If you do not want the size to depend on the number of cells assigned to a node, you can reset the node size. This can be used in combination with any of the plotting functions.

```
> fSOM <- UpdateNodeSize(fSOM, reset=TRUE)
> fSOM$MST$size <- fSOM$MST$size/2
> PlotStars(fSOM)
> fSOM <- UpdateNodeSize(fSOM)
```



It might also be interesting to compare with a manual gating. The `cellTypes` can be any factor which has a value for each individual cell, so you can also map other clustering results.

```
> #<<>>=
> library(flowUtils)
> flowEnv <- new.env()
> ff_c <- compensate(ff,description(ff)$SPILL)
> colnames(ff_c)[8:18] <- paste("Comp-",
+                                colnames(ff_c)[8:18],
+                                sep="")
> gatingFile <- system.file("extdata","manualGating.xml",
+                          package="FlowSOM")
> gateIDs <- c( "B cells"=8,
+                  "ab T cells"=10,
+                  "yd T cells"=15,
+                  "NK cells"=5,
+                    "NKT cells"=6)
> cellTypes <- names(gateIDs)
> gatingResult <- ProcessGatingML(ff_c, gatingFile,
+                                  gateIDs, cellTypes)
> PlotPies(fSOM,cellTypes=gatingResult$manual)
```

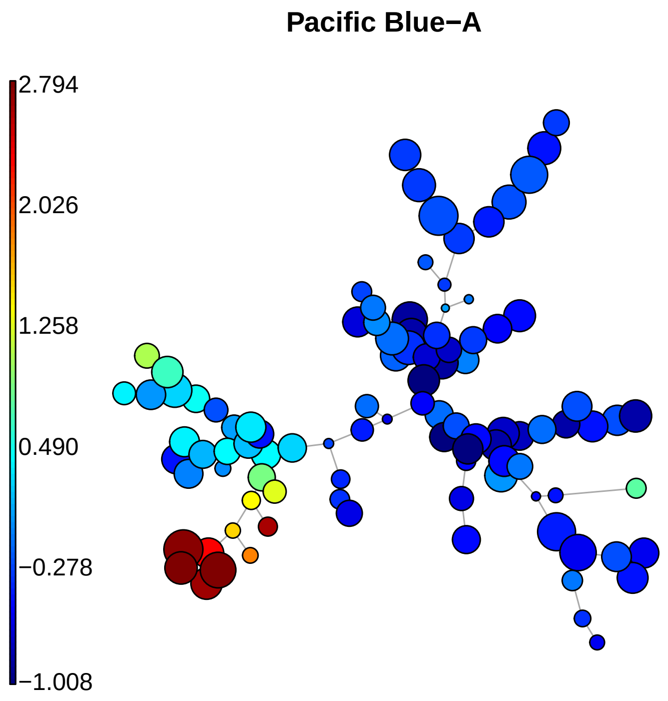


If you are interested in one specific marker, you can use the `PlotMarker` function.

```
> print(colnames(fSOM$map$medianValues))
```

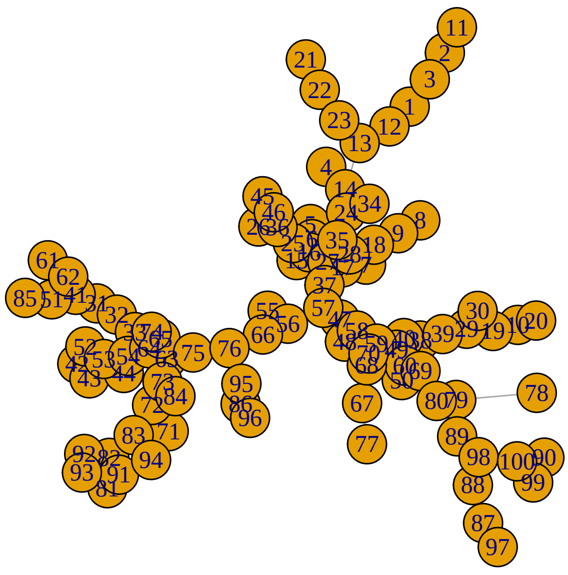
```
              $P1N                $P2N
            "Time"             "FSC-A"
              $P3N                $P4N
           "FSC-H"             "FSC-W"
              $P5N                $P6N
           "SSC-A"             "SSC-H"
              $P7N                $P8N
              "SSC-W"               "FITC-A" 
                 $P9N                  $P10N 
   "Pacific Blue-A"            "AmCyan-A" 
                $P11N                  $P12N 
        "Qdot 605-A"                "APC-A" 
                $P13N                  $P14N 
"Alexa Fluor 700-A"          "APC-Cy7-A" 
                $P15N                  $P16N 
               "PE-A"     "PE-Texas Red-A" 
                $P17N                  $P18N 
          "PE-Cy5-A"            "PE-Cy7-A" 
```

```
> PlotMarker(fSOM,"Pacific Blue-A")
```



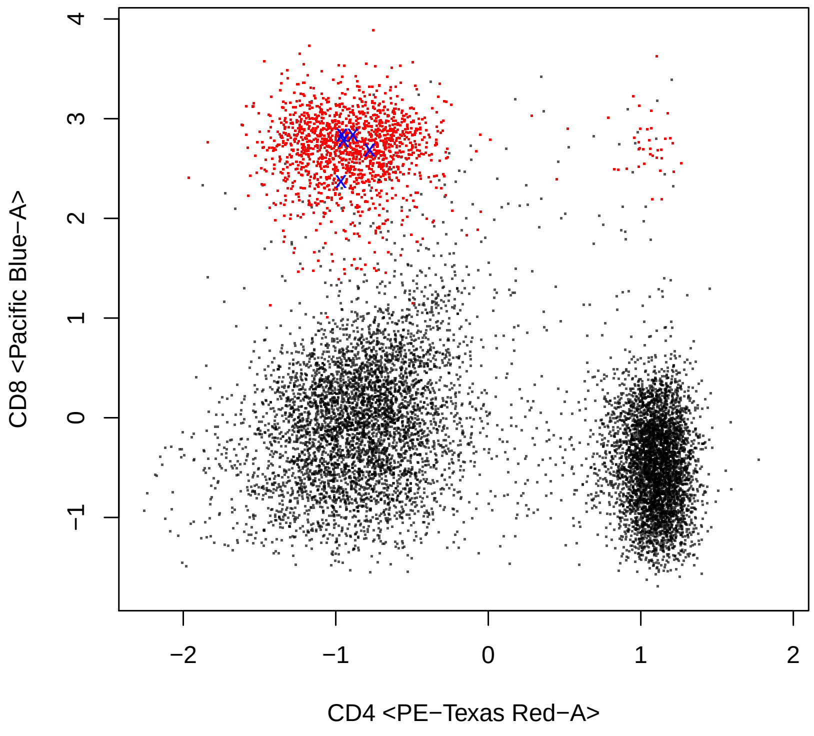
If you need to refer to the nodes, it might be useful to number them.

```
> PlotNumbers(UpdateNodeSize(fSOM,reset=TRUE))
```



You can use this number for a 2D scatter plot indicating the node values.

```
> PlotClusters2D(fSOM,"PE-Texas Red-A","Pacific Blue-A",
+                c(81,82,91,92,93))
```

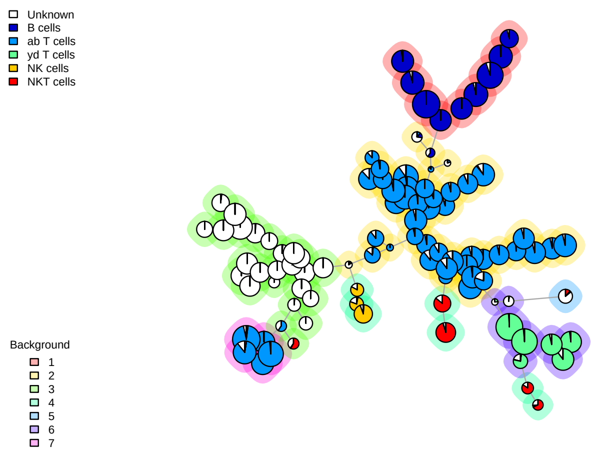


## 5 Meta-clustering the data

The fourth step of the FlowSOM algorithm is to perform a meta-clustering of the data. This can be the first step in further analysis of the data, and often gives a good approximation of manual gating results.

If you have background knowledge about the number of cell types you are looking for, it might be optimal to provide this number to the algorithm.

```
> #<<>>=
> metaClustering <- metaClustering_consensus(fSOM$map$codes,k=7)
> PlotPies(fSOM,cellTypes=gatingResult$manual,
+          backgroundValues = as.factor(metaClustering))
```



You can also extract the meta-clustering for each cell individually

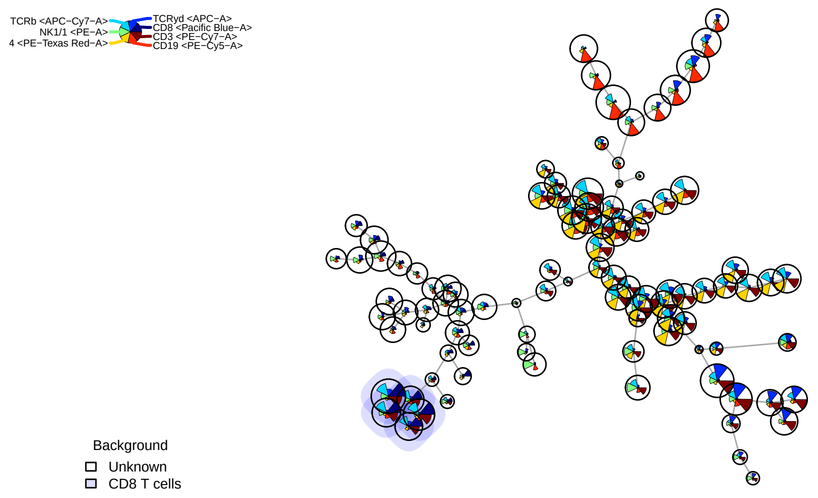
```
> metaClustering_perCell <- metaClustering[fSOM$map$mapping[,1]]
```

## 6 Detecting nodes with a specific pattern

If you do not have a manual gating to map on the tree, it might be time-consuming to interpret all the different nodes. Therefore, you can also query the tree to indicate nodes similar to a specified pattern. This function is still being optimized, so make sure to check the marker values to see if it corresponds to your expectations.

```
> # Look for CD8+ ab T cells
> query <- c("PE-Cy7-A" = "high", #CD3
+            "APC-Cy7-A" = "high", #TCRb
+            "Pacific Blue-A" = "high") #CD8
> query_res <- QueryStarPlot(UpdateNodeSize(fSOM,reset=TRUE),
+                            query,
+                            plot = FALSE)
> cellTypes <- factor(rep("Unknown",49),
+                     levels=c("Unknown","CD8 T cells"))
```

```
> cellTypes[query_res$selected] <- "CD8 T cells"
> PlotStars(fSOM,
+              backgroundValues=cellTypes,
+              backgroundColor=c("#FFFFFF00","#0000FF22"))
```



## 7 Comparing different groups

It is possible to compare between groups with the FlowSOM package as well. The tree should be build on either a concatenation of all files, or a representative subset of all cell types. Then a list identifying which files belong to specific groups should be defined, and the differences will be computed. For a smaller number of samples, you can look at the fold change between the groups. For coloring, a treshold is used. A treshold of 0.50 means the difference should be at least 50% of the max of both groups, which corresponds with a 2-fold change. The higher the threshold, the stricter, a threshold of 0 will colour each node. For a larger number of samples you can also use a wilcox test. This will be selected when a value is provided for the p_tresh parameter.

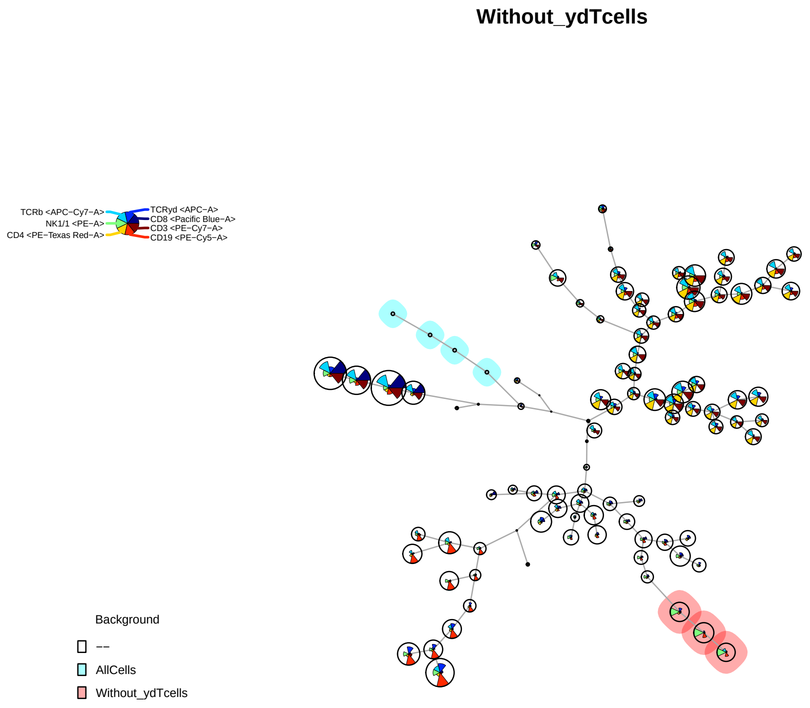
```
> library(FlowSOM)
> # Build the FlowSOM tree on the example file
> fileName <- system.file("extdata","lymphocytes.fcs",
+                          package="FlowSOM")
> flowSOM.res <- FlowSOM(fileName,
+                         compensate=TRUE,transform=TRUE,
+                         scale=TRUE,
+                         colsToUse=c(9,12,14:18),
+                         nClus = 10,
+                         seed=1)
> # Have a look at the resulting tree
> # PlotStars(flowSOM.res[[1]],
> #            backgroundValues = as.factor(flowSOM.res[[2]]))
>
> # Select all cells except the branch that corresponds with
> # automated cluster 7 (CD3+ TCRyd +) and write te another
> # file for the example. In practice you would not generate
> # any new file but use your different files from your
> # different groups
> ff <- flowCore::read.FCS(fileName)
> ff_tmp <- ff[flowSOM.res[[1]]$map$mapping[,1] %in%
+             which(flowSOM.res[[2]] != 7),]
> flowCore::write.FCS(ff_tmp,file="ff_tmp.fcs")

[1] "ff_tmp.fcs"

> # Make an additional file without cluster 7 and double
> # amount of cluster 10
> ff_tmp <- ff[c(which(flowSOM.res[[1]]$map$mapping[,1] %in%
+                  which(flowSOM.res[[2]] != 7)),
+                which(flowSOM.res[[1]]$map$mapping[,1] %in%
+                  which(flowSOM.res[[2]] == 5))),]
> flowCore::write.FCS(ff_tmp,file="ff_tmp2.fcs")

[1] "ff_tmp2.fcs"

> # Compare the original file with the two new files we made
> groupRes <- CountGroups(flowSOM.res[[1]],
+                          groups=list("AllCells"=c(fileName),
+                                      "Without_ydTcells"=
+                               c("ff_tmp.fcs","ff_tmp2.fcs")))

[1] "C:/Users/.../FlowSOM/extdata/lymphocytes.fcs"
[1] "ff_tmp.fcs"
[1] "ff_tmp2.fcs"
```

```
> # PlotGroups(flowSOM.res[[1]], groupRes)
>
> # Compare only the file with the double amount of
> # cluster 10
> groupRes <- CountGroups(flowSOM.res[[1]],
+                          groups=list("AllCells"=c(fileName),
+                           "Without_ydTcells"=c("ff_tmp2.fcs")))
```

```
[1] "C:/Users/.../FlowSOM/extdata/lymphocytes.fcs"
[1] "ff_tmp2.fcs"
```

```
> PlotGroups(flowSOM.res[[1]], groupRes)
```

```
$Without_ydTcells
  [1] --               --               --               --
  [6] --               --               --               --
 [11] --               --               --               --
 [16] --               --               --               --
 [21] --               --               --               --
 [26] --               --               --               --
 [31] --               --               --               --
 [36] --               --               --               --
 [41] --               --               --               --
 [46] --               --               Without_ydTcells --
 [51] AllCells         AllCells         --               --
 [56] --               Without_ydTcells Without_ydTcells --
 [61] AllCells         AllCells         --               --
 [66] --               --               --               --
 [71] --               --               --               --
 [76] --               --               --               --
 [81] --               --               --               --
 [86] --               --               --               --
 [91] --               --               --               --
 [96] --               --               --               --
Levels: -- AllCells Without_ydTcells
```



## 8 Summary

In summary, the FlowSOM package provides some new ways to look at cytometry data. It can help to keep an overview of how all markers are behaving on different cell types, and to reduce the probability of overlooking interesting things that are present in the data.