Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Jan-Christoph Meier

Konzeption und Entwicklung eines Systems zur automatisierten Clusteranalyse von Daten aus der Durchflusszytometrie

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan-Christoph Meier

Konzeption und Entwicklung eines Systems zur automatisierten Clusteranalyse von Daten aus der Durchflusszytometrie

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Master of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Wolfgang Gerken

Eingereicht am: 24.09.2013

Jan-Christoph Meier

Thema der Arbeit
Konzeption und Entwicklung eines Systems
zur automatisierten Clusteranalyse
von Daten aus der Durchflusszytometrie

Stichworte
Data Mining, Clusteranalyse, Durchflusszytometrie, FACS, R-Programmierung, Bioconductor, Django Webframework, Python

Kurzzusammenfassung
Gegenstand dieser Arbeit ist die Entwicklung eines automatisierten Analyseverfahrens für Daten aus der Durchflusszytometrie. Das Verfahren basiert auf verschiedenen Cluster-Algorithmen, mit denen Häufigkeiten von Zellpopulationen in den Daten bestimmt werden. Grundlage für das Analyseprogramm ist ein Framework, das in der Programmiersprache R entwickelt wurde.

Jan-Christoph Meier

Title of the paper
Development of a software platform for
automated cluster analysation of FACS data

Keywords
Data Mining, Cluster analysis, fluorescence-activated cell sorting, FACS, R-Programming, Bioconductor, Django Webframework, Python

Abstract
Topic of this paper is the development of a software platform to analyse facs data with clustering algorithms. The software depends on a framework which has been built in R and which combines reusable components for analysing facs data.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1. Einleitung

Am Zentrum für Molekulare Neurobiologie Hamburg (ZMNH[1]), das zum Universitätsklinikum Hamburg Eppendorf gehört, werden Blutproben von Multiple Sklerose erkrankten Personen mithilfe der Durchflusszytometrie untersucht. Multiple Sklerose ist eine Autoimmunkrankheit, bei der körpereigene Zellen als fremd erkannt werden und durch das Immunsystem angegriffen werden. Die Ursachen für eine Multiple Sklerose Erkrankung sind bis zum jetzigen Zeitpunkt nicht endgültig erforscht.
Mit der Durchflusszytometrie können Zellpopulationen im Blut bestimmt werden, wodurch festgestellt werden kann, ob Medikamente wirken oder ob es zu einer Verbesserung oder Verschlechterung des aktuellen Krankheitszustandes gekommen ist. Dieses Verfahren wird auch mit dem englischen Begriff »Fluorescence Activated Cell Sorting« (kurz FACS) bezeichnet.
Bei der Durchflusszytometrie wird das Blut durch eine dünne Messkammer geleitet und mit einem Laser bestrahlt. Das Laserlicht wird gemessen und die Signale werden ausgewertet, um die Häufigkeiten verschiedener Zelltypen im Blut zu bestimmen. Die jeweiligen Zelltypen werden mit spezifischen Fluoreszenzmarkern versehen und können aufgrund der unterschiedlichen Wellenlängen des Fluoreszenzlichtes unterschieden werden. Die Messungen werden mit einem sogenanntes Durchflusszytometer durchgeführt, der mit einem Computer gekoppelt ist. Dieser erfasst die Messung und speichert sie in einem standardisierten Dateiformat namens FCS ab.
Ausgewertet werden die Daten durch einen geschulten Mitarbeiter mit einer speziellen Software, diese kann zum Beispiel FACSDiva[2] oder Flowjo[3] sein. Hierbei werden die gemessenen Signale in sogenannten Dotplots so gegeneinander aufgetragen, dass die gesuchten Informationen anhand von Häufungen bzw. Cluster identifiziert werden können. Die Cluster repräsentieren die unterschiedlichen Zellpopulationen, die im Blut gemessen wurden. Problem hierbei ist, dass die manuellen Auswertungen fehleranfällig und zeitaufwändig sind. Der am ZMNH vorhandene Datenbestand von Messdaten aus der Durchflusszytometrie hat

[1] http://www.zmnh.uni-hamburg.de abgerufen am 17.09.2013
[2] http://www.bdbiosciences.com/eu/instruments/software/facsdiva/index.jsp abgerufen am 15.07.2013
[3] http://www.flowjo.com/ abgerufen am 17.09.2013

mittlerweile einen Umfang von einem Terabyte, wovon jedoch ein Großteil der Daten nicht ausgewertet wurde.

1.1. Motivation

Zurzeit werden FACSDiva und FlowJo für die Datenanalyse eingesetzt. Diese beiden Programme bieten keine Möglichkeit, die Auswertungen zu automatisieren. Jede Analyse wird durch einen Fachanwender manuell durchgeführt, wobei in mehreren Arbeitsschritten die Zellpopulationen identifiziert werden. Die Daten sind teilweise sehr unterschiedlich, wodurch die exakte Position und Größe der Cluster nicht vordefiniert werden können. Hierdurch werden die Cluster durch jede Person individuell gewählt, was zur Folge hat, dass die Analyseergebnisse nicht exakt reproduzierbar sind.
Die aktuelle Datenorganisation bietet keine Möglichkeit, FCS-Dateien anhand bestimmter Kriterien, wie zum Beispiel der gemessenen Zellpopulationen zu finden. Jede FCS-Datei muss mit der Analysesoftware geöffnet und es muss geprüft werden, ob die Datei den Kriterien entspricht. Dies ist sehr zeitaufwändig, da das Öffnen einer Datei, trotz eines Leistungsstarken Rechners, mehrere Sekunden in Anspruch nimmt.
Bevor eine Datei analysiert werden kann, muss geprüft werden, ob sie verschiedene Qualitätsmerkmale erfüllt. Zwei Qualitätsmerkmale sind beispielsweise, dass die Datei eine minimale Anzahl an Signalen enthält und alle Zelltypen, die später analysiert werden sollen, gemessen wurden. Eine weitere Herausforderung besteht darin, dass die Signale der gleichen Zelltypen in den Dateien teilweise unterschiedlich benannt wurden.
Wünschenswert wäre ein standardisiertes und automatisiertes Analyseverfahren, das reproduzierbare Ergebnisse liefert und den hohen Arbeitsaufwand der Analyse verringert. Eine Schwierigkeit hierbei ist, dass die Daten sehr unterschiedlich sind, wodurch für die Analyse verschiedene Algorithmen aus dem Bereich Data Mining eingesetzt werden müssen. Von den Fachanwendern wurde bisher davon ausgegangen, dass eine automatisierte Bestimmung der Zellpopulationen nicht möglich ist.

1.2. Ziele

Das Ziel der Masterarbeit ist es, zu prüfen, ob es möglich ist, die Auswertung der Messdaten mit einer Software zu automatisieren. Hierfür soll für eine spezifische wissenschaftliche Fragestellung ein Algorithmus entwickelt werden, der definierte Zellpopulationen in den Daten identifizieren kann. Das entwickelte Verfahren soll trotz der Vielfältigkeit der Daten

zuverlässig arbeiten und korrekte Werte für die Zellpopulationen liefern.
Dieser »Proof of concept« soll als Grundlage für weitere Analyseprogramme dienen und nach Möglichkeit so generisch implementiert sein, dass es in Zukunft auch möglich ist, verschiedene weitere Zellpopulationen in den Daten zu identifizieren.
Sofern es gelingt, die Daten automatisiert zu analysieren, kann das Verfahren eingesetzt werden, um den großen Datenbestand am ZMNH auszuwerten. Hierbei sind völlig neue Erkenntnisse für die Multiple Sklerose Forschung denkbar.

2. Grundlagen

In diesem Kapitel wird ein Überblick über verschiedene Themenbereiche gegeben, die für die Masterarbeit relevant sind.
Zu Beginn des Abschnittes werden verschiedene medizinische Grundlagen und die Durchflusszytometrie erläutert. Im weiteren Verlauf wird sich mit verschiedenen mathematischen Grundlagen beschäftigt, die für die Erläuterung unterschiedlicher Clustering-Algorithmen verwendet werden. Im Anschluss werden zwei Algorithmen betrachtet, die speziell für die Analyse von Daten aus der Durchflusszytometrie entwickelt wurden.
Abschließend werden die Vorteile der beiden Programmiersprachen R und Python dargelegt und das Bioconductor-Projekt, sowie das Webframework Django präsentiert.

2.1. Medizinische Grundlagen

Die Zellen im Blut des Menschen können in drei verschiedener Typen unterschieden werden (vgl. Fanghänel u. a. (2009) - Abschnitt 2.4). Für den Sauerstofftransport sind die roten Blutkörperchen namens Erythrozyten zuständig. Die Thrombozyten sind für die Blutgerinnung wichtige Bluttplättchen. Die dritte Gruppe sind die Leukozyten, die für die Immunabwehr des menschlichen Körpers verantwortlich sind. Für die Multiple Sklerose Forschung sind die Leukozyten von besonderem Interesse, da sie bei einer MS-Erkrankung eine Fehlreaktion hervorrufen, durch die körpereigene Zellen vernichtet werden.
Die Leukozyten können in Granulozyten, Monozyten und Lymphozyten unterschieden werden. In der Abbildung 2.1 sind die verschiedenen Untergruppen der Leukozyten hierarchisch dargestellt.

2.1.1. Granulozyten und Monozyten

Die Granulozyten sind die zahlenmäßig größte Untergruppe der Leukozyten. Sie sind wichtig für die Abwehr von Infektionen, die durch Bakterien, Pilze oder Parasiten verursacht werden. Die Monozyten sind die Zellen mit dem größten Volumen, sie haben eine Lebensdauer von

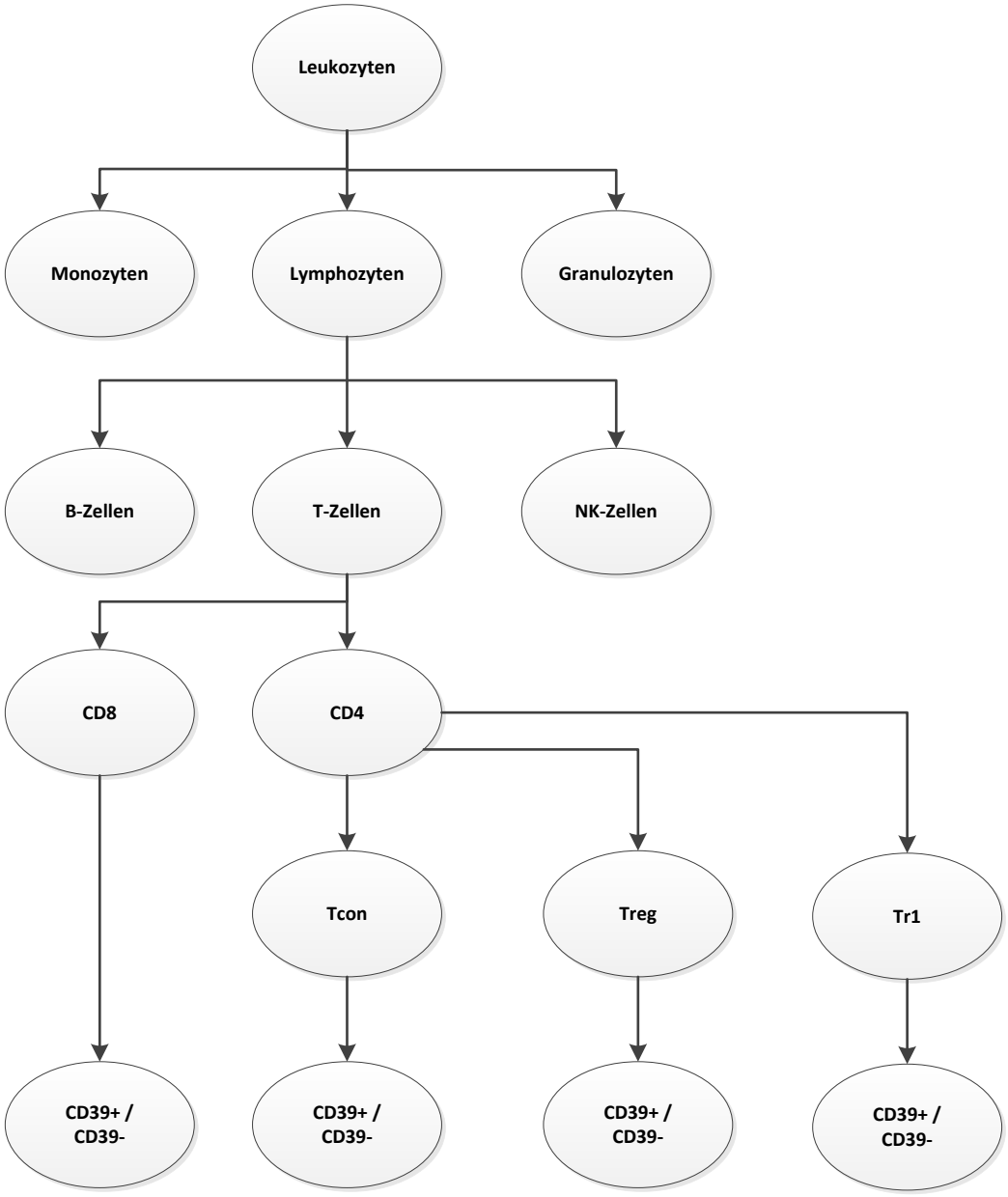

Abbildung 2.1.: Hierarchie der verschiedenen Zelltypen

nur einem bis drei Tagen. Danach verwandeln sie sich in sogenannte »Fresszellen«, deren Aufgabe es ist, fremde Stoffe im Blut zu eleminieren.

2.1.2. Lymphozyten

Die Lymphozyten sind Bestandteil des Immunsystems und werden in B-, T- und NK-Zellen unterschieden.
B-Lymphozyten produzieren die Abwehrstoffe, die als Antikörper bezeichnet werden. Die Antikörper haften an körperfremden Stoffen, wie zum Beispiel den Erregern von Viren, und verhindern, dass diese sich weiter ausbreiten. Der hierbei enstehende Komplex aus Antikörper und Erreger kann dann von Fresszellen aufgenommen und vernichtet werden.
Die natürlichen Killerzellen (NK-Zellen) können virusinfizierte Zellen oder Krebszellen erkennen und diese zerstören.
T-Helferzellen detektieren körperfremde Stoffe im Blut und lösen eine Immunreaktion aus.
Bei einer Entzündung oder Infektion erhöht sich die Anzahl der Lymphozyten im Blut, was durch einen Arzt für die Diagnose verschiedener Krankheiten verwendet werden kann. Anhand der Anzahl der Lymphozyten im Blut kann der aktuelle Zustand der Körperabwehr diagnostiziert werden. Krankheiten wie Krebs oder AIDS führen zu einer geringen Anzahl an Lymphozyten.
Bei der Erforschung von Autoimmunerkrankungen wie Multiple Sklerose wird die Verteilung der verschiedenen Zellpopulationen im Blut von erkrankten Personen untersucht, um zu prüfen, ob Medikamente wirken oder den Verlauf der Krankheit zu beobachten.

2.1.3. CD-Zelltypen

Zellen können anhand verschiedener Merkmale ihrer Oberfläche in weitere Gruppen unterschieden werden. Diese Untergruppen werden als »Cluster of Differentiation« (kurz CD) bezeichnet, wovon es mehr als 200 verschiedene Gruppen gibt.
Für die Multiple Sklerose Forschung sind die CD-Untergruppen CD4, CD8 und CD39 (siehe Abbildung 2.1) von besonderem Interesse.
Weitere Untergruppen der CD4 Zellen sind die Regulatorischen-T-Zellen (kurz Treg), die Tr1- und Tcon-Zellen.

2.2. Einführung in die Durchflusszytometrie

Die Durchflusszytometrie ist ein Messverfahren, mit dem die Häufigkeiten von Zellpopulationen im Blut bestimmt werden können (vgl. Sack u. a. (2006)). Hierfür wird Blut durch eine dünne Messkammer geleitet und mit einem Laser bestrahlt. Die Messkammer ist so dünn, dass die einzelnen Zellen im Blut hintereinander aufgereiht diese passieren. Die Lasersignale werden dabei mit verschiedenen Detektoren gemessen, die die Stärke des Vorwärts- und Seitwärtsstreulichts, sowie verschiedene Spektren bzw. Wellenlängen des Laserlichts messen. Die unterschiedlichen Detektoren werden auch als »Kanäle« bezeichnet, da das Laserlicht durch Kanäle zu verschiedenen Dioden geleitet wird, die dann die Signalstärke des jeweiligen Lichtspektrums messen.
Pro Zelle wird das Signal des Lasers, abhängig vom eingesetzten Durchflusszytometer, mit bis zu 17 verschiedenen Detektoren gemessen (vgl. Perfetto u. a. (2004)).

2.2.1. Vorwärts- und Seitwärtstreulicht

In der Abbildung 2.2 ist der Aufbau für die Messung des Seitwärts- und Vorwärtsstreulichts schematisch dargestellt. Mit dem Vorwärtsstreulicht (engl. »Forward Scatter«, kurz FSC) wird die Größe der Zelle bestimmt, bei großen Zellen wird ein starkes Signal des Vorwärtsstreulichts gemessen. Anhand des Seitwärtsstreulichts (engl. »Side Scatter«, kurz SSC) wird die Granularität (Körnigkeit) der Zelle bestimmt. Bei Zellen, die eine hohe Granularität haben, wird ein starkes Signal des Seitwärtsstreulichts gemessen.

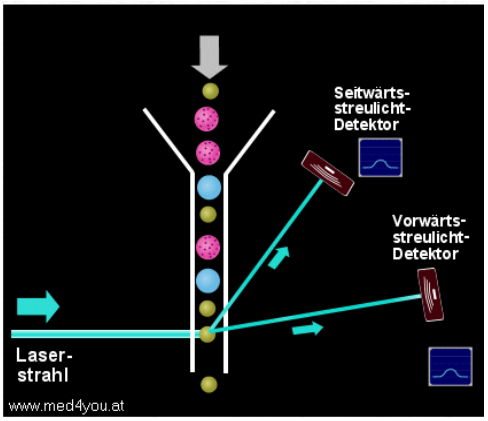

Abbildung 2.2.: Schematik der Messung des Seitwärts- und Vorwärtstreulichts
Quelle: Hübl (2013)

Die gemessen Signale werden zur Auswertung als Dotplot dargestellt, wobei die Signale von

zwei Detektoren verwendet werden. Die Abbildung 2.3 zeigt einen Dotplot, in dem die Signale der Detektoren für das Seitwärts- und Vorwärtsstreulicht aufgetragen sind. Jeder Punkt entspricht einer Zelle, die gemessen wurde. Anhand der gemessen Signalstärken können die unterschiedlichen Zelltypen im Blut identifiziert werden. Diese werden anhand der Größe und der Granularität unterschieden.

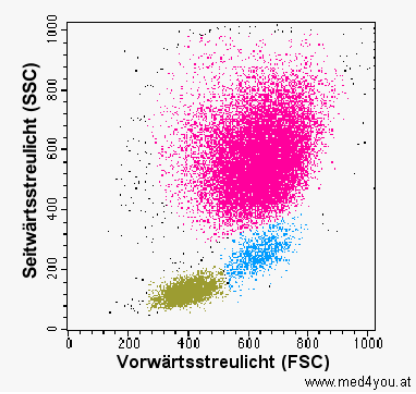

Abbildung 2.3.: Signale des Vorwärts- und Seitwärtsstreulicht als Dotplot - Quelle: Hübl (2013)

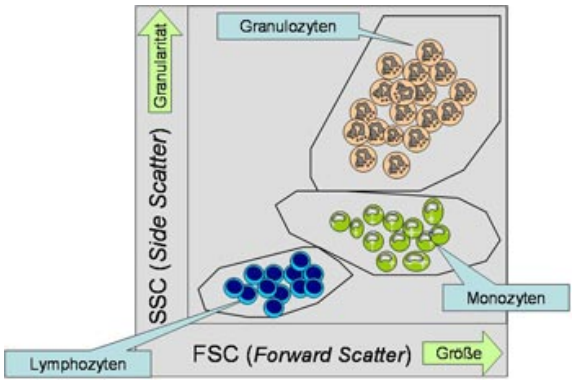

Abbildung 2.4.: Verschiedene Zelltypen im Dotplot - Quelle Hübl (2013)

Lymphozyten sind klein und haben eine geringe Granularität. Sie sind in dem Dotplot in Abbildung 2.3 braun dargestellt. Die Monozyten sind in dem Plot blau dargestellt, sie sind groß und haben eine geringe Granularität. Die am häufigsten vorkommenden Zellen sind die Granulozyten. Diese sind in dem Dotplot rosa dargestellt. Sie erzeugen auf beiden Kanälen ein starkes Signal.
In Abbildung 2.4 ist dargestellt, wie die unterschiedlichen Zelltypen im Dotplot identifiziert werden können.

2.2.2. Fluoreszenzsignale und Marker

Damit die Häufigkeiten bestimmter Zelltypen im Blut gemessen werden können, wird dem Blut ein Antigen hinzugefügt, das mit einem fluoreszierendem Stoff eingefärbt ist. Diese Antigene werden auch als Marker bezeichnet, da sie bestimmte Zellen »markieren«. Das Antigen haftet an der Oberfläche eines bestimmten Zelltyps, der, sobald er mit dem Laser bestrahlt wird, durch den fluoreszierenden Stoff farbiges Licht erzeugt. Jeder Zelltyp, an dem ein eingefärbtes Antigen haftet, erzeugt ein unterschiedliches Farbspektrum des Laserlichts, anhand dessen die verschiedenen Zellpopulationen im Blut bestimmt werden.
Um die Häufigkeit der CD8-Zellen (siehe Abschnitt 2.1.3) im Blut festzustellen, wird der Marker »FITC« zum Blut hinzugefügt. Er ist mit einem Fluoreszenzmarker eingefärbt, der blaues Laserlicht erzeugt. Mit dem Marker »APC« kann die Anzahl der CD4-Zellen (siehe Abschnitt 2.1.3) im Blut gemessen werden, er erzeugt rotes Laserlicht. Zur Auswertung werden die gemessenen Signale der beiden Marker in einem Dotplot dargestellt (Abildung 2.5), dessen Achsen in einer logarithmischen Skala aufgetragen sind. Diese Darstellung hat den Vorteil, dass sehr starke Signale nahe beieinander sind und so Cluster gefunden werden können, anhand derer die Häufigkeiten der Zellpopulationen abgeleitet werden.
Die Abbildung 2.5 zeigt die gemessenen Signale der Marker FITC und APC. In dem Dotplot sind insgesamt drei Cluster von Datenpunkten zu erkennen, die mit den Zahlen 1, 2 und 3 gekennzeichnet sind. In dem Cluster mit der Nummer 2 sind Signale mit hohen Werten von FITC und geringen Werten für APC enthalten. Die Signale wurden somit von CD8-Zellen erzeugt, deren Häufigkeit anhand der Anzahl der Datenpunkte in dem Cluster bestimmt wird. Das Cluster mit der Nummer 3 enthält hohe Signale von APC und geringe Signale von FITC, somit handelt es sich hierbei um CD4-Zellen. In dem Cluster mit der Nummer 1 sind Signale enthalten, die auf den Kanälen FITC und APC geringe Signalstärken haben. Hierbei handelt es sich somit um andere Zelltypen, die für die Auswertung der CD4- und CD8-Zellen nicht relevant sind.

2.2.3. FCS-Dateiformat

Die mit einem Durchflusszytometer erzeugten Messdaten werden in einem standardisierten Dateiformat namens »FCS« abgespeichert (vgl. Seamer (2001)). In einer FCS-Datei werden verschiedene Parameter der Messung hinterlegt, unter anderem der Name der Blutprobe, die Namen der Fluoreszenzmarker und das Datum der Messung.
Wichtigster Bestandteil der Datei sind die gemessenen Signale des Lasers. Diese werden in tabellarischer Form in der FCS-Datei hinterlegt. Die Tabelle 2.2.3 zeigt einen Auszug hiervon.

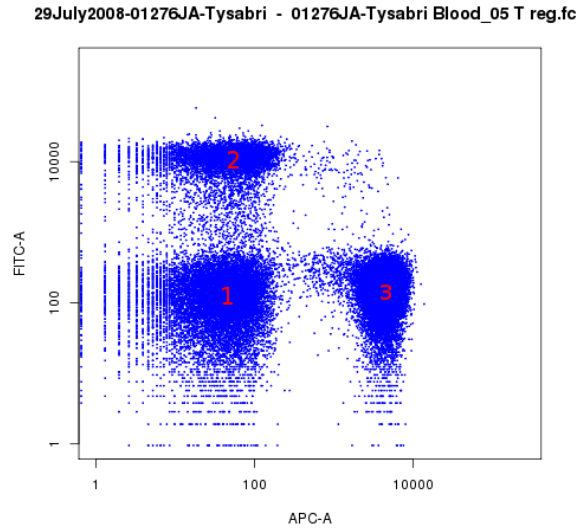

Abbildung 2.5.: Signale der Kanäle APC und FITC

Time	FSC-A	SSC-A	FITC-A	PE-A	APC-A
0	45827.68	12855.399	22.8	4.75	21.449999
0.5	59075.98	29868.949	12122	2184.05	68.899994
0.6	53085.22	46936.648	-58.9	-178.6	17.549999
0.7	105998.45	27432.199	222.3	159.6	-21.449999
0.9	56983.6	13787.35	13799.7	2698	139.099991
1	44372.56	14269.95	327.75	754.3	4472.649902
1.3	61854.64	13281.95	-12.35	-56.05	16.25
1.4	87448.24	24013.15	231.8	604.2	7112.949707
1.5	46937.08	19694.449	89.3	19.95	3404.049805
1.8	93857.82	23082.15	-2.85	-19.95	71.5

Tabelle 2.1.: Messdaten einer FCS-Datei

Jede Zeile enthält die Signale, die von einer einzelnen Zelle erzeugt wurden. Die erste Spalte der Tabelle beinhaltet den relativen Zeitpunkt der Signalerfassung in Abhängigkeit zum Start der Messung. Die weiteren Spalten zeigen die Signale der unterschiedlichen Fluoreszenzmarker, die dem Blut hinzugefügt wurden. In dem Beispiel wurden insgesamt fünf verschiedene Zelltypen gemessen.

2.2.4. Kompensation

Die Wellenlängen des Laserlichts überlappen sich, wie in der Abbildung 2.6 veranschaulicht wird. Bei der Messung der Signalstärken der unterschiedlichen Lichtspektren kann es zu Verfälschungen kommen, sofern Fluoreszenzmarker dem Blut hinzugefügt wurden, deren Wellenlängen sich überschneiden.

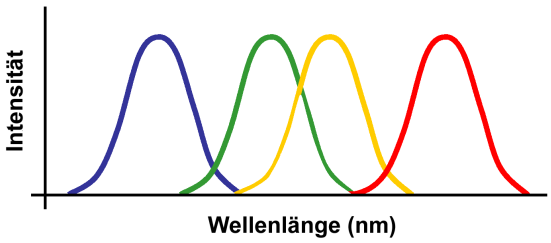

Abbildung 2.6.: Überlappung der Wellenlängen

Um diese Messfehler zu korrigieren werden die Signale von Lichtspektren, deren Wellenlängen sich überlappen, voneinander abgezogen. Wenn zum Beispiel der reale Wert vom Kanal $K1$ bestimmt werden soll, die Wellenlängen sich jedoch mit dem Kanal $K2$ überschneiden, wird folgende Formel angewendet:

$$K1_{real} = K1_{gemessen} \cdot P\% \cdot K2_{gemessen} \tag{2.1}$$

Es wird also der gemessene Wert vom Kanal $K1$ um $P\%$ vom gemessenen Wertes des Kanals $K2$ verringert. Der Wert von P wird festgelegt, indem die gemessenen Signale von zwei überlappenden Lichtspektren in einem Dotplot aufgetragen werden. P wird dann manuell angepasst, bis zwei Cluster von Signalen in dem Dotplot klar identifiziert werden können.

2.2.5. Analyse der Daten / Gating

Zur Auswertung der Messdaten wird ein mehrstufiges Verfahren angewendet, mit dem die Zellpopulationen bestimmt werden. Im Kapitel 2.1 wurde erläutert, dass Zelltypen hierarchisch in Gruppen eingeteilt werden können.
In der Abbildung 2.7 ist die Gruppenzugehörigkeit am Beispiel der Treg-Zellen dargestellt. Die Häufigkeit der Treg-Zellen wird aus einer Messung abgeleitet, indem zuerst die Lymphozyten identifiziert werden. Im nächsten Schritt werden die T-Zellen und darauffolgend die CD4-Zellen identifiziert. Aus den CD4-Zellen wird dann die Anzahl der Treg-Zellen bestimmt.

Abbildung 2.7.: Gruppenzugehörigkeit der Treg-Zellen

Dies Vorgehen wird als »Gating« bezeichnet. Wie in Abschnitt 2.2.2 beschrieben wurde, können bestimmte Zelltypen anhand von Dotplots identifiziert werden, in denen die Signale von Fluoreszenzmarkern aufgetragen sind. Beim Gating werden diese Dotplots eingesetzt, um in mehreren Schritten Cluster von Signalen zu selektieren und die Messung anhand der Selektion zu filtern. Die Abbildung 2.8 veranschaulicht das Vorgehen. Aus dem Plot der APC- und FITC-Signale wird das Cluster mit hohen Werten bei FITC selektiert und diese Signale extrahiert, was in Schritt 2 der Abbildung dargestellt ist. Im Schritt 3 werden die PE-A- und PerCP-Cy5-Signale der zuvor selektierten Zellen im Dotplot dargestellt, aus denen dann wiederum Zellen selektiert werden. Dies wird wiederholt, bis die Signale des Zelltyps extrahiert wurden, der in den Daten gefunden werden soll.
Für das Gating der FCS-Dateien werden am ZMNH die Programme FacsDiva[1] und FlowJo[2] eingesetzt.

[1] http://www.bdbiosciences.com/instruments/software/facsdiva/
[2] http://www.flowjo.com/

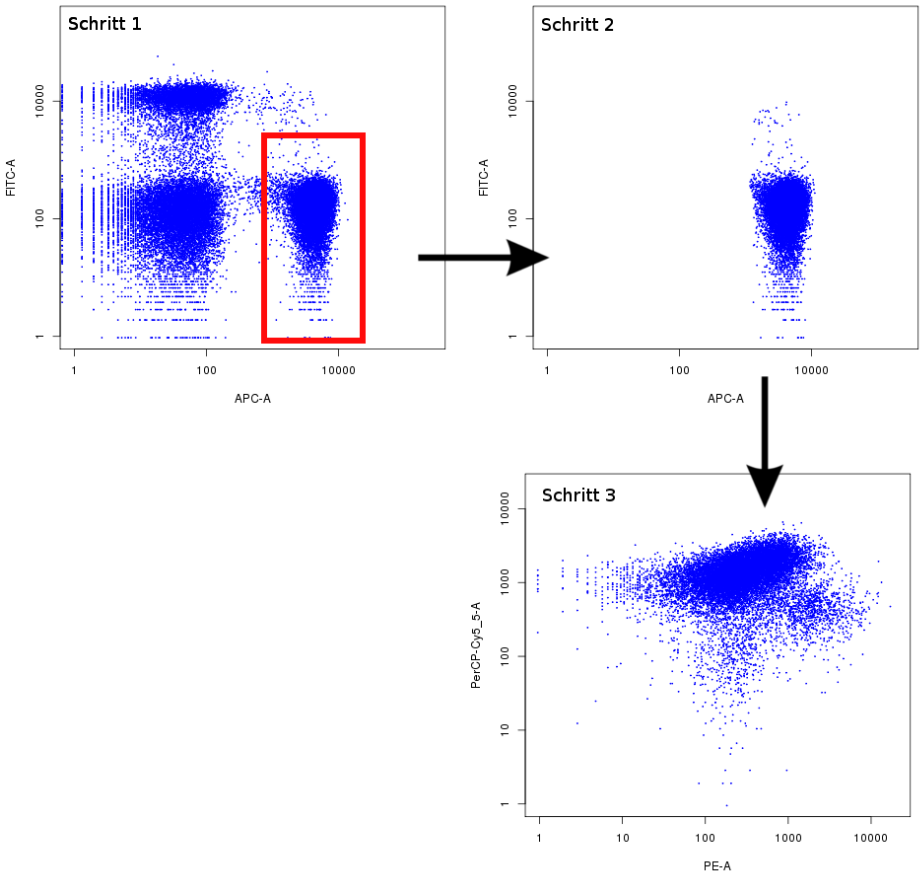

Abbildung 2.8.: Schematik des Gatings

2.3. Mathematische Grundlagen

In diesem Kapitel werden verschiedene mathematische Grundlagen aus dem Bereich Statistik erläutert.

2.3.1. Erwartungswert

Unter dem Erwartungswert versteht man eine Zahl, die eine Zufallsvariable im Mittel annimmt (vgl. Fischer (2005), Kapitel 2.7).
Sei folgende Menge mit Ergebnissen gegeben:

$$\Omega := \omega_1, \ldots, \omega_n \tag{2.2}$$

Dann ist die Wahrscheinlichkeit P für jedes $\omega \in \Omega$ folgendermaßen definiert:

$$P(\omega) \text{ mit } 0 \leq P(\omega) \leq 1 \tag{2.3}$$

Der Erwartungswert einer Zufallsvariablen $X : \Omega \to \mathbb{R}$ ist folgende Zahl:

$$E(X) := \sum_{\omega \in \Omega} X(\omega) \cdot P(\omega) \tag{2.4}$$

2.3.2. Varianz und Covarianz

Die Varianz ist ein Maß für die Streuung einer Zufallsvariablen $X : \Omega \to \mathbb{R}$, die folgendermaßen definiert ist mit $\mu := E(X)$:

$$Var(X) := \sum_{\omega \in \Omega} (X(\omega) - \mu)^2 \cdot P(\omega) \tag{2.5}$$

Die Standardabweichung ist basierend auf der Varianz folgendermaßen definiert:

$$\sigma := \sqrt{Var(X)} \tag{2.6}$$

Die Covarianz von zwei Variablen X und Y ist folgendermaßen definiert:

$$Cov(X, Y) = E((X - \mu) \cdot (Y - \nu)) \tag{2.7}$$

Wobei $\mu = E(X)$ und $\nu = E(Y)$ gilt.

2.3.3. Wahrscheinlichkeitsverteilungen

Unter der Wahrscheinlichkeitsverteilung (oder Wahrscheinlichkeitsfunktion) einer Zufallsvariablen X versteht man die Funktion f deren Funktionswert $f(x_i)$ die Wahrscheinlichkeit angibt, mit der die Variable X den Wert x_i annimmt:

$$f : x_i \rightarrow P(X = x_i) \tag{2.8}$$

Gauß-Verteilung

Die Normalverteilung oder auch Gauß-Verteilung (siehe Fischer (2005), Kapitel 2.8) ist eine Verteilungsfunktion, deren grafische Darstellung eine Glockenkurve ergibt (siehe Abbildung 2.9). Die Dichtefunktion ϕ der Gauß-Verteilung hat folgende Berechnungsvorschrift:

$$\phi(x \mid \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \cdot \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\} \tag{2.9}$$

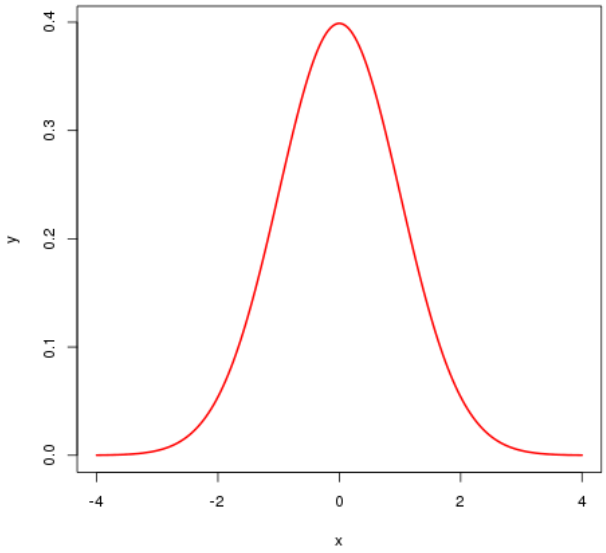

Abbildung 2.9.: Gauß-Verteilung

Poisson-Verteilung

Die Poisson-Verteilung ist eine diskrete, einparametrige Verteilung, deren einziger Parameter λ ist.
Die Wahrscheinlichkeitsfunktion der Poisson-Verteilung hat folgende Berechnungsvorschrift:

$$p(x|\lambda) = \begin{cases} \frac{e^{-\lambda} \cdot \lambda^x}{x!} & \text{für x=0,1,...} \\ 0 & \text{sonst} \end{cases} \tag{2.10}$$

Die Verteilungsfunktion der Poisson-Verteilung ist folgendermaßen definiert:

$$F(x) = \sum_{x=0}^{n} \frac{e^{-\lambda} \cdot \lambda^x}{x!} \tag{2.11}$$

2.3.4. Lineare Regression

Die Grundidee der linaeren Regression ist es, eine Gerade zu finden, die einen Zusammenhang zwischen zwei Variablen x und y darstellen kann. Gegeben sei eine Vielzahl von Zahlenpaaren $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_n, y_n)$. Für diese soll eine lineare Funktion bestimmt werden, mit der $x \rightarrow y$ beschrieben werden kann. Somit soll eine Geradengleichung $f(x) = ax + b$ gefunden werden, so dass $y_i \approx f(x_i)$ gilt.
Wenn die Punktepaare (x_i, y_i) in einem X-Y-Koordinatensystem eingetragen werden, erhält man eine Punktewolke, die in der Abbildung 2.10 dargestellt ist. Mit der linearen Regression wird eine Gerade gesucht, der alle Punkte so nah wie möglich sind. Die Abbildung 2.11 zeigt für die gegebene Punktewolke die als Regressionsgerade bezeichnete Gerade. Die Regressionsgerade wird durch Berechnung der Steigung a und dem Y-Achsen-Abschnitt b bestimmt. Da es sich nicht um eine empirisch bestimmte Geradengleichung handelt wird zur Beschreibung der Geradengleichung folgendes Modell verwendet:

$$\hat{y} = b_0 + b_1 x_i \tag{2.12}$$

Ziel bei der Berechnung der Geradengleichung ist, die Differenzen zwischen den empirischen Werten und den der Geradengleichung $f(x_i) - y_i$ möglichst zu minimieren. Günstig hierfür ist es die Quadrate dieser Abweichung zu betrachten:

$$\sum_{i=1}^{n} (f(x_i) - y_i)^2 \tag{2.13}$$

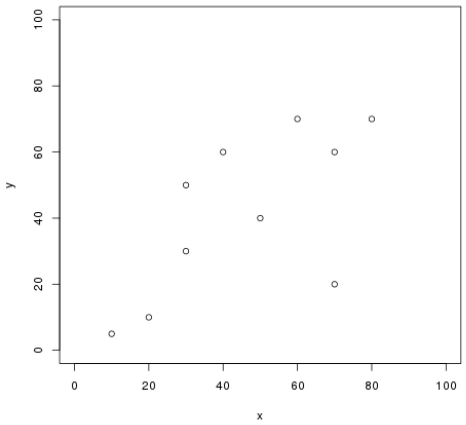

Abbildung 2.10.: Punktewolke

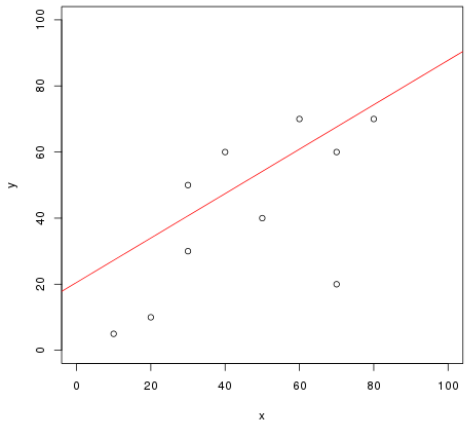

Abbildung 2.11.: Regressionsgerade

Ziel ist es somit die Summe der quadratischen Abweichungen zu minimieren. Für $f(x_i)$ wird eine lineare Funktion gesucht, wodurch sich folgende Gleichung ergibt:

$$\sum_{i=1}^{n}(b_0 + b_1 x_i - y_i)^2 \tag{2.14}$$

Es wird somit b_0 und b_1 gesucht, so dass die Gleichung minimal ist. Die beiden Variablen können mit der Methode der kleinsten Quadrate berechnet werden:

$$b_0 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=n}^{n}(x_i - \bar{x})^2} \tag{2.15}$$

$$b_1 = \bar{y} - a\bar{x} \tag{2.16}$$

2.4. Übersicht Cluster-Algorithmen

Bei der Clusteranalyse werden Mengen von Objekten in Gruppen bzw. Klassen aufgeteilt, wobei die Objekte innerhalb der Gruppe ähnlich und die Gruppen möglichst unterschiedlich sind (vgl. Han u. a. (2011), Kapitel 10.1). Zur Bestimmung der Ähnlichkeit oder Unähnlichkeit von Objekten können bei der Clusteranalyse mehrere Merkmale herangezogen werden, wodurch der Vergleich der Objekte sich über mehrere Dimensionen erstrecken kann. Eine Herausforderung hierbei ist es, möglichst genau zu bestimmen, wie ähnlich oder unähnlich zwei Objekte sind.
Im Gegensatz zu klassifizierenden Verfahren, wie »Neuronale Netze« oder »Support Vector Machines« ist bei der Clusteranalyse kein Vorwissen über die zu analysierenden Daten nötig. Die Algorithmen müssen bei der Clusteranalyse nicht mit vorklassifizierten Daten trainiert werden.
Es wurden bereits eine Vielzahl an Algorithmen für die Clusteranalyse entwickelt, die anhand der Modellierung der Cluster, dem algorithmischen Vorgehen und der Toleranz gegenüber Streuungen in den Daten unterschieden werden können. Herausforderung bei der Clusteranalyse ist es eine geeignete Ähnlichkeitsfunktion für die Daten zu finden und die Anzahl der Cluster, die in den Daten gefunden werden sollen, festzulegen oder aus den Daten zu bestimmen.
Nachfolgend werden vier Untergruppen der Cluster-Algorithmen beschrieben und verschiedene Ansätze, um die Anzahl der Cluster aus den Daten zu bestimmen, vorgestellt.

2.4.1. Partitionierende Cluster-Verfahren

Die partitionierenden Cluster-Verfahren haben gemeinsam, dass die Anzahl der Klassen vorgegeben ist und bereits eine initiale Einteilung der Objekte in Gruppen durchgeführt wurde. In mehreren Iterationsschritten werden die Objekte den Klassen zugewiesen, denen sie am ähnlichsten sind, und neue Gruppen von Objekten bestimmt. Der Vorgang wird solange wiederholt, bis sich die Zugehörigkeiten nicht mehr verändern oder bis die Verbesserung so gering ist, dass sie ein Abbruchkriterium erfüllt.
Ein Beispiel für ein partitionierendes Cluster-Verfahren ist der K-Means-Algorithmus (vgl. MacQueen (1967)).
Ziel von K-Means ist es, für eine Menge d von Vectoren mit $x = (x_1, \ldots, x_n)$ in $K < n$

Partitionen $S = S_1, S_2, \ldots, S_n$ aufzuteilen, so dass die Summe der Quadrate innerhalb des Clusters minimiert wird:

$$\underset{s}{argmin} \sum_{i=1}^{K} \sum_{x_j \in S_i} ||x_j - c_i||^2 \tag{2.17}$$

In der Gleichung 2.17 ist c_i das Zentrum der Partion S_i, wobei diese anhand des Mittelwerts berechnet wird.

Distanzfunktionen

Bei der euklidischen Distanz wird der Abstand über den Satz des Pythagoras mit folgender Berechnungsvorschrift berechnet, wobei die Koordinaten: $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_n)$ gegeben sind:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.18}$$

Eine weitere Distanzfunktion ist Mahalanobis-Distanz, die bei multivariaten Verteilungen eingesetzt werden kann:

$$d(x, y) = \sqrt{(x - y)^T \cdot C^{-1} \cdot (x - y)} \tag{2.19}$$

Die Variable C^{-1} stellt die inverse Kovarianzmatrix dar.
Bei der »Manhattan«-Distanz wird der Abstand von zwei Punkten über die Summe der absoluten Differenzen berechnet:

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{2.20}$$

2.4.2. Modellbasierte Cluster-Verfahren

Bei der modellbasierten Clusteranalyse (vgl. Hastie u. a. (2008) Kapitel 6.8, Fraley und Raftery (1998), Fraley und Raftery (2000)) wird für die Darstellung der Cluster ein Modell verwendet, dessen Parameter aus den Daten berechnet werden. Das Modell liefert für einen bestimmten Datensatz eine Wahrscheinlichkeit, mit dem das Objekt dem Cluster zugehörig ist. Ein Objekt kann somit mit verschiedenen Wahrscheinlichkeiten zu mehreren Clustern gehören, weshalb dieses Verfahren auch als »weiche« Clusteranalyse bezeichnet wird.

Als Modelle für die Daten dienen Verteilungsfunktionen, die zum Beispiel eine Gauß-Verteilung oder Studentsche t-Verteilung sein kann. Damit mehrere Cluster mit diesem Ansatz modelliert werden können, werden die einzelnen Verteilungen zu einer zusammengesetzten Verteilung, einer Mischverteilung, zusammengefasst.

Mischverteilung

Eine Mischverteilung ist eine gewichtete Summe mehrerer Haufigkeitsverteilungen:

$$f(x) = \sum_{m=1}^{M} p_m \cdot f_m(x) \tag{2.21}$$

Hierbei ist f_m die Dichtefunktion der Verteilungsfunktion. Die Variable p_m ist die Wahrscheinlichkeit, dass x zu der Verteilungsfunktion f_m gehört. Die Summe der Wahrscheinlichkeiten ergibt eins:

$$\sum_{m=1}^{M} p_m = 1 \tag{2.22}$$

In der Abbildung 2.12 ist eine Mischverteilung dargestellt, die aus zwei Häufigkeitsverteilungen besteht.

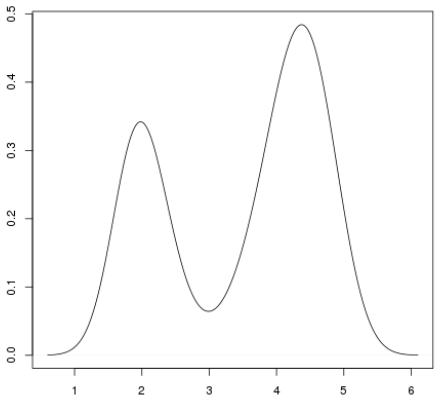

Abbildung 2.12.: Mischverteilung

Eine Zusammensetzung mehrere Gauß-Verteilungen wird als »Gaußsche Mischverteilung« oder im englischen als »Gaussian Mixture Model« bezeichnet.

Gaussian Mixture Models

Gaußsche Mischverteilungen werden durch eine Summe von gewichteten Normalverteilungen (vgl. Abschnitt 2.3.3) dargestellt:

$$f(x) = \sum_{m=1}^{M} p_m \cdot \phi(x \mid \mu, \sigma^2) \tag{2.23}$$

Damit diese für das Clustering von mehrdimensionalen Daten verwendet werden kann wird eine mehrdimensionale Normalverteilung benötigt, deren Dichtefunktion für eine n-dimensionale Zufallsvariable $\vec{x} = (x_1, x_2, \ldots, x_n)$, definiert ist:

$$\Phi(\vec{x} \mid \mu, \Sigma, n) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \cdot \exp\left\{ -\frac{1}{2} (\vec{x} - \mu)^T \Sigma^{-1} (\vec{x} - \mu) \right\} \tag{2.24}$$

Hierbei ist der Erwartungswert $\mu = (\mu_1, \mu_2, \ldots, \mu_n)$ n-Dimensional und die Kovarianz wird als Matrix angegeben:

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2n} \\ \sigma_{m1} & \sigma_{m2} & \ldots & \sigma_{mn} \end{pmatrix}$$

Clustering

Jedes Cluster wird durch eine mehrdimensionale Dichtefunktion der Gauß-Verteilung modelliert. Die Anzahl der Dichtefunktionen entspricht der Anzahl der Cluster, die dann in einer Mischverteilung zusammengefasst werden:

$$f(\vec{x}|\lambda) = \sum_{n=1}^{N} p_i \cdot \Phi_i(\vec{x}|\lambda) \tag{2.25}$$

$$\text{mit } \lambda = \{\mu, \Sigma, n\} \tag{2.26}$$

Damit die Cluster-Zugehörigkeiten bestimmt werden können, müssen die Parameter der N Dichtefunktionen berechnet werden. Diese werden dann mit dem »Expectation Maximization«-Algorithmus (vgl. Dempster u. a. (1977)) optimiert. Dieser besteht aus zwei Schritten, dem »Expectation«-Schritt und dem »Maximization«-Schritt, die solange wiederholt werden, bis eine Abbruchbedingung erfüllt ist.
Im »Expectation«-Schritt wird über alle Punkte iteriert, wobei für jeden Punkt und jede der N

Gauss-Verteilungen die Zugehörigkeits-Wahrscheinlichkeiten berechnet werden.
Beim »Maximization«-Schritt wird für jede Gauss-Verteilung die Varianz und der Erwartungswert berechnet. Bei der Berechnung werden die Punkte anhand der im »Expectation«-Schritt bestimmten Wahrscheinlichkeit gewichtet. Die Wahrscheinlichkeit für eine bestimmte Gaussverteilung wird bestimmt, indem die Summe der Zugehörigkeits-Warscheinlichkeiten aller Punkte durch die Anzahl der Punkte geteilt wird.
Ob ein Objekt $\vec{x}$ zu der Komponente m gehört, kann folgendermaßen berechnet werden:

$$P_{im} = \frac{p_m \cdot \Phi(\vec{x}; \mu, \Sigma_m, n)}{\sum_{k=1}^{M} p_k \Phi(\vec{x}; \mu, \Sigma_m, n)} \tag{2.27}$$

2.4.3. Hierarchische Cluster-Verfahren

Bei hierarchischen Cluster-Verfahren werden die Daten in eine abgestufte Folge von Clustern eingeteilt (vgl. Hastie u. a. (2008), Kapitel 14.3.12). Hierbei wird zwischen zwei verschiedenen Typen von Verfahren unterschieden, dem divisiven und dem agglomerativen.

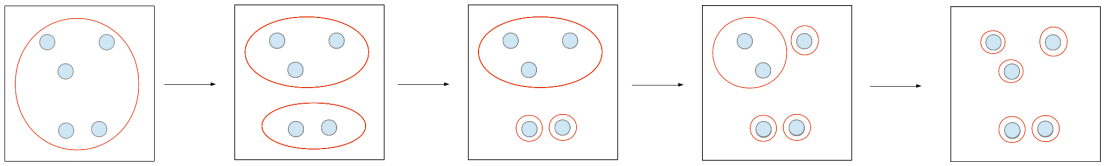

Abbildung 2.13.: Divisives Clustering-Verfahren

Bei den divisiven Verfahren werden alle Objekte zu einem Cluster zusammengefasst, das dann schrittweise in neue Cluster aufgeteilt wird (siehe Abbildung 2.13).

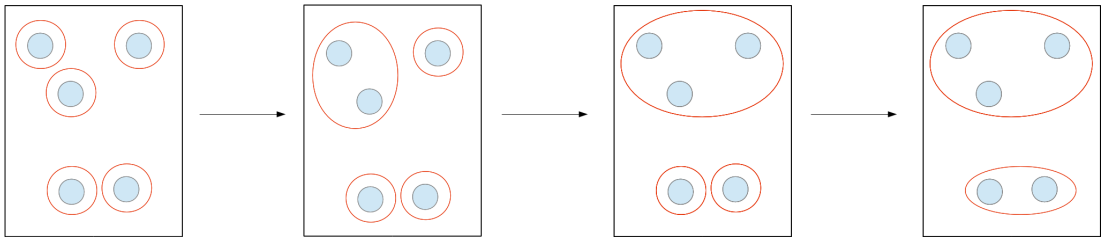

Abbildung 2.14.: Aglomeratives Cluster-Verfahren

Die agglomerativen Verfahren arbeiten andersherum, hierbei stellt zu Anfang jedes Objekt ein eigenständiges Cluster dar, die dann zu neuen Clustern zusammengefasst werden (siehe Abbildung 2.14).

Vorteil der hierarchischen Cluster-Verfahren ist, dass die Anzahl der Cluster nicht vorher festgelegt werden muss. Allerdings ist es erforderlich, eine Abbruchbedingung zu definieren, anhand der bestimmt wird, ob das Clustering vollständig durchgeführt wurde.
Das Ergebnis des Clusterings wird in einem sogenannten Dendogramm dargestellt (siehe Abbildung 2.15). In diesem werden die einzelnen Datenpunkte auf der X-Achse aufgetragen, in dem Beispiel sind es insgesamt fünf. Die Y-Achse gibt an, ab welchem Abstand zwei Punkte miteinander verschmolzen wurden. In dem Beispiel in der Abbildung 2.15 wurden die Punkte 1 und 2 bei einem Abstand von 10 zu einem Cluster zusammengefasst.

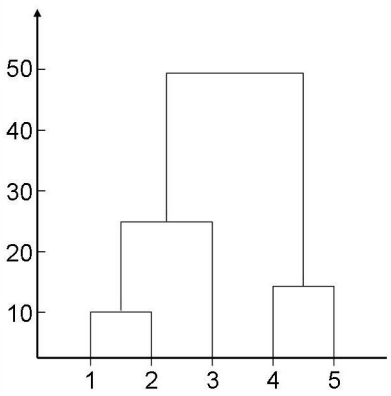

Abbildung 2.15.: Dendogramm

Berechnung der Abstände zwischen zwei Clustern

Der Abstand D zwischen zwei Clustern A und B kann anhand verschiedener Vorschriften bestimmt werden.

- **Single linkage**
  Hierbei wird der kürzeste Abstand zwischen zwei Punkten der beiden Cluster berechnet:

$$D(A,B) = \min_{a \in A\,,\, b \in B} d(a,b) \tag{2.28}$$

- **Complete link**
  Gibt den größten Abstand zwischen den beiden Clustern wieder:

$$D(A,B) = \max_{a \in A\,,\, b \in B} d(a,b) \tag{2.29}$$

- **Average link**
  Berechnet den durchschnittlichen Abstand der Punkte aus zwei Clustern:

$$D(A,B) = \frac{\sum_{a \in A, b \in B} d(a,b)}{|A||B|} \tag{2.30}$$

- **Centroids**
  Hierbei wird der Abstand zwischen den beiden Clustern anhand des Abstandes der Cluster-Mittelpunkte berechnet:

$$D(A,B) = d(\bar{a}, \bar{b}) \tag{2.31}$$

  Wobei $\bar{a}$ der Mittelpunkt vom Cluster A und $\bar{b}$ der Mittelpunkt von Cluster B ist.

Das Ergebnis des hierarchischen Clusterings hängt stark von der Berechnung des Abstandes der Cluster ab. Von Vorteil ist, dass keine Parameter für die Initialisierung benötigt werden. Nachteilig ist jedoch, dass bei sehr vielen Datenpunkten das Verfahren rechenaufwändig wird. Eine weitere Schwierigkeit stellt die Wahl der geeigneten Abbruchbedingung dar, die vorgibt, wann die Cluster nicht mehr geteilt oder zusammengefasst werden.

2.4.4. Dichtebasierte Cluster-Verfahren

Bei dichtebasierten Cluster-Algorithmen werden Cluster anhand der Datenpunktdichte identifiziert. Grundlage hierfür ist die Annahme, dass innerhalb von Clustern die Dichte von Objekten höher ist als außerhalb. Ein Beispiel für einen dichtebasierten Cluste-Algorithmus ist der 1996 entwickelte DBSCAN (vgl. Ester u. a. (1996)).

Abbildung 2.16.: Beispiele für Clustering mit DBSCAN aus Ester u. a. (1996) Abbildung 6

DBSCAN

Mit DBSCAN können Cluster in Daten gefunden werden, die eine beliebige Form haben (siehe Abbildung 2.16). Der Algorithmus benötigt nur zwei Eingabeparameter, um das Clustering durchzuführen, und kann mit beliebigen Distanzfunktionen oder Ähnlichkeitsmaßen arbeiten. Bei der Bestimmung der Cluster müssen nicht alle Objekte im Hauptspeicher gehalten werden, wodurch DBSCAN auf sehr große Datenmengen angewendet werden kann.
Bei DBSCAN werden die Datenpunkte in Cluster und Rauschen aufgeteilt, wobei Cluster eine hohe Dichte an Punkten haben. Die Dichte eines Datenpunktes wird anhand der Anzahl der Punkte in seiner ϵ-Umgebung berechnet. Diese besteht aus allen Punkten, deren Abstand maximal ϵ zum Punkt P ist. Die Abbildung 2.17 veranschaulicht dies.

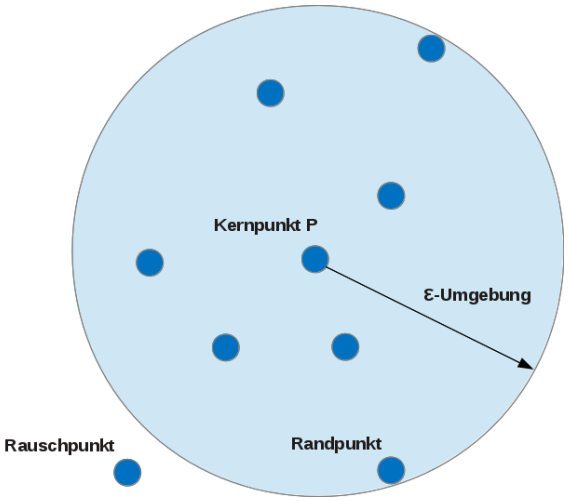

Abbildung 2.17.: ϵ-Umgebung eines Punktes P

Die Anzahl N_ϵ der Nachbarn von einem Punkt p wird nach folgender Vorschrift berechnet, wobei D die Menge aller Punkte ist:

$$N_\epsilon(p) = \{q \in D \mid dist(p,q) \leq \epsilon\} \tag{2.32}$$

Vor Anwendung von DBSCAN muss ein möglichst optimaler Wert für ϵ festgelegt werden. Wird er zu klein gewählt, besteht die ϵ-Umgebung aus nur einem Punkt, sofern er zu groß gewählt wird, liegen alle anderen Punkte in der ϵ-Umgebung von P.
Ein weiterer Parameter des DBSCAN-Algorithmus ist $MinPts$, der festlegt, wie viele Punkte in einer ϵ-Umgebung eines Punktes liegen müssen, damit er zu einem Cluster gehört.
Bei DBSCAN wird jeder Punkt anhand einer von drei Kategorien charakterisiert:

- **Kernpunkt**
  Die Anzahl der Punkte in der ϵ-Umgebung beträgt mindestens MinPts:

$$N_\epsilon(p) \geq MinPts \tag{2.33}$$

- **Randpunkt**
  Ein Randpunkt liegt in der ϵ-Umgebung eines Kernpunktes, es sind jedoch nicht MinPts Nachbarn in seiner eigenen ϵ-Umgebung enthalten (siehe Abbildung 2.17).

- **Rauschpunkt**
  Rauschpunkte sind alle Punkte, die keine Kernpunkte oder Randpunkte sind.

Der Ablauf des DBSCAN-Algorithmus erfolgt in folgenden fünf Schritten:

1. Jeder Datenpunkt wird als Kern-, Rand- oder Rauschpunkt festgelegt.
2. Die Rauschpunkte werden aus der Menge von Punkten entfernt und nicht weiter betrachtet.
3. Kernpunkte, die in der ϵ-Umgebung vorhanden sind, werden als zusammengehörig markiert.
4. Eine Menge von als zusammengehörig markierten Kernpunkten wird zu einem Cluster vereint.
5. Jeder Randpunkt wird dem Cluster eines benachbartem Kernpunkts zugewiesen.

Vorteil von DBSCAN im Gegensatz zu partitionierenden Cluster-Algorithmen ist, dass die Anzahl der Cluster im Vorfeld nicht bekannt sein muss. Die Wahl der Parameter ϵ und $MinPts$ ist für das Ergebnis des Clusterings entscheidend.
Eine Weiterentwicklung von DBSCAN ist der Cluster-Algorithmus OPTICS (vgl. Ankerst u. a. (1999)), der verschiedene Verbesserungen enthält. Bei OPTICS können die einzelnen Cluster eine unterschiedliche Dichte haben und der Parameter ϵ muss nicht mehr zwingend vorgegeben werden.

2.5. Clustering von Daten aus der Durchflusszytometrie

Die manuelle Auswertung von Daten aus der Durchflusszytometrie ist aufwändig und fehleranfällig. Aus diesem Grund wurden eine Reihe von Verfahren für die Analyse von FACS-Daten entwickelt, die auf Cluster-Algorithmen basieren. Diese sollen es ermöglichen, automatisiert die Häufigkeiten von Zellpopulationen in den Daten zu bestimmen. Die hierbei zu analysierenden Datenmengen sind umfangreich, da die Cluster in mehreren tausend Datenpunkten gefunden werden sollen. Aus diesem Grund müssen die Algorithmen möglichst performant arbeiten und auf einen geringen Speicherverbrauch optimiert sein. Eine gute Übersicht über die zur Verfügung stehenden Verfahren gibt die Publikation von Bashashati und Brinkman (2009). Der erste Ansatz, der entwickelt wurde, um Cluster-Algorithmen auf Daten aus der Durchflusszytometrie anzuwenden, basierte auf K-Means (vgl. Murphy (1985)).
Mittlerweile wurde eine Vielzahl von Ansätzen entwickelt, wovon im Folgenden die Algorithmen FlowPeaks und FlowMeans vorgestellt werden.
Weitere Cluster-Algorithmen für FACS-Daten sind: curvHDR (Naumann u. a. (2010)), flowMerge (Finak u. a. (2009)), flowClust (Lo u. a. (2009)) und FLAME (Pyne u. a. (2010)).

2.5.1. FlowPeaks

Bei FlowPeaks (vgl. Ge und Sealfon (2012)) handelt es sich um ein Cluster-Verfahren, das Bestandteil des Bioconductor-Projektes ist. Um die Cluster in den Daten zu finden, durchläuft FlowPeaks mehrere Schritte, in denen unter anderem K-Means und »Gaussian Mixture Models« angewendet werden.

Anzahl der Cluster berechnen

Die Wahl der Anzahl der Cluster K ist für das Ergebnis der Clusteranalyse entscheidend. Bei FlowPeaks wird K mithilfe folgender Formal von Freedman und Diaconis berechnet (vgl. Freedman und Diaconis (1981)):

$$K_j = \frac{x^j_{(n)} - x^j_{(1)}}{2 \cdot IQR(x^j) \cdot n^{-\frac{1}{3}}} \text{ für } j = 1, \ldots, d \tag{2.34}$$

Die Variablen $x^j_{(n)}$ und $x^j_{(1)}$ enthalten das Minimum und das Maximum der Dimension j der Daten $x^j = (x^j_1, x^j_1, \ldots, x^j_n)$. In der Variablen n ist dieAnzahl der Elemente in der Dimension j festgelegt. Die Funktion $IQR()$ gibt den Wert des Inter-Quartilsabstands zurück, dieser ist definiert als die Differenz zwischen dem unteren und oberen Quartil: $Q_{0,75} - Q_{0,25}$.

Für jede Dimension d der Daten wird ein K_j mit $j = 1, \ldots, d$ berechnet. Als endgültiges K wird dann der Median aus der Menge von K_j verwendet, dessen Wert aufgerundet wird (Die Symbole $\lceil \cdot \rceil$ stehen für das Aufrunden):

$$K = \lceil median(K_1, \ldots, K_d) \rceil \tag{2.35}$$

Bestimmen der initialen Cluster-Zugehörigkeiten mit K-Means++

Nachdem die Anzahl der Cluster bestimmt wurde, werden die initialen Cluster-Zugehörigkeiten mit K-Means++ (vgl. Arthur und Vassilvitskii (2007)) für die Daten berechnet. Durch die Initialisierung mit K-Means++ wird die Laufzeit und das Ergebnis von K-Means wesentlich verbessert.

LLoyds K-means unter Verwendung eines k-d-Baums

Der K-Means-Algorithmus von LLoyd (vgl. Lloyd (1982)) wird im nächsten Schritt mit den zuvor festgelegten Cluster-Zugehörigkeiten initialisiert und ausgeführt. Als Distanzfunktion wird bei K-Means die euklidische-Abstand eingesetzt. Zur Verbesserung der Laufzeit des K-Means-Algorithmus werden die Daten in einem k-d-Baum (vgl. Bentley (1975)) organisiert.

Hartigan und Wongs K-Means-Algorithmus

Im dritten Schritt wird der von Hartigan und Wong entwickelte K-Means-Algorithmus (vgl. Hartigan und Wong (1979)) auf die Daten angewendet, um das Clustering zu verbessern. Dieser wird aus Performanz-Gründen nicht direkt nach der Anwendung von K-Means++ ausgeführt.

»Gaussian Mixture Models«

Nachdem verschiedene Implementierungen des K-Means-Algorithmus eingesetzt wurden, um die Cluster zu berechnen, wird mithilfe von »Gaussian Mixture Models« eine Mischverteilung berechnet. Anhand der Dichtefunktion der Mischverteilung können dann die Cluster-Zugehörigkeiten festgelegt werden (vgl. Abschnitt 2.4.2).
Die Dichtefunktion der Mischverteilung wird anhand folgender Formel berechnet:

$$f(x) = \sum_{k=1}^{K} p_k \cdot \phi(x, \mu_k, \Sigma_k) \tag{2.36}$$

In der Gleichung 2.36 ist p_k die Gewichtung des Clusters k und ϕ die Gauss-Verteilungsfunktion. Die Gauss-Verteilung erhält als Parameter den Erwartungswert μ und die Kovarianz-Matrix Σ. Diese wird mit folgender Funktion geglättet:

$$\tilde{\Sigma}_k = \lambda_k \cdot h \cdot \Sigma_k + (1 - \lambda_k) \cdot h_0 \cdot \Sigma_0 \tag{2.37}$$

Die Variablen h und h_0 können als Parameter an FlowPeaks übergeben werden und sind folgendermaßen initialisiert: $h = 1.5$ und $h_0 = 1$. Die Variable λ wird mit folgender Formel berechnet:

$$\lambda_k = \frac{n \cdot p_k}{k + n \cdot p_k} \tag{2.38}$$

Finden der lokalen Maxima und Zusammenfassen der Cluster

Anhand der Dichtefunktion der Gaußschen-Mischverteilung werden die lokalen Maxima (Peaks) für jedes Cluster bestimmt. Dies erfolgt indem die Dichtefunktion negiert und ein Gradientenverfahren angewendet wird. Sofern die Peaks von zwei Cluster einen geringen Abstand zueinander haben werden sie zu einem Cluster zusammengefasst.

Implementierung

FlowPeaks kann über das Bioconductor-Projekt bezogen werden. Es ist in C++ implementiert und stellt eine Schnittstelle für die Programmiersprache R bereit, über die flowPeaks in ein R-Script eingebunden werden kann. Mit FlowPeaks können beliebig dimensionale Daten geclustert werden.

2.5.2. FlowMeans

FlowMeans (vgl. Aghaeepour u. a. (2011)) ist ein Cluster-Verfahren für Daten aus der Durchflusszytometrie, das auf K-Means basiert. Problem bei K-Means ist, dass die Anzahl der Cluster vorgegeben werden muss, was FlowMeans umgeht, indem die Anzahl der Cluster berechnet wird.

Anzahl der Cluster berechnen

Der K-Means-Algorithmus benötigt als Parameter die Anzahl der Cluster K. Dieser wird bei FlowMeans berechnet, indem das Kerndichteschätzer-Verfahren (»kernel density estimation«) eingesetzt wird. Hierbei handelt es sich um ein Verfahren, mit dem die Verteilungsfunktion

einer Zufallsvariablen geschätzt wird.
Der Kerndichteschätzer ist eine mathematische Funktion, die folgendermaßen definiert ist:

$$\hat{f}(x) = \frac{\sum_{i=1}^{n} K(\frac{x-X_i}{h})}{n \cdot h} \tag{2.39}$$

Der Parameter $h > 0$ der Formel ist die Bandbreite, die entscheidend für die Qualität der Approximation ist. Die Funktion K wird als Kern bezeichnet und ist eine Dichtefunktion, die zum Beispiel die Gaußdichte sein kann:

$$K(x) = \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{-\frac{x^2}{\sqrt{2}}} \tag{2.40}$$

Zum Berechnen der Anzahl der Cluster wird auf den Kerndichteschätzer ein Gradientenverfahren angewendet. Die Anzahl der Cluster wird dann anhand der Anzahl der Peaks der Funktion festgelegt, indem gezählt wird, wie häufig der Gradient von 0 zu 1 wechselt.

Bestimmen der Cluster

Bei FlowMeans modellieren mehrere Cluster eine einzige Zellpopulation, wodurch es möglich ist, Cluster zu finden, die nicht rund sind. Als Distanzfunktion wird eine Mahalanobis-Distanz (siehe Abschnitt 2.4.1 - Distanzfunktionen) eingesetzt.

2.6. Softwaretechnologien

Dieses Kapitel wird genutzt, um einen Überblick über verschiedene Programmiersprachen und Software-Technologien zu geben. Die beiden Scriptsprachen Python und R werden vorgestellt. Im weiteren Verlauf dieses Abschnittes wird auf das Bioconductor-Projekt eingegangen, das verschiedene Bibliotheken aus dem Bereich der Bioinformatik enthält, die mit der Programmiersprache R verwendet werden können. Zum Abschluss dieses Abschnittes wird sich mit dem Webframework Django befasst, das auf Python basiert.

2.6.1. Programmiersprache R

Die Programmiersprache R[3] (vgl. Wollschläger (2012)) ist eine Skriptsprache, die für statistische Berechnungen und zum Erzeugen von Grafiken optimiert ist. Der Interpreter für R ist Open-Source und wird als Teil des GNU-Projekts weiterentwickelt. Mit R ist es sehr einfach, mathematische Operationen auf Vektoren oder Matrizen anzuwenden, da diese in der Programmiersprache als Datenstrukturen verfügbar sind. Die Standardbibliothek von R enthält eine Vielzahl von Methoden, mit denen statistische Berechnungen durchgeführt werden können. Eine objektorientierte Programmierung mit R ist möglich, wobei der Sprachumfang hierbei auf Klassen und Objekte beschränkt ist.
Der Funktionsumfang von R kann über eine Vielzahl von Zusatzpaketen erweitert werden, diese werden über das »Comprehensive R Archive Network« (CRAN[4]) bezogen. Dort sind eine Reihe von Cluster-Algorithmen enthalten[5], wie zum Beispiel DBSCAN oder der EM-Algorithmus.
Mit »Rstudio«[6] steht eine komfortable Entwicklungsumgebung für das Programmieren in R zur Verfügung. Diese ermöglicht es, R-Scripte in Projekten zu organisieren und bietet eine Reihe von Funktionen, um R-Scripte zu debuggen.

2.6.2. Bioconductor

Bioconductor[7] (vgl. Gentleman u. a. (2004)) ist ein Open-Source-Projekt, das verschiedene Bibliotheken für die Bioinformatik bereitstellt, die mit der Programmiersprache R verwendet werden können. Gestartet wurde Bioconductor am »Fred Hutchinson Cancer Research Center«

[3] http://www.r-project.org
[4] http://cran.r-project.org/ (abgerufen am 18.07.2013)
[5] Übersicht von Cluster-Algorithmen in R:
http://cran.r-project.org/web/views/Cluster.html (Abgerufen am 16.07.2013)
[6] http://www.rstudio.org
[7] http://www.bioconductor.org (Abgerufen am 01.07.2013)

im Jahr 2001. Ziel des Projekts ist es, gut dokumentierte und wiederverwendbare Module bereitzustellen, die hilfreich für die Forschung im Bereich der Bioinformatik sind.
Für die Verarbeitung von Daten aus der Durchflusszytometrie sind eine Reihe von Bibliotheken im Bioconductor-Projekt enthalten (vgl. Klinke und Brundage (2009)), unter anderem FlowCore, FlowQ und FlowViz, die im Folgenden vorgestellt werden.

FlowCore

Für die Verarbeitung von FCS-Dateien stellt FlowCore (vgl. Hahne u. a. (2009)) verschiedene Methoden bereit, die es ermöglichen, FCS-Dateien zu öffnen und auf die darin abgespeicherten Daten zuzugreifen. Hierbei können die Metadaten, wie zum Beispiel die Namen der gemessenen Kanäle oder das Datum der Messung, ausgelesen werden. Mithilfe von FlowCore kann mit R direkt auf die gemessenen Signale des Lasers zugegriffen werden, was eine Analyse der Daten möglich macht.

FlowQ

Mit FlowQ[8] können verschiedene Qualitätskriterien von FCS-Dateien geprüft werden. Hierbei ist es möglich, sowohl die bereits in FlowQ enthaltenen Prüfungen durchzuführen, als auch FlowQ zu erweitern, um eigene Prüfungen zu implementieren. Die Ergebnisse der Qualitätsprüfung können als HTML abgespeichert werden, um sie dann mit einem Browser zu betrachten. Alternativ ist es möglich, FlowQ in einem R-Script aufzurufen und die Ergebnisse weiterzuverarbeiten.
Es kann geprüft werden, ob die Anzahl der gemessenen Signale signifikant vom Mittelwert der Anzahl der gemessenen Signale mehrerer FCS-Dateien abweicht. Eine weitere Prüfung, die durchgeführt wird ist, ob gemessene Signale in der Signalstärke sehr stark abweichen. Dies kann zum Beispiel aufgrund von Messfehlern oder einem falsch eingestellten Durchflusszytometer der Fall sein. Ein weiteres Qualitätsmerkmal ist, ob es Zeit-Anomalien in der Datei gibt. Bei einer Messung wird pro Zeiteinheit eine Zelle im Blut gemessen. Sollte es hierbei Lücken in den Daten geben, ist die Messung unvollständig und somit nicht für weitere Analysen nützlich.

FlowViz

FlowViz enthält eine Vielzahl von Methoden, um FCS-Dateien zu visualisieren. Die einzelnen Kanäle einer FCS-Datei können mit FlowViz als Dotplot dargestellt werden. In der Abbildung

[8] http://www.bioconductor.org/packages/release/bioc/html/flowQ.html (Abgerufen am 01.07.2013)

2.18 ist ein Plot dargestellt, bei dem alle Kanäle der FCS-Datei gegeneinander aufgetragen wurden.

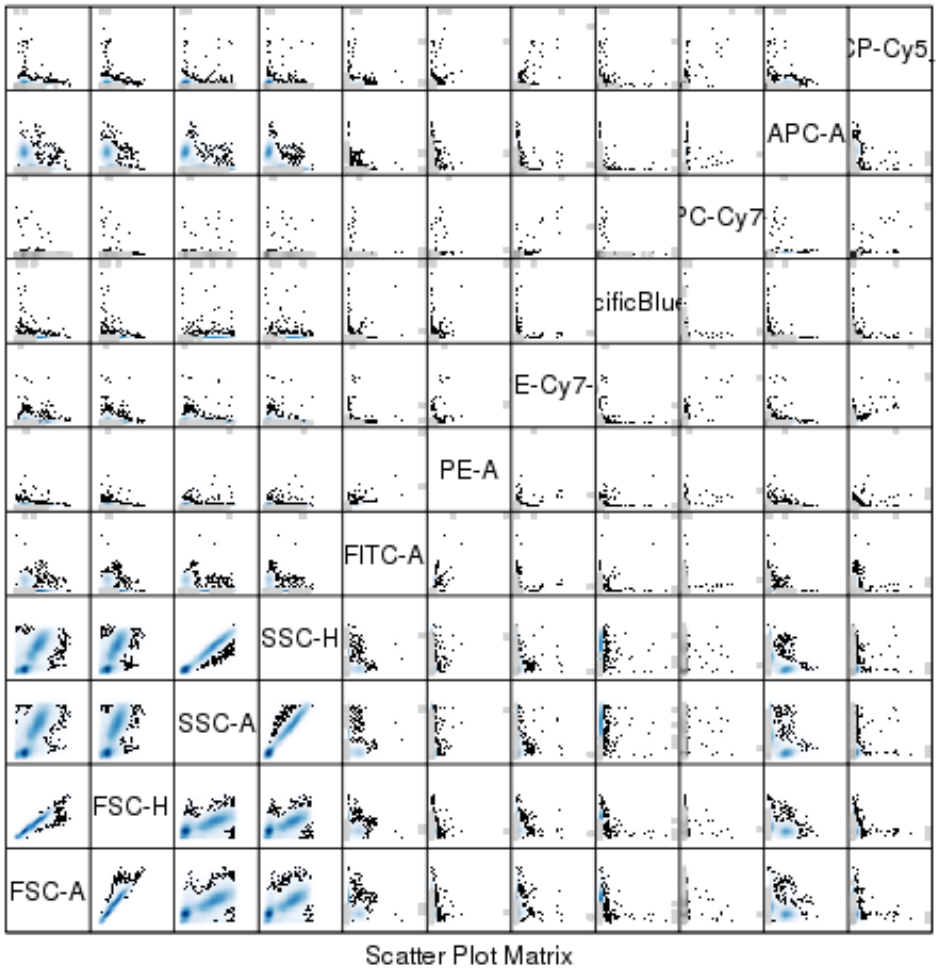

Abbildung 2.18.: Plot aller Kanäle einer FCS-Datei

2.6.3. Programmiersprache Python

Python[9] (vgl. Lutz (2011) und Alchin (2010b)) ist eine objektierentierte Skriptsprache, deren Interpreter für alle gängigen Betriebsysteme, wie Windows, Linux und MacOS, portiert wurde. Schöpfer von Python ist Guido van Rossum, der die Programmiersprache Anfang der 90er

[9] http://www.python.org/

Jahre am »Centrum voor Wiskunde en Informatica« in Amsterdam entwickelte. Mittlerweile wird Python von einer großen Entwicklergemeinde permanent weiterentwickelt, wobei Spracherweiterungen und neue Funktionen stetig in die Sprache einfließen.
Python basiert auf dynamischer Typisierung. Ein besonderes Merkmal von Python ist es, dass die Syntax auf Einrückungen basiert. Anders als die gägingen Programmiersprachen wie C, C++ oder Java, wo zum Beispiel Schleifenrümpfe in Klammern gefasst werden, werden diese bei Python durch Einrückungen hervorgehoben.
Programmkomponenten können in Python in Module ausgelagert werden, die vom Hauptprogramm importiert werden. Namespaces werden hierbei unterstützt, wodurch Kollisionen im Namensraum vermieden werden.

2.6.4. Webframework Django

Django ist ein Webframework (vgl. Holovaty (2010), Bennett (2008), Alchin (2010a)), das in der Programmiersprache Python entwickelt wurde. Es basiert auf dem »Model view controller«-Prinzip (MVC, vgl. Reenskaug (1979)), das von den Django-Entwicklern etwas anders interpretiert wurde (vgl. Django-FAQ[10]). Bei Django wird im Gegensatz zum MVC-Grundgedanken die View nicht zur Datenrepräsentation verwendet, sondern zum Erzeugen von HTML. Der Controller wird durch das Framework implementiert, wobei die verschiedenen Views anhand einer URL-Konfguration aufgerufen werden. Das Model wird durch verschiedene Klassen des objektrelationen Mappings dargestellt, mit dem Daten in der relationalen Datenbank gespeichert werden können.
Ursprünglich wurde Django für die News-Website »Lawrence Journal-World«[11] entwickelt und 2005 unter der Open-Source BSD-Lizenz für einem größeren Personenkreis zur Verfügung gestellt. Seit der Open-Source-Stellung wird an Django von einem weltweit wachsenden Entwicklerkreis gearbeitet.
Mit Django werden typische Aufgaben der Webentwicklung wesentlich vereinfacht. Zum Erzeugen von HTML-Code wird auf eine Template-Engine zurückgegriffen. Eine weitere häufige Aufgabe ist das Implementieren und Validieren von Formularen. Diese können mit Django einfach über Klassen definiert werden, wobei die Validierung von Eingaben durch Methoden von Django übernommen wird. Anhand von Klassen des objektrelationalen Mappings können Formulare erzeugt werden, die zum Erzeugen und Verändern von Datensätzen in der Datenbank benutzt werden können.

[10] https://docs.djangoproject.com/en/1.5/faq/ (abgerufen am 21.07.2013)
[11] http://www.lawrance.com

3. Analyse

Zu Beginn dieses Kapitels wird erläutert, wie die Anforderungen an die zu entwickelnde Software in Gesprächen mit den Fachanwendern erhoben wurden. Das zweite Kapitel gibt einen Einblick, wie die Daten zurzeit organisiert und von den Fachanwendern ausgewertet werden. Zum Abschluss dieses Kapitels werden die fachlichen und die nicht-fachlichen Anforderungen im Detail vorgestellt.

3.1. Anforderungsanalyse

Die Anforderungen wurden in verschiedenen Interviews mit den Fachanwendern ermittelt. Initiiert wurden die Gespräche durch den Wunsch, die aufwändigen manuellen Analysen zu automatisieren und es möglich zu machen, den umfangreichen Datenbestand auszuwerten. Um die Anforderungen an die Software und die Problemstellung erfassen zu können, mussten zunächst die Grundlagen der Durchflusszytometrie erarbeitet werden.
Die Gespräche mit den Fachanwendern haben ergeben, dass der bisherige Workflow auf der Software FACSDiva basiert, mit der die einzelnen FCS-Dateien manuell ausgewertet werden. Da FACSDiva keine Möglichkeit bietet, die verschiedenen Arbeitsschritte zu automatisieren, wird die Auswertung einer Vielzahl von Messdaten sehr zeitaufwändig. Als Alternative wird FlowJo eingesetzt, dieses Produkt bietet jedoch ebenfalls keine Möglichkeit, die Analyse zu automatisieren.
Ziel der Auswertung ist es, die Häufigkeit bestimmter Zellpopulationen im Blut zu bestimmen, was als Gating bezeichnet wird (vgl. Abschnitt 2.2.5). Damit dieser Vorgang per Software automatisiert werden kann, müssen die einzelnen Schritte des Gatings klar definiert werden. Weiterhin müssen die Messdaten aus der Durchflusszytometrie ausgelesen werden können. Für den Zugriff auf das FCS-Dateiformat stehen eine Reihe von Bibliotheken für die unterschiedlichen Programmiersprachen, wie zum Beispiel Python oder R, zur Auswahl. Weiterhin bietet sich die Programmiersprache R sehr an, da sie eine Reihe von Funktionen für die Datenanalyse und Statistik bietet (vgl. 2.6.1). Ein weiterer Vorteil von R ist, dass mit dem

Bioconductor-Projekt (vgl. 2.6.2) diverse Bibliotheken für die Verarbeitung von Daten aus der Durchflusszytometrie zur Verfügung stehen.

3.2. Ist-Zustand

In diesem Abschnitt wird der Ist-Zustand beschrieben. Hierbei wird auf den aktuellen Workflow, die Organisation und die Analyse der Daten eingegangen.

Workflow

In der Abbildung 3.1 sind die einzelnen Arbeitsschritte aufgezeigt, die für die Messung mit dem Durchflusszytometer und die Analyse der Daten notwendig sind.
Der erste Schritt hierbei ist, die Messung mit dem Durchflusszytometer durchzuführen. Das Durchflusszytometer wird mit der Software FACSDiva gesteuert. Sobald die Messung abgeschlossen wurde, wird eine FCS-Datei erzeugt, die dann auf dem zentralen Fileserver abgespeichert wird. Pro Messung einer Blutprobe wird somit eine FCS-Datei erzeugt.
Zu einem späteren Zeitpunkt wird die FCS-Datei wieder mit FACSDiva geöffnet und durch einen Fachanwender analysiert. Die Ergebnisse der Analyse werden mit FACSDiva gespeichert oder für weitere Auswertungen in ein anderes Dateiformat, wie zum Beispiel MS-Excel, exportiert. Die Abbildung 3.2 zeigt einen Screenshot von FACSDiva.

Organisation der Daten

Die erzeugten FCS-Dateien werden in einfachen Verzeichnisstrukturen im Dateisystem des Fileservers abgelegt. In der Abbildung 3.3 ist der Verzeichnisbaum in Auszügen dargestellt. Pro Tag, an dem die Messungen durchgeführt werden, wird ein Verzeichnis angelegt, das als Namen den Tag der Messung zugewiesen bekommt. Diese werden wiederum in weiteren Verzeichnissen gruppiert, deren Namen einen Zeitraum beschreiben, zum Beispiel »ab 03. Juni 2009«. Der Bestand von FACS-Daten am ZMNH hat mittlerweile einen Umfang von ungefähr einem Terabyte erreicht.

Metadaten

Bei der Analyse der Daten sollen für eine Vielzahl von FCS-Dateien die Häufigkeiten verschiedener Zelltypen im Blut bestimmt werden. Hierfur wurde vor der Messung mit dem Durchflusszytometer verschiedene Antigene (Marker), die an bestimmten Zelltypen haften

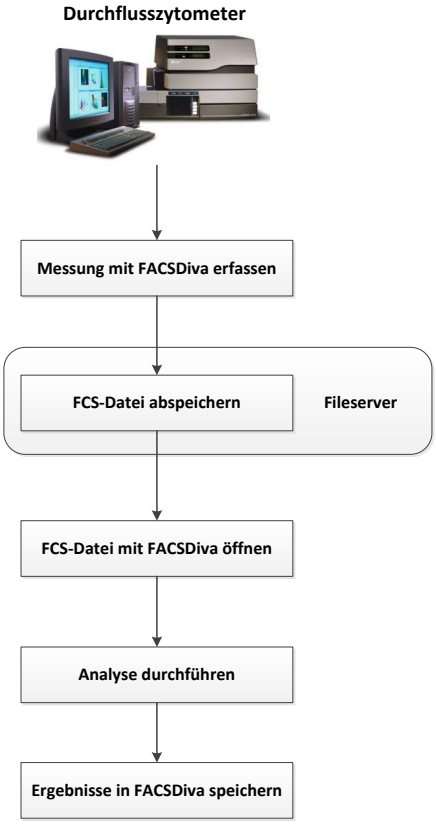

Abbildung 3.1.: Aktueller Workflow

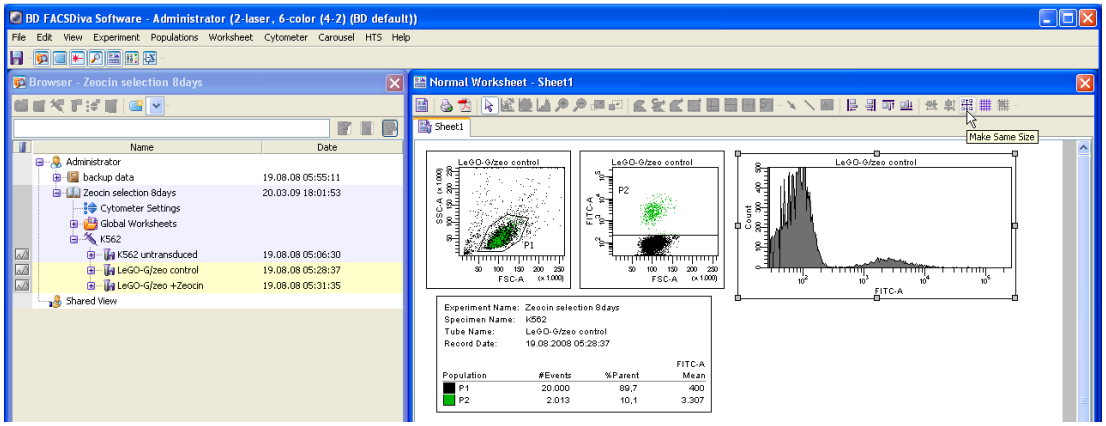

Abbildung 3.2.: Analyse mit FACSDiva

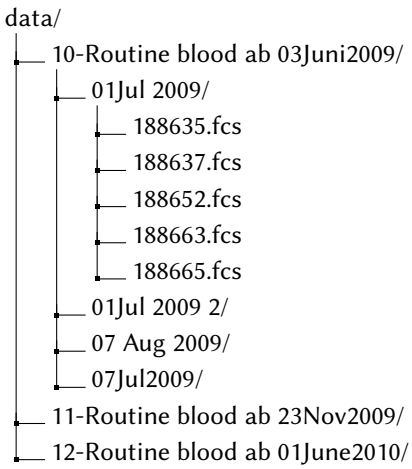

Abbildung 3.3.: Verzeichnisstruktur der FCS-Dateien

und mit einem fluoreszierenden Stoff eingefärbt sind, der Blutprobe hinzugefügt (siehe Abschnitt 2.2.2). Die Namen der Marker werden in den Metadaten der FCS-Datei hinterlegt. Für die Analyse müssen FCS-Dateien, bei denen bestimmte Marker gemessen wurden, gefunden werden. Dies ist mit einem hohen Aufwand verbunden, da hierfür jede FCS-Datei mit der Software FACSDiva geöffnet werden und geprüft werden muss, ob der Marker in der FCS-Datei enthalten ist. Ein Problem hierbei ist, dass die Marker teilweise für den gleichen Zelltyp unterschiedlich benannt sind, da sie von verschiedenen Herstellern produziert wurden.

Analyse der Daten

Zur Bestimmung der Zellpopulationen werden einzelne FCS-Dateien in die Software FACSDiva geladen und manuell die Häufigkeiten in mehreren Auswertungsschritten bestimmt (siehe Abschnitt 2.2.5). Dieses Vorgehen ist für eine große Menge an Daten sehr aufwändig und fehleranfällig, da die Häufigkeiten der Zellpopulationen visuell bestimmt werden. Hierbei werden durch den Benutzer Cluster von Signalen identifiziert, anhand derer dann die Häufigkeiten der Zellpopulationen bestimmt werden. Das Vorgehen ist für jeden Benutzer individuell, wodurch sich abhängig von der jeweiligen Person unterschiedliche Ergebnisse ergeben können.
Da die Analysen sehr aufwändig sind, wurde bisher nur ein geringer Teil der Daten ausgewertet. Viele Experimente mit wissenschaftlich wertvollen Messreihen wurde daher noch nicht analysiert.

Aktuelle Probleme

Die Organisation der Daten ist nicht optimal, da es sehr aufwändig ist, FCS-Dateien anhand bestimmter Kriterien zu finden. Diese können zum Beispiel der Name eines Markers, der gemessen wurde, oder ein bestimmtes Datum, an dem die Messung durchgeführt wurde sein. Die Analysen sind aufwändig, fehleranfällig und für die Person, die sie durchführt, individuell. Bisher wurde wenig automatisiert, da die eingesetzten Programme keine Möglichkeiten hierfür bieten. Der Datenbestand ist so umfangreich, dass ein Großteil der Daten zum jetzigen Zeitpunkt nicht analysiert werden konnte und somit ungenutzt ist.

3.3. Fachliche Anforderungen

Das am ZMNH manuell durchgeführte Gating-Verfahren soll mit einer Software automatisiert werden. Ziel hierbei ist es, möglichst exakte Werte für die Häufigkeiten der Zellpopulationen festzustellen (siehe Abschnitt 2.2.5). Das aktuell eingesetzte Analyse-Verfahren besteht aus insgesamt fünf Schritten, in denen mit FACSDiva Zellpopulationen selektiert und anhand der Selektion gefiltert werden. In jedem Schritt der Analyse wird die Anzahl der Signale reduziert, wobei die im Abschnitt 2.1 beschriebene Hierarchie der unterschiedlichen Zelltypen verfolgt wird. Hierdurch wird es möglich, am Ende der Analyse möglichst genaue Werte für die Häufigkeiten der Zellpopulationen zu bestimmen. In der Abbildung 3.4 ist das aktuelle Vorgehen schematisch dargestellt. In den grau dargestellten Rechtecken sind die Namen der Zellen aufgetragen, die in dem jeweiligen Schritt selektiert werden. Im letzten Schritt werden anhand der verschiedenen Zelltypen die CD39+ und CD39- Zellen bestimmt, diese sind in der Abbildung als blaue Rechtecke hervorgehoben.
Im Folgenden werden die einzelnen Schritte des Gatings im Detail erläutert.

Lymphozyten

Im ersten Schritt werden die Lymphozyten in den Daten identifiziert. Diese erzeugen auf den Kanälen, auf denen das Vorwärts- und das Seitwärts-Streulicht gemessen wird, schwache Signale, da sie klein sind und eine geringe Granularität haben. Beim manuellen Gating werden die Lymphozyten in einem Dotplot, der die Signale der Kanäle des Vorwärts- und Seitwärtss-treulichts enthält, identifiziert (siehe Abbildung 3.3).
Die Selektion der Lymphozyten ist rechteckig, wobei die Dimensionen des Rechtecks vom Benutzer festgelegt wird. Hierbei wird Höhe und Breite relativ frei gewählt. Vorschrift hierbei ist es, möglichst schwache Signale auf beiden Kanälen zu selektieren.

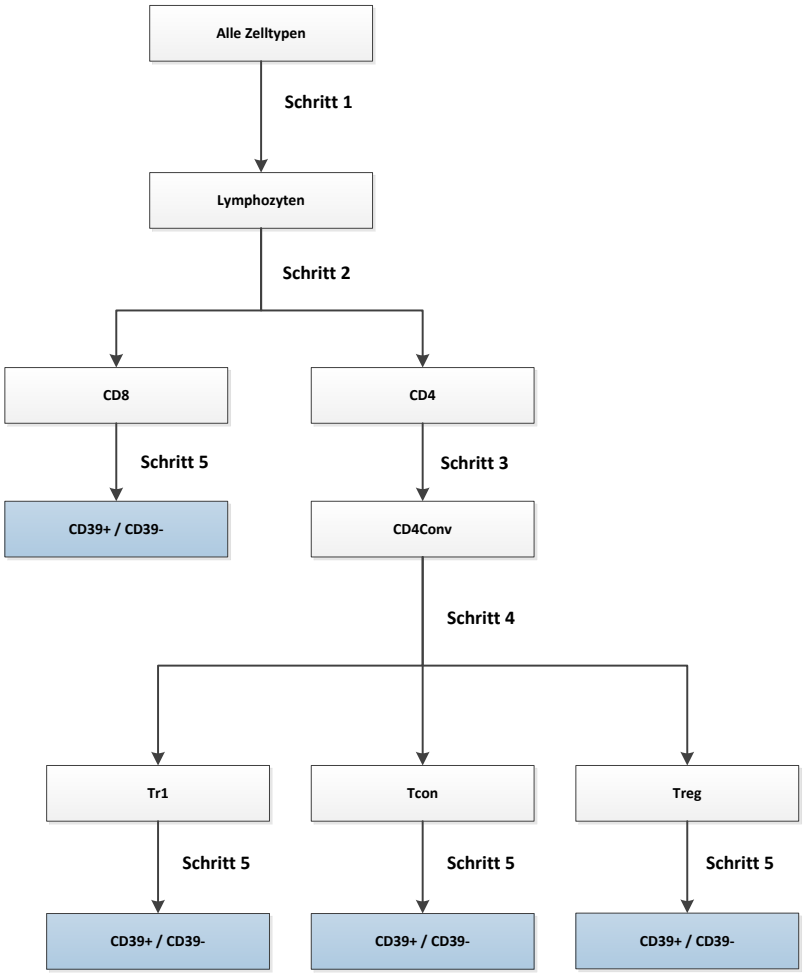

Abbildung 3.4.: Gating-Strategie

Da keine klare Definition zur Höhe und Breite des rechteckigen Gates gegeben ist, lässt sich dieses Vorgehen nur schwer automatisieren. Aus diesem Grund werden die Dimensionen des Rechtecks über ein zweistufiges Verfahren festgelegt. Im ersten Schritt werden in einem Plot die Kanäle FSC und APC aufgetragen (siehe Abbildung 3.6). In diesem Plot können zwei Cluster identifiziert werden. Das in der Abbildung grün dargestellte Cluster enthält die Signale von Lymphozyten, da sie hohe Werte auf dem Kanal APC haben. Im zweiten Schritt werden die als Lymphozyten identifizierten Signale in einem Plot der Kanäle SSC und FSC hervorgehoben (siehe Abbildung 3.6 Schritt 2), diese sind in dem Plot grün dargestellt und geben die

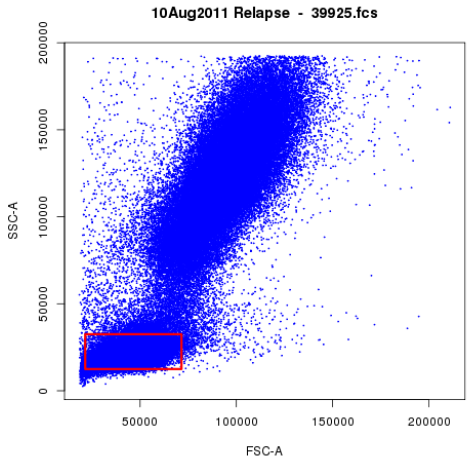

Abbildung 3.5.: Gaten der Lymphozyten anhand der Kanäle des Vorwärts- und Seitwärtsstreulichts

Dimensionen des rechteckigen Gates vor. Hierdurch sind die Dimensionen des Gates für die Lymphozyten festgelegt.

CD8- und CD4-Zellen

Um die CD8- und CD4-Zellen zu bestimmen, werden die Signale der Kanäle APC und FITC in einem XY-Dotplot aufgetragen (Abbildung 3.7). APC ist ein Fluoreszenzmarker, der an CD4-Zellen haftet. FITC haftet an CD8-Zellen, wodurch deren Häufigkeit im Blut bestimmt werden kann.
In der Abbildung 3.7 sind insgesamt drei Cluster von Punkten zu erkennen. Die CD8-Zellen werden in dem Plot anhand der Signale identifiziert, die einen niedrigen Wert auf dem APC-Kanal und einen hohen Wert auf dem FITC-Kanal haben. Dieses ist in der Abbildung mit der Zahl 2 beschriftet.
Die Häufigkeit der CD8-Zellen kann anhand des Clusters, das mit der Zahl 3 in der Abbildung beschriftet ist, bestimmt werden. In diesem sind Signale enthalten, die eine hohe Signalstärke auf dem Kanal APC und eine geringe Signalstärke auf dem Kanal FITC haben.
Das Cluster, das mit der Zahl 1 beschriftet ist, enthält Signale, die geringe Werte auf beiden Kanälen haben und somit für diese Analyse nicht weiter relevant sind.

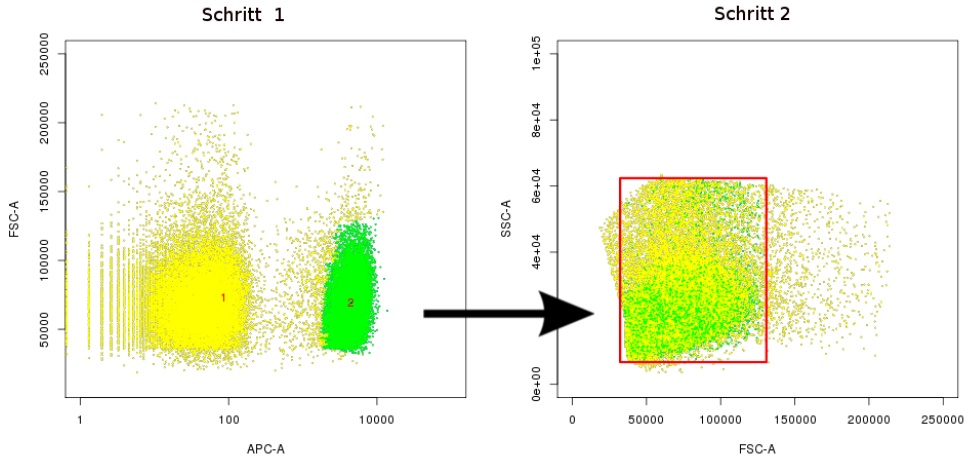

Abbildung 3.6.: Lymphozyten selektieren

Tcon, Tr1 und Treg

Die im vorherigen Schritt identifizierten Signale der CD4-Zellen werden in einem Dotplot, der die Kanäle PE-A und PerCP-Cy5_5 enthält, dargestellt (siehe Abbildung 3.8). Es können insgesamt drei Cluster identifiziert werden, die mit den Zahlen 1, 2 und 3 in der Abbildung beschriftet sind. Das Cluster mit der Nummer 1 hat eine sehr hohe Dichte an Punkten und enthält Signale der Tcon-Zellen. Die beiden anderen Cluster sind schwieriger zu identifizieren, wobei das Cluster mit der Nummer 2 die Signale von Tr1-Zellen und das dritte Cluster die Signale von Treg-Zellen enthält.

Histogramme erzeugen

Im letzten Schritt werden die Signale der CD8-, Tcon-, Tr1- und Treg-Zellen als Histogramme dargestellt. Diese Darstellung ermöglicht es, die Signale in CD39+ und CD39- zu unterscheiden. In der Abbildung 3.9 ist anhand von Signalen der Tcon-Zellen beispielhaft die Überführung in ein Histogramm dargestellt.
Die Häufigkeit der CD39+ und CD39- Zellen wird bestimmt, indem das Histogramm in zwei Hälften geteilt wird, was Abbildung 3.10 dargestellt ist.

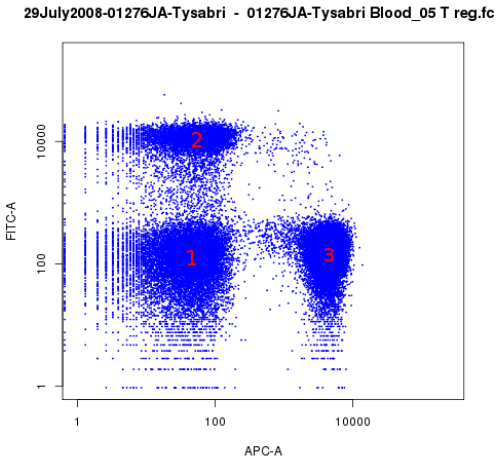

Abbildung 3.7.: CD8- und CD4-Zellen bestimmen

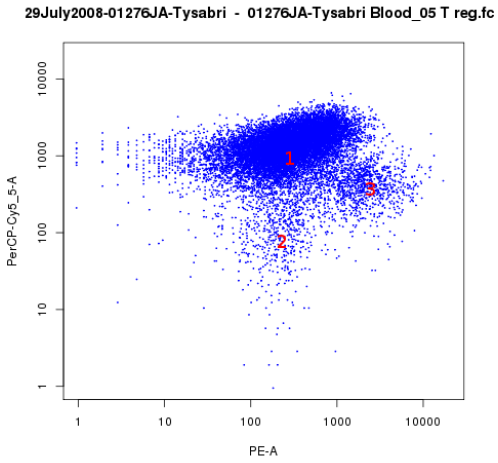

Abbildung 3.8.: Verarbeitung der CD4-Signale

Ergebnis der Analyse

Aus dieser Analyse ergibt sich die Verteilung der unterschiedlichen Zelltypen im Blut und es können Rückschlüsse über die Wirksamkeit von Medikamenten oder den Krankheitsverlauf der Patienten gewonnen werden.

3.4. Nichtfunktionale Anforderungen

Aus den Anforderungen an die Software ergeben sich diverse nichtfunktionalen Anforderungen, die in diesem Abschnitt erläutert werden.

Analyseprogramm als Kommandozeilenanwendung

Damit das Analyseprogramm auf einem Server ohne grafische Oberfläche ausgeführt werden kann, soll es die Möglichkeit bieten, als Kommandozeilen-Programm ausgeführt zu werden.

Unter Linux lauffähig

Ein Großteil der Server am ZMNH wird mit Linux betrieben. Damit diese für die Analyse verwendet werden können, soll das Analyseprogramm unter Linux lauffähig sein.

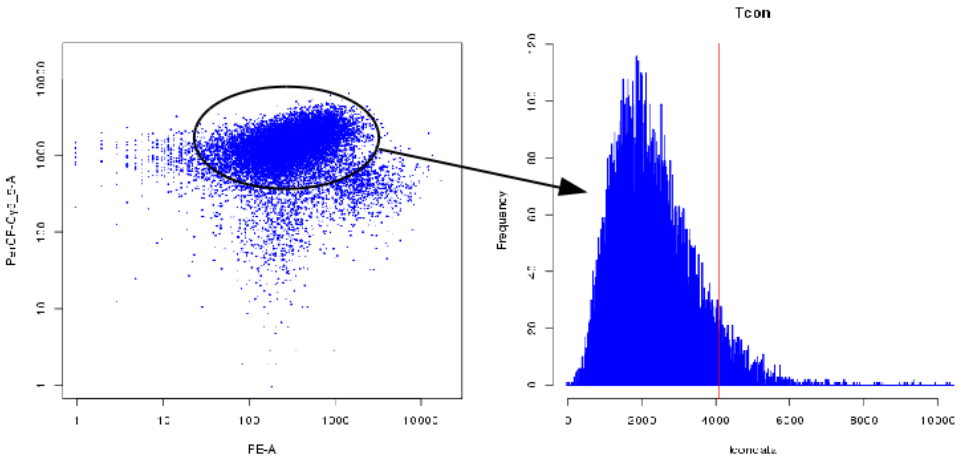

Abbildung 3.9.: Überführung der Signale der Tcon-Zellen in ein Histogramm

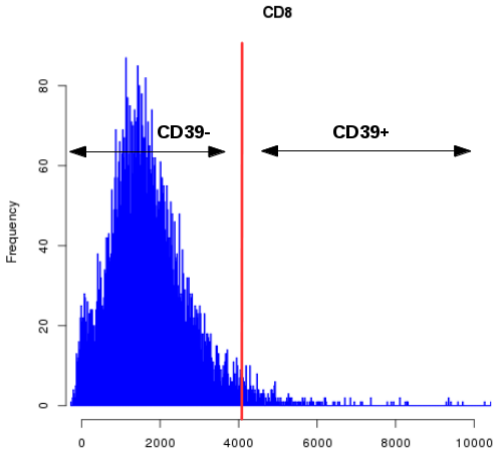

Abbildung 3.10.: Bestimmen der CD39+ und CD39- Zellen im Histogramm

Laufzeit der Analyse einer FCS-Datei

Mit der Analysesoftware soll ein großer Bestand an Messdaten analysiert werden. Eine Messreihe besteht aus mehreren hundert FCS-Dateien, deren Analyse in einem Zeitraumen von zwei Tagen möglich sein soll. Als maximale Laufzeit für die Analyse einer FCS-Datei wird fünf Minuten festgelegt, wordurch es möglich wird, am Tag 288 FCS-Dateien zu analysieren.

Parallele Ausführung

Damit die Rechenleistung von Mehrprozessorsystemen genutzt werden kann, soll die Software die Analyse mehrerer FCS-Dateien parallel durchführen.

Speichern der Ergebnisse in einer Datenbank

Zur Auswertung der Ergebnisse der Analysen zu einem späteren Zeitpunkt sollen diese in einer Datenbank abgespeichert werden.

Erzeugen von Grafiken

Um die einzelnen Verarbeitungsschrite nachzuvollziehen, sollen Grafiken erzeugt werden, die die einzelnen Schritte der Analyse veranschaulichen.

Plattformunabhängiger Zugriff auf die Analyseergebnisse

Der Benutzer soll die Möglichkeit haben, über eine Oberfläche die Analyseergebnisse und die erzeugten Grafiken zu betrachten, um neue Erkenntnisse aus den Ergebnissen ableiten zu können. Hierfür bietet sich die Entwicklung einer Webanwendung an, da auf diese mit einem beliebigen Webbrowser zugegriffen werden kann, der für alle gängigen Betriebsysteme zur Verfügung steht.

4. Entwurf

In diesem Kapitel wird die Architektur der zu entwickelnen Software vorgestellt. Die Analyseergebnisse werden in einer Datenbank abgespeichert, deren Layout im zweiten Abschnitt dieses Kapitels beschrieben wird. Der letzte Abschnitt dieses Kapitels wird genutzt, um die Auswahl der Cluster-Algorithmen für die Datenanalyse vorzustellen und zu begründen.

4.1. Software-Architektur

Bei der Entwicklung der Architektur für die Analyseplattform stehen verschiedene Konzepte zur Auswahl, deren Eignung abgewogen werden muss.
Eine Möglichkeit besteht darin, die Software als Desktop-Anwendung zu implementieren. Hierbei wird die Analyse der Daten direkt auf der Plattform durchgeführt, auf der die Anwendung gestartet wurde. Dies hat den Nachteil, dass der Benutzer der Anwendung über einen leistungsstarken Rechner verfügen muss und bei einer langen Laufzeit der Analyse paralleles Weiterarbeiten nur eingeschränkt möglich ist.
Eine Alternative dazu ist es, die Analyse und die Präsentationsschicht zu entkoppeln und diese über eine Client-Server-Architektur miteinander kommunizieren zu lassen. Auf dem Desktop des Benutzers ist hierbei ein Programm installiert, das mit dem Server kommuniziert, die Analyse steuert und die Ergebnisse präsentiert. Die schwergewichtigen Rechenoperationen werden hierbei auf dem leistungsstarken Server durchgeführt.
Bei der Architektur für die Analyseplattform wurde sich für einen Ansatz entschieden, der ähnlich der Client-Server-Architektur ist. Die Analyseplattform wurde in zwei Kernbestandteile aufgeteilt, das Analyseprogramm und die Visualisierungsplattform. Das Analyseprogramm kommuniziert jedoch nicht direkt mit der Visualisierungsplattform, sondern legt die Ergebnisse in einer Datenbank ab und erzeugt Grafiken, die im Dateisystem abgespeichert werden. Die Visualisierunsplattform greift auf die Datenbank und die im Dateisystem gespeicherten Grafiken zu, um diese dem Benutzer zu präsentieren. Eine Anforderung an die Software ist der plattformunabhängige Zugriff auf die Analyseergebnisse (vgl. Abschnitt 3.4), dies wird durch Entwicklung der Visualisierungsplattform als Webanwendung ermöglicht. Eine weitere

Anforderung an die Software ist, dass das Analyseprogramm als Kommandozeilenprogramm auf einem Server ausgeführt werden kann. Dieses ist mit der entwickelten Architektur ebenfalls möglich.
Die konzeptionelle Architektur der Plattform kann anhand der Abbildung 4.1 nachvollzogen werden. Das Analyseprogramm greift auf eine FCS-Datei zu und führt in mehreren Schritten eine Analyse durch. Hierbei werden bei jedem Schritt eine Reihe von Grafiken erzeugt und das Ergebnis der Analyse in einer MySQL-Datenbank abgespeichert. Mit der Visualisierungsplattform kann die Analyse zu einem späteren Zeitpunkt ausgewertet werden. Sie ermöglicht es, die Grafiken und die in der Datenbank gespeicherten Ergebnisse zu betrachten.

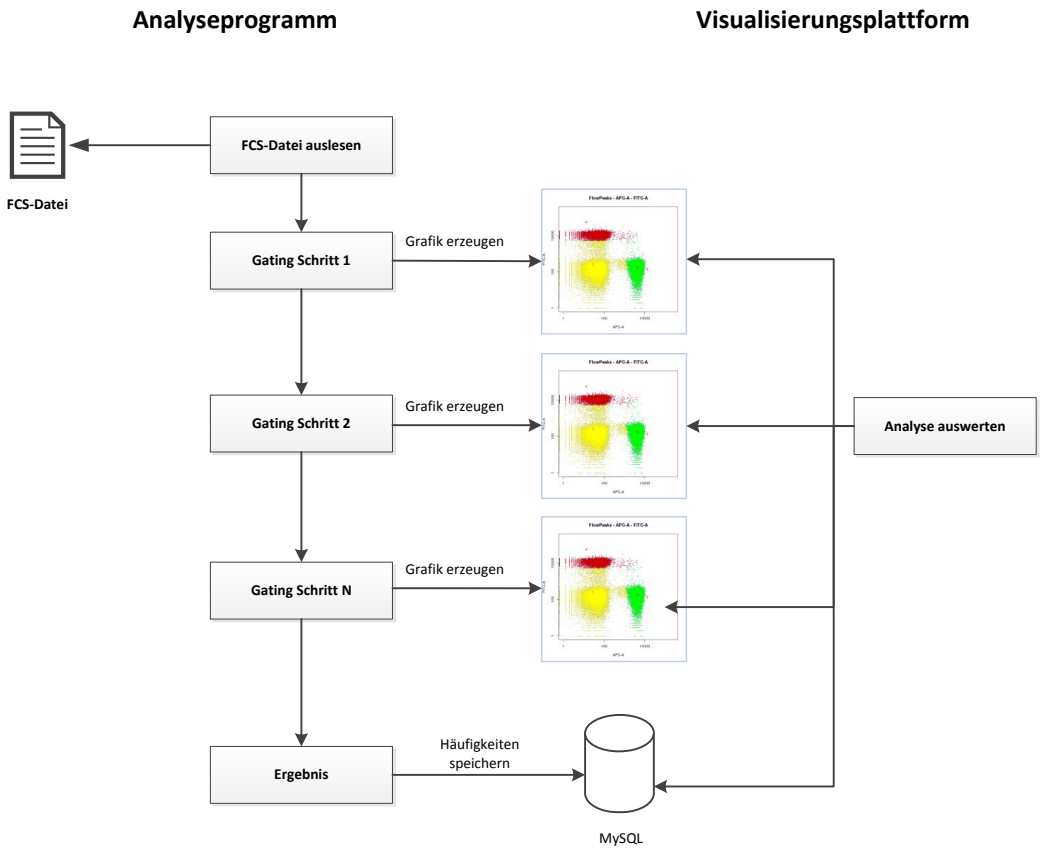

Abbildung 4.1.: Konzeptionelle Architektur der Analyseplattform

Architektur des Analyseprogramms

Für die Architektur des Analyseprogramms wurde ein Framework-Ansatz gewählt. Das Ziel ist, dass der Programmierer lediglich die Programmlogik für die Analyse der FCS-Datei vorgeben muss und wiederkehrende Aufgaben durch das Framework übernommen werden. Der Vorteil hierbei ist, dass die Analyse einfach modifiziert werden kann, sofern andere Zellpopulationen im Blut gefunden werden sollen.
In der Abbildung 4.2 ist die Architektur des Framework-Ansatz als Komponentendiagramm dargestellt. Gesteuert wird die Analyse durch einen Controller. Das Analyseprogramm wird durch den Benutzer des Frameworks vorgegeben und durch den Controller gestartet. Zum Speichern der Ergebnisse wird die Datenbank-Komponente verwendet, die einen Zugriff auf die MySQL-Datenbank ermöglicht. Wichtiger Bestandteil der Analyse ist es, Grafiken zu erzeugen. Dies erfolgt mit der Visualisierungs-Komponente. Die Analyse der FCS-Datei erfolgt mit einer Reihe von Cluster-Algorithmen, die durch die Clustering-Komponente bereitgestellt wird.

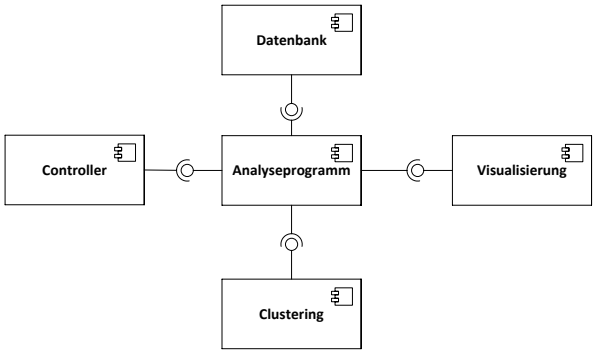

Abbildung 4.2.: Architektur des Frameworks

Architektur der Visualisierungsplattform

Für die Entwicklung der Visualisierungsplattform wurde sich für das Webframework Django (vgl. Abschnitt 2.6.4) entschieden. Django bietet eine Reihe von Funktionen für wiederkehrende Aufgaben, wie zum Beispiel der Zugriff auf die Datenbank und das Erzeugen von HTML und minimiert somit den Entwicklungsaufwand.
Bei der Entwicklung mit Django wird die Struktur der Webananwendung durch das Framework vorgegeben. Die Abbildung 4.3 zeigt die Architektur der Visualisierungsplattform, die auf den

Vorgaben von Django basiert. In der Views-Komponente sind eine Reihe von Methoden enthalten, die mithilfe von Templates HTML erzeugen. Für den Zugriff auf die Datenbank wird ein objektrelationales Mapping verwendet, dessen Klassen als »Models« bezeichnet werden und in der Abbildung in der Models-Komponente zusammengefasst sind. Anhand der URL-Konfiguration wird festgelegt, welche View bei einer bestimmten Anfrage aufgerufen wird. Die Anfrage wird vom Django-Framework an die URL-Konfiguration weitergegeben, die dann die entsprechende View aufruft.

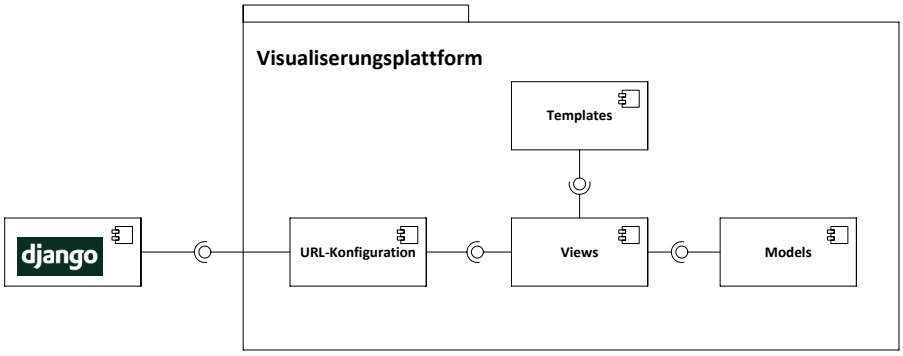

Abbildung 4.3.: Architektur der Visualisierungsplattform

4.2. Datenbankmodell

Die Abbildung 4.4 zeigt das »Entity Realationship Diagram« (ERM) der Datenbank, das aus insgesamt drei Tabellen besteht.
In der Tabelle »Analysation« werden die Analysen angelegt, wobei bei jedem Datensatz der Name der Analyse, der Status der Analyse und eine ID gespeichert werden. Die Tabelle »File« speichert das Ergebnis der Analyse einer FCS-Datei. Sie enthält neben dem Namen der Datei, die analysiert wurde, die Ergebnisse der Analyse. Hierbei werden die Häufigkeiten der verschiedenen Zellpopulationen gespeichert.
Bei der Analyse einer Datei werden eine Reihe von Plots im Dateisystem erzeugt. Die Dateinamen werden in dem Attribut »filename« der Entität »Plot« abgespeichert. Diese ist einem »File« zugeordnet, wodurch es möglich ist, alle Plots, die bei der Analyse einer FCS-Datei erzeugt wurden, auszulesen.

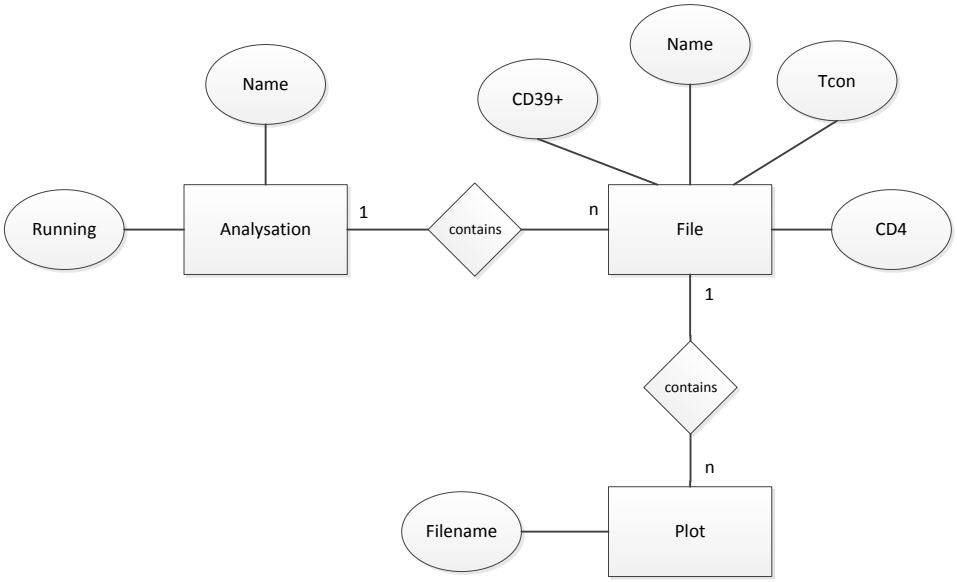

Abbildung 4.4.: ERM-Diagramm der Datenbank

4.3. Konzeption des automatisierten Analyseverfahrens

Im Abschnitt 3.3 wurden die fachlichen Anforderungen vorgetellt. Sie beschreiben den Workflow der Datenanalyse, wie er manuell mit der Software FACSDiva durchgeführt wird. Ziel der Analyse ist es, die Häufigkeiten verschiedener Zellpopulationen in der Messung zu bestimmen. Hierzu werden in mehreren Schritten die Daten gefiltert und Häufungen (Cluster) von Datenpunkten identifiziert.
Um dies zu automatisieren, muss ein Verfahren entwickelt werden, das die einzelnen Analyseschritte mit verschieden Algorithmen durchführt. Um den manuellen Workflow zu automatisieren, müssen für die Analyseschritte Algorithmen gefunden und implementiert werden, welche die jeweiligen Zelltypen optimal identifizieren. Dabei ist insbesondere auch die Laufzeit der Algorithmen zu berücksichtigen.
Die Größe und Position der Cluster ist bei den verschiedenen Blutproben sehr unterschiedlich. Der Einsatz von Verfahren, die zuvor trainiert werden müssen, wie zum Beispiel »neuronale Netze« oder »Support Vector Machines«, wäre mit einem hohen Aufwand verbunden, da diese mit einer großen Menge von bereits analysierten Daten trainiert werden müssten. Aus diesem Grund sind Cluster-Algorithmen für diese Aufgabe besser geeignet, da sie kein Vorwissen über die Daten benötigen.

Dieser Abschnitt wird genutzt, um zu beschreiben, welche Verfahren und Algorithmen in den einzelnen Schritten eingesetzt werden, um die Zellpopulationen in den Daten zu identifizieren.

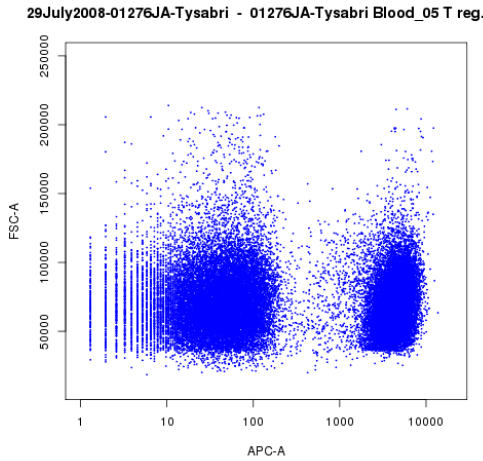

Abbildung 4.5.: Kanäle FSC und APC als Dotplot

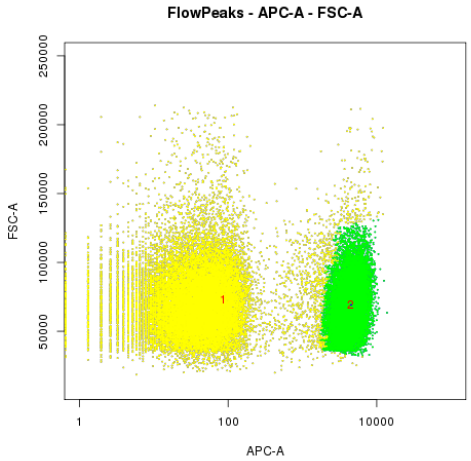

Abbildung 4.6.: Analyse mit FlowPeaks

Schritt 1 - Lymphozyten identifizieren

Im ersten Schritt des Analyse-Verfahrens sollen Zellpopulationen anhand der Kanäle APC und FSC erkannt werden (vgl. Abschnitt 3.3). Die Abbildung 4.5 zeigt den Dotplot der beiden Kanäle, der insgesamt ungefähr 50.000 gemessene Lasersignale enthält. In dem Plot sind zwei Cluster von Datenpunkten zu erkennen, die gefunden werden sollen.
Als Cluster-Algorithmus wurde der Algorithmus FlowPeaks (vgl. Abschnitt 2.5.1) ausgewählt, da dieser beim Test mit 25 FCS-Dateien optimale Ergebnisse bei der Clusteranalyse der Kanäle APC und FSC geliefert hat. Die Abbildung 4.6 zeigt das Ergebnis des Clusterings mit FlowPeaks. Hierbei wurden zwei Cluster gefunden, die in der Abbildung mit den Zahlen 1 und 2 beschriftet sind.

Schritt 2 - CD8- und CD4-Zellpopulationen

Zur Bestimmung der CD8- und CD4-Zellpopulationen werden die Signale der Kanäle APC und FITC analysiert. In der Abbildung 4.7 sind diese als Dotplot dargestellt, in dem insgesamt drei Cluster zu erkennen sind. Das Cluster mit starken Signalen auf dem APC-Kanal enthält die CD4-Zellen, das Cluster mit starken Signalen auf dem FITC-Kanal repräsentiert die gemessenen CD8-Zellen. Das dritte Cluster, das schwache Signale auf beiden Kanälen enthält,

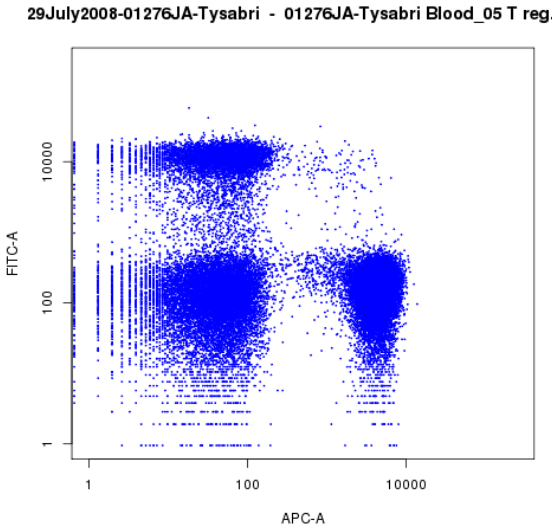

Abbildung 4.7.: Kanäle FSC und APC als Dotplot

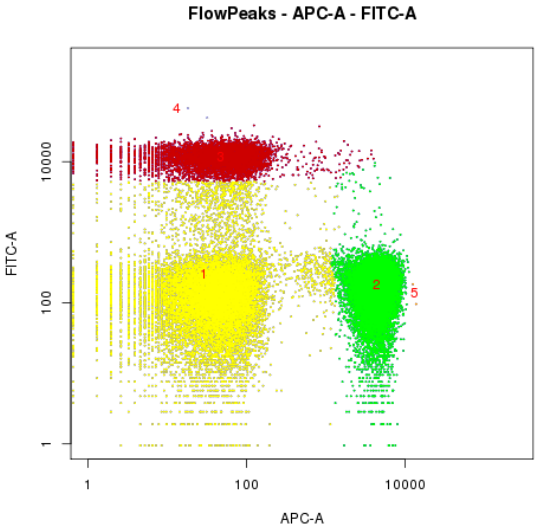

Abbildung 4.8.: Analyse mit FlowPeaks

ist für die Analyse nicht relevant.
Wie in Schritt 1 wurde empirisch geprüft, ob sich der FlowPeaks-Algorithmus zum Finden der Cluster eignet. In der Abbildung 4.8 ist das Ergebnis des Clusterings mit FlowPeaks dargestellt. Hierbei ist zu erkennen, dass mehrere Cluster gefunden wurden und das Ergebnis den Erwartungen entspricht.

Schritt 3 - Tcon-, Tr1- und Treg-Zellpopulationen finden

Im dritten Schritt müssen insgesamt drei weitere Cluster in den Daten gefunden werden. Hierfür werden Signale, die auf den Kanälen PE-A und PerCP-Cy5_5 gemessen wurden in einem Dotplot dargestellt (Abbildung 4.9). In der Abbildung sind die Cluster, die von Bedeutung sind, mit den Zahlen 1, 2 und 3 hervorgehoben.
Zum Finden des größten Clusters, das mit der Zahl 1 beschriftet ist, bietet sich das dichtebasierte Clustering-Verfahren DBSCAN (vgl. Abschnitt 2.4.4) an. Mit DBSCAN können Cluster gefunden werden, deren Punkte einen geringen Abstand zueinander haben und somit eine hohe Datendichte in dem Cluster vorhanden ist. In der Abbildung 4.10 ist das Ergebnis der Anwendung von DBSCAN dargestellt, wobei das Cluster rot hervorgehoben ist.
Zum Erkennen der beiden Cluster, die in der Abbildung 4.9 mit den Zahlen 2 und 3 beschriftet sind, wird zuerst die Menge an Datenpunkten reduziert. Dies erfolgt indem eine Regressionsgerade für die unteren Randpunkte des größten Clusters berechnet wird (siehe Abbildung

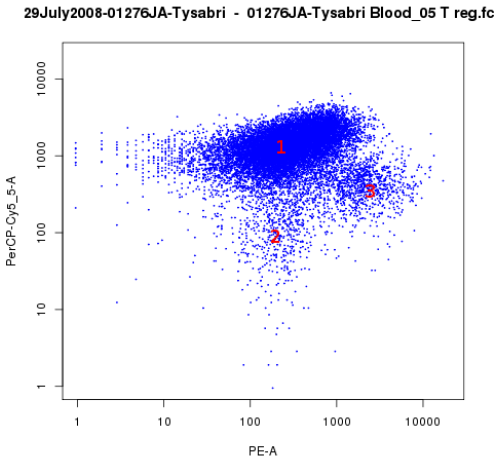

Abbildung 4.9.: Signale der Kanäle PE-A und PerCP-Cy5_5

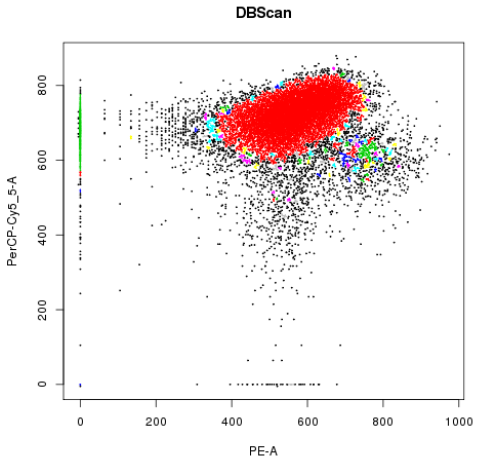

Abbildung 4.10.: Ergebnis nach Anwendung von DBScan

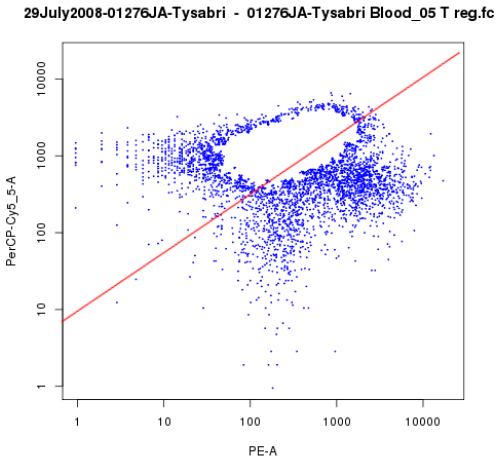

Abbildung 4.11.: Regressionsgerade

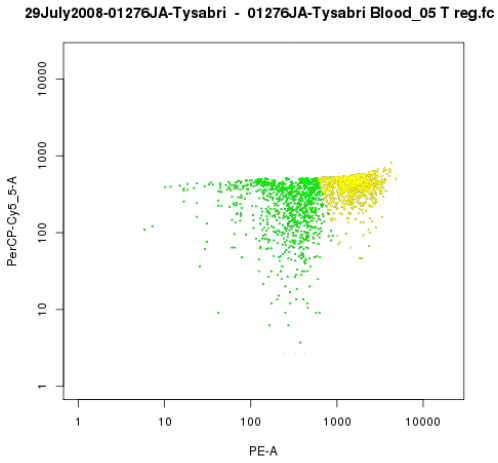

Abbildung 4.12.: Ergebnis von K-Means

4.11). Danach werden alle Datenpunkte oberhalb der Regressionsgeraden entfernt. Die beiden verbleibenden Cluster können dann mithilfe des K-Means-Algorithmus (vgl. Abschnitt 2.4.1) gefunden werden. Die Abbildung 4.12 zeigt das Ergebnis nach Anwendung von K-Means.

Schritt 4 - Histogramme analysieren

Im letzten Schritt werden die Signale der Tcon-, Tr1- und Treg-Zellen als Histogramme dargestellt, um diese in CD39+ und CD39- zu unterscheiden. In der Abbildung 4.13 sind die

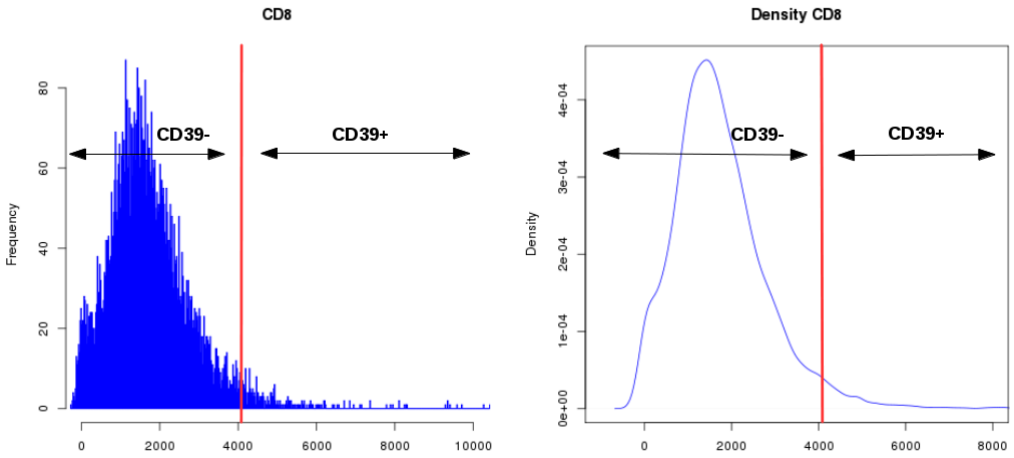

Abbildung 4.13.: Bestimmen der CD39+ und CD39- Zellen anhand der Dichtefunktion

Signale der CD8-Zellen als Histogramm dargestellt. Um die Signale in CD39+ und CD39- zu unterscheiden, wird für das Histogramm eine Dichtefunktion berechnet, die ebenfalls in der Abbildung gezeigt wird. Diese Dichtefunktion verfügt über ein Maximum, dessen Abstand zum linken Minimum verdoppelt wird, um den Schwellenwert zu berechnen, anhand dessen die Aufteilung in CD39+ und CD39- erfolgt.

5. Realisierung

Dieses Kapitel gibt einen Einblick in die Umsetzung der verschiedenen Software-Komponenten, die im Rahmen der Masterarbeit implementiert wurden. Der erste Abschnitt zeigt das Framework, das für die Analyse der Daten entwickelt wurde. Im zweiten Abschnitt wird das Analyseprogramm vorgestellt, das die am ZMNH eingesetzte Analyse-Strategie umsetzt. Im letzten Abschnitt wird die Visualisierungsplattform vorgestellt, die es ermöglicht, Analyseergebnisse und Plots über eine Weboberfläche zu betrachten.

5.1. Framework für die Analyse von FACS-Daten

Die Implementierung des Frameworks für die Analyse von FACS-Daten erfolgte in der Programmiersprache R (vgl. Abschnitt 2.6.1). Für R stehen eine Reihe von Bibliotheken zur Verfügung, die für die Analyse von Daten aus der Durchflusszytometrie (FACS-Daten) eingesetzt werden können (siehe Abschnitt 2.5 und 2.6.2). Eine Anforderung an die Software ist, dass sie unter dem Betriebsystem Linux lauffähig ist (siehe Abschnitt 3.4). Diese Bedingung ist erfüllt, da der Interpreter von R auf Linux portiert ist. Eine weitere Anforderung ist, dass das Analyseprogramm ohne grafische Oberfläche lauffähig ist, was dadurch erfüllt ist, dass R-Programme über die Kommandozeile gestartet werden können und keine grafische Oberfläche zur Laufzeit benötigen.
Die in Abschnitt 4.1 entwickelte Architektur für das Framework wurde bei der Realisierung um einige Komponenten ergänzt. Einen Überblick über die Komponenten des Frameworks und deren Schnittstellen gibt das Komponentendiagramm in der Abbildung 5.1.
Die Analyse einer einzelnen FCS-Datei kann grob in folgende Schritte aufgeteilt werden:

1. FCS-Datei öffnen
2. Qualitätsprüfung durchführen
3. Cluster von Lasersignalen in den Messdaten finden
4. Grafiken erzeugen
5. Ergebnisse abspeichern

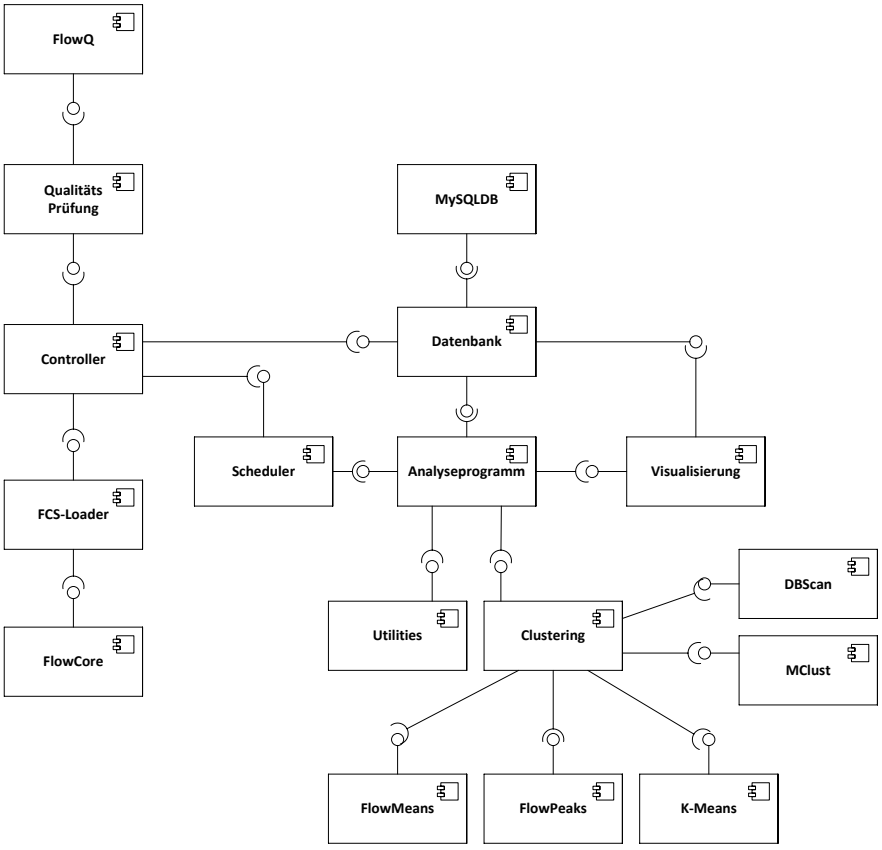

Abbildung 5.1.: Komponenten des Frameworks

Die Grundidee des Frameworks ist, dass durch den Benutzer des Frameworks die Anwendungslogik zum Erkennen der Zellpopulationen vorgegeben wird. Dabei können spezifische Cluster- und Filteralgorithmen ausgewählt und für die jeweilige Zellpopulation optimiert werden.
Gestartet wird die Analyse durch den Controller. Dieser kümmert sich um die parallele Ausführung mehrerer Instanzen des Analyseprogramms und übergibt diesem eine FCS-Datei, die analysiert werden soll.
Das Framework vereint eine Vielzahl von Bibliotheken und stellt für diese eine Programmierschnittstelle bereit. Beispielsweise bietet die Clustering-Komponente eine einheitliche Schnittstelle für verschiedene Cluster-Algorithmen, wodurch es mit geringem Änderungsaufwand möglich ist, die Algorithmen im Analyseprogramm auszutauschen.

Ziel des Frameworks ist es, das »Don't repeat yourself«-Prinzip zu ermöglichen. Wiederkehrende Aufgaben, wie das Erzeugen von Grafiken oder das Speichern von Daten in der Datenbank, können mithilfe von verschiedenen Komponenten des Frameworks implementiert werden.
In den folgenden Abschnitten werden die einzelnen Komponenten des Frameworks im Detail erläutert.

5.1.1. Controller

Der Controller ist ein wichtiger Bestandteil des entwickelten Frameworks, da er den Ablauf der Analyse steuert. Eine Analyse besteht aus mehreren Schritten, die durch den Controller initiiert werden.

Analyse in der Datenbank anlegen

Zu Beginn wird die Analyse in der Datenbank angelegt. Hierfür verwendet der Controller die Schnittstelle der Datenbank-Komponente. Beim Speichern wird ein Name, der die Analyse eindeutig identifizert, mit in der Datenbank hinterlegt. Dieser wird als Parameter beim Starten der Analyse angegeben und solte für jede Analyse unterschiedlich sein.

FCS-Dateien öffnen

Als weiterer Parameter wird entweder eine einzelne FCS-Datei oder ein Verzeichnis mit mehreren FCS-Dateien übergeben. Sofern ein Verzeichnis angegeben wurde, muss zusätzlich ein Suchkriterium definiert werden, anhand dessen die zu analysierenden FCS-Dateien gefunden werden können. Dieses Kriterium kann ein bestimmter Eintrag in den Metadaten, wie zum Beispiel der Name der Probe, sein.
Der Zugriff auf die FCS-Dateien erfolgt mithilfe der »FCS-Loader«-Komponente, die verschiedene Schnittstellen zum Öffnen der Dateien anbietet.

Qualitätsprüfung

Bevor eine FCS-Datei zur Auswertung an das Analyseprogramm übergeben wird, wird geprüft, ob sie verschiedene Qualitätsmerkmale erfüllt. Hierdurch können fehlerhafte Messungen vom Analyseprozess ausgeschlossen werden. Zur Qualitätsprüfung werden die FCS-Dateien an die QA-Komponente übergeben, die für jede FCS-Datei einen Rückgabewert liefert, der steuert, ob die FCS-Datei zur Analyse freigegeben oder gesperrt wird.

Starten des Analyseprogramms

Die Anwendungslogik für die Analyse der FCS-Datei wird durch die »Multithreading-Manager«-Komponente gestartet. Diese koordiniert die parallele Ausführung, damit die Rechenleistung mehrerer Prozessoren ausgenutzt werden kann.

Beenden der Analyse

Nachdem alle Aufrufe des Analyseprogramms beendet wurden, kann der gesamte Analyseprozess terminieren. Im letzten Schritt wird die Analyse als beendet in der Datenbank markiert. Die geöffneten FCS-Dateien werden geschlossen und die Analyse wird beendet.

5.1.2. FCS-Loader

Der FCS-Loader bietet eine Reihe von Methoden, um auf FCS-Dateien zuzugreifen. Die Komponente basiert auf der Bibliothek »FlowCore« (siehe Abschnitt 2.6.2), mit der FCS-Dateien ausgelesen werden können. Einzelne FCS-Dateien werden durch Klasse »FlowFrame« repräsentiert. Diese stellt eine Reihe von Methoden bereit, um auf die gemessenen Lasersignale und die Metadaten der FCS-Datei zuzugreifen.

Öffnen einer einzelnen Datei

Bei der Analyse einer einzelnen Datei wird diese durch den FCS-Loader aus dem Dateisystem geladen und als »FlowFrame«-Datenstruktur zurückgegeben.

Öffnen eines Verzeichnisses

Bei der Analyse eines Verzeichnisses kann ein Suchkriterium definiert werden, anhand dessen bestimmte FCS-Dateien für die Analyse ausgewählt werden. Sofern kein Kriterium angegeben wurde, werden alle FCS-Dateien in dem Verzeichnis analysiert.
Um die FCS-Dateien zu finden wird über das angegebene Verzeichnis und alle Unterverzeichnisse iteriert. Hierbei wird jede FCS-Datei geöffnet und geprüft, ob sie dem Suckriterium entspricht. Sofern dies der Fall ist, wird die Datei zu der Liste mit den zu analysierenden Dateien hinzugefügt. Hierbei wird der absolute Pfad zur Datei angegeben und nicht der geöffnete Dateideskriptor.
Im Listing 5.1 ist der Pseudocode der Klasse AnalysationFiles dargestellt. Über die Methode

»addFile(fileName)« wird ein absoluter Pfad zu der Liste der zu analysierenden Dateien hinzugefügt. Die Methode »getFile()« gibt ein FlowFrame-Objekt zurück, das eine FCS-Datei enthält. Hierdurch wird die FCS-Datei erst geöffnet, wenn auf sie zugegriffen werden soll. Dadurch müssen nicht alle Dateien im Speicher vorgehalten werden und die Ressourcen des Betriebssystems werden geschont.

```
class AnalysationFiles {
        fileNames = list()

        function addFile(fileName) {
                fileNames.push(fileName)
        }

        function getFile() {
                fileName = filesNames.pop()
                # read.FCS() gibt einen FlowFrame zurueck
                fcsFile = read.FCS(fileName)
                return fcsFile
        }
}
```

Listing 5.1: Pseudocode der Liste mit FCS-Dateien

5.1.3. Qualitätsprüfung

In der Komponente zur Qualitätsprüfung sind eine Reihe von Prüfungen implementiert, mit denen festgestellt wird, ob eine FCS-Datei für die Analyse geeignet ist.

Minimale Anzahl von gemessenen Signalen

Ein wichtiges Qualitätsmerkmal ist, ob eine minimale Anzahl an Signalen gemessen wurde. Dieser Wert liegt häufig bei ungefähr 10.000 gemessenen Zellen. Bei einer geringeren Anzahl an Signalen ist die Messung nicht mehr repräsentativ und würde das Ergebnis der Analyse verfälschen.

Bestimmte Kanäle vorhanden

Jede FCS-Datei enthält Signale bestimmter Dektoren (Kanäle). Eine FCS-Datei wird nur für die Analyse freigegeben, wenn sie die Daten der zu untersuchenden Kanäle auch enthält.

FlowQ

FlowQ (siehe Abschnitt 2.6.2) kann nur eingesetzt werden, wenn mehrere FCS-Dateien analysiert werden. Einzelne Dateien können nicht mit FlowQ geprüft werden, da für die Qualitätsprüfung immer mehrere Dateien miteinander verglichen werden. Hierfür berechnet FlowQ den Mittelwert der Häufigkeiten der Signale. Für jede Datei wird dann geprüft, ob sie um einen signifikanten Wert vom Mittelwert abweicht. Ist dies der Fall, wird sie von der weiteren Analyse ausgeschlossen. Weiterhin wird geprüft, ob in der Datei starke Abweichungen von Signalstärken im Gegensatz zu den anderen Dateien vorhanden sind.

Ergebnis der Qualitätsprüfung

Die Ergebnisse der Qualitätsprüfung werden an den Controller zurück gegeben, wobei ein Wahrheitswert geliefert wird, der steuert, ob die Dateien in der weiteren Analyse berücksichtigt oder ausgeschlossen werden sollen.

5.1.4. Scheduler

Der »Scheduler« hat die Aufgabe, die Ausführung des Analyseprogramms zu koordinieren.

Multithreading

Wenn die Analyse der FCS-Dateien auf einem System mit mehreren Prozessoren ausgeführt wird, können mehrere Prozesse des Analyseprogramms gestartet werden, wodurch sich die Laufzeit verringert. Die Anzahl der gestarteten Prozesse entspricht hierbei der Anzahl der Prozessoren.
Pro Prozess wird eine Datei analysiert, wobei jeder Prozess autark arbeitet. Sobald ein Prozess beendet ist, wird vom Scheduler ein neuer gestartet. Dies wird solange wiederholt, bis alle Dateien analysiert wurden.

5.1.5. Analyseprogramm

Das Analyseprogramm implementiert die Anwendungslogik, mit der die FCS-Dateien analysiert werden. Als Parameter erhält das Analyseprogramm eine FCS-Datei, die in mehreren Schritten analysiert wird. Das Listing 5.2 zeigt Auszüge aus einem Analyseprogramm.
Die Funktionen »analyseFolder()« und »analyseFile()« rufen die generischen Funktionen, zum Analysieren einer Datei oder eines Verzeichnisses auf, die durch das Framework implementiert werden. Als Parameter wird die Callback-Funktion übergeben, die vom Framework für die

Analyse der FCS-Datei aufgerufen wird. In dem Beispiel ist es die Funktion »analyseFileCallback()«.
Im Analyseprogramm in Listing 5.2 werden die Signale von zwei Kanälen geplottet und der FlowPeaks-Cluster-Algorithmus auf die Kanäle »channel1« und »channel2« angewendet. Das Ergebnis des Clustering wird im Anschluss in einem Plot ausgegeben.

```
source("AnalysationFramework/analysationFunctions.R")
source("AnalysationFrameowrk/clusteringFunctions.R")

analyseFileCallback <- function(fcsFile, analysationID) {
        # Analyse der FCS-Datei, die als FlowFrame in der
        # Varibalen fcsFile uebergeben wird.
        plotFCS("channel1", "channel2", fcsFile)
        result <- doFlowPeaksClustering("channel1",
                                        "channel2",
                                        fcsFile)
        plotClusteringResult(result)
        # [...]
}

analyseFolder <- function(path, analysationName) {
     analyseFolderGeneric(path,
                          analysationName,
                          analyseFileCallback,
                          "Tubename='T reg'")
}

analyseFile <- function(filename, analysationName) {
     analyseFileGeneric(filename,
                        analysationName,
                        analyseFileCallback)
}
```

Listing 5.2: Grundgerüst des Analyseprogramms

Hierbei werden eine Reihe Komponenten des Frameworks verwendet. Die Zellpopulationen werden mit spezifischen Cluster-Algorithmen erkannt, die über die Clustering-Komponente eingebunden werden. Mit der der Visualisierungs-Komponente werden Grafiken erzeugt, anhand deren der Analyseprozess später betrachtet werden kann.

5.1.6. Datenbank

Mit der Datenbank-Komponente ist es möglich, Analyseergebnisse in einer relationalen Datenbank abzuspeichern. Als Datenbanksoftware wird MySQL eingesetzt, wobei die Komponente auch für die Verwendung einer anderen Datenbank angepasst werden kann.

RMySQL

Der Zugriff auf die MySQL-Datenbank erfolgt mit Funktionen aus dem RMySQL[1]-Paket. Im Listing 5.3 ist der Zugriff auf die Datenbank unter Verwendung der Datenbank-Komponente beispielhaft dargestellt. In dem Beispiel wird eine Analyse in der Datenbank angelegt, die den Namen »Analyse 1« erhält (Zeile 4). In Zeile 7 wird eine Datei, die analysiert wurde, in der Datenbank abgelegt und der Analyse zugeordnet. Zum Abschluss des Beispiels wird in Zeile 12 die Anzahl der Lymphozyten gesetzt, die für die Datei festgestellt wurde.

```
1  source(AnalysationFramework/database.R,)
2
3  # Analyse in der Datenbank anlegen
4  analysation_id <- dbCreateAnalysation("Analyse 1")
5
6  # FCS-Datei in der Datenbank anlegen
7  file_id <- dbCreateAnalysationFile("/data/facs/29932.fcs",
8                                     analysation_id)
9
10 # Die Anzahl der Lymphozyten festlegen,
11 # die in der FCS-Datei gefunden wurden
12 dbUpdateNumberOfLymphocytes(file_id, 4950)
```

Listing 5.3: Speichern von Daten in der MySQL-Datenbank

5.1.7. Clustering

In der Clustering-Komponente sind eine Reihe von Cluster-Algorithmen vereint, die über einheitliche Schnittstellen angesprochen werden können.
Die Schnittstelle zu den Algorithmen ist beispielhaft im Listing 5.4 aufgeführt. Über den Parameter »fcsFile« wird die FlowFrame-Datenstruktur übergeben, die eine FCS-Datei repräsentiert. Die Parameter »channelX« und »channelY« enthalten die Namen der Kanäle, deren Daten geclustert werden sollen. Die grafische Ausgabe kann über die Parameter »createPlots«

[1] http://cran.r-project.org/web/packages/RMySQL/index.html Abgerufen am 23.08.2013

deaktiviert werden.
Sofern Grafiken erzeugt werden, kann mit »xlim« und »ylim« der Wertebereich der X- und Y-Achse angegeben werden. Über den Parameter »log« wird festgelegt, ob die Achsen des Plots in der logarithmischen Skala aufgetragen werden sollen.

```
doClusteringAlgorithmX <- function(fcsFile,
                        channelX="FSC-A",
                        channelY="SSC-A",
                        createPlots=TRUE,
                        log="", xlim=c(), ylim=c(),
                        ) {
                        [...]
}
```

Listing 5.4: Schnittstelle der Cluster-Algorithmen

Ergebnis des Clusterings

Als Rückgabewert liefert die Schnittstelle des jeweiligen Cluster-Algorithmus eine Liste mit Zahlen. Jede Zahl repräsentiert hierbei eine Cluster-Zugehörigkeit, die von dem Algorithmus berechnet wurde. Diese kann dann mit den Daten zusammengeführt werden, um für jeden Datenpunkt eine Cluster-Zugehörigkeit zu ermitteln.
In der Tabelle 5.1 ist beispielhaft die Zuordnung der Messwerte zu den unterschiedlichen Clustern aufgeführt.

Kanal1	Kanal 2	Cluster
5	5	1
4	6	1
50	60	2
54	58	2

Tabelle 5.1.: Messwerte mit Cluster-Zugehörigkeiten

K-Means

Für die Integration von K-Means in die Clustering-Komponente wurde die Standard R-Implementierung von K-Means verwendet, die auf der Publikation von Hartigan und Wong (1979) basiert. Für die Berechnung der Cluster werden als Parameter die mehrdimensionalen Daten und die Anzahl der Cluster-Mittelpunkte übergeben.

MClust (Model based clustering)

Bei MClust handelt es sich um ein »Model based clustering«-Verfahren (vgl. Abschnitt 2.4.2), wobei die Daten über eine Gaußsche Mischverteilung modelliert werden. Die Parameter der Mischverteilung werden in mehren Iterationen mithilfe des EM-Algorithmus (Expectation Maximization) berechnet.
Als Parameter nimmt die MClust-Implementierung die mehrdimensionalen Daten und optional die Anzahl der Cluster, die gefunden werden sollen, entgegen. Wird die Anzahl der Cluster nicht angegeben, versucht MClust die optimale Anzahl zu berechnen.

FlowPeaks

Zur Integration von FlowPeaks (vgl. Abschnitt 2.5.1) wurde die Implementierung aus dem Bioconductor-Projekt verwendet. Der Algorithmus benötigt außer den Eingabedaten keine weiteren Parameter, die Anzahl der Cluster wird von FlowPeaks berechnet.

FlowMeans

Bei FlowMeans (vgl. Abschnitt 2.5.2) wurde ebenfalls die Implementierung aus dem Bioconductor-Projekt verwendet, um den Algorithmus in die Clustering-Komponente zu integrieren. Beim Ausführen von FlowMeans muss die Anzahl der Cluster nicht mit angegeben werden, als Parameter müssen ausschließlich die zu clusternden Daten übergeben werden.

5.1.8. Visualisierung

Mit der Visualisierungs-Komponente können Grafiken erzeugt werden, die als PNG-Datei im Dateisystem abgespeichert werden. Damit die Plots später über die Weboberfläche betrachtet werden können, wird der absolute Pfad zum Plot in der Datenbank abgespeichert. Hierbei wird die FCS-Datei, die analysiert wurde, in der Datenbank referenziert.
Zum Erzeugen der Grafiken wird die Standard-Implementierung von R verwendet.

5.1.9. Utilities

In der »Utilities«-Komponente sind verschiedene Methoden vereint, mit denen FCS-Dateien manipuliert werden können. Diese ermöglichen es, die gemessenen Signale der FCS-Datei anhand bestimmter Signalstärken zu filtern, oder die Signale zu logarithmieren.

5.2. Implementierung des automatisierten Analyseverfahrens

Für die Analyse der am ZMNH erhobenen Messdaten aus der Durchflusszytometrie wurde ein Analyseprogramm in der Programmiersprache R entwickelt. Dieses basiert auf dem im vorherigen Abschnitt beschriebenen Framework und entspricht den im Abschnitt 3.3 beschriebenen fachlichen Anforderungen. Hierfür wurde die im Abschnitt 4.3 vorgestellte Analysestrategie implementiert, wobei alle Schritte der Analyse umgesetzt wurden. Im Anhang A sind alle Plots, die bei der Analyse einer FCS-Datei erzeugt werden aufgeführt.
Bei der Implementierung wurden mehrere Methoden entwickelt, die die einzelnen Schritte der Analyse durchführen und die Signale der FCS-Datei filtern, bis die Häufigkeiten der Zellpopulationen bestimmt werden können. Das Listing 5.5 zeigt einen Auszug aus dem Analyseprogramm. Die Analyse ist in der Methode »tregAnalysation()« implementiert, an die eine FCS-Datei übergeben wird. Die einzelnen Schritte der Analyse sind in den Methoden »gatingStepX()« implementiert, an die die FCS-Datei weitergereicht wird. Als Rückgabewert liefern die Methoden eine FCS-Datei, deren Signale reduziert wurden.

```
1 tregAnalysation <- function(fcsFile) {
2   fcsFile <- gatingStep1(fcsFile)
3   fcsFile <- gatingStep2(fcsFile)
4   fcsFile <- gatingStep3(fcsFile)
5   gatingStep4(fcsFile)
6 }
```

Listing 5.5: Auszug aus dem Analyseprogramm

Analyseschritt im Detail

Im Listing 5.6 ist ein Auszug des ersten Analyseschritts dargestellt, hierbei wird ein Plot der FCS-Datei erzeugt und Cluster in den Daten mit dem FlowPeaks-Algorithmus gefunden. Die Methode »doFlowPeaks()« führt das Clustering durch und gibt das Ergebnis zurück. Dies wird verwendet, um mit der Methode »findRightCluster()« das Cluster zu finden, dessen Mittelpunkt den höchsten X-Wert hat. Die Größe des gefundenen Clusters entspricht der Anzahl der Lymphozyten im Blut. Dieser Wert wird mit der Methode »dbUpdateNumberOfLymphocytes()« in der Datenbank gespeichert. Mit der Methode »filterByCluster()« werden die Datenpunkte des Clusters aus der FCS-Datei extrahiert und eine neue FCS-Datei erzeugt, die von der Methode zurückgegeben wird. Die gefilterte FCS-Datei wird dann von

der »gatingStep1()«-Methode an den Aufrufer zurückgegeben.

```
gatingStep1 <- function(fcsFile) {
  createPlot(fcsFile, "APC-A", "FSC-A")
  flowPeaksResult <- doFlowPeaks(fcsFile,
                                 channelX="APC-A",
                                 channelY="FSC-A")
  cluster <- findRightCluster(flowPeaksResult)
  fcsFile <- filterByCluster(fcsFile, cluster, flowPeaksResult)
  dbUpdateNumberOfLymphocytes(sizeof(cluster))
  return (fcsFile)
}
```

Listing 5.6: Analyseschritt im Detail

In dem Beispiel wurde der FlowPeaks-Algorithmus eingesetzt, um das Cluster zu finden. Im gesamten Analyseprogramm wurden zusätzlich die Cluster-Algorithmen K-Means und DBSCAN eingesetzt, die über die Clustering-Komponente des Frameworks (vgl. Abschnitt 5.1.7) in die Analyse eingebunden werden.

Filtern bestimmter Cluster

In den einzelnen Schritten der Analyse müssen nicht nur die Datenpunkte verschiedenen Clustern zugeordnet werden, sondern zusätzlich bestimmte Cluster für die weitere Analyse aus den Daten extrahiert werden. Welches Cluster hierbei weiter verarbeitet werden soll, wurde zuvor definiert, zum Beispiel, indem das Cluster, dessen Mittelpunkt einen hohen Y-Wert hat. In der Abbildung 5.2 ist das Ergebnis der Anwendung von FlowPeaks dargestellt. Es ist zu erkennen, dass der Algorithmus insgesamt fünf Cluster gefunden hat, die in der Abbildung mit Zahlen beschriftet sind. Die blau eingekreisten Cluster mit der Beschriftung 4 und 5 enthalten nur wenige Datenpunktepunkte und können somit als Fehler angesehen werden. Die Abbildung 5.3 zeigt das gewünschte Ergebnis nach Filterung des Clusters.
Bei der Selektion der Cluster muss beachte werden, dass es Fehler im Clustering geben kann. Eine falsche Selektion kann verhindert werden, indem das Cluster mit einem hohen Y-Wert selektiert wird, das zusätzlich die Bedigung erfüllt, dass es mindestens 100 Datenpunkte enthalten muss.

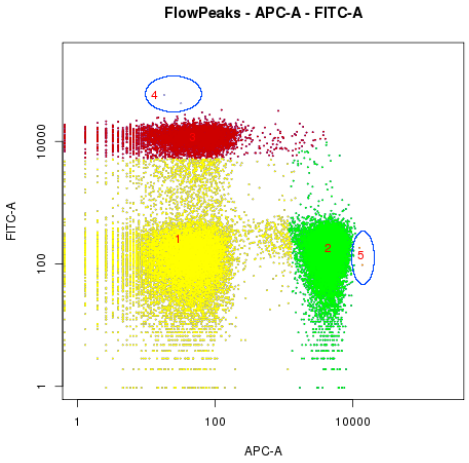

Abbildung 5.2.: Clustering mit Flowpeaks

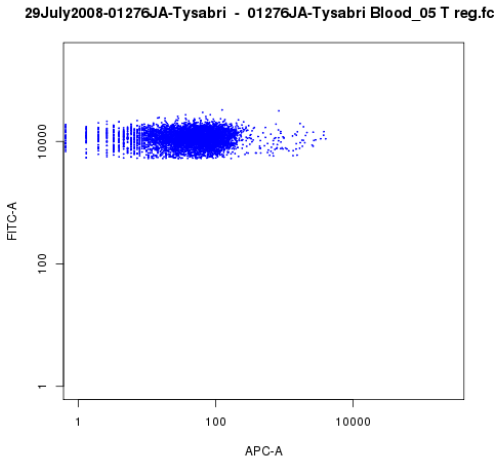

Abbildung 5.3.: Filtern des Clusters

Bausteinprinzip

Im Idealfall sollte das Analyseprogramm nur aus einzelnen Programmbausteinen (siehe Abbildung 5.4) bestehen, die aneinandergereiht die Logik der Analyse bilden. Diese Programmbausteine sind Methoden für das Erzeugen von Plots, Analysieren der Daten mit Cluster-Algorithmen, Selektion des Clusters und Speichern der Ergebnisse in der Datenbank.

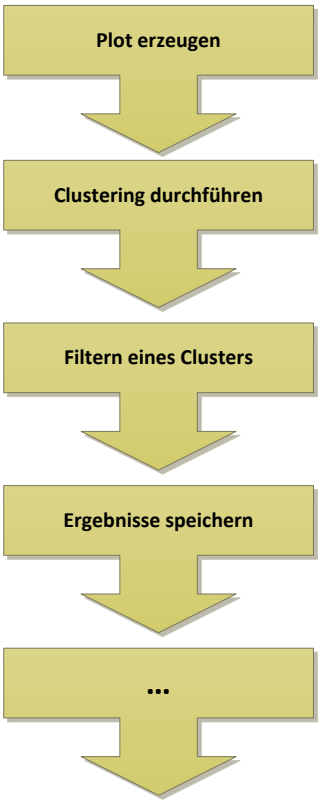

Abbildung 5.4.: Programmbausteine einer Analyse

5.3. Visualisierungsplattform

Zur Auswertung der Analysen wurde eine Visualisierungsplattform entwickelt, die es dem Benutzer ermöglicht, die mit dem Analyseprogramm erzeugten Ergebnisse zu betrachten. Eine Anforderung an die Software ist der plattformunabhängige Zugriff auf die Analyseergebnisse (vgl. Abschnitt 3.4). Auf die Webanwendung kann mit einem beliebigen Webbrowser zugegriffen werden, wordurch diese Anforderung erfüllt ist. Im Anhang B sind Screenshots von der Weboberfläche aufgeführt.
Für das Design wurde das von Twitter entwickelte CSS-Framework Bootstrap[2] verwendet. Dieses liefert Vorgaben für die Schriftgröße und -art und enthält eine Reihe von wiederverwendbaren Komponenten, mit denen die Navigation realisiert wurde.

[2] siehe http://getbootstrap.com/ abgerufen am 11.09.2013

Funktionalität

Die Funktionalität der Visualisierungsplattform ist auf das Anzeigen der Analyseergebnisse, die in der Datenbank gespeichert und in Form von Grafiken im Dateisystem erzeugt wurden, beschränkt.
Auf der Startseite wird dem Benutzer eine Dropdown-Auswahlbox präsentiert, über die eine Analyse ausgewählt werden kann (siehe Abbildung 5.5), die zuvor in der Datenbank angelegt wurde. Der aktuelle Fortschritt der laufenden Analyse, kann über die Weboberfläche eingesehen werden.

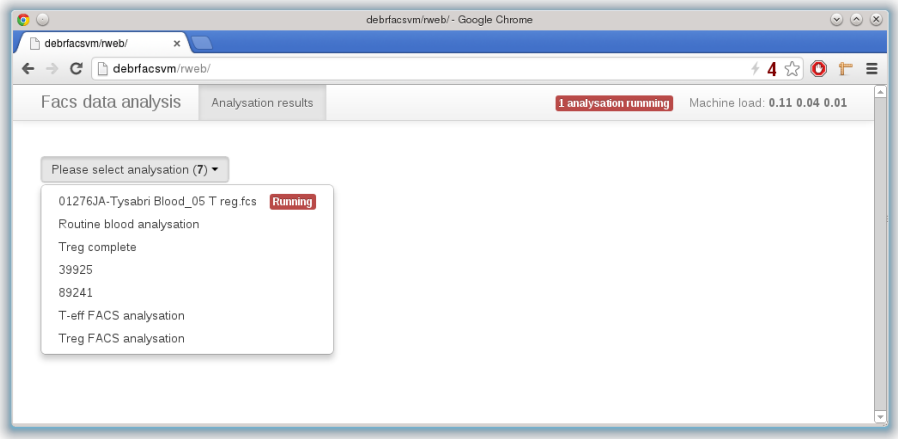

Abbildung 5.5.: Auswahl einer Analyse über eine Dropdown-Auswahlbox

Nachdem der Benutzer eine Analyse über die Dropdown-Auswahlbox selektiert hat, wird diese ihm präsentiert (siehe Abbildung 5.6). In der Detailseite zu einer Analyse kann der Benutzer den Namen und den aktuellen Status, einsehen. Pro analysierter Datei werden dem Benutzer die Häufigkeiten der unterschiedlichen Zellpopulationen tabellarisch dargestellt. Zusätzlich können die durch das Analyseprogramm erzeugten Grafiken betrachtet werden (siehe Abbildung 5.7).

5.4. Softwaretest

Beim Softwaretest wurden das Analyseprogramm und die Visualisierungsplattform auf Fehler überprüft. Beim Test der Anwendungen wurde in vier unterschiedlichen Prüfebenen unterschieden (vgl. Hoffmann (2008) Abschnitt 4.2.1):

- Unit-Tests
- Integrationstests
- Systemtests
- Abnahmetest

Beim Unit-Testing werden einzelne Methoden geprüft, dies erfolgt durch Aufruf der Methode und Prüfen des Rückgabewertes. Der Integrationstest beinhaltet das Prüfen, ob die implementierten Module miteinander kompatibel und lauffähig sind. Der Systemtest überprüft im nächsten Schritt, ob das System als ganzes lauffähig ist und ob es alle Anforderungen erfüllt sind, die spezifiziert wurden. Zulezt erfolgt der Abnahmetest, bei dem die Software durch den Fachanwender oder Kunden abgenommen und getestet wird.

Testen des Frameworks und des Analyseprogramms

Mit RUnit[3] steht ein Framework für die Programmiersprache R zur Verfügung, mit der Unit-Tests implementiert werden können. Dies erfolgte für einen Teil der Komponenten, die im Framework enthalten sind. Es wurden Unit-Tests zum Prüfen der Qualitätssicherungs-Komponente oder der Komponente zum Auslesen von FCS-Dateien implementiert. Der Integrationstest erfolgte durch Implementierung des Analyseprogramms, hierbei wurden alle Komponenten des Frameworks eingesetzt und zusammengeführt.
Am umfangreichsten wurde der Systemtest durchgeführt. Hierbei wurde geprüft, ob das Analyseprogramm korrekt arbeitet und korrekte Werte für die Zellpopulationen bestimmen kann. Verifiziert wurde dieses anhand von manuell durchgeführten Analysen, mit denen dann die Werte verglichen wurde. Zusätzlich wurden mehrere hundert FCS-Dateien analysiert, um zu prüfen, ob die Software bei einer Vielzahl von Eingabedaten korrekt arbeitet.
Beim Abnahmetest wurden die Ergebnisse der Analyse den Fachanwendern präsentiert.

Testen der Visualisierungsplattform

Die Visualisierungsplattform dient ausschließlich zum Anzeigen der Analyseergebnisse und hat wenig Anwendungslogik, die durch Unit-Tests abgeprüft werden kann. Aus diesem Grund

[3] http://cran.r-project.org/web/packages/RUnit/index.html abgerufen am 10.09.2013

wurden keine Unit- und Integrationstests für die Visualisierungsplattform durchgeführt. Beim Systemtest wurde geprüft, ob alle erzeugten Grafiken in der Weboberfläche angezeigt werden und ob die Werte aus der Datenbank korrekt angezeigt werden. Der Abnahmetest beinhaltete die Vorstellung der Visualisierungsplattform bei den Fachanwendern.

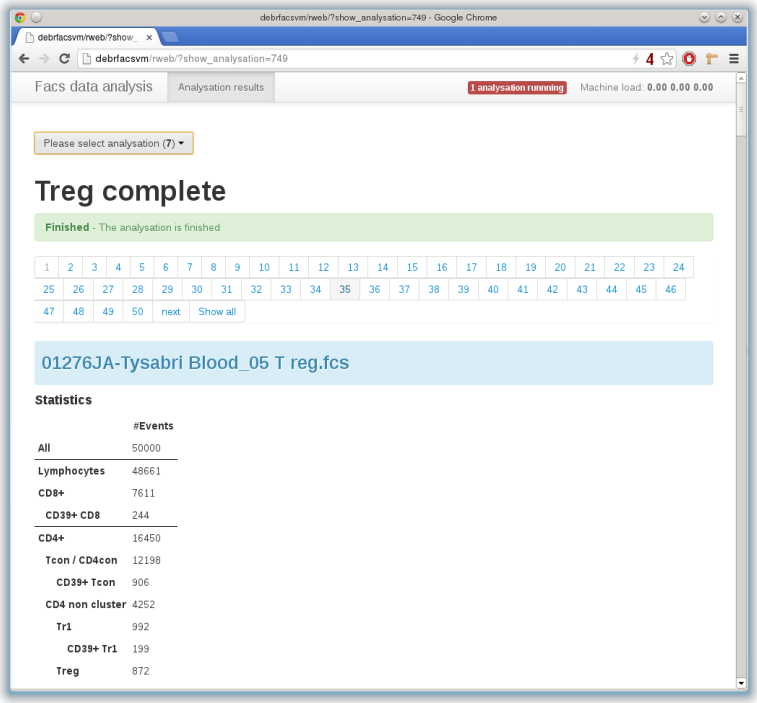

Abbildung 5.6.: Analyse im Detail

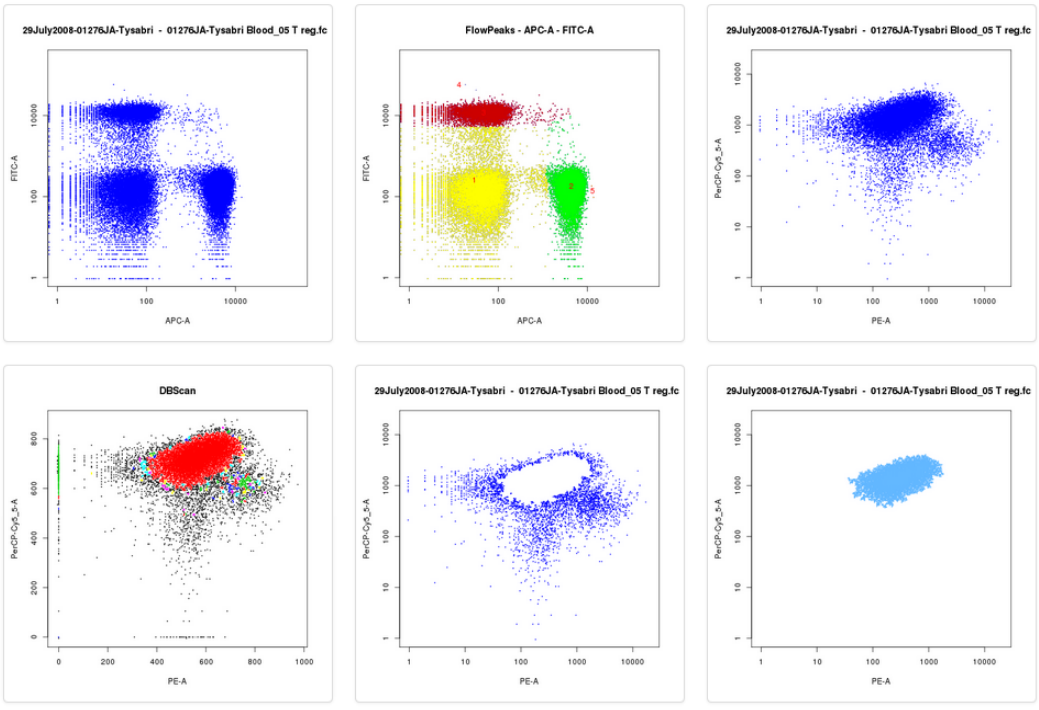

Abbildung 5.7.: Vom Analyseprogramm erzeugte Dotplots

6. Evaluierung

Damit neue wissenschaftliche Erkenntnisse für die Multiple Sklerose Forschung aus dem großen Datenbestand an Messdaten aus der Durchflusszytometrie abgeleitet werden können, muss das Analyseprogramm korrekte Werte für die Zellpopulationen liefern. Um dies zu verifizieren wurden eine Reihe von FCS-Dateien manuell mit der Software FlowJo ausgewertet und die Ergebnisse mit dem automatisierten Analyseverfahren verglichen. Zusätzlich wurde eine Messreihe mit mehreren hundert Dateien analysiert, um die Stabilität des Analyseprogramms bezüglich der Vielfältigkeit der Daten zu überprüfen. Zum Abschluss dieses Kapitels wird das entwickelte Analyseverfahren bewertet und mögliche Verbesserungen vorgestellt.

6.1. Verifikation des Analyseverfahrens

Das entwickelte Analyseverfahren wurde mithilfe einer manuell durchgeführten Studie verifiziert. Hierfür wurden einhundert FCS-Dateien ausgewählt, die manuell und automatisiert ausgewertet wurden.

Manuelle Analyse

Die manuelle Auswertung erfolgte mit der Software FlowJo, wobei die im Abschnitt 3.3 beschriebene Analysestrategie angewendet wurde. Die Häufigkeiten der unterschiedlichen Zellpopulationen wurden aus FlowJo exportiert und als Excel-Datei gespeichert. Ein Auszug aus der Excel-Datei kann in der Tabelle 6.1 eingesehen werden.

File	CD4	CD8	Tcon	Treg	Tr1	CD39+Tcon	CD39+Treg
83296.fcs	11629	4780	10842	438	284	48	37
83927.fcs	15585	5877	14301	1002	159	4	79
83961.fcs	26928	7757	25107	1413	283	72	314
85977.fcs	24761	5963	22501	1750	316	9	106
…	…	…	…	…	…	…	…

Tabelle 6.1.: Ergebnisse der manuellen Analyse von 100 FCS-Dateien (Auszug)

Automatisierte Analyse

Bei der automatisierten Auswertung wurden die Häufigkeiten der Zellpopulationen mit dem Analyseprogramm bestimmt. Die erzeugten Plots zeigten, dass die Cluster-Algorithmen nicht immer ein optimales Ergebnis liefern. Die Abbildung 6.1 zeigt ein Beispiel, in dem der Algorithmus FlowPeaks nur zwei von drei Clustern erkannt hat. Das fehlende Cluster ist in der Abbildung 6.2 durch einen roten Kreis hervorgehoben.
Ein weiteres Problem bestand darin, dass nicht alle Datenpunkte zum Cluster hinzugefügt wurden. Dies ist in den Abbildungen 6.3 und 6.4 dargestellt. Zu erkennen ist, dass das gefundene wesentlich kleiner als das optisch identifizierbare Cluster ist.
Durch Auswertung der durch das Analyseprogramm erzeugten Grafiken wurde festgestellt, dass die Cluster-Algorithmen bei ca. 85% der Eingabe-Dateien ein optimales Ergebnis geliefert haben und es in den restlichen Fällen zu Fehlern beim Clustering gekommen ist.

Gegenüberstellung der beiden Analysen

Das entwickelte Analyseprogramm speichert die Häufigkeiten der Zellpopulationen in einer MySQL-Datenbank ab. Bei der manuellen Analyse wurden die Ergebnisse in eine MS-Excel-Datei exportiert. Um die Ergebnisse beider Verfahren gegenüberzustellen, wurde aus der MS-Excel-Datei eine CSV-Datei erzeugt, die mit einem Python-Programm ausgelesen wird. Das Python-Programm greift auf die MySQL-Datenbank zu, um beide Datenquellen zusammenzuführen. Als Ergebnis wird eine CSV-Datei erzeugt, die eine Gegenüberstellung beider Analyseergebnisse enthält.
Die Tabelle 6.1 zeigt einen Ausschnitt aus der CSV-Datei. In dem Beispiel werden für jede FCS-Datei die Häufigkeiten der CD4-Zellpopulationen verglichen. In der Tabelle werden die

Filename	CD4 manuell	CD4 automatisch	CD4-Differenz	CD4-Differenz in %
83296.fcs	11629	12065	436	3%
83927.fcs	15585	16339	754	4%
83961.fcs	26928	27953	1025	3%

Tabelle 6.2.: Vergleich der manuell und automatisiert durchgeführten Analyse

manuell und automatisiert festgestellten Werte gegenübergestellt, darüber hinaus wird die Differenz beider Werte dargestellt. Dies gibt einen guten Überblick über die Abweichungen zwischen den beiden Analyseverfahren.
Die Ergebnisse haben gezeigt, dass es generell möglich ist, die Analysen zu automatisieren. Das entwickelte Analyseverfahren liefert für einen Großteil der Eingabedaten korrekte Werte

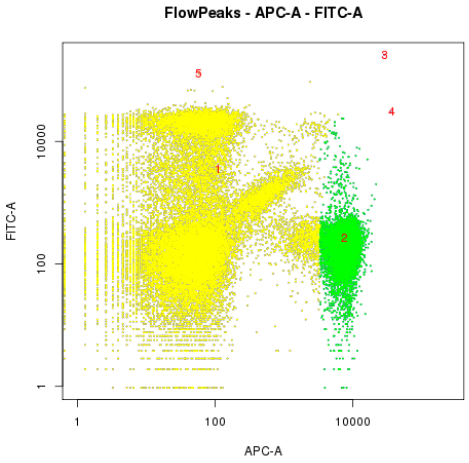

Abbildung 6.1.: Nicht optimales Ergebnis des Clusterings

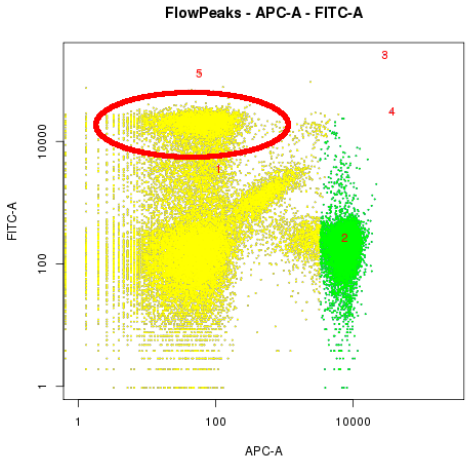

Abbildung 6.2.: Cluster wurde nicht gefunden

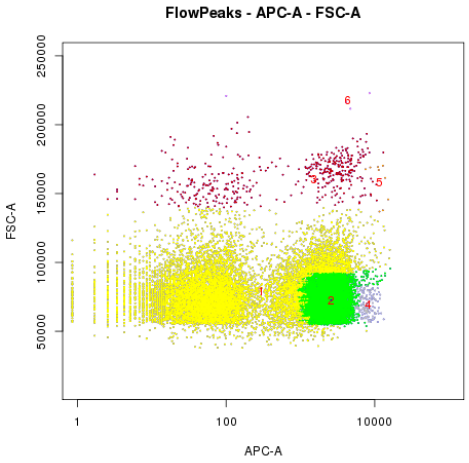

Abbildung 6.3.: Nicht alle Datenpunkte sind im Cluster enthalten

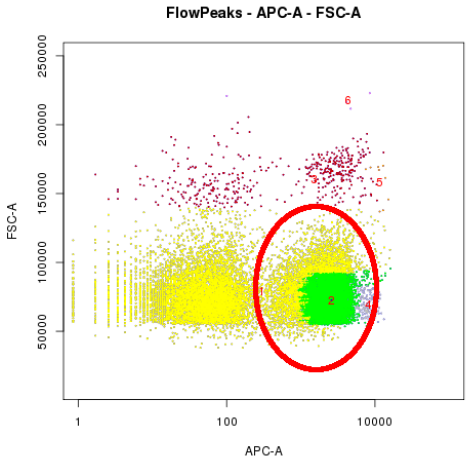

Abbildung 6.4.: Es soll ein größeres Cluster gefunden werden

für die Zellpopulationen. Nachteilig ist, dass es in ca. 15% der Analysen zu Fehlern beim Clustering gekommen ist, wodurch die Ergebnisse verfälscht wurden.

Analyse einer größeren Messreihe

Bei der Analyse einer größeren Messreihe wurden insgesamt 1500 FCS-Dateien mit dem Analyseprogramm ausgewertet. Die Analyse wurde auf einem leistungsstarken Server durchgeführt

und hatte eine Laufzeit von zwei Tagen.
Eine Schwierigkeit stellte der Speicherverbrauch dar, da es zu »Memory Leaks« kam und der Arbeitsspeicher von insgesamt acht Gigabyte nicht mehr ausreichend war. Dies Problem wurde behoben, indem das Analyseprogramm um Aufrufe des Garbage-Collectors von R ergänzt wurde, wodurch Speicher häufiger freigegeben wird.
Das Experiment hat gezeigt, dass die entwickelte Software eine Vielzahl von Datensätzen auswerten kann und es somit möglich ist, den umfangreichen Datenbestand an Messdaten effizienter auszuwerten.

6.2. Bewertung des entwickelten Analyseverfahrens

Das in Abschnitt 3.3 vorgestellte Analyseverfahren konnte mit der entwickelten Software erfolgreich automatisiert werden. Alle Auswertungsschritte konnten umgesetzt werden. Die Wahl eines Frameworks und damit eines generischen Ansatzes bietet die Möglichkeit, bei Bedarf mit geringem Programmieraufwand weitere Analyseprogramme zu implementieren.
Im Folgenden soll überprüft werden, inwieweit die in Abschnitt 3.4 an die Software gestellten nichtfunktionalen Anforderungen erfüllt sind. Bisher erfolgte die Auswertung am Desktop-Computer des Fachanwenders. Nach Umsetzung des Projektes kann nun das Analyseprogramm auf einem Server ausgeführt werden. Dies hat den Vorteil, dass die meist höhere Rechenleistung des Servers für die Analyse ausgenutzt werden kann.
Die manuellen Analysen waren bisher sehr zeitaufwändig und personengebunden, da sie nur von geschulten Fachpersonal durchgeführt werden konnten. Nun ist ist eine Analyse personenunabhängig und so performant, dass an einem Tag bis zu 300 FCS-Dateien ausgewertet werden können. Das stellt einen wesentlichen Effizienzgewinn und auch eine Minimierung des Fehlerpotenzials dar. Die Individualität der Analysen, je nach Fachanwender und dessen Erfahrungsschatzes, war ein weiter Kritikpunkt bei der Vorgehensweise vor Automatisierung. Das Analyseprogramm verfolgt daher jetzt eine klar definierte Analysestrategie und liefert reproduzierbare und vergleichbare Ergebnisse.
Es wurde darüber hinaus sichergestellt, dass ein plattformunabhängier Zugriff auf die Analyseergebnisse möglich ist, was eine weitere Anforderung war. Mit der webbasierten Visualisierungsplattform kann mit einem beliebigen Webbrowser auf die Ergebnisse zugegriffen werden. Das macht eine plattform- und standortunabhängige Auswertung der Ergebnisse möglich und bietet vor allem eine Möglichkeit der Skalierung des Einsatzgebietes des Analyseprogramms.
Die Verifikation des Verfahrens hat ergeben, dass das Analyseprogramm korrekte Werte für die Zellpopulationen liefern kann. Allerdings sind die Ergebnisse nicht immer optimal, wie

sich herausgestellt hat. Es bleibt nach jeder Auswertung anhand der erzeugten Grafiken zu überprüfen, ob die Analyse korrekt war. Der Aufwand für die Überprüfung ist gering, jedoch wäre es wünschenswert, zu 100% zuverlässige Werte für die Zellpopulationen zu erhalten.
Ausbaufähig ist auch die Usability der Analyseplattform. Das Analyseprogramm wird als Kommandozeilenprogramm auf einem Server ausgeführt, wodurch es für den Fachanwender schwierig ist, eigene Analysen zu starten und FCS-Dateien für die Analyse auszuwählen.
Die Visualisierungsplattform bietet derzeit nur die Möglichkeit der reinen Ergebnisbetrachtung, von hier aus können keine Analysen gestartet werden. Für den Anwender wäre es sicher in Zukunft von Vorteil, die Möglichkeit zu haben, Analysen eigenständig zu starten.

Cluster-Verfahren

Im Analyseprogramm wurden die Cluster-Algorithmen K-Means, DBSCAN und FlowPeaks eingesetzt (vgl. Abschnitt 4.3). K-Means wurde zur Partitionierung der Daten eingesetzt, wobei das Ergebnis des Algorithmus in allen Fällen zufriedenstellend war. Die eingesetzte Implementierung berechnet selbst bei mehreren Tausend Datenpunkten ein Ergebnis in maximal einer Sekunde, wodurch sie für den Einsatz im Analyseprogramm geeignet ist.
Der DBSCAN-Algorithmus wurde verwendet, um Cluster mit einer hohen Datendichte zu identifizieren. Für diese Aufgabenstellung hatte sich DBSCAN als optimal herausgestellt, da verschiedene Verfahren, wie z.B. FlowPeaks, MClust oder K-Means das Cluster nicht exakt identifiziert haben.
FlowPeaks, das Verfahren spziell für Daten aus der Durchflusszytometrie, wird im Analyseprogramm zweimal eingesetzt, um in unterschiedlichen Daten die Cluster zu finden. FlowPeaks bietet die Möglichkeit, in 50.000 Datenpunkten innerhalb weniger Sekunden Cluster zu identifizieren. Hierbei sind in 85% der Fälle die Ergebnisse optimal, in den verbleibenden 15% werden Cluster nicht exakt ausgewiesen (siehe Abschnitt 6.1).
Eine Möglichkeit, das Analyseverfahren weiter zu verbessern, besteht darin, statt FlowPeaks ein adaptiertes Cluster-Verfahren zu implementieren und anzuwenden. Dieses wäre auf die Identifizierung der vorgegebenen Cluster spezialisiert und würde, ähnlich wie FlowPeaks, verschiedene Cluster-Algorithmen vereinen.

Neuer Workflow

Aus der Integration der Analysesoftware und Visualisierungsplattform ergibt sich ein neuer Workflow für die Datenanalyse, der in Abbildung 6.5 dargestellt ist. Wie im bisherigen Workflow wird die Messung mit dem Durchflusszytometer durchgeführt, das mit FACSDiva

gesteuert wird. Das Ergebnis wird dann als FCS-Datei auf dem Fileserver abgespeichert. Im nächsten Schritt werden die FCS-Dateien ausgewertet. Dies erfolgte bisher mit FACSDiva oder FlowJo und wird im neuen Workflow mit dem Analyseprogramm direkt auf dem Server ausgeführt. Zu einem späteren Zeitpunkt können die Ergebnisse der Analyse über die Visualisierungsplattform betrachtet werden.

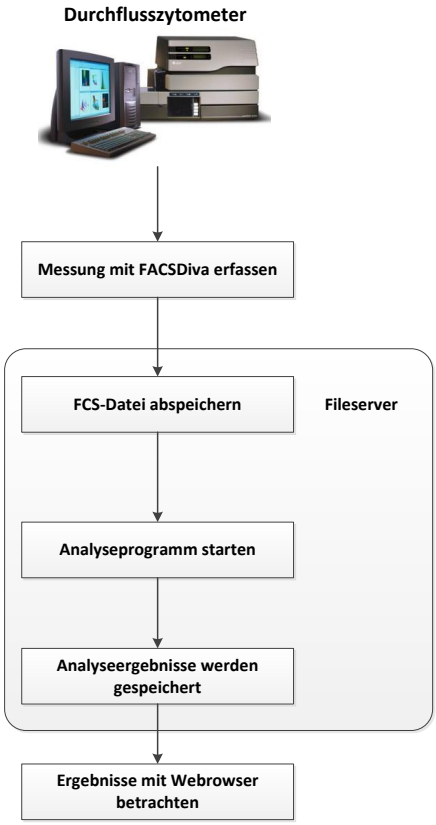

Abbildung 6.5.: Neuer Workflow

Mögliche Verbesserungen

Die Verifikation hat gezeigt, dass das Analyseprogramm nicht in allen Fällen korrekte Werte für die Zellpopulationen liefert. Eine wesentliche Verbesserung wäre ein Verfahren zum Detektieren von Fehlern beim Clustering, die aufgrund der Vielfältigkeit der Daten entstehen. Hier ist denkbar im Fehlerfall den Einsatz eines anderen Cluster-Algorithmus zu initiieren, der dann exaktere Resultate liefert.

Damit Analysen über die Weboberfläche gestartet werden können, muss eine Schnittstelle zwischen Visualisierungsplattform und Analyseprogramm geschaffen werden. Vorteilhaft wäre es, wenn der Benutzer die Möglichkeit hätte die gefundenen Cluster anzupassen. Hierbei würde er Einfluss auf Größe und Position der Cluster nehmen und so die Ergebnisse verbessern.

7. Schluss

Im ersten Abschnitt dieses Kapitels wird die Masterarbeit zusammengefasst, wobei die wichtigsten Merkmale der Arbeit hervorgehoben werden. Der darauffolgende Abschnitt beinhaltet ein Fazit, in dem die Ergebnisse der Arbeit kurz bewertet werden. Der letzte Abschnitt gibt einen Ausblick, welche Möglichkeiten der Weiterentwicklung für die Zukunft bestehen.

7.1. Zusammenfassung

Bei der Durchflusszytometrie werden Blutproben mit einem Laser bestrahlt und die Signale gemessen, wodurch die im Blut vorhanden Zellpopulationen bestimmt werden. Das Verfahren wird für die Multiple Sklerose Forschung eingesetzt, um den aktuellen Krankheitszustand von Patienten und die Wirksamkeit von Medikamenten zu diagnostizieren. Die Zellpopulationen im Blut geben Auskunft über den aktuellen Zustand der Immunabwehr.
Die Messdaten wurden bisher manuell ausgewertet, was fehleranfällig und zeitaufwändig ist. Grundlage der Analyse ist das Identifizieren verschiedener Cluster in den Daten, die unterschiedliche Zellpopulationen im Blut repräsentieren. Die Analysen konnten bisher nur schwer reproduziert werden, da Größe und Position der Cluster von der jeweiligen Person, die die Daten auswertet, individuell festgelegt werden. Um zu prüfen, ob es möglich ist, die Auswertung der Messdaten mit einer Software zu automatisieren, wurde ein Analyseprogramm entwickelt, das die Zellpopulationen für eine genau definierte wissenschaftliche Fragestellung bestimmt. Das Programm basiert auf verschiedenen Cluster-Algorithmen, mit denen die Zellpopulationen in den Daten identifiziert werden.
Die Entwicklung des Analyseprogramms erfolgte mit der Programmiersprache R, da diese eine Reihe von Bibliotheken für den Zugriff auf die Messdaten, sowie eine Vielzahl von Methoden für die Datenanalyse bietet. Hierbei wurde ein möglichst generischer Ansatz gewählt, indem ein Framework entwickelt wurde, das verschiedene Komponenten für die Datenanalyse bereitstellt.
Für die gegebene Fragestellung wurde ein Analyseprogramm auf Basis des Frameworks implementiert, das automatisiert die Zellpopulationen bestimmt und die Ergebnisse abspeichert.

Zusätzlich wurde eine Webanwendung mit dem Django-Framwork entwickelt, mit der die Analyseergebnisse betrachtet und ausgewertet werden können.
Um zu prüfen, ob das Analyseprogramm korrekte Werte liefert, wurden 100 Messdaten manuell ausgewertet und die Ergebnisse mit denen des Analyseprogramms verglichen. Hierdurch konnte gezeigt werden, dass das Analyseprogramm korrekte Werte liefern kann. Jedoch wurde festgestellt, dass die eingesetzen Cluster-Algorithmen nicht immer optimale Ergebnisse liefern und es hierdurch zu Verfälschungen der Resultate kommen kann. Bei der Auswertung einer Messreihe mit 1500 Datensätzen wurde gezeigt, dass es mit dem Analyseprogramm möglich ist, eine große Menge an Eingabedaten zu verarbeiten.

7.2. Fazit

In der Masterarbeit wurde gezeigt, dass es möglich ist, die Messdaten aus der Durchflusszytometrie völlig automatisiert zu analysieren. Allerdings muss das entwickelte Verfahren noch weiter optimiert werden, um für die vielfältigen Messdaten zuverlässig Ergebnisse zu liefern. Der Datenbestand am ZMNH von insgesamt einem Terabyte wurde bisher nur zu einem geringen Bestandteil analysiert und ist größtenteils ungenutzt. Durch die entwickelte Analysesoftware wird es erstmalig möglich, einen Großteil der Messdaten auszuwerten.
Das entwickelte Framework ermöglicht es, mit geringem Programmieraufwand neue Analyseverfahren zu implementieren, die Antworten für unterschiedliche Fragestellungen aus der Multiple Sklerose Forschung liefern können.

7.3. Ausblick

Das entwickelte Analyseprogramm kann verbessert werden, indem Fehler beim Clustering detektiert und behandelt werden. Die Entwicklung weiterer Analyseprogramme ist in Planung. Im Rahmen einer Studie sollen eine Vielzahl von FCS-Dateien analysiert werden, wobei geprüft wird, ob ein bestimmtes Medikament wirkt.
Die Visualisierungsplattform bietet verschiedene Verbesserungspotenziale. Wünschenswert wäre es, wenn der Benutzer in der Weboberfläche die Analysen starten und parametrisieren kann. Eine weitere Verbesserung wäre es, dem Benutzer eine Möglichkeit zu bieten, die Ergebnisse der Analyse zu beeinflussen. Hierbei wäre es denkbar, dass der Benutzer die durch die Cluster-Algorithmen vorgegebenen Cluster anpasst, indem er Größe und Position verändert. Dies hätte den Vorteil, dass die Analysen automatisiert durchgeführt werden können und nachträglich die Möglichkeit besteht, die Ergebnisse zu verbessern.

Damit das Framework einem größeren Benutzerkreis zur Verfügung gestellt werden kann, soll es unter einer OpenSource-Lizenz veröffentlicht werden.

A. Analyse einer FCS-Datei

In diesem Anhang werden alle Plots, die bei der Analyse einer FCS-Datei erzeugt werden gezeigt.

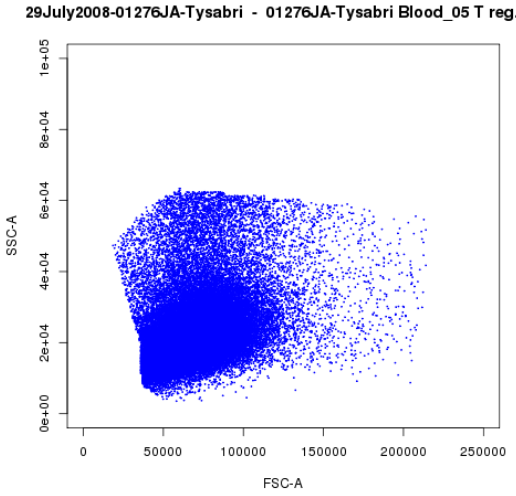

Abbildung A.1.: Plot der Kanäle FSC und SSC

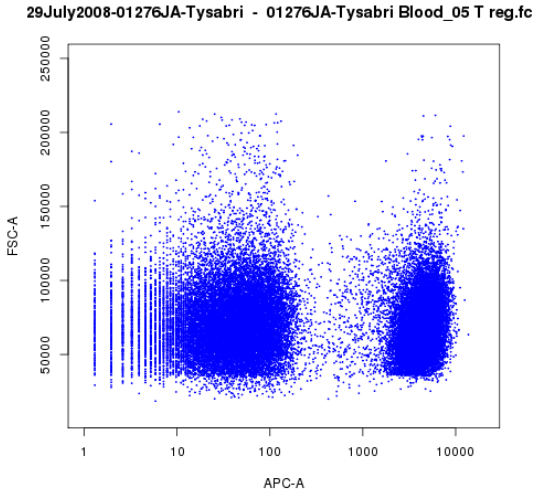

Abbildung A.2.: Plot der Kanäle APC und SSC

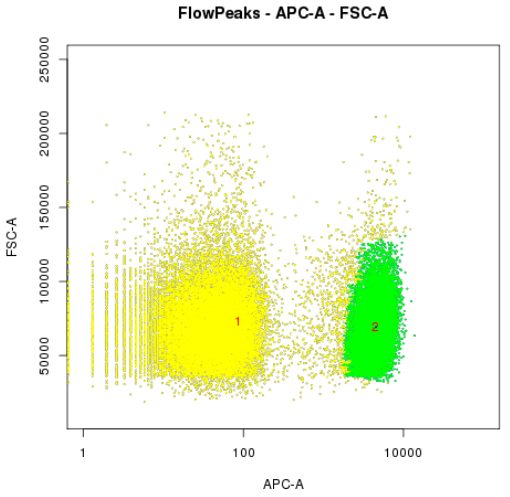

Abbildung A.3.: FlowPeaks Clustering

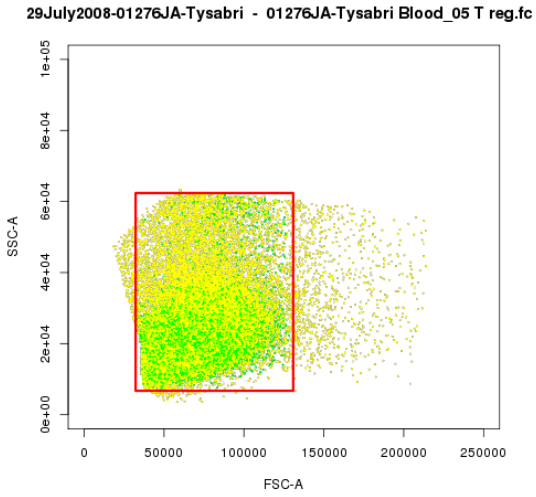

Abbildung A.4.: Lymphozyten selektieren

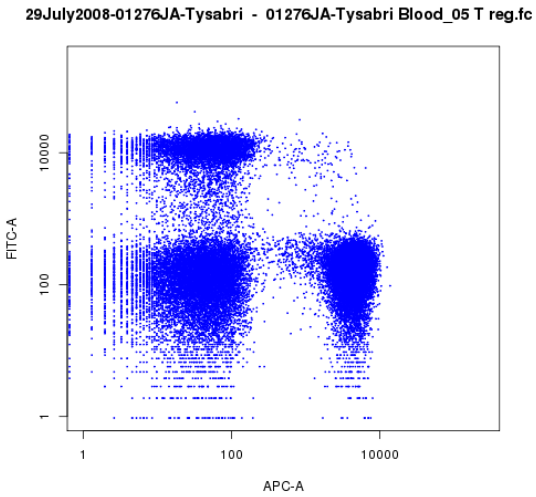

Abbildung A.5.: Plot der Kanäle APC und FITC

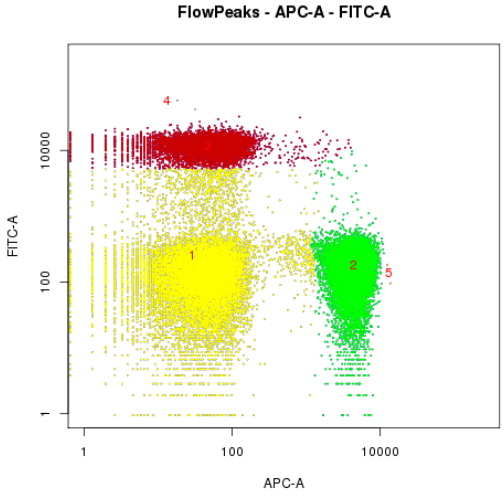

Abbildung A.6.: FlowPeaks Clustering

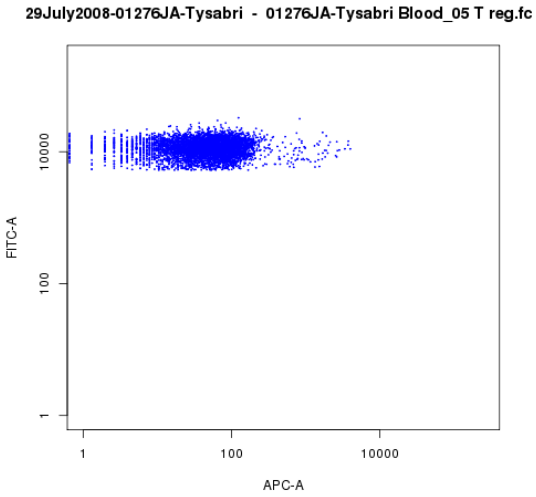

Abbildung A.7.: FITC-Cluster extrahiert

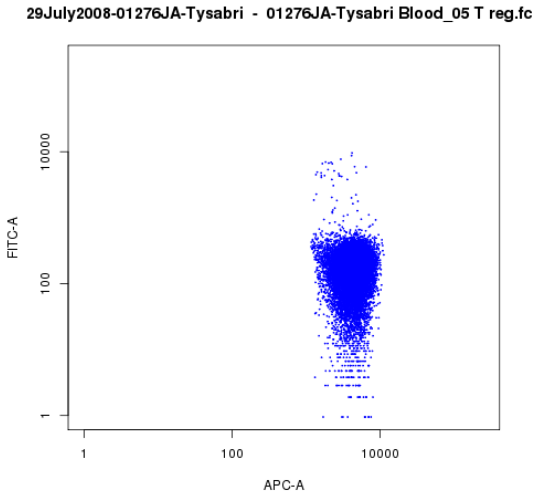

Abbildung A.8.: APC-Cluster extrahiert

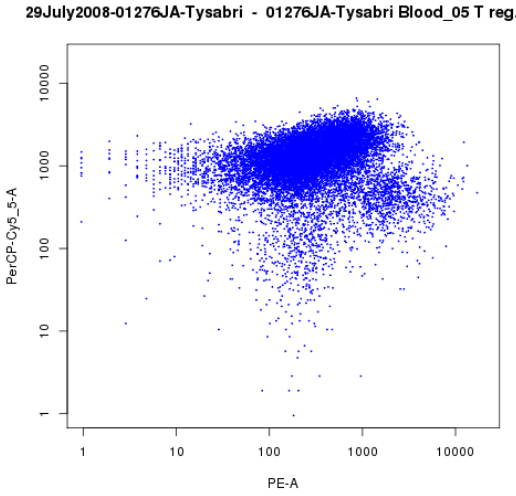

Abbildung A.9.: Plot der Kanäle

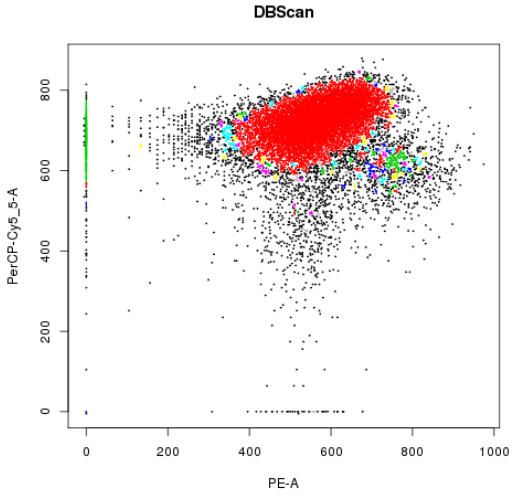

Abbildung A.10.: DBSCAN-Clustering

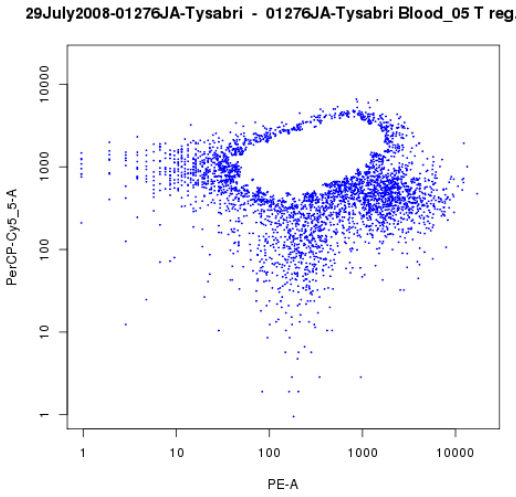

Abbildung A.11.: Cluster wurde entfernt

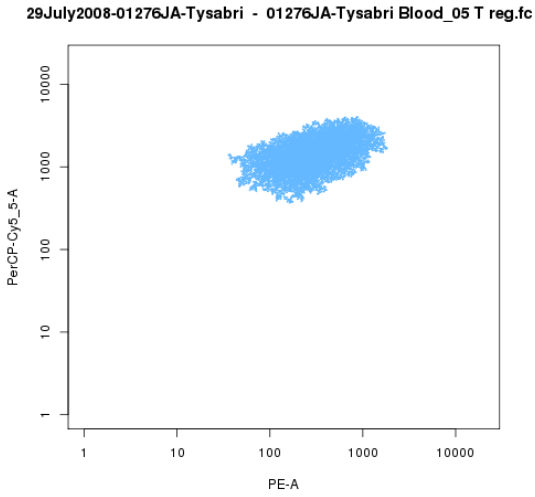

Abbildung A.12.: Cluster extrahiert

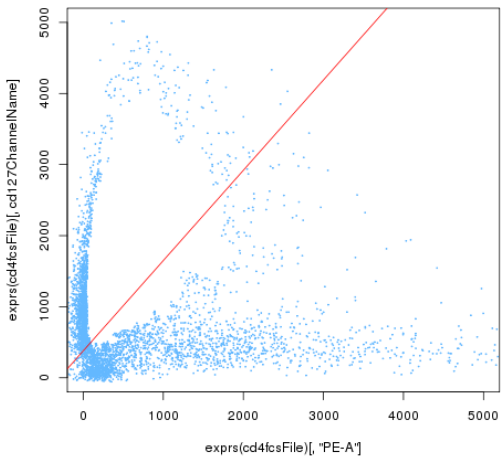

Abbildung A.13.: Regressionsgerade

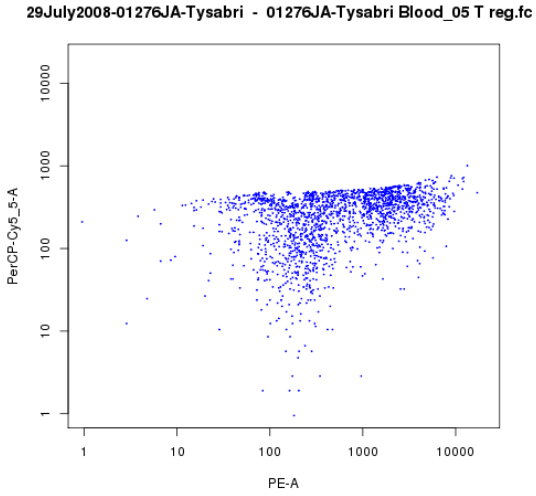

Abbildung A.14.: Extrahierte Datenpunkte

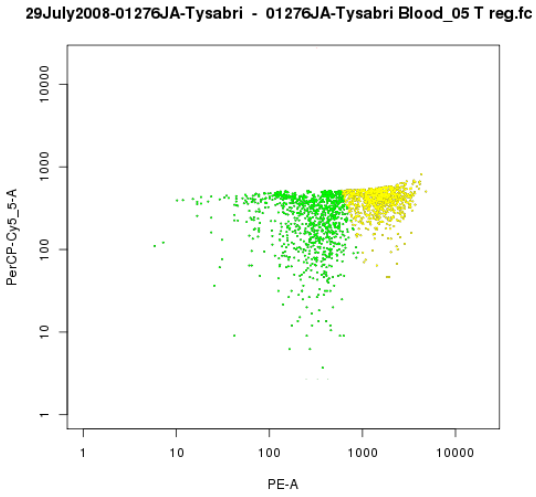

Abbildung A.15.: Ergebnis von K-Means

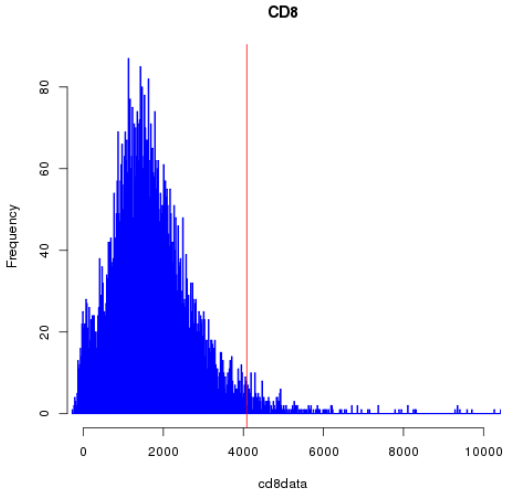

Abbildung A.16.: CD8-Signale

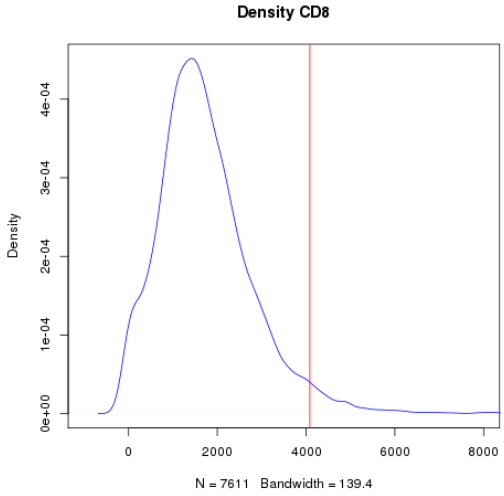

Abbildung A.17.: Dichtefunktion CD8

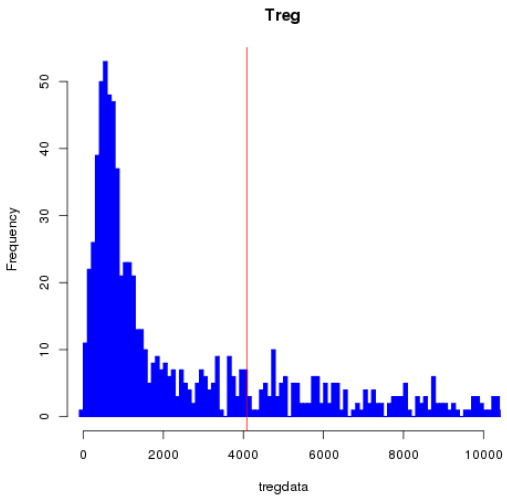

Abbildung A.18.: Treg-Signale

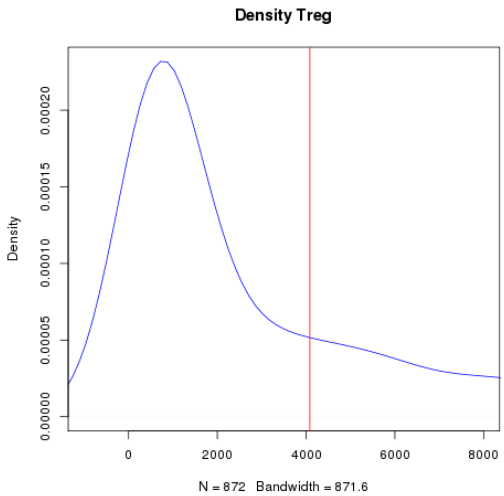

Abbildung A.19.: Dichtefunktion Treg

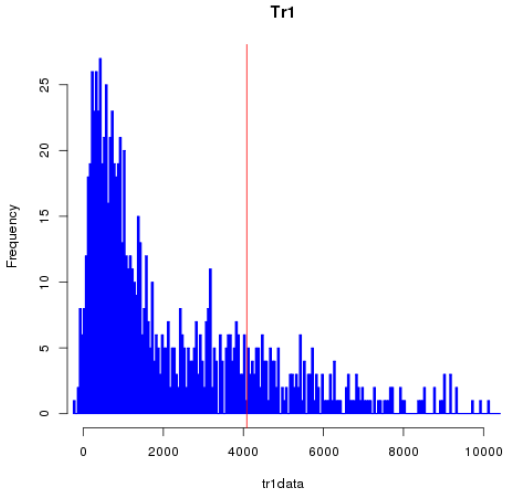

Abbildung A.20.: Tr1-Signale

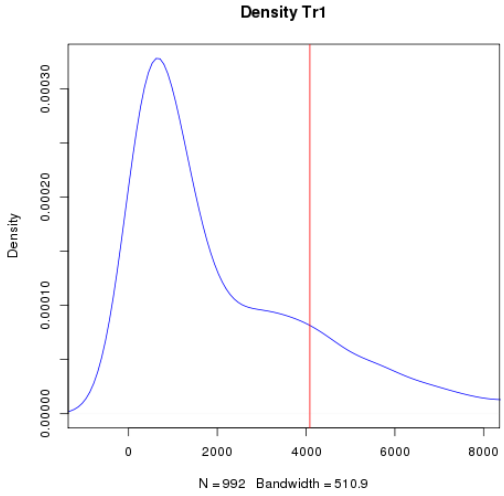

Abbildung A.21.: Dichtefunktion Tr1

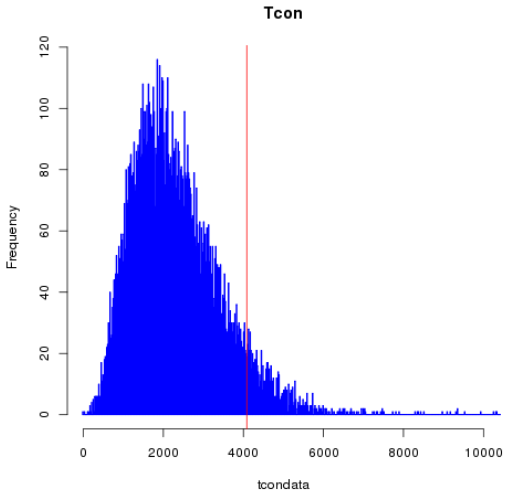

Abbildung A.22.: Tcon-Signale

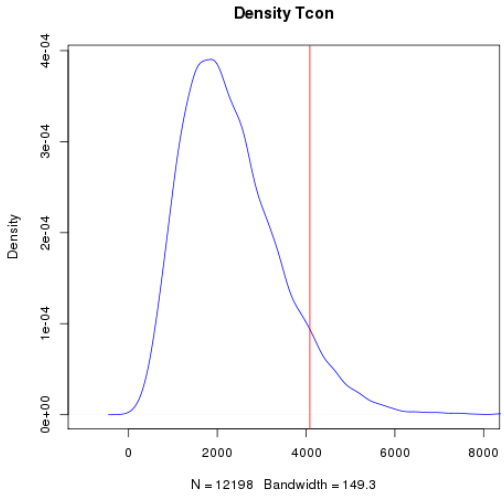

Abbildung A.23.: Dichtefunktion Tcon

B. Screenshots der Visualisierungsplattform

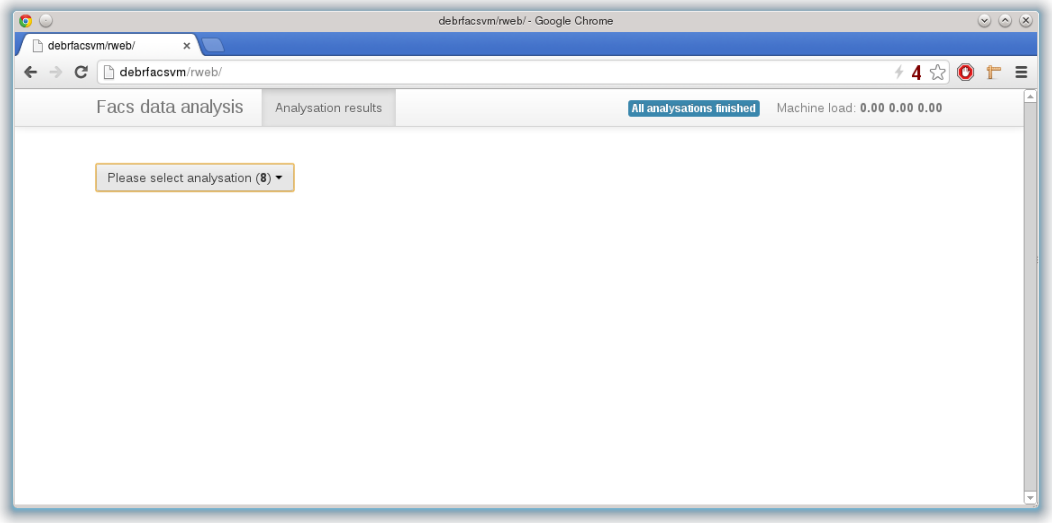

Abbildung B.1.: Startseite der Webanwendung

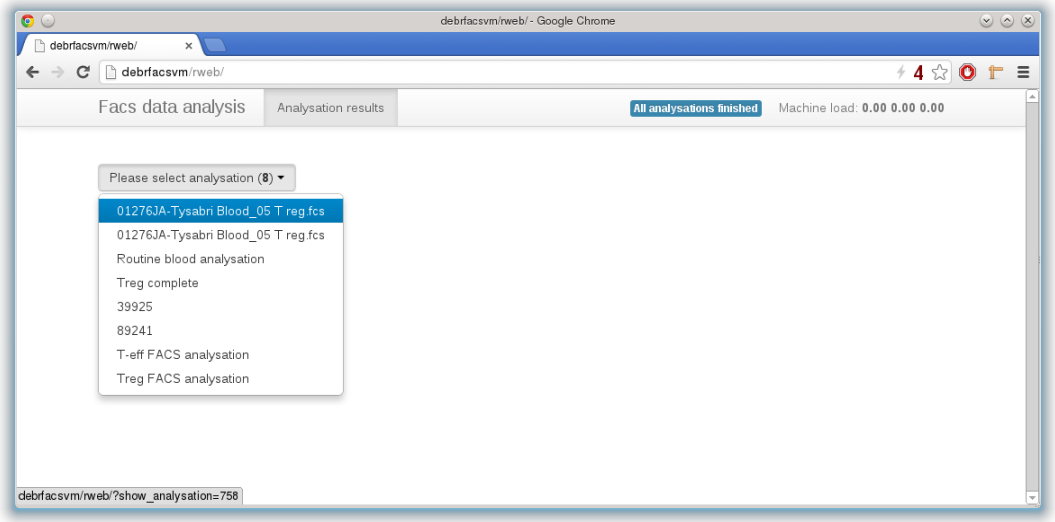

Abbildung B.2.: Auswählen einer Analyse

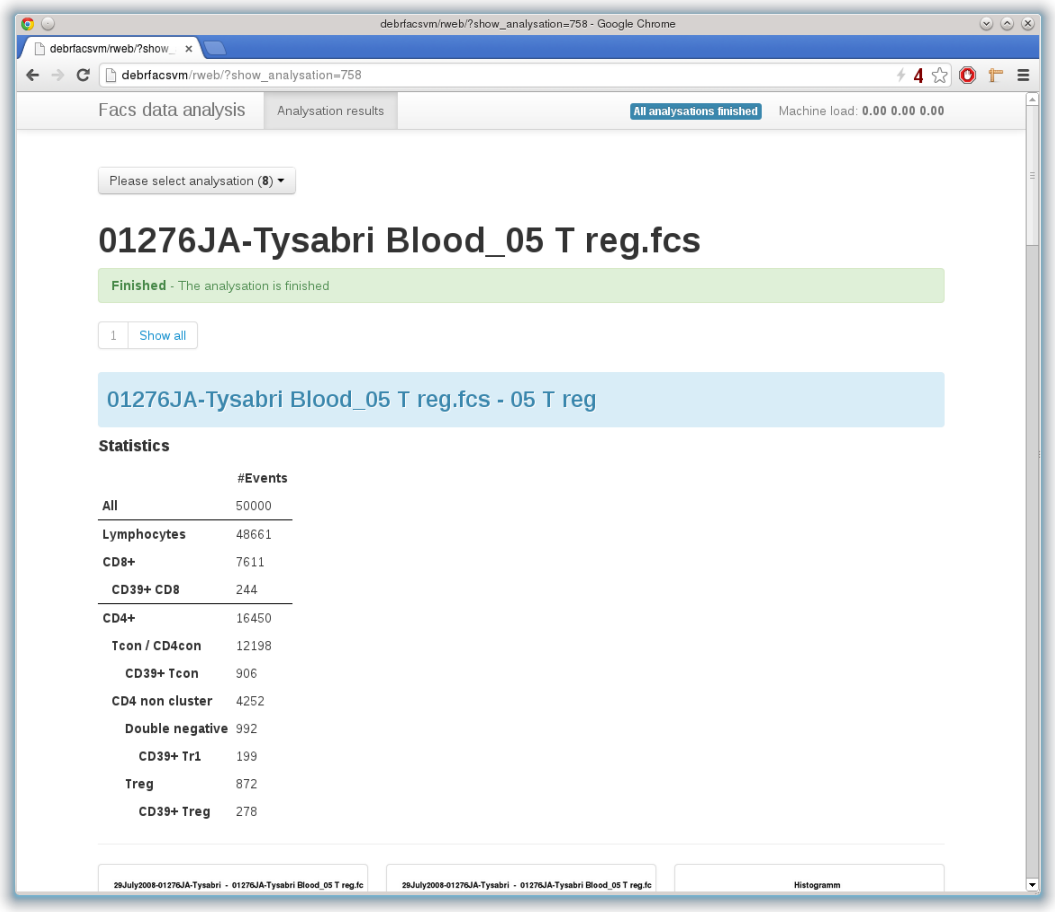

Abbildung B.3.: Häufigkeiten der Zellpopulationen

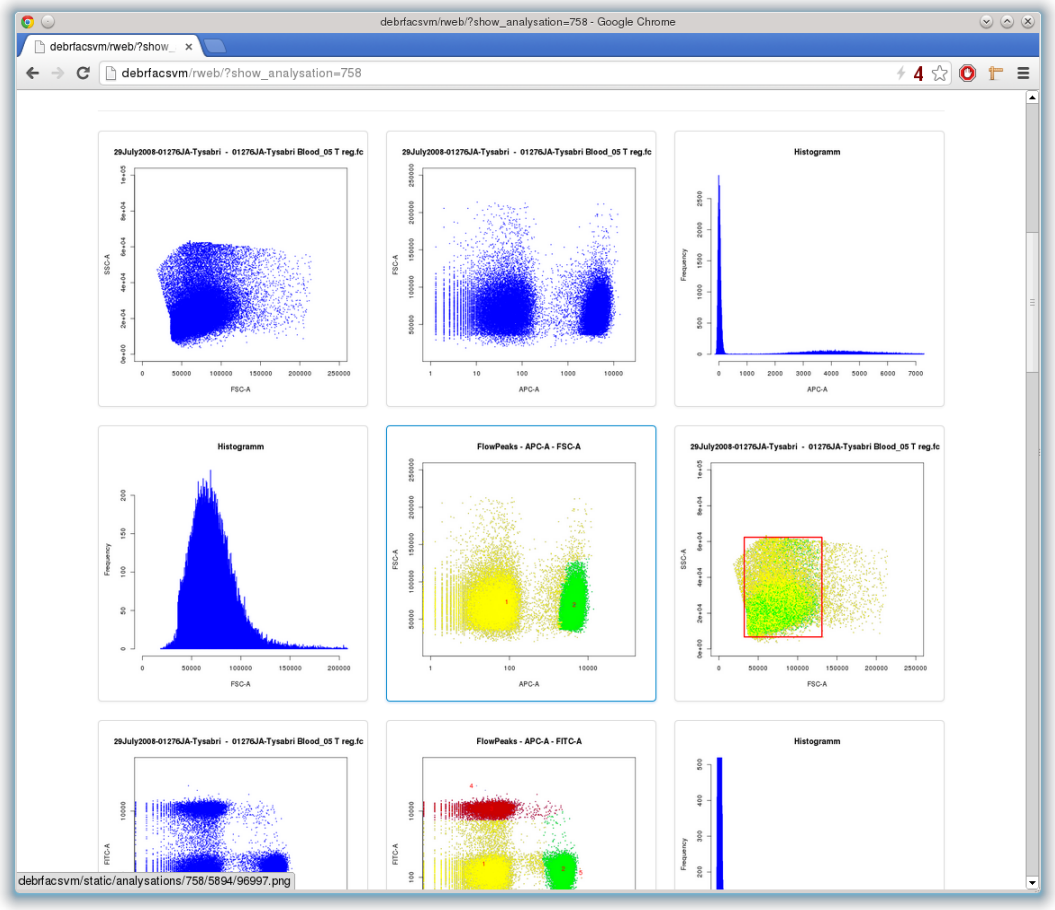

Abbildung B.4.: Plots der einzelnen Analyseschritte

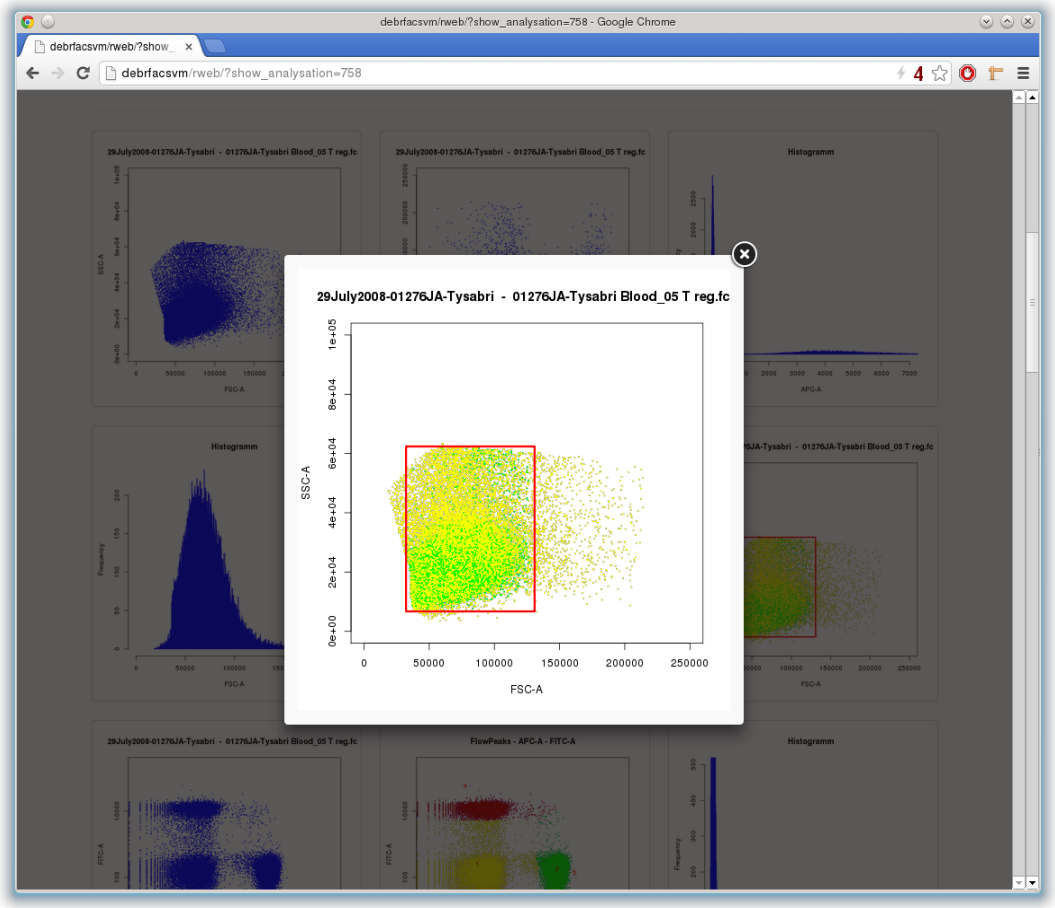

Abbildung B.5.: Betrachten eines einzelnen Plots

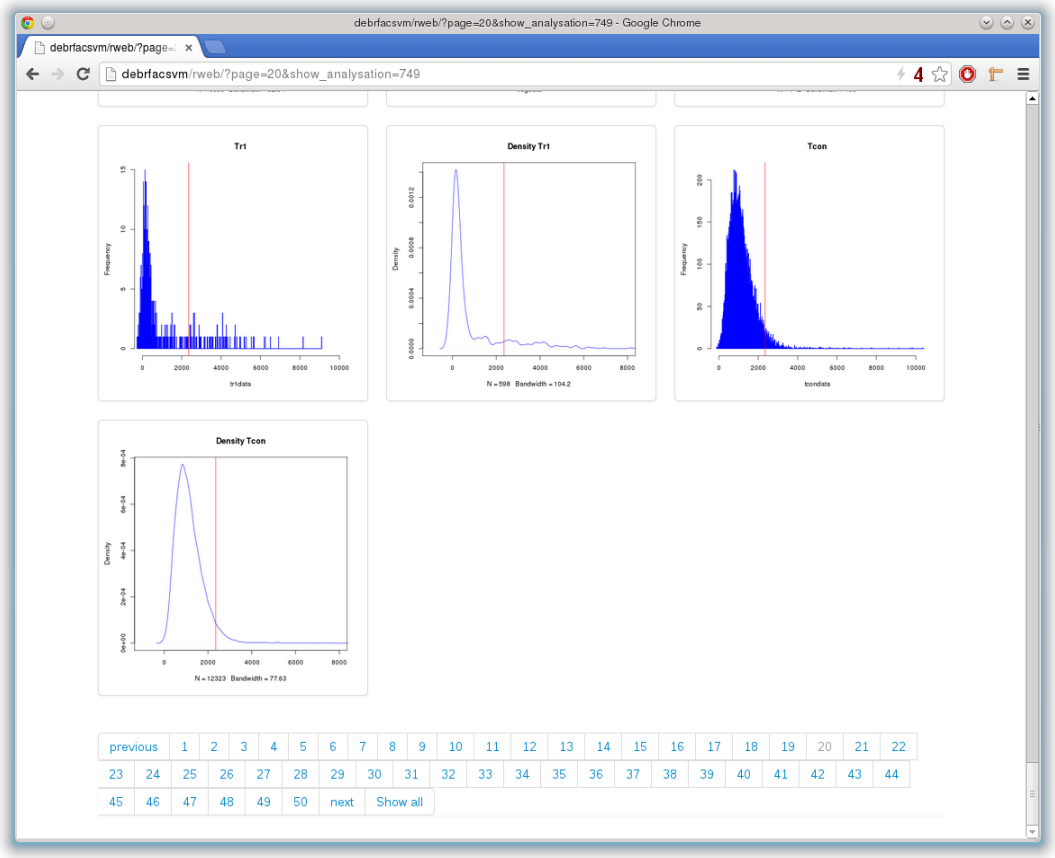

Abbildung B.6.: Navigation bei einer Vielzahl von Analysen

Literaturverzeichnis

[Aghaeepour u. a. 2011] Aghaeepour, Nima ; Nikolic, Radina ; Hoos, Holger H. ; Brinkman, Ryan R.: Rapid cell population identication in ow cytometry dat. In: *Cytometry Part A* 79A (2011), Januar, S. 6–13

[Alchin 2010a] Alchin, Marty: *Pro Django*. Apress, 2010. – ISBN 978-1430210474

[Alchin 2010b] Alchin, Marty: *Pro Python*. Apress, 2010. – ISBN 978-1430227571

[Ankerst u. a. 1999] Ankerst, Mihael ; Breunig, Markus M. ; Kriegel, Hans peter ; Sander, Jörg: OPTICS: Ordering Points To Identify the Clustering Structure, ACM Press, 1999, S. 49–60

[Arthur und Vassilvitskii 2007] Arthur, David ; Vassilvitskii, Sergei: k-means++: the advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA, USA : Society for Industrial and Applied Mathematics, 2007 (SODA '07), S. 1027–1035. – ISBN 978-0-898716-24-5

[Bashashati und Brinkman 2009] Bashashati, Ali ; Brinkman, Ryan R.: A Survey of Flow Cytometry Data Analysis Methods. In: *Adv. Bioinformatics* 2009 (2009)

[Bennett 2008] Bennett, James: *Practical Django Projects*. Apress, 2008. – ISBN 978-1590599969

[Bentley 1975] Bentley, Jon L.: Multidimensional binary search trees used for associative searching. In: *Commun. ACM* 18 (1975), September, Nr. 9, S. 509–517. – ISSN 0001-0782

[Dempster u. a. 1977] Dempster, A. ; Laird, N. ; Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. In: *J. Royal Statistical Society, Series B* 39 (1977), Nr. 1, S. 1–38

[Ester u. a. 1996] Ester, Martin ; Kriegel, Hans peter ; S, Jörg ; Xu, Xiaowei: A density-based algorithm for discovering clusters in large spatial databases with noise, AAAI Press, 1996, S. 226–231

[Fanghänel u. a. 2009] Fanghänel, J. ; Pera, F. ; Anderhuber, F. ; Nitsch, R. ; Waldeyer, A.J.: *Waldeyer - Anatomie des Menschen*. De Gruyter, 2009. – ISBN 9783110221046

[Finak u. a. 2009] Finak, Greg ; Bashashati, Ali ; Brinkman, Ryan R. ; Gottardo, Raphael: Merging Mixture Components for Cell Population Identification in Flow Cytometry. In: *Adv. Bioinformatics* 2009 (2009)

[Fischer 2005] Fischer, G.: *Stochastik einmal anders: Parallel geschrieben mit Beispielen und Fakten, vertieft durch Erläuterungen*. Vieweg+Teubner Verlag, 2005. – ISBN 9783528039677

[Fraley und Raftery 1998] Fraley, C. ; Raftery, A. E.: How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis. In: *The Computer Journal* 41 (1998), S. 578–588

[Fraley und Raftery 2000] Fraley, Chris ; Raftery, Adrian E.: Model-Based Clustering, Discriminant Analysis, and Density Estimation. In: *JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION* 97 (2000), S. 611–631

[Freedman und Diaconis 1981] Freedman, David ; Diaconis, Persi: On the histogram as a density estimator: L2 theory. In: *Probability Theory and Related Fields* 57 (1981), Dezember, Nr. 4, S. 453–476

[Ge und Sealfon 2012] Ge, Y. ; Sealfon, S.C.: flowPeaks: a fast unsupervised clustering for flow cytometry data via K-means and density peak finding. In: *Bioinformatics* 28 (2012), Nr. 15, S. 2052–8

[Gentleman u. a. 2004] Gentleman, Robert ; Carey, Vincent ; Bates, Douglas ; Bolstad, Ben ; Dettling, Marcel ; Dudoit, Sandrine ; Ellis, Byron ; Gautier, Laurent ; Ge, Yongchao ; Gentry, Jeff ; Hornik, Kurt ; Hothorn, Torsten ; Huber, Wolfgang ; Iacus, Stefano ; Irizarry, Rafael ; Leisch, Friedrich ; Li, Cheng ; Maechler, Martin ; Rossini, Anthony ; Sawitzki, Gunther ; Smith, Colin ; Smyth, Gordon ; Tierney, Luke ; Yang, Jean ; Zhang, Jianhua: Bioconductor: open software development for computational biology and bioinformatics. In: *Genome Biology* 5 (2004), Nr. 10, S. R80. – ISSN 1465-6906

[Hahne u. a. 2009] Hahne, Florian ; LeMeur, Nolwenn ; Brinkman, Ryan R. ; Ellis, Byron ; Haaland, Perry ; Sarkar, Deepayan ; Spidlen, Josef ; Strain, Errol ; Gentleman, Robert: flowCore: a Bioconductor package for high throughput flow cytometry. In: *BMC Bioinformatics* 10 (2009), S. 106

[Han u. a. 2011] Han, J. ; Kamber, M. ; Pei, J.: *Data Mining: Concepts and Techniques: Concepts and Techniques*. Elsevier Science, 2011 (The Morgan Kaufmann Series in Data Management Systems). – ISBN 9780123814807

[Hartigan und Wong 1979] Hartigan, J. A. ; Wong, M. A.: Algorithm AS 136: A k-means clustering algorithm. In: *Applied Statistics* 28 (1979), Nr. 1, S. 100–108. – ISSN 00359254

[Hastie u. a. 2008] Hastie, Trevor ; Tibshirani, Robert ; Friedman, Jerome: *The elements of statistical learning: data mining, inference and prediction*. 2. Springer, 2008. – ISBN 978-0-387-84858-7

[Hoffmann 2008] Hoffmann, D.W.: *Software-Qualität*. Springer London, Limited, 2008. – ISBN 9783540763239

[Holovaty 2010] Holovaty, Adrian: *The Definitive Guide to Django: Web Development Done Right*. Apress, 2010. – ISBN 978-1430219361

[Hübl 2013] Hübl, Prof. Dr. W.: *Informationsseite zu verschiedenen Themen aus dem Bereich der Labormedizin*. 2013. – URL http://www.med4you.at/

[Klinke und Brundage 2009] Klinke, David J. ; Brundage, Kathleen M.: Scalable analysis of flow cytometry data using R/Bioconductor. In: *Cytometry Part A* 75A (2009), Nr. 8, S. 699–706. – ISSN 1552-4930

[Lloyd 1982] Lloyd, S.: Least squares quantization in PCM. In: *Information Theory, IEEE Transactions on* 28 (1982), Nr. 2, S. 129–137. – ISSN 0018-9448

[Lo u. a. 2009] Lo, Kenneth ; Hahne, Florian ; Brinkman, Ryan R. ; Gottardo, Raphael: flowClust: a Bioconductor package for automated gating of flow cytometry data. In: *BMC Bioinformatics* 10 (2009), Nr. 1, S. 145

[Lutz 2011] Lutz, Mark: *Programming Python*. O'Reilly Media, 2011. – ISBN 978-0596158101

[MacQueen 1967] MacQueen, J. B.: Some Methods for Classification and Analysis of MultiVariate Observations. In: Cam, L. M. L. (Hrsg.) ; Neyman, J. (Hrsg.): *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, University of California Press, 1967, S. 281–297

[Murphy 1985] Murphy, Robert F.: Automated identification of subpopulations in flow cytometric list mode data using cluster analysis. In: *Cytometry* 6 (1985), Nr. 4, S. 302–309. – ISSN 1097-0320

[Naumann u. a. 2010] Naumann, Ulrike ; Luta, George ; Wand, Matthew P.: The curvHDR method for gating flow cytometry samples. In: *BMC Bioinformatics* 11 (2010), S. 44

[Perfetto u. a. 2004] Perfetto, S. P. ; Chattopadhyay, P. K. ; Roederer, M.: Seventeen-colour flow cytometry: unravelling the immune system. In: *Nat Rev Immunol* 4 (2004), August, Nr. 8, S. 648–655. – ISSN 1474-1733

[Pyne u. a. 2010] Pyne, Saumyadipta ; Hu, Xinli ; Wang, Kui ; Rossin, Elizabeth ; Lin, Tsung I. ; Maier, Lisa ; Baecher-Allan, Clare ; McLachlan, Geoffrey J. ; Tamayo, Pablo ; Hafler, David ; Jager, Philip L. D. ; Mesirov, Jill P.: Automated High-Dimensional Flow Cytometric Data Analysis. In: Berger, Bonnie (Hrsg.): *RECOMB* Bd. 6044, Springer, 2010, S. 577. – ISBN 978-3-642-12682-6

[Reenskaug 1979] Reenskaug, Trygve: Thing-Model-View-Editor – An example from a planningsystem. (1979), May

[Sack u. a. 2006] Sack, U. ; Tárnok, A. ; Rothe, G.: *Zelluläe Diagnostik: Grundlagen, Methoden und klinische Anwendungen der Durchflusszytometrie.* Karger Verlag, 2006. – ISBN 9783805579285

[Seamer 2001] Seamer, L.: Data file standard for flow cytometry, FCS 3.0. In: *Current protocols in cytometry / editorial board, J. Paul Robinson, managing editor ... [et al.]* Chapter 10 (2001), Mai. – ISSN 1934-9300

[Wollschläger 2012] Wollschläger, Daniel: *Grundlagen der Datenanalyse mit R.* Springer Berlin Heidelberg, 2012 (Statistik und ihre Anwendungen). – ISBN 978-3-642-25799-5

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 24.09.2013 Jan-Christoph Meier