Institut für Informatik
Algorithmische Bioinformatik

Universitätsstr. 1 D–40225 Düsseldorf

Bestimmung des HLA-Genotyps auf Basis von genomweiter Sequenzierung und k-mer-Häufigkeiten

Yulian Martynovsky

Abgabe:	25.10.2018
1. Prüfer:	Prof. Dr. G. Klau
2. Prüfer:	Dr. A. Dilthey
Betreuer:	S. Schrinner

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Düsseldorf, 25. Oktober 2018

Yulian Martynovsky

Zusammenfassung

Das humane Leukozyten-Antigen-System (Abk. *HLA*) wird mit vielen Autoimmun und Infektionskrankheiten in Verbindung gebracht. Deswegen ist das Bestimmen der Allele der HLA-Region eines Individuums für uns interessant. Der hohe Polymorphismus der HLA-Region und die Sequenzähnlichkeit der Gene in dieser erschweren die Bestimmung dieser Allele. Wir beschreiben das Problem der Allelbestimmung als ganzzahliges lineares Programm auf Basis von k-mer-Häufigkeiten.Wir verwenden die IMGT/HLA-Datenbank mit bekannten Allelen und den dazugehören Sequenzen. Mit Hilfe dieser Daten bestimmen wir für ein Individuum die vorhandenen Allele auf G-group Ebene. Wir beobachten, dass dieser Ansatz schlechter abschneidet als andere Verfahren und dass unser Modell noch erweiter werden muss, um eine höhere Genauigkeit zu erzielen. Wir evaluieren unseren Algorithmus auf sieben Genen, welche zu den Klassen I und II gehören (HLA-A, B, C, DPB1, DQA1, DQB1, DRB1) auf 7 Datensätzen. Auf höchster Auflösung inferieren wir 37 von 56 Allelen korrekt(66%) für unsere ausgewerteten 1000g Datensätze 2.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Problemstellung . . . 1
- 1.2 Zielsetzung . . . 1

2 Definitionen **2**
- 2.1 Lineare Programmierung . . . 2
  - 2.1.1 Aufbau . . . 2
  - 2.1.2 Beispiel . . . 2
- 2.2 Biologie . . . 3
  - 2.2.1 Allel . . . 3
  - 2.2.2 Allelnotation . . . 3
  - 2.2.3 Genklassen . . . 4
  - 2.2.4 k-mere . . . 4
  - 2.2.5 Coverage . . . 4
  - 2.2.6 G-groups . . . 5

3 Methoden **6**
- 3.1 Idee . . . 6
- 3.2 Formalisierung . . . 6
- 3.3 Modellierung . . . 7

4 Evaluierung **10**
- 4.1 Implementierung . . . 10
- 4.2 Daten . . . 10
  - 4.2.1 Beschreibung der Daten . . . 10
  - 4.2.2 Bestimmung der Coverage . . . 11
  - 4.2.3 Evaluierungsprozess . . . 13
- 4.3 Ergebnisse . . . 14
  - 4.3.1 Auswirkung der Coverage . . . 15

5 Diskussion **17**
- 5.1 Verwandte Arbeiten . . . 17
- 5.2 Zukünftige Arbeiten . . . 17
- 5.3 Konklusion . . . 18

6 Danksagungen **19**

7 Literaturverzeichnis **19**

A Code **22**

1 Einleitung

Das humane Leukozyten-Antigen-System ist eine Region im menschlichen Genom, welche viele Proteine kodiert, die für das Immunsystem relevant sind. Hauptsächlich werden mit der HLA-Region Autoimmun-[1] und Infektionskrankheiten [2] in Verbindung gebracht. Der HLA-Typ spielt außerdem eine wesentliche Rolle für Immuntherapien gegen Krebs [4] und die Erfolgsrate von Gewebe- und Organtransplantationen [3]. Ein Beispiel ist Spondylitis akylosans, eine Entzündungskrankheit, die mit dem Vorhandenseins des Allels HLA-B27 in Verbindung gebracht werden kann [10].

1.1 Problemstellung

Aufgrund der hohen Relevanz der HLA-Region versucht man mittels Genotypisierung zu bestimmen, welche konkreten Informationen sich in der HLA-Region eines Individuums befindet. Die HLA-Region beinhaltet mehrere Gene. Wir wollen herausfinden, welche Allele sich in den Loci dieser Gene befinden. Die Gene der HLA-Region haben untereinander eine hohe Sequenzähnlichkeit. Das bedeutet, dass Allele von unterschiedlichen Genen ähnliche Nukleotidsequenzen untereinander haben. Des weiteren ist die HLA-Region sehr polymorph innerhalb von Populationen. Polymorph bedeutet, dass innerhalb einer Population viele verschiedene Allele eines Gens vorhanden sind [13]. Durch diese Eigenschaften ist das Bestimmen des HLA-Typs ein schwieriges Problem, welches in unterschiedlichen Arbeiten untersucht wird. In den üblichen Verfahren werden die Gene getrennt voneinander betrachtet.

1.2 Zielsetzung

In dieser Arbeit untersuchen wir einen Ansatz, bei dem der HLA-Genotyp mittels k-mer-Häufigkeiten global optimal bestimmt wird und bei dem einzelne Gene nicht isoliert betrachtet werden. Dafür entwickeln wir einen Algorithmus der auf Basis der IMGT/HLA-Datenbank, einer Datenbank von bekannten Allelen und den zugehörigen k-mer-Häufigkeiten, den Genotyp inferiert. Die k-mer-Häufigkeiten sollen dabei nur aus den Reads stammen, welche zu den Peptidbindestellen-kodierenden Regionen der Gene gehören. Anschließend evaluieren wir anhand der Ergebnisse, wie gut sich der HLA-Genotyp mit dem genannten Algorithmus bestimmen lässt und vergleichen diesen mit anderen Ansätzen.

2 Definitionen

In diesem Kapitel werden die mathematischen und biologischen Grundlagen erläutert, welche zum Verständnis dieser Arbeit benötigt werden. Wir führen lineare Programme ein, da wir diese verwenden werden, um unser Problem zu modellieren.

2.1 Lineare Programmierung

Lineare Programmierung, auch lineare Optimierung genannt, beschäftigt sich mit der Optimierung linearer Zielfunktionen unter Beachtung von Nebenbedingungen. Die Optimierung bezieht sich dabei auf die Maximierung oder Minimierung der Zielfunktion. Die Zielfunktion sowie die Nebenbedingungen sind dabei Linearkombinationen von reellen Variablen. Ein Spezialfall der linearen Programmierung ist die ganzzahlige lineare Programmierung, in der die Variablen nur ganzzahlige Werte annehmen dürfen. Wir definieren lineare Programme nach Markus Bauer[5].

2.1.1 Aufbau

Lineare Programme können in einer Standardform angegeben werden.

Seien $A \in \mathbb{R}^{m \times n}$ eine Matrix und $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ Vektoren. Ein lineares Programm besteht aus einer Zielfunktion und Bedingungen. Die Notation der Standardform ist dabei wie folgt:

$$\begin{array}{ll} \min & c^T x \\ \text{unter} & Ax \leq b \end{array}$$

Die Zielfunktion ist $c^T x$, der Teil, der minimiert wird. Für die Definition nehmen wir an, dass wir die Zielfunktion minimieren wollen. Ein Vektor $\hat{x} \in \mathbb{R}^n$ heißt zulässige Lösung, falls $A\hat{x} \leq b$ gilt. Wenn dieser unter allen zulässigen Lösungen die Zielfunktion minimiert, heißt $\hat{x}$ optimale Lösung. Ein lineares Programm, für welches kein zulässiges $\hat{x}$ mit $\hat{x} \in \mathbb{R}^n$ existiert, heißt nicht zulässig. Falls ein lineares Programm zulässig ist, aber keine optimale Lösung existiert, heißt das lineare Programm unbeschränkt. Es können alle linearen Funktionen sowie alle Gleichheits- und Ungleichheitsbedingungen in diese Standardform transformiert werden.[6]

2.1.2 Beispiel

Wir demonstrieren die Modellierung von Problemen als ganzzahliges lineares Programm anhand eines Beispiels. Wir verwenden hierzu das Knapsack Problem. Gegeben sind eine endliche Menge von Objekten $O = \{o_1, ..., o_n\}$, eine Gewichtsfunktion w, eine Nutzenfunktion v und eine Gewichtsschranke G. Wir suchen eine Teilmenge $S \subseteq O$ für die gilt: $\sum_{s \in S} w(s) \leq G$ und $\sum_{s \in S} v(s)$ maximiert.

Um dieses Problem als ganzzahliges lineares Programm dazustellen benötigen wir Indikatorvariablen $x_1, ..., x_n \in \{0, 1\}$. x_i soll 1 sein, wenn $o_i \in S$ gilt. Demnach sieht unser lineares Programm wie folgt aus:

$$\begin{array}{ll} \max & \sum_{i=1}^{n} x_i \cdot v(o_i) \\ \text{unter} & \sum_{i=1}^{n} x_i \cdot w(o_i) \leq G \end{array}$$

Eine optimale Lösung $x*_1, ..., x*_n$ des linearen Programms führt dann zu der optimalen Lösung des Knapsack Problems $S = \{o_i \in O | x_i = 1\}$

2.2 Biologie

Im Folgenden werden die notwendigen biologischen Grundlagen, die für das Verständnis dieser Arbeit benötigt werden, vermitteln. Wir werden diese nur soweit erklären, wie es zum Verständnis der Problemstellung und des Lösungsansatzes dieser Arbeit notwendig ist.

2.2.1 Allel

Ein Allel ist eine spezifische Ausprägung eines Gens, die in einem Individuum vorhanden ist.

Jeder Mensch besitzt Erbinformationen in Form von DNA. Diese ist als Sequenz von DNA-Basen kodiert. Sie bestimmen die Proteinbildung und dadurch die Merkmale des Individuums. Die DNA bzw. das Genom sieht bei jedem Individuum etwas anders aus. Ein Gen ist nun ein Ausschnitt des Genoms, das für eine bestimmte Funktion zuständig ist. Gene haben kodierende Abschnitte, die man Exons nennt, und nicht kodierende Abschnitte, die man Introns nennt.

2.2.2 Allelnotation

In Abbildung 1 wird die für die HLA-Region verwendete Nomenklatur erklärt.

Diese Form der Notation wird verwendet, um unterschiedliche Allele unterscheiden zu können. Der Präfix HLA symbolisiert, dass es sich um ein Gen der HLA-Region handelt. Darauf folgt durch einen Bindestrich getrennt das Gen und durch ein * getrennt die Kodierung des Allels. Diese Kodierung ist hierarchisch aufgebaut. Alle von uns betrachteten Gene beginnen mit HLA, deswegen werden wir diesen Teil im Folgenden weglassen. Die mit einem Doppelpunkt getrennten Zahlen bezeichnen wir als Felder. Anzumerken ist, dass das dritte und vierte Feld nicht existieren müssen. Sind zwei Allele im ersten Feld gleich, so sind sie serologisch gleich, diese werden auch Allelgruppen genannt. Mit dem zweiten Feld wird ein Protein spezifiziert. Das dritte Feld steht für synonyme Substitutionen in der kodierenden Region. Das vierte Feld steht für Unterschiede in nicht-kodierenden Regionen. Bei der Evaluierung werden wir unterschiedliche Auflösungen betrachten. Auflösung bezieht sich dabei auf die Anzahl der Felder die wir betrachten. Wenn wir also das Allel HLA-A*01:02:03:04 betrachten ist die Darstellung in 2-Feld Auflösung HLA-A*01:02. Es wird immer nur von rechts abgeschnitten, da die rechtsstehenden Felder die Information der vorherigen spezifizieren.

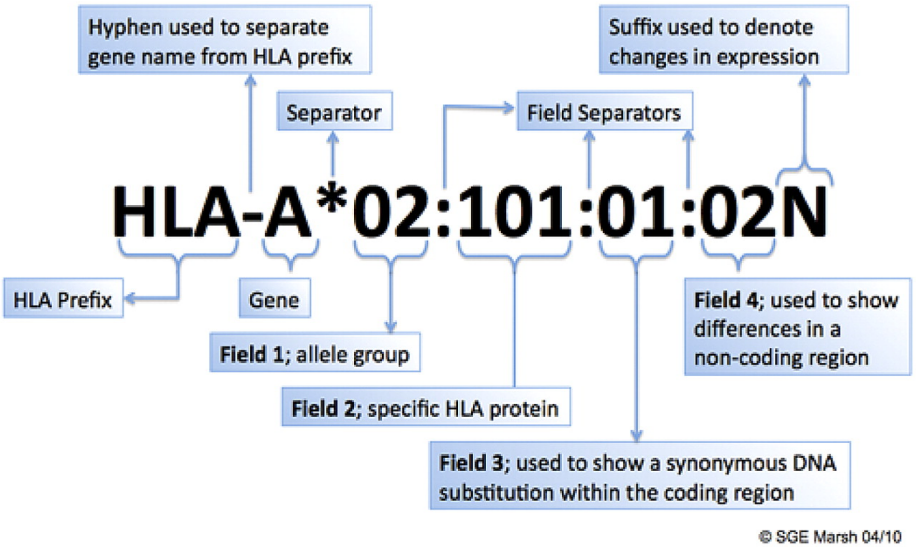

Abbildung 1: Nomenklatur zur Benennung von HLA-Genen [7]

2.2.3 Genklassen

Wir betrachten HLA-Gene aus zwei Klassen, diese sind Klasse I und Klasse II. Gene, die zu Klasse I gehören, kodieren Proteine, welche auf den Zelloberflächen von fast allen Zellen vorhanden sind. Klasse II Gene kodieren Proteine, die fast ausschließlich auf Zelloberflächen von Zellen des Immunsystem vorkommen [8].

2.2.4 k-mere

Ein k-mer ist ein Substring eines Referenzstrings der Länge k. Durch Vergleich der k-mer-Häufigkeiten zweier Strings kann man Information über die Ähnlichkeit der Strings gewinnen. Daher werden wir die Häufigkeit von k-meren innerhalb der Gensequenz eines Individuums betrachten, um mit Hilfe dieser ein lineares Programm aufzustellen, welches auf den HLA-Typen schließen lässt. Für diese Arbeit verwenden wir eine k-mer Länge 31.

2.2.5 Coverage

Coverage bezeichnet die Häufigkeit, mit der ein bestimmter Abschnitt in einer Gesamtsequenz sequenziert wurde. Als haploide Coverage bezeichnen wir die Häufigkeit, mit der ein bestimmter Abschnitt auf genau einem Chromosom sequenziert wurde. Die haploide Coverage ist in der Hinsicht interessant, dass das menschliche Genom diploid ist, also jedes Chromosom zweimal vorkommt ist. Bei dem Sequenzieren eines Genoms ist jedoch nicht klar, von welchem der beiden Chromosomen ein Read stammt.

2.2.6 G-groups

G-groups sind Gruppen von Allelen, welche identische Nukleotidsequenzen für die Peptidbindestellen haben[8]. Bei Klasse I Genen und bei DPB1 sind diese auf Exon 2 und 3 kodiert. Bei den restlichen Klasse II Genen sind sie auf Exon 2 zu finden.[8] Die Peptidbindestellen sind relevant für die Erkennung infizierter Zellen [19]. Unser Algorithmus liefert als Lösung kein spezifisches Allel sondern lediglich die G-group des korrekten Allels, siehe Kapitel 4.

3 Methoden

In diesem Kapitel werden wir ein lineares Programm modellieren, welches auf Grundlage von k-mer-Häufigkeiten Rückschlüsse über den vorliegenden Datensatz zieht. Wir werden mit der Idee beginnen und die Zusammenhänge grob erklären. Dann werden wir die diese formalisieren und unser Modell mit diesen in Form eines ganzzahligen linearen Programms aufbauen.

3.1 Idee

Wir werden nun ein ganzzahliges lineares Programm(im folgenden nur noch lineares Programm) konstruieren, das mit Hilfe von k-mer-Häufigkeiten versucht, die vorliegenden Daten zu erklären. Wir verwenden dazu bestehende Daten über die Vorkommen von k-meren in den jeweiligen Allelen.

Unser lineares Programm benötigt Bedingungen, welche sicherstellen, dass unsere Auswahl von Allelen gültig ist. Grundlage für diese Bedingungen ist ein Datensatz bestehend aus k-mer-Häufigkeiten. Mit Hilfe dieser Häufigkeiten wird eine Auswahl von Allelen berechnet, welche diesen Datensatz mit geringstmöglichem Fehler erklärt.

Dies funktioniert so, dass wir für eine Auswahl von Allelen für jedes Allel bestimmen können, welches k-mer wie oft in diesem Allel zu finden ist. Dies können wir für alle Gene tun, die wir betrachten, und mit den k-mer-Häufigkeiten des Datensatzes, welchen wir behandeln, vergleichen. Damit diese Allelauswahl nun optimal wird, werden wir mit der Zielfunktion die Abweichung dieser beiden k-mer-Häufigkeiten minimieren.

3.2 Formalisierung

Nun formalisieren wir die in 3.1 beschriebene Idee. Die Daten, auf welchen das lineare Programm basiert, verwenden wir in Form einer $m \times n$ Matrix K, in welcher jede Zeile für ein k-mer und jede Spalte für ein Allel steht. Ein Eintrag $K_{i,j}$ ist die Anzahl der Vorkommen des k-meres i in dem Allel j. Unsere Allelauswahl modellieren wir als einen Vektor x mit n Einträgen. Die Einträge von x entsprechen der Auswahl der Allele. Für ein Allel kann dabei die Auswahl 0, 1 oder 2 sein, da wir für beide Chromosome ein Allel auswählen und für jedes Allel gilt, dass es entweder gar nicht, einmal oder auf beiden Chromosomen das gleiche Allel vorhanden ist. Eine 0 steht für "nicht ausgewählt" und Einträge $\neq 0$ stehen für eine Auswahl mit entsprechender Häufigkeit. Die k-mer-Häufigkeiten aus dem zu untersuchenden Datensatz modellieren wir als Vektor y mit m Einträgen, wobei der Eintrag y_i die Anzahl der Vorkommen des i-ten k-mers ist. Die Abweichung e ist ebenfalls ein Vektor mit m Einträgen. Die haploide Coverage c wird in Form eines Skalars dargestellt.

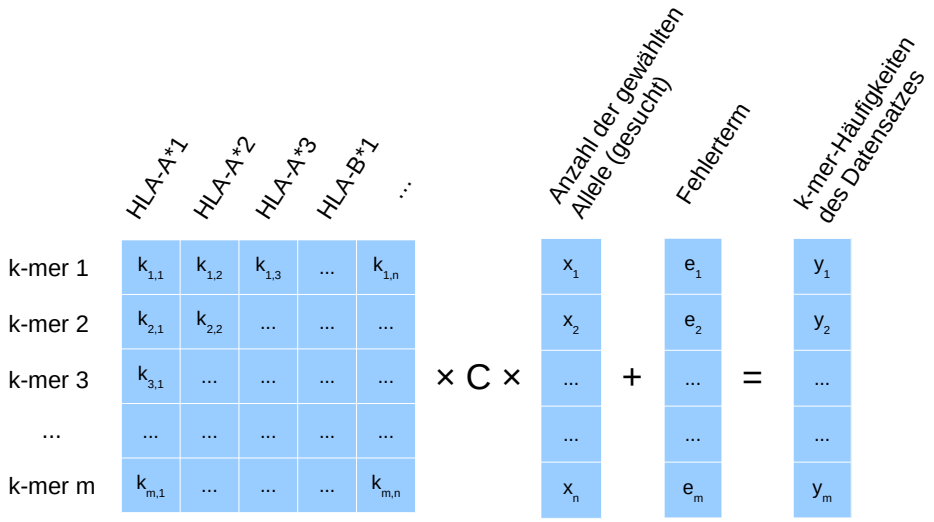

Abbildung 2: Die Einträge der Matrix K sind in dieser Abbildung mit $k_{i,j}$ bezeichnet.

3.3 Modellierung

Nachdem wir die Variablen formalisiert haben, wollen wir ein Modell in Form eines ganzzahligen linearen Programms erstellen. Dafür haben wir eine Gleichung, welche unser Problem beschreibt, basierend auf einer E-mail (Alexander Dilthey, 2018, persönliche Kommunikation). Im Folgenden wird die Gleichung erklärt. Diese wird durch Abbildung 2 veranschaulicht.

Die rechte Seite der Grafik stellt den Datensatz dar. Die Gleichheit entsteht durch die Multiplikation von K mit x und dem Ausgleichen der Unterschiede durch den Fehlerterm e. Der Wert $(Kx)_i$ ist die Anzahl des i-ten k-meres für ein festes x. Der Datensatz wurde jedoch mit einer bestimmten haploiden Coverage sequenziert, deswegen muss die linke Seite entsprechend mit c skaliert werden. Da die Daten Fehler enthalten, welche durch das Sequenzieren entstehen, und die haploide Coverage nicht überall c entspricht, müssen wir für die Gleichheit den Fehlerterm e hinzufügen. Diese Gleichheit werden wir im Folgenden Schritt als Grundlage für unser lineares Programm verwenden.

Da wir den Fehler minimieren wollen, wählen wir eine von e abhängige Zielfunktion. Die Einträge von e können auch negative Werte enthalten, deswegen wählen wir als Zielfunktion:

$$\min \quad \sum_{i=1}^{m} |e_i|$$

Die Betragsfunktion ist zwar nicht linear, jedoch ist es möglich durch Hinzufügen weiterer Nebenbedingungen eine äquivalente Zielfunktion zu erzeugen [6]. Nun müssen wir die Bedingung für e modellieren. Dies erfolgt über die Gleichung aus Abbildung 2. Somit sieht unser Modell nun wie folgt aus:

$$\begin{array}{ll} \min & \sum_{i=1}^{m} |e_i| \\ \text{unter} & cKx + e = y \end{array}$$

Da wir ein diploides Genom betrachten, müssen wir Bedingungen einfügen, welche beschreiben, dass es für jedes Gen genau zwei Allele geben muss. Das heißt, dass für jedes Gen die Summe der Einträge in x, welche zu jenem Gen gehören, insgesamt zwei ergeben muss und die Einträge positiv und ganzzahlig sein müssen. Deswegen gilt $x \in \{0, 1, 2\}^n$. Nun erstellen wir Bedingungen, die die Summe der Einträge beschreiben. Dazu brauchen wir eine Partition P von $\{x_1, ..., x_j\}$, wobei x_i mit $1 \leq i \leq j$ Einträge von x sind, welche nicht zu Allelen der Gene DRB3, DRB4 und DRB5 gehören. Diese werden in einem weiteren Schritt gesondert behandelt. Sei nun

$$P = \{\{x_1, ..., x_{i_1}\}, \{x_{i_1+1}, ..., x_{i_2}\}, ..., \{x_{i_{l-1}+1}, ..., x_{i_l}\}\}.$$

Die einzelnen Mengen in P beinhalten jeweils die zu einem Gen gehörenden Einträge. Damit können wir unser Modell nun erweitern, um die Diploidie zu modellieren:

$$\begin{array}{ll} \min & \sum_{i=1}^{m} |e_i| \\ \text{unter} & cKx + e = y \\ & \forall p \in P : \sum_{x_i \in p} x_i = 2 \end{array}$$

DRB3, DRB4 und DRB5 müssen getrennt behandelt werden, da diese abhängig von der Belegung von den zu DRB1 gehörenden Einträgen sind [9]. Diese Datenbank ist jedoch nicht aktuell und beinhaltet nicht alle Allele, welche wir betrachten. Deswegen verwenden wir folgende Abbildung, die auf einer E-mail (Alexander Dilthey, 2018, persönliche Kommunikation, 11 Mai) basiert:

DRB1	**DRB345**
01, 08, 10	Keine DRB3/4/5 Assoziation
03, 11, 12, 13, 14	DRB3
04, 07, 09	DRB4
15, 16	DRB5

Tabelle 1: Die Einträge in der Spalte DRB1 beziehen sich auf das erste Feld der HLA-Notation. Der Eintrag 01 zum Beispiel steht für DRB*01. Die Spalte DRB345 gibt an mit welchem Gen die DRB1 Allele assoziert werden. Hier gibt es Ausnahmen, diese werden in unserem Modell jedoch nicht behandelt

Entsprechend dieser Assoziationen erzeugen wir eine Partition D von $\{x_c, ..., x_d\}$ mit $c \leq i \leq d$ wobei die Einträge x_i zum Gen DRB1 gehören. Sei $D = \{d_0, d_3, d_4, d_5\}$, wobei d_0 die x_i enthält, welche nicht mit DRB3, DRB4 oder DRB5 assoziiert sind und die restlichen d_i jene x_i enthalten, welche mit DRBi assoziiert sind. Nun muss in unserem Modell gelten, dass die Summe der Einträge in den drei Elementen, welche jeweils mit DRB3, DRB4 oder DRB5 assoziiert sind, den Vorkommen der DRB3, DRB4 und DRB5 Allele entsprechen. Seien

$DRB3'$, $DRB4'$, $DRB5'$ die Mengen, welche jene x_i enthalten, die entsprechend Allele von DRB3, DRB4 und DRB5 sind. Damit können wir dem Modell eine weitere Bedingung hinzufügen:

$$\begin{array}{ll} \min & \sum_{i=1}^{m} |e_i| \\ \text{unter} & cKx + e = y \\ & \forall p \in P : \sum_{x_i \in p} x_i = 2 \\ & \forall d_i \in D, i \in \{3,4,5\} : \sum_{x_j \in d_i} x_j = \sum_{x_k \in DRBi'} x_k \end{array}$$

Eine optimale Lösung x^* dieses linearen Programms liefert uns eine Auswahl von Allelen, welche einen möglichst geringen Fehler beim Vergleichen der zu x^* gehörenden k-mer-Häufigkeiten verursacht.

Nun können wir unser Modell noch etwas verbessern, indem wir die Anzahl der Bedingungen verringern. Dies tun wir, indem wir Einträge in y entfernen. Wir entfernen jene Einträge, die die k-mer-Häufigkeit zu einem k-mer enthalten, welches nicht in der Matrix K vorkommt. Damit die Gleichung aus Abbildung 2 erfüllt ist, müssen diese k-mer-Häufigkeiten, wenn sie nicht entfernt werden, im Fehlerterm e ausgeglichen werden. Diese Fehler lassen sich mit keiner Belegung von x vermeiden. Das heißt, wir können diese Einträge aus dem Modell entfernen ohne das Ergebnis zu beeinträchtigen.

4 Evaluierung

In diesem Kapitel werden die Daten, welche als Grundlage für das lineare Programm verwendet werden, und deren Vorverarbeitung, beschrieben. Zudem wird erklärt, wie der Coverage-Parameter bestimmt wird. Anschließend werden die Ergebnisse des Algorithmus evaluiert.

4.1 Implementierung

Die Software für diese Arbeit wurde mit Python angefertigt [23]. Sie verwendet Gurobi Optimization [22], eine Bibliothek zur Lösung von linearen Programmen. Die Berechnungen wurden auf dem HPC-Knoten der Heinrich Heine Universität Düsseldorf (16 Intel(R) Xeon(R) E5-2667 v4 cores (3.20 GHz) sowie 512 GB RAM) durchgeführt.

4.2 Daten

Wir werden nun eine kurze Beschreibung der Daten betrachten und unsere Methode zur Bestimmung des Coverage Parameters beschreiben.

4.2.1 Beschreibung der Daten

In Kapitel 3 haben wir für unser lineares Programm eine Matrix K und einen Vektor y von k-mer-Häufigkeiten betrachtet. Die Matrix K hat, als Tabelle gelesen, als Einträge zu allen Allelen die jeweiligen k-mer-Häufigkeiten. Wir betrachten Allele der Gene HLA-A, B, C, E, F, G, H, K, L, V, DMA, DMB, DOA, DOB, DPA1, DPB1, DQA1, DQB1, DRA, DRB3, DRB4 und DRB5. Bei diesen werden jedoch nur die Exon-Sequenzen betrachtet, die die Peptidbindestellen kodieren. Da diese für alle Allele innerhalb einer G-group gleich sind, gibt uns der Algorithmus nur Auskunft über die G-group, aus der das zu bestimmende Allel stammt. Die Informationen über die Allele und ihre zugehören Sequenzen bekommen wir aus der IMGT/HLA-Datenbank [8]. Diese beruht auf bekannten Allelen und den dazugehörigen k-mer-Häufigkeiten.

Für den Vektor y, welcher die k-mer-Häufigkeiten eines Datensatzes als Einträge hat, betrachten wir die Reads eines sequenzierten Individuums. Von diesen Reads betrachten wir den Teil, welcher die Peptidbindestellen kodiert, also den Teil, welcher die G-group bestimmt. Die Einträge von y sind die k-mer-Häufigkeiten der Gensequenzen, die aus diesem Bereich stammen.

Dieses Mapping wird mit dem BWA-MEM Algorithmus und die Extraktion mit einem Modul von HLA*PRG durchgeführt [13, 20]. Wir betrachten Reads, die mit unterschiedlichen Sequenzierverfahren erstellt wurden. Die Datensätze sind mit Präfixen markiert, welche die Sequenzierart beschreiben. Das Präfix „Platinum“ steht für 2 x 100bp Whole-Genome Illumina Daten [21], „1000G“ steht für 2 x 250 Whole-Genome Illumina Daten [17] und „SRR“ steht für 2 x 100bp Exome-Sequencing Daten vom HapMap-Project [18].

Unser Algorithmus inferiert auf G-group Ebene, das heißt die Ausgabe des Algorithmus repräsentiert die zu dem Allel gehörende G-group. Die Informationen über die G-groups sind ebenfalls aus der IMGT/HLA-Datenbank [8].

Für die Auswertung betrachten wir die medizinisch relevantesten Gene [16]. Diese sind HLA-A, B, C, DQA1, DQB1, DPB1 und DRB1. Für die Datensätze wurden für diese Gene bereits ihre Allele bestimmt. Damit können wir das Ergebnis unseres Algorithmus mit dem tatsächlichen Ergebnis vergleichen. Dies tun wir mit unterschiedlichen Allelauflösungen.

4.2.2 Bestimmung der Coverage

Unser lineares Programm benötigt die haploide Coverage als Parameter. Diese muss für einen gegeben Datensatz bestimmt werden, falls sie nicht bereits bekannt ist. Um die haploide Coverage für einen gegeben Datensatz zu bestimmen, betrachten wir die Verteilung der k-mer-Häufigkeiten in diesem Datensatz. Dazu erstellen wir ein Histogramm, welches auf der x-Achse k-mer Häufigkeiten abbildet und auf der y-Achse die Anzahl der k-mere mit dieser Häufigkeit.

Anhand eines Histogramms zu einem Datensatz erklären wir nun, was für ein Bild wir erwarten und wie wir daraus die haploide Coverage ablesen können. Die Werte für x=1 und x=2 wurden auf null gesetzt, da diese hauptsächlich

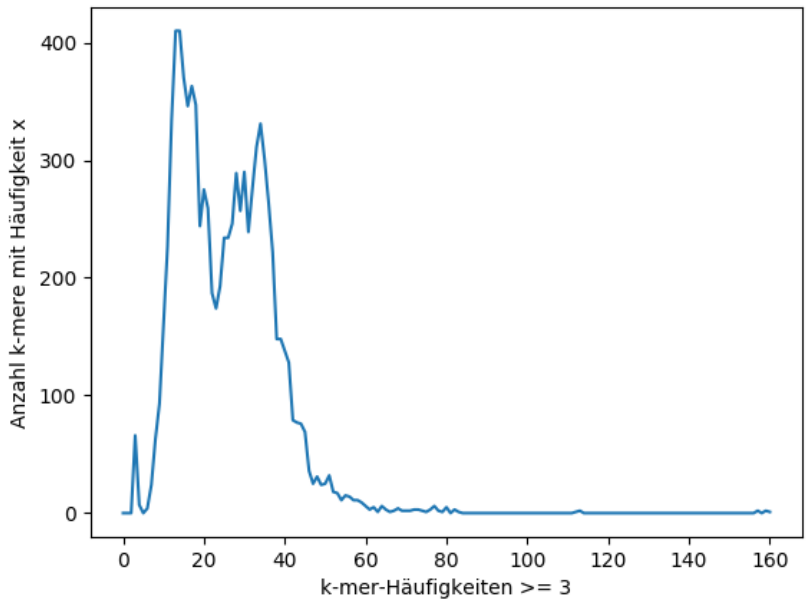

Abbildung 3: Datensatz: 1000G_NA12878
Der Wert für x=1 ist 16040 und für x=2 433.

Sequenzierfehler darstellen. In dem Histogramm sind zwei ausgeprägte lokale Maxima zu erkennen. Das erste Maximum liegt bei x=13 und das zweite bei x=34. Der x-Wert des zweiten Maximums ist die Coverage und der x-Wert des ersten Maximums ist die haploide Coverage. Das erste Maximum stellt die k-mere da, welche auf jeweils einem Chromosom häufig vorhanden sind. Das zweite Maximum stellt die k-mere dar, die auf beiden Chromosomen häufig vorhanden sind. Folglich wird für diesen Datensatz als haploide Coverage 13 gewählt. Wenn das Histogramm dieses Muster hat, können wir die haploide Coverage wie oben beschrieben ablesen.

Wegen der Polymorphie der HLA-Region erwarten wir, dass das erste Maximum höher ist als das zweite. Selbst geringe Veränderungen in den Sequenzen würden sichtbar sein, da selbst eine Basenänderung bereits in k k-meren zu sehen ist.

Wenn es keine Fehler durch die Sequenzierung und das Mapping der Daten gäbe, und jedes k-mer nur einmal in der Gensequenz vorkommen würde, würden wir ein Histogramm erwarten, welches nur an der Coverage und der haploiden Coverage einen Eintrag hat. Wenn wir nun hinzunehmen, dass k-mere mehrfach vorkommen können, würden Einträge an den Vielfachen der Coverage und haploiden Coverage hinzukommen. Da unsere Daten nicht frei von Fehlern sind haben wir ein Rauschen in ihnen.

Für den Platinum NA12891 Datensatz gilt dies genauso.

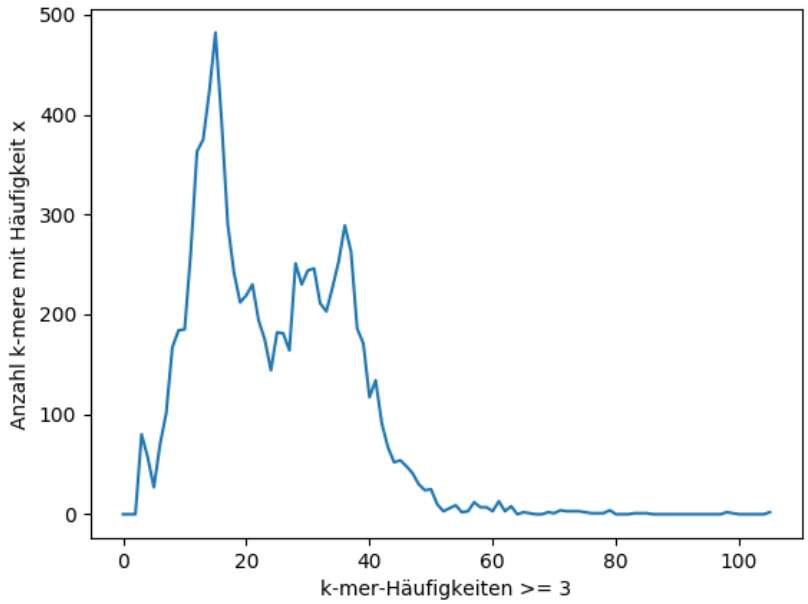

Abbildung 4: Datensatz: Platinum_NA12891

Hier kann die Coverage, sowie die haploide Coverage mit derselben Methode wie für 3 abgelesen werden.

Dies funktioniert nicht immer so. Die „SRR“ Datensätze weisen dieses Muster nicht auf. Dieser Datensatz entspricht nicht dem oben beschriebenen Muster.

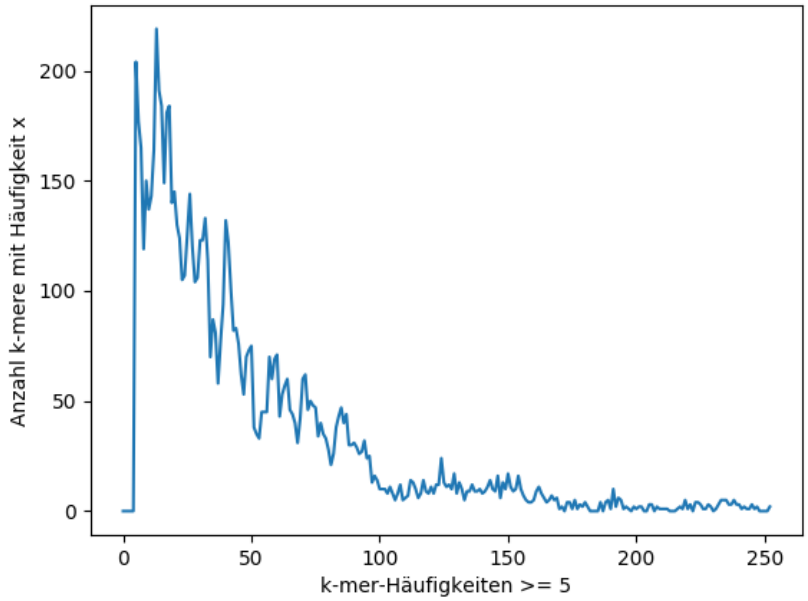

Abbildung 5: Datensatz: SRR_718067

Hier haben wir das globale Maximum als haploide Coverage gewählt.

4.2.3 Evaluierungsprozess

Wir behandeln nun, wie die Auswertung unserer Inferenz erfolgt. Unser Algorithmus liefert uns zu den betrachteten Genen jeweils zwei Allele. Diese können gleich sein. Da der Algorithmus auf G-group Ebene, arbeitet steht ein Allel für die gesamte G-group, in der es enthalten ist. Das heißt, wir expandieren unsere Inferenzallele zu Mengen, welche alle Allele der gleichen G-group enthalten. Für jedes Gen sind jeweils zwei korrekte Allele angegeben, die auch können gleich sein. Die Matches werden für beide Allele eines Gens separat betrachtet. Das heißt, wenn beide Allele für ein Gen richtig inferiert wurden, zählen wir zwei Matches.

Wir betrachten nun den Fall, dass die zwei Allele unterschiedlich sind und dementsprechend die Mengen auch unterschiedlich sind. Seien die beiden Mengen, die durch Expandierung der Inferenz entstanden sind, A1 und A2 (im Fall, dass das gleiche Allel zwei mal vorhanden ist gilt A1 = A2). Seien die möglichen

korrekten Allele für dieses Gen A3 und A4. Wir zählen zwei Matches, falls gilt: $A3 \in A1$ und $A4 \in A2$ oder $A4 \in A1$ und $A3 \in A2$. Wir zählen einen Match, falls gilt: $A3 \in A1$ und $A4 \notin A2$ oder $A3 \notin A1$ und $A4 \in A2$ oder $A4 \in A1$ und $A3 \notin A2$ oder $A4 \notin A1$ und $A3 \in A2$ Sonst zählen wir null Matches. Wenn das korrekte Allel eine geringere Auflösung hat als die Allele in den Mengen, schneiden wir die Allele in den Mengen auf die gleiche Auflösung.

Wenn wir unsere Evaluierung auf einer geringen Auflösung durchführen wollen machen wir nach dem expandieren der Inferenzallele zu Mengen einen Zwischenschritt. Wir schneiden die Allele in den Mengen auf unsere gewünschte Auflösung. Falls ein Allel bereits kleiner aufgelöst ist bleibt es wie es ist. Dann führen wir die Evaluierung wie oben beschrieben durch.

4.3 Ergebnisse

Für die Berechnungen hat unser Algorithmus pro Datensatz zwischen einer und zwei CPU-Stunden benötigt.

Gen	Genauig.: volle Auflösung	Genauig.: 2-Feld	Genauig.: 1-Feld
A	0.375	0.375	1
B	0.75	0.75	1
C	0.625	0.75	1
DPB1	0.375	0.375	0.375
DQA1	0.75	0.75	0.75
DQB1	1	1	1
DRB1	0.75	0.75	1

Tabelle 2: Es wurden vier „1000G “ Datensätze für diese Auswertung verwendet. Die verwendeten Datensätze sind: NA12878(13), NA12891(15), NA12892(18), NA18939(13).
In den Klammern ist die verwendete haploide Coverage angegeben. Die Einträge in der Tabelle sind die relativen Matches.

Wir betrachten nun die Ergebnisse zu den „1000g “ Datensätzen 3 genauer. Wir sehen, dass die Ergebnisse sich mit geringerer Auflösung verbessern. Dies liegt an der hierarchischen Struktur der Allelnomenklatur. Wenn wir bei voller Auflösung einen Match haben, wird dieser bei geringeren Auflösungen bestehen bleiben. Wir erkennen weiterhin, dass sich die Genauigkeit bei DPB1 bei geringeren Auflösungen nicht verbessert. Das hängt damit zusammen, dass sich die meisten Allele von DPB1 in der IMGT/HLA-Datenbank bereits im ersten Feld unterscheiden. Das heißt, bei allen Auflösungen wird für die meisten Allele dieses Gens nur das erste Feld geprüft und, wenn dieses bereits falsch ist, bleibt es in geringeren Auflösungen auch falsch.

Für die Daten aus Tabelle 2 haben wir auf höchster Auflösung 37 von 56 Allelen (66%) richtig vorhergesagt. Auf 2-Feld Auflösung haben wir 38 von 56 Allelen (68%) richtig vorhergesagt. Wir sehen, dass unser Algorithmus auf 2-Feld Auflösung fast genauso gut ist, wie auf voller Auflösung für diese Daten.

Gen	Genauig.: volle Auflösung	Genauig.: 2-Feld	Genauig.: 1-Feld
A	0.25	0.25	0.75
B	0	0	1
C	0.75	0.75	1
DPB1	0.25	0.25	0.25
DQA1	0.75	0.75	0.75
DQB1	1	1	1
DRB1	0.25	0.25	0.75

Tabelle 3: Es wurden zwei „SRR " Datensätze für diese Auswertung verwendet. Die verwendeten Datensätze sind: SRR701474(12), SRR718067(13).
In den Klammern ist die verwendete haploide Coverage angegeben. Die Einträge in der Tabelle sind die relativen Matches.

Gen	Genauig.: volle Auflösung	Genauig.: 2-Feld	Genauig.: 1-Feld
A	0	0	1
B	0	0	1
C	0	0	1
DPB1	0	0	0
DQA1	1	1	1
DQB1	1	1	1
DRB1	0.5	0.5	1

Tabelle 4: Es wurde ein Datensatz verwendet: Platinum_NA12891(15).
In den Klammern ist die verwendete haploide Coverage angegeben. Die Einträge in der Tabelle sind die relativen Matches.

Die Evaluierung fällt für die 1-Feld Auflösung am besten aus mit 49 von 56 richtig vorhergesagten Allelen (88%).

Die Daten aus Tabelle 3 hat unser Algorithmus schlechter vorhergesagt. Auf voller und 2-Feld Auflösung haben wir 13 von 28 Allelen (46%) korrekt vorhergesagt. Wir vermuten, dass dies damit zusammenhängt, dass wir die haploide Coverage für diese Datensätze nicht gut genug schätzen konnten. Auf 1-Feld Auflösung hat unser Algorithmus 22 von 28 Allelen (79%) richtig inferiert.

In Tabelle 4 wurde nur ein Datensatz verwendet, das heißt man kann die gesamt-Matches direkt aus der Tabelle ablesen.

4.3.1 Auswirkung der Coverage

Hier stellen wir kurz dar, dass die Coverage selbst bei kleinen Änderungen bereits Auswirkungen auf unsere Inferenz zeigt. Wir betrachten dazu den Datensatz „1000G_NA12878" welchen wir bereits in 4.2.2 betrachtet haben. Mit Abbildung 3 haben wir die haploide Coverage des Datensatzes auf 13 geschätzt. Wir betrachten nun den gleichen Datensatz, jedoch haben wir die Inferenz mit der haploiden Coverage 15 durchgeführt.

Gen	Genauig.: volle Auflösung	Genauig.: 2-Feld	Genauig.: 1-Feld
A	0	0	2
B	2	2	2
C	1	1	2
DPB1	2	2	2
DQA1	2	2	2
DQB1	2	2	2
DRB1	2	2	2

Tabelle 5: Datensatz: „1000G_NA12878“ mit haploider Coverage 13. Korrekt inferierte Allele in absoluten Angaben.

Gen	Genauig.: volle Auflösung	Genauig.: 2-Feld	Genauig.: 1-Feld
A	0	0	2
B	2	2	2
C	0	0	2
DPB1	2	2	2
DQA1	2	2	2
DQB1	2	2	2
DRB1	2	2	2

Tabelle 6: Datensatz: „1000G_NA12878“ mit haploider Coverage 15. Korrekt inferierte Allele in absoluten Angaben.

Wir sehen, dass wir mit haploider Coverage 13 auf höchster Auflösung sowie auf 2-Feld Auflösung 11 von 14 Allelen korrekt inferiert haben, mit Coverage 15 jedoch nur 10 von 14 für höchste Auflösung und 2-Feld Auflösung.

Daraus folgern wir, dass unsere Inferenz mit einer besseren Schätzung für die haploide Coverage ein präziseres Ergebnis liefert.

5 Diskussion

In diesem Kapitel werden wir die Ergebnisse diskutieren und Ausblicke auf weitere Arbeiten zu diesem Thema geben. In 4.3 sehen wir, dass unser Algorithmus im Vergleich zu anderen Verfahren, wie HLA*PRG [13] und xHLA [15], schlechter abschneidet. Einer der Gründe dafür ist, dass wir nur die k-mer-Häufigkeiten verwenden. Das heißt, wir lassen die Information über die Struktur der Reads weg. Wir bauen unser Modell mit der Idee als Grundlage, dass ähnliche Strings auch ähnliche k-mer-Häufigkeiten haben. Daher ist zu erwarten, dass unser Algorithmus ein qualitativ schlechteres Ergebnis liefert.

5.1 Verwandte Arbeiten

Wir betrachten einige andere Arbeiten, welche sich mit dem Thema der HLA-Typisierung befasst haben. Es gibt bereits Arbeiten, die einen Ansatz mit linearer Programmierung verfolgt haben. Auf diese gehen wir im Folgenden ein. Der OptiType Algorithmus erstellt auch ein ganzzahliges lineares Programm, jedoch verwendet er, im Gegensatz zu unserem Algorithmus, die Reads als Basis des linearen Programms. Die Nebenbedingungen sorgen, wie bei unserem Algorithmus, dafür, dass die richtige Anzahl an Allelen ausgewählt wird. Der Algorithmus versucht die Anzahl an abgebildeten Reads zu maximieren. Ein weiterer auf linearer Programmierung basierender Algorithmus ist xHLA. Dieser versucht das Problem ähnlich wie OptiType zu lösen, beinhaltet jedoch einen weiteren Schritt, der versucht die durch das lineare Programm gefunden Lösung iterativ durch in Betracht ziehen weiterer Exons zu verbessern. OptiType und xHLA sind beide für eine Inferenz auf 2-Felder optimiert und haben eine Genauigkeit von über 90% auf dieser. Sie schneiden im Vergleich also besser ab als unser Algorithmus.

Ein weiterer Algorithmus ist HLA*PRG [13]. Dieser stellt die Datenbank der bekannten Allele als Graph dar. Der Algorithmus aliniert Reads, welche aus der HLA-Region stammen, auf Population Reference Graphs, welche alle bekannten Allele kodieren, und gibt die wahrscheinlichsten Allele aus der Datenbank aus. HLA*PRG hat ebenso eine höhere Genauigkeit wie OptiType und xHLA, kann jedoch auch auf voller Auflösung inferieren. xHLA benötigt für eine Inferenz ca. drei Minuten. HLA*PRG benötigt hingegen über zehn Stunden.

Alle genannten Algorithmen können nur bestehende Allele inferieren aber keine neuen entdecken. Die sie ihre Inferenz so wie unser Algorithmus auf Basis von Datenbanken machen.

5.2 Zukünftige Arbeiten

Unser vorgestelltes Modell ist noch erweiterbar. Durch Hinzuziehen von mehr Informationen und weiteren Bedingungen kann das lineare Programm verbessert werden. Einige Ausblicke darauf werden im Folgenden beschrieben.

In unserem Modell haben wir die haploide Coverage c als Skalar betrachtet. Die haploide Coverage ist jedoch nicht für alle Sequenzierten Abschnitte c. Eine

bessere Möglichkeit diese zu modellieren wäre es, die haploide Coverage als Vektor zu betrachten, welcher die haploide Coverage für ein Gen als Einträge hat. Dies hätte ein genaueres Modell zur Folge.

Bisher ist es so, dass wir die Coverage über die Histogramm 4.2.2 manuell bestimmen müssen. Wenn wir diesen Prozess automatisieren könnten, wäre es möglich den Algorithmus in eine Pipeline einzubinden, sodass wir direkt von den Daten unsere Inferenz erhalten.

Weiterhin kann man die Inferenzen mehrerer Individuen verwenden, um Fehler zu erkennen. Dazu können wir Trio-Konsistenz verwenden. Das heißt, wenn wir bei einem Individuum ein Allel inferieren, muss es bei den Eltern auch enthalten sein. Im Folgenden ein kurzes Beispiel: Seien die Genotypen der Eltern für ein Gen A1/A2 und A3/A4, dann sind die möglichen Genotypen des Kindes A1/A3, A1/A4, A2/A3 und A2/A4. Das heißt, wenn der Genotyp des Kindes beispielsweise A1/A5 wäre, wissen wir, dass die Inferenz in einem der Individuen falsch sein muss, da sie die Trio-Konsistenz nicht erfüllt. Dies wäre ein Schritt, der zwar nicht die Inferenz verbessert, aber Fehler des Algorithmus schnell erkennen kann. Alternativ könnten wir das lineare Programm, welches Grundlage unseres Algorithmus ist, erweitern, so dass wir mehrere Datensätze gleichzeitig betrachten, für die die Trio-Konsistenz gilt und dadurch versuchen eine höhere Genauigkeit zu erzielen.

Weiterhin sind HLA-Typen in unterschiedlichen Populationen unterschiedlich verteilt. Dies könnten wir im linearen Programm verwenden. Beispielsweise könnten wir modellieren, dass HLA-Typen, die weniger wahrscheinlich in einer Population sind stärker in den Fehlerwert einfließen, als wahrscheinlichere HLA-Typen. Dazu müssten wir für einen Datensatz die Information mitgeben können, aus welcher Population dieser stammt.

5.3 Konklusion

Abschließend kann man sagen, dass unser Algorithmus noch nicht ausgereift genug ist, um mit aktuellen Methoden mitzuhalten. Andere Algorithmen zum Typisieren der HLA-Region verwenden lineare Programme nur als einen Bestandteil für ihre Inferenz und ziehen noch weitere Informationen hinzu. Unser Algorithmus basiert dagegen hauptsächlich auf einem linearen Programm und verwendet nicht die gesamten Reads, sondern nur deren k-mer-Häufigkeiten. Um den Algorithmus zu verbessern könnten wir wie OptiType oder xHLA mehr Informationen verwenden. Dazu könnten wir die Vorschläge aus dem Kapitel 5.2 verwenden.

6 Danksagungen

Abschließend möchte ich mich bei Prof. Dr. Gunnar W. Klau bedanken für die Möglichkeit diese Arbeit zu schreiben. Weiterhin möchte ich Dr. Alexander Dilthey danken für seine Geduld mit meinen vielen Fragen und das bereitstellen der Daten. Zugleich danke ich Sven Schrinner für seinen Rat und die regelmäßigen Treffen, welche motivierend waren und für zielgerichtetere Arbeit sorgten.

7 Literaturverzeichnis

Literatur

[1] International Multiple Sclerosis Genetics C, Beecham AH, Patsop oulos NA, Xifara DK, Davis MF, Kemppin en A, et al. **Analysis of immune-related loci identifies 48 new susceptibility variants for multiple sclerosis.** *Nature genetics. 2013; 45(11):1 353–60.* doi: 10.1038/ng .2770 `https://www.nature.com/articles/ng.2770`

[2] Chapman SJ, Hill AV. **Human genetic susceptibility to infectious disease.** *Nature reviews Genetics. 2012;13(3):175–88.* doi: 10.1038/nrg3114 `https://www.nature.com/articles/nrg3114`

[3] Flomenberg N, Baxter-Lowe LA, Confer D, Fernandez-Vi na M, Filipovic h A, Horowitz M, et al. **Impact of HLA class I and class II high-resolution matching on outcomes of unrelated donor bone marrow transplantation: HLA-C mismatching is associated with a strong adverse effect on transplantation out-come.** *Blood.2004;104(7):1923–30.* doi:10.1182/blood-2004-03-0803 `http://www.bloodjournal.org/content/104/7/1923?sso-checked=true`

[4] Snyder A, Makarov V, Merghoub T, Yuan J, Zaretsky JM, Desrichard A, et al. **Genetic basis for clinical response to CTLA-4 blockade in melanoma.** *The New England journal of medicine.* 2014;371(23):2189–99. doi:10.1056/NEJMoa1406498

[5] Bauer, Markus (2008) **A combinatorial approach to RNA sequence-structure alignments**. *FU Berlin* `https://refubium.fu-berlin.de/handle/fub188/4820`

[6] Bertsimas, D. and Tsitsiklis, J. (1997) **Introduction to Linear Optimization.** *Athena Scientific*

[7] SGE Marsh, ED Albert, WF Bodmer, RE Bontrop, B Dupont, HA Erlich, M Fernández-Vina, DE Geraghty, R Holdsworth, CK Hurley, M Lau, KW Lee, B Mach, WR Mayr, M Maiers, CR Müller, P Parham, EW Petersdorf, T Sasazuki, JL Strominger, A Svejgaard, PI Terasaki, JM Tiercy, J Trowsdale: **Nomenclature for factors of the HLA system, 2010.**

Tissue Antigens 2010 75:291-455 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2848993/

[8] Robinson J, Halliwell JA, Hayhurst JH, Flicek P, Parham P, Marsh SGE **The IPD and IMGT/HLA Database: allele variant databases** *Nucleic Acids Research (2015)* 43:D423-431

[9] Lefranc, M.-P. et al., *Nucleic Acids Res.*, 43:D413-422 (2015); doi: 10.1093/nar/gku1056 http://www.imgt.org/ http://www.imgt.org/IMGTrepertoireMHC/LocusGenes/nomenclatures/human/MHC/hla_correspondance_dr_drb.html#notes

[10] Evans DM, Spencer CC, Pointon JJ, Su Z, Harvey D, Kochan G, et al. **Interaction between ERAP1 and HLA-B27 in ankylosing spondylitis implicates peptide handling in the mechanism for HLA-B27 in disease susceptibility.** *Nature genetics.* 2011;43(8):761–7. Epub 2011/07/12. PubMed Central PMCID: PMC3640413. pmid:21743469 https://www.nature.com/articles/ng.873

[11] Pappas DJ, Tomich A, Garnier F, Marry E, Gourraud PA. **Comparison of high-resolution human leukocyte antigen haplotype frequencies in different ethnic groups: Consequences of sampling fluctuation and haplotype frequency distribution tail truncation.** texititHum Immunol. 2015;76(5):374–80. doi: 10.1016/j.humimm.2015.01.029 https://www.sciencedirect.com/science/article/pii/S0198885915000300

[12] de Bakker PI, McVean G, Sabeti PC, Miretti MM, Green T, Marchini J, et al. **A high-resolution HLA and SNP haplotype map for disease association studies in the extended human MHC.** *Nature genetics.* 2006;38(10):1166–72. Epub 2006/09/26. PubMed Central PMCID: PMC2670196. doi:10.1038/ng1885 https://www.nature.com/articles/ng1885

[13] Dilthey AT, Gourraud PA, Mentzer AJ, Cereb N, Iqbal Z, McVean G. **High-accuracy HLA type inference from whole-genome sequencing data using population reference graphs.** *PLoS Comput Biol.* 2016;12(10):e1005151. doi: 10.1371/journal.pcbi.1005151. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5085092/

[14] Szolek A, Schubert B, Mohr C, Sturm M, Feldhahn M, Kohlbacher O. **OptiType: precision HLA typing from next-generation sequencing data.** *Bioinformatics.* 2014;30(23):3310–16. doi: 10.1093/bioinformatics/btu548. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4441069/

[15] Xie C, Yeo ZX, Wong M, Piper J, Long T, Kirkness EF, et al. textbfFast and accurate HLA typing from short-read next-generation sequence data with xHLA. textitProc Natl Acad Sci USA.

2017;114(30):8059–64. doi: 10.1073/pnas.1707945114. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5544337/

[16] Shaw BE, Gooley TA, Malkki M, et al. **The importance of HLA-DPB1 in unrelated donor hematopoietic cell transplantation.** *Blood.* 2007;110(13):4560-4566. https://www.ncbi.nlm.nih.gov/pubmed/17726164

[17] 1000 Genomes Project Consortium,Abecasis GR,Auton A, Brooks LD, DePristo MA, Durbin RM, et al. **An integrated map of genetic variation from 1,092 human genomes.** *Nature. 2012; 491 (7422):56 –65. 3498066.* doi: 10.1038/nature11632

[18] International HapMap C. **A haplotype map of the human genom** *e. Nature. 2005; 437(7063): 1299–320.*

[19] van Deutekom HW, Keşmir C. **Zooming into the binding groove of HLA molecules: which positions and which substitutions change peptide binding most?.** *Immunogenetics.* 2015;67(8):425-36. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4498290/

[20] Heng Li **Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM.** arXiv:1303.3997 [q-bio.GN] https://arxiv.org/abs/1303.3997

[21] Eberle, MA et al. (2017) **A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree.** *Genome Research 27: 157-164.* doi:10.1101/gr.210500.116

[22] http://www.gurobi.com/

[23] https://www.python.org/

A Code

Der folgende Link wurde am 25.10.2018 besucht. `https://gitlab.cs.uni-duesseldorf.de/klau/BSc-thesis-HLA_inference`