Albert-Ludwigs-Universität Freiburg

Bachelorarbeit

RNA Barcodes for High-Throughput Sequencing Experiments

Autor:
Daniel Desirò

Supervisor:
Prof. Dr. Rolf Backofen
Dipl. Inf. Daniel Maticzka

July 2013

Impressum

Autor	Daniel Desirò
Bearbeitungszeit	15. Mai 2013 bis 15 August 2013
Gutachter	Prof. Dr. Rolf Backofen, Lehrstuhl für Bioinformatik
Betreuer	Dipl. Inf. Daniel Maticzka, Lehrstuhl für Bioinformatik
Prüfungsordnung	Die eingereichte Bachelorarbeit ist gemäß den Bestimmungen der Prüfungsordnung der Albert-Ludwigs-Universität Freiburg für den Bachelorstudiengang Informatik vom 31.08.2010 erstellt worden.
Prüfstelle	Lehrstuhl für Bioinformatik Institut für Informatik Albert-Ludwigs-Universität Freiburg
Erklärung	Hiermit erkläre ich, dass ich die vorliegende Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder sinngemäßverwendete Schriften wurden als solche gekennzeichnet. Weiterhin erkläre ich, dass diese Abschlussarbeit nicht bereits für eine andere Prüfung angefertigt wurde. Freiburg, den 13. August 2013 Daniel Desirò

Zusammenfassung

In dieser Bachelorarbeit wurden zunächst verschiedene, mit iCLIP Hochdurchsatz-sequenzierte Datensätze ausgewertet. iCLIP verwendet zur Identifikation der einzelnen PCR-Duplikate kurze, zufällig erstellte Sequenz Tags. Diese Sequenz Tags können während der Durchführung von iCLIP mutieren, wodurch ein neuer Sequenz Tag entsteht. Dies wird als negatives Event bezeichnet und führt zu Fehlern in der Identifizierung der verschiedenen PCR-Duplikationen. Ein Sequenz Tag der korrekt, also ohne Mutation, dupliziert wird, ist ein positives Event. Um das Verhältnis von negativen zu positiven Events zu untersuchen, wurden hier vier unterschiedlich erstellte Sequenz Tags analysiert. Hierzu gehört zum einen ein Set aus zufällig erstellten Sequenz Tags, zwei verschiedene Sets an Sequenz Tags, welche beide auf der Hamming Codierung basieren und ein, mit Hilfe der Levenshtein Distanz erstelltes, Set an Sequenz Tags.
Aus den ausgewerteten iCLIP Datensätzen wurden zunächst die Wahrscheinlichkeiten der einzelnen Basenmutationen berechnet. Diese wurden dann verwendet, um für jedes Sequenz Tag Set PCR-Duplikationen zu simulieren. Die aus den Simulationen entstandenen Daten wurden dann verglichen, um die Unterschiede der einzelnen Methoden und deren Eignung zum korrekten Erkennen von Events zu erläutern.

Danksagung

Ich möchte mich bei Professor Dr. Rolf Backofen für die Möglichkeit, an seinem Lehrstuhl diese Bachelorarbeit schreiben zu dürfen, bedanken. Auch möchte ich mich bei Daniel Maticzka für das interessante Thema und die Betreuung meiner Bachelorarbeit bedanken. Mein Dank gilt auch meiner Mutter, die mich in der letzten Wochen vor der Abgabe so gut bewirtet hat und auch meiner Freundin, für ihre Unterstützung.

Inhaltsverzeichnis

Abkürzungen

iCLIP	**I**ndividual-Nucleotide Resolution UV **C**ross-**L**inking and **I**mmuno**p**recipitation
PCR	**P**olymerase **C**hain **R**eaction
PNT	**P**ercentage of **N**egative to **T**otal Events
RTE	**R**ate of **T**rue **E**vents
TFT	X **T**imes More **F**alse than **T**rue Events
MDS	**M**ultiplex **D**eep **S**equencing

Kapitel 1

Einleitung

Aktuelle Multiple Hochdurchsatz-Sequenzierung (MDS) ist ein unentbehrliches Werkzeug in der DNA Forschung. Sie erlaubt es, große Mengen an verschiedenen DNA Fragmenten in einem einzigen Durchlauf zu sequenzieren [3]. Dies bietet im Vergleich zu älteren Sequenzierungsmethoden, bei welchen DNA Fragmente einzeln mit Hilfe der Polymerasen-Kettenreaktion (PCR) vervielfältigt und dann aligniert werden, einen klaren Vorteil an Zeit-, Kosten- und Arbeitsaufwand.
Bei der Multiplen Hochdurchsatz-Sequenzierung werden Sequenz Tags verwendet. Diese wurden zunächst zufällig aus der Gesamtmenge an Codewörtern, einer bestimmten Länge ausgewählt. Um jedoch weniger Fehler und dadurch eine bessere Ausbeute zu erhalten, wurden Auswahlverfahren auf Sequenz Tags Sätze bestimmter Länge angewendet [5–7]. Dabei wurden zunächst jegliche Permutationen einer DNA Sequenz mit bestimmter Länge generiert und dann nach bestimmten Kriterien einzelne Sequenz Tags herausgefiltert. Ein Auswahlkriterium ist z.B. die Menge an GC Gehalt im Sequenz Tag. Die meisten Auswahlverfahren von kommerziell verfügbaren Sequenz Tags, wie z.B. von Illumina und Epicentre, sind jedoch nicht zugänglich [8–10]. Diese ausgewählten Sets wurden allerdings für bestimmte Plattformen und Plattform spezifische Fehler erstellt [2]. Um sie fehlerkorrigierend und dadurch plattformübergreifend zu machen, könnten diese Sequenz Tags mit Hilfe von Algorithmen codiert werden. Die Hochdurchsatz-Sequenzierung wird dabei auch in Zusammenhang mit anderen Methoden verwendet. Ein Beispiel hierfür ist

iCLIP, welches mit Hilfe von cDNA das genaue Lokalisieren von Protein-RNA-Interaktion in intakten Zellen erlaubt [1]. iCLIP benutzt dabei die Sequenz Tags, um cross-link Fragmente zu markieren [1]. Dadurch löst es ein inhärentes Problem aller aktuellen Multiplen Hochdurchsatz-Sequenzierungs Methoden, welche Schwierigkeiten mit PCR Artifakten wie Homopolymeren oder Fehlern in der Reversen Transkription, PCR oder Sequenzierung haben [1]. Diese eindeutigen Markierungen, welche bei iCLIP verwendet werden, bestehen aus einer kurzen, zufälligen k-meren DNA Sequenz. Die Länge ist dabei von der gewünschten Anzahl an Sequenz Tags abhängig. Aus der Anzahl an Basen n und der Länge der Sequenz k ergibt sich die maximale Anzahl an verschiedenen Codewörtern k^n [3]. Die zuvor genannten Idee von codierenden Sequenz Tags könnte auch auf die iCLIP Sequenz Tags angewendet werden.

Bei der Arbeit mit PCR muss immer mit Fehlübersetzungen wie DNA Mismatche, sowie Insertionen und Deletionen gerechnet werden. Wenn diese Fehler nicht erkannt werden, entsteht ein neues Event und somit ein neuer Sequenz Tag. Aus der Anzahl an Events kann iCLIP auf die Anzahl an gefundenen cross-linking Events in der DNA rückschließen. Es können, besonders bei zufälligen Sequenz Tags, auch Überschneidungen zwischen verschiedenen Sequenz Tags (*Overlaps*) auftreten. Dabei entsteht aus einem Sequenz Tag ein bereits vorhandener Sequenz Tag, welcher dann falsch zugeordnet wird. Dies ist natürlich auch ein neues negatives Event und es kann bei gehäuftem Auftreten dieser, *Overlaps* zu erheblichen Fehlern in der Analyse kommen. Bei komplett zufällig erzeugten Sequenz Tags kann dies sogar schon bei einzelnen Mismatch Fehlern auftreten.

In dieser Bachelorarbeit wird deshalb genauer auf die Vorteile in der Verwendung von codierenden Sequenz Tags für iCLIP eingegangen. Diese codierenden Sequenz Tags werden Barcodes genannt. Zunächst wird der Datensatz einer iCLIP Sequenzierung mit zufälligen Sequenz Tags analysiert. Anschließend werden mit Hilfe einer Simulation die zufälligen Sequenz Tags des Datensatzes mit 3 verschiedenen Barcodes verglichen und analysiert. Die ersten beiden Barcodes basieren auf dem von Hamming [11] entwickelten codierungs System, der 3. auf der von Levenshtein [12] entwickelten Distanzmethode.

Kapitel 2

Grundlagen

2.1 Ablauf von iCLIP

Im Ablauf von iCLIP[1] (Abbildung 2.1) werden zunächst lebende Zellen mit UV-Licht bestrahlt, um Proteine und RNA in vivo kovalent zu vernetzen. Danach wird die kovalent gebundene RNA mit einem RNA-bindenden Protein (RBP) co-immunoprecipiert und anschließend an das 3' Ende eines RNA Adapters gebunden. Durch einen von Proteinase K hervorgerufenen Abbau, kann ein kovalent gebundenes Polypeptide Fragment auf der RNA gewonnen werden. Dieses Fragment bewirkt an der Vernetzungsstelle einen vorzeitigen Abbruch der Reversen Transkription (RT).
Die resultierenden cDNA Moleküle werden dann identifiziert, linearisiert, mit Hilfe von PCR vermehrt und einer Hochdurchsatz-Sequenzierung unterzogen. Die ersten paar Nukleotide jeder Sequenz enthalten dann die Sequenz Tags gefolgt von dem Nukleotid an welchem die cDNA, während der Reversen Transkription, abgebrochen wurde.

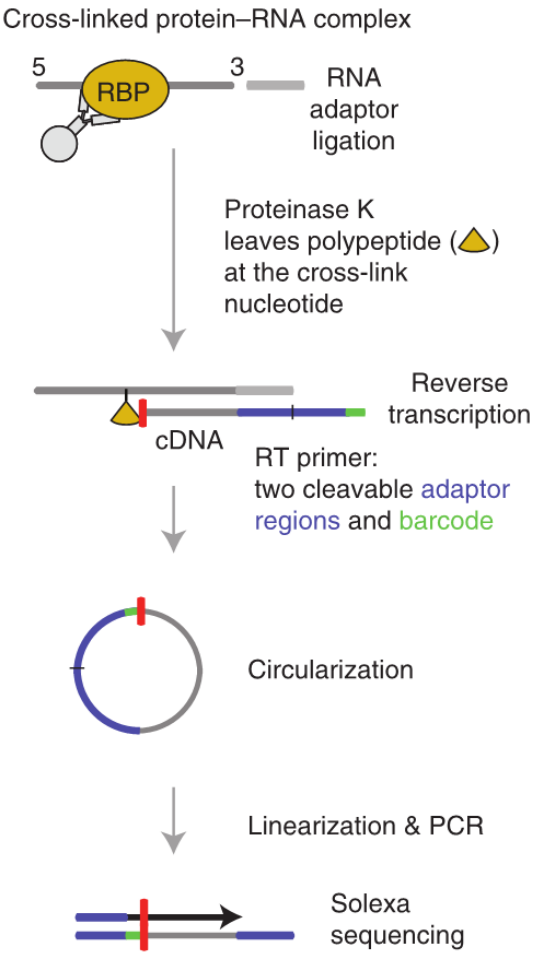

Abbildung 2.1: Abbildung erstellt von König et al. [1]. Schematische Darstellung eines iCLIP Ablaufs. Der rote Strich bezeichnet das letzte Nukleotid, welches während der Reversen Transkription hinzugefügt wurde [1].

2.2 Barcode Codierung

2.2.1 Hamming Codierung für Barcodes (binär)

Hamming Code ist ein fehlererkennender und fehlerkorrigierender Code, welcher auf Bitebene arbeitet. Er besteht zum Teil aus Datenbits und beinhaltet an jeder 2^n Stelle ein Paritätsbit, welches zur Berechnung der Prüfsumme verwendet wird. Die erste Prüfsumme ergibt sich, beginnend an der ersten Position des Codewortes, aus jeder ungeraden Position, die zweite Prüfsumme beginnt mit der 2. Position und beinhaltet im Intervall von 2 Bits immer 2 aufeinanderfolgende Bits, die 3. Prüfsumme beginnt mit der 4. Position und enthält im Intervall von 4 Bits

immer 4 aufeinanderfolgende Bits, und so weiter [3]. Die Minimaldistanz zwischen verschiedenen Codewörtern gleicher Länge ist dabei immer 3 oder größer. Es kann somit ein Fehler korrigiert werden und zwei erkannt werden. Die Position des Fehlers kann dann direkt über die Prüfsummen ermittelt werden. Dabei wird für jede Prüfsumme die Anzahl an 1en an den entsprechenden Stellen im Code gezählt. Ist diese ungerade, ist die Prüfsumme 1, sonst 0. Anschließend werden die einzelnen Ergebnisse in umgekehrter Reihenfolge verbunden. Die neu entstandene Binärzahl gibt dann die Stelle des Fehlers im Code an.

Unterschiedliche Hamming Codes werden anhand ihrer Länge n und Anzahl an Datenbits d unterschieden und mit Hamming(n,d) benannt. Die Menge an möglichen Codewörtern lässt sich durch 2^d berechnen [4]. Ein Beispiel ist der meistverwendete Code, Hamming(7,4) welcher eine Länge von 7 Bits hat, von denen 4 Datenbits und 3 Paritätsbits sind, und mit dem sich 2^4 verschiedene Codewörter erstellen lassen.

Das Problem bei Hamming Codes in Verbindung mit DNA ist leicht erkennbar. Hamming Codes arbeiten nach einem binärem Schema (0,1), DNA hingegen ist ein quartärer Code (A,C,G,T). Die DNA Barcodes müssen somit erst in die Binärebene übersetzt werden. Hierfür wurde von Hamady et al. [4] ein einfaches Übersetzungsschema von je 2 Bits pro DNA Base vorgeschlagen. Für diese Bachelorarbeit wurde dabei „A“ als 00, „C“ als 01, „G“ als 10 und „T“ als 11 codiert. Da Hamming Codes an sich eine ungerade Länge haben, dieses Übersetzungsschema jedoch zwangsläufig auf Codes gerader Länge hinausläuft, muss an letzter Stelle des Binärcodes noch ein zusätzliches Paritätsbit hinzugefügt werden [3]. Hamming(7,4) würde somit zu Hamming(8,4) werden und hätte als DNA Barcode eine Länge von 4.

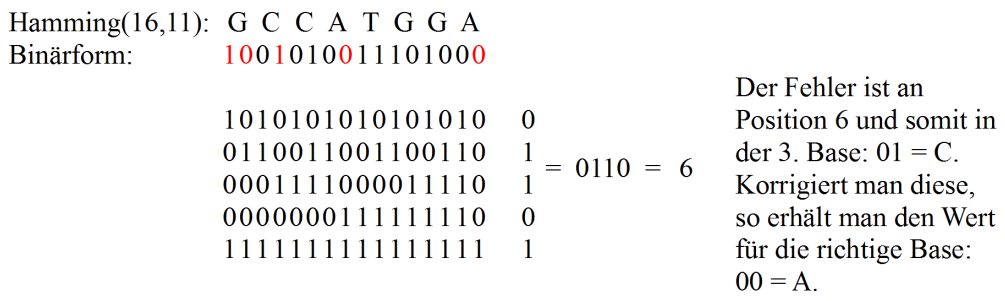

ABBILDUNG 2.2: Beispiel für eine Hamming(16,11) Fehlerkorrektur. Rote Einträge sind Paritätsbits. Es erfolgt eine Matrixmultiplikation zwischen dem binären Barcode und der Encodingsmatrix. Die ersten 4 Positionen der resultierenden Matrix bestimmen die Position des Fehlers. Sollten diese 0 sein, dann wird die 5. Position überprüft. Steht dort eine 1, so deutet dies auf einen Fehler an der letzten Stelle des Barcodes hin. Ist diese auch 0, wird der Barcode als fehlerfrei deklariert.

2.2.2 Hamming Codierung für Barcodes (quartär)

Der binäre Hamming Code lässt sich auch wie folgt direkt auf die quartäre DNA Ebene anwenden. Hierfür werden zunächst die DNA Basen A,C,G,T als 0,1,2,3 codiert [3]. Die Positionen der Kontrollziffern sind gleich denen der Paritätsbits im binären Hamming Code. Ebenso werden auch dieselben Positionen für die Berechnung der einzelnen Prüfsummen verwendet. Die an diesen Positionen stehenden Zahlen werden dann zur Prüfsummen aufsummiert. Um dann daraus die Position des Fehlers zu ermitteln, muss zunächst die modulo 4 Funktion auf die einzelnen Prüfsummen angewendet werden. Dadurch werden die Reste der Divisionen zwischen den verschiedenen Summen und 4 ermittelt. Die neuen Prüfsummen werden dann in umgekehrter Reihenfolge verbunden und ergeben so die Fehlernachricht. Durch diese Fehlernachricht kann nun zum einen die Fehlerposition und zum anderen der Fehlertyp berechnet werden. Die Fehlerposition ergibt sich, wenn die Fehlernachricht in eine Darstellung aus 0 und 1 übertragen wird. Dabei werden 0en direkt übertragen und alles was größer als 0 ist, wird zu einer 1. Die daraus entstandene Binärzahl gibt die Position des Fehlers an. Der Fehlertyp ergibt sich aus dem maximalen Wert der Fehlernachricht. Um den korrekten Wert der Base zu ermitteln, wird der Fehlertyp von dem Wert der fehlerhaften Base subtrahiert und das Resultat modulo 4 genommen. Wenn das Ergebnis größer oder gleich 0

ist, kann daraus direkt die korrekte Base ermittelt werden. Sollte das Ergebnis kleiner als 0 sein, so wird der Wert für die Base nach folgendem Schema ermittelt: -3 ist 1, -1 ist 3, -2 ist 2 [3].

Die Menge an möglichen Codewörtern einer Länge n mit d Datenstellen, ergibt sich aus 4^d. Ein quartärer Hamming(7,4) hätte somit $4^4 = 256$ mögliche Codewörter [3].

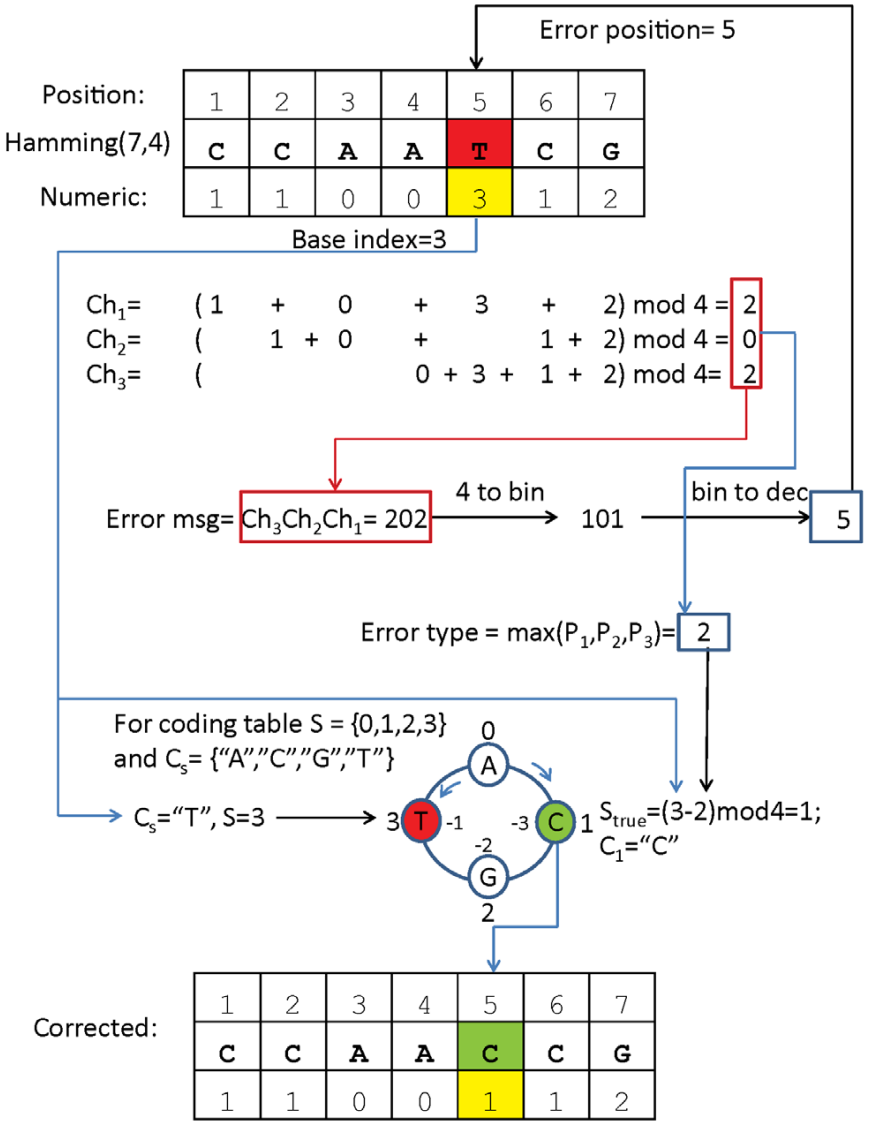

ABBILDUNG 2.3: Abbildung erstellt von Bystrykh [3]. Hier wird das Decodierungsschema eines quartären Hamming(7,4) erläutert. Die Fehlernachricht ergibt sich aus den Prüfsummen $Ch_3Ch_2Ch_1 = 202$. Durch die Binärumwandlung entsteht 101, was einen Fehler an Position 5 beschreibt. Der Fehlertyp berechnet sich aus $max(Ch_3, Ch_2, Ch_1) = 2$. Daraus und aus dem Wert der Fehlerhaften Base „T“ = 3, ergibt sich dann der Wert der korrekten Base mit $S_{true} = (3 - 2)\, mod\, 4 = 1$ und somit die Base 1 = „C“ [3].

2.2.3 Levenshtein Distanz für Barcodes

Mit Hilfe der Levenshtein Distanz können nicht nur, wie bei Hamming Codes, einfache Substitutionen erkannt werden, sondern zusätzlich auch Insertionen und Deletionen. Barcodes welche mit Hilfe der Levenshtein Distanz erzeugt werden haben somit einen klaren Vorteil. Es kann somit auch mehr als nur ein Fehler erkannt werden. Die benötigte Editierdistanz um eine bestimmte Anzahl an Fehler zu erkennen, kann dabei mit nachfolgender Gleichung (Gleichung 2.1) berechnet werden [2].

$$benötigte\,Editierdistanz\ =\ 2\,x\,(erkennbare\,Fehler)\ +\ 1 \tag{2.1}$$

Um diese Barcodes zu erzeugen, wurden von Faircloth et al. [2] zunächst jegliche Kombinationen von DNA Codewörtern (A,C,G,T) einer bestimmten Länge n generiert. Um die Rechenoperationen zu verringern, wurden Barcodes mit problematischen Strukturen aussortiert. Hierzu gehörten Barcodes mit Homopolymeren, ungeeigneten GC Gehalt und perfekten Selbstkomplement [2]. Aus dem übrigen Set wurde dann ein neues Set aus Barcodes, welche untereinander die gewünschte Minimaldistanz haben, erstellt.
Um dann die Fehler in der Auswertung zu erkennen, müssen für jeden Barcode die Editierdistanzen zwischen dem Barcode und allen erstellten Sequenzen berechnet werden. Bei einer Editierdistanz zwischen 0 und der Anzahl Fehlern k, welche das Barcode Set erkennen kann (Gleichung 2.1), wird die Sequenz dem Barcode zugeordnet. Ist die Distanz $k + 1$, so kann die Sequenz zwar nicht zugeordnet werden, wird aber eindeutig als fehlerhaft erkannt. Ist die Distanz größer als $k + 1$, so muss die Sequenz zu einem anderen Barcode gehören. Ein Barcode Set mit einer Basenlänge n und einer Editierdistanz d kann mit Levenshtein(n,d) beschrieben werden.

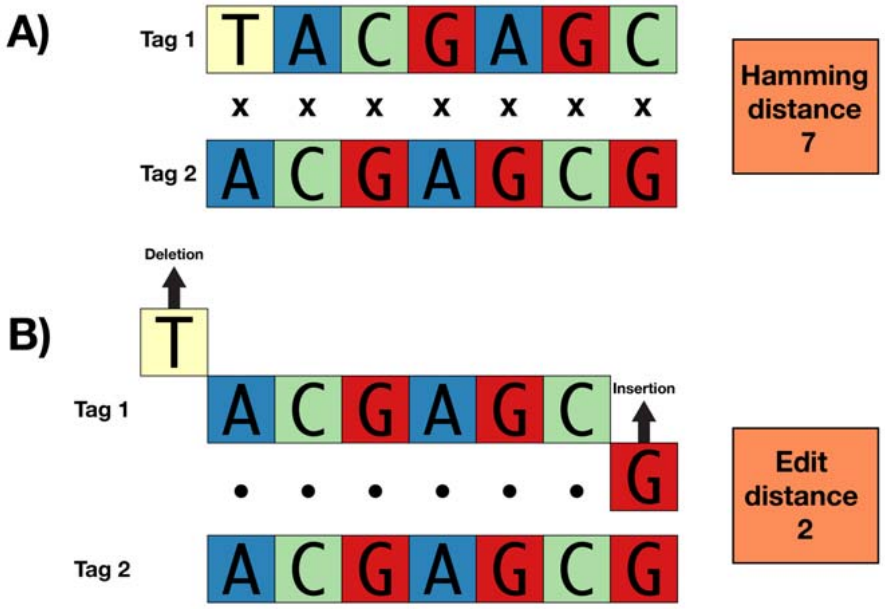

Abbildung 2.4: Abbildung erstellt von Faircloth et al. [2]. Distanz Vergleich zwischen Hamming (A) und Levenshtein (B).

2.3 Verwendete Methoden

2.3.1 Vorbereitung

Als Vorbereitung für das Mismatch Script (Script S1) wurden die iCLIP Datensätze sortiert und Einträge mit gleichem Sequenz Tag und gleicher Position zu einem Eintrag zusammengefasst. Dabei wurde für jeden dieser Einträge auch die Anzahl an identischen Einträgen notiert. Die resultierenden Datensätze hatten dann die nachfolgende Struktur (Struktur 2.2). Dabei dienten die „start“, „chr“, „strand“ und „stop“ Einträge als Schlüssel um die Sequenz Tags in einzelne Sets aufzuteilen. Der „count“ Eintrag beschreibt die Anzahl an Vorkommen von Einträgen in den original Datensätzen mit gleichem Sequenz Tag und gleicher Position.

$$[start]\,[chr]\,[strand]\,[stop]\,[barcode]\,[count] \tag{2.2}$$

2.3.2 Fehlerwahrscheinlichkeiten und zufällige Sequenz Tags

Anschließend wurden mit Hilfe des Mismatch Scripts (Script S1) die unabhängigen Wahrscheinlichkeiten der einzelnen Basenmutationen in den Datensätzen analysiert. Hierbei wurden Einträge mit gleicher Position gruppiert. Der Sequenz Tag mit dem höchsten „count“ Wert in jeder Gruppe wurde dann als „top“ Sequenz Tag gekennzeichnet, und es wurden für alle restlichen Sequenz Tags die Anzahl an Mismatche zu dem jeweiligen „top“ Sequenz Tag berechnet. Dabei konnten direkt die Vorkommen der einzelnen Basen, sowie die Anzahl der einzelnen Mismatchtypen (A zu C, A zu G, etc.) berechnet werden. Um möglichst sichere Events zu erhalten, wurde ein Schwellwert für den „top“ Sequenz Tag gewählt. Dadurch wurden nur Gruppen verwendet, bei welchen der „top“ Sequenz Tag einen „count“ von mindestens dem des Schwellwerts hatte. Zusätzlich wurden nur Sequenz Tags betrachtet, welche maximal 2 Mutationen zum „top“ Sequenz Tag hatten, wodurch eventuelle, durch Insertionen und Deletionen bedingte Mutationen umgangen wurden. Während dieser Analyse wurden auch statistisch ungünstige Chromosome aussortiert. Diese hatten eine zu hohe Anzahl an Events pro Position, was zu vielen *Overlaps* führte. Als Ausgabe wurden für jeden der 12 Datensätze 3 verschiedene Dateien erstellt.
Die erste Datei (Datensatz DS1) wurde zur Kontrolle und Übersicht der für die Wahrscheinlichkeiten verwendeten Sequenz Tags erstellt. In ihr befinden sich, außer den aussortierten Einträgen, alle Sequenz Tags mit den dazugehörigen Schlüsseldaten. Zu den jeweiligen Einträgen wurde noch der Sequenz Tag Typ (top, 1MM, other), die Anzahl an Mismatche und der Anteil zum „top“ Sequenz Tag, in nachfolgender Struktur (Struktur 2.3) eingetragen.

$$[start]\,[chr]\,[strand]\,[stop]\,[barcode]\,[count]\,[type]\,[mismatches]\,[fraction] \quad (2.3)$$

Um die für die Simulation benötigten zufälligen Sequenz Tags zu erhalten, wurden in der zweiten Datei (Datensatz DS2) alle erkannten „top“ Sequenz Tags, welche mindestens 10-fach vorhanden sind, gespeichert. Die für die Simulation gebrauchten Auftreten der einzelnen Basen und Mutationen wurden in der dritten Datei

(Datensatz DS3) gespeichert. Zusätzlich können dort noch einmal alle Informationen über den verwendeten Schwellwert, die aussortierten Chromosomen und die Fehlerwahrscheinlichkeiten nachgesehen werden.
Die Sequenz Tags und Wahrscheinlichkeiten der verschiedenen Datensätze wurden dann für die anschließende Verwendung im Simulations Script in 2 Dateien (Datei D1, D2) zusammengeführt.

2.3.3 Simulationen

Das Simulations Script (Script S2) benötigt als Eingabe ein Barcode Set, die Anzahl an gewünschten Simulationen und eine Mismatchdatei. In dieser ist das Auftreten der einzelnen Basenmutationen und die Gesamtzahl, der jeweils für die Mismatchstatistik verwendeten Basentypen abgespeichert. Das Script wurde hierbei mit der, aus den iCLIP Datensätzen gewonnenen, Mismatchstatistik (Datei D2) auf das Set aus zufälligen Sequenz Tags (Datei D1) und jeweils ein Set der 3 verschiedenen Barcodetypen (Datei D3, D4, D5) angewandt. Es wurden für jeden Durchlauf 1000 mal 200, 500 und 1000 PCR Kopien simuliert und pro Durchlauf der gleiche Seed für die Wahrscheinlichkeiten verwendet.
Das Script berechnet zunächst aus der Mismatchstatistik (Datei D2) für alle möglichen Mutationen die Wahrscheinlichkeit, dass eine Base in eine andere mutiert. Anschließend werden alle Sequenz Tags/Barcodes im Set vervielfältigt. Dabei wird für jede Base in den Sequenz Tags/Barcodes eine Zufallszahl generiert, welche dann mit den 3 möglichen Mutationen verglichen wird. Ist die Zufallszahl im Bereich einer der möglichen Mutationen, so wird die Base verändert. Die resultierenden simulierten Sequenz Tags/Barcodes werden dann in einen Hash mit einer Referenz zum original Sequenz Tag/Barcode gespeichert, um dann mit der jeweiligen Methode überprüft zu werden. Bei allen Sets werden die simulierten Sequenz Tags/Barcodes jeweils mit ihrem original Sequenz Tag/Barcode verglichen und die Anzahl an fehlerhaften Sequenz Tags/Barcodes gezählt. Zusätzlich wird während der Überprüfung, bei allen Methoden, die Anzahl an richtig positiven Events, negativen Events und korrigierten positiven Events berechnet.

Für zufälligen Sequenz Tags wird zusätzlich die Anzahl an Mismatche für jeden Sequenz Tag berechnet.
Simulierte Binäre Hamming Barcodes werden zunächst in Binärcodes übersetzt. Anschließend werden die einzelnen Prüfsummen berechnet und konkateniert. Die entstandene Binärzahl gibt dann im Falle einer Mutation die Fehlerposition an, wodurch dann der Barcode gegebenenfalls korrigiert werden kann. Danach werden sie wieder zurück in DNA Code übersetzt und mit dem original Barcode verglichen. Hierbei wird die Anzahl der erfolgreich und der nicht erfolgreich erkannten Barcodes gezählt und gespeichert.
Für die Korrektur der quartären Hamming Barcodes wurde der Algorithmus von Bystrykh [3] verwendet. Der Algorithmus überprüft und korrigiert, mit Hilfe der von Bystrykh [3] entwickelten Quartären Hamming Decodierung, die simulierten Barcodes. Anschließend wird der eventuell korrigierte Barcode mit dem Original verglichen und die Zahl der erfolgreich korrigierten Barcodes aufsummiert und gespeichert.
Die mit der Levenshtein Distanz erzeugten simulierten Barcodes werden mit Hilfe eines Editierdistanz messenden Algorithmus von Faircloth et al. [2] kontrolliert. Um die Rechenzeit zu verringern, wurden nur die Distanzen zwischen den simulierten Barcodes und dem jeweiligen original Barcode berechnet. Bei einer Distanz von 1 wurden die Barcodes als erfolgreich zuordenbar abgespeichert und bei einer größeren als nicht erkennbar.
Nach der jeweiligen Überprüfung wurden noch alle simulierten Sequenz Tags/Barcodes mit allen Sequenz Tags/Barcodes aus dem Set verglichen, um entstandene Überschneidungen zu finden.
Am Ende kann das Script für jeden der Durchläufe 5 verschiedene Ausgabedateien (Datensatz DS4) erzeugen, welche die einzelnen Berechnungen und Ergebnisse beibehalten. Aufgrund der großen Datenmenge wurde aber die Erstellung der meisten, rein zur Fehlerkontrolle gebrauchten Dateien deaktiviert. Die Statistik Datei (Datensatz DS4) enthält die Endergebnisse der Simulation. Hierzu zählen Informationen über fehlerhafte Simulationen, Korrekturen, nicht korrigierbare Barcodes, Überschneidungen und Events. PNT bezeichnet das Verhältnis von negativen

Events zu allen Events. RTE ist die Rate von richtig positiven Events. TFT zeigt den Faktor aller negativen Events zu positiven Events.

2.4 Datensätze

2.4.1 iCLIP Datensätze

Zur Berechnung der Mismatch Wahrscheinlichkeiten wurden 12 verschiedene iCLIP Datensätze verwendet, welche mit zufälligen, 5 Nukleotid langen Sequenz Tags erstellt wurden. Da bei diesen 12 Datensätzen 1024 verschiedene Sequenz Tags verwendet wurden, war ein Sequenz Tag Set von 1000+ auch eine notwendige Voraussetzung für die anderen Barcode Sets. Für die Analyse wurden die statistisch problematischen Chromosome chrX und chrXHet herausgefiltert und die zufälligen Sequenz Tags bestimmt.

2.4.2 Binär codierte Hammig Barcodes

Da Hamady et al. [4] keine Barcodes mit binärer Hammingcodierung freigestellt haben, wurden die binären Hamming(16,11) Barcodes (Datei D3), welche Bystrykh [3] zur Verfügung gestellt hat, benutzt. Binäre Hamming(16,11) haben eine Bitlänge von 16, wovon 11 Datenbits und 5 Paritätsbits sind. Zur Übersetzung wurden immer 2 aufeinanderfolgende Bits in ein Nukleotid mit dem Übersetzungsschema 00 zu „A“, 01 zu „C“, 10 zu „G“ und 11 zu „T“, übersetzt. Die Barcodes sind somit 8 Basen lang und es steht eine ausreichende Menge von 2048 Stück zu Verfügung.

2.4.3 Quartär codierte Hamming Barcodes

Um eine Menge von 1000+ verschiedene quartär codierten Hamming Barcodes zu erhalten, wurden die von Bystrykh [3] mitgelieferten quartären Hamming(9,5)

(Datei D4) verwendet. Diese haben eine Länge von 9 Basen, wovon 5 Datenstellen und 4 Paritätsstellen sind. Die Menge an verschiedenen Barcodes ist hierbei 1023.

2.4.4 Levenshtein Distanz Barcodes

Für die durch die Levenshtein Distanz differenzierten Barcodes wurde das von Faircloth et al. [2] bereitgestellte Set an Levenshtein(9,3) Barcodes (Datei D5) verwendet. Diese haben eine Länge von 9 Basen und einer minimalen Editierdistanz von 3 untereinander. Es kann somit einen Fehler korrigieren und bietet einen optimalen Vergleich zu den anderen beiden codierten Barcode Sets. Die Menge an verwendeten Barcodes war dabei 1936.

2.5 Verwendete Programme

Alle verwendeten Programme wurden als Perl Scripte implementiert. Darunter fallen die beiden Hauptscripte mismatches.pl (Script S1) und simulator.pl (Script S2), sowie die Hilfsprogramme (Script S3, S4, S5), welche verwendet wurden, um die Daten für die beiden Hauptscripte vorzubereiten. Alle Scripte, sowie Ergebnisse und Sequenz Tags/Barcodes sind als Zusatzmaterial verfügbar.

Kapitel 3

Ergebnisse

3.1 iCLIP Datensätze und Wahrscheinlichkeiten

Für die Berechnung der Mismatch Statistiken wurden in jedem der 12 iCLIP Datensätze nur „top“ Sequenz Tags mit einem Auftreten von 10 oder mehr benutzt. Zusätzlich wurden jeweils die Chromosome chrX und chrXHet ignoriert und für die Wahrscheinlichkeiten wurden nur Punktmutationen von Basen betrachtet und keine Insertionen oder Deletionen. Die Mutationen wurden dabei als unabhängig voneinander betrachtet. Das Auftreten von Mutationen (Abbildung 3.1, Datei D7) war in allen Datensätzen sehr ähnlich. DataSet-4_3 hatte als einziger Datensatz gröbere Abweichungen, was allerdings mit der geringen Größe des Datensatzes begründet werden kann. Um einen guten Durchschnitt zu erhalten, wurden, anstelle der Wahrscheinlichkeiten, direkt die je Datensatz berechnete Anzahl an a zu b Mutationen und die Gesamtzahl an Basen eines Typs zusammengefasst (Datei D2). Mit Hilfe dieser Gesamtdaten wurden dann im Simulationsprogramm (Script S2) die neuen Wahrscheinlichkeiten berechnet.

Es wurden insgesamt 1024 verschiedene „top“ Sequenz Tags in den Datensätzen erkannt, was genau der Gesamtzahl an 4^5 möglichen Kombinationsmöglichkeiten bei einer Länge von 5 Basen entspricht. Während des Mismatch Scripts wurden bereits bei einem Schwellwert von nur 10, insgesamt 5,01% aller „top“ Sequenz

Tags aussortiert. Selbst bei „top“ Sequenz Tags mit einer Anzahl von 10 kann nicht genau festgestellt werden, ob es sich dabei um statistisch relevante PCR Kopien handelt. Dies zeigt, wie problematisch zufällige Sequenz Tags selbst bei Anwendung einer Methode wie iCLIP sein können.

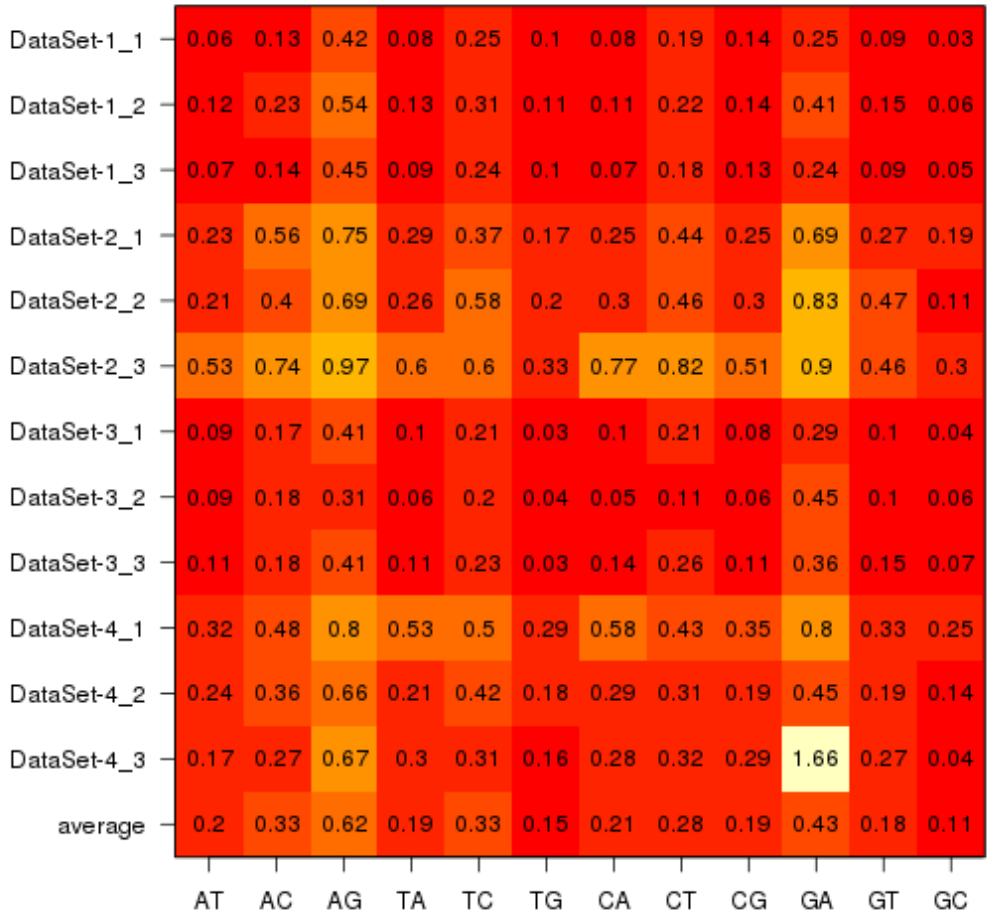

Abbildung 3.1: Heatmap zu den Wahrscheinlichkeiten aller a zu b (ab) Basenmutationen. Die Bedingte Wahrscheinlichkeit gegeben für eine Mutation des jeweiligen Nukleotids ist prozentual zur Gesamtzahl der jeweiligen Ausgangsbase im Datensatz. „Average“ zeigt die im Simulationsprogramm berechneten Gesamtwahrscheinlichkeiten.

3.2 Simulationen

3.2.1 Übersicht

In der nachfolgenden Tabelle (Tabelle 3.1) sind die Ergebnisse aller Simulationsdurchläufe abgebildet. Es wurden für jeden Sequenz Tag/Barcode des Sequenz Tag Sets und der 3 Barcode Sets 1000 mal 200, 500 und 1000 PCR Kopien simuliert. Der Eintrag „failed“ enthält den Anteil an mutierten Sequenz Tags/Barcodes, „corrected“ zeigt wie viele davon erkannt und korrigiert wurden, „defective“ beschreibt den Anteil an nicht erkannten, „overlap“ ist der Anteil an *Overlaps* zu allen Simulationen, PNT bezeichnet den Anteil von negativen Events zu allen Events, RTE ist die Rate von richtig positiven Events und TFT zeigt den Faktor aller negativen Events zu positiven Events. R-5 bezeichnet das Set aus 5 Nukleotide langen zufälligen Sequenz Tags (Datei D1). H2-16-11 steht für ein Set aus binär codierten Hamming Barcodes mit einer Länge von 8 Nukleotiden und somit codiert 16 Bits, wovon 5 zur Kontrolle verwendet werden (Datei D3). H-4-9-5 beschreibt das Set aus quartär codierten Barcodes, welche eine Länge von 9 Nukleotiden haben, wovon 4 zur Kontrolle verwendet werden (Datein M4). L-9-3 steht für das mit Hilfe der Levenshtein Distanz erstellte Barcode Set, wobei die 3 eine Minimaldistanz von 3 zwischen den einzelnen Barcodes im Set angibt (Datei D5).
Bei Betrachtung jedes Sequenz Tag/Barcode Sets auf eine unterschiedliche Anzahl an Simulationen, konnten bei den „failed“ , „corrected“ , „defective“ und „overlap“ Einträgen, keine nennenswerten Unterschiede beobachtet werden. Bei den PNT, RTE und TFT Einträgen hingegen gab es, besonders bei den quartär codierenden Hamming Barcodes und den Levenshtein Distanz Barcodes, erhebliche Unterschiede. In den Nachfolgenden Unterpunkten werden die Ergebnisse des Sequenz Tag Sets und der Einzelnen Barcode Sets im Hinblick auf 200 Simulationen genauer erläutert.

	failed	corrected	defective	overlap	PNT	RTE	TFT
200							
R-5	3,96	0	100	3,96	85,56	14,44	5,92
H2-16-11	6,26	33,85	66,15	$2{,}32E^{-5}$	59,93	13,75	1,5
H4-9-5	7,01	96,79	3,21	$6{,}5E^{-7}$	3,86	68,99	0,04
L-9-3	7,07	96,78	3,22	$4{,}21E^{-7}$	3,89	68,73	0,04
500							
R-5	3,96	0	100	3,96	91,29	8,71	10,49
H2-16-11	6,26	33,84	66,16	$2{,}33E^{-5}$	63,29	8,09	1,72
H4-9-5	7,01	96,79	3,21	$6{,}63E^{-7}$	5,57	47,15	0,06
L-9-3	7,07	96,78	3,22	$4{,}01E^{-7}$	5,62	46,82	0,06
1000							
R-5	3,96	0	100	3,96	93,22	6,78	13,74
H2-16-11	6,26	33,84	66,16	$2{,}35E^{-5}$	65,59	6,07	1,91
H4-9-5	7,01	96,8	3,2	$6{,}7E^{-7}$	8,39	30,92	0,09
L-9-3	7,07	96,78	3,22	$4{,}08E^{-7}$	8,49	30,62	0,09

Tabelle 3.1: Ergebnisse des Sequenz Tag Sets und der einzelnen Barcode Sets mit 200, 500, 1000 Simulationen je Sequenz Tag/Barcode und 1000 Wiederholungen. Alle außer den TFT Einträgen sind als Prozentsatz zu sehen. Die Einträge „failed“ und „overlap“ bezieht sich auf die Anzahl an simulierten Sequenz Tags/Barcodes. Die „corrected“ und „defective“ Einträge beziehen sich auf die Anzahl an mutierten Sequenz Tags/Barcodes.

3.2.2 Zufällige Sequenz Tags

Die zufälligen Sequenz Tags hatten, im Vergleich zu den Barcode Sets, einen geringeren Prozentsatz von insgesamt nur 3,96% mutierten Sequenz Tags. Dies ist wohl auf die geringere Länge von 5 zurückzuführen. Alle fehlerhaften Sequenz Tags waren natürlich schon bei einer einzigen Mutation ein *Overlap* zu einem anderen Sequenz Tag. Diese können in der Auswertung die Ergebnisse erheblich verfälschen, indem Fehlerhafte Sequenz Tags als andere erkannt werden. 85,56% aller Events waren negativ und es gab 5,92 mal soviel, negative wie positive Events. Die Rate an richtig positiven Events lag bei 14,44%.

3.2.3 Binär codierte Barcodes

Bei den 8 Basen langen binär codierten Hamming Barcodes lag die Gesamtzahl an mutierten Barcodes bei 6,26%. Die Anzahl an Mutationen steigt deutlich bei längeren Barcodes. 33,85% dieser fehlerhaften Barcodes konnten jedoch korrigiert werden, wodurch die tatsächliche Zahl an nicht nutzbaren simulierten PCR Duplikationen auf 4,14% sank. An sich hätten binäre Barcodes im Vergleich zu zufälligen Sequenz Tags somit keinen viel besseren Durchsatz an nutzbaren PCR Duplikationen, was sich auch an der etwas geringeren Rate von 13,75% an richtig positiven Events zeigt. Wichtig ist jedoch auch der Anteil an *Overlaps*. Dieser ist mit $2{,}32E^{-5}$% deutlich geringer als der bei zufälligen Sequenz Tags. Der Vorteil im Vergleich zu zufälligen Sequenz Tags ist somit, dass überhaupt erkannt werden kann, ob ein Barcode fehlerhaft ist. Dies sollte in Verbindung mit iCLIP eine deutliche Verbesserung erbringen. Dies zeigt sich auch durch den deutlich geringeren Anteil von insgesamt 59,93% negativen Events. Der Anteil an negativen zu positiven Events hat sich mit 1,5 auch deutlich verbessert. Vorteilhaft ist zudem auch die an sich schnelle Korrektur. Bei einer Anzahl von n erstellten PCR Produkten würde der Algorithmus pro Barcode nur $O(n)$ Zeit brauchen, da er jedes Produkt nur einmal betrachten müsste und sofort die Korrektur durchführen könnte.
Ein großes Problem bei binär codierten Barcodes ist jedoch die Codierung an sich. Hierbei muss beachtet werden, dass Mutationen auf DNA Ebene geschehen, und nicht auf binär Ebene. Binär codierte Barcodes sind nur bedingt Fehler erkennend bzw. korrigierend, da nur ein Fehler im Binärcode korrigiert werden kann. Bei einer „A“ zu 00, „C“ zu 01, „G“ zu 10 und „T“ zu 11 Codierung ist zu beachten, dass auch A zu T, C zu G und umgekehrt mutieren können. Die Distanz bei diesen Mutationen wäre 2 und der Barcode somit nicht korrigierbar (Abbildung 3.2). Dies könnte durch Verwendung einer fließende Codierung umgangen werden. Eine Option wäre es, immer zwei aufeinanderfolgende Bits in einem ein Bit Schrittmuster zu lesen. Die Sequenz 1101001, zum Beispiel, würde somit als 11,10,01,10,00,01 gelesen und in TGCGAC übersetzt werden. Die Redundanz auf DNA Ebene wäre dadurch allerdings ungemein größer. Ein binäres Hamming(16,11) Barcode Set, welches die benötigte Menge von mindestens 1024 verschiedenen Barcodes liefert,

wäre übersetzt 15 Basen lang. Dies würde mehr mutierte Barcodes im Ergebnis liefern und den Nutzen wieder zunichte machen. Hier können auch gleich Quartär codierte Hamming Barcodes verwendet werden, welche bei einer Länge von 9 Basen (für 1024+ verschiedene Barcodes) deutlich weniger anfällig für Mutationen sind. [3]

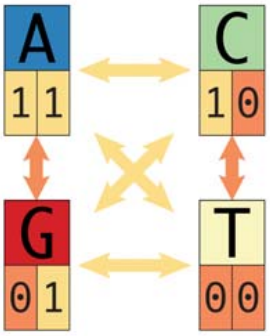

Abbildung 3.2: Abbildung erstellt von Faircloth et al. [2]. Problem bei binär codierten Hamming Barcodes. Mutationen kommen auf DNA Ebene vor und können bei dieser Codierung nicht immer erkannt werden.

3.2.4 Quartär codierte Barcodes

Der Anteil an mutierten Barcodes ist bei den 9 Nukleotid langen quartär codierten Hamming Barcodes, wegen der um 1 Nukleotid längeren Barcodes, erneut gestiegen. Mit 7,01% ist dieser höher als die der zufälligen, und die der binär codierten Barcodes. Dies zeigt deutlich, was für eine große Rolle die Länge der Barcodes spielt. Von diesen mutierten Barcodes konnten jedoch durch die Korrekturmethode 96,79% korrigiert werden. Dies bedeutet, dass insgesamt nur 0,23% aller simulierten PCR Duplikationen nicht verwendet werden konnten. Auch der Anteil an *Overlaps* ist mit $6{,}5E^{-7}$% deutlich niedriger als bei binär codierten und zufälligen Sequenz Tag Sets. Im Hinblick auf Events gibt es ebenso deutliche Verbesserungen. So zeigt die Rate von richtig positiven zu negativen Events mit 68,99% und der Anteil von 3,86% negativen zu allen Events eine deutliche Verringerung an resultierenden ungeeigneten Events. Der Faktor von unbrauchbaren zu brauchbaren Events hat sich mit 0,04 deutlich verbessert. Es existieren somit 25 mal mehr

positive Events.

Die quartär codierten Barcodes zeigen im Vergleich zu den binär codierten Barcodes einige Vorteile. Die Methode für die Korrektur ist im Kern gleich und behält damit bei n PCR Duplikationen pro Barcode die schnelle lineare Korrekturgeschwindigkeit von $O(n)$. Auch lassen sich die Barcodes mit einem einfachen Algorithmus schnell erstellen. Die Codierung ist jedoch tatsächlich fehlerkorrigierend, was sich auch in den genannten Resultaten zeigt [3].

Trotz dieser herausragenden Eigenschaften hat ein quartär codierter Barcode wie auch ein binär codierter Barcode das Problem, in einem festen Fehlerraster zu agieren. Es kann nur eine Mutation korrigiert werden und dies nur, wenn die Mutation an dieser Stelle war. Die quartäre Hamming Methode kann somit weder Insertionen noch Deletionen erkennen.

3.2.5 Levenshtein Distanz Barcodes

Das verwendete Barcode Set konnte, wie auch die quartären Hamming Codes, nur einen Fehler korrigieren und die Barcodes hatten auch die gleiche Länge. Die beiden Methoden haben somit die selben Eigenschaften und können deshalb direkt verglichen werden. Dies zeigte sich auch in den Ergebnissen, welche denen der quartär codierten Hamming Barcodes sehr ähnlich waren. So hatten die mit Hilfe der Levenshtein Distanz erzeugten Barcodes einen Fehlersatz von 7,07%, wovon 96,78% korrigiert werden konnten. Auch lag der Anteil an *Overlaps* nur bei $4{,}21E^{-7}$%, was nur ein sehr geringer Unterschied ist. In Bezug auf Events konnten sich genauso nur sehr geringe Abweichungen feststellen lassen. So lag die Rate an richtig positiven Events bei 68,73% und auch der Faktor von 0,04 mal so vielen negativen wie positiven Events, sowie der Anteil von 3,89% negativen zu allen Events zeigten wie nah beieinander die beiden Ergebnisse liegen.

Die Levenshtein Methode ist im Hinblick auf nur einen Mismatch nicht besser als die quartäre Hamming Methode, zumal bei einem einzigen Mismatch der Vorteil, Insertionen und Deletionen zu erkennen, nicht vorhanden ist. In Bezug auf die Laufzeiten, wird die Distanz Methode sogar schlechter. Hier muss zur Erkennung

und Korrektur der Mutationen für jeden ausgelesenen Barcode n ein Vergleich mit allen m Barcodes aus dem Set erstellt werden. Damit ergibt sich eine Laufzeit von $O(m \cdot n)$ pro Barcode. Auch ist die Barcode Erstellung im Vergleich zu quartär codierten Hamming Barcodes aufwendiger.

Kapitel 4

Diskussion

Es ist deutlich zu sehen, dass zufällige Sequenz Tags einige Probleme hervorrufen. Diese sind besonders gravierend, wenn die Minimaldistanz zwischen einzelnen Sequenz Tags < 3 ist, da sich dann die mutierten Sequenz Tags mit bereits vorhandenen Sequenz Tags überschneiden. Ohne iCLIP wären die PCR Duplikationen somit überhaupt nicht mehr differenzierbar. Allerdings können auch bei iCLIP die PCR Produkte nicht immer erkannt werden, was auch der relativ hohe Ausschuss von 5,01% bei einem Schwellwert von nur 10 zeigt. Es ist darum sehr ratsam Barcodes zu verwenden, welche einzelne Mutationen erkennen können. Der Anteil an richtig erkannten Barcodes verbessert sich dadurch deutlich. Selbst die nicht komplett fehlererkennenden, binär codierten Hamming Barcodes zeigen schon Verbesserungen. Mit der zuvor genannten kleinen Verbesserung im Codierungsschema, ein anderes Leseraster zu benutzen, könnten diese sogar verwendet werden. Der Nutzen würde sich, aufgrund der dadurch erhöhten Redundanz und somit erhöhten Anzahl an Gelegenheiten zur Mutation, voraussichtlich im Vergleich zur aktuellen Codierung nicht sonderlich verbessern. Die quartär codierten Barcodes bringen bei einer geringeren Länge die durch die Veränderung entstandene gleiche Möglichkeit zur Fehlererkennung.

Die quartär codierten Hamming und die mit Hilfe der Levenshtein Distanz erstellten Barcode Sets sind deutlich effektiver. Wenn eine einzelne Fehlerkorrektur ausreichend ist, hat die quartäre Hamming Codierung einen klaren Vorteil. Sie

lässt sich deutlich schneller und einfacher erstellen, sowie korrigieren und hat im direkten Vergleich keinerlei Unterschiede zu den Distanz Barcodes.
Da bei der Arbeit mit DNA eigentlich auch auf Insertionen und Deletionen geachtet werden sollte, sind die Levenshtein Barcodes nicht gänzlich irrelevant. Diese haben bei Betrachtung dieser besonderen Mutationen einen deutlichen Vorteil, benötigen allerdings dafür eine Fehlererkennung von mindestens 2. Dies liegt daran, dass bei der Untersuchung von gleich langen Sequenz Tags immer eine Insertion mit einer Deletion zusammen vorkommen muss. Denn wenn eine Base hinzukommt, wird automatisch eine Base aus dem Leseabschnitt gedrängt. Das Problem dabei ist jedoch die Länge der benötigten Barcodes, da diese, um zwei Mutationen erkennen zu können, eine Mindestdistanz von 5 zwischen allen Barcodes brauchen. Faircloth et al. [2] hat leider kein ausreichend großes Set von mindestens 1024 dieser zwei fehlererkennenden Barcodes veröffentlicht, wodurch die benötigte Barcodelänge nur abgeschätzt werden kann. Da 10 Basen lange Barcodes bei einer Mindestdistanz von 5 nur 164 verschiedene Barcodes liefern, sollte die für einen 1024+ Satz benötigte Länge bei mindestens 11 oder 12 liegen. Diese deutlich längeren Barcodes würden natürlich mehr Möglichkeiten für Mutationen liefern. Der genaue Nutzen müsste anhand von weiteren Studien untersucht werden. Gebraucht würden zunächst Datensätze mit längeren Barcodes bzw. mit längeren Abschnitten, da daraus die Wahrscheinlichkeiten für Insertionen und Deletionen berechnet werden könnten. Auch müssten die benötigten Barcodes neu erstellt und der Simulationsalgorithmus deutlich geändert werden. Eventuell könnten diese trotz der erhöhten Länge und Mutationsanfälligkeit einen besseren Durchsatz liefern.

Kapitel 5

Fazit

Die Simulationen der Sequenz Tags und Barcodes haben gezeigt, dass zufällige Sequenz Tags nicht gut mit *Overlaps* umgehen können. Sie haben dadurch eine hohe Rate an negativen zu positiven Events. Binär codierte Hamming Barcodes verbessern dieses Verhältnis etwas, haben allerdings das Problem, dass sie nicht wirklich fehlerkorrigierend sind. Quartär codierte Hamming und mit Hilfe der Levenshtein Distanz erstellte Barcodes sind im Hinblick auf Events und Fehlererkennung bei einzelnen Mutationen gleich. Der Vorteil der mit der Levenshtein Distanz codierten Barcodes zeigt sich erst bei Betrachtung von mehreren Mutationen im Barcode. Sollte eine solche Untersuchung tatsächlich ein besseres Ergebnis liefern, so darf dabei die deutlich erhöhte Laufzeit bei der Kontrolle nicht vernachlässigt werden. Diese ist sogar noch größer wenn beachtet wird, dass bei der Distanz Berechnung zwischen zwei DNA Strängen einer Länge q, $O(q^2)$ Zeit benötigt wird. Bei m Barcodes und n PCR Duplikationen ergibt sich somit eine Gesamtlaufzeit von $O(m \cdot n \cdot q^2)$. Auch ist zu bedenken, dass bei quartär codierte Hamming Barcodes insgesamt nur 0,23% nicht korrigiert werden konnten. Die Verbesserung im Durchsatz sollte bei Insertionen und Deletionen erkennenden Levenshtein Barcodes darum relativ gering ausfallen.
Zusammenfassend reichen die quartär codierten Barcodes in Verbindung mit iCLIP völlig aus.

Zusatzmaterial

Script S1-S5

Script S1 (mismatches.pl)
Berechnet die Anzahl an Mutationen in den Datensätzen.
Script S2 (simulator.pl)
Hauptscript für die Simulation der Barcodes.
Script S3 (rearrange.pl)
Ordnet die Datensätze für die anschließende Berechnung der Mutationen neu an.
Script S4 (sumrbc.pl)
Fasst die aus den einzelnen Datensätzen erhaltenen zufälligen Sequenz Tags zusammen.
Script S5 (summis.pl)
Summiert die aus den einzelnen Datensätzen erhaltenen Auftreten der Basenmutationen.

Datensatz DS1-DS4

Datensatz DS1 (mismatches)
Enthält die verschiedenen Datenätze nach Bearbeitung durch mismatches.pl.
Datensatz DS2 (mm_barcodes)
Enthält alle erkannten Sequenz Tags.
Datensatz DS3 (mm_statistics)
Enthält die Einstellungen für mismatches.pl, die berechneten Wahrscheinlichkeiten, Anzahl an aussortierten Einträgen und Debugdaten.
Datensatz DS4 (sim_statistics)
Enthält alle Simulationsergebnisse für 200, 500, 1000 Simulationen und 1000 Wiederholungen. Eine Zusammenfassung für jede Simulationseinstellung ist in all_BC_[Simulationen]x1000.txt.

Datei D1-D7

Datei D1 (barcodes_R-5.txt)
Enthält alle, aus den Datensätzen gewonnenen, zufälligen Barcodes für die Simulation.

Datei D2 (mm_summary.txt)
Enthält das anhand der iCLIP Datensätze berechnete Auftreten der einzelnen Basenmutationen.

Datei D3 (barcodes_H2-16-11.txt)
Enthält alle benutzten binär codierten Hamming Barcodes für die Simulation.

Datei D4 (barcodes_H4-9-5.txt)
Enthält alle benutzten quartär codierten Hamming Barcodes für die Simulation.

Datei D5 (barcodes_L-9-3.txt)
Enthält alle mithilfe der Levenshtein Distanz erstellten Barcodes für die Simulation.

Datei D6 (threshold.txt)
Enthält die Schwellwert Daten aller Datensätze.

Datei D7 (percentdata.txt)
Enthält alle Wahrscheinlichkeiten der einzelnen Basenmutationen.

Literaturverzeichnis

[1] König J, Zarnack K, Rot G, Curk T, Kayikci M, Zupan B, Turner DJ, Luscombe NM, Ule J (2010). *iCLIP reveals the function of hnRNP particles in splicing at individual nucleotid resolution.* Nature Structural & Molecular Biology 17(7): 909-916.

[2] Faircloth BC, Glenn TC (2012). *Not All Sequence Tags Are Created Equal: Designing and Validating Sequence Identification Tags Robust to Indels.* PLoS ONE 7(8): e42543. doi:10.1371/journal.pone.0042543.

[3] Bystrykh LV (2012). *Generalized DNA Barcode Design Based on Hamming Codes.* PLoS ONE 7(5): e36852. doi:10.1371/journal.pone.0036852.

[4] Hamady M, Walker JJ, Harris JK, Gold NJ, Knight R (2008). *Error-correcting barcoded primers for pyrosequencing hundreds of samples in multiplex.* Nat Methods 5(3): 235237.

[5] Meyer M, Stenzel U, Myles S, Prüfer K, Hofreiter M (2007). *Targeted high-throughput sequencing of tagged nucleic acid samples.* Nucleic Acids Res 35, 15: e97.

[6] Parameswaran P, Jalili R, Tao L, Shokralla S, Gharizadeh B, Ronaghi M, Fire AZ (2007). *A pyrosequencing-tailored nucleotide barcode design unveils opportunities for large-scale sample multiplexing.* Nucleic Acids Res 35, 19: e130.

[7] Frank, DL (2009). *BARCRAWL and BARTAB: software tools for the design and implementation of barcoded primers for highly multiplexed DNA sequencing.* BMC Bioinformatics 10: 362.

[8] Binladen J, Gilbert MT, Bollback JP, Panitz F, Bendixen C, et al. (2007). *The use of coded PCR primers enables high-throughput sequencing of multiple*

homolog amplification products by 454 parallel sequencing. PLoS One 14; 2(2): e197.

[9] Galan M, Guivier E, Caraux G, Charbonnel N, Cosson JF (2010). *A 454 multiplex sequencing method for rapid and reliable genotyping of highly polymorphic genes in large-scale studies.* BMC Genomics 11; 11: 296.

[10] Smith AM, Heisler LE, St Onge RP, Farias-Hesson E, Wallace IM, et al. (2010). *Highly-multiplexed barcode sequencing: an efficient method for parallel analysis of pooled samples.* Nucleic Acids Res 38, 13: e142.

[11] Hamming RW (1950). *Error Detecting and Error Correcting Codes.* The Bell System Technical Journal 29(2): 147160.

[12] Levenshtein VI (1966). *Binary codes capable of correcting deletions, insertions, and reversals.* Soviet physics- Doklady 10, 8: 707709.