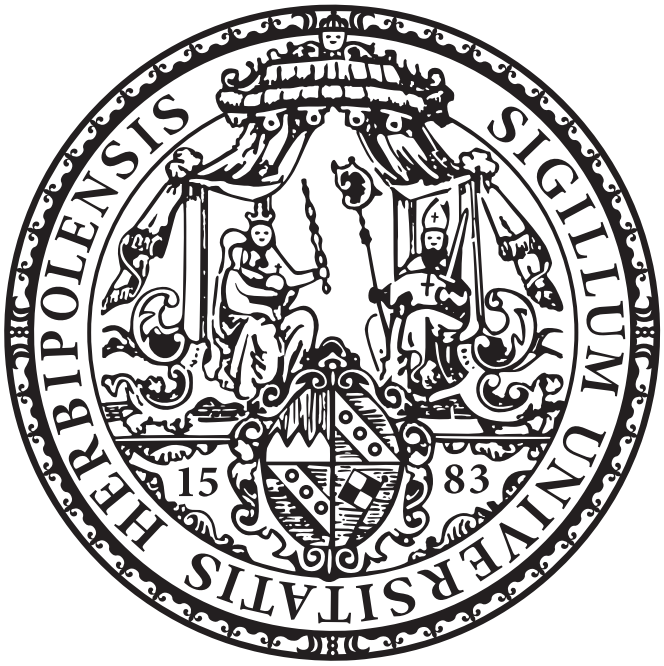
Master Thesis
Seperating the good from the bad... Exploring the genomic landscape of chloroplasts from genomic sequencing datatreduction

Simon Pfaff



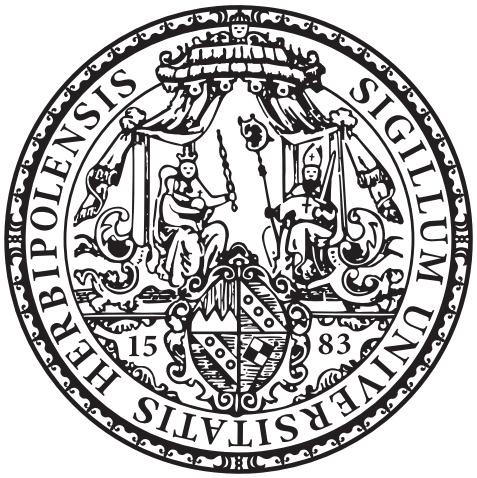
Julius-Maximilians-Universität Würzburg
Fakultät für Biologie

Master Thesis

Simon Pfaff

August 6, 2018

Seperating the good from the bad... Exploring the genomic landscape of chloroplasts from genomic sequencing datatreduction



Julius-Maximilians-Universität Würzburg
Betreuer: Dr. Markus Ankenbrand
Betreuer: Prof. Dr. Jörg Schulz
Betreuer: Dr. Frank Förster
Lehrstuhl für Bioinformatik
Center for Computational and Theoretical Biology

Contents

1 Zusammenfassung / Abstract

1.1 Deutsch

In der Heutigen Big Data Ära, werden immer mehr neue Daten erzeugt. Doch können auch bereits vorhandene Daten durchaus noch Informationen enthalten welche bisher ungenutzt sind. Dank der Open Science sind viele dieser Daten frei verfügbar. In diesem Konkreten Falle werden verschiedene Programme verwendet um in pflanzlichen Genom Daten, Chloroplasten Genome zu suchen. Chloroplasten haben ihre eigene zirkuläre DNA, und wenn beim Sequenzieren diese nicht gefiltert wurde, dann ist sie noch in den Daten vorhanden. So kann man ohne eine neue Sequenzierung machen zu müssen Chloroplasten DNA für Analysen erhalten. Es wurden verschiedene Programme getestet und verglichen, um so viele Chloroplasten wie nur möglich zu bauen und damit Analysen durchzuführen. Es wurde eine vollautomatische Pipeline erstellt mit denen Chloroplasten Genome einfach aus Pflanzen Genomen extrahiert werden können.

1.2 English

In todays Big Data Era, every day new data is created. But this is not necessary to get new informations. In older data oftentimes hides other data, which may not be used till today. Since open science is growing in the last years, many of this data is freely accessible. In this case, plant whole genome data is used. This data oftentimes includes chloroplast genom data, when in the sequencing process nobody removed it by cellnucleus extraction. There were different tools which provided a suit to extract this chloroplast genome data from plant genom data. They were tested, compared and the best were used to calculate alot of chloroplast genomes. This chloroplast genomes got analyzed with different methodes. Also there is now a fully automatic pipeline for chloroplast genome extraction from genomic sequencing data.

2 Einleitung

2.1 Chloroplasten

Chloroplasten sind extrem wichtig, alle Pflanzen besitzen diese und nutzen sie um Energie in Form von ATP durch Photosynthese zu erzeugen[1]. Ihr Genom gilt als hoch konserviert, sowohl in der Gen Orientierung als auch dem Gen Inhalt [2], dieser besteht zwischen 100 und 200 Genen[3]. Das Genom ist in etwa 30 bis 60 Mikrometer lang und besitzt eine Masse von ungefähr 80 - 130 Millionen daltons[4]. Chloroplasten zeigen eine auffällige Genom Struktur, das Genom ist ein einem Plasmid welches sich in drei Teile aufteilt. Dem Large Single Copy, dem Small Singles Copy, welche beide durch zwei Inverted repeats unterbrochen sind (Fig. 1)[5]. Chloroplasten zeigen eine viel geringere Substitutionsrate als in genomischer DNA, diese ist noch einmal signifikant geringer in den Inverted repeat Regionen [6]. Dennoch zeigen sich Einzelnucleotid-Polymorphismen (SNPs)[7]. Zudem gibt es ein seltenen Fällen eine Genwanderung von Genen auf dem IR zum SSC, wodurch ein IR weg fallen kann, sodass solche Chloroplasten nur noch ein IR besitzen [8] . Vor allem in gezüchteten Nutzpflanzen finden sich auch Invertierungen des IR [9]. Durch ihre hohe Konservierung sind Chloroplasten und ihre Gene sehr gut für Barcoding geeignet[10]. Mit diesem Barcoding können Pflanzen und ihre Varianten identifiziert werden. Neue Studien zeigen, dass der komplette Chloroplast selbst als ein Art "Ultra-barcode" verwendet werden könnte, da die Variation in Chloroplasten in einer Spezies doch mehr variiert als angenommen [11].

Die Gene auf einem Chloroplasten lassen sich in verschiedene Klassen einteilen (Fig.

[1] Purves Biologie, Sadava D, Hillis D.M, Heller H.C, Berenbaum M.R, (9. Auflage S. 14)

[2] Raubeson L, Jansen R. (2005). Chloroplast genomes of plants, Plant diversity and evolution: genotypic and phenotypic variation in higher plants. Diversity and Evolution of Plants-Genotypic and Phenotypic Variation in Higher Plants. 3. 10.1079/9780851999043.0045.

[3] Howe CJ. (2016). Chloroplast Genome. In eLS, John Wiley & Sons, 10.1002/9780470015902.a0002016.pub3

[4] Burgess, Jeremy (1989). An introduction to plant cell development. Cambridge: Cambridge university press. p. 62. ISBN 0-521-31611-1.

[5] Shaw J, Lickey EB, Schilling EE, et al. (2007). Comparison of whole chloroplast genome sequences to choose noncoding regions for phylogenetic studies in angiosperms: The tortoise and the hare III. American Journal of Botany. 10.3732/ajb.94.3.275.

[6] Wolfe KH, Li WH, Sharp PM. (1988). Rates of nucleotide substitution vary greatly among plant mitochondrial, chloroplast and nuclear DNA. Proc Natl Acad Sci USA 10.1073/pnas.84.24.9054.

[7] 1001 Genomes Consortium, (2016) 1,135 genomes reveal the global pattern of polymorphism in *Arabidopsis thaliana*. Cell. https://doi.org/10.1016/j.cell.2016.05.063

[8] Jansen RK, Wojciechowski MF, Sanniyasi E, et al. Complete plastid genome sequence of the chickpea (Cicer arietinum) and the phylogenetic distribution of rps12 and clpP intron losses among legumes (Leguminosae). Molecular phylogenetics and evolution. 10.1016/j.ympev.2008.06.013.

[9] Palmer JD, Jansen RK, Michaels HJ, et al. (1988). Chloroplast DNA Variation and Plant Phylogeny. Annals of the Missouri Botanical Garden, 10.2307/2399279

[10] Song Y, Wang S, Ding Y, et al. (2017) Chloroplast genomic resource of Paris for species discrimination. Sci. Rep. 10.1038/s41598-017-02083-7

[11] Kane N, Sveinsson S, Dempewolf H, et al.(2012), Ultra-barcoding in cacao (Theobroma spp.; Malvaceae) using whole chloroplast genomes and nuclear ribosomal DNA. American Journal of Botany, 10.3732/ajb.1100570

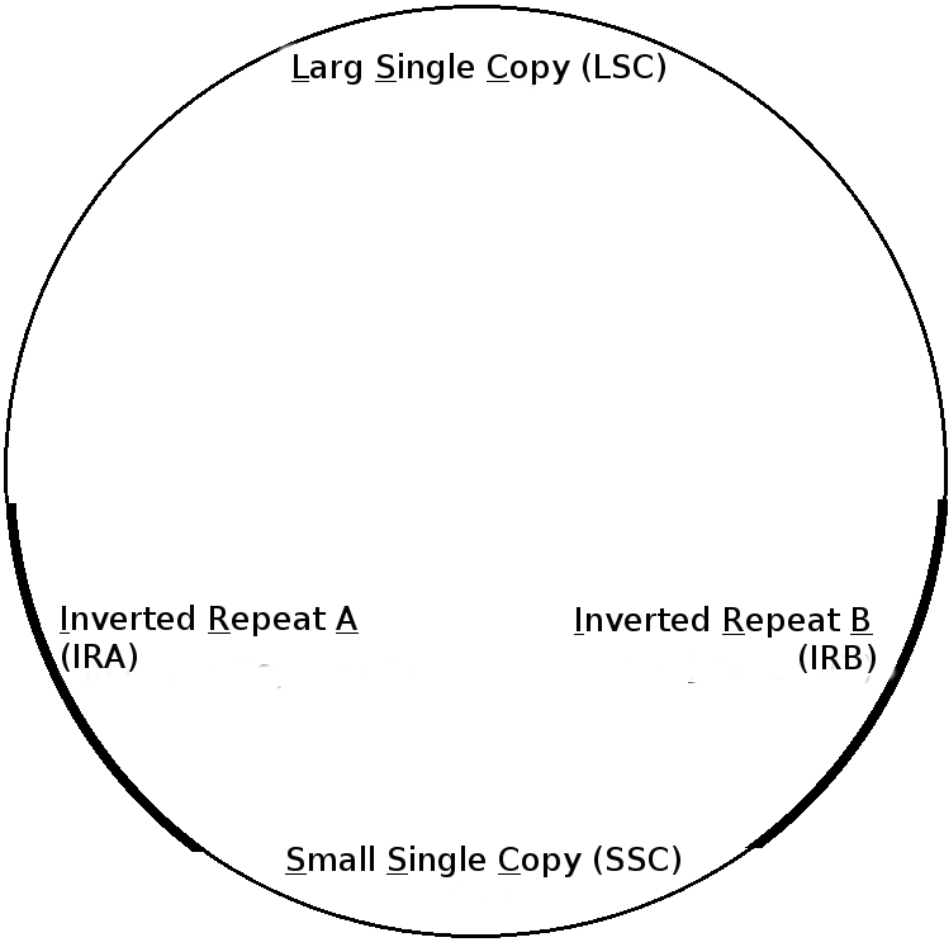


Figure 1: **Chloroplast Genom Einteilung** Der Chloroplast ist unterteilt in Large Single Copy, Small Single Copy und Inverted Repeat, diese unterteilen sich nochma in IRA und IRB. LSC und SSC werden jeweils von den IRs unterbrochen.

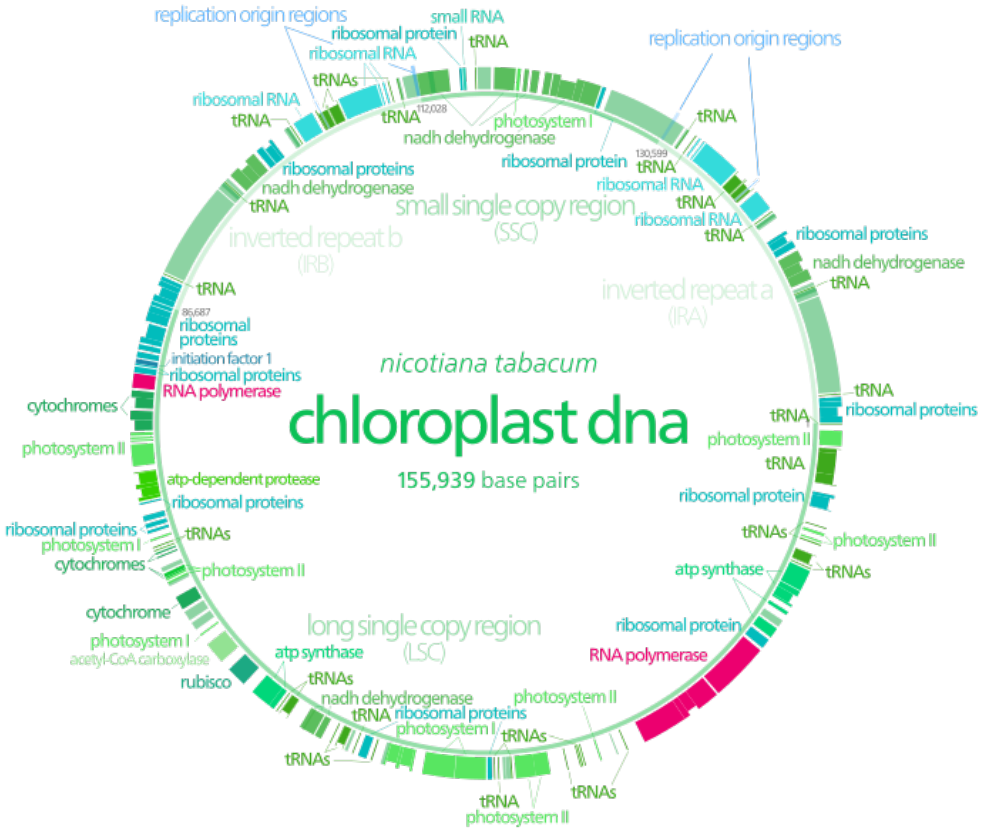


Figure 2: **Chloroplast Genom: Einteilung Gen Klassen** Das Chloroplast Genom der Tabak pflanze, die innere Beschriftung zeigt den - strand, die Äußere den + strand der DNA. Die Kerben visualisieren Introns.Wikipedia unter Wikimedia Commonsen Lizenz`https://en.wikipedia.org/wiki/File:CtDNA.svg`

2). Zunächst gibt es die Gene welche für die Photosynthese wichtig sind, hier unterteilt man nochmal in Photosystem I (psaA, psaB, etc.), Photosystem II (psbA, psbB, etc.), Cytochrome b6f (petA, petB,etc.), ATP Synthese (atpA, atpB, etc.), RuBisCo(rbcL) und NAD(P)H dehydrogenase Gene(ndhA, ndhB, etc.)[12]. Die zweite Klasse beinhaltet Gene welche für den Genetischen Apparat, also Transkription, Translation und Replikation nötig sind, sowie RNA Gene. Hierzu zählen Transfer RNA (trnH,trnK, etc.), ribosomale RNA (rrn16, rrn5, etc.), RNA Polymerasen (rpoA, rpoB, etc.) und ribosomale Gene (rps2, rps3, rpl2, rpl16,etc.). Die dritte und letzte Kategorie beschreibt Gene welche konservierte Open Reading Frames (ORFs) haben und ycfs (Hypotheticalchloroplast open reading frames) genannt werden sowie potentiell kodierende Gene wie matK und cemA[12]. Vor allem stark konservierte Gene wie rbcL und matK und cemA werden häufig für Barcoding oder zum berechnen von phylogenetischen Bäume verwendet.

2.2 Big Data

Die Big Data Ära zeichnet sich vor allem durch die Flut an Daten aus, welche kaum noch zu bewältigen ist. Im biologischen Sinne zeichnet sich diese Flut an Daten vor allem durch genomische Daten aus. Durch sogenannte Hochdurchsatz Methoden in modernen Sequenzierungs Technologien wie PacBio[13] oder Illumina[14] sie verwenden, können sehr viele genomische Daten in kurzer Zeit für vergleichsweise wenig Geld erzeugt werden. Um dieser Daten Herr zu werden sind vor allem Programme die diese Daten auswerten wichtig. Die Anforderung an diese Programme sind unter anderem eine hohe Geschwindigkeit und vor allem eine hohe Automatisierbarkeit. Um solche Programme zu entwickeln sind vor allem Kenntnisse in Informatik und in Biologie notwendig. Informatik ist wichtig um eine sinnvolle Programm Struktur zu entwickeln sowie natürlich das Programmieren an sich, und alles was dazugehört von Bugs fixen bis zur Optimierung der Software. Biologie ist wichtig um die oftmals komplizierten Biologischen Fragestellungen und Sachverhalte zu verstehen, die hinter einem solchen Programm liegen. Nur in der richtigen Kombination können solche Programme gut geschrieben werden. Deswegen wird die Bioinformatik immer wichtiger. Zu dem müssen diese Daten dann noch ausgewertet werden, etwas was die biologische Seite nochmals fordert.

2.3 Open Science

Mit der Flut der ansteigenden Daten wächst in den letzten Jahren auch immer mehr die Akzeptanz zur "Open Science". "Open Science" bezeichnet eine Bewegung welche fordert, dass Wissenschaft und alles was dazugehört, Daten, Programme, Ergebnisse öffentlich und für jedermann zugänglich sein soll. Befürworter dieser Bewegung argumentieren damit, dass Wissen für jedermann zugänglich sein sollte, und nicht nur für Ausgewählte oder gar gegen Bezahlung. So steigere sich unter anderem die Akzeptanz

[12] Ravi V, Khurana JP, Tyagi AK, et al. (2007). An update on chloroplast genome. Plant Systematics and Evolution. 10.1007/s00606-007-0608-0.

[13] https://www.pacb.com/

[14] https://www.illumina.com/

der Wissenschaft als auch deren Glaubwürdigkeit. Da die Ergebnisse von jedem nachvollziehbar veröffentlicht werden müssen, mit allen Rohdaten und Vorgehensweisen. Dies sei der eigentliche Gedanke hinter der Wissenschaft, sie solle jedem zugänglich sein! Diese Bewegung findet vor allem bei jung Wissenschaftler aber auch bei Älteren immer mehr Anklang. Mittlerweile gibt es mehrere Lizenz Modelle die unter Open Science laufen. Diese Regeln wie die Daten verwendet werden dürfen oder müssen. Dies reicht von Freigeben der Daten und jeglichem Verwendungszweck bis hin zum Zwang, dass alles was mit diesen Daten oder auch Programmen veröffentlicht wird wieder unter der gleichen Open Source Lizenz zu publizieren ist. Alle hier verwendeten Programme und Daten sind unter Open Source Lizenzen veröffentlicht, sonnst wäre diese Arbeit gar nicht möglich. Deswegen werden alle Ergebnisse wiederum öffentlich verwendbar sein. Denn so sollte Wissenschaft sein!

2.4 Daten in Daten

Bei den heutzutage geringen Kosten, Daten vor allem genomische Daten, zu erzeugen ist es nicht verwunderlich dass immer neue Daten generiert werden. Dennoch steckt in bereits erhobenen Daten meist mehr Information als zunächst verwendet. In genomischen Daten zum Beispiel, hier findet sich meistens Daten von Organellen, wie Mitochondrien oder Chloroplasten, welche ihre eigene DNA besitzen. Diese sind dort zu finden da vor einer Sequenzierung häufig keine Kern Extraktion durchgeführt wird, da diese mehr Zeit und Geld kosten würde. Diese Organellen DNA können mit bestimmten Programmen gefiltert werden, hierfür wurde unter anderem der chloroExtractor programmiert. Dieser kann in genomischen Pflanzen Daten Chloroplasten DNA finden und diese verwenden um einen vollständigen Chloroplasten zu bauen. Hiermit müssen somit keine neuen Sequenzierungen für Chloroplasten mehr durchgeführt werden, wenn man an Chloroplasten forschen möchte.

2.5 Verwendete Programme und ihre Ansätze

Es gibt verschiedene Ansätze um Chloroplasten Genome bzw. ihre DNA aus genomischen Pflanzen Daten zu extrahieren. Die wohl einfachste Möglichkeit ist ein Referenz basiertes Mapping der Daten auf einen Referenz Chloroplasten. Hierzu muss lediglich ein nah verwandter Chloroplast als Referenz benutzt werden. So können die Reads, welche auf diese Referenz passen genommen werden und assembliert werden, mit der gleichen Referenz. Dies funktioniert allerdings nur wenn man eine passende Referenz benutzt, diese sollte von der gleichen Spezies oder zumindest einer Nah verwandten Spezies stammen. Ein anderer Ansatz besteht darin den Chloroplasten de novo zu assemblieren, also ohne Referenz. Um diesen Ansatz zu benutzen müssen aber zunächst die Reads mit Chloroplasten Genom aus den Daten gezogen werden. Hier gibt es wiederum verschiedene Möglichkeiten. Eine Möglichkeit ist es die Reads gegen eine Datenbank von Chloroplasten Genen zu blasten, hierzu muss entweder eine Datenbank von Chloroplasten Genen gestellt werden oder der Benutzer muss eine Pseudo-Referenz einen sogenannten Seed angeben. Ein Seed, was von einigen Basenpaaren bis zu einem kompletten Chloroplasten

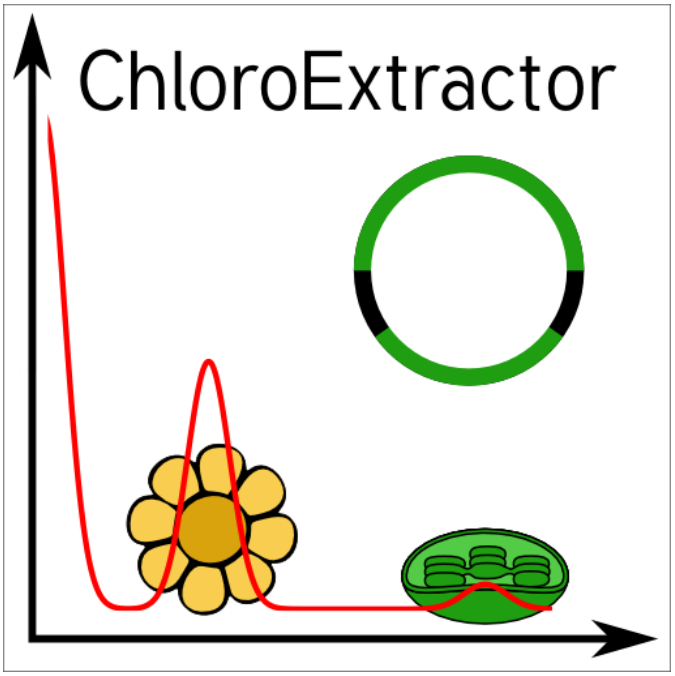


Figure 3: **chloroExtractor Logo** Das Logo des chloroExtractors zeigt die Verteilung der Genomischen Daten in einem Kmer Plot. Der erste Peak zeigt die Kmers Pflanzengenom, der zweite kleinere Peak zeigt die Kmers mit Chloroplasten Genom.

reichen kann, kann auch eingesetzt werden um durch ein reines Mapping Reads zu finden. Bei kleinen Seeds wird dieser häufig durch gefundene Reads erweitert und eine Liste von Seed erstellt. Auch hier muss aber sichergestellt werden, dass der Seed in den Chloroplasten Daten vorhanden ist. Um für alle Programme, welche einen Seed oder eine Referenz benötigen Chancengleichheit herzustellen wurde hier überall die gleiche Datei verwendet. Diese Datei benutzt der chloroExtractor, hier handelt es sich um einige Chloroplasten Gene. Liefert ein Programm seine eigenen Referenzen mit, wie es der chloroExtractor tut, wurden diese nicht geändert. Da diese als Standardparameter gelten, welche nicht verändert werden sollten. Von diesen Methoden gibt es auch Abwandlungen, wie z.b. das scannen der Daten durch Kmers, hier werden die Daten in verschiedene Kmers zerteilt, durch plotten dieser Kmers können an spezifischen Stellen überrepräsentierte Kmers gefunden werden, diese überrepräsentierten Kmer spiegeln häufig Plastome wieder. Diese sind unter anderem Chloroplasten aber auch Mitochondrien, sie besitzen ihre eigene DNA und kommen im Schnitt häufiger vor als DNA welche im Zellkern zu finden ist. Da der chloroExtractor einen Kmer basierten Ansatz benutzt ist ein solches Kmer Diagramm in dessen Logo zu finden(Fig. 3). Abgesehen von den Ansätzen der Programme gibt es zwei verschiedene Arten von Programmen per se, die einen benutzen bereits vorhandene Programme wie Assambler, Mapper oder Kmer-counter. Diese Bauen eine Pipeline um diese Programme, sodass diese in der richtigen Reihenfolge mit den richtigen Parame-

tern mit nur einem Befehl gesteuert werden können. Der Vorteil ist, solche Programme sind einfacher zu warten da sie meist kleiner sind als Programme die dies nicht tun und einfacher zu Programmieren. Allerdings sind sie von diesen drittanbieter Programmen abhängig und es können Probleme auftreten wenn diese Änderungen bzw. Updates ausgeben, weswegen meist die kompatiblen Versionen angegeben werden. Ein weiterer Nachteil, der Benutzer muss häufig weitere Programme, sogenannte Abhängigkeiten installieren bevor er das eigentliche Programm nutzen kann. Die andere Möglichkeit ist es die komplette Maschinerie selbst zu Programmieren, dies ist sehr aufwendig und bedeutet viel Wartungsarbeit. Vorteil hier ist das keine anderen Abhängigkeiten benötigt werden außer ein System welches das Programm verwenden kann. In dieser Arbeit wurden verschiedene Typen von Programmen verwendet. Es wurden von allen Programmen die jeweils neusten Versionen benutzt, und wenn es zu großen Änderungen wie Bug-fixes kam auf die neuere Version gewechselt, um das bestmögliche Ergebnis für die Daten zu erhalten.

2.5.1 chloroExtractor

Der chloroExtractor (Versionen: 1.0.2, 1.0.3, 1.0.4) [15, 16] ist ein Programm welches durch eine Kombination aus Kmer Analyse und Mapping auf bekannte Chloroplasten Gene, Reads von Chloroplasten aus Pflanzen Sequenzierungs Daten extrahiert. Es wurde 2018 vom chloroExtractorTeam veröffentlicht [15] und besteht hauptsächlich aus Perl und R Code. Es verwendet ein Pipeline Programm (PipeWrap.pm) um den richtigen Ablauf zu steuern. Dieses Pipeline Tool wird durch eine Konfigurationsdatei gesteuert, sodass ein Benutzer einfach neue Schritte einfügen könnte. Auch können hiermit einfach über eine Datei, Parameter gesteuert werden welche dann in allen verwendeten Programmen gleich sind. Es könnt so auch einzelnen Programmen speziellen Input mitgegeben werden. Auch verfügt der chloroExtractor dank PipeWrap über ein Checkpoint System. Bricht der Ablauf des Programms ab, kann er am genau diesem Punkt wieder gestartet werden ohne das Programm von neu starten zu müssen. Zunächst verwendet der chloroExtractor Jellyfish [17] um aus den Rohdaten Kmere zu bauen, diese werden über ein R Skript skaliert und gleichzeitig auf Chloroplasten Gene, mit Bowtie2 [18] gemappt (Fig. 4). Die reads welche gemapt haben und die richtigen Kmere werden mit SPAdes[19] anschließend assembliert, SPAdes arbeitet de novo und benötigt keine Referenz. SPAdes verwendet eine De Brujin-Graphen Methode um die Reads richtig zusammen zu fügen, diese werden dann durch ein Perl Skript (fcg.pl) zu einem zirkulären Chloroplasten zusammengebaut.

[15] Ankenbrand MJ, Pfaff S, Förster F, et al., (2018). chloroExtractor: extraction and assembly of the chloroplast genome from whole genome shotgun data. Journal of Open Source Software, `https://doi.org/10.21105/joss.00464`

[16] `https://github.com/chloroExtractorTeam/chloroExtractor`

[17] Marcais G, Kingsford C.(2011), A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics `10.1093/bioinformatics/btr011`

[18] Langmead B, Salzberg S. (2012), Fast gapped-read alignment with Bowtie 2. Nature Methods, `9:357-359`.

[19] Bankevich A, Nurk S, Antipov D, et al. (2012), SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing, Journal of Computational Biology, `10.1089/cmb.2012.0021`

Dieses Skript überprüft gleichzeitig mit BLAST+[20] ob es sich bei den ausgegebenen Reads wirklich um Chloroplasten handelt. Falls es dazu kommt das SPAdes den Chloroplasten nicht komplett zusammenbauen kann, dann gibt das fcg.pl Skript die Contigs, welches für den Chloroplasten verwendet werden aus. Hier gibt es verschiedene Fälle. Kann nur die Zirkularität des Chloroplasten nicht aufgelöst werden gibt der chloroExtractor LSC, SSC und IR aus. Sind gar keine Verbindungen der Contigs möglich gibt das fcg.pl Skript jene Contigs aus die einen BLAST+ Treffer besitzen und somit ein teil des Chloroplasten sind. Es wurden drei Verschiedene Versionen des chloroExtractors verwendet, dies brachten unter anderem Bug fixes welche das Programm zum Absturz brachten. Aber auch Verbesserungen am fcg.pl Skript.

2.5.2 fast-plast

Fast-plast (Version: 1.2.8) [21] ist ein weiteres Programm, welches verwendet wird um Chloroplasten DNA zu finden. Es ist in Perl und in C++ programmiert und verwendet auch SPAdes, aber zusätzlich Bowtie1 sowie Bowtie2. Auch hier wird Blast+ verwendet um die richtigen Reads zu finden.

2.5.3 NOVOPlasty

Im Gegensatz zu den anderen verwendeten Programmen, benutzt NOVOPlasty (Versionen: 2.6.8. 2.6.9, 2.7.0)[22, 23] keine drittanbieter Programme. Es benötigt somit keine Abhängigkeiten von anderen Programmen und ist komplett in Perl programmiert. NOVOPlasty benutzt sogenannte Seeds um Chloroplasten DNA zu finden, dies können einzelne Chloroplasten Gene sein, aber auch ein kompletter Chloroplast. Die Verschiedenen Verwendeten Versionen versprachen Bug fixes sowie neue Features.

2.5.4 Org.ASM

Org.ASM (Version: 1.0.00-alpha11) [24] ist ein Programm hauptsächlich geschrieben in Python. Es versucht überrepräsentierte Sequenzen zu finden und diese zu assemblieren[25]. Mit Hilfe eines Seeds versucht er diese Sequenzen zu finden.Chloroplasten und andere Organellen wie Mitochondrien sind in Zellen überrepräsentiert, vor allem wenn man eine geringe Coverage über das Pflanzen Genom hat, somit sind diese detektierbar[26].

[20] Camacho C, Coulouris G, Avagyan V, et al. (2009), Selecting control genes for RT-QPCR using public microarray data, BMC Bioinformatics `https://doi.org/10.1186/1471-2105-10-42`

[21] `https://github.com/mrmckain/Fast-Plast`

[22] `https://github.com/ndierckx/NOVOPlasty`

[23] Dierckxsens N, Mardulyn P , Smits G. (2016), NOVOPlasty: De novo assembly of organelle genomes from whole genome data. Nucleic Acids Research, `10.1093/nar/gkw955`

[24] `https://pythonhosted.org/ORG.asm/`

[25] `https://git.metabarcoding.org/org-asm/org-asm/wikis/home`

[26] `https://pythonhosted.org/ORG.asm/algorithms.html`

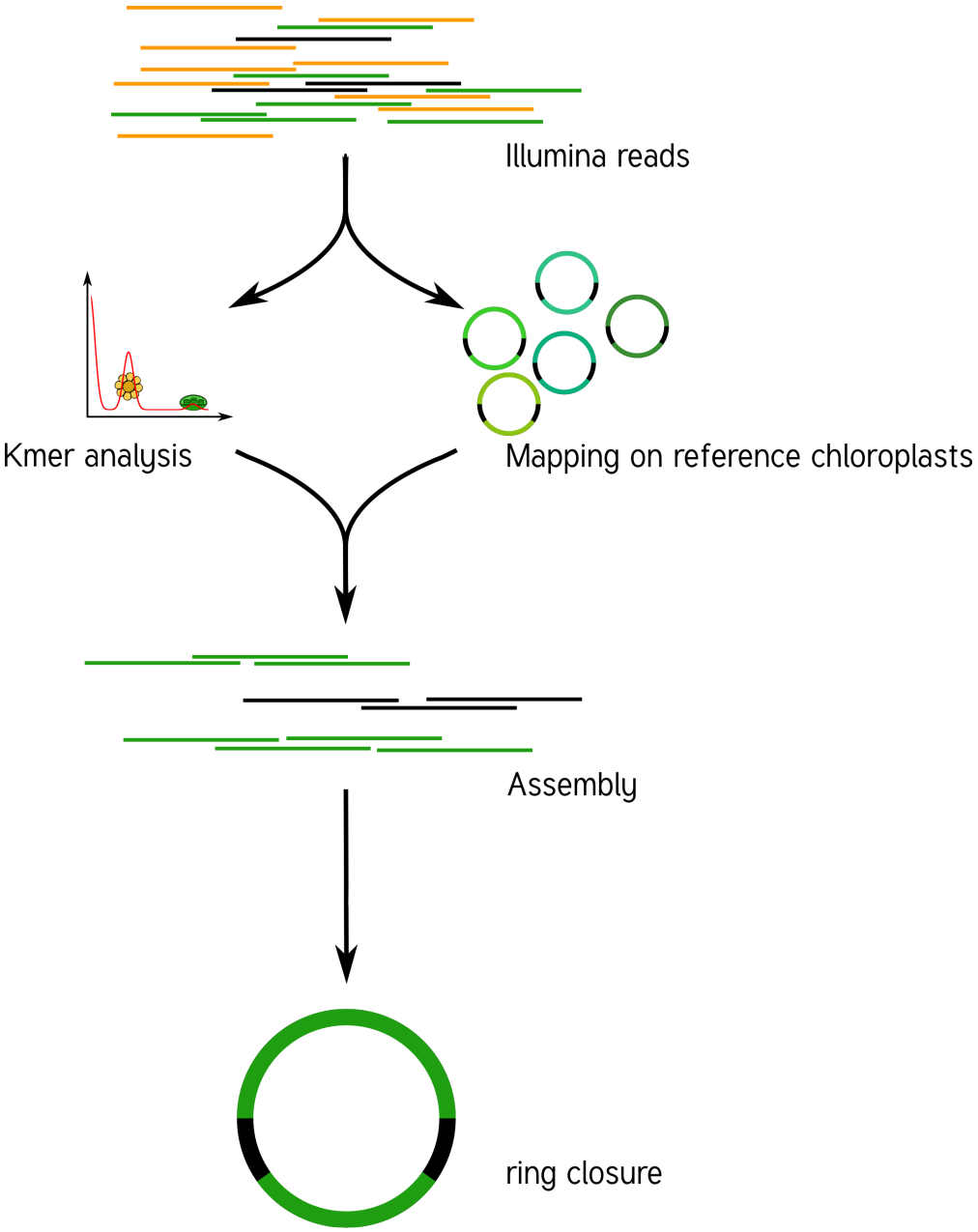


Figure 4: **Ablauf des chloroExtractors** Eine Kombination aus Kmer Analyse und Mapping auf bekannte Chloroplasten rekrutieren Chloroplasten reads um diese anschließend zu Assemblieren um anschließend einen Ringschluss herbeizuführen. (Ankenbrand et al., (2018).)

2.5.5 GetOrganelle

GetOrganelle (Versionen: 1.9.82, 1.0.1, 1.0.3)[27, 28] verwendet zum lokalisieren der Chloroplasten Reads ähnlich wie andere Programme Bowtie2 [18] und Blast+, nur muss hier eine Referenz mitgegeben werden. Diese wird nur hierfür verwendet, das assemblieren hingegen geschieht de novo mit SPAdes. Wie auch beim chloroExtractor wird hier der fastg-Graph verwendet um den Chloroplasten zu finden, aber dies muss in falle des GetOrganelle per Hand, mit Hilfe des Programms Bandage vollzogen werden. Wie bereits erwähnt nutzt der chloroExtractor ein Perl Skript welchen diesen händischen Schritt automatisiert.(Fig. 5) Vom GetOrganelle wurden drei Versionen verwendet. Zunächst 1.9.82, diese wurde geändert zu 1.0.1 (github commit: b390260 vom 31. März 2018) und 1.0.3. In den Verschiedenen Versionen gab es diverse Bug Fixes, sowie kleine Features. Zudem wurde das Programm GetOrganelle mit 1.0.1 in einer Wissenschaftlichen Arbeit veröffentlicht [28].

2.5.6 IOGA

Der Iterative Organellar Genome Assambly, kurz IOGA (Keine Versionsnummer vergeben, github commit: c460ea9 vom 10. Sep. 2016)[29, 30] verwendet BBmap [31] für das filtern und trimmen der reads, um anschließend mit SOAPdenovo2 [32] und SPAdes [19] die Reads zu assemblieren. Auch dieses Programm benötigt eine Referenz. Der IOGA ist in Python geschrieben.

2.6 Interesse an Chloroplasten, was tun damit mit diesen Daten?

Mit der steigenden Anzahl an frei erhältlichen Chloroplasten Genomen, welche gegen Ende 2016 erstmals die 1000 Genome überschritten hat[33], können immer mehr Versuche mit vielen Chloroplasten durchgeführt werden. So ist immer noch nicht geklärt wie genau die Replikation von Plastid Genomen wie von Chloroplasten wirklich funktioniert. Wie werden Mutationen im Inverted Repeat repariert oder bei der Replikation auf beide IRs übernommen? Da SNPs im IR immer auf beiden gefunden werden. Welche Mutationen treten am häufigsten auf und wie sind diese evtl. an die Struktur des Genoms

[27] `https://github.com/Kinggerm/GetOrganelle`

[28] Jin J, Yu W, Yang J, Song Y, et al. (2018), GetOrganelle: a simple and fast pipeline for de novo assembly of a complete circular chloroplast genome using genome skimming data. bioRxiv, `http://doi.org/10.1101/256479`

[29] `https://github.com/holmrenser/IOGA`

[30] Bakker FT, Lei D, et al. (2015), Herbarium genomics: plastome sequence assembly from a range of herbarium specimens using an Iterative Organelle Genome Assembly pipeline, Biol. J. Linnean Soc. `https://doi.org/10.1111/bij.12642`

[31] `https://jgi.doe.gov/data-and-tools/bbtools/`

[32] Luo R, Liu B, Xie Y, et al. (2012), SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. GigaScience. `10.1186/2047-217X-1-18`.

[33] Tonti-Filippini J, Nevill PG, Dixon K, et al. (2017), What can we do with 1000 plastid genomes?. Plant J, `10.1111/tpj.13491`

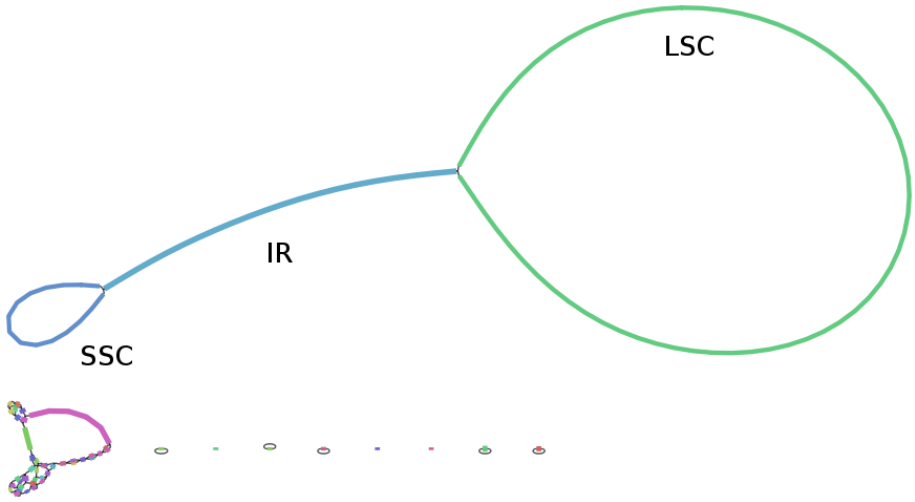


Figure 5: **Bandage - Fastg Visualisierung** Die Visualisierung einer fastg Datei, der eigentlich zirkuläre Chloroplast zeigt sich in einer Form inder SSC (Blau) und LSC (Grün) durch eine Kette welche den IR (Türkis) darstellt verbunden sind. Diese Form wird im fcg.pl Skript des chloroExtractors aufgelöst, wobei beim GetOrganelle diese Sturktur per Hand gefunden werden muss.

gekoppelt [34]? Auch ist immer noch nicht exakt verstanden wie Chloroplasten vererbt werden, es wird zwar angenommen das diese ähnlich wie Mitochondrien maternal vererbt werden doch gibt es bei Pflanzen auch viele Arten die biparental oder uniparental Chloroplasten vererben[35]. Die in den letzten Jahren stark steigende Anzahl an Chloroplasten Genomen gibt diesen Fragestellungen immer mehr neue Rohdaten die diese Probleme evtl. lösen können. Auch Probleme die nur mit kleinen Änderungen im Chloroplasten Genom zu tun haben (SNPs) können so auf den Grund gegangen werden, oder auch die Adaption von verschiedenen Chloroplasten Genen in das Pflanzengenom und der daraus folgenden Änderung im Photosynthese Systems[36]. Auch kann ohne große Änderung an der kodierenden Sequenz, alleine durch Änderung an Transkriptionsfaktoren oder deren Level viel Einfluss auf solche Systeme genommen werden, welche natürlich auch mit dem Chloroplasten zusammenhängen[33]. Wie bereits erwähnt eigenen sich Chloroplasten gut als Barcode Marker. Auch hier können Fortschritte mit mehr Daten erlangt werden. Zudem können mit vielen Chloroplasten Daten sehr gut Phylogenetische Bäume berechnet werden[37]. Dies sind alles Beispiele wie zwischen vielen Spezies mit Hilfe von Chloroplasten Forschung betrieben werden kann. Aber auch innerhalb einer Spezies tauchen Variabilitäten auf, und dies konnte nur mit vielen verschiedenen Chloroplasten der gleichen Spezies herausgefunden werden. So wurden beim 1001 Genom Projekt mehrere Tausend SNPs auf *A.thaliana* Chloroplasten gecalled[38, 7]. Doch können nicht nur viele Chloroplasten Probleme lösen, schon einzelne neue Chloroplasten können sehr aufschlussreich und informativ sein. So wurde die Idee des chloroExtractors z.B. nur aus dem Grund entworfen einen Chloroplasten aus dem Dionaea muscipula (Venusfliegenfalle) Genom zu extrahieren, um diesen separat zu haben, um das Genom leichter zu assemblieren und annotieren. Denn es kann durchaus vorkommen dass bei neuen Genomen, welche de novo assembliert werden müssen, Verunreinigungen durch Chloroplasten auftreten können. Denn ~5 - 20% der kompletten DNA wird von Plastiden-DNA ausgemacht, je nach Spezies und Gewebe[33].

2.7 Aufgaben in der Master Thesis

Die Aufgaben dieser Thesis ist grob in drei Teile eingeteilt. Zunächst sollen die verschiedenen Programme, der chloroExtractor [15, 16], fast-plast[21], IOGA[29, 30], GetOrganelle[27, 28], Org.ASM [24]und NOVOPlasty[22, 23] verglichen werden und herausgefunden werden welche das oder die besten Programme sind um damit so viele Chloroplasten Genome zu erzeu-

[34]Massouh A, Schubert J, Yaneva-Roder L, et al. (2016), Spontaneous Chloroplast Mutants Mostly Occur by Replication Slippage and Show a Biased Pattern in the Plastome of Oenothera. The Plant Cell. 10.1105/tpc.15.00879.

[35]Greiner S, Sobanski J, Bock R. (2015), Why are most organelle genomes transmitted maternally? Bioessays. 10.1002/bies.201400110.

[36]Wicke S, Schneeweiss GM, dePamphilis CW, et al. (2011) The evolution of the plastid chromosome in land plants: gene content, gene order, gene function. Plant Molecular Biology. 10.1007/s11103-011-9762-4.

[37]Chase MW, Fay MF. (2001) Ancient flowering plants: DNA sequences and angiosperm classification. Genome Biology. https://doi.org/10.1186/gb-2001-2-4-reviews1012

[38]http://1001genomes.org/

gen wie möglich. Hier soll vor allem darauf geachtet werden dass die Programme Automatisierbar sind um einen hohen Durchsatz zu haben. Zudem sollen die Programme Ressourcen schonend arbeiten. Der zweite Teil ist das Produzieren von Chloroplasten Genomen, hierzu werden die Pflanzen Genome des 1001 Genom Projektes verwendet. Nach internen Besprechungen und ersten Tests des chloroExtractors, wird angenommen das ca. 10 - 20% der Datensätze einen kompletten Zirkulären Chloroplasten erbringen könnten. Dies hängt von mehreren Variablen ab. Zunächst wie viel Chloroplasten DNA ist in den Daten vorhanden, dies unterscheidet sich je nachdem welches Gewebe verwendet wurde zum Sequenzieren. Hier haben Wurzeln weniger Chloroplasten als Blüten oder Blätter. Auch hängt es davon ab wie "gut" die Daten sind, generell gilt je größer die Reads desto besser zu assemblieren, auch zu beachten sind insert Size und Anzahl der Reads. Auf den so Produzierten Chloroplasten sollen verschiedene wissenschaftliche Analysen durchgeführt werden, so zum Beispiel eine Varianz Analyse sowie eine Genomweite Assoziationsstudie, kurz GWAS [39]. Eine GWAS versucht bestimmte Traits, also Eigenschaften mit Genomischen Varianten zu assoziieren, um anschließend eine Aussage darüber treffen zu können ob diese Variante einen Einfluss auf diese Eigenschaft hat oder nicht. Hierzu werden die einzelnen Chromosomen einzeln oder als komplettes Genom angesehen, je nach Ansatz oder Fragestellung. Auch sollte eine Struktur Varianz Analyse durchgeführt werden. Zudem könnten diese Daten benutzt werden um Chloroplasten besser als Genetische Marker zu benutzen. Der dritte Teil ist das Suchen nach bisher noch nicht dokumentierten Chloroplasten Genomen, hierzu sollen Daten verwendet werden welche noch keinen Eintrag in der Chloroplasten Datenbank haben.

3 Material / Methoden

3.1 Evaluation der Programme

Um die oben genannten Programme zu vergleichen habe ich mir verschiedene Ansätze überlegt. Um zunächst zu testen wie genau die Programme funktionieren und ob diese überhaupt funktionieren, wurden sie auf dem Testset SRR5216995 (*Arabidopsis thaliana*: Col-0) mit eine Millionen reads getestet, dieser ist frei zugänglich bei NCBI und dient als Testset beim chloroExtractor[15]. Um eine Automatisierung zu erhalten musste für jedes Programm ein Dockercontainer[40](s. Anhang Tab.9) gebaut werden, falls dieser nicht schon einer vorhanden war, letzteres traf nur für den chloroExtractor zu. Diese Dockercontainer sind auf Dockerhub[41] frei zur Verfügung[42]. Um das Ziel zu erreichen so viele Chloroplasten wie möglich zu extrahieren, musste eine Automatisierungslösung für alle Programme erstellt werden, damit keine evtl. Manuelle Schritte oder Auswertungen der zeitbestimmende Schritt sind. Um dies zu erreichen mussten zusätzlich einige Bash und Perl Skripte (s. Anhang) geschrieben werden, welche eine volle Automatisierung

[39] Korte A, Farlow A. (2013), The advantages and limitations of trait analysis with GWAS: a review. Plant Methods. 10.1186/1746-4811-9-29.

[40] `https://www.docker.com/`

[41] `https://hub.docker.com/`

[42] `https://hub.docker.com/u/chloroextractorteam/`

ermöglichen.

3.1.1 Testdaten

Es wurden verschiedene Größe von Dateien verwendet. So sind dies alles Illumina short Read Daten, doch unterscheiden sich diese in Readlänge, Insertsize und Anzahl der Reads.

Simulierte Daten Um zu Testen wie gut die verschiedenen Programme mit unterschiedlichen Anteilen von Chloroplasten DNA in Genom Daten zurechtkommen, wurden drei verschiedene Testdatensätze simuliert(Genom : Chloroplast - 1:10, 1:100, 1:1000). Mit diesen sollte auch getestet werden ob die Programme mit viel oder wenig Chloroplasten DNA Anteil zurecht kommen oder einen dieser Fälle bevorzugen. Um diese Daten vorzubereiten wurden von *Arabidopsis thaliana* (TAIR10 [43]) die jeweiligen Chromosomen wie auch die Daten des Chloroplasten von NCBI heruntergeladen. Diese wurden in den jeweiligen Verhältnissen zusammen kopiert. Diese Testdatensätze wurden mit ART[44, 45](Version: 2.5.8) erzeugt. ART wird dazu verwendet Short-reads zu erzeugen, ART kann keine zirkulären Daten wie Chloroplasten erzeugen, deswegen wurden diese als lineare Sequenzen verwendet mit der Abfolge LSC-IRB-SSC-IRA, mit einem Overlap zwischen IRA und LSC: (IRA)-LCS-IRB-SSC-IRA. Mitochondrien DNA wurde nicht mit simuliert, da diese zu Problemen führen könnte wenn diese aufgrund ihrer ähnlichen Häufigkeit und einigen gleichen Genen als Chloroplasten DNA identifiziert werden. Um die verschiedenen Verhältnisse von Genom und Chloroplasten zu bekommen wurden die Chloroplasten Daten einfach vervielfältigt und anschließend zusammen kopiert. Hiernach wurden sie mit folgenden ART Kommandos als short Reads simuliert. Hiernach wurden die Daten gemischt, da es zu Problemen kommt wenn diese Daten sortiert sind. Für diese Daten wurden 150 Basen paare Reads simuliert, und eine Coverage der Daten welche 100x beträgt. Für die Tests wurden eine Millionen Reads pro Datei benutzt, da diese genug Chloroplasten DNA enthalten sollten.
'art_illumina [options] -i <INPUT_SEQ_FILE> -l <READ_LEN> -f <FOLD_COVERAGE> -o <OUTPUT_FILE_PREFIX> -m <MEAN_FRAG_LEN> -s <STD_DE>'
'1:10 : ./art_illumina -p -i sequence-arabidopsis-thaliana-kern-chl-1zu10.fa -l 150 -f 100 -o a_thaliana_1_10_sim -m 500 -s 150'
'1:100 : ./art_illumina -p -i sequence-arabidopsis-thaliana-kern-chl-1zu100.fa -l 150 -f 100 -o a_thaliana_1_100_sim -m 500 -s 150'
'1:1000 : ./art_illumina -p -i sequence-arabidopsis-thaliana-kern-chl-1zu1000.fa -l 150 -f 100 -o a_thaliana_1_1000_sim -m 500 -s 150'

1001 Genom Projekt Um einen ersten Eindruck über die Programme und deren Erfolgs rate zu bekommen wurden parallel zu den Tests mit simulierten Daten, die ersten Tests mit realen Datensätzen vorgenommen. Hierzu wurden Daten aus dem 1001 Genom

[43] https://www.ncbi.nlm.nih.gov/assembly/GCF_000001735.3/
[44] Weichun H, Leping L, Jason RM, Gabor TM. (2015), ART: a next-generation sequencing read simulator, Bioinformatics, https://doi.org/10.1093/bioinformatics/btr708
[45] https://www.niehs.nih.gov/research/resources/software/biostatistics/art/index.cfm

Projekt[38] verwendet, dies sind alles Daten von *Arabidopsis thaliana*. Es wurden 11 Datensätze (SRR1945435 - SRR1945445) verwendet. Diese sind alle frei verfügbar und wurden von NCBI heruntergeladen. Es wurden jeweils zwei Millionen Reads pro Datei gezogen, mit 150 Basenpaaren pro Read.

GetOrganelle-Paper preprint Um zu weitere Testdaten zu ermitteln und ein Urteil darüber zu fällen welche Programme weiter verwendet werden, wurden 57 Datensätze welche im GetOrganelle Paper [28] verwendet wurden auf allen Programmen getestet. In dieser Arbeit wurden bei 47 Datensätzen von 57, mit dem GetOrganelle erfolgreich zirkuläre Chloroplasten extrahiert. Diese Daten sind auch frei zugänglich und wurden von NCBI heruntergeladen. Gerade hier gab es einige Abweichungen in Dateigrößen. Reads reichten von 75 Basenpaaren bis zu 300 Basenpaare pro Read. Es wurden hier fünf Millionen Reads pro Datei verwendet, da diese im GetOrganelle Paper auch verwendet wurden.

3.1.2 Welche Programme werden weiter verwendet.

Um alle Daten aus dem 1001 Genom Projekt (1135 Datensätze) zu berechnen, mussten aufgrund von Hardware technischen Limitierungen die besten Programme ausgewählt werden. Diese Programme müssen in in Geschwindigkeit sowie in Erfolgs- und Fehlerrate überzeugen. Des weiteren müssen diese Programme gut automatisierbar sein, d.h. am besten mit nur Befehl gestartet werden können, sodass kein weiterer Aufwand anfällt. Dies gilt vor allem auch bei der Wahl der Parameter mit denen das Programm gestartet wird. Diese können nicht für jeden Datensatz angepasst werden, was bedeutet dass die Standardparameter verwendet werden. Dies ist notwendig um einen hohen Durchsatz an Berechnungen zu ermöglichen.

Installation & Automatisierung Alle Programme konnten mit Hilfe von einigen Skripts und dem erstellen eines Dockercontainers so automatisiert werden, dass sie einen hohen Durchsatz erreichen konnten. Das Einzige Programm welches einen Händischen Schritt benötigt ist der GetOrganelle, hier muss die fastg Datei in Bandage geöffnet werden und der zirkuläre Chloroplast selbst heraus gesucht werden. Bei den verschiedenen Skripts handelt es sich vor allem um Start-Skripts. Aber es mussten auch ein paar kleine Skripts verwendet werden um kleine Bugs zu fixen. So kann der IOGA keine Unterordner verwenden da er sonnst versucht auf Falsche Dateien zuzugreifen und abstürzt. Dies scheint ein Bug in einem Splitt Befehl zu sein. Beim GetOrganelle mussten zusätzliche Befehle eingebaut werden damit SPAdes keine Fehlermeldungen bringt und abbricht, da er bestimmte Funktionen (hammer.py) nicht ausführen konnte welche für eine Fehler Korrektur verwendet werden, welche GetOrganelle gar nicht nutzt. Org.ASM konnte nur erfolgreich in einem Dockercontainer installiert werden, da dieses Programm sonnst verschiedenste Fehlermeldungen brachte. Alle Programme welche PERL verwenden, also chloroExtractor, fast-plast und NOVOPlasty, brachten Fehlermeldungen, da innerhalb des Dockercontainers Globale Variablen nicht vollständig gesetzt waren. Diese Fehler

waren aber nicht fatal, und konnten mit dem setzten dieser Variable leicht entfernt werden. Für jedes Programm wurde ein Skript geschrieben welches die Laufzeit überprüft und wenn dieses fertig ist, eine Auswertung startet.

Erfolgsrate Um zunächst zu überprüfen ob ein wirklich ein kompletter Chloroplast zusammengebaut wurden. Wurde bei den ersten Testdatensätzen ein Referenz Mapping auf TAIR10 benutzt. Hierzu wurde mit Bowtie2, später mit minimap2[46](Version: 2.10-r761) der Chloroplast auf das TAIR10 Chloroplasten Genom gemapt. Auch wurde mit AliTV [47] eine Visualisierung des Mappings erstellt. Nachdem klar war das es sich bei allen ausgegeben Daten um Chloroplasten handelt, und weil diese Art der Auswertung schlecht Automatisierbar war, wurde ein Bash Skript geschrieben welche die Auswertung übernimmt. Dieses Skript überprüft die Größe des Chloroplasten und in wie vielen Contigs der Chloroplast ausgegeben wurde. Hierzu wurde das SeqFilter[48](Version: 2.1.8) Skript verwendet, und anschließend über ein Bash Skript eine Entscheidung getroffen ob es sich um einen kompletten Chloroplasten handelt oder nicht (s. Anhang: ev_stat.sh). Hierzu wurden Verschiedene Kategorien eingeführt(s. Tab. 1). Diese Auswertung wurde für dann für alle Testdaten sowie die GetOrganelle Preprint Daten verwendet.

Table 1: **Erfolgsraten Einteilung** Das Skript ev_stat.sh scannt die Output Dateien und teilt diese je nach Größe und Anzahl der Contigs in verschiedene Kategorien ein.

Kategorie	Contigs	Basenpaare
Success	1	110 kbp - 180 kbp
Partial	> 1	110 kbp - 180 kbp
Incomp:high	> 1	$>$ 180 kbp
Incom:low	> 1	$>$ 110 kbp

Geschwindigkeit Einer der weniger entscheidenden aber dennoch wichtigen Punkte nach dem gefiltert wurde ist die Geschwindigkeit, oder besser die Laufzeit der Programme. Zunächst wurde hier die Durchschnittszeit genommen, die der Prozess zum rechnen benötigt, anschließend wurde mit dem time Linux Kommando die CPU als auch die Real-zeit gemessen. Die Geschwindigkeit von Programmen mit vielen Abhängigkeiten brauchen im Schnitt länger, da zum benutzen der Dockercontainer Singularity[49](Version:

[46]Li, H. (2018). Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. `10.1093/bioinformatics/bty191`

[47]Ankenbrand MJ, Hohlfeld S, Förster F, et al. (2017) AliTV—interactive visualization of whole genome comparisons. PeerJ Computer Science, `https://doi.org/10.7717/peerj-cs.116`

[48]`https://github.com/BioInf-Wuerzburg/SeqFilter`

[49]`https://singularity.lbl.gov/`

2.4.5-dist) verwendet wurde. Dieses Benötigt Zeit um den Container zu verwenden, zudem wird Zeit in Anspruch genommen wenn viele Daten in den Container gemountet werden müssen.

Benötigte Ressourcen Ein weiterer Punkt nachdem aussortiert wurde ist der benötigte RAM verbrauch. Es wurden verschiedene Größen von Dateien verwendet um in Erfahrung zu bringen wie sich dies auf Ressourcen und Laufzeit auswirkt. Zudem wurde zum Ausführen der Dockercontainer Singularity [49] verwendet, welches die benötigte Laufzeit und die benötigten Ressourcen, wie RAM beeinflusst.

3.2 Erzeugen von Chloroplasten aus genomischen Daten

Um so viele Chloroplasten wie möglich aus den genomischen Daten des 1001 Genom Projekts heraus zu holen, wurden der fast-plast und der chloroExtractor benutzt. Diese wurden mit Hilfe eines Dockercontainers und einigen Skripts (s. Anhang) voll automatisiert. Sodass nur ein Befehl nötig war um die komplette Pipeline zu starten und auszuwerten.

3.3 Varianz Analyse

Um mehr über die Chloroplasten und deren Verbreitung, sowie Mutationsrate und somit Varianz zu erfahren wurden zwei verschiedene Varianzanalysen durchgeführt. Zunächst sollte überprüft werden welche Einflüsse die Programme und ihre Strategien den Chloroplasten zu assemblieren, speziell deren Assambler, auf die Varianz der entstehenden Chloroplasten hat. Hierzu wurden die assamblierten Chloroplasten, welche beide verwendeten Programme gemeinsam hatten verwendet. Diese Läufe wurden zunächst zehn fach wiederholt, auch um einen Eindruck über die Reproduzierbarkeit der Ergebnisse zu bekommen. Diese Chloroplasten wurden anschließend mit minimap2 [46] auf das Referenzgenom (TAIR10 Chloroplast [50]) gemapt. Hiernach wurde eine Varianzanalyse mit Samtools[51](Version: 0.1.19-96b5f2294a) durchgeführt, hierzu wurde der Befehl 'mpileup/bcftools call' [52] (bcftools Versionen:0.1.19-96b5f2294a & 1.8) verwendet. Dieser führt eine Varianzanalyse bzw. ein SNP calling durch. Die zweite Varianzanalyse wurde auf allen Chloroplasten welche aus dem 1001 Genom Projekt gebaut wurden erstellt. Auch diese wurden auf den Referenz Chloroplasten mit minimap2 gemapt und anschließend mit Samtools' 'mpileup' Funktion einem SNP calling unterzogen.

3.4 GWAS

Häufig wird eine GWAS über das komplette Genom berechnet. Doch können auch einzelne Chromosomen oder Organellen bereits signifikante Varianten besitzen. So

[50] `https://www.ncbi.nlm.nih.gov/nuccore/NC_000932.1`

[51] Li H, Handsaker B, Wysoker A, et al. (2009) The Sequence alignment/map (SAM) format and SAMtools, Bioinformatics `10.1093/bioinformatics/btp352`

[52] Li H, (2011) A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data, Bioinformatics `10.1093/bioinformatics/btr509`

soll mit dieser GWAS der Einfluss von Chloroplasten Varianten auf Eigenschaften der *A.thaliana* getestet werden. Hierzu wurden die SNP callings aus der Varianzanalyse verwendet. Verschiedene Trait-Tabellen wurden von Arapheno[53], einer Trait Datenbank für *A.thaliana*, heruntergeladen und zusammen mit den Varianzanalyse Daten in ein R[54] Skript gegeben. Dieses R Skript nutzt zunächst vcfR[55], ein R Paket, um die verschiedenen VCF (Variance Calling File) Daten einzulesen. Anschließend ruft es ein weiteres R Skript auf welches freundlicher weiße von Korte et. al[39] zur Verfügung gestellt wurde und eine GWAS Analyse durchführt.

3.5 Struktur Varianz Analyse

Wie bereits erwähnt können Chloroplasten auch verschiedene Strukturelle Änderungen evolvieren. Diese sind durch die Rohdaten, welche meist short Reads sind, nicht aufzudecken. Da diese zu kurz sind um komplette Struktur Varianten zu überspannen.[7] Hierzu könnten nun die komplett de novo Assemblierten Chloroplasten verwendet werden. Es wurde versucht mit Delly[56] (Version: v0.7.8) und Breakdancer[57](Version: 1.3.6) Struktur Varianten in Chloroplasten zu finden.

3.6 Neue Chloroplasten

Um neue Chloroplasten von Spezies zu finden, welche noch nicht in der CP-Base [58, 59] Datenbank sind, wurde eine Liste von Möglichen Daten von NCBI mit CP-Base verglichen. Nur 49 Datensätze waren ohne Eintrag in CP-base und hatten somit noch keinen Dokumentierten Chloroplasten für diese Spezies. Auf diese 49 Datensätze wurden sowohl der chloroExtractor als auch der fast-plast angewendet. Um die NCBI liste von Interessanten Daten zu erhalten wurde mit folgendem Befehl gesucht:
' (((((((("green plants"[orgn]) AND "wgs"[Strategy]) AND "illumina"[Platform]) AND "biomol dna"[Properties]) AND "paired"[Layout]) AND "random"[Selection])) AND "public"[Access]'
Mit einem Skript (s. Anhang, cpbase.sh) wurden alle Spezies Einträge von CP-base geladen welche einen Chloroplasten besitzen. Anschließend wurde mit einem folgendem Perl-Einzeiler die Datensätze herausgegeben welche noch keinen Eintrag in CP-base haben. Zudem musste der Datensatz mindestens zwei Millionen Reads haben und mindestens 200 Basenpaare pro Read aufweisen.
'perl -F"," -ane 'print if $F[44]>399 and $F[8]>999999' SraRunInfo_plants.csv | grep -vf species_cpbase.list | sort -u -t, -k29,29 | shuf'

[53] https://arapheno.1001genomes.org/
[54] https://www.r-project.org/
[55] https://cran.r-project.org/web/packages/vcfR/index.html
[56] https://github.com/dellytools/delly
[57] https://github.com/genome/breakdancer

[58] http://rocaplab.ocean.washington.edu/old_website/tools/cpbase
[59] http://rocaplab.ocean.washington.edu/tools/cpbase_test/

4 Ergebnisse

4.1 Automatisierung

Um eine Automatisierung aller Programme zu erreichen wurde für jedes Programm ein Dockercontainer gebaut welcher mit Singularity verwendet wird. Zudem wird die komplette Auswertung von einigen Skripts übernommen. Um dies zu Bewerkstelligen wurden mehrere Skripts geschrieben welche sich gegenseitig aufrufen um den kompletten Ablauf sicherzustellen (Fig. 6). Das einzige Skript welches aktiv ausgeführt werden muss ist das run_SRRchl.sh. Dieses Skript setzt Links zu anderen Skripts, zum einen zwei Auswertungs Skripts (ev_stat.sh und percent_stat.sh) und zu einem Skript namens cp_skript.sh. Dieses cp_skript übernimmt den kompletten Aufbau der Ordner Struktur, und linkt all die Skripts die jedes Programm braucht, so brauchen IOGA und GetOrganelle eine Referenz, diese wird von diesem Skript in die passenden Ordner kopiert. Auch kopiert und führt dieses cp_skript.sh das Skript aus welches die NOVOPlasty Konfigurationsdatei automatisiert für jeden Datensatz schreibt (make_NP_config.pl). Für jeden Datensatz wird so ein Ordner erzeugt mit jeweils dem Programm als Unterordner. In jedem Unterordner werden die roh Daten verlinkt, sowie für jedes Programm das passende Evaluierungsskript und Runskript. Als letztes linkt es sbatch_run_all.sh und ev_all.sh in den jeweiligen Datenordner. Diese werden nun vom run_SRRchl.sh Skript ausgeführt. Das sbatch_run_all.sh Skript geht nun in jeden Unterordner und startet die jeweiligen Programme über sbatch und deren Runskript. Zudem startet es auch die dazugehörigen Evaluierungsskripts, welche auch gleichzeitig als Überwachungs Skript dienen. Sobald der Slurm Job fertig ist, startet das Evaluierungsskript des jeweiligen Programmes damit, die Finale Output Datei zu überprüfen und diese in eine der vier Erfolgs Kategorien einzuteilen. Zudem schiebt es alle Dateien welche keine Log Dateien oder Finale Output Dateien sind in einen raw_Programm Ordner, damit dieser mit dem clear_skript.sh gelöscht werden kann, falls diese Daten nicht mehr benötigt werden. Sobald alle Datensätze fertig sind, wird mit dem ev_stat.sh Skript eine Datei mit einer Erfolgstabelle mit jedem Programm erstellt. Percent_stat.sh kann dann genutzt werden um eine Zusammenfassung über alle Datensätze zu erhalten.

4.2 Daten: Simulierte Daten

Die Simulierten Daten, welche mit ART[44, 45] erzeugt wurden um das verhalten der Programme bei verschiedenen Verhältnissen zu testen, konnten von drei Programmen, dem chloroExtractor, fast-plast und Org.ASM bei allen drei Datensätzen geschafft werden. Diese schaffen es einen vollständigen zirkulären Chloroplasten zu bauen. NOVOPlasty baut zwar auch einen kompletten Chloroplasten doch gibt dieser nur die drei verschieden Contigs aus (IR, SSC, LSC), und schafft es nicht diese in einen zirkulären Chloroplasten zu vereinen. GetOrganelle wie auch der IOGA schaffen es nicht die simulierten Datensätze zusammen zu bauen da sie mit einem Fehler abbrechen oder wie im falle des IOGA nach zwei Wochen Laufzeit abgebrochen werden. (s. Tabelle 2)

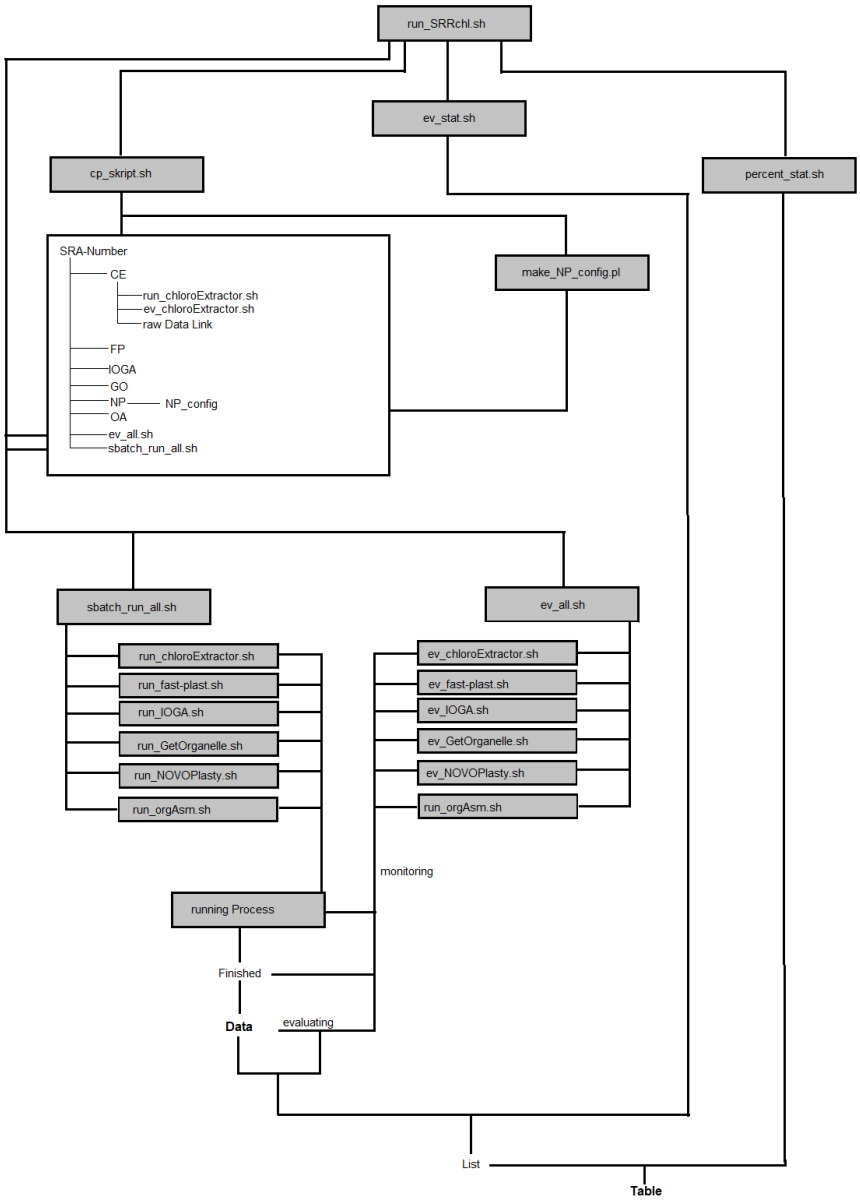


Figure 6: **Automatisierungsskripts** Ablauf der verwendeten Skripte um eine Automatisierung zu erwirken, hier wird nur das run_SRRchl.sh Skript ausgeführt und alle anderen Skripte werden automatisch bis zur Auswertung ausgeführt

Table 2: **Test Datensatz: Simulierte Daten** S steht für Success, E für Error, die angegebene Zahl steht für die Anzahl der Contigs. Bis auf IOGA und GetOrganelle konnten alle anderen Programme die Simulierten Daten zu einem Chloroplasten zusammenbauen, auch wenn im Falle des NOVOPlasty nicht zirkulär. Die IOGA Läufe mit "-" wurden nach zwei Woche Laufzeit abgebrochen.

Sim(Genome:Chloroplast)	CE	FP	NP	GO	OA	IOGA
1:10	S	S	S-3	E	S	E
1:100	S	S	S-3	E	S	-
1:1000	S	S	S-3	E	S	-

4.3 Daten: 1001 Genom Projekt, 11 Testdatensätze

Aus den Daten des 1001 Genom Projekts [38, 7] wurden zunächst elf Testdatensätze verwendet um auch reale Daten auf allen Programmen zu testen. Von den elf Testdatensätzen des 1001 Genom Projekts konnten sechs verschiedene vollständige zirkuläre Chloroplasten zusammengebaut werden. Von diesen sechs bringt der fast-plast fünf ein und der chloroExtractor einen. Keines der anderen Programme konnte einen weiteren zirkulären Chloroplasten erzeugen (s. Tab.3). Diese elf Datensätze des GetOrganelles wurden per Hand ausgewertet, keiner dieser elf Datensätze konnte einwandfrei mit Bandage zu einem zirkulären Chloroplasten gebaut werden, da immer kein Ringschluss vorhanden war. (vergleiche Fig.7, Fig. 8)

4.4 Daten: GO-Preprint

Um mehr Daten zu testen, wurden alle 57 Datensätze des GetOrganelle Papers [28] benutzt. Da der GetOrganelle diese Daten eigentlich erfolgreich schaffen sollte wurde hier versucht mit dem fcg.pl Skript des chloroExtractors eine Automatisierung der Daten zu erwirken. Doch versucht der GetOrganelle zunächst die die fastg-Graphen zu verbessern, dies führt dazu dass das fcg.pl Skript nicht mehr funktioniert. So wurden die fastg-Graphen aus SPAdes direkt verwendet, doch ergaben sich hier leider nur zwei Datensätze als zirkuläre Chloroplasten. Auf Nachfrage beim GetOrganelle Team, hieß es dass, wenn es notwendig war alle Parameter angepasst wurden und dass alle Chloroplasten per Hand aus Bandage geholt wurden. Von 57 Datensätzen, welche im GetOrganelle Paper verwendet wurden, konnten 40 mit allen Programmen fertig gestellt werden (s. Tab. 4). So konnten 35 von 40 Erfolgreichen Datensätzen durch den fast-plast und den chloroExtractor erreicht werden. Der fast-plast schafft es 31 Chloroplasten Genome komplett zu bauen, davon sind 17 nur von ihm geschafft worden.(Fig. 9) Der chloroExtractor schafft 14 Chloroplasten, wo von drei nur von ihm geschafft werden. Die zwei Erfolge des GetOrganelles, mit Hilfe des fcg.pl Skripts des chloroExtractors werden auch nur vom GetOrganelle geschafft. Von den sieben des NOVOPlasty ist einer dabei welcher nur von diesem geschafft wird. Von den elf

Figure 7: **Bandage: GetOrganelle SRRSRR1945443** Kein Chloroplast erkennbar im fastg Graphen. Keine vernetzung der Contigs per se erkennbar. Dazu im vergleich der graphen des chloroExtractors (Fig. 8)

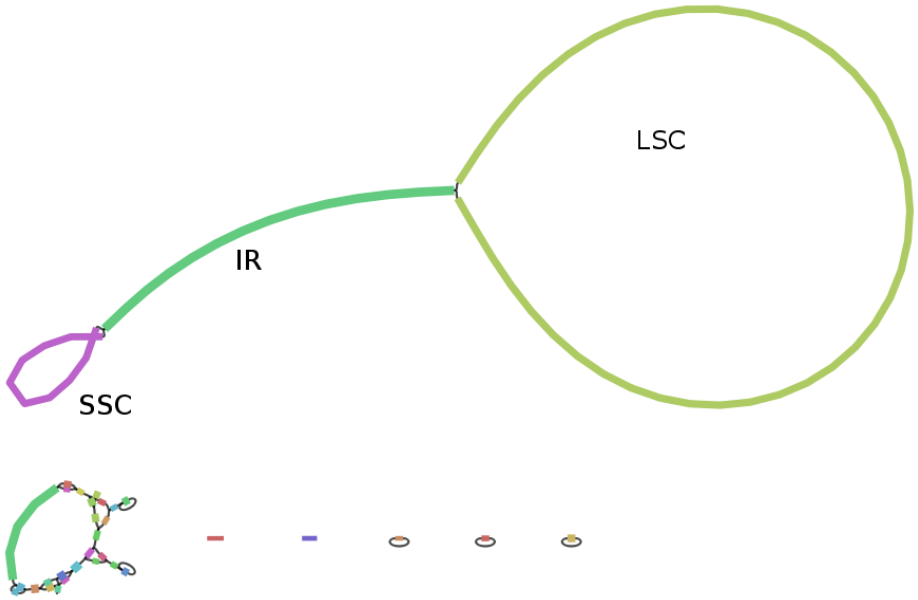


Figure 8: **Bandage: chloroExtractor SRRSRR1945443** Chloroplast klar erkennbar, dieser zeichnet sich aus durch einen Großen Kreis (LSC - Gelb), verbunden über eine Linie (IR - Grün) auf einen Kleinen Kreis (SSC - Violett).

Table 3: **Test Datensatz: 1001 Genom Project** S steht für Success, E für Error, I für Incomplete, die angegebene Zahl steht für die Anzahl der Contigs. Sechs verschiedene Chloroplasten konnten zu einem zirkulären Chloroplasten zusammengebaut werden, dabei werden bereits fünf vom fast-plast abgedeckt und einer wird von chloroExtractor beigesteuert. Felder mit einem "-" wurden Abgebrochen da sie nach einer Woche noch nicht fertig waren.

SRA	CE	FP	NP	GO	OA	IOGA
SRR1945435	I-5	I	I-4	I	E	I-6
SRR1945436	I-6	S	I-3	I	I	I-8
SRR1945437	I-5	I	I-4	I	I	I-10
SRR1945438	S-3	S	I-6	I	E	I-10
SRR1945439	I-4	S	I-1	I	I	I-10
SRR1945440	I-4	S	E	I	E	I-9
SRR1945441	I-5	S	E	I	I	I-6
SRR1945442	I-4	I	I-1	I	-	-
SRR1945443	S	I	I-2	I	I	I-8
SRR1945444	I-4	I	E	I	I	I-8
SRR1945445	I-4	I	E	I	E	I_7

des Org.ASM ist auch einer nur von diesem Geschafft worden. Doch wurden einige auch mehrfach geschafft. So sind vier Stück nur von fast-plast und chloroExtractor geschafft worden. Zwei Stück von fast-plast und Org.ASM, sowie jeweils einer von Org.ASM und NOVOPlasty; chloroExtractor und Org.ASM; fast-plast und NOVOPlasty. Drei Erfolge teilen sich fast-plast, chloroExtractor und Org.ASM. Sowie jeweils ein Erfolg teilen sich fast-plast, Org.ASM, NOVOPlasty, als auch einer von fast-plast, chloroExtractor und NOVOPlasty. Zwei Chloroplasten konnten von allen Programmen bis auf GetOrganelle und IOGA gelöst werden (u.a. Fig. 10: SRR5602602 - Laurus nobilis), wobei letzteres Programm nicht einen Erfolg hat(Fig. 9 & Tab. 4). So können bereits 35 von 40 Chloroplasten alleine durch fast-plast und chloroExtractor geschafft werden.

4.5 Die besten Programme: fast-plast und chloroExtractor

Da aus zeitlichen und hardwaretechnischen Gründen nicht alle Programme weiterverwendet werden konnten, wurde nach Erfolgsrate, Geschwindigkeit und benötigten Ressourcen (s. Tab 5) gefiltert, am wichtigsten war aber die Automatisierbarkeit der Programme. Bis auf der GetOrganelle konnte für jedes Programm eine Automatisierbarkeit erwirkt werden ohne Daten außen vor zu lassen. Der GetOrganelle benötigt das öffnen der fastg Datei in einem Visualisierungsprogramm für fastg-Graphen, hier wird Bandage empfohlen. Bandage hat allerdings eine schlechte Kommandozeilen Anbindung wodurch auch keine Automatisierbarkeit durch Skripts erfolgen konnte. Es wurde auch versucht mit

Table 4: **Test Datensatz: GetOrganelle Preprint** 40 von 57 Datensätze konnten komplett gelöst werden. 31 Datensätze konnten mit dem fast-plast zu einem Chloroplasten gebaut werden, die 14 die der chloroExtractor schafft enthalten die restlichen 9 um auf alle 40 Chloroplasten zu kommen. Somit konnten mit zwei Programmen 74% gelöst werden.

Tool	SUCCESS	%	ERROR	PARTIAL	INCOMPl	NO_PAIR	Total
CE	14	~26%	11	17	12	3	
FP	31	~57%	0	18	5	3	
GO	2	~4%	21	26	5	3	
IOGA	0	~0%	22	28	4	3	
NP	7	~13%	19	8	20	3	
OA	11	~20%	36	4	3	3	
Summary	40	~74%	-	-	-	3	57

dem fcg.pl Skript aus dem chloroExtractor, welches genau diesen Schritt im chloroExtractor automatisiert, zu verwenden um auch beim GetOrganelle eine Automatisierbarkeit zu erreichen. Doch führte dies nur bei sehr wenigen Daten zum Erfolg, da der GetOrganelle die von SPAdes erstellte fastg Datei versucht zu verbessern, und die getrimmte Datei nicht mehr vom fcg.pl Skript verwendet werden kann. Dies passiert wohl weil der GetOrganelle beim verbesserten fastg Graphen versucht Namen und Sequenzen anzupassen, womit das fcg.pl Skript nicht zurecht kommt. Es wurde auch versucht die roh fastg Dateien des GetOrganelle zu benutzen dies ergab zwar eine Automatisierbarkeit, doch würden so Teile des GetOrganelles, nämlich das verbessern der fastg Datei unterschlagen. Die Laufzeiten der Programme unterscheiden sich sehr, von 30 Minuten bis über eine Stunde, auch die RAM werte sind sehr unterschiedlich, diese reichten von wenigen 20 Gigabyte bis zu 60 Gigabyte. All diese Werte sind Durchschnittswerte, da verschiedene Größen von Dateien als Eingabe verwendet wurden. Da nicht alle Dateien die gleiche Anzahl an Reads hatten, sowie die Größen der einzelnen Reads sich unterschieden. Diese reichten von 75 Basen paare bis zu 300 Basen paare, Anzahl der Reads und somit Größe der Dateien reichten von eine Millionen Reads bis zu fünf Millionen Reads. Die Laufzeiten sind, vor allem bei Programmen mit vielen Abhängigkeiten, erhört. Da zum nutzen der Dockercontainer Singularity [49] verwendet wurde. Die Programme welche in oben genannten Punkte überzeugt haben sind der fast-plast und der chloroExtractor. Der fast-plast benötigt zwar die meisten Ressourcen und ist nicht der schnellste, aber hat mit Abstand die größte Erfolgschance. Zudem ist er voll automatisierbar und erreicht dies mit den vorgegebenen Standard Parametern. Als zweites Programm wird der chloroExtractor verwendet, dieser ist schnell, Ressourcen arm und hat nach dem fast-plast die zweit höchste Erfolgsrate. Mit beiden Programmen konnten 35 von den 40 Erfolgen von 57 Chloroplasten der GetOrganelle-Preprint Daten berechnet werden. Zudem haben diese beiden Programme die wenigsten Probleme bei der Handhabung wie

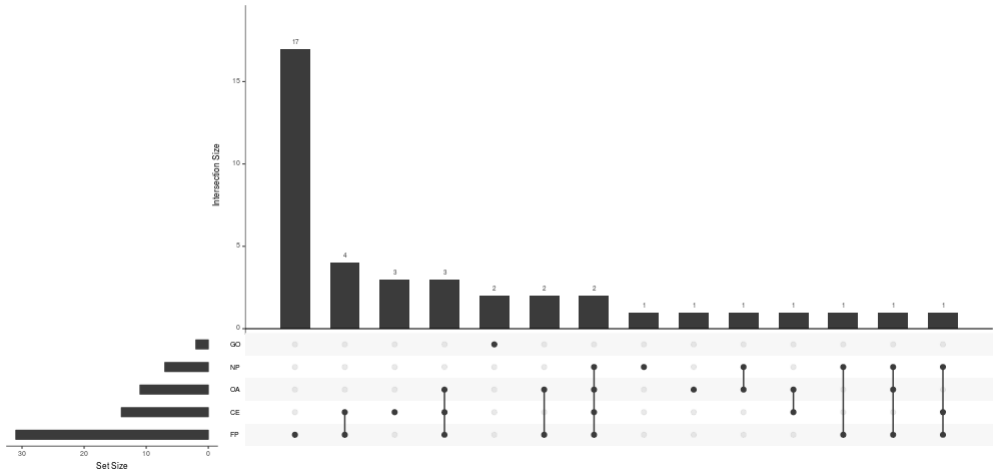


Figure 9: **Upset Diagramm GO-Preprint** Hier wird gezeigt wie sich die einzelnen Erfolge auf die 40 Stück aufteilen. So schafft der fast-plast 17 Chloroplasten welches kein anderes Tool schafft. Fast-plast und chloroExtractor haben vier Erfolge gemeinsam, der chloroExtractor schafft drei Chloroplasten welche kein anderes Programm schafft. usw. So werden 35 der geschafften 40 durch fast-plast und chloroExtractor abgedeckt.

Table 5: **Laufzeit und Ressourcenverbrauch** Alle Laufzeiten sind Durchschnittswerte, RAM werte zu Peakzeiten. Die Laufzeiten reichen von 30 Minuten (chloroExtractor) bis zu 100 Minuten (IOGA), die RAM Nutzung unterschied sich auch erheblich, diese reichen von 20 GB (chloroExtractor) bis hin zu 60 GB (fast-plast). Aufgrund der Nutzung von verschieden großen Datensätzen können nur Durchschnittswerte Angegeben werden.

Tool	Laufzeit	RAM
CE	~ 30 min	~ 20 GB
FP	~ 60 min	~ 60 GB
GO	~ 40 min	~ 50 GB
IOGA	~ 100 min	~ 40 GB
NP	~ 30 min	~ 30 GB
OA	~ 60 min	~ 30 GB

auch bei der Installation zu beginn gemacht. Sie sind durch die gegebenen Parameter einfach zu verwenden und zu Automatisieren. Die von den Programmen geschriebenen Log Dateien sind einfach gehalten um dem Ablauf zu folgen und klar verständlich, der fast-plast gibt sogar drei dieser Dateien aus, da er unterscheidet zwischen Warn- und

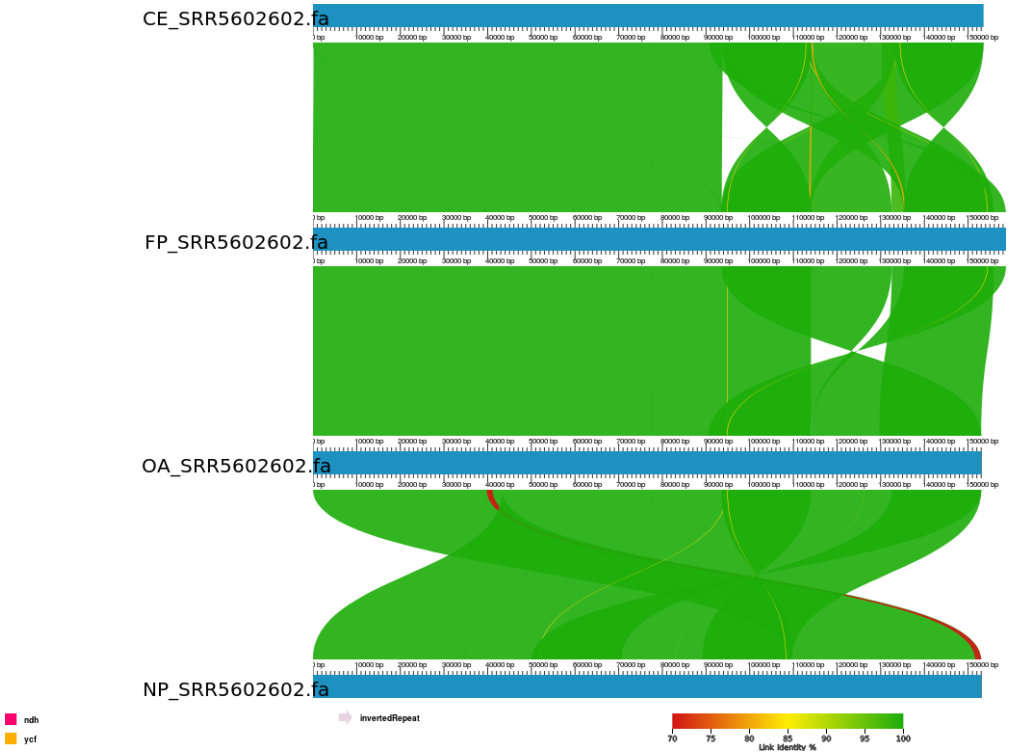


Figure 10: **AliTV SRR5602602 - *Laurus nobilis*** SRR5602602 -*Laurus nobilis* , SRR5602602 - Laurus nobilis, Daten des GO-Preprints, geschafft von ChloroExtractor, NOVOPlasty, fast-plast und Org.ASM. Orientierung von LSC sowie von SSC und IRs können nicht perfekt aufgelöst werden und können durchaus Verdreht sein.

Fehlermeldungen sowie Standard Meldungen, und eine Datei für den Output der eingebundenen Programme. Der chloroExtractor gibt seine Kompletten Meldungen über ein übergeordnetes Programm aus, welche den Ablauf steuert (PipeWrap). Dieses Programm gibt alles auf STDERROR aus und kann damit einfach mit geloggt werden. In diesem Fall wurde über die slurm Datei, welche von dem verwendeten queueing System ausgegeben wird, mit geloggt. Diese beiden Programme wurden auf allen Daten des 1001 Genom Projekts laufen gelassen, um möglichst viele Chloroplasten zu generieren.

4.6 1001 Genom Projekt

Ziel so viele Chloroplasten wie möglich vollautomatisch aus kompletten Genom Datensätze zu erzeugen, wofür zwei Programme ausgewählt worden sind, wurde zunächst auf Datensätzen des 1001 Genom Projekt versucht. Von den 1135 Datensätzen welche im 1001 Genom Projekt gesammelt wurden, konnten 946 Datensätze erfolgreich von NCBI heruntergeladen werden. Die restlichen 189 konnten nicht richtig heruntergeladen werden aufgrund von Downloadfehlern. Zudem waren 47 Datensätze keine paired end Datensätze, und konnten deshalb nicht verwendet werden. Von diesen 899 restlichen Datensätzen konnten mit dem fast-plast und dem chloroExtractor 303 komplette zirkuläre Chloroplasten vollautomatisch gebaut werden, dies entspricht etwa 34%. (Tab. 6).

Table 6: **Datensatz: 1001 Genom Project** SUCCESS, echte zirkuläre Chloroplasten. Error, Fehler oder Abbrüche im Programm. Partial, keine zirkulären Chloroplasten aber Contigs richtig identifiziert. Incomplete, Nicht richtig identifizierte Chloroplasten.

Tool	SUCCESS	%	ERROR	PARTIAL	INCOMPLETE	NO_PAIR	Total
CE	136	~15%	54	3	706		
FP	266	~30%	29	11	593		
Summary	303	~34%	-	-	-	47	946

4.7 Varianz Analyse

Um die Varianz Analyse durchzuführen und vor allem zu überprüfen ob die Assambler bzw. die Programme an sich einen Einfluss darauf haben, indem sie z.B. zufällige Seeds verwenden oder zufällige Daten bevorzugen, wurden 89 Datensätze des 1001 Genom Projekts verwendet. Diese 89 Datensätze zeichnen sich dadurch aus, dass sowohl der chloroExtractor als auch der fast-plast diese zu vollständigen Chloroplasten zusammengebaut haben. Diese Datensätze wurden noch zehn weitere Male berechnet. So wurden auf elf mal 89 Datensätzen überprüft welche Einflüsse die Programme auf die Varianz haben. Der chloroExtractor und somit der Assambler SPAdes brachte bei allen elf Durchläufen die exakt gleichen Sequenzen heraus. Dieses Programm arbeitet also 100% Reproduzierbar (s. Abb. 11). Im Gegensatz dazu der fast-plast, dieser schaffte

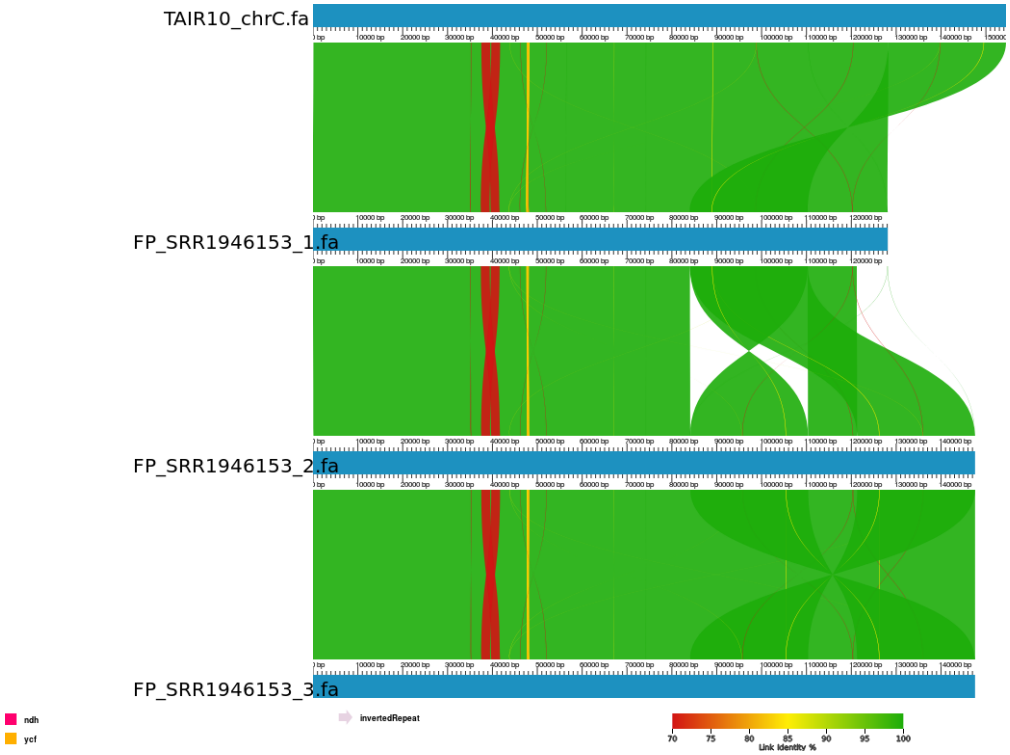


Figure 11: **fast-plast SRR1946153** Drei verschiedene Läufe auf den selben Daten, der fast-plast schafft einen davon nicht (Lauf 1), den anderen aber schon (2 u. 3). Hier fehlt ein Teil des IR, wodurch auch nicht als Erfolg gewertet wird.

es nicht einmal bei allen elf Durchläufen alle Chloroplasten wieder korrekt zusammen zubauen, bei bis zu neun verschiedenen Datensätzen konnte kein Erfolgreiches Ergebnis erzielt werden(s. Abb. 12). Interessanter weiße waren nicht immer die selben Datensätze betroffen, so konnten bei einigen Durchläufen ein Erfolg erreicht werden, bei dem nächsten Durchlauf aber nicht. Ob dies ein Zufalls Effekt des Programms oder der verwendeten Rechner-Infrastruktur ist, konnte nicht überprüft werden. Die zweite Varianz Analyse bzw. SNP calling wurde auf allen Erfolgreich zusammengebauten Chloroplasten durchgeführt. Das SNP calling ergab dass auf allen 303 Chloroplasten insgesamt 2128 SNPs gefunden wurden. Diese Ergebnisse werden für die GWAS Analyse verwendet.

4.8 GWAS

Die GWAS Analyse, welche mit den 303 kompletten Chloroplasten aus den Daten des 1001 Genom Projekts und den 2128 gefunden SNPs durchgeführt wurde, konnte nur auf zwei verschieden Traits berechnet werden. Dies waren die Eigenschaften Flowering Time bei 16°C sowie bei 10°C. Dies sind die beiden Traits am besten untersucht sind und deswegen auch die meisten Daten beinhalten. Für alle anderen Traits konnten keine Berechnungen erstellt werden da die Datenmenge nicht für eine GWAS ausreichend

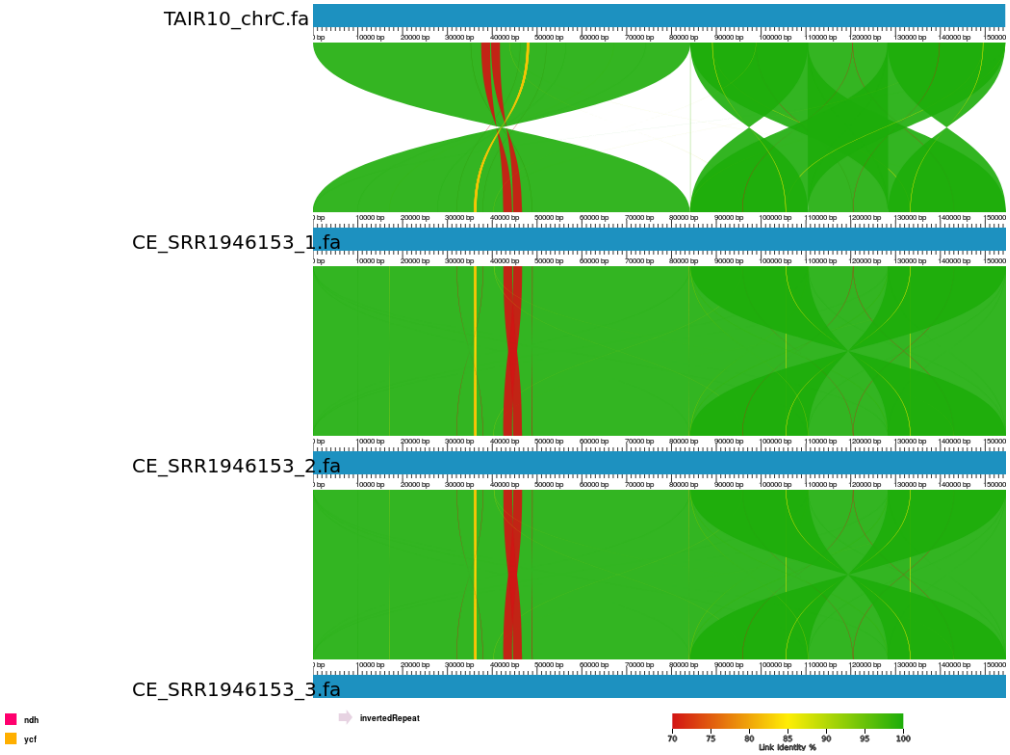


Figure 12: **chloroExtractor SRR1946153** Drei verschiedene Läufe auf den selben Daten, der chloroExtractor bringt das gleiche Ergebnis für alle Durchläufe. Da die orientierung von LSC, SSC und IR nicht aus short Reads herausgelesen werden kann, kommt es vor das diese Verdreht sind zur Referenz.

ist. Dies hängt damit zusammen wie die verschiedenen Matrizen zueinander berechnet werden. Zudem konnte für beide berechneten Traits keine Signifikanz für einen SNP gefunden werden.

4.9 Struktur Varianz Analyse

Für die Struktur Varianz Analyse konnten keine Ergebnisse erzielt werden, Grund hierfür war unter anderem fehlende Zeit. Aber auch konnten keine guten Programme gefunden werden welche mit kompletten Chloroplasten umgehen konnten. Die meisten nutzten direkt Illumina short reads, wie Delly[57] oder Breakdancer[56].

4.10 Neue Chloroplasten

Aus NCBI wurden 79657 Datensätze heruntergeladen, dies sind alles Pflanzengenome. Diese Liste wurde mit den Einträgen von CP-base verglichen. Es blieben die Übrig, welche keinen Eintrag in CP-base haben. Von diesen 79657 blieben nur 49 Datensätze. Diese wurden auf fast-plast und chloroExtractor benutzt, und es wurden 17 zirkuläre Chloroplasten erfolgreich zusammengebaut (s. Tab. 7). Somit wurden 17 neue Chloroplasten von Spezien welche zuvor noch keinen genomisch bekannten Chloroplasten hatten erfolgreich erstellt (s. Tab. 8).

Table 7: **Neue Chloroplasten** Von den 49 Spezies welche bisher noch keinen Eintrag in CP-base hatte konnten mit Hilfe des fast-plasts und des chloroExtractors 17 neue bisher nicht bekannte Chloroplasten Genome gebaut werden

Tool	SUCCESS	ERROR	PARTIAL	INCOMPl	NO_PAIR	Total
CE	4	20	16	9	0	
FP	15	7	22	5	0	
Summary	17	-	-	-	0	49

5 Diskussion

5.1 Definition von Success, Einteilung der Erfolge über Genom Länge.

Jegliche Einteilung in die Erfolgs Kategorien: Success, Partial, Incomplete_high und Incomplete_low werden von einem Skript übernommen welches zunächst den SeqFilter benutzt um Informationen über diese Datei zu erhalten. Der SeqFilter zählt die Sequenzen sowie deren Größe. Das Evaluationsskript des jeweiligen Programms teilt aufgrund dieser Daten in die Kategorien ein (s. Tab. 1). Diese Variante ist zwar voll Automatisiert doch nicht Fehlerlos, so können Falsch Positive Sequenzen vorkommen. Diese könnte eine Sequenz aus 150 kbp Adenin sein, und das Skript würde es als einen Success ansehen. Die Daten wurden Stichprobenartig überprüft und dies kam in diesen Stichproben nicht vor,

Table 8: **Liste neue Chloroplasten** Liste von 17 Spezies welche mit Hilfe des fast-plast und des chloroExtractors nun ein bekanntes Chloroplasten Genom besitzen.

SRA	Spezies
DRR057122	*Momordica charantia*
DRR089517	*Betula chichibuensis*
ERR1462646	*Hippophae rhamnoides*
ERR2001942	*Betula pendula*
ERR2003066	*Potentilla micrantha*
ERR2174632	*Solanum pennellii*
ERR2187925	*Geum urbanum*
SRR1503730	*Agave tequilana*
SRR2847417	*Manihot glaziovii*
SRR3194007	*Artocarpus altilis*
SRR3724930	*Taraxacum S3*
SRR4457832	*Pityopsis pinifolia*
SRR5046394	*Ephedra gerardiana*
SRR5464169	*Trema orientalis*
SRR5590327	*Lagenaria siceraria*
SRR5799057	*Fragaria vesca*
SRR5838021	*Populus deltoides*

doch ist es nicht auszuschließen. Um sicher zu gehen müsste jeder erstellter Chloroplast auf eine Referenz gemapt werden oder sogar durch Sequenzierung bestätigt werden. Erste Möglichkeit wäre nur Rechenaufwand, könnte aber bei Chloroplasten die noch nicht veröffentlicht wurden oder keine Referenz besitzen schwer werden, zweite Möglichkeit ist sehr Kosten intensiv würde aber letzte Zweifel beseitigen. Eine Verbesserung des Skripts könnte auch eine strengere Beurteilung sein, zumindest wenn man mehr Grundinformationen hat. So könnten bei den Versuchen mit den *A.thaliana* des 1001 Genom Projekts die Grenzen Strenger gewählt werden, da es sich hier immer um die gleiche Spezies handelt. Doch könnten somit die Anzahl der Falsch Negativen erhöht werden, z.B. wenn eine *A.thaliana* Art eine Struktur Variante besitzt mit Verlust eines IR. Die Grenzen wurden bewusst großzügiger Gewählt, da dies den größten Teil der Chloroplasten abdecken dürfte. Gerade bei Chloroplasten welche bisher nicht veröffentlicht oder bekannt sind ist eine Abschätzung schwer, da die Größen von Chloroplasten doch sehr Variieren können. Eine weitere Möglichkeit zu testen ob es sich wirklich um einen Chloroplasten handelt wäre die Verwendung von Benchmarking Universal Single-Copy Orthologs (BUSCO[60]), hierzu werden extrem konservierte orthologe Gene verwendet und überprüft ob diese alle vorhanden sind. Da ein Chloroplast Genom an sich sehr konserviert ist könnte eine An-

[60]Simão FA, Waterhouse RM, Ioannidis P,et al. (2015), BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics, `10.1093/bioinformatics/btv351`

zahl von Genen genommen werden und diese in einem solchen Modell verwendet werden.

5.2 Die Entscheidung für fast-plast und chloroExtractor

Es wurde im Ergebnis Teil erklärt warum gerade der fast-plast und der chloroExtractor weiter verwendet wurden. Doch gibt es auch gründe warum sich speziell gegen andere Programme entschieden wurden. So wurde sich gegen den IOGA entschieden, nicht nur weil er extrem langsam ist sondern auch weil er keinerlei Log File während des Prozesses schreibt, erst wenn dieser Komplett beendet ist, so war vor allem zu beginn extrem schwer nachzuvollziehen ob der IOGA nun wirklich noch Arbeitet oder evtl in irgendeinem Loop fest hängt oder sogar aufgehört hat zu arbeiten aber den Prozess nicht beendet. Auch wurde der IOGA zum letzten mal vor zwei Jahren geupdated, es scheint also keine Regelmäßige Wartung oder Verbesserung statt zu finden. Es wurde sich auch gegen den NOVOPlasty entschieden, dieser benötigt zwar keine Abhängigkeiten da er komplett in Perl geschrieben ist, doch hat dies einige Probleme mit sich gebracht. So werden z.B. nicht alle Read header richtig eingelesen wenn der dazugehörige Reguläre Ausdruck (Regular Expresion - RegEx) nicht komplett passt, dies kam häufiger vor da nicht alle Header gleich aufgebaut sind und wohl ein paar nicht abgedeckt wurden. Das zweite Problem mit NOVOPlasty ist die Konfigurationsdatei, diese muss exakt dem Beispiel entsprechen und darf nicht ein Zeichen mehr oder weniger enthalten, oder gar Zeilen. Da diese Datei nicht über RegEx eingelesen wird sondern Zeile für Zeile durchgegangen wird. So kam es gerade am Anfang vor das der NOVOPlasty gar nicht funktionierte da ein Leerzeichen in einer nicht verwendeten Option fehlte. Der NOVOPlasty scheint noch Regelmäßig geupdated zu werden, doch Änderte sich bei diesen Updates der Aufbau der Konfigurationsdatei, weswegen jedes mal das Skript zum erstellen dieser Datei umgeschrieben werden musste. Auch warf der NOVOPlasty Fehler in denen gesagt wird dass der Seed nicht lesbar oder inkompatibel sei. Doch wurde bei jedem Versuch als Seed die gleiche Datei verwendet, und dieser Fehler trat nur ab und zu auf. Der Org.ASM lief zwar nachdem er installiert wurde gar nicht so schlecht, im Vergleich würde er auf dem dritten Platz landen, doch gab es einige Probleme bei der Installation. Nur in einem Dockercontainer mit einigen Tricks konnte es geschafft werden dieses Programm erfolgreich zu installieren. Der GetOrganelle konnte zwar mit dem fcg.pl Skript des chloroExtractors einigermaßen automatisiert werden, doch unterschlägt dies dann das eigentliche Endprodukt des GetOrganelles, da das verbesserte bzw. getrimmte fastg nicht vom fcg.pl Skript erkannt wurde und deshalb nur das fastg aus SPAdes selbst verwendet werden kann, dies aber häufig schlechter Ausfällt als das getrimmte oder gar das fastg welches SPAdes im chloroExtractor ausgibt.

5.3 Fazit aus der Erfolgschance

Es wurden in dieser Arbeit 303 Chloroplasten Genome von *Arabidopsis thaliana* und 17 Neue (s. Tab. 8) erstellt. Nimmt man von den Versuchen die gesamt Zahl, so konnten in etwa 30% der Datensätze zu Chloroplasten Genomen führen. Dies entspricht Tatsächlich mehr als am Anfang der Arbeit angenommen, hier wurden in etwa 10 - 20% geschätzt.

Allerdings auch ohne die anderen Programme, abgesehen von chloroExtractor, großartig getestet zu haben. Nimmt man den nur die Erfolgschance von chloroExtractor war diese erste Abschätzung gar nicht so schlecht. Dies zeigt mir, als einer der Entwickler des chloroExtractors, dass dieser noch mehr verbessert werden kann.

5.4 Erhöhen der Erfolgsrate

Es gibt mehrere Möglichkeiten wie eine Erfolgsrate erhöht werden könnte. So könnte versucht werden auf die Daten speziell die Start Parameter festzulegen. Dies würde allerdings einiges an Tests benötigen. Auch könnten die Parameter jedes mal geändert werden, dann aber unter dem Verlust einer Automatisierbarkeit. In diesen Versuchen wurden verschieden Große Datensätze verwendet, und es lässt sich nicht sagen ob eine Erhöhung dieser einen echten Vorteil bringen würde, hierzu müssten alle Daten noch einmal gestartet werden, dann mit erhöhten oder niedrigeren Readmengen. Theoretisch kann dies einen Zuwachs an Erfolg bringen, wenn das verwendete Programm denn auch alle Daten verwendet, die es bekommt und nicht irgendeinen Cutoff ab einer bestimmten Daten bzw. Read Menge hat. Wenn die kompletten Daten verwendet werden würden hätte dies auch den Vorteil das man sicher gehen kann dass die Daten nicht sortiert wurden, indem man diese einfach nochmal durch mischt. Dies ist wichtig, vor allem bei Programmen welchen einen Cutoff benutzen, denn hier könnte es vorkommen, dass wenn eine Datei sortiert ist Chloroplasten Reads am Ende der Datei liegen und diese somit gar nicht erst benutzt werden. Dennoch ist zu beachten, je mehr Daten natürlich verwendet werden desto länger brauchen die Programme, zudem kommt eine erhöhte Download Zeit und evtl. die Zeit die gebraucht wird um die Dateien zu mischen.

5.5 Etablieren einer einfachen scanning Routine

In dieser Arbeit wurde gezeigt das eine voll Automatische Lösung für das scannen von Chloroplasten in Pflanzen Genom Daten möglich und auch erfolgreich ist. Die hier verwendeten Skripte können frei verwendet und angepasst werden. Doch kann dies alles auch in einem kompletten Dockercontainer benutzt werden. Der chloroExtractorTeam Screening Container[61], kann verwendet werden um komplett automatisch die Daten von NCBI herunter zu laden, diese zu mischen (mit einem festen Seed) und dann den chloroExtractor und den fast-plast zu verwenden um diese Daten zu verarbeiten. Hierzu muss lediglich der Container gestartet werden und der run.sh Befehl mit der Passenden SRA Nummer gegeben werden. Dieser Container wird gerade verwendet um weitere 12393 Datensätze zu durchsuchen und Chloroplasten zu bauen. Diese Container ist sehr einfach zu benutzen, und alles was dafür gebraucht wird ist Docker[40] oder ein Programm welches Dockercontainer ausführen kann wie z.B. Singularity[49].

[61] `https://github.com/chloroExtractorTeam/screening_container`

5.6 GWAS

Es wurde eine GWAS Studie auf den 303 *A.thaliana* Chloroplasten durchgeführt, doch konnte dies nur auf zwei verschiedenen Eigenschaften berechnet werden. Hierzu gehört die Flowering Time bei 16°C sowie bei 10°C. Die restlichen Arapheno Traits konnten nicht berechnet werden. Dies ist vor allem der Fall da zu wenige Daten zur Verfügung stehen, sowohl von unserer Seite aus als auch von der Eigenschaften Seite aus. Eigenschaften wie Größen, Blütenbreite oder Form sind nicht gut genug Katalogisiert um eine geringe Datenmenge, wie sie hier benutzt wurde abzudecken. Dies heißt nicht das keinerlei Assoziation zwischen Chloroplasten Varianz und diesen Eigenschaften besteht. Dies zeigt lediglich dass noch mehr Daten benötigen werden um diese GWAS Studie zu beenden.

5.7 Struktur Varianz

Es wurde angenommen, dass ein kompletter Chloroplast für eine Struktur Varianz Analyse einen großen Vorteil bringt im Vergleich zu Illumina short Reads. Diese sind häufig zu kurz um große Invertierungen oder neu Anordnungen zu überspannen. Leider konnte diese Annahme nicht überprüft werden, da zu wenig Zeit vorhanden war. Als auch Programme, welche eine solche Analyse durchführen nicht erfolgreich benutzt werden konnten

5.8 Fazit und Zukunftsaussichten

Chloroplasten Genome können vielseitig verwendet werden und mit der immer steigenden Anzahl dieser Genome können mehr und mehr Analysen durchgeführt werden. Hier wurde eine Möglichkeit gezeigt wie man solche Chloroplasten Genome aus bereits existierenden Daten bauen kann. Dies ist aber nur möglich da alle Daten danke Open Science verfügbar waren. Es wurden 320 Chloroplasten aus bereits vorhandenen Daten erzeugt, die meisten aus *Arabidopsis thaliana* Daten, die vom 1001 Genom Projekt verwendet wurden. Es wurden aber auch 17 Neue Chloroplasten Genome erzeugt. Zudem wurde ein erster Ausblick auf die verschiedenen Analysen gegeben welche mit Chloroplasten Genomen möglich sind. Als einer der Entwickler des chloroExtractors, bin ich froh darüber wie der chloroExtractor im Vergleich mit anderen Programmen abgeschnitten hat. Doch zeigt mir dies auch, dass der chloroExtractor durchaus noch verbessert werden kann. So könnten verschiedene Features noch hinzugefügt werden, wie z.B. das finden und assemblieren von Mitochondrien Genomen, wie es bereits andere Programme versuchen. Je mehr Chloroplasten Genome mit der Zeit verfügbar werden, desto mehr Analysen können mit diesen Chloroplasten durchgeführt werden. So zeigte sich hier bei dem versuch einer GWAS, dass es bei zu wenigen Daten zu Problemen kommen kann. Diese Arbeit zeigt vor allem eines, es ist durch Automatisierung möglich Chloroplasten Genome aus Pflanzlichen Sequenzdaten zu bauen. Dank der Automatisierung ist hier lediglich Rechenpower von Nöten. So können in Zukunft noch viel mehr Chloroplasten Genome erzeugt werden.

6 Abbildungs- und Tabellenverzeichnis

List of Figures

List of Tables

7 Anhang

7.1 Dockercontainer

Table 9: **Dockercontainer** Alle erstellten Dockercontainer stehen zur freien Verfügung.

Programm	Dockerhub link
chloroExtractor	`https://hub.docker.com/r/chloroextractorteam/chloroextractor/` Build: b5uvjvdnbcyndhjngua85nv
fast-plast	`https://hub.docker.com/r/chloroextractorteam/fast-plast_docker/` Build: bgrmngfwpil4sk2kezi9f
NOVOPlasty	`https://hub.docker.com/r/chloroextractorteam/novoplasty_docker/` Build: bf9adepndze96bcnteyeabk
IOGA	`https://hub.docker.com/r/chloroextractorteam/ioga_docker/` Build: bwnf4xtzhohfcstqjqsvqvw
GetOrganelle	`https://hub.docker.com/r/chloroextractorteam/getorganelle_docker/` Build: bwt3bus2r7utsjpgrmjmjuc
Org.ASM	`https://hub.docker.com/r/chloroextractorteam/org.asm_docker/` Build: bmcx88c2d79orvuykgivy4q
Screening Container	`https://hub.docker.com/r/chloroextractorteam/screening_container/` Build: bdsankaqpukcgfmkmdq9yud

Table 10: **Git Links** Alle verwendeten Programme stehen zur freien Verfügung.

Programm	git link
chloroExtractor	`https://github.com/chloroExtractorTeam/chloroExtractor`
fast-plast	`https://github.com/mrmckain/Fast-Plast`
NOVOPlasty	`https://github.com/ndierckx/NOVOPlasty`
IOGA	`https://github.com/holmrenser/IOGA`
GetOrganelle	`https://github.com/Kinggerm/GetOrganelle`
Org.ASM	`https://git.metabarcoding.org/org-asm/org-asm`
Screening Container	`https://github.com/chloroExtractorTeam/screening_container`

7.2 Danksagung

Ich möchte allen Danken, welche mich in meiner Zeit als Student vor allem in den letzten Semestern unterstützt haben. Zunächst Danke ich dem kompletten CCTB, in dem ich in den letzten Semestern mit mehr als nur Kollegen zusammenarbeiten durfte. Hier möchte ich im besonderen meinen Betreuern Markus Ankenbrand und Jörg Schulz danken, welche sich die Zeit nehmen müssen diese Arbeit zu korrigieren und immer mit Rat zur Seite standen. Speziell möchte ich auch meinen ganz besonderen Dank an Frank Förster aussprechen, der uns mehr als nur beratend zur Seite stand und mehrere Nächste geopfert hat um unseren chloroExtractor zu verbessern und zu fixen. Auch möchte ich aber meiner kompletten Familie danken, meinen Eltern Roland und Maria, sowie meinen beiden Brüdern Florian und Christopher, sowie meinem Bruder im Geiste Martin "Löwe" Piekar, sowie natürlich den Damen welche es mit diesen drei Chaoten aushalten, die da wären Vanja, Julia und Charlotte. Natürlich danke ich auch all meinen anderen Freunden welche es mit mir all die Jahre ausgehalten haben, und dies planen noch weiter zu tun, so wie der World of Warcraft Community, der Gilde Maestri delle Arte und dem Raid Invictus, welche mich immer gut unterhalten haben, und das ein oder andere mal evtl. auch zu viel von der Arbeit abgehalten haben. Auch geht mein Dank an Kaffee, vielen dank für Substitution von Schlaf mit Koffein.

7.3 Skripte

Alle verwendeten Skripte können gefunden werden unter:
`https://github.com/chloroExtractorTeam/chloroplast_landscape/tree/Documentation_SPfaff` Zudem sind diese auf der beiliegenden CD zu finde.

7.4 Tabellen

Alle Ergebniss Tabellen können eingesehen werden unter:
`https://github.com/chloroExtractorTeam/chloroplast_landscape/tree/Documentation_SPfaff` Zudem sind diese auf der beiliegenden CD zu finde.

Eigenständigkeitserklärung

ERKLÄRUNG gemäß ASPO vom 5.8.2009 § 23 Abs. 10

Hiermit versichere ich, dass ich vorliegende Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt habe.

Würzburg, August 6, 2018 Simon Pfaff