A large-scale computational framework for comparative analyses in population genetics and metagenomics

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Eun-Cheon Lim

aus Südkorea

Tübingen

2016

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	04. May. 2016
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Detlef Weigel
2. Berichterstatter:	Prof. Dr. Daniel Huson

Erklärung

Hiermit erkläre ich, dass ich diese Arbeit selbstständig und nur mit den angegebenen Hilfsmitteln angefertigt habe, und dass alle Stellen, die im Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angabe der Quellen kenntlich gemacht sind. Die Arbeit wurde bislang nicht an einer Hochschule zur Erlangung eines akademischen Grades eingereicht.

Tübingen, 2016 Eun-Cheon Lim

Zusammenfassung

In der Populationsgenetik werden die räumlichen und zeitlichen Verteilungen von genetischen Varianten in Individuen einer Population untersucht. Über die Generationen ändert sich die Frequenz von Genen und Allelen. Die Auswirkungen der durch diese evolutionären Mechanismen gebildete Diversität zeigt sich auf verschiedenen Stufen biologischer Organisation, von einzelnen Molekülen bis hin zu gesamten Organismen. Sind Eigenschaften betroffen welche einen Einfluss auf die Überlebens- und Reproduktionsrate haben, werden die zugrundeliegenden Allele mit höherer Wahrscheinlichkeit in die nachfolgende Generation übetragen werden. Allele mit positiver Auswirkungen auf die Fitness eines Organismus könnten sich so in einer Population verbreiten.

Zufällige Mutationen sind eine Quelle für neue Allele in einer Population. Die zugrundeliegenden Veränderungen der DNA-Sequenzen können durch Fehler bei der DNA-Replikation oder von DNA-Reparaturmechanismen, sowie Insertionen und Deletionen von mobilen genetischen Elementen entstehen. In sich sexuell fortpflanzenden Organismen sorgt genetische Rekombination für eine Vermischung der Allele auf den Chromosomen. Obwohl die Allelfrequenzen nicht verändert werden, entstehen dadurch neue Kombinationen von Allelen. Auf der molekularen Ebene können Genloci durch Mutationen an Aktivität gewinnen oder funktionslos werden, was wiederum eine Auswirkung auf den entstehenden Phänotyp und die Überlebensfähigkeit des Organismus hat. Trotz der höherer Verbreitung neutraler Mutationen, kann das Ansammeln von kleinen Veränderungen im Laufe der Zeit die Fitness beeinflussen und weiter der Evolution beitragen.

Das Ziel dieser Arbeit war es ein Rahmenwerk für die vergleichende Analyse großer genomischer Datensets zur Verfügung zu stellen. Im Besonderen für Datensätze mit vielen Individuen einer Spezies wie im *1001 Genomes Project* (*Arabidopsis thaliana*), im *1000 Genomes Project* (*Homo sapiens*) sowie in metagenomische Datensätzen. Für die folgenden Problemstellungen wurden Algorithmen entwickelt: 1) Fehlerkorrektur und Verbesserung der effektiven *Coverage* von genomischen Rohdaten (Trowel), 2) multiple Gesamt-Genomalinierungen (*whole genome alignments;* WGAs) und die Detektion kleiner Unterschiede innerhalb einer Population (Kairos), 3) Identifikation struktureller Varianten (SV) (Apollo), und 4) Klassifikation von Mikroorganismen in metagenomischen Datensätzen (Poseidon). Diese Algorithmen nehmen keine Interpretation biologischer Rohdaten vor sondern stellen Ausgangspunkte für biologische Hypothesen zur Verfügung.

Auf Grund der Fortschritte in verteiltem und paralellem Rechnen nutzen viele moderne Bioinformatikalgorithmen Paralellisierung auf CPUs oder GPUs. Diese erhöhte Rechenkapazität erlaubt es uns größere und komplexere Probleme zu lösen. Allerdings machen diese technische Fortschritte allein es noch nicht möglich, sehr große Datensätze zu nutzen und bringen auch keine Antworten auf biologische Fragen. Um von diesen Fortschritten zu profitieren und hochqualitative Informationen aus Rohdaten extrahieren zu können, sind gut durchdachte Datenstrukturen und Algorithmen notwendig. Für die Populationsgenetik sollte eine effiziente Repräsentation eines Pan-Genoms und dazugehöriger Formeln geschaffen werden. Zusätzlich zu einer solchen Repräsentation spielen Sequenzalalinierungen eine entscheidende Rolle im Lösen biologischer Probleme wie der Berechnung von Allelfrequenzen, der Detektion seltener Varianten, der Assoziation von Genotypen und Phänotypen und Inferenz von Kausalität bezüglich bestimmter Krankheiten. Um Mutationen in einer Population zu detektieren wird die konventionelle Alinierungsmethode verbessert da mehrere Genome gleichzeitig aliniert werden.

Obwohl die Anzahl vollständiger Genomsequenzen stetig gestiegen ist, ist die Analyse dieser großen und komplexen Datensätze immer noch schwierig. Die Hochdurchsatz-Sequenzierung (*Next Generation Sequencing*; NGS), die ein präziseres und detaillierteres Bild der Genomik geliefert hat, ist einer der großen Fortschritte in der Biotechnologie. Die Länge und Genauigkeit der Sequenzier-

Abschnitte (*Reads*) hat sich so weit verbessert, dass in Zukunft wahrscheinlich ein vollständiges Genom von nur einer einzelnen Zelle als Ausgangsmaterial rekonstruiert werden kann. Obwohl die wichtigsten Schritte zur Realisierung von Sequenzierungsfortschritten eine Domäne der Verfahrenstechnik sind, haben auch die Informatik und Computertechnik die Qualität der Sequenzen entscheidend beeinflusst. Sequenzierdaten enthalten Fehler in Form von Substitutionen, Insertionen oder Deletionen von Nukleotiden. Außerdem ist die Länge der erzeugten *Reads* deutlich kürzer als die eines vollständigen Genoms. Diese Schwierigkeiten können durch Fehlerkorrekturen und Genomassemblierung verringert werden, wodurch nachfolgende Analysen genauer werden.

Programme zur Alinierung kurzer *Reads* waren bisher die wichtigste Methode um genetische Mutationen zu detektieren. Da nun duch neue Technologien häufig längere *Reads* oder auch Contigs verfügbar sind, werden Kartierungsmethoden benötigt die sich an langen Ähnlichkeiten orientieren und sich nicht von kurzen lokalen Übereinstimmungen fehlleiten lassen. Die Parameter für Programme zur Alinierung von kurzen *Reads* welche nichtübereinstimmende Basen und das Eröffnen und Verlängern von Lücken bestrafen, sind nicht direkt auf die Alinierung längerer *Reads* anwendbar. Alternativ können WGA-Algorithmen verwendet werden, die das Alinierungsproblem in einem längeren Kontext lösen und dadurch essentielle Daten für vergleichende Studien liefern. Allerdings haben bisherige WGA-Algorithmen noch Probleme in der praktischen Anwendung für die Populationsgenetik wegen ihrer hohen Zeit- und Speicherkomplexität. Außerdem wurde der Definition idealer Datenformate für Anwendungen der komparativen Genomik nur wenig Aufmerksamkeit gewidmet.

Um Datensätze großer Populationen verarbeiten zu können sollten Algorithmen für multiple Sequenzalinierung (MSA) mit WGA-Methoden zur multiplen Gesamtgenomalinierung (MWGA) kombiniert werden. Obwohl bereits viele MWGA-Methoden vorgestellt wurden, wurde ihre Genauigkeit noch nicht aussagekräftig überprüft. Vielmehr lieferten Qualitätskontrollen sehr unterschiedliche Ergebnisse, abhängig von der Auswahl von Organismen und verwendeten Sequenzen. Ein noch größeres Problem ist die ungenaue Beschreibung von Experimenten zur Messung der Funktionalität von MWGA-Methoden. Daher war es schwierig die multiplen Alinierungs-Ergebnisse zu interpretieren. Ich beschreibe in dieser Arbeit eine deutlich umfassendere Methode um die Genauigkeit eines MWGA-Algorithmus zu bestimmen. Sie macht von vorab bekannten Positionen der Varianten Gebrauch wozu Simulationen und standardisierte Statistiken herangezogen werden.

Die Metagenomik untersucht die genetische Zusammensetzung einer (oft hauptsächlich mikrobiellen) natürlichen Organismen-Gemeinschaft. Sie ist unabhängig von der Kultivierung einzelner Mikroben und liefert auch quantitative Informationen zur Zusammensetzung der Gemeinschaft. Während Proben aus der Umwelt ein natürlicheres Ausgangsmaterial liefern ist gleichzeitig auch die Komplexität der Analysen deutlich höher: die Anzahl der enthaltenen Arten kann sehr groß sein, so dass nur ein Bruchteil der Genome tatsächlich analysiert wird. Ich stelle einen Algorithmus vor, Poseidon, der *Reads* zur taxonomischen Identifikation mit Arten-genauer Auflösung zuordnet und damit hilft deren relative Häufigkeit in einer Probe zu quantifizieren. Die Interaktionen zwischen Bakterien kann Konflikte und auch Kooperationen hervorrufen. Die spezielle Mischung unterschiedlicher Artem kann daher eine Reihe funktionaler Anpassungen an eine bestimmte Umgebung aufzeigen. Die Zusammensetzung der Arten könnte durch biotische oder abiotische Faktoren verändert werden, was im Kontext eines Krankheitsbildes zu einer Veränderung der Anfälligkeit eines Wirts bezüglich eines bestimmten Erregers führen kann. Daher sind die genaue Quantifizierung von Arten und die Entschlüsselung ihrer funktionalen Rolle in einer bestimmten Umgebung grundlegend für metagenomische Studien.

Zusammenfassend stelle ich in dieser Arbeit fortgeschrittene bioinformatische Methoden, Trowel, Kairos, Apollo und Poseidon vor. Trowel korrigiert Fehler in Sequenzabschnitten mit Hilfe von *k*-mer Informationen von hoher Qualität. Kairos führt die Alinierung einer Sequenz zu multiplen Genomen

einer Art durch. Apollo charakterisiert genomweit genetische Varianten basierend auf den Alinierungen von Kairos, und erfasst sowohl Punktmutationen als auch große strukturelle Varianten. Poseidon ordnet metagenomische Datensätze taxonomischen Identifikatoren zu. Auch wenn keine spezifischen biologischen Fragestellungen beantwortet werden, wird die Basis für zukünftige Fragen geschaffen.

Summary

Population genetics is the study of spatio-temporal genetic variants among individuals. Its purpose is to understand evolution: the change in frequency of alleles over time. The effects of these alleles are expressed on different levels of biological organization, from molecular complexes to entire organisms. Eventually, they will affect traits that can influence the survival and reproduction of organisms. Fitness is a probability of transferring alleles to subsequent generations with respect to successful survival and reproduction. Due to differential fitness, any phenotypic properties that confer beneficial effects on survival and reproduction may presumably become prevalent in a population.

Random mutations introduce new alleles in a population. The underlying changes in DNA sequences can be caused by replication errors, failures in DNA repair processes, or insertion and deletion of transposable elements. For sexual organisms, genetic recombination randomly mixes up the alleles in chromosomes, in turn, yielding a new composition of alleles though it does not change the allele frequencies. On the molecular level, mutations on a set of loci may cause a gain or loss of function resulting in totally different phenotypes, hereby influencing the survival of an organism. Despite the dominance of neutral mutations, the accumulation of small changes over time may affect the fitness, and further contribute to evolution.

The goal of this study is to provide a framework for a comparative analysis on large-scale genomic datasets, especially, of a population within a species such as the 1001 Genomes Project of *Arabidopsis thaliana*, the 1000 Genomes Project of humans, or metagenomics datasets. Algorithms have been developed to provide following features: 1) denoising and improving the effective coverage of raw genomic datasets (Trowel), 2) performing multiple whole genome alignments (WGAs) and detecting small variations in a population (Kairos), 3) identifying structural variants (SVs) (Apollo), and 4) classifying microorganisms in metagenomics datasets (Poseidon). The algorithms do not furnish any interpretation of raw genomic data but provide analyses as basis for biological hypotheses.

With the advances in distributed and parallel computing, many modern bioinformatics algorithms have come to utilize multi-core processing on CPUs or GPUs. Having increased computational capacity allows us to solve bigger and more complex problems. However, such hardware advances do not spontaneously give rise to the improved utilization of large-size datasets and do not bring insights by themselves to biological questions. Smart data structures and algorithms are required in order to exploit the enhanced computing power and to extract high quality information. For population genetics, an efficient representation for a pan genome and relevant formulas should be manifested. On top of such representation, sequence alignments play pivotal roles in solving biological problems such that one may calculate allele frequencies, detect rare variants, associate genotypes to phenotypes, and infer causality of certain diseases. To detect mutations in a population, the conventional alignment method is enhanced as multiple genomes are simultaneously aligned.

The number of complete genome sequences has steadily increased, but the analysis of large, complex datasets remains challenging. Next Generation Sequencing (NGS) technology is considered one of the great advances in modern biology, and has led to a dramatically more precise and detailed understanding of genomes and their activities. The contiguity and accuracy of sequencing reads have been improving so that a complete genome sequence of a single cell may become obtainable from a sequencing library in the future. Though chemical and optical engineering are main drivers to advance sequencing technology, informatics and computer engineering have significantly influenced the quality of sequences. Genomic sequencing data contain errors in forms of substitution, insertion, and deletion of nucleotides. The read length is far shorter than a given genome. These problems can be alleviated by means of error corrections and genome assemblies, leading to more accurate downstream analyses.

Short read aligners have been the key ingredient for measuring and observing genetic mutations using Illumina sequencing technology, the dominant technology in the last decade. As long reads from newer methods or assembled contigs become accessible, mapping schemes capturing long-range context, but not lingering in local matches should be devised. Parameters for short read aligners such as the number of mismatches, gap-opening and -extending penalty are not directly applicable to long read alignments. At the other end of the spectrum, whole genome aligners (WGA) attempt to solve the alignment problem in a much longer context, providing essential data for comparative studies. However, available WGA algorithms are not yet optimized concerning practical uses in population genetics due to high computing demands. Moreover, too little attention has been paid to define an ideal data format for applications in comparative genomics.

To deal with datasets representing a large population of diverse individuals, multiple sequence alignment (MSA) algorithms should be combined with WGA methods, known as multiple whole genome alignment (MWGA). Though several MWGA algorithms have been proposed, the accuracy of algorithms has not been clearly measured. In fact, known quality assessment tools have yielded highly fluctuating results dependent on the selection of organisms, and sequencing profiles. Of even more serious concern, experiments to measure the performance of MWGA methods have been only ambiguously described. In turn, it has been difficult to interpret the multiple alignment results. With known precise locations of variants from simulations and standardized statistics, I present a far more comprehensive method to measure the accuracy of a MWGA algorithm.

Metagenomics is a study of the genetic composition in a given community (often, predominantly microbial). It overcomes the limitation of having to culture each organism for genome sequencing and also provides quantitative information on the composition of a community. Though an environmental sample provides more natural genetic material, the complexity of analyses is greatly increased. The number of species can be very large and only small portions of a genome may be sampled. I provide an algorithm, Poseidon, classifying sequencing reads to taxonomy identifiers at a species resolution and helping to quantify their relative abundances in the samples. The interactions among individual bacteria in a certain population can result in both conflict and cooperation. Thus, a mixture of diverse bacteria species shows a set of functional adaptations to a particular environment. The composition of species would be changed by distinct biotic or abiotic factors, which may lead to a successive alteration in susceptibility of a host to a certain disease. In turn, basic concerns for a metagenomics study are an accurate quantification of species and deciphering their functional role in a given environment.

In summary, this work presents advanced bioinformatics methods: Trowel, Kairos, Apollo, and Poseidon. Trowel corrects sequencing errors in reads by utilizing a piece of high-quality *k*-mer information. Kairos aligns query sequences against multiple genomes in a population of a single species. Apollo characterizes genome-wide genetic variants from point mutations to large structural variants on top of the alignments of Kairos. Poseidon classifies metagenomics datasets to taxonomy identifiers. Though the work does not directly address any specific biological questions, it would provide preliminary materials for further downstream analyses.

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor, Professor Detlef Weigel, for his sincere and continuous supports, patience, and caring during my PhD study, and for providing an excellence research environment. I have been inspired by his active involvement in science and passion for learning. I am grateful to my second supervisor, Professor Daniel Huson, and a committee member, Dr. Richard Neher, and Dr. Stefan R. Henz for the inspiration of this study, and discussion. I am also grateful to Jonas Müller, Dr. Ilja Bezrukov, Anna-Lena Keller, Dino Jolic, and Dr. Rebecca Schwab for translating the abstract of the dissertation into German.

I wish to express my sincere thanks to Dr. Jörg Hagmann, Dr. Stefan R. Henz, and Jonas Müller for their warm introduction to Tübingen. I would have never been able to get accustomed to this new place such quickly without their helps. I am deeply grateful to Jörg, Jonas, and Stefan for improving the Trowel manuscript and for helping me to understand biology. My special gratitude goes to Dr. Sang-Tae Kim, and Dr. Eunyoung Chae who kindly answered questions about basic biology and suggested directions for my PhD study at the initial stage. Dr. Sang-Tae Kim also helped to review the Trowel paper. I wish to express my deepest thanks to Maricris Zaidem, Dr. Eshita Sharma, Dr. Subhashini Muralidharan, Diep Thi Ngoc Tran, Dino Jolic, Dr. Danelle Seymour, Dr. Xi Wang, Dr. Wangsheng Zhu, Dr. Wanyan Xi, Dr. Congmao Wang, Dr. Chang Liu, Prof. Markus Schmid, Leily Rabbani, Anna-Lena Keller, and Marion Dubarry for invaluable discussions and helps. I also thank Andre Noll for resolving computational problems in computing nodes. I really appreciate Dagmar Sigurdardottir, Dr. Rebecca Schwab and Hülya Wicher for their administrative supports.

I am grateful to Prof. Pavel A. Pevzner for providing an open bioinformatics course at the Coursera platform. I really enjoyed the open course materials and great feedbacks from the researchers all over the world. They resolved my academic thirsty in bioinformatics algorithms and data structures. This learning opportunity has greatly accelerated my PhD study. I also thank Dr. Sangtae Kim, who studied in Prof. Pevzner's lab and I met at the Cold Spring Harbor laboratory, for his critical comments on this study and discussions. In addition, I would like to express my gratitude to Dr. Daehwan Kim currently in the Prof. Steven Salzberg's Lab for invaluable discussion and the inspiration for this work.

I would like to express my sincere gratitude to Dr. Jared Simpson and Alexander Bowe for their straightforward explanation about the FM-index and discussions. I am also grateful to Joseph Gentle for sharing his understanding about the rope data structure. I thank Rachid Ounit for discussion about the taxonomy classification problem. I am exceptionally grateful to Dr. Hyunmin Kim, and Dr. Jihye Kim for sharing their insights, and for inspiring and guiding fresh researchers toward a right direction.

Finally, I regret that I did not finish my PhD study before Aug., 2015 since I missed the chance to show my last graduation event to my father who wholeheartedly had been happy with my postgraduate educations. My beloved father may gladly see my academic progress from the heaven. This work may suffice his unfulfilled dream of education interrupted by financial difficulties when he was young. I would like to express my exceptional appreciation to his unconditional love and dedication for my whole life. I am deeply grateful to my mother for her sacrifice and devotion for me. My sincere gratitude also goes to my younger sister for caring mother during the difficult time and for supporting emotionally throughout this academic journey. I would not be able to finish this study without their interests and emotional supports.

Contents

0. Prologue

0.1 Evolution?

Diverse organisms with different phenotypes may live in the same place, experiencing contemporary events. The competition between individuals of the same species or different species foraging for the same nutritional sources in the ecosystem may influence individual survival and reproduction, or fitness. The progenitors of each species may have had to habituate given a hierarchical ecology food chain or different types of persistent biological interaction such as symbiosis of mutualism, commensalism, or parasitism, leaving off several inscriptions on each individual genome of coevolved lifeforms.

The modern understanding about evolution began with the theory of catastrophism, proposed by French naturalist [G. Cuvier, 1815], who is known as Father of Paleontology. The idea that catastrophic events may cause mass extinctions of species initiated from the observation of skeletons of Indian and African elephants, and mammoth and mastodon fossils. Cuvier conceived that new species could have appeared after periodic catastrophic events. He had a strong objection to the theories of evolutionary change based on observations such as mummified cats and ibises from thousands of years not being distinctly different from contemporary counterparts. He criticized the concepts of evolution proposed by Jean-Baptiste de Lamarck, and Geoffroy Saint-Hilaire. He rather attempted to comply with a religious belief that lifeforms will not change over time and that the divinity had designed all cycles of birth and extinction.

Lamarck proposed a theory of inheritance of acquired traits [J. Lamarck, 1809]. He thought that environmental changes may have led to a development of new traits for organisms, which increases their survival. For instance, the use of an elongated neck of a giraffe ancestor to reach for higher leaves may change the length of the neck and such acquired traits would be accumulated over time to manifest in giraffes as we know them today. In contrast to this positive use-pattern, he believed furthermore that frequent disuses of an organ may negatively affect the traits of the organ, leading to a degeneration. He did not agree with the idea of extinction, but instead, he proposed that lifeforms alter their structures from simple to complex ones. Though his arguments were based on less rigorous attempts in logical reasoning and, in fact, misguided, he provided a new perspective about evolution in spite of following prevailing religious beliefs.

The mechanism of heredity remained unexplained until Gregor Mendel revealed the rule of inheritance [J.G. Mendel, 1866]. His experiments of 29,000 pea plants were carried out in an isolated monastery. He observed seven characteristics of pea plants, and uncovered the law of segregation, whereby two alleles for each trait are segregated during meiosis and a single gamete conveys one of the alleles, and the law of independent assortment such that each of alleles is transferred independently from parents to their progeny. Though his finding could explain the distinct characteristics of qualitative differences and the secrets of heredity, his studies were not accepted by the scholars during his lifetime [P.J. Bowler, and I.R. Morus, 2005], including Charles Darwin, who wrote *On the Origin of Species* [C.R. Darwin, 1859].

Darwin strengthened the concept of common ancestors from which all the living creatures have descended over time. The theory of natural selection was independently developed by Darwin and Alfred Russel Wallace, and they jointly presented their theories [C.R. Darwin, and A.R. Wallace, 1858]. Although Darwin's theory of evolution demonstrated irresistible evidence, the rejection of Lamarckian inheritance did not fall into place at once. An important next step was the advent of population genetics, which finally led to the modern evolutionary synthesis [W.J. Bock, 1981]. Ronald Fisher together

with J.B.S. Haldane, and Sewall Wright combined Mendelian genetics with Darwin's natural selection theory, leading to the foundation of population genetics [U. Kutschera, and K.J. Niklas, 2004]. Their studies presented that Mendelian genetics is consistent with gradual evolution over time by the process of natural selection.

Fisher described now the collective actions of discrete genes directly contributes to continuous variation, and that natural selection can affect the allele frequency in a population, starting in 1918. J.B.S. Haldane established a mathematical formalism for natural selection in a series of scientific papers, named *A Mathematical Theory of Natural and Artificial Selection*, whereby the observations that changes in allele frequencies have a certain direction and rate was demonstrated. He also pioneered the application of maximum likelihood for estimating human linkage map [J. Bell, and J.B.S. Haldane, 1937], and the method to calculate mutation rates of a human gene [J.B.S. Haldane, 1935]. Sewall Wright formulated the theory of genetic drift, which denotes the changes in allele frequency caused by the random events of births, deaths, and segregations, rather than by selection. He visualized the relation between traits and fitness as adaptive or fitness landscapes. The horizontal axes represent either the allele frequency or the average phenotype of a population while the vertical axis denotes the mean population fitness. In turn, natural selection can be seen as pushing individuals up different hills. Since genetic drift represents stochastic cumulative processes, the random movements along the gradient fitness would be observed. In an adaptive landscape of a sufficiently small population, genetic drift can move a population by chance from the current peak to another higher peak [S. Wright, 1932]. The developed mathematical methods by Fisher, Wright and Haldane provide for a framework that describes the evolutionary changes in a population.

The origin of species yet had to be fully understood. Theodosius Dobzhansky brought the first reasonable answer to speciation based on the observation of fruit fly populations. In his 1937 book, *Genetics and the Origin of Species*, he described reproductive isolation between lineages caused by interaction among spontaneous neutral mutations that do not have harmful effects on the survival and reproduction of each species [T. Dobzhansky, 1937]. Over time, the isolation would increase the distinct genetic profile of each population, leading to an emergence of a set of genes incompatible with other populations. In turn, different species may not be able to successfully inter-mate anymore.

0.2 Toward complete genomics

0.2.1 Sequencing technology

The language of evolution, which stores and transfers the genetic information to new cells and successive generation, is deoxyribonucleic acid (DNA). It is composed of one of four nucleotides, or bases: Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). These bases have the complementary characteristics derived from the hydrogen bond, which is the electrostatic attraction between polar groups occurring when a hydrogen atom is strongly bound to an electronegative atom. Thus the hydrogen bonds of each base pair (guanine-cytosine and adenine-thymine) preserve a stable helical structure, the DNA double helix. The double helix structure was discovered by James Watson, Francis Crick, Maurice Wilkins and Rosalind Franklin in 1953 [J.D. Watson, and F.H.C. Crick, 1953].

The first practical method for sequencing DNA was devised by Fredrick Sanger and colleagues in 1977 [F. Sanger et al. 1977]. Chain termination sequencing, widely known as Sanger sequencing, applied modified nucleotides to a DNA chain to randomly terminate extension. Thus, partial DNA fragments of the template DNA, ending at specific nucleotides, are produced by these terminations. The application of gel electrophoresis to these partial copies allows to sort out the fragments by length, and to obtain the sequences of the fragment from the gel.

The sequence of DNA base pairs in the original molecule from which the individual sequences are derived could be reconstructed by shotgun sequencing, or by serial sequencing, by means of primer walking [A.C. Chinault, and J. Carbon, 1979]. The primer walking method begins by sequencing one DNA fragment, and then designing new primers, from which the next round of sequencing is initiated. Until the DNA fragment of interest is fully identified, the "walking" keeps going on. In shotgun sequencing, DNA is randomly split into small fragments and the sequence of each fragment is identified by gel electrophoresis. A computer program finally combines random DNA sequences based on their overlaps [R. Staden, 1979]. The first fully-automated Sanger sequencing instrument was introduced by Applied Biosystems, based on innovations by Tim Hunkapiller and Leroy Hood [J.M. Prober et al., 1987]. Technically, gel electrophoresis and radio-activation of nucleotides are replaced by the capillary tubes and florescent nucleotides [L.M. Smith et al., 1985].

Sanger sequencing relies on amplification of a DNA template with a primer binding site, and purification to obtain clear signals. The next generation sequencing (NGS) technology, avoiding manual cloning, and enabling mixture of DNA fragment was introduced. Unlike Sanger sequencing, NGS treats each DNA fragment independently, allowing for parallelized sequencing. In NGS, DNA molecules are randomly sheared, amplified, and then sequenced. The first NGS technology, Massively Parallel Signature Sequencing, or MPSS, was developed by Lynx Therapeutics for RNA expression profiling [S. Brenner et al., 2000]. The first commercial NGS instrument for DNA sequencing was developed by 454 Life Sciences [M. Margulies et al., 2005]. The method is called pyrosequencing, which detects luciferase-induced light signals at each addition of one of four bases. The most successful NGS technology, however, has been that of Solexa, which was acquired by Illumina in 2007 just one year after the first Solexa sequencer was announced. Illumina sequencing creates on a modified microscope slide thousands to millions of DNA clusters, each of which consisting of about a thousand identical DNA molecules [E.R. Mardis, 2008]. In every round of the sequencing-by-synthesis (SBS) process, a fluorescent base is added and imaged on its solid support.

A newer, very promising NGS technology is Single Molecule Real Time Sequencing (SMRT) [M.J. Levene et al., 2003]. This technology utilizes zero-mode waveguides (ZMWs) to detect the characteristically colored-fluorophore dyed at a single molecule. The method directly observes the activity of DNA polymerase such that when a base is processed by the polymerase, incorporation is detected by a fluorophore lingering longer in a ZMW. The first SMRT experiment based on a reference sequence was demonstrated in 2009 [J. Eid et al., 2009]. This prototype sequencer contained 3000 ZMWs to enable parallel DNA sequencing. Pacific Biosciences started marketing the first SMRT machine, PacBio RS, in 2010 [GenomeWeb, 2010]. Due to the long read length, up to 40 kilo-bases, with advanced chemistry [PacificBiosciences, 2014], they started to fill the missing sequences in the reference assemblies of complex genomes such as of human [M.J. Chaisson et al., 2015].

0.2.2 Genome assemblers

The reconstruction of a genome sequence from a set of reads is known as *de novo* assembly. The first-generation assemblers followed a greedy strategy such that a read is joined if it overlaps with another read [R. Staden, 1979]. The early genome assemblers were even not automated, requiring human inference and only applicable to small genomes. Modern *de novo* assemblers are developed from the graph theory [H. Peltola et al., 1984]. One of the branches is the overlap graph and its strategy is known as overlap-layout-consensus (OLC) [J.D. Kececioglu, and E.W. Myers, 1993]. The formulation of a graph representation started from an NP-hard Shortest Common Superstring problem. In the overlap graph, each fragment read is a vertex and an edge between two vertices are created if they overlap significantly. In the layout phase, the relative location of every vertex is calculated. The final consensus phase determines the base at each position based on the multi-alignment if the coverage of the alignment is two or more. OLC structure is reformulated to the string graph by E.W. Myers in 2005 [E.W. Myers, 2005]. The new string graph formulation contained a transitive reduction phase, which

removes transitive edges from an overlap graph. Still, detecting overlaps was the major performance bottleneck of OLC approaches.

Yet another approach, which is known as de Bruijn graph assembly, is pioneered by P.A. Pevzner [P.A. Pevzner, 1989]. The term, de Bruijn graph, is named after Nicolaas de Bruijn who studied a combinatorial problem of certain cycles of digits 0 or 1 [N.G. de Bruijn, 1946]. Unlike OLC methods, a de Bruijn graph could implicitly build the overlap information. In a de Bruijn graph, each read is broken up to consecutive *k*-mers, subsequences of fixed length k, leading to continuous overlapping (*k*-1) bases. Each *k*-mer represents an edge connecting two (*k*-1)-bp prefix and suffix. The sequence graph, which stores position information in each edge, was proposed to deal with shotgun sequencing [R.M. Idury, and M.S. Waterman, 1995]. P.A. Pevzner suggested an Euler-DB algorithm to find Eulerian Superpaths from a de Bruijn graph and addressed that sequencing error correction can reduce the number of false vertices, simplifying the graph [P.A. Pevzner et al., 2001]. The algorithms, which can practically assemble NGS datasets appeared in 2008: EULER-SR [M.J. Chaisson, and P.A. Pevzner, 2008], and Velvet [D.R. Zerbino, and E. Birney, 2008].

Though de Bruijn graph approaches had reduced the computational time to find overlaps, the memory requirements for large genomes, which may need billions of vertices, were unhandled. T.C. Conway and A. Bromage suggested the use of a succinct representation, sparse bitmap [T.C. Conway, and A. Bromage, 2011]. The String Graph Assembler (SGA), exploited the FM-index, which gains compressibility from the Burrows-Wheeler transform [J.T. Simpson, and R. Durbin, 2012]. J. Pell and his colleagues suggested the use of a probabilistic data structure, bloom filter [B.H. Bloom, 1970], to encode *k*-mers in a de Bruijn graph [J. Pell et al., 2012]. However, this package did not contain a genome assembler. R. Chikhi and G. Rizk provided the first practical genome assembler, Minia, which applies the bloom filter [R. Chikhi, and G. Rizk, 2013]. This algorithm finished assembling short reads of human genome using only 5.7 Gb of memory.

0.2.3 Combination of sequencing technology and genome assemblers

The first complete genome of a bacterium, *Haemophilus influenzae* of 1,830,137 base pairs was reported by the combination of capillary shotgun sequencing and assembly programs [R.D. Fleischmann et al., 1995] In the following, more model organisms were sequenced such as yeast *Saccharomyces cerevisiae* [A. Goffeau et al., 1996], *Escherichia coli* [F.R. Blattner et al., 1997], *Caenorhabditis elegans* [C. elegans Sequencing Consortium, 1998], *Drosophila melanogaster* [M.D. Adams et al., 2000], and *Arabidopsis thaliana* [Arabidopsis Genome Initiative, 2000]. The successes in obtaining complete sequences of key model organisms convinced researchers to put further efforts on sequencing the human genome and provided practical methods for the sequencing plan [E.S. Lander et al., 2001].

The Human Genome Project (HGP) was initiated in 1990 and funded by the U.S. Department of Energy (DOE) and the National Institutes of Health (NIH) [J.D. Watson, and R.M. Cook-Deegan, 1991]. The HGP utilized a hierarchical shotgun sequencing, where an amplified genome is scattered into large pieces of size 50-200 kb, and the positions and the sequences of each chunk are identified later. Physical maps of the human genome were built to identify relative positions of DNA fragments by means of restriction mapping [K. Danna, and D. Nathans, 1971], Fluorescent in situ hybridization (FISH) [P.R. Langer-Safer et al., 1982], or Sequence tagged site (STS) mapping [M. Olson et al., 1989]. Following the shotgun sequencing practice, contigs were generated based on overlapping bacterial artificial chromosome (BAC) library for each chromosome with the fourfold sequence coverage on average. Celera Genomics, led by J.C. Ventor, independently initiated a private human genome project. Celera produced their own capillary sequencing dataset of five human genomes, and designed a computational workflow to assemble the private datasets and HGP BAC contigs into the human ge-

nome sequence. The first draft of complete human genome sequence was published in 2001 [J.C. Venter et al., 2001, E.S. Lander et al., 2001].

After the successful sequencing of the human genome, more genomes have been sequenced such as that of mouse [Mouse Genome Sequencing Consortium, 2002], short grain rice *Oryza sativa* [J. Yu et al., 2002], chicken [International Chicken Genome Sequencing Consortium, 2004], chimpanzee [Chimpanzee Sequencing and Analysis Consortium, 2005], dog [K. Lindblad-Toh et al., 2005], cat [J.U. Pontius et al., 2007], horse [C.M. Wade et al., 2009], cow [The Bovine Genome Sequencing and Analysis Consortium, 2009], orangutan [D.P. Locke et al., 2011], pig [A.M. Martien et al., 2012], gorilla [A. Scally et al., 2012], tigers, lions, and snow leopard [Y. Cho et al., 2013], many birds [E.D. Jarvis et al., 2014], and most recently waterbear [T.C. Boothby et al., 2015]. NGS technologies can be used not only for reconstructing genomes and genome resequencing, but also as a counting technology for transcriptomic profiling, DNA-protein interaction, and epigenome studies [F.A. San Lucas et al., 2015, C.A. Meyer, and X.S. Liu, 2014, J.L. McClay et al., 2015]. The cost to sequence a single genome has gradually decreased and with modern technology, a more complete genetic landscape of many more organisms will be clearly delineated in the near future.

0.3 Challenges

0.3.1 Representation of genomic datasets

First, in the era of NGS technology, one of the biggest challenges is the representation of a large volume of genomic datasets. From a bird's eye view, learning is a process of recognizing patterns in heterogeneous phenomenon. To "learn" the lives in the universe, biologists would be interested in finding similarity and differences among some objects if one greatly simplified their learning. The objects could be molecules, genes, proteins, cells, a genome, a population, or ancient genomes. The object of interest in this study is a population of eukaryotes, or microorganisms. Then, a cornerstone of this study would be the representation of a genome, or even a pan genome, which reflects the full gene sets of all the strains in a single clade, or in this study, a species [H. Tettelin et al., 2005]. The identification of similarity and differences among genomes is a pattern matching problem, thus the representation should support efficient string operations. A possible data structure would be either a bloom filter based graph (refer to section 0.2.2) or an FM-index [P. Ferragina, and G. Manzini, 2000].

Many indexing schemes, where genomic sequences are stored and retrieved, have been proposed for comparative studies (refer to Chapter 1). Among them the FM-index offers markedly distinct and efficient properties for a pan genome notation. The genome of the same species may share highly conserved DNA sequences, allowing for a compact representation when applied with compression schemes. A successful deployment of this index leads to linear time sequence alignments and detection of variations across thousands of individual genomes [H. Li, and R. Durbin, 2009, B. Langmead et al., 2009, B. Langmead, and S.L. Salzberg, 2012]. The sequence alignment has played a pivotal role in identifying the causal genes of certain genomic disorders or genetic mutations relevant to biological questions [K.K. Farh et al., 2015, W. Pan et al., 2015, Roadmap Epigenomics Consortium, 2015, R. Do et al., 2015, D.F. Gudbjartsson et al., 2015]. Conceptually, a graph can be used to represent a pa genome though it requires a sophisticated scheme to overcome cyclic edges, where repeat sequences exist, and to preserve relative positional information.

The FM-index can overcome these problems over a naive graph representation due to its property, allowing for calculating a precise position given recently visited symbols. Multiple genome references can co-exist in solely a single index so that even more difficult mutation events such as gene fusions, fissions, and lateral gene transfers can also be identified. Given sequence alignments of long reads, the identification of genetic variations can be realized by the *sequence length-based* prediction, which

may confer a less fluctuation in results than the conventional *read coverage-based* inference of mutations (refer to Chapter 4). Depending on the sensitivity of short read alignment algorithms, and SV callers, the variants identified have been highly variable in terms of genomic positions, the types of mutation, and the number of bases affected [D.F. Conrad, and M.E. Hurles, 2007, L. Tattini et al., 2015, P. Zhao et al., 2016]. To cope with such problems, ensemble methods have been suggested but it does not address the underlying causes of such variability and cannot offer a definitive solution for calling variants [K. Wong et al., 2010, H.Y. Lam et al., 2012, T. Mimori et al., 2013].

Second, the NGS datasets are not error-free. To represent a pan genome, false signals should be removed before building an index. While a genomic study is going on, redundant or noisy signals should be normalized and corrected. To achieve a high degree of statistical significance, a sampling process should undergo biological or technical replications, leading to less variable and reliable observations. Quantitative methods in a population-scale perspective may be able to improve the accuracy of statistical inference, but they do not always guarantee a non-biased interpretation. Technical errors should be determined at the initial stage of an analysis to hinder them from propagating toward the downstream analysis.

0.3.2 Metagenomics

Metagenomics is the study of genomic profiles of organisms in environmental samples, mainly targeting the microbial community, and often focusing on bacteria. It is divided into two basic goals: the classification of microorganisms, and understanding the functions of a microbial community. Though the term metagenomics, was coined in 1998 by J. Handelsman [J. Handelsman et al., 1998], questions about the taxonomy of microorganisms have been raised earlier. Organisms have been divided into prokaryotes and eukaryotes by Stanier and Van Niel [R.Y. Stanier, and C.B. Van Niel, 1962]. The third kingdom, archaebacteria, later shortened to Archaea, was introduced by Carl Woese based on 16S ribosomal RNA (rRNA) signatures [C.R. Woese, and G.E. Fox, 1977]. 16S rRNA sequences were used to determine the identity of a prokaryote since they change only slowly and distinguish species. C. Woese wanted to build a comprehensive 16S rRNA collection, which has been widely used to characterize novel organisms [D.J. Lane et al., 1985].

Though Pace introduced the idea of characterizing mixed microbial populations [N.R. Pace, et al., 1985], which is regarded as the root of metagenomics, the aims of his studies were in parallel with Woese's in terms of a more comprehensive taxonomy of micro-organisms. Culturability, which is the ratio of culturable bacteria to the total cell count, was reported typically low in several studies, and, in turn, raised doubts if the cultured organisms represent the true bacterial universe since late 1950s [R.I. Amann et al., 1995, S.J. Giovannoni et al., 1990]. Staley, and Konopka described the unculturable nature of microbial communities as the "Great Plate Count Anomaly" [J.T. Staley, and A. Konopka, 1985]. Soon after, polymerase chain reaction (PCR) began to be used to identify micro-organisms [S.J. Giovannoni et al., 1990, D.H. Persing et al., 1990, T.M. Schmidt et al., 1991]. Larger scale studies became feasible when the latest sequencing technologies were applied [J.C. Venter et al., 2004, H.N. Poinar et al., 2006, R.A. Edwards et al., 2006, S. Yooseph et al., 2010]. The available resources and data have been dramatically increased as the NGS technology started being incorporated. Due to the massive amount of noisy data, and the high number of organisms, however, the characterization and analyses in metagenomics remain very challenging.

For metagenomics studies, assigning a sequencing read to a particular taxonomic rank is required as a prior condition to estimate the abundance of a certain group of organisms in the sample. The 16S rRNA methods are especially suitable for obtaining accurate identification of microorganisms. It is known that 16S rRNA genes are divided into nine hypervariable regions, where V2, V3, and V6 re-

gions are the most heterogeneous, providing the highest discriminating power [S. Chakravorty et al., 2007]. However, for some species, i.e., *Burkholderia pseudomallei*, and *Burkholderia thailandensis*, and closely related *Staphylococcus* species, the 16S rRNA gene sequences cannot be used to distinguish their identities [P.C. Woo et al., 2008]. Sequencing errors and chimeric reads due to PCR amplification degrade the accuracy of the 16S rRNA profiling such that the species diversity would be overestimated due to these artifacts [C. Quince et al., 2009].

Whole genome shotgun (WGS) methods could avoid PCR amplification biases of 16S rRNA genes. However, it may not be able to capture the whole diversity because of the limited capacity of a sequencer. For instance, some rare species would not be identified if the sequencing depth is not sufficient [N. Shah et al., 2011]. However, NGS methods can capture functional diversity other than 16S rRNA genes. A human body provides a shelter for trillions of microorganisms. The human microbiome carries 8 million unique protein coding genes while the human genome has only about 22,000 protein coding genes, which may indicate that bacteria have a much higher impact on human homeostasis and development of diseases than human genes. The NIH Human Microbiome Project (HMP) defined normal bacteria composition by sampling 242 healthy US volunteers, processing 3.5 Tb of genomic dataset [The NIH HMP Working Group, 2009].

0.4 Structure

This study provides important contributions to bioinformatics. First, the concept of the population index to represent a pan genome and to provide a higher level abstraction for various genomic applications is delineated. Second, I propose a sequencing error correction algorithm, emphasizing the importance of an initial denoising step for an analysis to stop propagating the errors to the downstream rounds. Third, I introduce a novel *multiple WGA* algorithm, which can map thousands of long reads of genomes in a population for a single species in an amortized constant time[1]. This algorithm can potentially replace conventional short read aligners, which are designed for a single reference genome and for short reads. Fourth, variants from point mutations to large structural differences within a population can be identified without significant efforts for merging short read alignment results. Fifth, I provide a highly sensitive taxonomy classifier, which surpasses conventional fixed *k*-mer based methods in sensitivity at the species resolution.

The dissertation assembles five distinct chapters starting with the primary concept of the study, the population index. Chapter 1 describes established data structures, the FM-index, and finally the population index, providing basic knowledge for the following chapters. Chapter 2 restates a formerly published sequencing error correction module, Trowel. In section 2.3, I introduce a novel sequencing error correction algorithm based on the FM-index. Chapter 3 proposes an advanced *multiple WGA* algorithm, Kairos, that can align multiple individuals simultaneously at the whole genome level in an amortized constant time given genome size and the number of genomes in the index. The genome assemblies of 1,037 *A. thaliana* strains have been aligned to detect point mutations and small indels. Chapter 4 introduces a structural variant calling algorithm, Apollo, which predicts SVs on top of the Kairos alignments. Chapter 5 explains a highly sensitive and efficient taxonomy classifier, Poseidon. Metagenomics analyses gaining substantial attentions recently often begin with the identification and quantification of species in the sample. The evaluation shows a comprehensive advantage of Poseidon over conventional methods. Though every chapter draws independent conclusions, a dedicated chapter for the final remarks is presented.

[1] Amortized constant time: average time taken per operation is constant given genome size and the number of genomes.

0.5 References

M.D. Adams et al. (2000) The genome sequence of Drosophila melanogaster, Science, 287(5461):2185-95.
R.I. Amann, W. Ludwig, and K.H. Schleifer (1995) Phylogenetic identification and in situ detection of individual microbial cells without cultivation, Microbiol Rev., 59(1):143-69.
Arabidopsis Genome Initiative (2000) Analysis of the genome sequence of the flowering plant Arabidopsis thaliana, Nature, 408(6814):796-815.
J. Bell, and J.B.S. Haldane (1937) The Linkage between the Genes for Colour-Blindness and Haemophilia in Man, Proceedings of the Royal Society of London. Series B, Biological Sciences, 123(831): 119-150.
F.R. Blattner et al. (1997) The complete genome sequence of Escherichia coli K-12, Science, 277(5331):1453-62.
B.H. Bloom (1970) Space/time trade-offs in hash coding with allowable errors, Communications of the ACM, 13(7):422-426.
W.J. Bock (1981) Reviewed Work: The Evolutionary Synthesis. Perspectives on the Unification of Biology, The Auk (McLean, VA: American Ornithologists' Union) 98(3):44-646.
T.C. Boothby et al. (2015) Evidence for extensive horizontal gene transfer from the draft genome of a tardigrade, Proc Natl Acad Sci USA, 112(52):15976–81.
P.J. Bowler, and I.R. Morus (2005) Making modern science: a historical survey, Chicago: University of Chicago.
S. Brenner et al. (2000) Gene expression analysis by massively parallel signature sequencing (MPSS) on microbead arrays. Nat Biotechnol. 18(6):630-4.
C. elegans Sequencing Consortium (1998) Genome sequence of the nematode C. elegans: a platform for investigating biology, Science, 282(5396):2012-8.
M.J. Chaisson, and P.A. Pevzner (2008) Short read fragment assembly of bacterial genomes, Genome Res., 18(2):324-30.
M.J. Chaisson et al. (2015) Resolving the complexity of the human genome using single-molecule sequencing, Nature, 517(7536):608-11.
S. Chakravorty, D. Helb, M. Burday, N. Connell, and D. Alland (2007) A detailed analysis of 16S ribosomal RNA gene segments for the diagnosis of pathogenic bacteria, J Microbiol Methods, 69(2):330-9.
R. Chikhi, and G. Rizk (2013) Space-efficient and exact de Bruijn graph representation based on a Bloom filter, LNCS 7534, 19:236-48.
Chimpanzee Sequencing and Analysis Consortium (2005) Initial sequence of the chimpanzee genome and comparison with the human genome, Nature, 437(7055):69–87.
A.C. Chinault, and J. Carbon (1979) Overlap hybridization screening: Isolation and characterization of overlapping DNA fragments surrounding the leu2 gene on yeast chromosome III, Gene 5(2):111-126.
Y. Cho et al. (2013) The tiger genome and comparative analysis with lion and snow leopard genomes, Nature Communications, 4:2433.
D.F. Conrad, and M.E. Hurles (2007) The population genetics of structural variation, Nat Genet., 39(7 Suppl):S30-6.
T.C. Conway, and A. Bromage (2011) Succinct data structures for assembling large genomes, Bioinformatics, 27(4):479-486.
G. Cuvier (1815) Second edition of “Essay on the Theory of the Earth”, Blackwood.
K. Danna, and D. Nathans (1971) Specific Cleavage of Simian Virus 40 DNA by Restriction Endonuclease of Hemophilus Influenzae, Proc Natl Acad Sci USA., 68(12):2913–2917.
C.R. Darwin (1859) On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life, London: Murray.

C.R. Darwin, and A.R. Wallace (1858) On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection, Journal of the Proceedings of the Linnean Society of London. Zoology 3:45-50.
N.G. de Bruijn (1946) A Combinatorial Problem. Koninklijke Nederlandse Akademie v. Wetenschappen 49:758–64.
R. Do et al. (2015) Exome sequencing identifies rare LDLR and APOA5 alleles conferring risk for myocardial infarction. Nature, 518(7537):102-6.
T. Dobzhansky (1937) Genetics and the Origin of Species. Columbia University Press, New York.
R.A. Edwards et al. (2006) Using pyrosequencing to shed light on deep mine microbial ecology, BMC Genomics, 7:57.
J. Eid et al. (2009) Real-time DNA sequencing from single polymerase molecules, Science, 323(5910):133-8.
K.K. Farh et al. (2015) Genetic and epigenetic fine mapping of causal autoimmune disease variants, Nature, 518(7539):337-43.
P. Ferragina, and G. Manzini (2000) Opportunistic Data Structures with Applications, FOCS 2000, 390.
R.D. Fleischmann et al. (1995) Whole-genome random sequencing and assembly of Haemophilus influenzae Rd, Science, 269(5223):496-512.
GenomeWeb (2010) PacBio Reveals Beta System Specs for RS; Says Commercial Release is on Track for First Half of 2011
S.J. Giovannoni, T.B. Britschgi, C.L. Moyer, and K.G. Field (1990) Genetic diversity in Sargasso Sea bacterioplankton, Nature, 345(6270):60-3.
A. Goffeau et al. (1996) Life with 6000 genes, Science, 274(5287):546, 563-7.
D.F. Gudbjartsson et al. (2015) Large-scale whole-genome sequencing of the Icelandic population, Nat Genet., 47(5):435-44.
J.B.S. Haldane (1935) The rate of spontaneous mutation of a human gene. J Genet. 31:317-26.
J. Handelsman, M.R. Rondon, S.F. Brady, J. Clardy, and R.M. Goodman (1998) Molecular biological access to the chemistry of unknown soil microbes: A new frontier for natural products, Chemistry & Biology 5 (10): R245-R249.
R.M. Idury and M.S. Waterman (1995) A new algorithm for DNA sequence assembly, J. Comp. Bio., 2(2):291-306.
International Chicken Genome Sequencing Consortium (2004) Sequence and comparative analysis of the chicken genome provide unique perspectives on vertebrate evolution, Nature, 432(7018):695–716.
E.D. Jarvis et al. (2014) Whole-genome analyses resolve early branches in the tree of life of modern birds, Science, 346(6215):1320-31.
J.D. Kececioglu, and E.W. Myers (1993) Combinatiorial Algorithms for DNA sequence assembly, Algorithmica, 13:7-51.
U. Kutschera, and K.J. Niklas (2004) The modern theory of biological evolution: an expanded synthesis. Naturwissenschaften. 91(6):255-76.
H.Y. Lam et al. (2012) Detecting and annotating genetic variations using the HugeSeq pipeline, Nat Biotechnol., 30(3):226-9.
J. Lamarck (1809) Philosophie zoologique, Dentu et L'Auteur.
E.S. Lander et al. (2001) Initial sequencing and analysis of the human genome, Nature, 409(6822):860-921.
D.J. Lane, B. Pace, G.J. Olsen, D.A. Stahl, M.L. Sogin, and N.R. Pace (1985) Rapid determination of 16S ribosomal RNA sequences for phylogenetic analyses, Proc Natl Acad Sci USA, 82(20):6955-9.
P.R. Langer-Safer, M. Levine, and D.C. Ward (1982) Immunological method for mapping genes on Drosophila polytene chromosomes, Proc Natl Acad Sci USA, 79(14):4381–5.
B. Langmead, and S.L. Salzberg (2012) Fast gapped-read alignment with Bowtie 2. Nature Methods, 9:357–359.
B. Langmead, C. Trapnell, M. Pop, and S.L. Salzberg (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, Genome Biol., 10(3):R25.

M.J. Levene, J. Korlach, S.W. Turner, M. Foquet, H.G. Craighead, and W.W. Webb (2003) Zero-mode waveguides for single-molecule analysis at high concentrations, Science, 299(5607):682-6.
H. Li, and R. Durbin (2009) Fast and accurate short read alignment with Burrows-Wheeler transform, Bioinformatics, 25(14):1754-60.
K. Lindblad-Toh et al. (2005) Genome sequence, comparative analysis and haplotype structure of the domestic dog, Nature, 438(7069):803–19.
D.P. Locke et al. (2011) Comparative and demographic analysis of orangutan genomes, Nature, 469(7331):529–533.
E.R. Mardis (2008) Next-generation DNA sequencing methods, Annu Rev Genomics Hum Genet., 9: 387–402.
M. Margulies et al. (2005) Genome Sequencing in Open Microfabricated High Density Picoliter Reactors, Nature. 437(7057):376-380.
A.M. Martien et al. (2012) Analyses of pig genomes provide insight into porcine demography and evolution, Nature, 491(7424):393–398.
J.L. McClay et al. (2015) High density methylation QTL analysis in human blood via next-generation sequencing of the methylated genomic DNA fraction, Genome Biol., 16(1):291.
J.G. Mendel (1866) Versuche über Pflanzenhybriden Verhandlungen des naturforschenden Vereines in Brünn, Bd. IV für das Jahr, 1865 Abhandlungen:3-47.
C.A. Meyer, and X.S. Liu (2014) Identifying and mitigating bias in next-generation sequencing methods for chromatin biology, Nat Rev Genet., 15(11):709-21.
T. Mimori, N. Nariai, K. Kojima, M. Takahashi, A. Ono, Y. Sato, Y. Yamaguchi-Kabata, and M. Nagasaki (2013) iSVP: an integrated structural variant calling pipeline from high-throughput sequencing data, BMC Syst Biol., 7 Suppl 6:S8.
Mouse Genome Sequencing Consortium (2002) Initial sequencing and comparative analysis of the mouse genome, Nature, 420(6915):520-62.
E.W. Myers (2005) The fragment assembly string graph, Bioinformatics, 21 Suppl 2:ii79-85.
M. Olson, L. Hood, C. Cantor, and D. Botstein (1989) A common language for physical mapping of the human genome, Science, 245(4925):1434-5.
N.R. Pace, D.A. Stahl, D.J. Lane, and G.J. Olsen (1985) Analyzing natural microbial populations by rRNA sequences, ASM News 51:4-12.
PacificBioscience (2014) http://investor.pacificbiosciences.com/releasedetail.cfm?ReleaseID=876252
W. Pan, W. Gu, S. Nagpal, M.H. Gephart, and S.R. Quake (2015) Brain tumor mutations detected in cerebral spinal fluid, Clin Chem., 61(3):514-22.
J. Pell, A. Hintze, R. Canino-Koning, A. Howe, J.M. Tiedje, and C.T. Brown (2012) Scaling metagenome sequence assembly with probabilistic de Bruijn graphs, Proc Natl Acad Sci USA, 109(33):13272-7.
H. Peltola, H. Söderlund, and E. Ukkonen (1984) SEQAID: a DNA sequence assembling program based on a mathematical model, Nucleic Acids Res. 12(1 Pt 1):307-21.
D.H. Persing, S.R. Telford, A. Spielman, and S.W. Barthold (1990) Detection of Borrelia burgdorferi infection in Ixodes dammini ticks with the polymerase chain reaction, J Clin Microbiol, 28(3):566-72.
P.A. Pevzner (1989) 1-Tuple DNA sequencing: computer analysis, J Biomol Struct Dyn., 7(1):63-73.
P.A. Pevzner, H. Tang, and M. S. Waterman (2001) An Eulerian path approach to DNA fragment assembly Proc Natl Acad Sci USA, 98(17):9748-53.
H.N. Poinar et al. (2006) Metagenomics to Paleogenomics: Large-Scale Sequencing of Mammoth DNA, Science, 311(5759): 392–394.
J.U. Pontius et al. (2007) Initial sequence and comparative analysis of the cat genome, Genome Research, 17(11):1675–89.
J.M. Prober, G.L. Trainor, R.J. Dam, F.W. Hobbs, C.W. Robertson, R.J. Zagursky, A.J. Cocuzza, M.A. Jensen, and K. Baumeister (1987) A system for rapid DNA sequencing with fluorescent chain-terminating dideoxynucleotides, Science 238(4825):336-41.

C. Quince, A. Lanzén, T.P. Curtis, R.J. Davenport, N. Hall, I.M. Head, L.F. Read, and W.T. Sloan (2009) Accurate determination of microbial diversity from 454 pyrosequencing data, Nat Methods., 6(9):639-41.
Roadmap Epigenomics Consortium (2015) Integrative analysis of 111 reference human epigenomes, Nature, 518(7539):317-30.
F.A. San Lucas et al. (2015) Minimally invasive genomic and transcriptomic profiling of visceral cancers by next-generation sequencing of circulating exosomes, Ann Oncol., pii:mdv604.
F. Sanger, S. Nicklen, and A.R. Coulson (1977) DNA sequencing with chain-terminating inhibitors Proc Natl Acad Sci USA, 74(12):5463-5467.
A. Scally et al. (2012) Insights into hominid evolution from the gorilla genome sequence, Nature, 483(7388):169–175.
T.M. Schmidt, E.F. Delong, and N.R. Pace (1991) Analysis of a marine picoplankton community by 16S rRNA gene cloning and sequencing, Journal of Bacteriology, 173(14):4371-4378.
N. Shah, H. Tang, T.G. Doak, and Y. Ye (2011) Comparing bacterial communities inferred from 16S rRNA gene sequencing and shotgun metagenomics, Pac Symp Biocomput., 165-76.
J.T. Simpson, and R. Durbin (2012) Efficient de novo assembly of large genomes using compressed data structures, Genome Res., 22(3):549-56.
L.M. Smith, S. Fung, M.W. Hunkapiller, T.J. Hunkapiller, and L.E. Hood (1985) The synthesis of oligonucleotides containing an aliphatic amino group at the 5' terminus: synthesis of fluorescent DNA primers for use in DNA sequence analysis, Nucleic Acids Res., 13(7):2399–412.
R. Staden (1979) A strategy of DNA sequencing employing computer programs Nucleic Acids Research 6(7):2601–10.
J.T. Staley, and A. Konopka (1985) Measurement of in situ activities of nonphotosynthetic microorganisms in aquatic and terrestrial habitats. Annu. Rev. Microbiol. 39:321-346.
R.Y. Stanier, and C.B. Van Niel (1962) The concept of a bacterium, Archiv fur Mikrobiologie 42:17-35.
L. Tattini, R. D'Aurizio, and A. Magi (2015) Detection of Genomic Structural Variants from Next-Generation Sequencing Data, Front Bioeng Biotechnol., 3:92.
H. Tettelin et al. (2005) Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: implications for the microbial "pan-genome", Proc Natl Acad Sci USA, 102(39):13950-5.
The Bovine Genome Sequencing and Analysis Consortium (2009) The genome sequence of taurine cattle: a window to ruminant biology and evolution, Science, 324(5926): 522–528.
The NIH HMP Working Group (2009) The NIH Human Microbiome Project, Genome Res., 19: 2317-2323.
J.C. Venter et al. (2001) The sequence of the human genome, Science, 291(5507):1304-51.
J.C. Venter et al. (2004) Environmental Genome Shotgun Sequencing of the Sargasso Sea, Science 304(5667): 66–74.
C.M. Wade et al. (2009) Genome sequence, comparative analysis, and population genetics of the domestic horse, Science, 326(5954):865–867.
J.D. Watson, and R.M. Cook-Deegan (1991) Origins of the Human Genome Project. FASEB Journal, 5(1):8–11.
J.D. Watson, and F.H.C. Crick (1953) A Structure for Deoxyribose Nucleic Acid, Nature, 171:737-738
C.R. Woese, and G.E. Fox (1977) Phylogenetic structure of the prokaryotic domain: The primary kingdoms, Proceedings of the National Academy of Sciences 74(11):5088-5090.
K. Wong, T.M. Keane, J. Stalker, and D.J. Adams DJ (2010) Enhanced structural variant and breakpoint detection using SVMerge by integration of multiple detection methods and local assembly, Genome Biol., 11(12):R128.
P.C. Woo, S.K. Lau, J.L. Teng, H. Tse, and K.Y. Yuen (2008) Then and now: use of 16S rDNA gene sequencing for bacterial identification and discovery of novel bacteria in clinical microbiology laboratories, Clin Microbiol Infect., 14(10):908-34.

S. Wright (1932) The roles of mutation, inbreeding, crossbreeding, and selection in evolution, Proceedings of the Sixth International Congress on Genetics, 355-366.
S. Yooseph et al. (2010) Genomic and functional adaptation in surface ocean planktonic prokaryotes, Nature, 468(7320):60–66.
J. Yu et al. (2002) A draft sequence of the rice genome (Oryza sativa L. ssp. indica), Science, 296(5565):79–92.
D.R. Zerbino, and E. Birney (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res., 18(5):821-9.
P. Zhao, J. Li, H. Kang, H. Wang, Z. Fan, Z. Yin, J. Wang, Q. Zhang, Z. Wang, and J.F. Liu (2016) Structural Variant Detection by Large-scale Sequencing Reveals New Evolutionary Evidence on Breed Divergence between Chinese and European Pigs, Sci Rep., 6:18501.

Chapter 1

1. Population Index

1.1 Introduction

One fundamental way to understand biology is from genome sequences in which the adaptation history of an organism to the environment is encoded. Each genome contains sequences of four nucleotides, adenine, cytosine, guanine and thymine. These four letters are sufficient to encode all the protocol of an organism for survival in an environment.

Genomes have different levels of complexity. Prokaryotes may have only a single chromosome, while eukaryotes typically have multiple chromosomes. Thus, we may assume that eukaryotes are more complex than prokaryotes. A genome of long length does, however, not imply a more complex physical structure. Known as C-value paradox, a genome of a protozoan organism, *Polychaos dubia*, is about 200 times larger than the one of *Homo sapiens* [C.T. Friz, 1968, C.A. Jr. Thomas, 1971]. The fact is that though a genome may contain all information for an organism to survive in a certain condition, vast amount of regions in a genome can be non-coding. This discovery of non-coding gene resolved the C-value paradox [G. Elgar, and T. Vavouri, 2008]. The coding genes and non-coding elements are widely spread in a genome, and their mutations are observed in a population. The changes in frequency and location of such components in a certain environment over time have been attained substantial interests of biologists. To find a set of differences among hundreds or thousands of similar signals, we need to cut down the complexity and avoid distractions.

The first step for such dimensionality reduction is the sequence alignment. A conventional way is to perform sequence mapping against a single linear reference genome [H. Li et al., 2009]. The linearity means that a reference genome only represents a single organism and does not reflect all the combinations of mutations in a population. The gap between the true population-wide genome representation and the single linear reference genome can cause misinterpretation. Those discrepancy can be alleviated locally by allowing for some mismatches while matching patterns. However, there might be some missing sequences which are only present in the population but not in the single reference genome.

A species in a population can be distinctly identified by appraising dissimilarities in genomic materials based on the extensive number of pair-wise sequence comparisons. For instance, when one tries to find genetic mutations among hundreds of individuals, the reference genome is used as a set of known sequence locations. We may perform sequence alignments millions of times in order to detect mutations. In spite of such problems, a single reference based method can achieve a decent level of sensitivity for detecting small mutations. It is because that majority of sequences within a single species is highly conserved though the positional information can be varying due to the recombination.

More efficient and precise analysis necessitates a non-linear representation of a population to address all mutations in a single place. This improved representation minimizes the number of sequence alignments for population-wide comparisons. In section 1.3, I explain the suffix array to introduce the concept. The FM-index, a specialized version of the suffix array for genomic datasets, is explained in section 1.4. Experiments on the 1001 Genome datasets are shown to present its computational space efficiency. In section 1.5, the population index is described, which is an enhancement over the FM-index.

1.2 Definition of "text"

A text T = T[0…n) = $T_0T_1…T_{n-1}$ is consecutive letters of length n over an ordered symbol set E = {E_0, $E_1…E_{\sigma-1}$}. A text of length n is of n suffixes S_i = T[i…n) and n prefixes P_i = T[0…i). The lexicographical order is defined for two texts. Given text A = A[0…l) and text B = B[0…m), the lexicographic order of A is smaller than B if and only if at the first index i where A_i is alphabetically smaller than B_i. If the length of two texts are not identical, then the sentinel, *S*, which is the alphabetically smallest character in E, is assumed to occupy those positions. A text representing a series of nucleotides can be denoted by E={$, A, C, G, T, N} with σ=6. The symbol 'N' indicates an ambiguous letter. A reverse complementary text $\overline{T}$ represents the characters in reverse direction such that $\overline{T}$[0…i) = T(i…0] with a complementary ordered symbol set E'={$, T, G, C, A, N}.

1.3 The Suffix Array

Since the sequence similarities based on alignments are the essential information for genomics, data structures supporting for pattern matching and retrieval of associating data have been applied in biology. Note that space means the amount of memory needed as working space during the calculation.

A Trie is an ordered tree data structure, coined by E. Fredkin [E. Fredkin, 1960], storing key and value associations. The time complexity of the Trie is $\mathcal{O}$(M) while the space complexity is roughly $\mathcal{O}$(MN), where M is the length of the longest pattern and N is the number of patterns. All the descendants of a node share a common prefix of the pattern and only leaf nodes have associated values. In practice, due to its high memory consumption, Trie has been replaced by more space-efficient data structures.

A suffix tree is a compressed Trie, introduced by P. Weiner [P. Weiner, 1973], mapping all the suffixes of a string and values. A suffix tree of a text T is an ordered tree having n leaves. All internal nodes are connected to at least two sub-nodes by edges labeled with non-empty substring of T. Two edges spanned from a node must not share the same initial character. A path of edges directed from the root node to a leaf represents a suffix of T. Ukkonen's suffix tree construction algorithm is of a linear time complexity for the constant number of symbols [E. Ukkonen, 1995]. Though a suffix tree provides decent performance gains in several pattern matching problems, the space consumption for the structure is far higher than the one for the original string itself.

More precisely, an explicit version of a suffix tree requires $\mathcal{O}(n^2 \log \sigma)$ bits to represent all nodes and edges. We can reduce the space complexity to $\mathcal{O}(n \log n)$ by storing the text T, pointers to suffixes p⇒v in T, and the length of the suffix at each node, where p is the parent node of v. In practice, for the 32-bit machines, the size of a suffix tree is about 20 times larger than the original string, and for the 64-bit architecture, which is common nowadays, the space requirement is almost doubled except the space for T. A generalized suffix tree is the suffix tree built on a set of strings with unique sentinel characters.

A suffix array is the sorted array of all suffix-start positions of a string in lexicographical order invented by U. Manber and G. Myers [U. Manber, and G. Myers, 1990]. More formally, a suffix array of a text T is an array SA[0…n) storing the pointers to each suffix following the lexicographical order. The data structure occupies $n \log n$ bits. SA[0] contains the pointer to the sentinel. If we generalize

the suffix array, SA[0..k) are pointers to the sentinels following the original order of all texts, where k is the number of texts. Since a suffix tree can be constructed in linear time, the equal time complexity can be expected. A naive approach to build a suffix array requires $\mathcal{O}(n \log n)$ suffix comparisons, and each suffix comparison takes $\mathcal{O}(n)$. In turn, the time complexity becomes $\mathcal{O}(n^2 \log n)$.

Three research groups independently proposed linear time suffix array construction algorithms [P. Ko, and S. Aluru, 2003, J. Kärkkäinen, and P. Sanders, 2003, D.K. Kim et al., 2003]. However, their performance were worse than highly optimized $\mathcal{O}(n^2 \log n)$ algorithms in practice. The Suffix Array-Induced Sorting (SA-IS) algorithm of G. Nong and his colleagues is one of the fastest suffix array construction algorithms [G. Nong et al., 2009], which is competitive against the optimized $\mathcal{O}(n^2 \log n)$ algorithms. Meanwhile, the space of a suffix array is roughly 4 times larger than the original string for 32-bit machine.

The suffix array can be applied to quickly find a pattern P of length m within a text T, which is equivalent to locating all suffixes that begins with P. For any substring of P, the lexicographical ordering can be retrieved by two binary searches. The first search represents the minimum index sp in SA, and the second one discovers the maximum index ep. We call this SA-interval [i, j] for a partial string ρ of the P ρ-interval, where ρ is a substring of T and also ρ is a prefix of T[SA[k]…n) for all indexes i ≤ k ≤ j. The next interval [sp', ep'] can be determined with the same procedure and the equality condition sp ≤ sp' ≤ ep' ≤ ep must be hold. Otherwise, the pattern P does not exist in the text T. The pattern matching can be solved in $\mathcal{O}(m \log n)$ time, where m is the length of the pattern P. A noticeable property of this search is that the frequency of a pattern ρ in T can be calculated by the difference of indexes in the ρ-interval, (ep-sp).

A longest common prefix (LCP) array stores the lengths of the LCPs between each pair of successive suffixes. With the LCP array, the time complexity of the pattern matching problem can be improved to $\mathcal{O}(m + \log n)$ [U. Manber, and G. Myer, 1990]. Moreover, another algorithm achieved $\mathcal{O}(m)$ time complexity, which equals to the theoretical performance of suffix trees [M.I. Abouelhoda et al., 2004]. Although the space efficiency of a suffix array is better than the one of suffix trees, the space requirements are still excessive for a large text. The length of a string to construct a suffix array is bounded by the word length of a computer architecture. For instance, in the 32-bit architecture a computer can handle the text length up to 2^{32}. Hence, a population-level index containing strings longer than such limit requires an architecture dealing with higher bits. In an M-bit architecture, a suffix array for a string of length N requires a constant MN bits.

1.4 The FM-index

1.4.1 Compressibility

The Burrows-Wheeler transform (BWT) is a reversible permutation of the input string invented by Michael Burrows and David Wheeler in 1994. All rotations of the string are sorted lexicographically and the concatenation of the last character of each rotation yields the transformed text. Due to this fact, an equation, $T^{BWT}[i] = T[SA[i] - 1]$, where S is the input string, and SA is the suffix array, is inducible. The equation means that the *i*-th symbol of the BWT is the letter just before the *i*-th suffix. Thus, an uncompressed BWT text occupies $n \log \sigma$ bits, which are for the bitwise encoded T.

1. Population Index

An FM-index is introduced as an opportunistic data structure by P. Ferragina, and G. Manzini since it allows for the compression of input strings and fast string operations [P. Ferragina, and G. Manzini, 2000]. The compressibility comes from the use of the BWT. To understand the compressibility, the zeroth order empirical entropy, H_0, of the text T of length n should be defined. Let n_i be the frequency of the symbol E_i in the text T.

$$H_0(T) = -\sum_{i=1}^{\sigma-1} \frac{n_i}{n} \log \frac{n_i}{n} \quad (1.1)$$

, where 0log0=0. H_0 represents an expected space for encoding a symbol E_i with $\log \frac{n}{n_i}$ bits. Thus, the lower bound space requirement of the target text, where each symbol is independently compressed, can be calculated by multiplying H_0 with n. The zeroth order entropy can be extended to address the entropy of preceding symbols, context, with respect to the compression. For the word w of length k, let w_T denote a word following w. The k-th order empirical entropy defined as follows:

$$H_k(T) = \frac{1}{n} \sum_{w \in E^k} |w_T| \mathrm{H}_0(w_T) \quad (1.2)$$

The lower bound of the k-th order compression, $nH_k(T)$, can be calculated based on the recently observed k symbols. The common words in T can have the same subsequent symbols, thus the entropy for such words decreases in terms of the compression. Though $H_{k+1}(T) \leq H_k(T)$ is true, the increase of k adversely affects the space complexity due to the demand for larger-size codes. The BWT is the heart of many compression methods thanks to its property such that adjacent elements in T^{BWT} are lexicographically sorted by the subsequent strings in the original suffixes, thus many runs of the same characters can be found. Let the number of runs be γ. Then the following upper bound of the entropy is made [V. Mäkinen, and G. Navarro, 2004]:

$$\gamma \leq nH_k + \sigma^k \quad (1.3)$$

As indicated, the presence of similar sequences or suffixes are the key factor increasing the compression efficiency. Thus, the compression ratio heavily relies on the number of genomes and the number of homologous sequences among genomes in the index. The more number of genomes an index gets, the higher the compression ratio it can achieve. Hence, the index of simple organisms or closely related genomes can obtain a decent compression ratio. Practical compression ratios by run length encoding are calculated on complete or draft genomes downloaded from NCBI. The genomes of *Arabidopsis thaliana* are not complete but short-read assemblies of 1001 genome datasets generated by the SPAdes 3.5.0. [A. Bankevich et al., 2012]. All experiments are performed on a 64-core, 1-Tb, Linux-compatible machine (Table 1, and Fig. 1).

Table 1. Experimental compression ratios. Each compression ratio is taken when the construction of the FM-index is finished.

Source	Number of genomes	Compression ratio
Protists	32	0.656057755
Fungi	53	0.673270018
Arabidopsis thaliana	346	0.170364402
Bacteria	6545	0.216114006

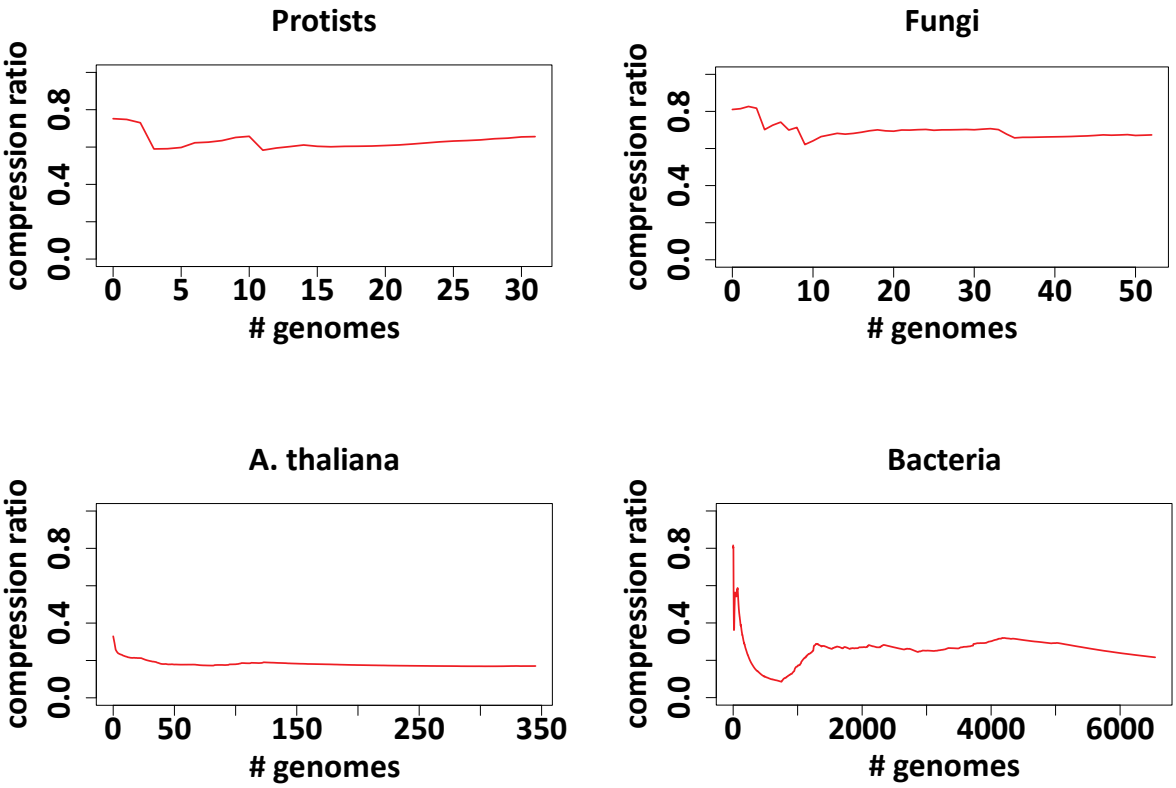

Figure 1. Changes in compression ratios:

(A) Protists; (B) Fungi, (C) *A. thaliana*; (D) Bacteria.

Each compression ratio data point is taken after every incremental build finishes.

The first group of two eukaryotic organisms (A, B) shows poorer compression results than the one of *A. thaliana* genomes. The individual genomes in 1001 genomes datasets are expected to be very close each other, the same species in fact, thus a high compression ratio is achieved. While incrementally building the FM-index, the compression ratio almost constant for these datasets. Figure 1 (D) for bacteria demonstrates peaks, indicating a variability among genomes in the index. The number of protists and fungi genomes are insufficient to observe a drop in the ratio, but based on Figure 1 (C), a high compression can be expected for a single species.

1.4.2 Pattern Matching

In the pattern matching problem, the backward search algorithm guarantees the time complexity of pattern matching bound to the length of the pattern. The practical running time of the algorithm is

heavily dependent on the efficiency of suffix locating functions: Last-to-First (LF) function, and Ψ-function. The LF mapping needs an additional inverse suffix array (ISA) data structure, which answers the suffix index in SA given index i. The LF-mapping is defined as follows:

$$LF[i] = ISA[\mathrm{SA[i]} - 1] \quad (1.4)$$

Given suffix j = SA[i], the LF-function yields the index of suffix j-1, which represents a suffix just before the suffix j in the cyclic permutation of the original text T. The direction of the Ψ-function is exactly opposite to the one of the LF-function.

$$\Psi[i] = ISA[SA[i] + 1] \quad (1.5)$$

Hence, given suffix j = SA[i], the Ψ-function confers the index of suffix j+1. The LF values in a range [i,j] increases by 1 when T^{BWT}[i,j] are a series of the same symbols. Similarly, the Ψ values in a range increases when the first symbol of consecutive suffixes are equal. By focusing on Ψ values, the space complexity for the compressed suffix array becomes $H_0 + \mathcal{O}(n \log\log n)$ bits [G. Navarro, and V. Mäkinen, 2007].

A previous LF-mapped value indicates the next position in T^{BWT}. For each iteration, the LF-mapping is calculated by the equation below.

$$LF[i] = C[T^{BWT}[i]] + \mathrm{rank}(\mathrm{T^{BWT}[i], i}) \quad (1.6)$$

, where C is a global occurrence table, and rank() is a function returning the frequency of a symbol in T^{BWT} for all the indices k in $0 \le k \le i$. C maps each symbol with the frequency of lexicographically smaller symbols in T^{BWT}. A compressed suffix array takes only the every d-th entry out of the entire suffix indexes. Hence, we can retrieve all entries if i = kd, where k is the sampling size. For suffix entries with $i \neq kd$, either LF- or Ψ-function is applied D times until x = SA[ks] exists. x becomes the suffix locating apart from either left or right of the suffix in D distance, respectively. The worst case time complexity for an access to a suffix entry is $\mathcal{O}(k)$. For a backward search, we use the LF-mapping property. Refer to [P. Ferragina, and G. Manzini, 2000] for details about the backward search algorithms.

1.5 The Population Index

An FM-index construction of a large-scale datasets needs a scheme to perform the BWT on a collection of strings. Conceptually, the BWT can be done for a concatenated text of the input strings [S. Mantaci et al., 2005] though the space requirement is too high in practice. A recent space-efficient solution, the BCR algorithm named after the initials of inventors [M.J. Bauer et al. 2011], transposes the original set of strings and updates T^{BWT} retaining intermediate states at each iteration. The algorithm also supports for an external construction method, leading to a substantial reduction in the space requirements. T^{BWT} is divided into σ-BWT blocks where the first block represents the characters just before the sentinels (Fig. 2). Ropebwt allows for an incremental FM-index construction on top of the B^+tree data structure, meaning that a set of strings can be inserted to an existing index [H. Li, 2014].

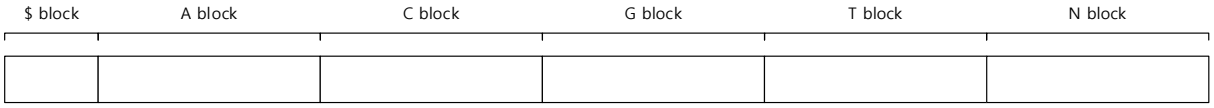

Figure 2. σ-BWT blocks. The blocks in a population index represents lexicographically sorted symbols in ordered set E.

A population index integrates a higher-level abstraction describing all the strings in datasets together with the FM-index. A suffix pointer for each symbol in the sentinel block has a special characteristic due to the LF-mapping that the order of sentinels in the BW-transformed text represents the indices of input strings. To associate suffix pointers with indices of all strings, backtrackings from all symbols in the sentinel block are performed until encountering the other sentinel and only uniformly sampled suffix pointers are mapped to reduce the space. An inverse sentinel array, I coined, associates the suffix pointers with descriptions of all strings. After backtracking a pattern, descriptions for the pattern can be retrieved along with the locations (Fig. 3).

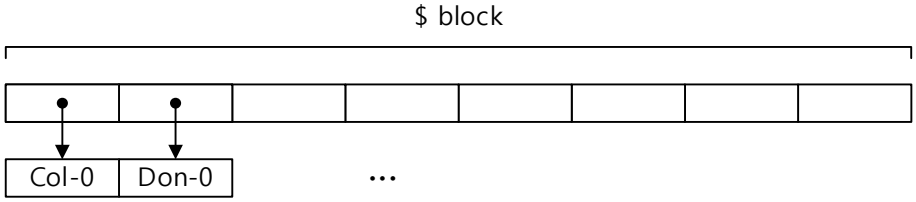

Figure 3. An example abstraction of an inverse sentinel array. Each text terminated by the sentinel symbol ($) is linked to the name of a strain of *Arabidopsis thaliana*.

1.6 Conclusion

I explained a brief history of data structures for the pattern matching problems in genomics, and recapitulates a few important aspects of the FM-index through this study. The FM-index provides a space-economic representation, which can emulates a colored graph describing a pan genome. Pattern matching operations can be solved very time-efficiently. I explicitly mentioned that the inverse sentinel array can be used to map each text to a certain description, which endows a power to annotate genomic datasets and has many applications in genomics. The realizations, or results, of the population index are rendered in each successive chapter: Trowel 2, Kairos, Apollo, and Poseidon.

1.7 References

M.I. Abouelhoda, S. Kurtz, and E. Ohlebusch (2004) Replacing suffix trees with enhanced suffix arrays, Journal of Discrete Algorithms, 2:53.
A. Bankevich, et al. (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing, J Comput Biol., 19(5):455-477.
M.J. Bauer A.J. Cox and G. Rosone (2011) Lightweight BWT construction for very large string collections, CPM, Springer, LNCS 6661, 219-231.
G. Elgar, and T. Vavouri (2008) Tuning in to the signals: Noncoding sequence conservation in vertebrate genomes, Trends in Genetics, 24(7):344–352.

P. Ferragina, and G. Manzini (2000) Opportunistic Data Structures with Applications, FOCS 2000, 390.
E. Fredkin (1960) Trie Memory, Communications of the ACM, 3(9):490-499.
C.T. Friz (1968) The biochemical composition of the free-living Amoebae Chaos chaos, Amoeba dubia and Amoeba proteus, Comp Biochem Physiol, 26:81-90.
J. Kärkkäinen, and P. Sanders (2003) Simple linear work suffix array construction, ICALP 03, Springer, LNCS 2719, 943-955.
D.K. Kim, J.S. Sim, H. Park, and K. Park (2003) Linear-time construction of suffix arrays, Proc. 14th Annual Symposium, CPM, 200-210.
P. Ko, and S. Aluru (2003) Space efficient linear time construction of suffix arrays, Springer, CPM, LNCS 2676, 203-210.
H. Li (2014) Fast construction of FM-index for long sequence reads, Bioinformatics, 30(22):3274-5.
H. Li and R. Durbin (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform, Bioinformatics, 25:1754-1760.
U. Manber, and G. Myers (1990) Suffix arrays: a new method for on-line string searches, First Annual ACM-SIAM Symposium on Discrete Algorithms. 319–327.
S. Mantaci, A. Restivo, G. Roson, and M. Sciortino (2005) An Extension of the Burrows Wheeler Transform and Applications to Sequence Comparison and Data Compression. CPM, 178-189.
V. Mäkinen, and G. Navarro (2004) Compressed Compact Suffix Arrays, CPM, LNCS 3109, 420-433.
G. Navarro, and V. Mäkinen (2007) Compressed full-text indexes, ACM Computing Surveys (CSUR), 39(1).
G. Nong, S. Zhang, and W.H. Chan (2009) Linear Suffix Array Construction by Almost Pure Induced-Sorting, Data Compression Conference. 193.
C.A. Jr. Thomas (1971) The genetic organization of chromosomes, Annu Rev Genet., 5:237-56.
E. Ukkonen (1995) On-line construction of suffix trees, Algorithmica, 14(3):249-260.
P. Weiner (1973) Linear pattern matching algorithm, Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory, 1-11.

Chapter 2

2. The Sequencing Error Correction

Note that the chapter contains revised materials published in [E.C. Lim et al., 2014] by permission of Oxford University Press. The licenses are obtained on Feb. 11, 2016 with the license numbers of 3805831316375, 3805830411432, and 3805831087522 provided by Copyright Clearance Center. Any collaborative works in the original paper are not included, and I hereby confirm that the chapter only contains researches independently conducted by myself.

2.1 Introduction

Next generation sequencing (NGS) technologies does not yet provide reads that are exactly identical to randomly sampled genomic fragments and have base calling errors. The quality of some downstream analyses can be degraded without the sequencing error correction. Recent sequencing error correction algorithms rely mainly on *k*-mer spectrum originated from the spectral alignment (SA) theory introduced by P.A. Pevzner [P.A. Pevzner et al., 2001]. Although diverse variants of this branch has been introduced, they share some common principles and data structures. The first stage of most algorithms in this branch is to build an index of trusted *k*-mers. A *k*-mer can be classified as either trusted or untrusted by a certain threshold, e.g. frequency, or quality value of a *k*-mer. Known algorithms are often dependent on the frequency criteria. For instance, a *k*-mer occurring more often than a given threshold is relatively trustworthy.

A SA approach tries to maximize the number of trusted *k*-mers in a reads by altering the erroneous bases. Quake [D.R. Kelley et al., 2010], a well-known SA approach, applies a mixed model of the distributions of trusted and untrusted *k*-mers incorporating quality values. Musket [Y. Liu et al., 2013] employs two-stage corrections based on frequencies of *k*-mers and does not use base qualities. BLESS [Y. Heo et al., 2014] applies the bloom filter to reduce the space requirement. For instance, the memory consumption is reduced by the factor of 14 to 24 for human datasets compared with Musket. Lighter [L. Song et al., 2014] avoids the counting stage by only storing randomly taken *k*-mers. The problem of Lighter is that k-mers residing in a low-depth but high quality region highly likely will be ignored by the sampling process, leading to lower accuracy.

Coral [L. Salmela, and J. Schröder, 2011] corrects indel errors by performing multiple alignments for similar reads with Needleman-Wunsch algorithm, which is a global alignment method. Coral is not practical for very large datasets due to its high time and space complexity. Hybrid SHREC [L. Salmela, 2010] also corrects indels by replacing low-weight nodes in a suffix tree. This tool introduces random sequences which differs from the true genome. Given several trials with the equal parameter, it is not possible to obtain exactly same results due to the flaws in software implementation and has not been maintained for a long time.

I introduce two error correction modules, Trowel 1 and Trowel 2, suitable for Illumina datasets. The former is a *k*-mer based algorithm while the latter is an FM-index based corrector. Trowel 1 is unique that it determines a trusted *k*-mer from sequences with continuous high quality values rather than a frequency threshold, which fluctuates stochastically because the assumption depends on the uniform distribution of sequencing coverage. Trowel 2 is the first algorithm that combines minimizer concepts [M. Roberts et al. 2004] and FM-index with parallel distributed Input-Output supports.

2.2. *k*-mer spectrum based error correction (Trowel 1)

2.2.1. Overview

Trowel 1 is a branch of spectral alignment approach correcting mainly substitution sequencing errors on top of different *k*-mer indices. Unlike conventional methods, which relies on the frequency of *k*-mers, Trowel 1 determines trusted *k*-mers (solids) by base quality values. This fact allows Trowel 1 to catch correct *k*-mers regardless of the sequencing depth. All trusted *k*-mers are called *bricks*, i.e., consecutive stretches of high quality bases (>= q^), to differentiate them from "solids" which are determined by the frequency of *k*-mers. Trowel 1 utilizes two brick indices with different *k*-mer compositions.

The first algorithm, Double Bricks & Gap (DBG), exploits the first index with an asymmetric *k1*-gap-*k2* structure, where gap is a single base, and *k=k1+k2*. This *k*-mer composition allows for a higher accuracy over the second index at repeat element boundaries. Let *k* be the length of *k*-mer, and n be the number of neighboring *k*-mers. The membership queries to find a correct base can be solved in $\mathcal{O}(k+\sigma)$ while it took $\mathcal{O}(k(2+n+\sigma))$ for the other common algorithms storing a single *k*-mer for each entry such as Musket, and BLESS. The quality value of a gap is increased to the maximum quality value when two bricks are exactly matched against the sequence and the gap is of a low quality value. The gap is corrected to another base only if a unique association between two bricks and a high quality value of the gap is found.

The second algorithm, Single Brick & Edge (SBE), is designed to correct continuous ambiguous bases in the middle and erroneous bases at the 5' or 3' end. After the DBG applied, the distribution of *k*-mer changes globally and locally. Thus, the corrected read should reflect the changes by improving the quality values of error corrected bases. The newly created brick indices for SBE can detect more *k*-mers due to improved base quality values, leading to more number of long stretch of high quality bases. The SBE uses an edge-*k*-edge index to correct edges, where an edge is a single base, or increase their quality values as the DBG algorithm does.

2.2.2. Trusted k-mer indexing

2.2.2.1. Parameter k

k determines the uniqueness of a sequence in a given genome. It is defined as follows:

$$k = \left\{ |B| : 1 \le |B| < \min(32, l); |B| \bmod 2 \ne 0 \right\} \quad (2.1)$$

with B being a k-mer, l being the read length. *k* should be an odd number in order to avoid palindromic *k*-mers, which degrade the accuracy of error correction. Since the palindromic *k*-mers are practically rare and thanks to the high precision of the DBG, it is not such that risky to use even numbers. Quake authors [D.R. Kelley et al., 2010] suggested an "optimal estimate" *k* if the following equation is fulfilled:

$$S = \frac{2G}{4^k} \leq 0.01 \quad (2.2)$$

with S called saturation rate and 2G being two times the genome size G, as both strands have to be considered. A proposed *k* derived from the equation does not always guarantee a decent performance and it is difficult to define a model reliably predicting an optimal *k* in a sense that several factors affect the accuracy such as the type of organism, repeats, low-complexity sequences, uneven coverage, quality of sequencing, and error rate. Empirically, an acceptable *k* is known to be in a range [19…27].

A selection of high *k* can significantly reduce the error correction attempts due to the insufficient number of solids though it guarantees an excellent precision. It directly affects the runtime that with a high *k* we can finish the correction much faster than the one with the optimal estimate *k*. A high *k* is better option for species of large genome sizes, of low-complexity sequences, or of highly repetitive genomes with medium- or high-coverage depth. For example, the largest genome, freshwater amoeboid *Polychaos dubium* with an estimate of size 670 Gb [L.W. Parfrey et al., 2008] can be addressed by *k*=27 with sufficiently high saturation rate of ~0.00008.

However, the saturation rate does not account for factors other than genome size. The optimal *k* fluctuates depending on the sequence context, so ideally the parameter *k* should be variable with different sequence compositions. For instance, a sequence taken from a TATA box has very low complexity, which consists of mainly two symbols. To distinguish the errors at the boundary of TATA box needs much longer *k* such that up to the length of TATA sequence. The final remark is about the limitation of all *k*-mer based algorithms employing a fixed *k* including Trowel 1. The formula (2.2) represents the minimum *k*-mer length to obtain an acceptable accuracy with an assumption that uniquely identifiable *k*-mers are evenly distributed across a genome.

2.2.2.2. Parameter q^

q^ is the quality threshold that determines a level of trustworthy of a base at each position. Let T_{ref} be the set of *k*-mers found only in a reference genome, T_{int} be the set of *k*-mers found both in the reference and reads, and E be the *k*-mers exists only in reads. E contains artifacts due to sequencing errors, E_e, and those truly exist but not found in T due to mutations, E_t. Similar to the characteristic of parameter *k*, error corrections with high q^ bring better precision at the expense of reducing the number of correction trials. For high-depth datasets, it is mandatory to apply a more strict criteria, i.e., higher q^, such that one can reduce the number of artifacts mistakenly stored in the brick indices. Low q^ is one of the options to increase the number of correction attempts but artificial sequences from mis-corrections deteriorate the precision. Trowel 1 automatically determines q^ based on the empirical experiments such that all the bases with a quality of q^ or higher cover at least 8% of the input sequences.

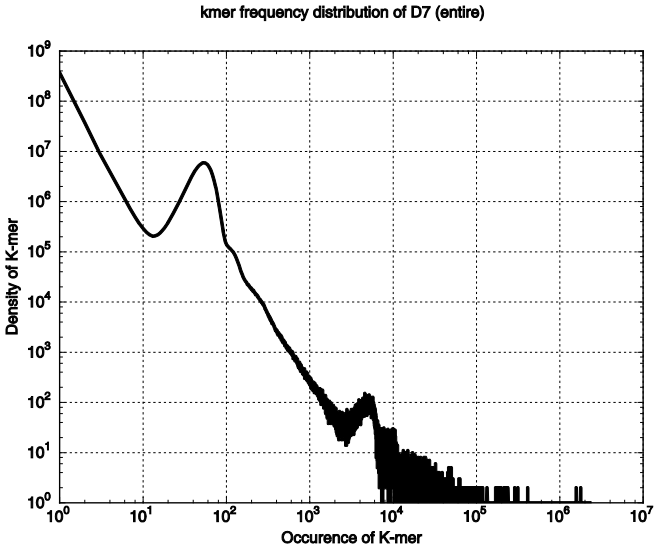

Figure 4. The log-scaled frequency distribution of the entire *k*-mer set of *A. thaliana* reads. [E.C. Lim. et al., 2014].

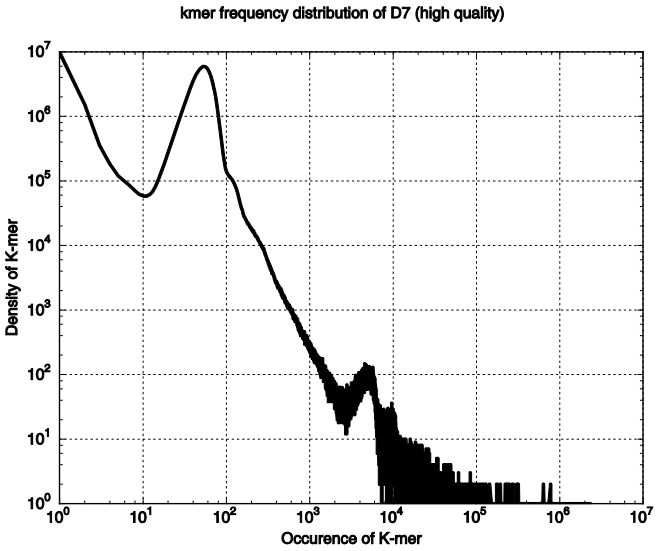

Figure 5. The log-scaled frequency distribution of *k*-mers from high quality regions of *A. thaliana* reads. With the threshold value *q*^, only Trowel can collect *k*-mers of high-quality, but low-coverage. All the other algorithms based on a frequency cut-off discard such high quality *k*-mers. [E.C. Lim, et al., 2014].

k-mer indices built from the frequency of *k*-mers ignore true *k*-mers of low-coverage depth. Even a high-depth dataset suffers from this problem due to unevenly sampled genome regions. I demonstrated the presence of low-coverage but high quality *k*-mers in [E.C. Lim et al., 2014]. The Fig. 4 shows the frequency distribution of all *k*-mers (E+T). Meanwhile, the Fig. 5 illustrates the bricks in T_{int}. One can observe that there are numerous true *k*-mers of low-coverage depth. Trowel's *k*-mer indices include T_{int}, leading to less mis-corrections than frequency-only based methods. For instance, when k=19, Trowel identifies 107,130,202 *k*-mers in T_{int} intersecting between an *A. thaliana* dataset and the reference genome. When a filtering with the frequency cutoff of 4 is applied, the *k*-mer index

based on frequency loses 11,694,294 true *k*-mers, which exist in the reference, leading to a diminution of accurate corrections. This limitation applies to all frequency-based algorithms such as Musket, and BLESS.

2.2.2.3. Construction of brick indices

The *k*-mer indices of Trowel 1 are built on top of trusted *k*-mers where all the contained bases are of higher quality than a threshold q^, and are not ambiguous. Such trusted *k*-mers are called bricks and states highly likely true genome sequence given a cut off q^. Trowel 1 ignores low quality regions, thus any untrusted *k*-mers are avoided being inserted to the brick index.

Let B be a brick where its quality values are denoted as Q. A brick B[b_0:b_k) and its quality values Q[q_0:q_k) are represented as *k*-length pairs. A BQ-pair is defined as follows:

$$(B,Q)=\left\{(b_i,q_i):b_i\in\{A,C,G,T\},q\hat{}\le q_i<2^8;0\le i<k\right\} \quad (2.3)$$

with q representing a Phred score, which is a log-scale error probability induced from the signal intensity [B. Ewing et al., 1998]. Only B is inserted to a brick index if all elements in Q is above the threshold, q^. A brick is encoded as a 64-bit integer reducing memory use by a factor of 4. A sequence exact-matching problem can be solved by a simple numeric equality check. Each brick is associated with an array, the base-quality statistics, denoting the observed maximum quality values of adjacent bases around the brick. A base of a Phred score less than 10 is ignored since the error probability is higher than 0.1.

Trowel 1 constructs a brick index in a block-wise manner. A block is a portion of all bases and quality values with almost identical lengths. The entire strings are split into N chunks of M blocks where N is the number of cores and M is the number of blocks. The lock-freedom of brick indices is gained by applying the thread local concept that each child thread performs the computation with its own storage. The parent thread iteratively merges the bricks from each local storage after the completion of all sub-procedures. A non-blocking technique, Compare-And-Swap (CAS), updates the base-quality statistics as applied in the jellyfish *k*-mer counter [G. Marçais et al., 2011].

2.2.3. Error Correction

The actual correction procedure combines a very stringent Double Bricks & Gap (DBG) algorithm, and a less precise but higher sensitive Single Brick & Edge (SBE) algorithm. The former utilizes two *k*-mers of asymmetric lengths enclosing a single base associating with a base-quality statistics. The latter defines an association between a single brick and two adjacent bases, leading to two base-quality statistics. The DBG inherently has very high precision due to dual supports from adjacent different *k*-mers to determine a true base, thus it should be always preceded the SBE. Both algorithms improves the quality values of corrected bases, so consecutive attempts of each algorithm alters the eventual accuracy of error correction.

The SBE is typically very weak when it comes to corrections on low-depth sequences. It is also fragile to deal with contaminated sequences. With a sufficient length of the secondary *k*-mer, the DBG can keep away from committing such mis-corrections. An improper error correction at the middle of a

read can generate random sequences, which are never found in the true genome. However, SBE is the only way to correct continuous stretches of ambiguous base and the errors at the read boundary.

2.2.3.1. Double Bricks & Gap algorithm

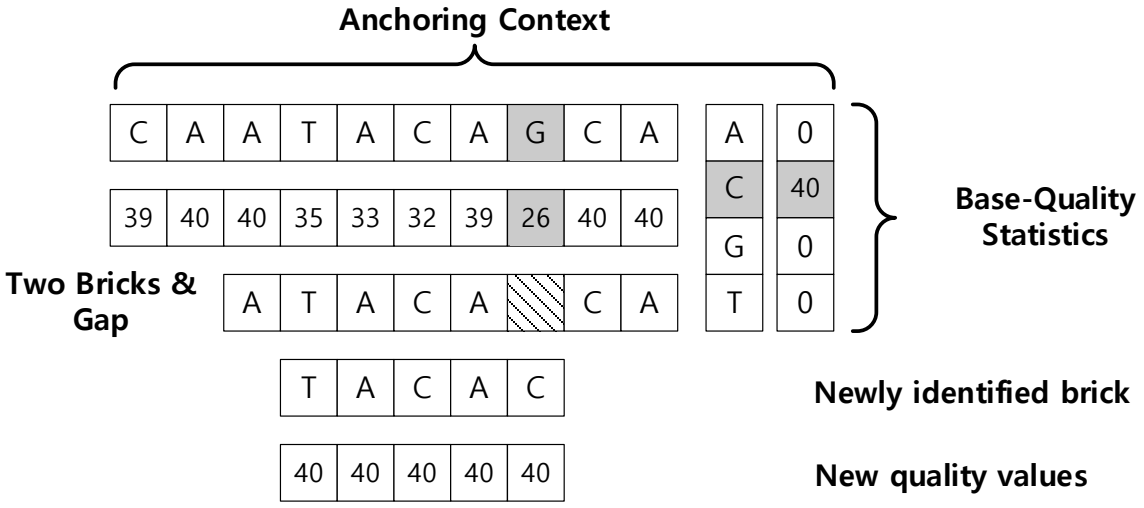

Figure 6. The *k*-mer data structure for the *DBG* algorithm. For the *SBE* algorithm, the data structure does not contain the second *k*-mer (*k2* = 0) [E.C. Lim et al., 2014]

The DBG aims at expanding continuous high-quality bases in the reads, permitting more number of bricks to be identified for successive quality-aware algorithms. Two asymmetric bricks B_1 of length *k1* and B_2 of length *k2* are consecutive bases with a skipping base (gap) at position *k1*, denoting a combined brick of bases $[b_0, \ldots, b_{k1})$ and $[b_{k1+1}, \ldots, b_{k1+k2}]$. *k1* is the length of the primary brick affecting the overall accuracy and the length of the secondary brick, *k2*, furnishes subsidiary accuracy. A short *k2* increases the number of error correction attempts. The gap is deemed to be corrected if the quality is below q^ or the base is ambiguous (Fig. 6).

The “core base correction method” replaces the target bases and improves their quality values. A gap of a low quality value is exchanged with the base of the highest quality value in a base-quality statistics. When ties are present, the algorithm does not change the gap. The DBG polishes a quality value of either a corrected base or a gap found in the base-quality statistics with a high quality value. The method is performed in both forward and reverse complementary directions.

The asymmetric structure of the brick allows to handle a base at the repeat boundary whose repeat sequence is longer than *k* bp. A genome of highly complex organisms consists of countless repeats of variable lengths. It is inevitable to encounter the commonly repeating prefix with a single nucleotide polymorphism (SNP) at the repeat boundary. The successive sequences after the boundary can be completely different from each other.

An experiment illustrates the error correction at the repeat boundary in Fig. 7. Four reads share a common prefix where two are taken from true genome sequence and the others incorporate sequencing errors. Algorithms pivoting on a single k-mer structure literally cannot determine a base at the bounda-

ry despite the correct bases are obvious for our intuition. Methods based on a *k*-mer frequency distribution cannot reliably solve this problem even though we increase the number of error-free reads. The experiments suggested that only Trowel 1 can reliably correct bases at the boundary.

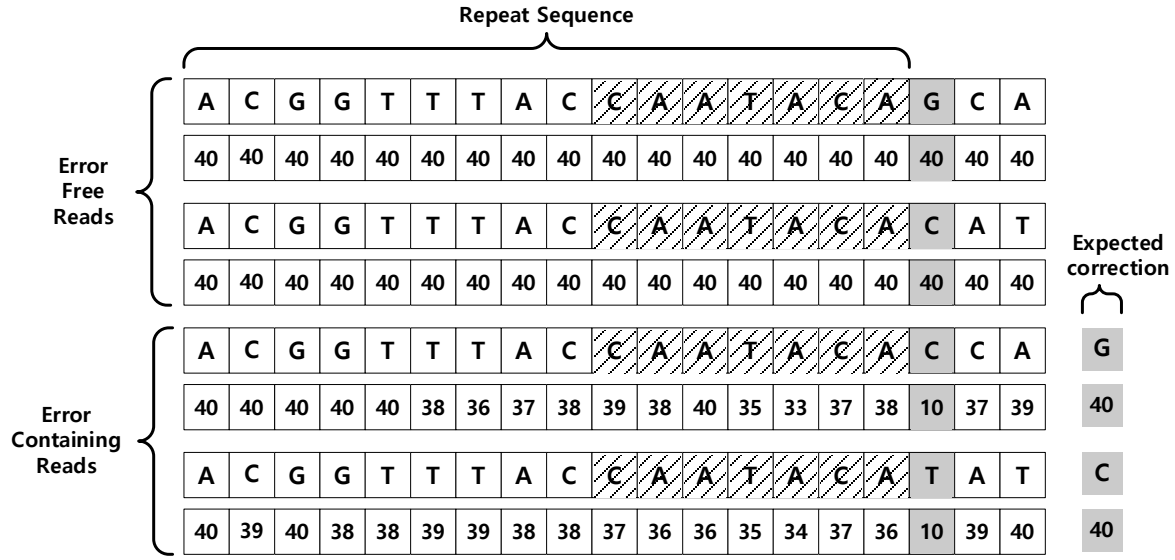

Figure 7. Contradictory base calls at the end of a long repeat sequence (*k* = 9, *k1* = 7, and *k2* = 2). Only Trowel can practically correct these confusing errors at the repeat boundary. [E.C. Lim et al., 2014]

Trowel 1 detects error positions by an error vector. All asymmetric two *k*-mers in a read are numerically compared against the brick index to construct the error vector. The error vector is a bit set indicating the match states of bricks for each position: '0' if an asymmetric brick is found and '1' otherwise. A position of an error is at a (*k*-1) distance from the first '1' of each a stream of '1'. A low quality base appears at the positions of isolated '1's (Fig. 8).

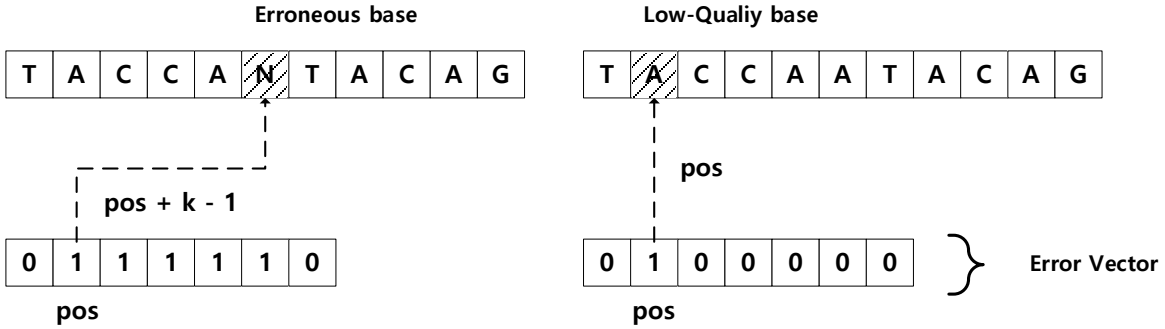

Figure 8. Detection of erroneous or low-quality bases with an error vector (*k* = 5). An error vector determines the location of an erroneous base. [E.C. Lim et al., 2014]

2.2.3.2. Single Brick & Edge algorithm

The SBE creates a new brick index consisting of single ungapped *k*-mers subsequent to the DBG. The *k* of SBE is always larger than the one for the DBG to narrow down the unfavorable accuracy

losses. Bases [b_1, …,b_{k1}) of a brick B and two adjacent bases, called edge, at b_0 and b_{k1} are associated with the base-quality statistics. The SBE is designed to correct consecutive ambiguous bases and the outermost bases with which the secondary brick of the DBG cannot be matched. The "core base correction method" applies to correct edges (Fig. 9).

As discussed, the DBG is highly accurate since more context around the erroneous bases are considered. On the contrary, the SBE does not take into account the successive sequences and solely relying on the uniqueness of a single brick. Trowel 1 reduces mis-corrections of the SBE by not correcting the bases of high quality. Since each algorithm improves a quality value after a successful correction, any subsequent algorithms never override the previous changes, granting a better accuracy.

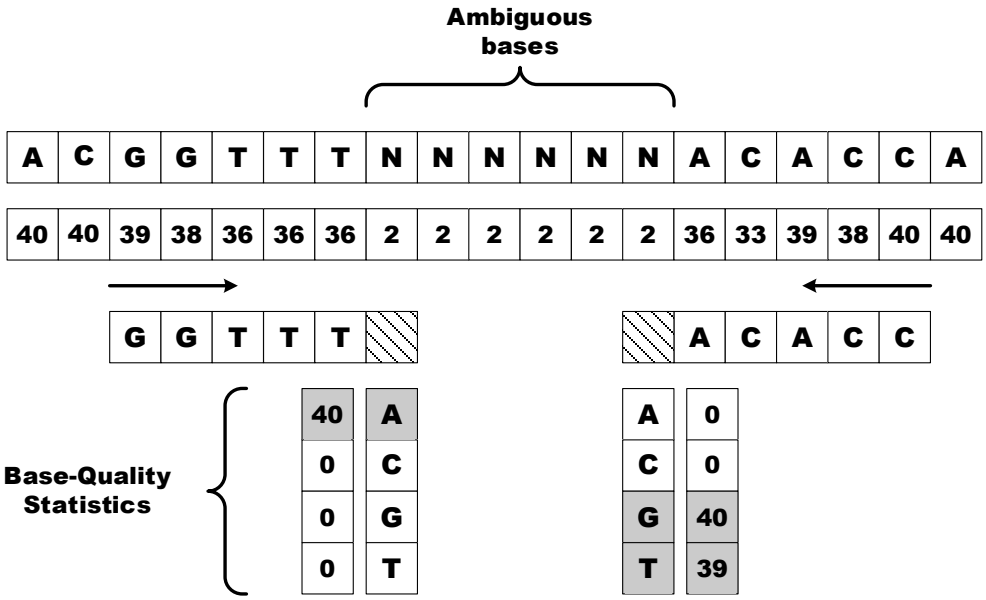

Figure 9. SBE algorithm (*k*=5). SBE algorithm reconstructs consecutively missing bases or corrects contaminated bases at the 3' end in a read.

2.3 FM-index based error correction (Trowel 2)

2.3.1. Introduction

Trowel 2 is a sequencing error corrector targeting the substitution errors on top of the distributed FM-indices. Classical *k*-mer based methods have not provided any robust solution for the fluctuation of accuracy mainly depending on the choice of parameter *k*. A fixed value of *k* dictates overall sensitivity and specificity of error corrections and additional controls to improve the accuracy to change the *k* given contexts are nonviable. Trowel 2 dynamically changes the length of *k*-mers to correct errors by the backtracking algorithm.

The growing volume of sequencing datasets demands for an algorithm to improve time and space efficiency. To address the space efficiency for genomic datasets, the "minimizer" concept has been suggested by M. Roberts and his colleagues [M. Roberts et al., 2004], though the idea has not been actively applied since its invention. A minimizer is the *k*-mer lexicographically smallest in a window accounting for both forward and reverse complementary strands. The sparseness of the *k*-mer index is

inherently achieved with the use of minimizers because a minimizer is shared across consecutive sequences while moving the window. Only a small fraction of all *k*-mers are minimizers, thus the space requirements are significantly reduced with high sensitivity losses. The fact is that the sequencing errors at the window boundaries cannot be corrected. Thus, Trowel 2 can only be applied to very huge datasets, i.e., of size greater than 300 Gb.

Trowel 2 defines "premer", which is a minimizer with a selective first base, to classify all reads into bins. To avoid poly-A sequences, the default selective base is 'C'. This classification of reads notably reduces the space requirement for the FM-indices due to the low complexity of each bin. A bin holds either similar or the same sequences containing a certain premer, thus the diversity of sequences is reduced. Instead of constructing an FM-index for the entire reads, only a proportion of reads is applied to construct temporary FM-indices. The algorithm finishes the error correction in a single-pass since the information for any combinations of *k*-mers are accessible without rebuilding the indices due to the properties of the FM-index.

2.3.2. Distribution of reads

Trowel 1 builds two brick indices over whole datasets so that it spends much more time and space for large datasets to build indices rather than error corrections themselves. Trowel 2 splits the whole datasets into small chunks, where of each contains similar sequences, to save the memory consumption and time. The assumption behind is that two reads of totally different sequences never be sampled from the same genome location. If two reads do not have any similarity, the error correction can be independently performed. This atomicity is highly important for the parallelization.

The raw sequencing reads do not have a particular order, thus splitting a dataset into equal-size blocks causes a large deviation among block-wise *k*-mer indices. In an extreme case, a single block may contain the whole *k*-mers, breaking the sparseness, and weakening the purpose of such clustering. The size of *k*-mer indices is directly proportional to the number of premers. It is practically impossible to assign the exactly equal number of premers to each block because when it comes to the parallelization each block should be defined based on the length of text T rather than the total number of premers. To calculate the size of each block, the reads should be classified by premers. Let *k*-mer be K, bit-encoded *k*-mer be E, and a premer be P as follows:

$$\begin{aligned} K &= \left\{ K_i : K_i \in \{\mathrm{A},\mathrm{C},\mathrm{G},\mathrm{T}\}; 0 \le i < k \right\} \\ E &= \left\{ E_i : E_i \in \{00,01,10,11\}, \mathrm{E}_0 = \mathrm{c}; 1 \le i < k \right\} \\ P &= \min(E) \end{aligned} \quad (2.4)$$

with c is the selective base. The length of a premer is shorter than usual parameter *k*, i.e., in a range of [11…15]. To obtain a premer in a read, it is the same procedure as to collect a minimizer, but with an exception that a premer should be started with the selective base c. The entire dataset is divided into N-blocks, where N is the number of cores, in order to collect all premers. Each thread stores premers in a local set. Ideally, the total length of reads in each bin should be uniformly distributed, which can be achieved by adjusting the number of premers in each bin. Since a short read is expected to have a single premer, we can roughly calculate the length of T as follows:

$$len(T) = \sum_{i=1}^{n} len(R_i) \ge \sum_{j=1}^{m} L(P_j) \quad (2.5)$$

where L returns the total length of reads sharing the same premer, m is the number of premers, R is a read, and n is the number of reads. The sum of L is often smaller than the length of T because some reads do not have premers. Assuming that the selective base is 'C', for example, the L value of a poly-A sequence is zero while it is possible to calculate len(R_i). Above L-calculation is performed by following the numerical order of premers. The L indicates how abundant a premer in the entire datasets. Given two premer P1, and P2, L(P1) < L(P2) if Occ(P1) < Occ(P2). Let D be the number of blocks. The overall size of B approximate to len(T)/D. Hence, when the local sum of L exceeds B, P_{j-1} becomes the block boundary. If a premer is shared by highly abundant repeat sequences, or if a block contains even this single premer, the size of block would be greater than B. Due to these two reasons, the number of blocks are always smaller than D.

All reads are distributed and distinguished by premers. The direction of reads can be changed when the premer is found in the reverse complementary of R. The memory consumption is reduced by factor D. For instance, when the total length of the text is 300 Gb and D is 1024, the maximum memory consumption of each core becomes around 300 Mb. The actual memory consumption is 9 Gb and 19 Gb for 32-core and 64-core machine, respectively. This space usage is far less with respect to the original size of T due to extra compressions. With larger D, the memory consumption can be drastically reduced at the expense of sensitivity.

2.3.3. The construction of the FM-index

The FM-index is built for both forward and reverse directions. The forward index supports the backward searches when the error correction starts at the 3' end. The reverse index does not represent the reverse complementary strings, but just reverse strings, because the reads have already changed the direction in the previous stage. Note that it is for forward searches. However, the increase in the space requirement is not profound due to the compressibility of the FM-index.

Though each distributed block does not have the equal size, the reads in a block have a high probability of sharing the same or very similar sequences thanks to the previous distribution step. T^{BWT} would be more compressed by a run-length encoding (RLE) scheme. The *k*-th order entropy for those consecutive runs of symbols is much lower by following the equation, $\gamma \leq nH_k + \sigma^k$. The actual memory consumption can be quite much trivialized as shown in a straightforward calculation based on D.

2.3.4. Error correction

An FM-index allows for the use of variable-length *k*-mers without rebuilding the indices. Unlike Trowel 1, the correction finishes in a single pass since any composition of *k*-mers can be accessed in real time. Fundamentally, the error correction is based on the DBG algorithm. The SBE can be easily emulated by limiting the length of the second brick to 0. The parameter *k* indicates the minimum length of the first brick. When there is an erroneous base after the first brick sequence compared, the algorithm increases the length of the second brick until the occurrence of the second brick becomes 0. The erroneous base can be detected where the occurrence of the first brick becomes 1 or the base of the position is ambiguous. However, the 1-occurrence after the backtracking does not always indicate that the base is erroneous since the read can be a rare sequence. Or else, the read can be derived from contamination or chimeric read due to PCR amplification. Hence, after the two consecutive 1-occurrence events, the error correction should be stopped. When there are more than or equal to 2 can-

didate bases, Trowel 2 takes the base yielding the longest second brick. The error correction fails if all the candidates are of 1-occurrence or the quality value of the erroneous base is higher than q^.

Table 2. Datasets used for evaluation taken from the original Trowel paper. See more detailed description in text. [E.C. Lim et al., 2014]

Dataset	Identifier	Reference Genome	Fragment Size (bp)	Read Len. (bp)	# reads	Coverage Depth
D1	SRR001665	*E. coli* K-12 MG1655	200	36	20,816,448	163
D2	ERR022075		600	100	28,428,648	618
D3	SRR022918		3,000	47	14,408,630	147
D4	SRR352384	*S. cerevisiae* S288C	300	76	52,061,664	319
D5	SRR022866	*S. aureus* MW 2	169	76	25,551,716	691
D6_1	SRR060098	*D. melanogaster* release 5	200	95	37,921,094	26
D6_2	SRR018294		200	75	18,927,440	10
D6_3	SRR018292		200	45	24,483,260	8
D6_4	SRR018293		200	45	17,088,290	6
D7	-	A. thaliana TAIR9	250	151	79,810,700	86

SRA – NCBI Short Read Archive accession number

2.4. Evaluation

I have implemented two different version of error correction algorithms, Trowel 1 and Trowel 2. I do not include the evaluation results for Trowel 2 in this study since the goal of Trowel 2 is to perform the error correction very quickly for huge datasets at the expense of far lower sensitivity than Trowel 1. Thus, I only include the results for Trowel 1, which will be referred to as "Trowel".

To evaluate the performance of Trowel, I made use of the high quality gold standard reference genome for *Arabidopsis thaliana* [The Arabidopsis Genome Initiative (AGI), 2000, S. Ossowski. et al., 2010]. The reference sequence was generated from a homozygous line, Col-0, with a genome of size approximately 150 Mb, of which about 120 Mb are accessible to genome assembly. I used paired end 151-bp Genome Analyzer II reads generated in house for evaluation. The reads came from a line closely related to the sequenced Col-0 individual, estimated to have diverged by fewer than 10^{-6} mutations per site [S. Ossowski et al., 2010]. Additional datasets were paired end Illumina datasets referenced in X. Yang's survey [X. Yang et al., 2012] for *E. coli*, *S. aureus*, *S. cerevisiae* and *D. melanogaster*, with different read lengths and genome coverages (Table 2). All evaluations were conducted on a single 64-bit Linux machine, using 32 of the machine's 64 AMD Opteron 6274 (2.2 GHz) cores and 512 GB main memory.

Table 3. The number of total mapped (exact match) uncorrected reads and of corrected reads by Hybrid SHREC. Hybrid SHREC rather introduces arbitrary sequences. [E.C. Lim et al., 2014]

Dataset	Uncorr. Read 1	Corr. Read 1	Uncorr. Read 2	Corr. Read 2
D1	9,054,243	4,477,004	8,621,037	4,495,852
D2	8,921,906	349,984	7,266,868	344,438
D3	2,519,012	121,155	0	1,208
D4	15,729,438	3,161,223	15,096,670	2,999,542
D5	3,507,055	99,632	3,193,664	67,932
D6_1	9,598,925	9,032,759	8,467,088	8,216,740
D6_2	4,122,144	3,435,285	4,117,820	3,407,190
D6_3	6,519,677	3,499,010	5,053,354	2,866,824
D6_4	4,609,152	3,264,963	4,732,671	3,231,347
D7	29,073,732	7,780,713	6,740,204	2,883,296

Both stand-alone error correction tools and modules integrated into the pre-processing step of genome assembly tools are available [R. Luo et al., 2012, S. Gnerre et al., 2011]. I initially have tried to evaluate the performance of Reptile [X. Yang et al., 2010], ECHO [W.C. Kao et al., 2011], SHREC [J. Schröder et al., 2009], and HiTEC [L. Ilie et al., 2010]. However I excluded those results from the study since they failed to yield complete output with default or optimal parameters according to guidelines given by their authors when applied to *A. thaliana* reads. Moreover, Hybrid SHREC [L. Salmela, 2010] changed the sequence IDs while performing the correction. Thus, I could not directly quantify its accuracy. Instead, I counted the total number of exactly matched reads before and after applying Hybrid SHREC. The number of reads usually has been drastically reduced, which indicates that Hybrid SHREC creates artifacts rather than true sequences present in the genome, and that the error corrected reads contains more errors than the original reads (Table 3).

2.4.1 Accuracy

To evaluate the accuracy for *A. thaliana* DNA genomic data, I first aligned uncorrected and corrected reads with BWA [H. Li et al., 2009a] against Col-0 reference genome of version name, The Arabidopsis Information Resource (TAIR) 9, allowing for a small number of mismatches depending on read length. The assumption behind is that the reference does not contain any errors though it is untrue. In practice, it is not possible to evaluate the accuracy of the error correction without such assumption. The following definitions are used for base-level (read-level) measurements: TP, erroneous base accurately corrected (unalignable read become alignable); FP, error-free base inappropriately "corrected" (alignable read become unalignable); TN, unchanged error-free base (unchanged alignable read); FN, unchanged erroneous base (unchanged unalignable read); Sensitivity: |TPs| / (|TPs| + |FNs|), fraction of accurate corrections for erroneous bases (fraction of improved mappability of reads); Specificity: |TNs|

/ (|TNs| + |FPs|), fraction of not altering error-free bases (fraction of not altering mappability of alignable reads); Precision: |TPs| / (|TPs| + |FPs|), fraction of accurate corrections in given correction trials (fraction of accurate improvements of mappability in given alignment trials); F-Score: 2 * (Precision * Sensitivity) / (Precision + Sensitivity), harmonic means of Precision and Sensitivity; Gain: (TPs-FPs) / (TPs + FNs), percentage of corrected (positive) or introduced errors (negative) (percentage of improved or degraded mappability)

I adopted the definition of base-level accuracy metrics from Yang's survey [X. Yang et al., 2012]. However, I do not agree with the use of Yang's toolkit due to the discrepancy between the definitions and the software implementation. The Error Correction Evaluation Toolkit (ECET) has serious problems in its implementation, yielding arbitrarily results and the true statistics are unknown. More specifically, this toolkit does not count correctly TPs, FPs, and FNs. Unfortunately, so many researchers have never verified the correctness of the toolkit and cited the paper without any doubts. Instead, I directly calculated all metrics based on the Sequence Alignment/Map(SAM) files.

Table 4. Performance evaluations for base level accuracy of Trowel. The statistics were derived from SAM (Sequence Alignment/Map) files of uncorrected and corrected reads [E.C. Lim et al., 2014]

Edit distance	TNs	TPs	FPs	FNs	# bases	Sensitivity	F-Score	Gain
1	702,787,200	26,179,893	35,553	923,091	27,138,537	96.59	98.20	96.46
2	718,371,346	11,266,109	9,256	279,026	11,554,391	**97.58**	**98.73**	**97.50**
3	723,056,071	6,679,359	2,184	188,123	6,869,666	97.26	98.59	97.22
4	724,394,357	5,368,072	1,452	161,856	5,531,380	97.07	98.50	97.04
5	724,746,204	5,028,592	1,160	149,781	5,179,533	97.10	98.52	97.08
6	724,862,818	4,918,205	1,088	143,626	5,062,919	97.16	98.55	97.14
7	724,877,571	4,898,490	1,126	148,550	5,048,166	97.05	98.49	97.03
8	724,877,566	4,897,114	1,131	149,926	5,048,171	97.02	98.48	97.00
9	724,877,566	4,897,114	1,131	149,926	5,048,171	97.02	98.48	97.00

Table 5. Performance evaluations for base level accuracy of Trowel. The statistics were calculated by ECET TEF (Error Correction Evaluation Toolkit Target Error Format) files of uncorrected and corrected reads [E.C. Lim et al., 2014]

Edit distance	TNs	TPs	FPs	FNs	# bases	Sensitivity	F-Score	Gain
1	-	453,448	8,489	934	462,871	**99.79**	**98.97**	**97.92**
2	-	536,490	17,129	26,471	580,090	95.29	96.09	92.25
3	-	568,066	20,352	43,339	631,757	92.91	94.69	89.58
4	-	579,585	20,912	49,395	649,892	92.14	94.28	88.82
5	-	583,137	20,956	50,600	654,693	92.01	94.21	88.70
6	-	584,152	20,995	50,781	655,928	92.00	94.21	88.69
7	-	584,392	20,993	50,809	656,194	92.00	94.21	88.69
8	-	584,375	21,001	50,811	656,187	92.00	94.21	88.69
9	-	584,375	21,001	50,811	656,187	92.00	94.21	88.69

Table 4 and 5 clearly demonstrate the incorrectness of the ECET. Though I have reported TPs in Table 5 according to the outputs from the ECET, they are actually TNs by definition. Thus, I was able to induce that the ECET ignores many alignments and, in turn, the statistics are incorrect. Moreover, since TPs in Table 4 are actually TNs, the additional metrics such as F-Score, and Gain are also inappropriate. In addition, one cannot calculate the read-level accuracy with this toolkit.

Table 6. Performance evaluation of read level accuracy for the *A. thaliana* dataset (D7). The statistics reflects changes in mappability of alignable reads. [E.C. Lim et al., 2014]

Data	Sensitivity	Specificity	Precision	F-Score	Gain
Quake	52.00	94.43	76.59	61.95	36.11
Coral	39.53	95.76	76.57	52.15	27.44
Musket	37.36	99.94	99.58	54.33	37.20
SOAPec	46.47	98.69	92.53	61.87	42.72
Trowel	**67.65**	**99.99**	**99.99**	**80.70**	**67.64**

The best in each column is highlighted in bold.

Table 7. Performance evaluation of base level accuracy for the *A. thaliana* dataset (D7). The statistics reflects changes in bases. [E.C. Lim et al., 2014]

Method	Sensitivity	Specificity	Precision	F-Score	Gain
Quake	**67.81**	**99.98**	**99.96**	**80.81**	**67.78**
Coral	39.46	94.77	83.15	53.52	31.46
Musket	35.15	99.84	99.32	51.93	34.91
SOAPec	53.36	99.93	99.78	69.53	53.24
Trowel	65.75	99.97	99.91	79.31	65.69

The best in each column is highlighted in bold.

The read level accuracy and base level accuracy has a hierarchical meaning as a lower read-level accuracy indicates that the number of alignments is less than the other algorithms. The number of alignments directly affects the accuracy in a long-range context and the number of bases to be accounted in order to calculate the base-level accuracy. Hence, for those having similar base-level accuracy values, one should compare the read-level accuracy altogether. For example, Trowel yields similar base-level accuracy with Quake, which is the best, for the *A. thaliana* dataset (Table 7). However, Quake's read-level accuracy is far lower than the Trowel (Table 6). One can induce that Trowel is of higher accuracy than Quake due to its higher read-level accuracy. Additional tables are presented in [E.C. Lim et al., 2014]. Trowel was found among the two most accurate tools for all datasets. For read (base) level accuracy, Trowel was the best in 8 (4) out of 10 cases and the second best in 2 (5) out of 10 cases.

To consider the effect of the parameter k, I evaluated the standard metrics on the all datasets with different k-mer sizes but with a fixed edit distance. Exploring all parameter spaces is time-consuming and meaningless. We can observe the overall performance differences even with the fixed edit distance. All additional figures are accessible in [E.C. Lim et al., 2014]. I select the figures for the *A. thaliana* datasets to show the performance changes in terms of the parameter k (Fig. 10, 11). The figures for the specificity are not shown since the differences are indistinguishable. I also omit the figures for F-score and gain since their trends are similar with those for sensitivity.

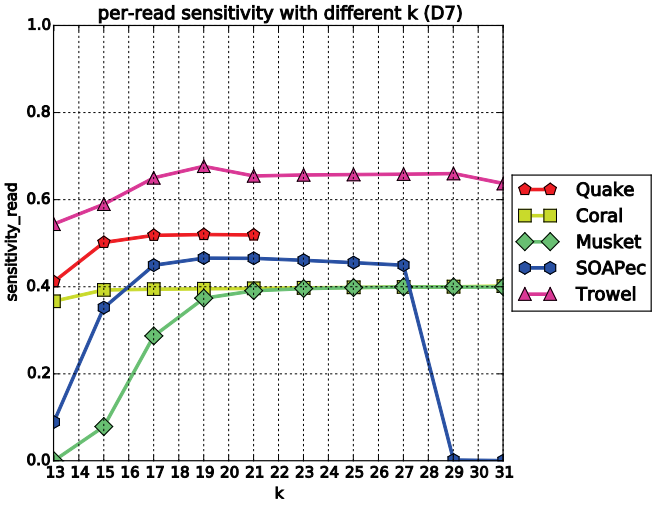

Figure 10. Per-read sensitivity of D7 dataset with different *k* (optimal estimated *k* = 19). Missing points indicates that the error corrections failed. Trowel is the best at *k* of 19. Quake is the second best at *k* of 19**.** [E.C. Lim et al., 2014]

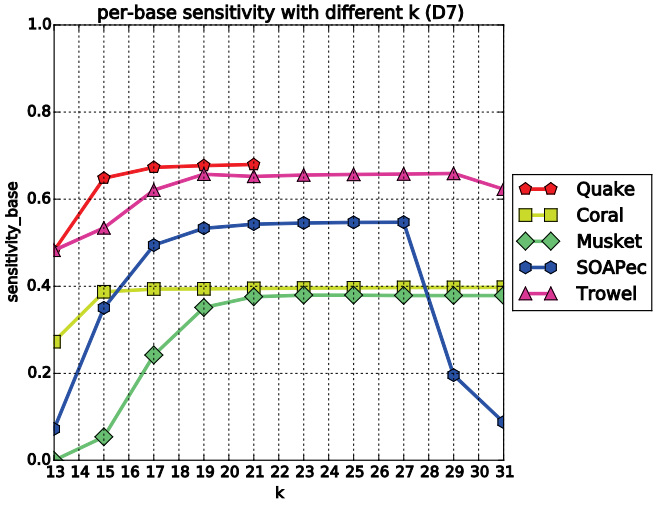

Figure 11. Per-base sensitivity of D7 dataset with different *k* (optimal estimated *k* = 19). Missing points indicates that the error corrections failed. Quake is the best at *k* of 21. Trowel is the second best at *k* of 29. [E.C. Lim et al., 2014]

Quake often appeared in the best performing tool in terms of base-level accuracy though the range of valid measurements are limited due to the error correction failures. Trowel achieved the best read-level accuracy for almost all the experiments regardless of the changes in parameter *k*. There are 80 measurements in total and Trowel is the best and the second best in 49 and 21 cases, respectively. As indicated by the read-level accuracy, Trowel's strategy does not change the original mapping status from alignable to unalignable but does in reverse direction, leading to a higher number of sequence alignments in paired-end mode. All *k*-mer based algorithms lose the accuracy with the *k* below the value derived by the saturation rate since those *k*-mers cannot guarantee the uniqueness within a ge-

nome. The *k*-mers become more common as the length gets shorter. A larger *k* decreases the number of error correction trials since the coverage of bricks also drops. The loss of the number of bricks in the figure is not in a continuous trend but rather a sudden event.

The sequence alignment is a key technic to analyze the biological meaning in a quantitative way. Hence, the changes in alignment status are important measure for the quality of the error corrections. Some sequencing error correction modules trim out the reads. One may conjecture that the trimming has beneficial effects on the quantity of the sequence alignments. I evaluated the read-level accuracy whether an algorithm trims out reads or not. According to the experiments, Quake and SOAPec do not improve the alignment status that much even with their trimming strategy. Trimming of both tools can be regarded rather as the loss of sequence information. A positive trimming strategy increases the amount of sequence alignments as Trowel does. For non-trimming algorithms such as Musket and Coral demonstrate the similar trends in both base- and read-level accuracy. I can conclude that only Trowel's trimming strategy is productive according to all experiments.

2.4.2 Genome Assembly

The evaluations in this section are indirect measurements of the accuracy for the error correctors. If a dataset were influenced by contaminations, a genome assembler would produce artificial contigs. Hence, when we align such contigs against the reference genome, the matches will be hardly found. A successful algorithm may trim off those contaminations, leading to a better connectivity of genome assemblies. This evaluation can capture such improvements while the alignment-based method cannot. The failures in error corrections propagate through the nodes in a graph while assembling the sequences, leading to mis-assemblies or artificial bases within the contigs.

2.4.2.1 QUAST report

I assessed the quality of de novo genome assemblies by QUAST adopting its metrics [A. Gurevich et al., 2013]. QUAST utilizes the NUCmer aligner from MUMmer [S. Kurtz et al., 2004] in order to align scaffolds to a reference genome in the genome scale. The metrics are based on scaffolds to account for the mis-assemblies. The adopted definitions of metrics are as follows:

- # scaffolds (>= 0bp) is the number of all scaffolds.
- N50 is the length of the smallest scaffold in the list of scaffolds sorted by the length that make up half of the bases of the assembly.
- # mis-assemblies is the number of positions in the assembled scaffolds where the left flanking sequence aligns over 1 kb away from the right flanking sequence on the reference (relocation), or where they overlap by more than 1 kb (relocation), or where flanking sequences align on different strands (inversion) or on different chromosomes (translocation).
- Cov. (%) is the total number of aligned bases in the reference, divided by the genome size.
- Max. scaffold is the length of the longest scaffold in the assembly.

The metrics have potentially different priorities with respect to interpreting the accuracy of sequencing error corrections. The order is as follows: Cov (%), # mis-matches and # mis-assemblies, # scaffolds, N50 (bp), and Max. scaffold. It is difficult to determine which is better when two assemblies have different Cov (%) less than 3% and 1% for a high and low coverage dataset, respectively. Then one should consider the next metrics, both # mismatches and # mis-assemblies, at the same time. For high coverage datasets, Trowel is beneficial for high coverage datasets in terms of # mis-matches and # mis-assemblies while preserving Cov (%) compared with uncorrected cases. The most prominent benefit is demonstrated in the metrics for D3 that Trowel is the only algorithm significantly improves the genome assembly for short mate-pair library though such very short reads are not common practice nowadays [E.C. Lim et al., 2014].

Table 8. QUAST results and memory consumption of *A. thaliana* genome assembly (D7). Cov (%), # miss-matches and # mis-assemblies are indirectly related to the accuracy of an error correction algorithm. [E.C. Lim et al., 2014]

Assembly	Tool	# scaffolds	N50 (bp)	# mis-matches	# misas-semblies	Cov. (%)	Max. scaf-fold (bp)	Peak memory (Gb)
SOAPdenovo	**Uncorr.****	64,828	59,212	3,298	67	93.8	414,649	17.9
	Trowel*	59,318	61,674	3,711	65	93.8	405,218	17.9
	Coral	97,070	24,207	61,078	810	91.6	230,035	11.4
	Musket	64,231	59,813	4,847	68	93.7	389,029	11.4
	SOAPec	64,706	59,291	3,424	64	93.7	389,103	11.4
	Quake	64,205	60,980	3,374	67	93.8	321,891	11.4
Velvet	**Uncorr.****	24,087	34,011	18,950	812	96.1	207,105	42.9
	Trowel	23,206	37,663	19,570	819	96.0	221,390	36.4
	Coral	-	-	-	-	-	-	-
	Musket	23,069	38,957	20,971	897	96.0	221,319	31.1
	SOAPec*	23,836	35,393	18,393	740	96.0	204,812	29.1
	Quake	23,654	35,267	18,914	867	95.9	240,981	25.9

I suggest a guideline for the interpretation of metrics (Table 8). When Cov (%) is degraded after the error correction, some information are lost. For very low coverage datasets (D6_3 and D6_4), only Coral and Trowel maintain the true information. The number of mismatches and mis-assemblies are expected to be low since individuals are close to the reference organisms. # mismatches and mis-assemblies are both relied on the changes in Cov (%). A positive error correction would reduce # mismatches and mis-assemblies as Cov (%) goes down. The remaining metrics such as # scaffolds, Max. scaffold, and N50 are rather directly related to contiguity than accuracy of genome assembly. Hence, their priorities are lower than the former metrics.

Different de novo assemblers do not yield the same contigs and their accuracy also not equal. Hence by analyzing the results from two different de novo assemblers, one can reduce the chance of

mis-interpretations. The Velvet [D.R. Zerbino and E. Birney, 2008], and SOAPdenovo [R. Luo et al., 2012] are used for all comparisons. The general tendency is that the corrected reads improve N50 metrics while the number of mis-assemblies varies little. In the best case, genome coverage is significantly increased, but the requirement is that the coverage-depth should be high enough. For instance, the metrics for D5, N50 is improved to 62 kb (240x longer than the uncorrected dataset) for SOAPdenovo and to 38 kb (613x) for Velvet after applying Trowel without read trimming. The read trimming is beneficial for SOAPdenovo that results in 256 times longer N50, but not for Velvet since N50 is only 551 times longer [E.C. Lim et al., 2014].

Since one can apply the sequencing error correction to each library independently, the net memory consumption for de novo assemblies can be reduced. For example, Quake and Trowel reduces the consumption from half up to one fifth of the originals. Higher coverage depth improves the Cov (%). Four *D. melanogaster* datasets contain different level of coverages, so we can observe the Cov (%) changes accordingly. Trowel accomplishes such boosts in 30 out of 44 cases. A genome assembly is obtained from the mixed library for *E. coli* of D2 and D3, where noticeable benefits over uncorrected reads are found in Cov (%) and memory consumption. Relevant supporting tables are accessible in [E.C. Lim et al., 2014].

As a note, no *k*-mer based error correction algorithm is suitable for low coverage datasets (< 10x) since a *k*-mer based method can significantly decrease Cov (%). The lack of true *k*-mers introduces high number of false positives, leading to invalid graph representation. Interestingly, an alignment based module, Coral, performs well for low coverage datasets.

2.4.2.2 The number of mis-assemblies and mismatches

The mis-assemblies can be true structural variations or software errors in scaffolding stages or caused by missing sequences in the reference genomes. The true number of mis-assemblies is unknown. We can only estimate it by taking the mean or median value of # mis-assemblies for all the experiments. For high coverage datasets, assemblies derived from Coral corrected reads demonstrate strangely high # mis-assemblies. Thus, one can speculate that Coral corrupts the graph representation by introducing artificial sequences and confounds the graph traversal by changing the alignment status. This inclination can also be observed in the low-coverage datasets. # mis-assemblies varies little among tools except for Coral.

The mismatches represent either true single nucleotide polymorphisms (SNPs) or sequencing errors. By the same way as for # mis-assemblies, one can approximate the true number of mismatches. The assemblies generated with Coral corrected reads have failed to produce the mean or median number of mismatches in 29 out of 44 experiments. This record is followed by Musket, which is the second worst in 25 out of 44 cases. More specifically, two assemblies for *A. thaliana* from Musket corrected reads contain about 7,000 and 1,000 more mismatches than Trowel's metrics in non-trimming and manual trimming cases, respectively. The worst performance for *A. thaliana* is obtained by Coral, where SOAPdenovo assembly contains about 41,000 and 57,000 more mismatches than those from Trowel in non-trimming and manual trimming cases, respectively. Even worse, the Velvet failed to yield a valid assembly due to high number of artificial bases in the reads.

2.4.3 An erroneous-base-next-to-repeats problem

I consider two bricks and a gap (*k1* + gap + *k2*) structure is better in detecting the erroneous base (gap) compared with existing methods, which simply use (*k* + gap) structure to identify the "gap",

where k can be chosen to be $k = k1 + k2$. When k is shorter than the repeat sequences, one might need to increase k if we use a (k + gap) structure. For ($k1$ + gap + $k2$) structure, k is "equal to or much less than" ($k_1 + k_2$). Assume that we have two reads of length 151 bp, the repeat sequence of 146 bp long without any errors, and the last four bases are totally different. We could assign 13 to $k1$, and 2 to $k2$. We need an equivalent length of 16 bases for representing both structures: (13+1+2) and (15+1). However, their abilities to detect errors are significantly different. The ($k1$ + gap + $k2$) structure can identify which base is correct at the gap position while (k + gap) one does not. To identify the same case with (k + gap) structure, we need to assign at least 147 to k, which is much longer than 15 (13+2). This is apparently why other error correction modules cannot correct the erroneous bases next to repeats. According to additional experiments (not shown), the changes in correcting direction never lead the other error correction modules to performing accurate error corrections.

The DBG algorithm is unique in a sense that the sequences of length longer than the parameter k can be corrected. I performed the evaluation on a set of simulated datasets by varying the coverage ratio of F to E, where F is the relative number of error-free reads and E is the relative number of error containing reads. The total number of reads is T=E·F·U, where U is the number of repeat instances with unique flanking sequences. A repeat instance consists of a common prefix and uncommon successive bases. After the error correction the number of unique sequences U' is counted to determine if the correction succeeds. The number of erroneous reads is E·U.

Since most k-mer based algorithm employs the low frequency cutoff, I changed the U in a range (2…4) to observe if their k-mer filtering strategies are appropriate. When U is 2, i.e., the total number of reads with the ratio 20:1 is 42 and with the ratio of 2:2 it becomes 8. Given U=2, only two unique error-free read instances should be kept after the error correction; if the error correction failed, one can observe U'>2 (conversions to unrelated error-free reads) or U'<2 (insufficient sensitivity). I simulated two reads of length 35/160 bp with a common prefix of length 17/143 bp at the beginning. When k=13, all k-mer based algorithms are expected to correct all the error-containing reads according to their descriptions in the papers.

Table 9. The number of accurately corrected reads (reads converted to incorrect error-free ones) for the base next to repeats problem. The read length is 160 bp. [E.C. Lim et al., 2014]

Tool	20:1(*U*=2)	20:2(*U*=2)	20:3(*U*=2)	20:4(*U*=2)
Expected	**2(0)**	**4(0)**	**6(0)**	**8(0)**
Trowel	**2(0)**	**4(0)**	**6(0)**	**8(0)**
SHREC	1(0)	2(0)	3(0)	0(0)
Coral	**2(0)**	**4(0)**	**6(0)**	**8(0)**
Musket	1(0)	2(0)	0(0)	0(0)
SOAPec	**2(0)**	**4(0)**	**6(0)**	0(0)
Quake	0(0)*	**4(0)**	**6(0)**	0(0)*

The column header means the ratio of the number of error-free reads to the number of error containing reads. The best in each column is highlighted in bold. *: Optimization of distribution likelihood function to choose k-mer cutoff failed. A negative value indicates the number of error-free reads that were removed.

Table 10. The number of accurately corrected reads (reads converted to incorrect error-free ones) for the base next to repeats problem. The read length is 160 bp. [E.C. Lim et al., 2014]

Tool	1:1(*U*=2)	2:2(*U*=2)	3:3(*U*=2)	4:4(*U*=2)
Expected	**2(0)**	**4(0)**	**6(0)**	**8(0)**
Trowel	**2(0)**	**4(0)**	**6(0)**	**8(0)**
SHREC	1(1)	2(2)	3(3)	0(0)
Coral	0(0)	0(0)	0(0)	0(0)
Musket	0(0)	2(2)	0(0)	0(0)
SOAPec	-2(0)	-4(0)	-6(0)	0(0)
Quake	0(0)	0(0)*	0(0)*	0(0)*

The column header means the ratio of the number of error-free reads to the number of error containing reads. The best in each column is highlighted in bold. *: Optimization of distribution likelihood function to choose *k*-mer cutoff failed. A negative value indicates the number of error-free reads that were removed.

Trowel is the only algorithm not committing any mis-corrections for all the 48 experiments (Table 9, 10) [E.C. Lim et al., 2014]. Quake fails in 34 experiments due to the incapability of finding a cutoff in the *k*-mer frequency distribution. SHREC converts some erroneous reads to unrelated error-free reads in 9 cases. Musket ignores the errors in 24 cases, thus the number of such invalid conversion is found in only few cases. However, such low-sensitivity can be problematic. Coral avoids all unrelated conversions, but it does not mean that Coral performs perfect error corrections. In fact, it either randomly ignores error-containing reads or changes the error-free reads to unknown sequences in total 41 experiments. The most problematic algorithm is SOAPec, which removes the error-free reads by random trimming.

Those conversions to unrelated error-free reads cannot be captured by a standard measurement since these conversions increase the number of true positives in an invisible way. The effects of these conversions can only be observed in the paired-end mode alignments or whole genome alignments on the genome assemblies. The number of mismatches and mis-assemblies reflect their effects in long-range contexts.

2.4.4 Runtime and memory consumption

Trowel is designed for a single mainframe with built-in multi-core processors and huge shared memory. Trowel is the one of best algorithms in terms of runtime. For all the datasets, Trowel excels all the competitors, reducing runtime between two and 100-fold (Table 11). For *A. thaliana* dataset, Trowel is three times faster than Musket, which is a strong competitor. The lock-free data structures along with well-balanced threading models allow Trowel to scale up as data volume increases. The Compare-And-Swap (CAS) technique also alleviates the delay in the update operations.

Table 11. Runtime of error correction modules (min). [E.C. Lim et al., 2014]

Data	Quake	Coral	Musket	SOAPec	Trowel
D1	261.1	43.9	4.1	130.5	**2.9**
D2	377.9	228.9	18.0	194.1	**9.9**
D3	119.8	36.6	8.8	91.8	**3.7**
D4	696.3	255.2	21.9	335.6	**14.1**
D5	262.2	156.0	26.0	186.4	**6.4**
D6_1	490.6	256.1	33.0	275.7	**18.4**
D6_2	243.3	105.5	15.7	137.5	**8.4**
D6_3	460.2	72.8	10.6	141.0	**4.2**
D6_4	359.9	50.0	8.0	96.9	**3.1**
D7	1088.2	886.7	145.4	595.6	**40.0**

The fastest in each row is highlighted in bold (see Table 2. for datasets information.).

Table 12. Peak memory usage by error correction and assemblers (either Velvet or SOAPdenovo) (Gb). [E.C. Lim et al., 2014]

Data	Trowel	Coral	Musket	SOAPec	Quake	Assembler
D1	**3.5****	12.4	**0.5***	13.2	7.2	4.0
D2	**8.0****	26.4	**1.6***	13.1	8.3	18.0
D3	**4.5****	14.9	**1.4***	13.2	6.7	18.3
D2+D3	**8.0****	26.4	**1.6***	13.2	8.3	26.1
D4	**10.4****	28.6	**1.8***	13.2	12.1	19.4
D5	**5.7****	24.2	**2.3***	13.2	9.0	33.5
D6_1	19.3	39.6	**3.4***	**13.2****	33.7	27.9
D6_2	15.4	24.5	**2.5***	13.2	**8.9****	20.4
D6_3	**5.4****	22.4	**2.1***	13.2	9.0	15.8
D6_4	**4.1****	18.3	**1.4***	13.2	8.1	10.8
D7	36.8	132.5	**15.4****	**13.2***	42.4	119.1

*: the best in each row. **: the second best in each row.

Applying an error correction module can reduce the memory consumption while assembling a genome. The error corrections yield a more compact graph representation given dataset by removing

false nodes, leading to less memory requirements (Table 12). Except for Coral, all the error correction modules are advantageous. Though Trowel is fast, Musket is the best with respect to the memory efficiency.

2.4.5 Sum-of-Rank table

Table 13. Sum-of-rank of performance metrics. [E.C. Lim et al., 2014]

Tool	Read Accuracy	Base Accuracy	Genome Assembly (overall)	Assembly (high coverage)	Assembly (low coverage)	Runtime	Memory usage
Uncorr.	-	-	6	6	4	-	-
Trowel	**1**	**1**	**1**	**1**	2	**1**	2
Coral	2	4	2	4	**1**	3	6
Musket	3	5	4	4	2	2	**1**
SOAPec	4	3	4	3	4	4	4
Quake	5	2	2	2	4	5	3
Assembler	-	-	-	-	-	-	5

The best in each column is highlighted in bold.

The sum-of-rank table summarizes all the evaluation results and demonstrates the overall rank of algorithms (Table 13). The sum of rank for each category is independently calculated over all the available datasets and parameters. The rank of read accuracy and assemblies for high coverage datasets is mostly consistent each other except for Quake. The rank of base accuracy is almost consistent with the one for low coverage datasets. It is difficult to generalize the correlation between metrics due to insufficient number of samples. However, the table demonstrates a relative rank for each evaluation category.

Trowel has maintained a high read accuracy for all the experiments, thus quantitative analyses based on the read alignments can make use of Trowel as a preprocessing stage. SOAPec and Quake sometimes have achieved the best base accuracy, but both tools are of poor performance in read accuracy. The loss of alignable reads due to error corrections or aggressive read trimming cannot be undone. Coral has a reasonable read accuracy but the base accuracy is not comparable. This inconsistency implies that some artificial SNPs can be introduced by Coral. For read accuracy, Trowel is the best and second best in 49 and 21 out of 80 experiments, respectively.

Since *A. thaliana* reads are very close to the reference genome, fewer number of mis-assemblies and mismatches indicates a better accuracy. The assemblies from Coral corrected *A. thaliana* reads contains more number of mis-assemblies and mismatches than the median values of the others. In addition, the assemblies derived from Musket corrected datasets often have more number of mismatches. Trowel's metrics has posed around the median. Coral is only beneficial for very low coverage datasets

such as D6 (<10x) in terms of assembly performance. Trowel is particularly advantageous for high coverage datasets. The read trimming may improve the quality of genome assemblies, but it is not a dominant factor according to the ranks. In the sum-of-rank table of all the genome assemblies, the rank is changed only in 6 out of 16 experiments, mostly between the first and second entries. In turn, the noticeable changes due to the read trimming only occur in 3 out of 16 cases.

Based on the experiments for the error corrections at the repeat boundary, most tools have converted error-containing reads to unrelated error-free reads. These conversions increases the number of true positives, leading to a better accuracy, but in reality, these are the losses of true sequence information or introducing locally correct ends in each paired-end read. Such invalid conversions can be detected only by observing the number of mismatches and mis-assemblies in the scaffolds.

2.5 Conclusion and discussion

Without deeply understanding about the algorithms of Trowel, one may argue that Trowel resembles Musket [Y. Liu et al., 2013] or Quake [D.R. Kelley et al., 2010]. All *k*-mer spectrum based algorithms preserve *k*-mer information in various forms such as hash map, bloom filter, suffix tree, and so on. Trowel, Musket, and Quake utilize the hash map for this purpose. The common information is the multiplicity of *k*-mers except for Trowel. Trowel uniquely uses a quality threshold as the major criterion to filter out noisy *k*-mers. Musket and Quake use every single-end *k*-mer to associate the information for the whole correction procedures. However, Trowel changes the composition of *k*-mers in order to increase the sensitivity. As explained in the erroneous-base-next-to-repeats problem, a brick composition of (*k1* + gap + *k2*) is much more sensitive than the common *k*-mer composition (*k* + gap) given parameter *k*. Musket and Quake utilize the common composition.

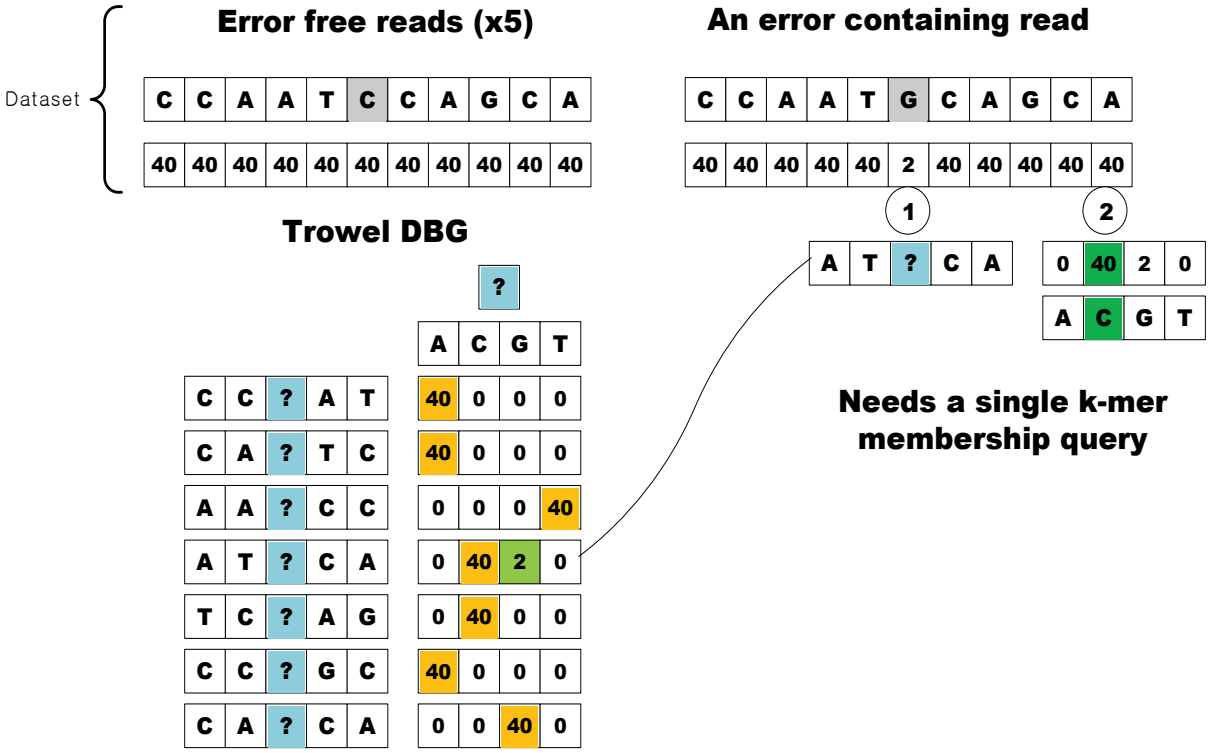

Figure 12. A simplified flow of Trowel's DBG. ① DBG algorithm encountered a *k*-mer having a lower quality value at the gap position than the quality threshold (40). ② Replace the suspicious base with the unique candidate base (C) having the highest quality value. [E.C. Lim et al., 2014]

Musket and Trowel apply multi-stage algorithms in order to improve the accuracy. The DBG is in parallel to the Two-Sided Correction (TSC) of Musket and similarly the SBE to the One-Sided Correction (OSC). One may claim that Trowel has copied the Musket's procedure. However, multi-stage procedures of Musket is merely a specialization of SA approaches. For the first stage, both algorithms detect untrusted bases by performing *k*-mer membership queries, which is the standard for *k*-mer spectrum based algorithms. The DBG is finished in a single pass, leading to the time complexity $\mathcal{O}(k+\sigma)$ at the error position (Fig. 12). Conceptually, both algorithms correct a base surrounded by two *k*-mers. In the next stage, the correction is performed on an untrusted base adjacent to a *k*-mer. The time complexity of both algorithms is mainly calculated in terms of the *k*-mer composition. The DBG is based on gapped *k*-mers, whereas the TSC uses overlapping *k*-mers. When encountering a suspicious base, the DBG quickly looks up bases in the base-quality statistic of a single gapped *k*-mer while the TSC has to look up five (or more) independent *k*-mers.

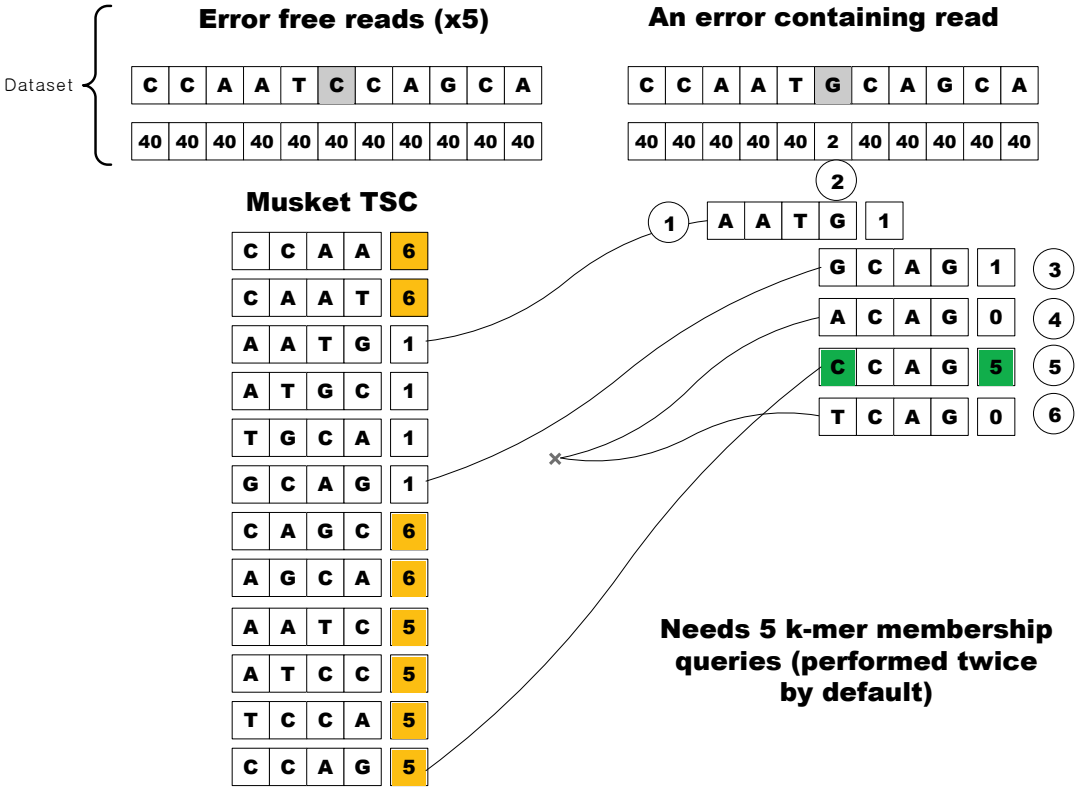

Figure 13. A simplified flow of Musket's TSC. ① TSC algorithm encountered a *k*-mer having a lower frequency than the frequency cutoff (3). ② Current *k*-mer becomes the left-most *k*-mer whose last base is untrusted. ③ Check the frequency of the right-most *k*-mer. ④ Generate an alternative right-most *k*-mer by a replacement for the untrusted base with 'A', leading to a hash miss. ⑤ Generate an alternative right-most *k*-mer by a replacement for the untrusted base with 'C'. Check the frequency of the right most *k*-mer. ⑥ Generate an alternative right-most *k*-mer by a replacement for the untrusted base with 'T', leading to a hash miss. ⑦ Replace the untrusted base with the unique candidate base (C) having the highest frequency. [E.C. Lim et al., 2014]

By considering left- and right-most *k*-mers, the TSC needs $\mathcal{O}(2k(\sigma+1))$ time. The figure 13 only illustrates one of the simplest cases, where the erroneous base has to be corrected to a unique answer. To calculate the worst case scenario, one needs to account for a situation where both left- and right-most *k*-mers should be compared simultaneously due to the conflict in possible bases. Not surprisingly, the SBE has the same time complexity with the DBG, $\mathcal{O}(k+\sigma)$. The OSC is conceptually simpler than the TSC but it is practically more complicated due to the additional stages, look-ahead validation, and voting-based refinement. These stages require more look-ups for overlapping *k*-mers. The time complexity of the OSC is claimed to be $\mathcal{O}$(k) by the author, which is incorrect. Due to two extra stages, the OSC has $\mathcal{O}(k(2+n+\sigma))$ time complexity where n is the number of additional neighboring *k*-mers.

SHREC [J. Schröder et al., 2009] exploits a suffix tree so that twice applications of the path-label function can be regarded as a simulation of the DBG *k*-mer composition (*k1*-gap-*k2*). However, such separate applications of path-label function do not automatically confer a linked context and its associated value, i.e., the multiplicity of a gapped *k*-mer. By applying a single path-label function, one can

correctly duplicate the DBG-composition. However, the suffix tree of SHREC is not designed to simulate the gapped *k*-mers. Due to the memory reduction mechanism, SHREC randomly loses valid paths.

2.5.1 Discussion

The performance of an error correction module is difficult to predict, whereby the standard metrics fluctuate among measurements even for the same dataset. The parameter *k* is one of the most critical factors determining the accuracy of *k*-mer spectrum based error corrections. In Yang's survey [X. Yang et al., 2012], the sensitivity of Coral varies from 0.03 to 58.3 and of Quake rises and falls in a range (33.2…81.7) among experiments. Even though the parameter *k* is fixed across the evaluations, the gain has a significant deviation, i.e., (0.3…57.9) for Coral, (-12.6…78.4) for Quake. To determine a reasonable parameter *k*, one may apply multiple *k* values to the same dataset, which is practically irrational. For the sake of the evaluation, however, it is recommended to perform a plenty of experiments in order to avoid cherry-picking. I have calculated both read accuracy and base accuracy according to the standard evaluation while varying *k*. For all the experiments, the changes in specificity is indistinct. Hence, one should pay more attention to sensitivity. Aside from parameter *k*, the read length, uneven coverage, and the quality of sequencing, contaminations may affects the results.

The "optimal estimated *k*" denoting the saturation rate assumes a uniform distribution of *k*-mers over a genome. The "optimal estimated *k*" value cannot reflect the globally changing dynamics of genome compositions. Hence, as one selects a fixed *k*, the accuracy of error corrections is immediately determined. Despite the limitation, the "optimal estimated *k*" practically acceptable according to the experiments. The performance started dropping around the "optimal estimated *k*" for most cases. Ideally, the value of parameter *k* should be flexibly changed depending on the complexity of sequences, and the *k*-mer frequency distribution for each error correction attempt. The optimal performance has obtained with equal to or larger than the "optimal estimated *k*", suggesting that the "optimal estimate *k*" should be served as the minimum *k* for the dataset whose size of genome is known. To avoid the parameter selection, in Trowel 2, I have started to support the variable parameter *k* thanks to the FM-index.

A recent *k*-mer spectrum based error correction tool, BLESS [Y. Heo et al., 2014] applies Error Correction Evaluation Toolkit (ECET) scripts from [X. Yang et al., 2012] on the dataset (*E. coli*, SRR001665) used in this study. I notified the differences between the results of BLESS and Trowel for this specific dataset, and manually inspected the source code of ECET. Though the ECET has been widely applied and still cited by many error correction module designers, it is incorrectly implemented so that the results are not reliable. More specifically, the published results relying on the ECET scripts are obviously incorrect. For instance, it counts arbitrary number of true negatives and reports them as true positives. Moreover, the ECET calculates statistics only from a small fraction of total bases (0.02% to 20.2% in the evaluations). It loses considerable amount of true base information while converting the Sequence Alignment/Map (SAM) files [H. Li, 2009b] to Target Error Format (TEF). In a correct implementation reflecting the definitions in Yang's survey, the sum of TNs, TPs, FPs, FNs, and trimmed bases have to be equal to the number of bases in the uncorrected reads. I calculated all the metrics exactly in that way.

The other overlooked factor that affects the measurement is the sequence alignment mode. To detect the long-range context changes (LRCC), the alignment should be performed in a paired-end mode. For instance, BLESS authors combined paired-end reads into single-end ones in order to simplify the evaluation procedure. With a single-end mode alignment, there is of a higher chance to obtain a locally correct alignment after the error correction. Nevertheless, a successful single-end alignment does not guarantee a collateral paired-end hit, accidentally hiding LRCC. More specifically, if an error correc-

tion tool mis-corrected a read in a pair, such paired-end reads would have a slightly higher chance of being unalignable in the paired-end mode. However, the other end still can be aligned in a single-end mode. I applied the same edit distance with BLESS paper, which is 7, for the *E. coli* dataset in order to compare the results. The gain of Musket is 0.814 in paired-end mode, but BLESS reported it as 0.926. This decrease in gain explains how much the mode of read alignments influences the evaluation results. Interestingly, the change in Quake is negligible that I obtained 0.828 gain, which is comparable to 0.837 in BLESS paper.

The edit distance, ED, for read alignments is the last source of fluctuation among evaluation results. I observed the changes in standard measurements on the *E. coli* dataset while changing ED from 1 to 10. Two different versions of BWA (0.7.3 and 0.6.2) are tested with exactly same parameters though the differences are insignificant. They are nearly equal in four decimal places. The edit distance values greater than 10 do not convey any meaningful message, hence they are omitted. The maximum gain values for Quake and Musket are 0.978 with ED=1, and 0.841 with ED=2, respectively.

2.5.2 Conclusion

Trowel is the first *k*-mer spectrum based algorithm applying only a quality threshold in order to classify *solids* and *weaks*. The asymmetric *k*-mer composition is a novel scheme increasing the sensitivity in long-range contexts. I have demonstrated the high accuracy of Trowel in various experiments with different parameter settings. Trowel reliably yielded results in competitive runtime and reasonable memory consumption. I also reported the inconsistency between the ECET software implementation and the definitions. The problems identified during the assessment of the ECET has been addressed in detail. Trowel is typically effective if the downstream analyses are related to the quantization of reads due to its outstanding read accuracy. The quality-awareness makes Trowel to be non-destructive in read trimming and be tolerant against the contaminations (see the result of genome assemblies for D3 [E.C. Lim et al., 2014]).

I have addressed the sequencing error correction problem for the genomic datasets and provided an accurate and efficient solution.

2.6 References

B. Ewing, L. Hillier, M.C. Wendl, and P. Green (1998) Base-calling of automated sequencer traces using phred. I. Accuracy assessment, Genome Res., 8(3):175-85.
S. Gnerre et al. (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data, Proc. Natl. Acad. Sci., 108:1513-1518.
A. Gurevich, V. Saveliev, N. Vyahhi, and G. Tesler (2013) QUAST: quality assessment tool for genome assemblies, Bioinformatics, 29:1072-1075.
Y. Heo, X.L. Wu, D. Chen, J. Ma, and W.M. Hwu (2014) BLESS: bloom filter-based error correction solution for high-throughput sequencing reads, Bioinformatics, 30(10):1354-62.
L. Ilie, F. Fazayeli, and S. Ilie (2010) HiTEC: accurate error correction in high-throughput sequencing data, Bioinformatics, 27:295-302.
W.C. Kao, A.H. Chan, Y.S. Song, (2011) ECHO: A reference-free short-read error correction algorithm, Genome Res., 21:1181-1192.
D.R. Kelley, M.C. Schatz and S.L. Salzberg (2010) Quake: quality-aware detection and correction of sequencing errors, Genome Biol., 11:R116.

S. Kurtz, A. Phillippy, A.L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S.L. Salzberg (2004) Versatile and open software for comparing large genomes, Genome Biol., 5(2):R12.
H. Li, and R. Durbin (2009a) Fast and accurate short read alignment with Burrows-Wheeler Transform, Bioinformatics, 25:1754-1760.
H. Li et al. (2009b) The Sequence Alignment/Map format and SAMtools, Bioinformatics, 25:2078-2079.
E.C. Lim, J. Müller, J. Hagmann, S.R. Henz, S.T. Kim, and D. Weigel (2014) Trowel: a fast and accurate error correction module for Illumina sequencing reads, Bioinformatics, 30(22):3264-5.
Y. Liu, J. Schroeder, and B. Schmidt (2013) Musket: a multistage k-mer spectrum based error corrector for Illumina sequence data, Bioinformatics, 29:308-315.
R. Luo et al. (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler, GigaScience, 1:18.
G. Marçais, and C. Kingsford (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers, Bioinformatics, 27:764-770.
S. Ossowski, et al. (2010) The rate and molecular spectrum of spontaneous mutations in Arabidopsis thaliana. Science, 327:92-94.
L.W. Parfrey, D.J.G. Lahr, and L.A. Katz (2008) The Dynamic Nature of Eukaryotic Genomes, Mol. Biol. Evol., 25:787-794.
P.A. Pevzner, H. Tang, and M.S. Waterman (2001) An Eulerian path approach to DNA fragment assembly, Proc. Natl. Acad. Sci., 98, 9748-9753.
M. Roberts, W. Hayes, B.R. Hunt, S.M. Mount and J.A. Yorke (2004) Reducing storage requirements for biological sequence comparison, Bioinformatics, 20(18):3363-3369.
L. Salmela (2010) Correction of sequencing errors in a mixed set of reads, Bioinformatics, 26:1284-1290.
L. Salmela, and J. Schröder (2011) Correcting errors in short reads by multiple alignments, Bioinformatics, 27:1455-1461.
J. Schröder, H. Schröder, S. J. Puglisi, R. Sinha, and B. Schmidt (2009) SHREC: a short-read error correction method, Bioinformatics, 25:2157-2163.
L. Song, L. Florea, and B. Langmead (2014) Lighter: fast and memory-efficient sequencing error correction without counting. Genome Biol., 15(11):509.
The Arabidopsis Genome Initiative (AGI) (2000) Analysis of the genome sequence of the flowering plant Arabidopsis thaliana, Nature, 408:796-815.
X. Yang, S.P. Chockalingam, and S. Aluru (2012) A survey of error-correction methods for next-generation sequencing, Brief. Bioinform., 14:56-66.
X. Yang, K.S. Dorman, and S. Aluru (2010) Reptile: representative tiling for short read error correction, Bioinformatics, 26:2526-2533.
D.R. Zerbino, and E. Birney (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs, Genome Res., 18, 821-829.

Chapter 3

3. Multiple whole genome alignment

3.1 Introduction

In a biological population, genetic differences among individuals can be used to identify causal genes for certain genetic traits. The sequence alignments can be regarded as raw material to identify differences between individuals. Though short read aligners have been widely accepted, they are not naturally scalable so that they can be easily applied to population-wide comparisons. A short-read aligner utilizes a linear reference genome sequence as a proxy to compare two individuals. Hence, for instance, one should manually integrate multiple alignments to compare many individuals. Even worse, a single reference genome may not be able to represent all the genetic dynamics in the population, i.e., due to missing sequences, and rare variants. To overcome the limitation of the linear reference genome, an alternative representation is proposed in this study, the population index. Moreover, a short-read alignment reflects only few local differences, meaning that changes in a small fraction of the entire genome are identified. Thus, large scale variants could often not be resolved.

In turn, a new alignment strategy to overcome the limitation of short read aligners should have the following features: (1) an all-against-all alignment or alternative method on top of a population-index like data structure, (2) supports for long sequence alignment, (3) easily interpretable symbols to represent genetic differences, and (4) computational efficiency. I briefly describe previous works regarding multiple sequence alignment and whole genome alignment, and then propose a novel alignment algorithm.

3.1.1 Multiple sequence alignment

A Multiple Sequence Alignment (MSA) is a process of identifying similar sequences in three or more sequences. A residue is a basic building block or monomer of a protein, an amino acid, or a nucleic acid, a nucleotide. An MSA can be performed either on DNA or protein sequences to determine a series of monomers representing a similar sequence. Among several applications, one of the primary applications of MSAs is to derive a model inferring a phylogenetic tree and studying the evolutionary process behind it. The number of complete genomes to which MSAs can be applied, has greatly increased due to advances in Next Generation Sequencing (NGS) technology.

A common ancestor of different species is modeled as a solution of a minimal mutation tree, thus finding the solution of the tree in a deductive process represents an MSA in given species [D. Sankoff, 1975]. Major algorithms for MSA have been introduced in 1980s. For instance, [H.M. Martinez, 1983] explained a method to find repeats in two or more similar sequences. An algorithm, qalign, used a common substring to order the sequences within a given distance, which is a metric measuring the sequence dis-similarity between two alignments. The algorithms in this generation were designed to perform local alignments for detecting similar patterns [S.B. Needleman, and C.D. Wunsch, 1970, M.S. Waterman, 1984]. A global-alignment algorithm, MULTAN, was subsequently proposed by W. Bains [W. Bains, 1986]. At the same time, an improved version of qalign, malign, was developed; it is capable of aligning two sequences or more [E. Sobel, and H.M. Martinez, 1986]. M. S. Waterman suggested an algorithm that maps sequences against a consensus word by maximizing the matching score [M.S. Waterman, 1986]. The Needleman-Wunsch algorithm [S.B. Needleman, and C.D. Wunsch, 1970] is then applied to perform an MSA [D.F. Feng, and R.F. Doolittle, 1987]. The sequence deference matrices derived from MSAs can then be used to build a phylogenetic tree. This idea of a guide

tree from iterative alignments, in fact, originated from [G.W. Moore et al., 1973] with a distinction that the gap information for each iteration is maintained. CLUSTAL, which directly applied the idea of Feng-Doolittle algorithm, was subsequently developed and has been widely used [D.G. Higgins, and P.M. Sharp, 1988].

Some "progressive" methods are influenced by the concept of consistency, which is a criterion to measure how much a precomputed global alignment and an ongoing alignment correspond to each other. The algorithm measures how many intermediate alignments support the residues in consideration [O. Gotoh, 1990]. A practical application of the consistency is initiated by T-Coffee [C. Notredame et al., 2000], re-implemented by [T. Rausch et al., 2008]. A critical problem of the progressive alignments is the property of error propagation due to its greedy nature. The errors occurred at early stage would propagate through succeeding steps. The problem becomes serious without any amendments that improve the accuracy of alignments. In ClustalW, phylogenetic distances are used as weights to reduce early error propagations [J.D. Thompson et al., 1994]. T-Coffee builds a primary alignment library where global alignments from ClustalW and local alignments from FASTA package [W.R. Pearson, and D.J. Lipman, 1988] are combined. A weight indicating the percent of identity is assigned to each pair of alignments and is applied to support a dynamic programming implementation for progressive alignments. Iterative alignments were introduced over progressive algorithms such that new alignments are added to the final MSAs following a guide tree. The initial sequences are continually re-aligned at each iteration. This type of refinement is implemented in PRRP [O. Gotoh, 1996]. The MSA algorithm MUSCLE is much faster than CLUSTAL with a high accuracy thanks to the fact that a guide tree is estimated by the *k*-mer distance and by the Kimura protein distance[2] [R.C. Edgar, 2004]. Yet another algorithm, CHAOS, rapidly calculates a chain of sequence similarities from seeds, which are genomic sequence pairs. In this approach, local alignments are used as anchors for DIALIGN, which performs the actual multiple alignments [M. Brudno et al., 2003].

The sensitivity and precision of progressive alignments can be further improved by introducing seed-and-extend matches. Seeds are divided into two types, either exact or inexact ones. The inexact ones are more tolerant to sequencing errors or mutations, increasing the sensitivity. Multiple seeds can be overlapped in a consecutive stretch of sequences, but they can be also sparsely selected [B. Ma et al., 2002]. The sparse seed approach not only reduces the space requirements, but also increases the sensitivity. With sparse seeds, one has a better chance of obtaining accurate alignments, even if sequencing errors or mutations exist in a sequence. This is mainly because, for any strings, the first *n* words are highly likely to be matched with a sparse seed than a continuous overlapping seed [U. Keich et al., 2004]. An error in overlapping seeds is counted only once, whereas the error is counted several times with overlapping seeds, in turn, degrading the alignment accuracy.

The early MSA algorithms often represented alignments as 2-dimensional (2D) matrices, hence they were limited in their ability to characterize non-linearity in MSAs derived from multiple genomes,

---

[2] Kimura's protein distance measures the distance between two alignments based on observed amino acid substitutions and actual substitutions. Exact matches are concerned but gaps are ignored. [M. Kimura, 1983]

S = exact matches / positions scored

D = 1 – S

$$Dist = -\ln(1 - D - 0.2D^2)$$

i.e. repeats, and transpositions. Complex mutational events such as rearrangements or duplications destroy the directionality of alignments and the complexity originating from multiple discrepancies cannot be stated with naive 2D matrices. [C. Lee et al., 2002] suggested a partial order graph (POG), which is a directed acyclic graph (DAG), to solve this problem. Due to the limitations in acyclic graphs, duplications and inversions are not incorporated. [Y. Zhang, and M.S. Waterman, 2003] applied therefore a de Bruijn graph to represent MSAs. The de Bruijn graph requires a stringent condition such that exact number of matches must be prepared in order to select a set of connected components. This strictness can be alleviated if approximate matches in each local alignment are allowed. The problems of local alignment methods are incorrect positions for duplications and "crosses" occurring among different genome segments. These problems can be solved by the A-Bruijn graph, which assigns a weight to each edge in order to denote duplications or inversions [B. Raphael et al., 2004]. This data structure is originally designed to detect repeat sequences and to assemble genomes as an alternative to the de Bruijn graph [P.A. Pevzner et al., 2004]. A recently re-discovered data structure, Cactus graph, where common sub-structures among closely related genomes are preserved in a hierarchical structure, can characterize both types of events [B. Paten et al., 2011].

3.1.2 Whole genome alignment

The distinction between an MSA and a whole genome alignment (WGA) may not be so obvious because both algorithms can perform all-against-all mappings given multiple sequences. A WGA has begun with a different assumption from than an MSA or paired-end mapping (PEM); specifically the length of genomes to be compared are similar one another. Unlike MSA methods, a WGA algorithm requires at least two genome sequences. Though theoretically MSA algorithms should be able to deal with more than three genomes, established algorithms are highly limited in terms of whole genome scale alignments. The WGA algorithms are intended to identify large mutation events in given complete genome sequences rather than to observe such signals in short reads. Though a primary purpose of WGAs is to find similar sequences on the genome scale, the genomes may be neither complete nor closely related. Since the quality and contiguity of NGS reads have gradually improved, WGA methods should at some point be able to replace short read aligners (refer to section 0.2.3).

One of the most popular algorithms, MUMmer, has a long history of software development. The algorithm is based on the suffix tree, which ensures a high time-performance due to its fast string queries [A.L. Delcher et al., 1999]. As the name implies, a maximal unique match (MUM) is a primitive unit for further alignments, denoting the longest unique sequence in "two" genomes. A gap-filling procedure is essential to find several types of genetic variations. The second version, MUMmer2, reduced the memory requirements by two means: changing the node representation in storage, and "streaming" such that a suffix tree is built only by a reference genome while the query genome is superimposed along the nodes in the suffix tree for the alignments. The use of a suffix link during the tree traversal provides a uni-directional maximal matching for fragmented contigs derived from failures in genome assemblies. Even if a contig was not correctly elongated, MUMmer2 can potentially find an MUM longer than the length of the contig. The partial matches of contigs are clustered in an alignment matrix and MUMmer2 can find the longest connected component along the diagonal representing the longest matches [A.L. Delcher et al., 2002]. The last version, MUMmer3, provides an even better performance, an improvement in the detection of duplications, and approximate matches for seeds [S. Kurtz et al., 2004].

A WGA algorithm can identify all-against-all pairwise matches without any distinction between exonic or intergenic regions. WGAs can be represented by matrices or a chain of collinear blocks. Collinearity, which is adopted from geometry, is a property of a set of blocks lying on a straight line. It is maintained when genomic sequences are not broken by any rearrangement events. Thus, small mutations such as single nucleotide polymorphisms (SNPs) do not change the collinearity. WGAs yield a chain of many collinear blocks, but the number of chains is far less than the quantity of entries in

pairwise alignments. Definitely, the number of genomes in comparison is proportional to the number of blocks while the length of each block becomes shorter. However, if the genomes are closely related, then the number of genomes would not be a dominant factor disturbing collinearity, but the number of rearrangement events. The individual genomes of a single species share a huge amount of common sequences (after all, the definition of a species requires shared sequences). Hence, the length of these common sequences in a population of a species would never become as small as "1", even with a very high number of genomes.

An unsophisticated way of performing WGAs in a population with hundreds or thousands individuals is to merge individual WGA results obtained by iterative executions of MUMmer-like algorithms. This strategy has serious limitation because of its high computational loads. When only a handful of genomes are involved in WGAs, the time complexity of alignments is not formidable. As the number of genomes in the index increases, the growth in time and space complexity becomes enormous due to the quadratic increase of pair-wise combinations that have to be analyzed. In addition, a robust scheme should be defined to obtain the intersection of all the WGAs, which necessitates further computation.

All known WGA solutions are incomplete, although a considerable number of algorithms have been suggested. First, optimization has been applied for genome length, rather than the number of genomes in a population. Only a minority of WGA algorithms can practically align hundreds of genomes simultaneously, and very few can handle thousands or more of them. Second, the evaluation of multiple WGA should be carefully considered. Alignathon claimed that it provides a benchmark platform for multiple WGAs [D. Earl et al., 2014]. However, the accuracy assessment itself is a difficult task. Heuristic algorithms differ in performance depending on the parameters. Because optimal parameters are often unknown, a huge parametric space confounds the best use of algorithms. Still, it is usually true that, for closely related genomes, most multi-WGA algorithms can find more alignments and achieve higher accuracy than for distantly related genomes. Lastly, little attention has been paid to the detection of small-scale genetic mutations on top of multi-WGAs. Most algorithms merely yield alignment results without easily interpretable information, i.e., the positions of differences for each entry with respect to the reference genome, the type of genetic mutations, and the length of each component in an alignment. This fact complicates the application of WGAs to variant calling problems. It is known that the detection of duplication events based on WGAs is reportedly typically difficult, thus a new algorithm is urgently needed.

I hereby introduce a new multiple whole genome aligner (MWGA), Kairos, which can not only handle a wide range of inputs, from long reads to genome assemblies and complete genomes, but also a large number of each of these. A distinct innovation is an inverse alignment strategy that maps a single reference genome against multiple genomes in a population rather than an all-against-all comparisons. The inverse mapping is performed in constant amortized time proportional to the length of the reference genome. The reference can either be a conventional reference genome, or a complete genome sequence of a new individual, or a few DNA fragments of a whole genome sequence. The number of genomes may influences the performance due to the increases in physical input-output (I/O) time but alignments themselves can be done in a linear fashion thanks to efficient string operations, and counting functions of population indices. Because of its novel approach, there is no straightforward comparison with existing tools (see section 3.3).

3.2 Method

Kairos expects that a population index contains a large number of genomes for a single species or a cell population of an organism, one example being the 1001 genome dataset of *A. thaliana*, which has over 1100 genomes. The distance of two genomes can be approximately calculated by applying a locality sensitive hashing where a genome sequence is mapped to a set of buckets. A bucket stores elements of high similarity due to collisions. Diverse distant metrics can be used to draw biologically meaningful interpretation. For instance, given two genomes A, and B, a Jaccard distance J can be calculated as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (3.1)$$

Each genome can be represented by smaller number of minimizers [M. Roberts et al., 2004] rather than by long *k*-mers. Thus, the minimizers allow for space-economic measurements of distances among genomes. The number of MSA entries can be reduced if one calculates the distances to choose genomes being added to a population index. When two genomes are closely related and their distance is short, the overall length of alignments increases, leading to a higher sensitivity and precision for detecting SNPs and SVs. Moreover, with closely related genomes a higher compression ratio can be achieved since identical or highly similar sequences are more common than with distantly related genomes.

A noteworthy innovation of Kairos is an inversion of alignment direction such that a reference genome sequence is the proverbial needle, while the other genomes in a population serve as the haystack. When a reference genome sequence is unknown, the most complete genome assembly from the population could be used instead. An MWGA needs a computational architecture to efficiently access homologous sequences in many individuals. A population index conceptually provides a colored graph against which the reference genome sequence aligns. In contrast to multi-WGAs, local or global alignments for short reads are performed against a single reference genome. When the read length is short, the number of breakpoints for large genetic mutations can be very few or none in a single alignment entry. Again, an alignment for a long sequence could identify many breakpoints, thus, in turn, a split-read mapping is more suitable. Millions of short-read alignments are replaced by a number of split-read mappings given the length of the genome.

A *collinear block* represents a homologous DNA segment shared by all the individuals and occurring at least at a threshold value, τ, such that $\tau \leq \mathrm{Occ}_g(s)$ (see next page). The collinear block follows the coordinate system in a reference genome. The initial pattern is used as a seed for each individual genome and an extension procedure is followed to determine individual breakpoint locations (both the start and the end) where a prediction of a SV event is made. A collinear block contains τ-number *split blocks*, and *broken blocks* in combination. Both *split* and *broken block* are defined in the coordinate system for each individual genome. A *split block* denotes an extended homologous DNA segment while a *broken block* characterizes an alignment block, which contains a partially heterogeneous segment at the end of the sequence. An *empty block* is a conceptual definition to describe entirely heterogeneous sequence with respect to the reference genome. Since the initial alignment procedure only captures the exact matching sequences at a certain genomic location, very different sequences cannot be identified.

A *breakpoint* divides a long sequence into blocks of homologous sequences with point mutations or structural variants. A *breakpoint* can be detected while forward-tracking a certain pattern *s* on an FM-index by observing the interval of suffix indices. However, *s* can be found more or less than ex-

pected number of genomes due to the repetitive nature of genomes and deletion-like mutational events. A forward-tracking is performed from the first symbol to the last one in the reference sequence to detect initial collinear blocks. One can detect genetic mutations by checking *breakpoint* locations and mapping orientation in adjacent *collinear blocks*. If a large event is detected, the locations of a *broken block* are restated according to the reference coordinates, in order to provide stable locations.

Let s_r be a unique pattern that occurs at least once in the reference genome while occurring repeatedly in all the other individuals, s_i a pattern that does not occur in the reference genome but in the others, and M the total number of genomes in the index. If a reference genome held all heterogeneous paths and mutations, the definition for variants could be greatly simplified since one can discard the concept of s_i. For ease of explanation, the assumption is made that the reference genome is perfect in the regions of interest. With this assumption, the concept of s_i is unnecessary, thus s always denotes s_r. A function $Occ_g(s)$ returns the number of "genomes" having pattern s. Assuming that a pattern "ACGTTTAA…" occurs three times in a single genome but not in any of the other genomes, $Occ(s)$ is three and $Occ_g(s)$ is one. Specific events (e.g., intragenome duplications, as in this example) can be identified by examining different criteria.

Unlike conventional paired-end mapping (PEM) methods, which are performed against a single linear reference genome, even exact pattern matching is non-trivial in multi-WGAs. First, a genome contains usually repetitive sequences, meaning that there can be multiple hits for a pattern. Based on neighboring alignments, such uncertainty can be reduced because the positional information can be tracked with a population index. Second, the locations of a match in each individual are likely to differ because of diverse mutations affecting each individual. For detecting large-scale events, closely located collinear blocks should be examined with respect to matching orientations, and locations of both a reference and test genome. Third, some split blocks cannot be reconstructed even with sequence tracking procedures. For example, a large deletion or an inversion event destroys the collinearity, leading to discontinuous positional information.

Given a single organism, if a pattern s is not found in all the other genomes, the pattern may be involved in a deletion event or the genome region of s may not be sampled. It could be due to errors or biases in genome assembly algorithms or chromosome walking while creating the reference genome. Such events can be detected by observing the equation below and the adjacent collinear blocks should be carefully examined.

$$Occ_g(\mathrm{s}) = 0 \quad (3.2)$$

The basic assumption of the alignment for a collinear block is that s is uniquely identifiable among all the other genomic regions, leading to $Occ_g(s) = \mathrm{M}$. Despite causing a loss in the accuracy, Kairos utilizes a much faster genome counting method of $Occ(s)$, rather than an exact slow function, $Occ_g(s)$. A population index provides a description for each individual through an inverse sentinel array, φ. Instead of estimating the unique occurrence of s, the exact counting solution is given by a bit array representing all individuals. s of a split block points to an interval of suffix indices in φ, which reflects all the occurrences of s including repeat DNA segments. The bit array ensures that only a single instance of s for each individual is reported. The number of genomes can be obtained by a bit counting operation.

$$Occ_g(s) = \mathrm{count}(\varphi(\mathrm{s})) \quad (3.3)$$

Let k be the number of bits in a word. In practice, the counting operation is a bitwise popcount, where an efficient implementation has $\mathcal{O}(1)$ time complexity for each word. Hence, the space com-

plexity, $\mathcal{O}$(M/k), is directly proportional to the time complexity of a bit array. For the exact solution, the criteria for terminating a forward-tracking are as follows:

$$Occ(s) < 1 \quad (3.4)$$

$$Occ_g(s) < M \quad (3.5)$$

The first criterion has a constant $\mathcal{O}(1)$ time complexity, hence it does not greatly influence the efficiency regardless of M and the length of all individual genome sequences. However, the second one has $\mathcal{O}$(M/k) overhead for every iteration of a tracking process. Assuming the average length of genomes is n, an overall $\mathcal{O}$(Mn/k) time complexity is expected. The changes in suffix indices while performing a tracking cause random memory access, leading to a poor cache locality. |s| is inversely proportional to the probability of occurrence of s given M. In turn, the chance encountering the second termination criteria gets higher as |s| becomes longer. The probability of the pattern uniquely identified in all the genomes can be calculated.

$$P(Occ_g(s) = 1) = 1 - \exp(-d\rho) \quad (3.6)$$

, where d is decay rate, and ρ is the length of s_r. Nonetheless, the true values of d and ρ are unknown before a tracking process is terminated. The true variables of the equation can be obtained for each split block, even though it is highly random depending on the context.

Even with efficient bitwise operations, $Occ_g(s)$ cannot compete with $Occ(s)$ in terms of computational efficiency. I tested $Occ_g(s)$ and $Occ(s)$ on chromosome one of three *A. thaliana* accessions. $Occ(s)$ appears to be 17 times faster than $Occ_g(s)$. Despite the accuracy of $Occ_g(s)$, for large-scale analyses it is not feasible to apply $Occ_g(s)$. This choice of function leads to a new complication, such that some individuals may not have split blocks in a collinear block. Specifically, due to the repetitive nature of genome, if a particular individual contains multiple copies of a DNA segment, $Occ(s)$ may report an exaggerated value of frequency as shown in the example of the pattern "ACGTTTAA…". After crossing over a breakpoint of a duplication event, a sudden drop in $Occ(s)$ value exacerbates this problem.

Those missing sequences increase the false discovery of SV events. To reduce the number of empty blocks, insufficiently-covered sequences in comparison with a given reference genome should be ignored while constructing a population index. Sequences with low coverage are defined as a set of contigs or long reads that do not cover 80% of the reference genome in length when aligned. As previously addressed, one can assess the coverage of long sequences by distance metrics or by coverage estimation using a tool like QUAST [A. Gurevich et al., 2013]. For instance, assuming that a set of contigs only covers the reference genome sequence by 30%. In turn, the highest probability of not capturing alignments for this individual becomes 0.7. In figure 14, the third individual of low coverage delineates the situation where two missing sequences inhibit Kairos from defining split blocks, leading to incomplete collinear blocks.

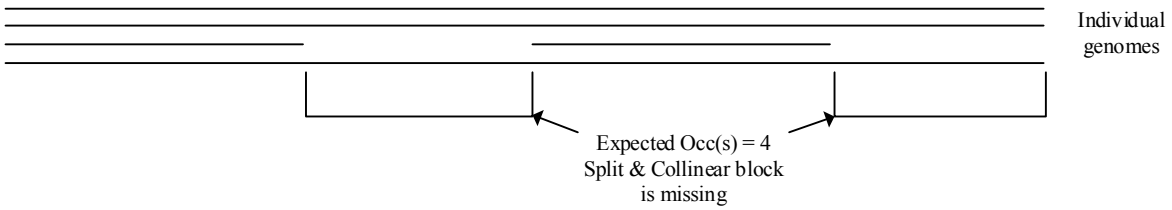

Figure 14. A low-coverage sequence and its effects on split & collinear block identification. Because the number of genomes in the population index is 4, Occ(s) must return 4 for this exemplary region. Since the bases are not sampled from the true genome of the third individual, the number of split and collinear block is increased, requiring more time to calculate.

The algorithm CollectCBs identifies collinear blocks given a population index, a reference genome sequence, and the minimum pattern length *k*. Though the pseudo code does not explicitly demonstrate any parallelization statements, the algorithm supports for evenly distributed load balancing. Note that alignments are generated by forward tracking on an FM-index built in a reverse direction.

Algorithm CollectCBs

Input: Population index PI, Reference sequence G_r, Minimum pattern length k, Count threshold τ, Number of processing blocks J

Output: Corrected Collinear Blocks CB

```
i_prev = 0
M = count(PI.sentinels) IF τ == -1
FOREACH position i in G_r[0..n):
  IF (i_prev - i) < k:
    CONTINUE
  P = R[i…i_prev]
  B = PI.ForwardTrack(P)
  m = count(φ(P))
  n = Occ(P)
  IF (n < 1) OR (m < M):
    IF a distance between the first element of LocalCB and B is less than 1,000
      OR the number of elements in LocalCB is < J:
      LocalCB <= B
    ELSE
      Reconstructs incorrect alignments in LocalCB
      Extends split blocks based on the adjacent collinear blocks in LocalCB
      CB <= LocalCB
      Remove all elements in LocalCB
  i_prev = i
```

CollectCBs algorithm explicitly denotes the inversion of alignment direction. The inverse alignment strategy guarantees an amortized constant time complexity due to the fact that $|G_r|$ is fixed while $|G_i|$ increases, where G_r is a reference genome sequence and G_i is a set of individual genome sequences in a population index. The optimal $|s|$ for each collinear block is highly variable and a breakpoint can be determined with the condition of Occ(s) = 0. When a split block distance d_s is much farther than the collinear distance d_c, for example, d_s > 10,000 and d_c = 1, the split block is highly likely mapped to a repeat DNA segment or involved in a SV event. Thus, a reconstruction procedure replaces the suffix identifiers and pattern of the split block by forward tracking from the previous split block or by backtracking from the next split block. To improve the reliability, the length of adjacent split blocks, $|s_{i-1}|$ and $|s_{i+1}|$, should be compared while determining from which split block the procedure starts. This procedure resembles the molecular biology procedure of PCR such that an initial collinear block represents a set of primers and the forward tracking attaches a new base as the DNA polymerase does during DNA synthesis.

The reconstruction improves the sensitivity and precision of alignments near repeat segments, which has not been realized in conventional MWGA methods. Repeat regions are usually masked to avoid spurious multiple hits. Kairos can detect SNPs and small indels reliably even near a repeat boundary due to the reconstruction procedure and neighboring information. The coordinates in successive split blocks should be numerically increased unless the underlying sequences are involved in SV events, which is ensured by the property of the progressive alignment. A radical change of the position in the split block often indicates a SV event. An exception, which should not be classified as an SV event, is caused by a split block of a very short repeat, where $|s| \leq k$. In this case, the coordinates and the pattern are adjusted and reconstructed.

The results of CollectCBs algorithm are the collinear blocks containing multiple split blocks and broken blocks as shown in the top block diagram in Figure 15. Some split blocks in a collinear block may not be connected due to an inversion, a deletion event, or dense small mutations in a certain genomic region. Broken blocks are the key signals for detecting SV events, thus incomplete sequences in a broken block should be reconstructed in forward-only direction. Allowing for reverse reconstruction in a broken block leads to redundant predictions for a single SV event. When d_s < 50, d_c = 1, and the mapping orientation is preserved, two additional algorithms can be applied. First, a local alignment method can be performed to detect small mutations. Second, a merging procedure can create a long collinear block and reports an enhanced CIGAR-like string [H. Li et al., 2009] as shown in the bottom block in Figure 15. The enhancement consists of representations for SV events, which are explained in section 4.2. Those local differences do not directly contribute to detecting SV events, but the resultant block reduces false breakpoints. The enhanced notation is addressed in Table 14.

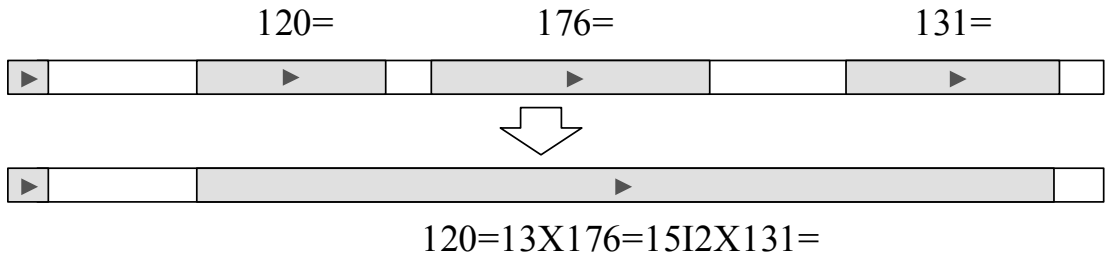

Figure 15. Merging procedure for small mutations on multiple collinear blocks and the representative Compact Idiosyncratic Gapped Alignment Report (CIGAR) strings. Local alignment at each boundary creates a single large collinear block.

Table 14. Enhanced CIGAR-like string. Kairos generates alignments denoting differences between an individual and a reference genome with following symbols alongside positional information.

Type	Notation
Match	=
Mis-match	X
Deletion	D
Insertion	I
Duplication	R
Inversion	V
Intra-chromosomal Insertion	P
Intra-chromosomal Deletion	O
Inter-chromosomal Insertion	F
Inter-chromosomal Deletion	E

MergeAdjacentCBs algorithm performs local alignments between adjacent split blocks and reduces the number of collinear blocks. Though a merged collinear block is conceptually valid, the length of the split block could become too long to analyze. Hence, only adjacent blocks of $d_s < 50$ are merged and the process is not propagated if the length exceeds 1 kb. A high number of concrete collinear blocks may be present in the alignments, but the effective number of collinear blocks would be much lower since they can be conceptually merged. Note that d_s is often 1 bp since SV events are rare but SNPs are not.

Algorithm MergeAdjacentCBs

Input: Collinear Blocks CB, Minimum pattern length k, Number of in-memory collinear block n
Output: Merged Collinear Blocks M

```
While(!CB.empty()):
  A = CB.PopFront()
  B = CB.PopFront()
  DREF = Distance(A, B)
  FOREACH SplitBlock Si, Si+1 in A, B:
    DTEST = Distance(Si, Si+1)
    IF 1 == DREF AND 1 == DTEST:
      MergeSNP(Si, Si+1)
    ELSE IF DREF < DTEST AND DTEST < 1000:
      MergeSmallInsertion(Si-1, Si):
      Invalidate(Si)
    ELSE IF DREF > DTEST AND DTEST > -1000:
      MergeSmallDeletion(Si, Si+1)
      Invalidate(Si)
```

```
    M.PushBack(A)
    CB.PushFront(B)
    IF (Length(M) > n):
      Output(M)
      M.Clear()
Output(M)
```

In summary, Kairos identifies collinear blocks, broken blocks, and breakpoints by inverse alignments. SNPs, small mutations, and structural variants among individuals in a population index can be readily detected on top of those basic signals. A genetic mutation can be predicted by the length of sequence matches surrounding a breakpoint rather than by a *coverage-based probability*. The alignment algorithm tries to overcome confusing repeat regions, which cause multiple hits, by reconstructing split blocks. This is a notable improvement over conventional short read alignment algorithms, which usually mask the repeat region to avoid confusion, and randomly choose one of multiple hits. The alignments are generated in an enhanced CIGAR-like format, thus downstream analyses would be facilitated.

3.3 Evaluation

There is no widely accepted standard method for the evaluation of MWGAs. The assumptions behind for each algorithm vary notably, thus, the resultant file formats are often incompatible. A quantitative assessment poses new complications. First, the definition of a ground truth and a false signal is unclear. It is not guaranteed that identical alignments are always generated for a certain genomic region. Thus, an evaluation scheme should be capable of interpreting overlaps among alignments generated by different algorithms. Second, explicit mapping information such as CIGAR string are not always available, which are essential to evaluate the performance. A multi-WGA for a long sequence may contain multiple elements representing local differences. A performance reviewer, who is not in charge of developing algorithms, should design a module to extract such information. For these reasons, I firstly address known evaluation methods, and suggest a novel scheme for the evaluation.

3.3.1 Brief overview of known methods

A common goal of MWGAs is to find homologous sequences in multiple genomes. It can be applied to several applications such as detection of functional motifs and domains, inference of a phylogenetic tree, primer design for PCR, and prediction of ancestral protein structure. Such diverse applications have introduced a new complexity in interpreting alignments. A simulation-based scheme allows for detection of algorithmic flaws. Even if one precludes biological assumptions on genetic mutations such as point mutations, and large structural variations destroying exon-intron boundaries, the scheme still provides objective measurements. In practice, a simulator usually generates sequences based on a certain evolution model leveraging small polymorphisms to SVs along with ground truth alignments.

Agreements between predicted alignments and the ground truths can be quantified by sum-of-pairs (SP) and true-column (TC) scores [J.D. Thompson et al., 1999]. The SP score is the sum of the number of identical residues in each column. The TC score is defined as the ratio of the number of columns, where all the residues in a column are identical, to the number of total columns. A simulation offers a high freedom of modeling thanks to the use of multiple parameters, which reflects the assumption of the experimenter. On the other hand, the misuse of the parameters can lead to a highly unrealistic experiment invalidating the concept of objective measurements. One should contemplate that the results from a simulation are only estimation to the true accuracy.

The meaning of consistency in the evaluation differs from the one in the alignment context. The idea behind the concept of consistency is that accurate aligners tend to be consistent with a general alignment whereas incorrect aligners do not [T. Lassmann, and E.L.L. Sonnhammer, 2005]. The intra-consistency compares a pair of alignments within a single MSA, if the alignments match each other. The inter-consistency check compares the alignments with ones generated by other tools. A biased selection of tools in an inter-consistency evaluation will of course lead to biased results. For example, a correct alignment generated by a new algorithm may get penalized by shortcoming of other aligners (when they fail to detect a true alignment). The inter-consistency method defines an overlap score as the ratio of the number of matched residues between a pair of alignments relative to the average number of residues in the union of alignments. The expected value of an overlap score reflects the difficulty of an alignment.

The heads-or-tails (HoT) evaluation considers alignments in both directions [G. Landan, and D. Graur, 2007]. The HoT authors claimed that the TC is too stringent and that therefore only the SP score, but not the TC score should be used in an evaluation. The HoT evaluation itself is useful to select a better algorithm or to optimize the parameters, but it cannot be used directly in a performance evaluation [B.G. Hall, 2008]. In fact, the HoT authors do not suggest a scheme for an evaluation, but merely a single metric. With long reads or contigs, the HoT approach loses its functionality since it cannot cope with directionality changes within long sequence alignments.

As mentioned in the Introduction, MSAs can also be applied to proteins. The geometry of two proteins can be similar even if the sequences have largely diverged during evolution [C. Chothia, and A.M. Lesk, 1986]. An evaluation relying on structural similarity may provide further insights on the accuracy measurement, which cannot be captured by sequence-similarity-based schemes alone. HOMESTRAD is a database of alignments based on three-dimensional (3D) protein structure [K. Mizuguchi et al., 1998]; BaliBASE contains manually curated alignment information [J.D. Thompson et al., 2005]; and SABmark is focused on protein sequences of very low similarity [I.V. Walle et al., 2005]. The PREFAB database, which is used by MUSCLE and which does not require as much expert knowledge as BaliBASE, contains reference alignments in various structures [R.C. Edgar, 2004]. Only 13% of the BaliBASE reference alignments contain known structures, but secondary protein structures are highly predictable by actual sequence [R.C. Edgar, 2010].

In addition to the sequences themselves, one can also make use of potential mutational paths, either backwards- or forwards-in-time [S. Hoban et al., 2012]. A backwards-in-time simulator, known also as coalescent simulator, takes a small portion of samples in a large population to approximate a genealogy that generated the sequence polymorphisms. A forwards-in-time counterpart, a so called individual-based simulator, generates sequences considering the population as a whole. The coalescent method only assesses the generations containing genomic materials that influence the target samples. In turn, it is much more time-efficient than the forward algorithm. Despite the inefficiency of the forward method, it is more preferred for an accurate modeling. A hybrid algorithm combining both benefits has been suggested by B. Padhukasahasram [B. Padhukasahasram et al., 2008].

Recently, another simulator has been designed in forward-in-time fashion by following a certain evolution model. The sgEvolver is a simulator integrated in Mauve [A.E. Darling et al., 2010]. It does not model gene duplications, thus evaluations on repeat containing sequences are not feasible with sgEvolver. The EvolSimulator simulates a genome-scale evolutionary process and supports sequences acquired by horizontal/lateral genetic transfer (HGT/LGT), which is most common in prokaryotes. EvolSimulator does not produce indels [R.G. Beiko, and R.L. Charlebois, 2007]. Another simulator, ALF, aims at modeling all evolutionary processes shaping genomes [D.A. Dalquen et al., 2012]. It builds a species tree from an ordered set of ancestral genome sequences in order to create simulation data. At both the nucleotide- or amino acid-level, not only substitutions and indels, but also duplications, losses, genomic rearrangements, and speciation modeling are supported. The Evolver simulator models an averaged long-term evolutionary process of a single species [R.C. Edgar et al., 2009]. It mainly consists of inter- and intra-chromosomal modules. The inter-chromosomal module supports fusions, fissions, moves, duplications, and translocations in two or more chromosome; the latter provides substitution, indels, duplication, and tandem repeat events within a single chromosome.

In summary, the biggest problem is a lack of generality and low confidence of the established evaluation methods. Alternatively, visual inspections and manual editing on sequence alignments have been commonly used, but they are rarely reproducible. The actual number of mismatches and gaps may be underestimated, and the underlying evolutionary process may be ignored [M. Anisimova et al., 2010]. Moreover, visual inspection cannot easily, if at all, deal with entire genomes with many differences. To maintain an objective approach, simulation and statistical methods must be used, with a set of reasonable metrics.

The team responsible for the Alignathon [D. Earl et al., 2014] have suggested that they provide a generalized framework for evaluating MSA methods, but I disagree. Different MSA algorithms rely on diverse assumptions and definitions for each alignment, hence predicted alignments do not always reflect the ground truth. Moreover, each aligner will typically generate a different number of alignments. With these discrepancies, evaluation is complicated. With Alignathon methods, the worst possible performance estimates were often made even for very small simulation datasets, such that no correct prediction was reported, meaning zero sensitivity and specificity.

3.3.2 Results

Instead of ambiguously comparing multiple alignments to each other, I propose to more reliably perform evaluations based on known “positions” of genetic mutations by simulations. In this section, only SNPs are taken into account. All the other intra-chromosomal SVs such as inversions, translocations, and duplications are present in the simulation datasets, but they distort the evaluation in terms of observing local collinearity; results with SV events are shown in section 4.3. All experiments were performed on a 64-core Linux-compatible machine with 1-Tb memory installed.

3.3.2.1 Computational efficiency and scalability

Firstly, the measurements were performed with respect to the computational efficiency and scalability for the *A. thaliana* genome assemblies generated with the 1001 Genomes dataset. As shown in Figure 16, only the assemblies met the criterion that coverage exceeds 80% of the reference. Time and space efficiency are measured on genome assemblies of *A. thaliana* strains referred to as “1001G dataset” in the following. Since Kairos has to construct a population index for long reads or genome assemblies, all short reads of the 1001G dataset were first assembled using SPAdes 3.5.0. [A. Bankevich et al., 2012]. Initially, 1220 strains were assembled, but poor assemblies based on QUAST reports [A. Gurevich et al., 2013] , covering less than 80% of the Col-0 reference genome, were removed, leaving 1037 for further analysis. In the future, one may be able to apply error corrected long reads instead, which will reduce the time and space for assembling genomes. Initially, Cactus [B. Paten et al., 2011],

and Mugsy [S.V. Angiuoli, and S.L. Salzberg, 2011] were chosen to compare the performance of Kairos.

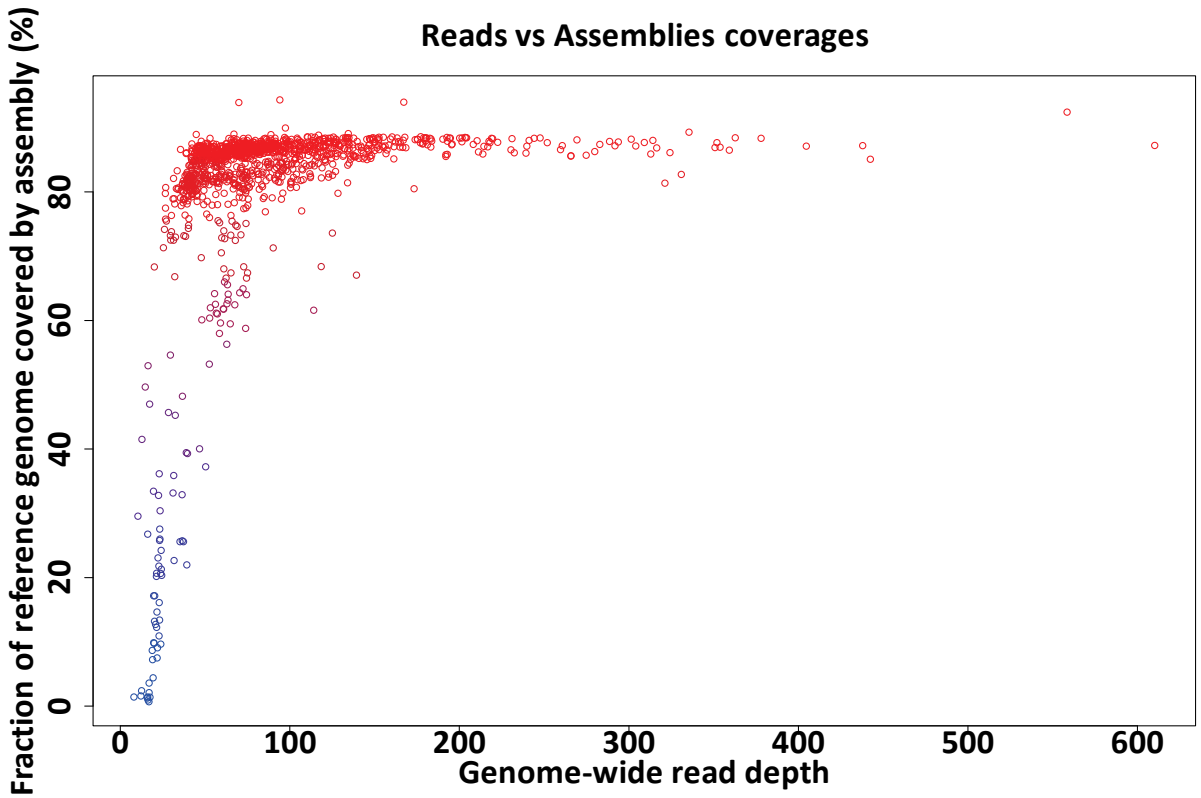

Figure 16. Relation between average genome-wide read depth and coverage of the reference genome by each assembly generated from these reads. Only assemblies covering at least 80% of *Arabidopsis thaliana* reference genome were selected to build a population index and aligned.

All the other aligners (Cactus, and Mugsy) failed to generate valid results in a month. In fact, they were designed to directly call neither SV events nor SNPs, which is the major reason why it has been difficult to evaluate their performance in comparative studies. Without such direct variant calling by a designer of a sequence aligner, the burden to develop a variant caller is imposed to the other software developers, obscuring the process of the evaluation, and ultimately may have led to the stagnation of improvements in MWGA studies. In turn, the evaluation scheme introduced here is a novel attempt to objectively measure the performance and hopefully convince developers of multiple whole genome aligners to attach the variant calling algorithms to their own aligners.

The alignments of the 1001G datasets were finished in 38 hours with a minimum *k*-mer size of 70. The multiple whole genome alignments alone took 9 hours. The peak memory consumption was reported while constructing the population index, at 121.1 Gb. Since the incremental construction of the population index can be applied at the expense of running time, the practical peak memory consumption would be 86.7 G when the population index is allocated in physical memory space. In a secondary storage, the population index consumes as little as 34.7 Gb.

3.3.2.2 Accuracy

Repeat DNA segments mainly complicate the evaluation of SNPs, thus both simple and complex genomes must be tested to observe their effects. *Escherichia coli, Arabidopsis thaliana,* and *Homo sapiens* genome sequencing projects have yielded high quality complete genomes. The *E. coli* K12 substrain W3110, the *A. thaliana* Col-0 reference genome, and the three largest chromosomes of a patched version of *H. sapiens* reference genome GRCh37 [M.J.P. Chaisson et al., 2015] were selected as ancestor genomes to simulate complete genomes. Mason variator 2.0.1 [M. Holtgrewe, 2010] was used to generate 64 artificial sequences for *E. coli* and *A. thaliana*, and 16 complete chromosomes of *H. sapiens*. Variant call format (VCF) files with a substitution rate of 0.001 and 1-6 bp indel rate of 0.000001 were also created by the simulator. In addition, intra-chromosomal translocations were simulated with the rate of 0.000001. Inversions, and duplications were generated with the rate of 0.000005.

The simulation of 16 *H. sapiens* sequences indicated that Kairos does not yet handle complex genomes well. For discussion, I briefly report the alignment performance on this dataset. Since it takes excessive time to create alignments, only the three largest *H. sapiens* chromosomes were tested with k=70. Chromosomes 1, 2, and 3, total length 658.4 Mb, were aligned in under 32 hours. The peak memory requirement for the construction of a population index was 102.1 Gb, and 25.9 Gb for the alignment.

Standard statistics were used to determine the accuracy of Kairos. The ground truth positions of SNPs and predictions were tested against each other. A true positive (TP) is defined as both position and state being correctly predicted. A false positive (FP) is a newly introduced base or a base that is different from the ground truth at a specific position. A false negative (FN) is a SNP that is not predicted by the aligner. Finally, a true negative (TN) is a non-variant position at which the aligner also does not predict a variant. Sensitivity is defined as TP/(TP+FN), which represents correct predictions over the entire ground truth. Specificity is defined as TN/(TN+FP), which denotes correctly rejected variants over all invariant positions. Precision is defined as TP/(TP+FP), which reflects all the correct predictions over all the predicted variants.

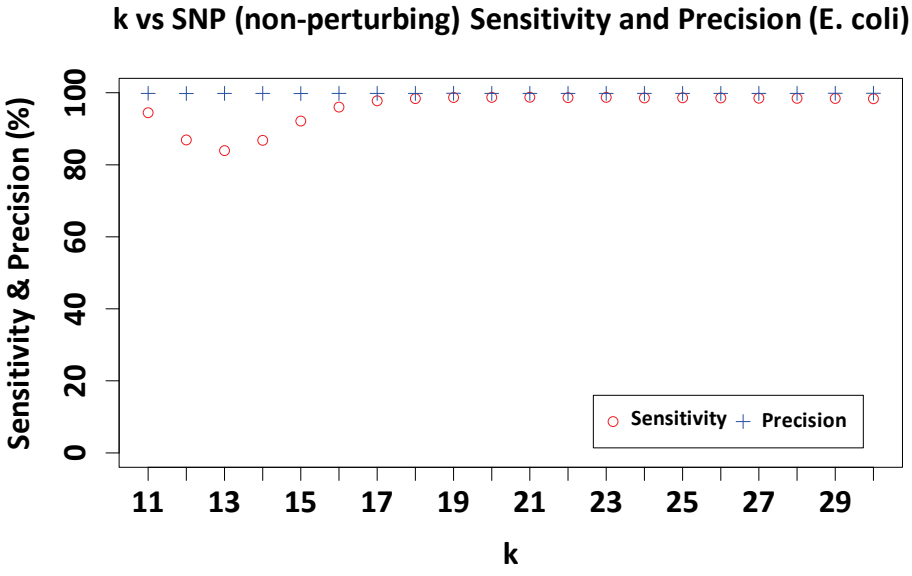

Figure 17. SNP sensitivity and precision without perturbing SVs on simulated *E. coli* datasets. The simulated datasets do not contain structural variants. Thus, SNPs and small indels are the only perturbing signals in the experiment.

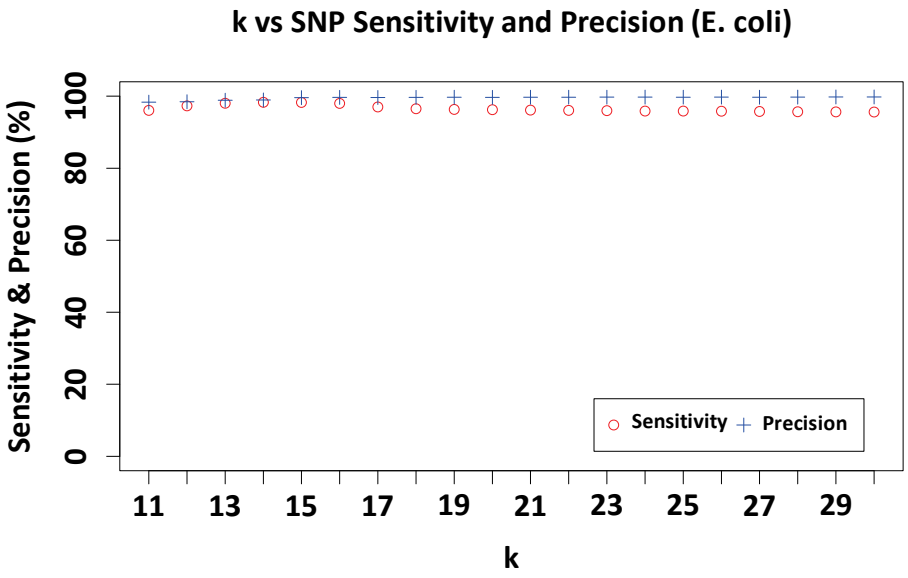

Figure 18. SNP sensitivity and precision with perturbing SVs on simulated *E. coli* datasets. The simulated datasets contain structural variants. Thus, SNPs, small indels, and SV events influence the performance.

As a first test for alignment performance, 64 complete *E. coli* genomes were simulated without SV events such as large insertions, deletions, inversions, duplications and intra-chromosomal-translocations; small insertions and deletions were permitted. The only parameter of Kairos that changes the sensitivity and precision is the minimum *k*-mer length. Different *k* values ranging from 11 to 30 were tested. Further experiments were not performed since results were already stable beyond k=18 (Fig. 17).

It was observed again that the optimal estimate *k* as defined in equation (2.2) could be used to select the initial minimum *k* value for any *k*-mer based extensions. For *E. coli* without large SVs (Fig. 17), *k* should be larger than 15 to guarantee uniqueness of *k*-mers. The highest sensitivity was observed at k=21 of 98.7583% with precision of 99.8335%. When performing the same evaluation along with SV events, the metrics are demonstrated a different trend. The highest performance is reported near the optimal *k* (k=15) such that the highest sensitivity of 98.3155% at k=14 is observed with the precision of 98.9670% (Fig. 18).

There is a slight loss of SNP information recovery without larger SVs with few *k* values because the uniqueness of pattern is not guaranteed and subsequently the reconstruction of split block fails (Fig. 17). The precision worsened near the optimal *k* value for highest performance without large SVs, but such a trend was not observed with large SVs. Thus, large SV events may affect SNP detection, increasing precision. The possible cause of the phenomenon is that more complexity in DNA sequences could lead to higher sequence specificity in defining collinear blocks. In other words, the uniqueness of a sequence become obvious if a certain split block involves in a SV event.

Peak memory consumption did not change much with different *k* (Fig. 19), since the population index preserves a constant memory footprint even when a different initial *k* is selected. However, to

obtain a higher sensitivity one may decrease the minimum *k* value, which in turn increases the time complexity (Fig. 20). A lower *k* increases the running time because it creates a larger suffix interval, which will be examined by the Occ() function. The exponential decay pattern reflects that the number of entries in a collinear blocks on average decreases as *k* increases since the uniqueness of the pattern is proportional to the length of pattern (equation 3.6).

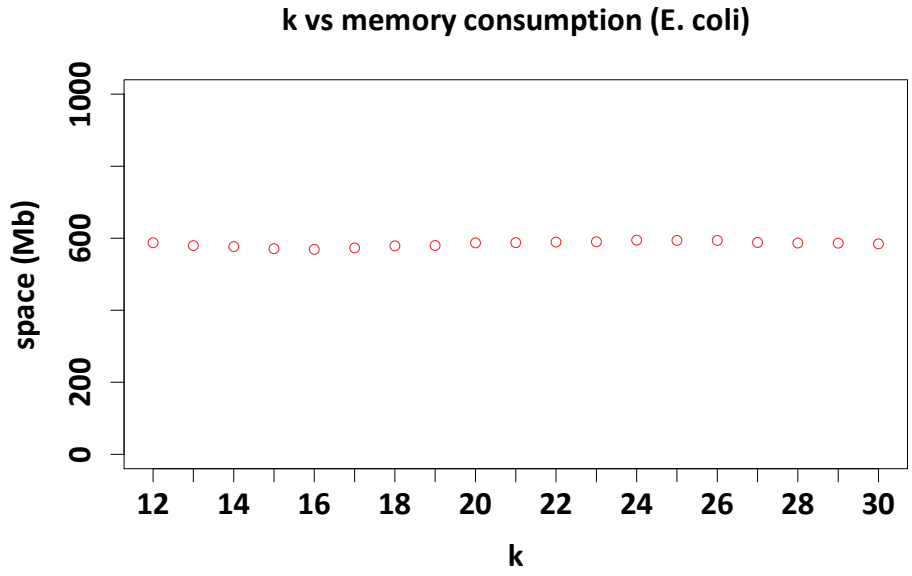

Figure 19. Peak memory consumption with large SVs in simulated *E. coli* datasets. The memory consumption is almost independent from the choice of the parameter *k* for *E. coli* dataset.

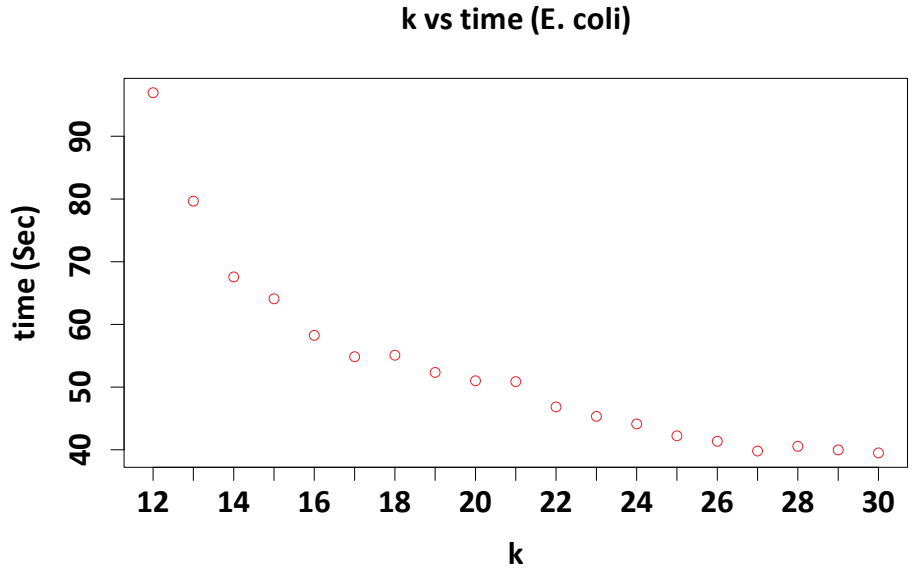

Figure 20. Time consumption with large SVs in simulated *E. coli* datasets. A small *k* value e.g. *k* = 12 increases the suffix interval of each collinear block, thus it takes long time to produce an alignment.

Since the initial experiment had demonstrated that larger SVs had only a small effect on sensitivity and precision, a similar trend could be expected for detecting SNPs. Thus, the experiment on *A. thaliana* simulation datasets without SVs is avoided. Instead, the final experiment was performed on a simulated datasets of 64 complete *A. thaliana* genomes with larger SVs. High sensitivity and precision were observed again, almost as high as for *E. coli* (Fig. 21). Moreover, the same trends in time (Fig. 22) and space complexity were witnessed (Fig. 23). I concluded that additional experiments are not necessary.

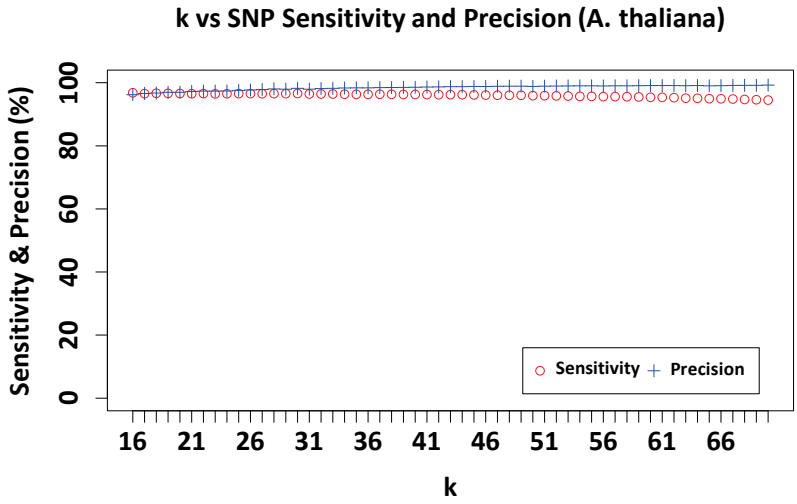

Figure 21. SNP sensitivity and precision with perturbing SVs on *A. thaliana* datasets. The simulated datasets contain structural variants.

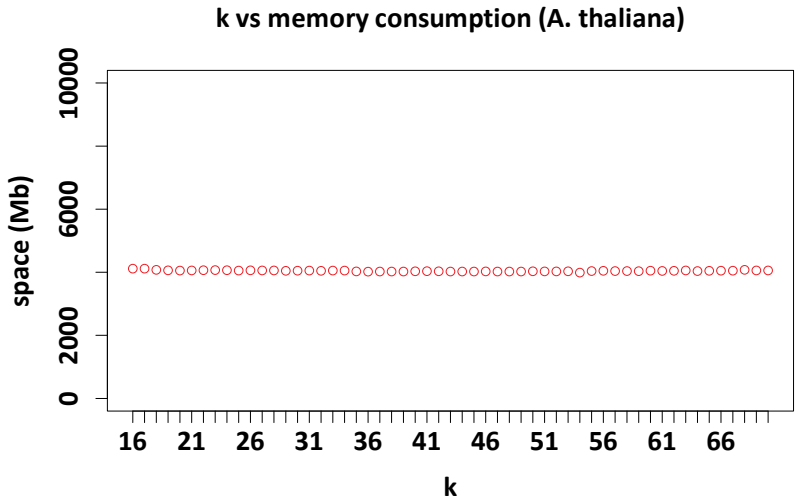

Figure 22. Peak memory consumption with perturbing SVs on *A. thaliana* datasets. The memory consumption of Kairos is likely to be independent from the choice of parameter *k*.

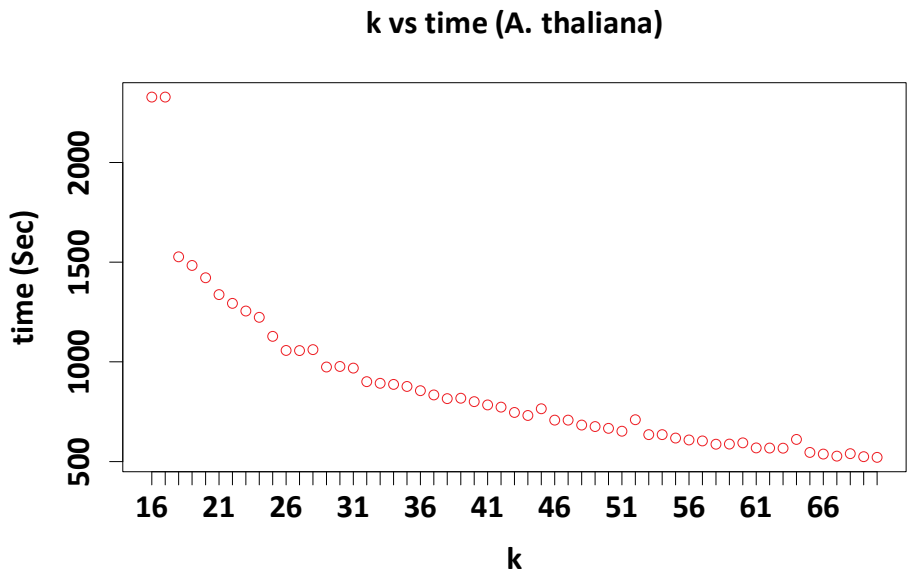

Figure 23. Time consumption with large SVs in simulated *A. thaliana* datasets. A small *k* value e.g. *k* = 16 increases the suffix interval of each collinear block, thus it takes long time to produce an alignment.

3.4 Discussion and Conclusion

The sequence alignment is a preliminary step for many biology analyses. The theories of multiple sequence alignment have emerged in early 1970s, and they have converged to multiple whole genome alignment algorithms after the MUMmer introduced [A.L. Delcher et al., 1999]. The established algorithms still cannot handle very large-scale genome datasets and their uses are limited mainly due to their space requirement and a lack of scalability. They are known to perform the alignments on large-scale micro-organism datasets but have not tested with a large number of eukaryotic genomes. Several algorithms have been proposed, but it is unclear when and where an algorithm performs the best. The difficulties in evaluation of MWGA are addressed and conceptually clearer evaluation methods are alternatively applied. I introduced a highly sensitive and computationally efficient multiple whole genome alignment algorithm, Kairos, in an attempt to overcome problems in previous MWGA methods.

With the advances in NGS technology, longer sequences have been available and immense engineering efforts have made to improve the quality of sequencing projects. In the future, we will be able to obtain a complete genome of a single organism from sequencers, thus an accurate and efficient multiple whole genome aligners may gain more significance. Short read alignment methods dependent on a single reference genome will be replaced by multiple whole genome aligners as the quality of genome assemblies improves. An accurate interpretation of heterogeneous biological datasets become feasible if one exploits a population-wide multiple sequence information rather than relying on the data obtained from pair-wise alignments. There have been several studies to provide an accurate and efficient multiple whole genome aligner to deal with large-scale datasets. However, Kairos is the first multiple whole genome aligner achieving both easily reproducible accuracy and efficiency for very large-scale datasets such as 1001G datasets. Aside from eukaryotic populations, the whole genome alignment algorithm would be useful to identify mutations, and horizontal gene transfers in microbiome genealogy.

The NGS technology is still not fully matured. For instance, we cannot obtain a complete genome of a human individual or a cell population from a single sequencing library yet. The genome assembly algorithms and error correction modules are imperfect in terms of accuracy and computational efficiency. When every problem is solved regarding the completeness of sequences, the value of multiple whole genome aligners become obvious in population genetics, metagenomics, and biomedical studies. In the future genomics studies, Kairos will be able to play a pivotal role though it does not provide a production-ready solution yet for all the MWGA problems as it could have not aligned human genomes in a population efficiently.

Understanding a genome of new organism in an algorithmic perspective is very different from how biologists see its abstraction. The algorithms may have to evolve themselves as the complexity of the organism increases. As a human being researcher, not every aspect of genomic context could have been observed given uncertainty in datasets, but enormous efforts are made to obtain a better sensitivity and a higher precision, especially, for 1001G datasets. Though the initial version of Kairos has been being optimized for *A. thaliana* genomes because of the accessibility, further improvements will be added to support larger and more complex genomes.

3.5 References

S.V. Angiuoli, and S.L. Salzberg (2011) Mugsy: fast multiple alignment of closely related whole genomes, Bioinformatics, 27(3):334-342.
M. Anisimova, G.M. Cannarozzi, and D.A. Liberles (2010) Finding the balance between the mathematical and biological optima in multiple sequence alignment, Trends Evol Biol., 2:7.
W. Bains (1986) MULTAN: a program to align multiple DNA sequences, Nucl.Acids Res., 14(1):159.
A. Bankevich, et al. (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing, J Comput Biol., 19(5):455-477.
R.G. Beiko, and R.L. Charlebois (2007) A simulation test bed for hypotheses of genome evolution, Bioinformatics 23:825-831.
M. Brudno, M. Chapman, B. Göttgens, S. Batzoglou, and B. Morgenstern (2003) Fast and sensitive multiple alignment of large genomic sequences, BMC Bioinformatics, 4:66.
M.J.P. Chaisson et al. (2015) Resolving the complexity of the human genome using single-molecule sequencing, Nature 517, 608–611.
C. Chothia, and A.M. Lesk (1986) The relation between the divergence of sequence and structure in proteins, EMBO J., 5(4):823-6.
D.A. Dalquen, M. Anisimova, G.H. Gonnet, and C. Dessimoz (2012) ALF - A Simulation Framework for Genome Evolution, Mol Biol Evol., 29(4):1115-1123.
A.E. Darling, B. Mau, and N.T. Perna (2010) progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement, PLoS One., 5(6):e11147.
A.L. Delcher, S. Kasif, R.D. Fleischmann, J. Peterson, O. White, and S.L. Salzberg (1999) Alignment of whole genomes. Nucleic Acids Res., 27(11):2369-76.
A.L. Delcher, A. Phillippy, J. Carlton, S.L. Salzberg (2002) Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res., 30(11):2478-83.
C. Dessimoz, and M. Gil (2010) Phylogenetic assessment of alignments reveals neglected tree signal in gaps, Genome Biol., 11(4):R37.
D. Earl et al. (2014) Alignathon: a competitive assessment of whole-genome alignment methods, Genome Res., 24(12):2077-89.
S.R. Eddy (1998) Multiple-alignment and -sequence searches. Trends Guide to Bioinformatics, 15-18.
R.C. Edgar (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput, Nucl Acids Res., 32(5):1792–97.

R.C. Edgar (2010) Quality measures for protein alignment benchmarks, Nucl. Acids Res., 38(7):2145-53.
R.C. Edgar. G. Asimenos, S. Batzoglou and A. Sidow (2009) EVOLVER, http://www.drive5.com/evolver
D.F. Feng, and R.F. Doolittle (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J Mol Evol., 25(4):351-60.
O. Gotoh (1990) Consistency of optimal sequence alignments. Bull Math Biol., 52(4):509-25.
O. Gotoh (1996) Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments, J Mol Biol., 264 (4): 823-38.
A. Gurevich, V. Saveliev, N. Vyahhi, G. Tesler (2013) QUAST: quality assessment tool for genome assemblies, Bioinformatics, 29:1072-1075.
B.G. Hall (2008) How well does the HoT score reflect sequence alignment accuracy?, Mol Biol Evol., 25(8):1576-80.
D.G. Higgins, and P.M. Sharp (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer, Gene, 73(1):237-44.
S. Hoban, G. Bertorelle, and O. Gaggiotti (2012) Computer simulations: tools for population and evolutionary genetics. Nature Reviews Genetics, 13(2):110-22.
M. Holtgrewe (2010) Mason - a read simulator for second generation sequencing data. Technical Report TR-B-10-06, Institut für Mathematik und Informatik, Freie Universität Berlin.
U. Keich, M. Li, B. Ma, and J. Tromp (2004) On Spaced Seeds for Similarity Search, Discrete Appl. Math, 138(3):253-263.
M. Kimura (1983) The Neutral Theory of Molecular Evolution, Cambridge University Press, Cambridge.
S. Kurtz, A. Phillippy, A.L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S.L. Salzberg (2004) Versatile and open software for comparing large genomes, Genome Biol., 5(2):R12.
G. Landan, and D. Graur (2007) Heads or tails: a simple reliability check for multiple sequence alignments, Mol Biol Evol., 24(6):1380-3.
T. Lassmann, and E.L.L. Sonnhammer (2005) Automatic assessment of alignment quality, Nucl Acids Res., 33(22):7120-8.
C. Lee, C. Grasso, and M.F. Sharlow (2002) Multiple sequence alignment using partial order graphs. Bioinformatics, 18(3):452-64.
H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup (2009) The Sequence Alignment/Map format and SAMtools, Bioinformatics, 25(16):2078-9.
K. Lin, S. Smit, G. Bonnema, G. Sanchez-Perez, and D. de Ridder (2015) Making the difference: integrating structural variation detection tools, Brief Bioinform., 16(5):852-64.
B. Ma, J. Tromp, and M. Li (2002) PatternHunter: faster and more sensitive homology search. Bioinformatics. 18(3):440-445.
H.M. Martinez (1983) An efficient method for finding repeats in molecular sequences. Nucleic Acids Res., 11(13):4629-34.
K. Mizuguchi, C.M. Deane, T.L. Blundell, and J.P. Overington (1998) HOMSTRAD: a database of protein structure alignments for homologous families. Protein Sci., 7(11): 2469-71.
G.W. Moore, M. Goodman, and J. Barnabas (1973) An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets, J Theor Biol., 38(3):423-57.
S.B. Needleman, and C.D. Wunsch (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins, J Mol Biol., 48:443-453.
C. Notredame, D.G. Higgins, and J. Heringa (2000) T-Coffee: a novel method for fast and accurate multiple sequence alignment. J. Mol. Biol., 302:205-217.
B. Padhukasahasram, P. Marjoram, J.D. Wall, C.D. Bustamante, and M. Nordborg (2008) Exploring population genetic models with recombination using efficient forward-time simulations, Genetics, 178(4):2417-27.

B. Paten, M. Diekhans, D. Earl, J.S. John, J. Ma, B. Suh, and D. Haussler (2011) Cactus graphs for genome comparisons. J Comput Biol., 18(3):469-81.
W.R. Pearson, and D.J. Lipman (1988) Improved tools for biological sequence comparison, Proc Natl Acad Sci USA, 85(8):2444-8.
P.A. Pevzner, H. Tang, and G. Tesler (2004) De novo repeat classification and fragment assembly, Genome Res., 14(9):1786-96.
B. Raphael, D. Zhi, H. Tang, and P.A. Pevzner (2004) A novel method for multiple alignment of sequences with repeated and shuffled elements, Genome Res., 14(11):2336-46.
T. Rausch, A.-K. Emde, D. Weese, A. Döring, C. Notredame, and K. Reinert (2008) Segment-based multiple sequence alignment, Bioinformatics, 24(16):i187-i192.
M. Roberts, W. Hayes, B.R. Hunt, S.M. Mount, and J.A. Yorke (2004) Reducing storage requirements for biological sequence comparison, Bioinformatics, 20(18):3363-3369.
D. Sankoff (1975) Minimal mutation trees of sequences. SIAM J. Appl. Math., 78:35-42.
E. Sobel, and H.M. Martinez (1986) A multiple sequence alignment program, Nucl. Acids Res., 14(1):363-374.
J.D. Thompson, D.G. Higgins, and T.J. Gibson (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice., Nucleic Acids Res., 22(22):4673-80.
J.D. Thompson, P. Koehl, R. Ripp, and O. Poch (2005) BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. Proteins, 61(1):127-36.
J.D. Thompson, F. Plewniak, and O. Poch (1999) A comprehensive comparison of multiple sequence alignment programs, Nucleic Acids Res., 27(13):2682-90.
I.V. Walle, I. Lasters, and L. Wyns (2005) SABmark--a benchmark for sequence alignment that covers the entire known fold space. Bioinformatics, 21(7):1267-8.
M.S. Waterman (1984) General methods of sequence comparison, Bull. Math. Biol., 46:473-500.
M.S. Waterman (1986) Multiple sequence alignment by consensus, Nucleic Acids Res., 14(22):9095-9102.
Y. Zhang, and M.S. Waterman (2003) An Eulerian path approach to global multiple alignment for DNA sequences, J Comput Biol., 10(6):803-19.

Chapter 4

4. Structural variant calling

4.1 Introduction

A genome is in principle a message delivering information that ultimately manifest itself in traits of an organism. The genome of each individual of even a single species typically contains different DNA sequences and DNA arrangements. These differences are called variations. On a small scale, single nucleotide polymorphisms (SNPs) are common. The belief that SNPs are major causes of various genetic disorders has attracted many researchers [International SNP Map Working Group, 2001, International HapMap Consortium, 2003, 2005]. More recently, the focus has expanded to include also copy number variants (CNVs) and further structural variants (SVs) [A.J. Iafrate et al., 2004, J. Sebat et al., 2004]. The frequency of SVs is lower than that of SNPs, but the length of DNA sequences involved is far longer, hence they can have often more drastic effects, especially when they are in coding regions. SVs can thus directly contribute to causing genetic disorders [J.L. Freeman, 2006]. A CNV denotes a copy number change of DNA fragments, typically of length 1 kb or more [L. Feuk et al., 2006].

SVs are classified as either balanced or imbalanced contingent upon the changes in genomic material (Fig. 24). The balanced variations encompass inversions where a genomic segment is reverse complementarily exchanged, and translocations or transpositions where a single genomic segment moves to a different genomic region. An intra-chromosomal translocation occurs within a homologous chromosome while an inter-chromosomal translocation arises between two chromosomes.

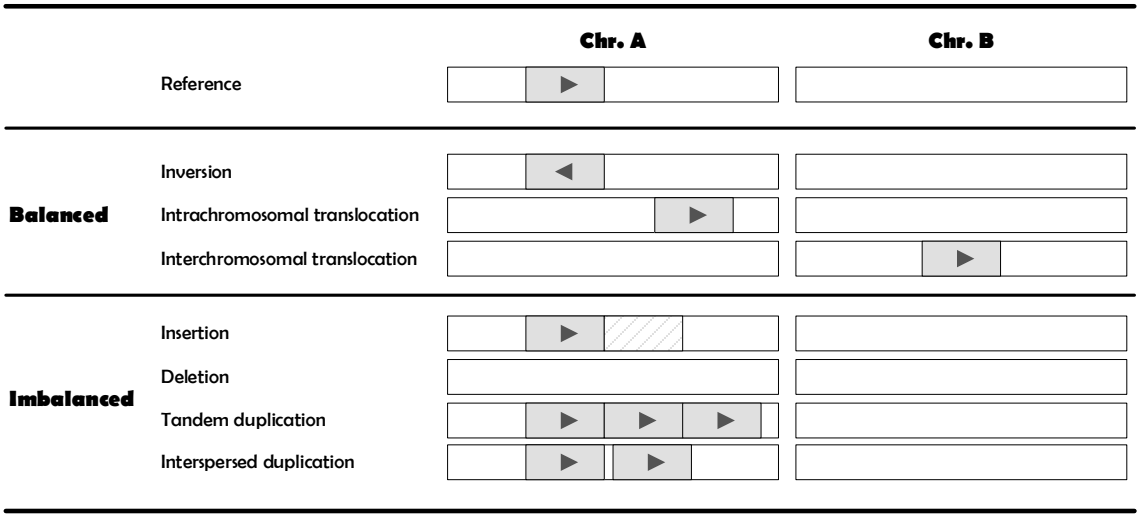

Figure 24. Structural variations. A balanced variant preserves the amount of genomic materials while an imbalanced variant increase or decrease the materials.

Imbalanced variants denote gain or loss events of DNA segments such as deletions, insertions, and duplications. A duplication appears in either a tandem or an interspersed disposition. The term

duplication does not imply that there is more than one copy of a perfectly identical genomic sequence, but that the sequences are typically at least 90% identical [E.E. Eichler, 2001, L. Feuk et al., 2006].

Comparative genomic hybridization (CGH) used to be a common method to detect CNVs. Initially it was carried out on metaphase chromosomes [A. Kallioniemi et al., 1992] , but was then replaced by array CGH (aCGH), which improved resolution to the order of 100 kb [S. Solinas-Toldo et al., 1997, D. Pinkel et al., 1998]. The aCGH recognizes variations based on the ratio of DNA labeled with different fluorochromes, hence inversions cannot be identified. Massive fosmid end sequencing using Sanger technology further increased resolution and detection capacity such that inversions could be detected [E. Tuzun et al., 2005]. The direction and insert size of a sequencing library are the fundamental data to detect variations. If a paired read is aligned at the distance and orientation expected from the reference sequence, it is called concordant match. The opposite is a discordant match. The throughput of paired-end mapping (PEM) methods have further been improved by utilizing NGS platforms such as 454 as well as de novo assemblies [M. Margulies et al., 2005]. The insert size of the library determines the maximum size of insertions that can be detected, while there are fewer limits for deletions or duplications. The most difficult SV type remain inversions [J.O. Korbel et al., 2007]. The increase in throughout provided by Illumina sequencing technology further increased sensitivity for SV and CNV detection [P.J. Campbell et al., 2008].

A probabilistic model based on sequence similarity, length distribution of SVs, and number of supporting reads was suggested for each SV [S. Lee, et al., 2008]. The SeqSeq algorithm calculated p-values of copy number changes by assessing the log ratio of reads in dynamic-length windows. A p-value would be calculated if it was below a specific threshold. Variations were finally predicted by taking into account multiple windows. The VariationHunter algorithm applied a probabilistic model on discordant pairs for detecting insertions, deletions, and inversions [F. Hormozdiari et al., 2009]. Though similar factors as for SeqSeq were applied in the modeling process, VariationHunter could not call duplications or translocations, since concordant information was not considered. In simulation, the detection of insertions suffered from high false positive rate. BreakDancer [K. Chen et al., 2009] was capable of calling indels and inversions as well as translocations with higher detection rate than either VariationHunter or another algorithm, MoDIL [S. Lee et al., 2009]. It is unclear whether the better performance reported is due to the application of a different sequence aligner, or because of more accurate modeling. Finally, GASV provided integrated predictions of SVs based on the signals from both PEM and aCGH [S. Sindi et al., 2009], but there was substantial uncertainty in determining the breakpoints. All these algorithms had both high false positive and false negative rates, and often did not agree in their prediction, due to idiosyncratic decisions to merge junctions or not and multi-state SVs at the same locus. A non-empty intersection of breakpoint regions is identified by the plane sweep algorithms. The sweep lines of different mathematical functions determine the events in which they are involved.

A separate set of algorithms used one perfectly mapping read paired with a partially or non-mapping pair to detect indels. For example, a unique perfect match of a pair can serve as anchor, and a soft-clipped match of the other paired read can be used to detect an indel [R.E. Mills et al., 2006]. The soft-clipped sequence is often called a split read, which is also applied in Pindel [K. Ye et al., 2009], and HYDRA [A.R. Quinlan et al., 2010]. Pindel extended the patterns during the alignment process in order to increase efficiency and accuracy. HYDRA increased sensitivity by a virtue of exploitation of two alignment algorithms, and reported combined results, thus the false positive rate of SV callings may have been reduced. HYDRA examined if a discordant pair maps to multiple locations with regard to segmental duplications. There is even an indel-specific algorithm, NovelSeq, which used genome assemblies for the sake of contamination removal [I. Hajirasouliha et al., 2010].

The challenge that I addressed in this chapter is the highly parallel analysis of a large number of genomes. GenomeSTriP is a unique analytical method handling many population samples [R.E. Hand-

saker et al., 2011]. Despite its generality, the GenomeSTriP paper demonstrated only an exemplary result for deletions of length 45 to 995,664 bp in the 1000 Genomes Project [The 1000 Genomes Project Consortium, 2012] , but the high overlap rate of 99.1% with array-based genotypes and 3.7% of composite false discovery rate (FDR) indicated the superiority of the method. High accuracy was achieved by considering "coherent" criteria such that all discordant read pairs at a certain location in a population set were likely to come from the same deletion allele. The "heterogeneity", which is derived from a true polymorphism of individual genomes, was assessed with a χ^2 test to avoid chimeric alignments. Furthermore, multiple signals having orthogonal error properties were integrated to identify putative deletions.

The AGE algorithm started an alignment at both 3' and 5' ends on top of two alignment matrices, hence breakpoints with large gaps in the middle could be accurately identified [A. Abyzov, and M. Gerstein, 2011a]. Thus, one can say that this algorithm originated from the concept of the HoT score [G. Landan, and D. Graur, 2007]. DELLY integrated a handful of established algorithms to recognize distinct genomic rearrangements. An undirected-weight graph, which was constructed from discordant pairs, represents a certain structural rearrangement followed by split-read alignments. The diagonals in a scoring matrix were filtered out by the number of *k*-mer hits to derive a consensus matrix to which double dynamic programming was applied, similar to the AGE algorithm [T. Rausch et al., 2012]. LUMPY jointly considered the distribution of PEMs, split-read alignments, and generic evidence to estimate the breakpoints, which together supposedly greatly increased accuracy [R.M. Layer et al., 2014].

With the advances of NGS technology, the number of complete genomes continues to grow tremendously. The accuracy of SV detection can be improved if a test and the reference genome are nearly complete [K. Lin et al., 2015]. Sequencing errors and low-coverage regions have been major barriers to accurate SV calling. Though many SV callers have been introduced, none has found broad acceptance in the community. A single caller may provide limited capability in variation detection and have poor sensitivity for a specific variant. Hence, an ensemble method would likely be most appropriate (see section 0.3.1). Notably, only few algorithms, such as GenomeSTriP, have been designed to take population information into account [R.E. Handsaker et al., 2011].

Four signals, assembly, split read, PEM, and read depth, have been widely applied to detect SVs, yet PEM and read depth are less relevant once long reads or contigs are available. Alas, it is unclear which previous solutions would succeed in generating legitimate SVs with long reads, genome assemblies, and complete genome sequences. As discussed, the accuracy of SV calling is highly dependent on correctness of sequence alignments. For comparative biology, a multi-WGA is far better than manifold PEMs in terms of space, time complexity, and accuracy.

To overcome the discussed hurdles, I introduce a structural variant caller, Apollo, which aims at fully-fledged supports for all balanced and unbalanced variations among individuals in a given population. Apollo is the first large-scale algorithm identifying different types of variations among individuals in a population. Apollo detects variations on top of outcomes from the Kairos algorithm described in section 3.2. Because of its novel approach, there is no straightforward comparison with existing tools (see section 3.3).

4.2 Method

4.2.1 Overview

The approach of multiple WGAs can be applied to error-corrected long reads, contigs, or complete genome sequences rather than short reads with low error rates or raw long reads with high error rates. To reliably detect SV events, signals such as positions and orientations of alignments are crucial. Even with genome assemblies generated on top of long reads, those signals are imperfect, especially in terms of positional information. Hence, in this study, a long sequence means a complete genome sequence rather than a long read or a contig.

Algorithm DetectSVs

Input: Collinear blocks CB
Output: Structural variants SV
SV := []
FOREACH (A, B, C) IN CB:
DetectSNPs (A, B)
DetectSNPs (B, C)
  FOREACH S_{i-1}, S_i, S_{i+1} IN A, B, C:
    IF S_i is a sequence involved in a deletion with respect to S_{i-1}, and S_{i+1}:
      B.type = “D”
      IF S_i is translocated:
        B.type = “O” // a deleted sequence of an intra-chromosomal translocation
        IF $Chr(S_i) != Chr(S_{i-1})$ and $Chr(S_{i-1}) = Chr(S_{i+1})$:
          B.type = “E” // a deleted sequence of an inter-chromosomal translocation
    IF S_i is a sequence involved in an insertion-type SV with respect to S_{i-1}, and S_{i+1}:
      B.type = “I”
      IF S_i is inverted:
        B.type = “V”
      ELSE IF $Occ_i(S_i) > 1$:
        B.type = “R” // duplication
      ELSE IF S_i is translocated:
        B.type = “P” // an inserted sequence of an intra-chromosomal translocation
        IF $Chr(S_i) != Chr(S_{i-1})$ and $Chr(S_{i-1}) = Chr(S_{i+1})$:
          B.type = “F” // an inserted sequence of an inter-chromosomal translocation
SV.PushBack(A)
SV.PushBack(B)
Output SV

Different SV events yield MSAs of distinct positions and orientation with respect to the reference genome. Some functions and notations are taken directly from section 3.2. $Occ_i(s)$ is a function that returns the occurrence of a pattern in a particular individual. $Occ_r(s)$ returns the occurrence of a pattern in a reference genome. Chr(s) returns the identifier of a chromosome in an individual organism given a pattern. For a prokaryotic genome, it will usually return 0 in the 0-index system. A raw MSA provides locally valid information to detect SNPs and small indels within 50 bp. To detect SVs, a collinear block should be re-examined in conjunction with neighboring collinear blocks, as shown in Figure 25.

A collinear block S contains two coordinates: S_{rs}, and S_{re}, indicating the starting and the last positions in a reference genome. A split block stores S_{ts}, and S_{te} denoting the positions in a test genome. Three collinear blocks A, B, and C are assumed to be present hereafter while explaining each SV event. The collinear block B indicates the current block involved in predicting a certain event. Conditions, either $S_{te}[i] = S_{ts}[i+1]$ or $S_{rs}[i] = S_{ts}[i]$ is not always satisfied because any imbalanced mutations located in proximal positions with S_{ts}, affect S_{ts} as a progressive alignment performs. B is always a broken block or a split block followed by an empty block while detecting a SV event.

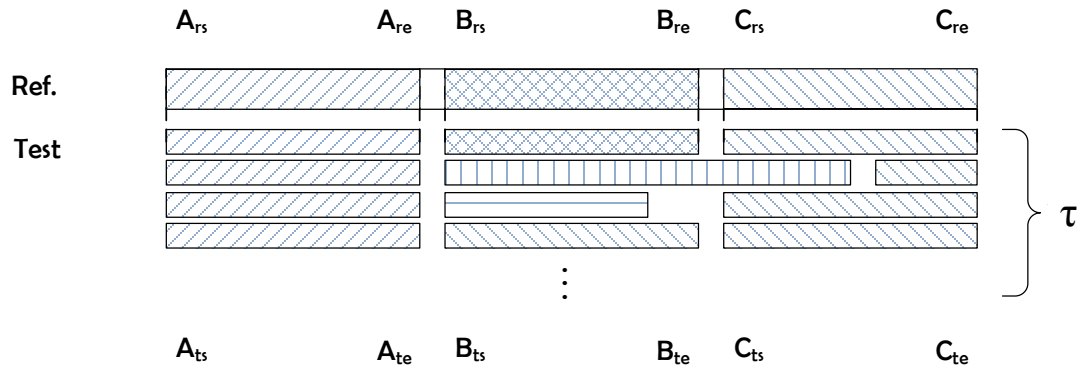

Figure 25. Collinear blocks of A, and C. B can be either a collinear block or a broken block. Depending on the types of SV event involved, the lengths of split blocks in B can be different.

To address differences among SV events, assumptions are made such that three collinear blocks are adjacent and the orientation of alignment is not changed while tracking the reference genome sequence as follows:

$$\overrightarrow{s_{r:j-1}[k...l]} \Rightarrow \overrightarrow{s_{r:j}[m...n]} \Rightarrow \overrightarrow{s_{r:j+1}[o...p]}$$

$$s_{r:j}[k...l] \Rightarrow s_{r:j+1}[m...n], m = l+1 \ \ or \ \ m = l$$

The condition to determine a SV event is summarized in Table 15 (Algorithm DetectSVs).

Table 15. Criteria of structural variants in a test genome.

Variant	Criterion
Deletion	$0 < Occ(\mathrm{s}_r)$ and $0 = Occ_i(\mathrm{s}_r)$
Insertion	$0 = Occ(\mathrm{s}_r)$ and $1 = Occ_i(\mathrm{s}_r)$
Duplication	$0 \leq Occ(\mathrm{s}_r)$ and $1 < Occ_i(\mathrm{s}_r)$
Intra-chromosomal translocation	$s_{i:j}[k'...l'] \Rightarrow s_{i:j+1}[m'...n'], k' > m', l' \ll m',\ 1 = Occ(\mathrm{s}_r)$ and $1 = Occ_i(\mathrm{s}_r)$
Inter-chromosomal translocation	$\mathrm{Chr}(\mathrm{S}_{i:j-1})=\mathrm{Chr}(\mathrm{S}_{i:j+1})\neq\mathrm{Chr}(\mathrm{S}_{i:j}),\ 1 = Occ(\mathrm{s}_r)$ and $1 = Occ_i(\mathrm{s}_r)$
Inversion	$\overrightarrow{s_{i:j-1}[k'...l']} \Leftrightarrow \overleftarrow{s_{i:j}[n'...m']} \Leftrightarrow \overrightarrow{s_{i:j+1}[o'...p']}$

4.2.2 Deletions

When a deletion relative to the reference occurs, the reference sequences are longer than the test sequences in the focal region of the genome. For a PEM method, two short sequences are mapped to a genomic region and an insertion size is estimated by a short read aligner. Both ends of a pair are mapped in opposite directions such that the second-end is aligned in the reverse complementary way. During molecular biology construction of a sequencing library, the distribution of insert sizes is determined. Both ends being mapped at a distance larger than the expected insert size (mean ± an arbitrarily chosen variance) potentially indicates a deletion event [E. Tuzun et al., 2005]. However, without determining the gap sequence between the ends, a prediction for a deletion cannot be complete, as it cannot be distinguished from a translocation event, since a translocation event can be viewed as a combination of a deletion and an insertion event.

In the algorithm proposed here, instead two adjacent collinear blocks B and C are required to detect a deletion. The distance between two split blocks, (C_{ts} - B_{te}), is usually one or zero in the test genome. However, two collinear blocks are located far apart in the reference genome of distance (C_{rs} - B_{re}), which indicates an approximated length of the deleted sequence. The deleted sequence must be absent from the test genome, thus the criterion, $Occ_i(s_r) = 0$, is checked. However, test genomes from other individuals may not have the same deletion event, thus $Occ(s_r)$ function may return a value larger than 0. The sequence information in a population increases the explanatory power of a MSA to determine a deletion event, which is impossible with PEM-only methods relying on a single reference genome. It is because such information is available only if the gap sequence between two ends is reconstructed by local genome assemblies.

4.2.3 Insertion-type events

4.2.3.1 Insertions

An insertion event has characteristics opposite to a deletion event such that the test genome contains additional sequences. An insertion can be detected with a PEM when the distance of mapped ends is shorter than the expected insert size, although additional SV events are significant confounders. Aside from a translocation event, additional confounders exist for detecting an insertion event, namely inversions, and duplications. It is obvious that a translocation and a duplication share certain characteristics, i.e., presence of sequence from the reference genome in a new place in the test genome. To correctly reject a duplication event, the length of the gap sequence should be smaller than 2. If orientations of both ends of a read pair are not considered, the distinction between an inversion and an insertion is unclear. To overcome this problem, a local assembly procedure can be performed when using PEM based methods. The local assembly strategy has been regarded as a reasonable approach to improve the accuracy, since short read alignments do not provide information about the sequences in the alignment gap. Thus, inversions cannot be reliably detected by PEM-only methods. It should be accompanied by sequence assembly comparisons [L. Feuk et al., 2006].

After completely assembling a genomic region of interest, an insertion can be much more accurately detected. Except for highly divergent regions, two collinear blocks are directly adjacent in the reference genome (distance 0 or 1 bp) due to the progressive nature of multi-WGAs. When an insertion is involved, adjacent split blocks are aligned far apart in the test genome, which is different from PEM methods. The inserted sequence is missing from the reference genome, but it should be found only once in a particular individual genome, thus the requirement is $Occ_r(s) = 0$. If $Occ_i(s)$ returns a value larger than, but not equal to 1, the detected event should be classified as a duplication rather than an insertion. If the criterion $Occ_i(s)=1$ met, then it is a candidate translocation. In addition, the orientation of *s* should be checked, so that an inversion is not erroneously predicted as an insertion.

4.2.3.2 Duplications

For PEM based methods, a duplication event is detected only if the same sequence is mapped to distinct genomic positions. However, such correct alignments with short reads are difficult, because sequences of high similarity are most likely aligned to a single genomic location rather than true multiple positions. Coverage estimates may help to infer a duplication. However, short read coverage is uneven and correct modeling is difficult [J.O. Korbel et al., 2007, A. Abyzov et al., 2011b]. In fact, the conventional algorithms have not explicitly addressed the "copy number" of duplication segments [K. Chen et al., 2009, K. Ye et al., 2009, I. Hajirasouliha et al., 2010, A. Abyzov, and M. Gerstein, 2011a]. Even if genome assemblies are available, neither the positional uncertainty nor the coverage problem can be easily solved. Contigs generated from a de Bruijn graph of short reads often do not convey repeat information when the length of duplication is longer than the *k*-mer length used to construct the de Bruijn graph.

A duplication can appear as either tandem or interspersed repeats, identified as multiple occurrence of a pattern near the end of a split block. The underlying FM-index of a population provides a way to track repeats shorter than the input sequences: $Occ_i(s)$. Hence a population index built on complete genome sequences can reveal any length of repeat regions. A partial sequence that is identified by backward tracking of a split block and that occurs more than once in an individual is an accurate signal for duplication. If the lengths of a repeat segment within a single split block are close to equal, then the block is involved in a tandem repeat, otherwise there is an interspersed repeat.

The first instance of a duplication event cannot be easily recognized in the first split block because the sequence of the first block is identical to the reference sequence at this position. The original locations of all the repeat segments can only be identified via the inserted segments at the last split block of the repeat DNA segments. These inserted DNA segments appear as a long broken block. Therefore, an insertion tag is assigned to the broken block. By checking the criterion, Occ(s) > 1, positions of all the DNA segments of a duplication event can be simultaneously discovered. Because of this algorithmic specificity, repeat detection would be become more difficult if more interspersed short repeats are in a genome. To increase precision, $Occ_r(s)$ should be checked altogether to learn the original copy number of *s*.

4.2.2.3 Inversions

For PEM methods, an inversion is identified by the inconsistency in orientation between the alignments of the fragment ends and the reference. Similarly, adjacent collinear blocks may reveal different directions of alignments while performing forward tracking. For instance, when the alignments of A and C are in forward direction while the one of B is in the reverse complementary orientation, B is predicted to be involved in an inversion event. Given the number of reads, *N*, a population index can store *2N* reads by inserting forward and reverse complementary reads. Hence, a suffix identifier pointing to a read number greater than *N* indicates a reverse complementary sequence with respect to the original input sequence. With a notation '+' for forward strand, '-' for reverse complementary strand, either (+-+) or (-+-) indicates an inverted DNA segment at block B.

Because an inversion event can include SNPs or small indels in the middle of an inverted DNA segment, the detection procedure should attempt to reveal the starting and the last position of the inversion with respect to the reference coordinates. Both forward- and backward-tracking are simultaneously performed for the suffix identifier of A. The condition for forward-tracking, $Occ_i(s_r) \geq 1$, is always satisfied because the suffix identifier of A belongs to a single individual. Backward-tracking is performed on a reverse complementary sequence, rendering the initial condition, $Occ_i(\overline{s_r}) \leq M$, for backward-tracking. Both tracking procedures are performed until the criterion for the backward-tracking turns into $Occ_i(\overline{s_r}) = 1$. With this procedure, the last position of the event is revealed. To find out the starting position of the event, the suffix identifier of C should be used. In this context, backward-tracking represents the current individual organism, thus $Occ_i(\overline{s_r}) \geq 1$ is kept. The letters obtained from the backward-tracking are reversed and provided to the forward tracking. When the criterion for the forward-tracking, $Occ_i(s_r) = 1$, is met, the tracking procedure to find the starting position is terminated.

4.2.4 Translocations

A translocation is much more difficult to be detected by PEM methods since the event can be identified as either an insertion or a deletion. In fact, it is impossible to detect a translocation if a population index was not built upon complete genome sequences. In this case, either a short read or a contig is redundant so that sequences sharing the same prefix in the index hinder the population index from returning the true frequency and location of prefix *s*. Moreover, the fragmented nature of contigs and short reads impede a reliable prediction of the SV length and class since they cause early termination of any tracking procedures.

An intra-chromosomal translocation event must be considered in a context with only sequences that are translocated with simultaneous deletion of the original copy to be distinguished from duplica-

tion events. At a given position, the sequence of B represents a deleted sequence with respect to the reference genome, having inserted elsewhere. The distance between adjacent blocks, A and C, in the test genome, (C_{ts} - A_{te}), may preserve collinearity but the starting position of block B does not follow the collinearity. If B is moved to the right side of C, (C_{ts} - B_{te}) is of a negative value. In this case, the block A is always located at the left side of B, hence the distance between A and B is ignored. If B is moved to the left side of A, (B_{ts} - A_{te}) becomes negative. A translocation segment is initially identified either insertion or deletion. To distinguish the translocation from either a deletion or an insertion event, the criterion, $Occ_i(s_r) = 1$, has to be checked.

An inter-chromosomal translocation comes with sudden changes in a read identifier such that A and C contain the same read identifier while B does not. Since the read identifier is changed, the positional information is without meaning. For such an event, only the transition from a chromosome to another is an important signal. However, with solely using a population index, it is not possible to retrieve such information. The reference genome against which $S_{r:j-1}$, $S_{r:j}$, and $S_{r:j+1}$ are mapped has to be additionally indexed. Since B is involved in an inter-chromosomal translocation, only the pattern $S_{r:j}$ may manifest a different chromosome identifier. In turn, by observing Chr() function, the algorithm can detect inter-chromosomal translocations. For the same reason as for the intra-chromosomal translocation case, this algorithm is only valid if complete genome sequences are used to build the population index.

4.3 Results

A standard evaluation was performed on the same simulated datasets used in section 3.3.2: 64 *E. coli* and *A. thaliana* datasets. In that section, an exact location was applied to calculate a metric since each label represents a point mutation (SNP). However, in the present section the evaluation is performed for each multi-base SV event. The evaluation scheme checks whether the distance between a ground truth position and the predicted one is within 10 bp. However, an exact prediction for a ground truth location of each event is almost impossible. For instance, some bases at the boundary of a split block will be lost or added to the alignment due to true mutations or idiosyncrasies of alignment algorithms. The definitions of standard evaluation are revised accordingly to address the characteristics of multi-bp SV events.

A true positive (TP_{class}) is reported when a prediction is within 10-bp distance from a ground truth label and the event type is identical to the ground truth. When a prediction introduces either a non-overlapping match or a different type of SV event with respect to the ground truth, it is a false positive (FP_{class}). A false negative (FN_{class}) indicates that a genomic region containing a ground truth SV event is not identified. A true negative (TN_{class}) is a reference-like region correctly rejected as different from the reference by the algorithm. $Sensitivity_{class}$ is the fraction of correct predictions over the ground truths for a certain SV event, TP_{class}/T_{class}. $Precision_{class}$ is the fraction of correct alignments over all the prediction attempts for a certain SV event, TP_{class}/P_{class}. As a note, ground truth locations for a duplication include all repeat DNA segments. For instance, if Occ(s) = 3 and |s| > 50, then all three locations of *s* are included in the ground truth set. Ground truth locations for a translocation entails a deletion and a corresponding insertion.

4.3.1 Deletions

The errors in determining a deletion event are caused mainly by the confusion with a deletion derived from a translocation event or a long broken block stemming from an incorrect alignment and a successive reconstruction procedure. Thus, if one removes the translocations from the evaluation, this

would inflate sensitivity and precision estimates. For larger genomes, a major source of complications in determining deletions are indeed translocations. This is evident in my dataset by comparing the results for *E. coli* and *A. thaliana*, because a prediction of a deletion event is less likely to be confused by most insertion-type events, but by a deletion segment of a true translocation. The best sensitivity and precision were obtained by Apollo with the maximal *k* attempted for *A. thaliana*, *k*=70 (88.7% / 81.2%, Fig. 27), and with *k*=23 for *E. coli* (94.5% / 91.6%, Fig. 26).

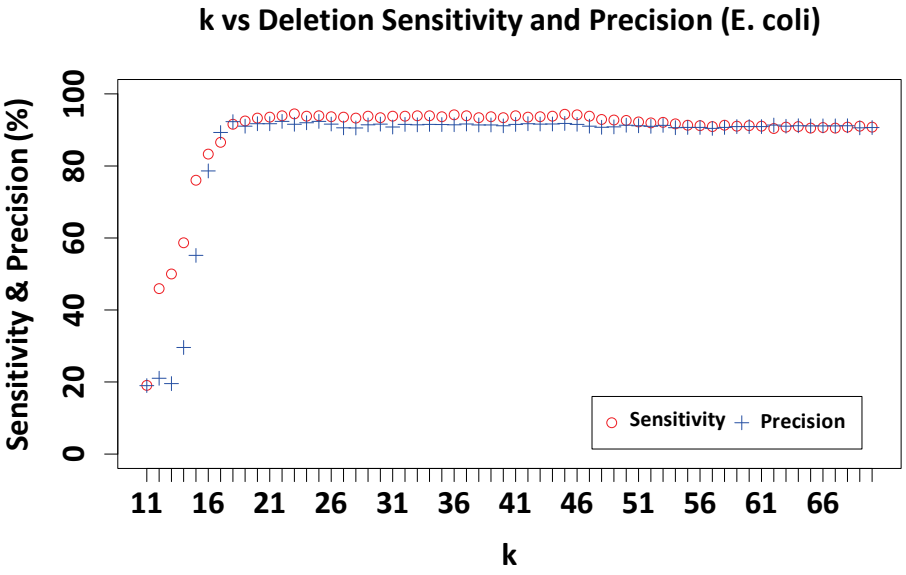

Figure 26. Sensitivity and precision of deletion detection in *E. coli* dataset. Near maximal sensitivity and precision are reached at *k*=18, and are maximal at *k*=23.

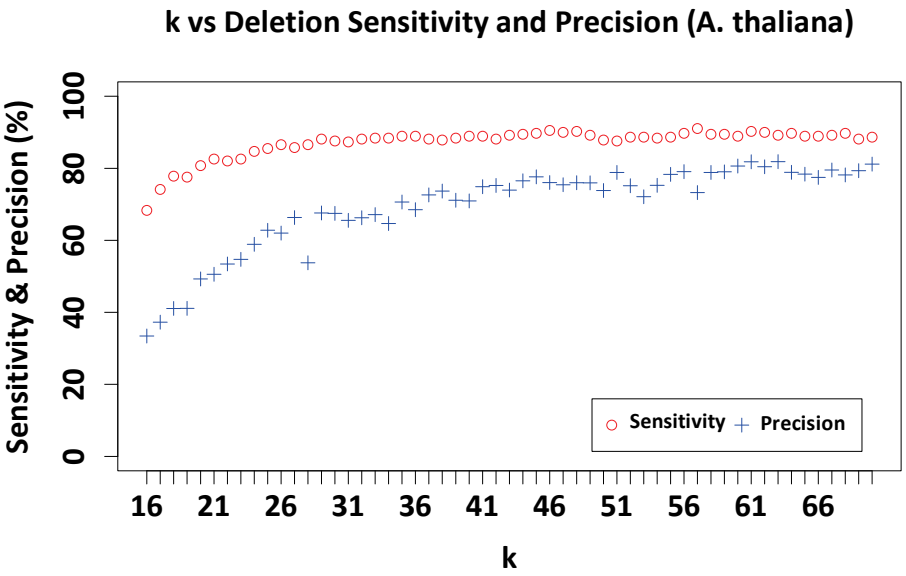

Figure 27. Sensitivity and precision of deletion detection in *A. thaliana* dataset. Compared with the *E. coli* simulation, sensitivity and precision are considerably worse. This reflects that *A. thaliana* genome naturally contains many more repeats. With the highest *k* (=70) value the best performance is achieved.

With $k \geq 18$, the metrics of *E. coli* datasets remained largely stable (Fig. 26). In contrast, both sensitivity and precision with the *A. thaliana* dataset increase with increasing k (Fig. 27). It is likely that greater k lengths support more confident localization of a unique pattern in each collinear block. Efforts to improve sensitivity and precision by refining the procedures for alignments were not successful, since they are derived from incorrect "assignments" to translocations, or sometimes inversions. A deletion often appears as consecutive broken blocks, but the profile of the sequence such as its length or base composition are unknown, making it difficult to identify it with adjacent blocks. Thus, to increase the precision, the deleted sequences should be taken from the reference genome and counted by $Occ_g(s)$. In addition, the fact that the *A. thaliana* reference contains many sizable repeats reduces precision.

4.3.2 Insertions

As discussed, either unbalanced insertions of new sequences or balanced insertions due to a translocation will initially appear as "insertions", and reconstruction from incorrect collinear blocks therefore adversely influences metrics. Obviously, insertions of sequences present in the reference are simpler to detect than insertions of new sequences absent from the reference. The identification of insertions of known sequences can therefore be achieved with high sensitivity. On the other hand, low precision reflects false positives that are actually insertions derived from translocation events or inversions. Especially repeat rich regions cause collectively incorrect predictions of insertions, meaning that two or more individuals in a single collinear block are predicted to have insertion events at exactly the same position or offset by a very short distance. A final contribution to sensitivity and precision comes from the uncertainty of $Occ_r(s) = 0$, which can be misguided by many factors, i.e., multiple SNPs in a very long collinear block, sequences derived from a small indel, or a large inversion. The best sensitivity and precision, 97.3% / 78.9%, were observed with Apollo for *A. thaliana* at k=65 (Fig. 29), and for *E. coli* at k=24 (91.2% / 97.1%, Fig. 28).

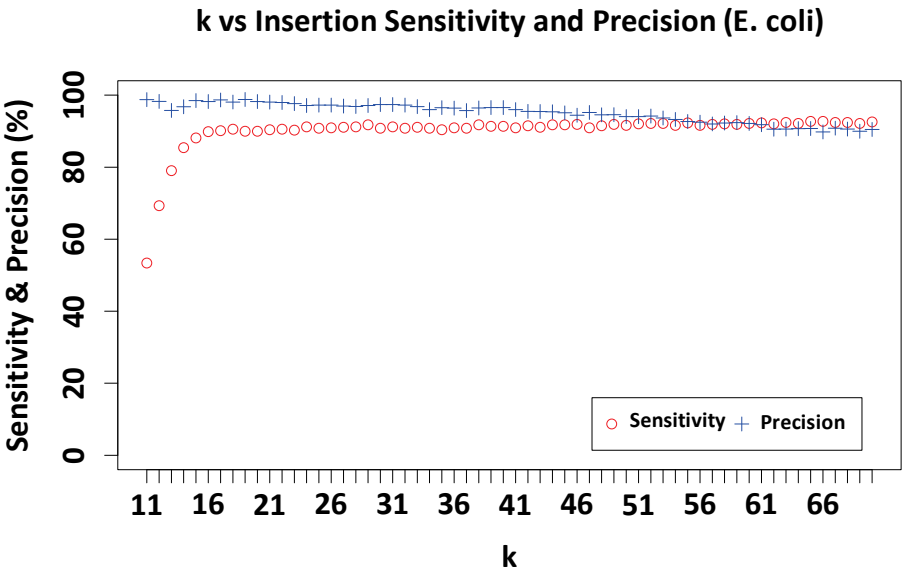

Figure 28. Sensitivity and precision of insertion detection in *E. coli* dataset. Precision is higher with small than with large k values.

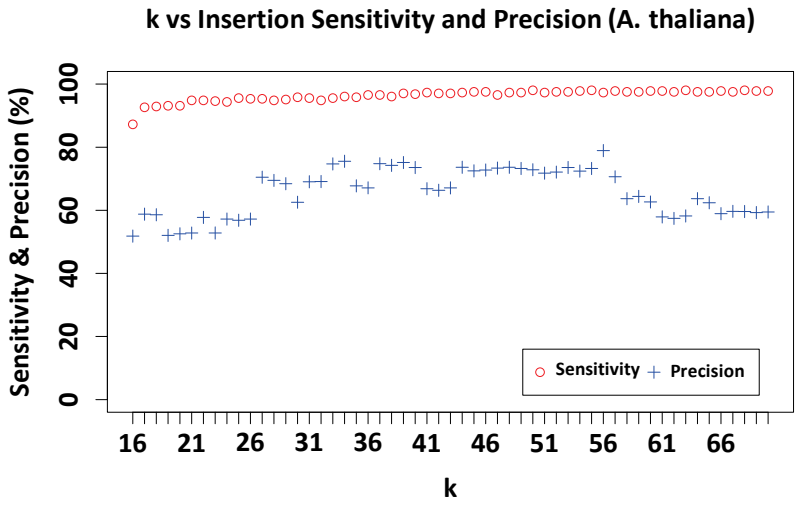

Figure 29. Sensitivity and precision of insertion detection in *A. thaliana* dataset. Precision is not as good as for deletions, most likely because insertions are easily mistaken for translocations and inversions.

4.3.3 Duplications

Duplications are predicted with higher precision than insertions, even though a duplication is an insertion-type event. The major reason is that $Occ_g(s) > 1$ is a strict and reliable criterion. The low precision with small *k*-mer values is most likely due to various types of repeats in the reference. Because a duplicated sequence can appear more than twice, a single false positive diminishes the precision. Based on the experiments, I conclude that the detection of duplications should be performed with a relatively large *k*. The best sensitivity and precision were observed with Apollo for *A. thaliana* at *k*=68 of (93.3% / 87.2%, Fig. 31), and for *E. coli* at *k*=18 of (94.2% / 99.23%, Fig. 30).

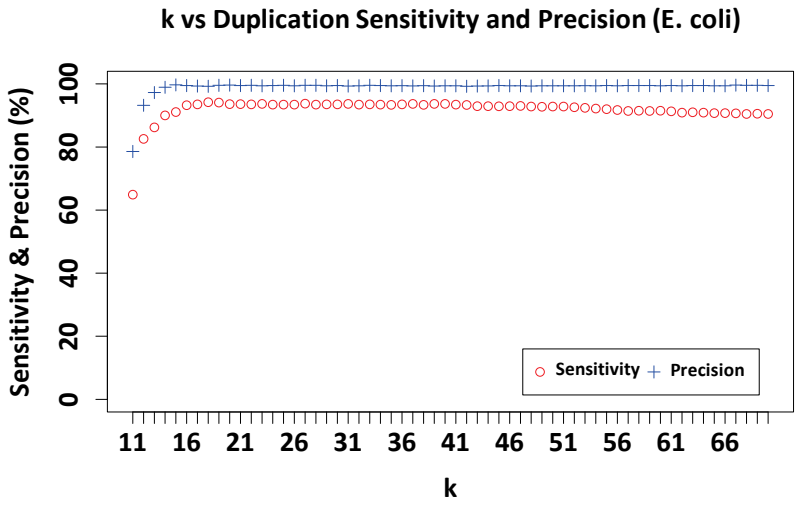

Figure 30. Sensitivity and precision of duplication detection in *E. coli* dataset. Both high sensitivity and precision are already reached at small *k* values.

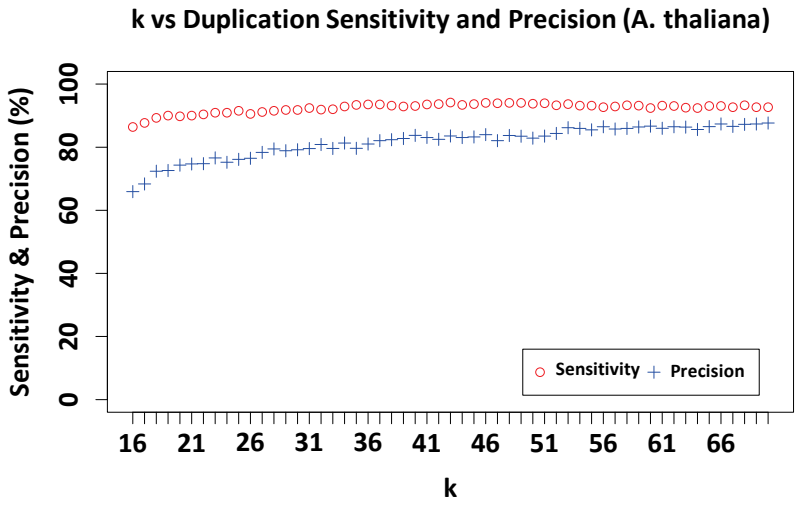

Figure 31. Sensitivity and precision duplication detection in *A. thaliana* dataset. Sensitivity and precision increase with increasing *k*, similar to the deletion experiment.

4.3.4 Inversions

Sensitivity and precision as a function of *k* are similar to what was observed for duplications. Sensitivity is always very high compared to other SV events, likely because one of the split blocks in collinear blocks being in the reverse complementary orientation provides a very clear signal. Misclassification most likely comes from repeats, and incorrect alignments derived from the reconstruction procedure. Since the prediction suffers from a low precision, in practice some heuristic filtering steps would be recommended. The best sensitivity and precision were observed with Apollo for *A. thaliana* at *k*=43 of (98.2% / 87.8%, Fig. 33), and for *E. coli* at *k*=24 of (99.4% / 98.78%, Fig. 32).

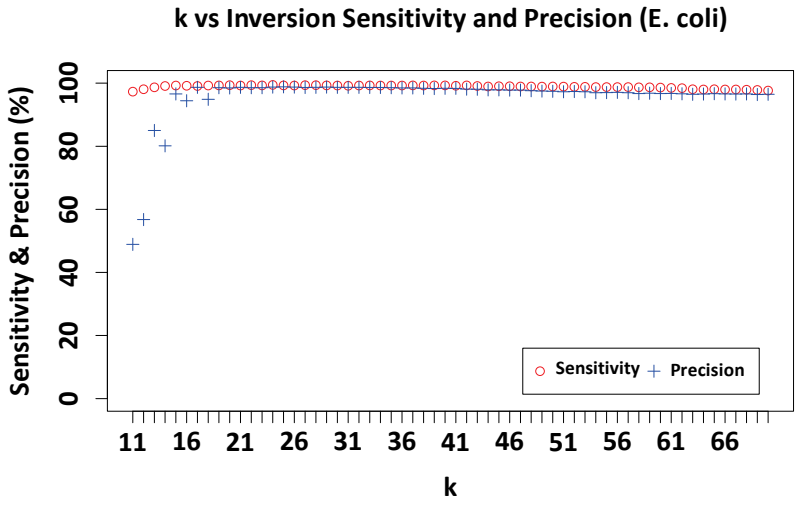

Figure 32. Sensitivity and precision of inversion detection in *E. coli* dataset. Both high sensitivity and precision are already reached at small *k* values.

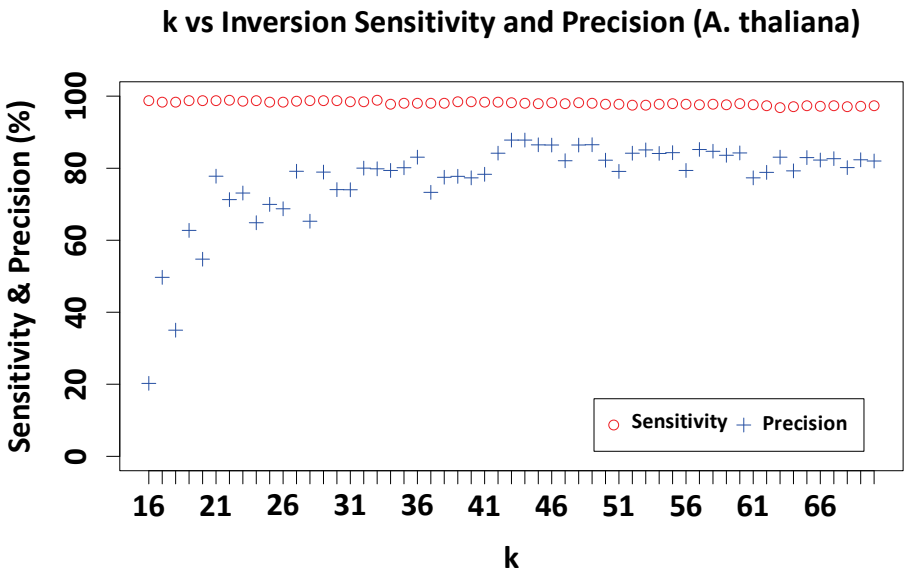

Figure 33. Sensitivity and precision of inversion detection in *A. thaliana* dataset. Precision markedly improves with higher *k* values.

4.3.5 Translocations

The last SV event addressed in this study is the intra-chromosomal translocation. Inter-chromosomal translocations are not tested because of simulation issues, thus from here on translocation means an intra-chromosomal translocation. A translocation is the most difficult event to identify among SV events, because it contains two different events, an insertion and a deletion. Thus, false signals from insertions and deletions reduce the sensitivity of translocation detection. Even with the *E. coli* dataset, the maximal sensitivity was only 81.3% with a precision of 91.8% at k=19 (Fig. 34). With the *A. thaliana* dataset, a sensitivity of 80.2% with a precision of 77.9% was achieved at k=25 (Fig. 35). By comparing the results with those for deletions and insertions (Figs. 26-29), one can infer that true insertions and true deletions indeed degrade the performance of Apollo for translocations.

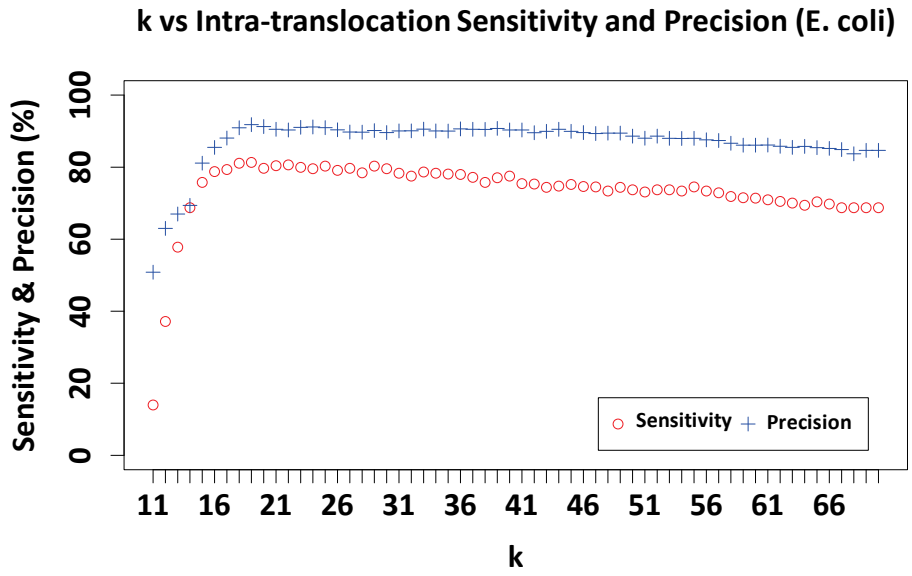

Figure 34. Sensitivity and precision of intra-chromosomal translocation detection in *E. coli* dataset. Overall sensitivity and precision are low compared to other SV types.

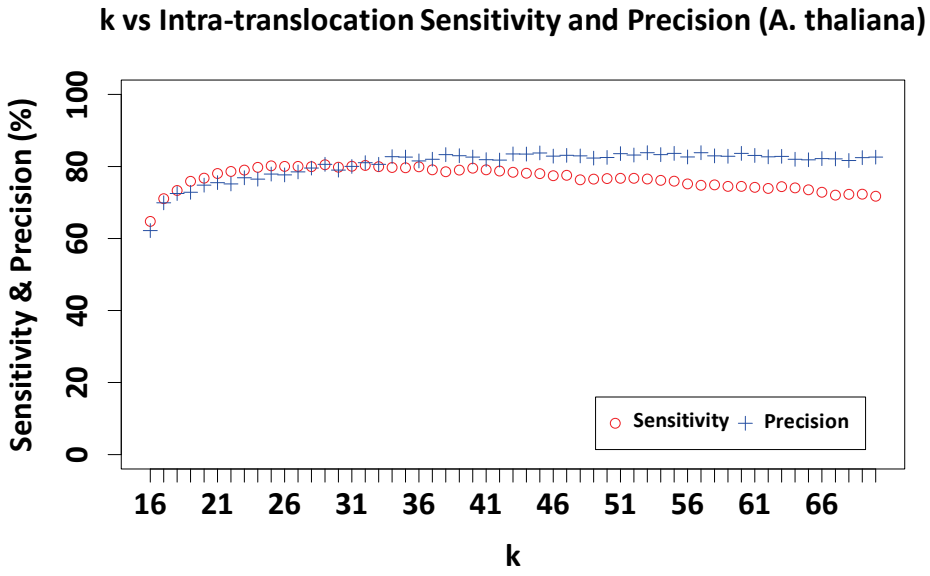

Figure 35. Sensitivity and precision of intra-chromosomal translocation detection in *A. thaliana* datasets. Sensitivity and precision are similarly low at different *k* values.

4.4 Discussion

Genomes contain often highly variable repetitive DNA segments, which causes ambiguities in short read alignments. The uncertainty in alignment propagates to the downstream analyses. In fact, a short read alignment lacking the ability to traverse a colored genome graph cannot distinguish differ-

ent repeat regions [H. Li, and R. Durbin, 2009, B. Langmead, and S.L. Salzberg, 2012]. The only practical and widely employed solution that reduces the ambiguity has been to mask the repeat regions in a reference genome sequence and exclude alignments to these regions. In turn, much of the genome of repeat-rich organisms cannot be accurately characterized by short read-only methods. The advantages of long sequence alignments are obvious, since they more often can generate correct alignments because they more often extend across repeats.

Detection algorithms for variants ranging from point mutations to large SVs are highly dependent on the accuracy of primary sequence alignments. Short read alignments intrinsically have a higher chance of multiple hits than long read alignments. Given an accurate long read alignment, a variant caller is less likely to yield spurious predictions of mutation events. With the advent of long read third generation sequencing technology such as PacBio [M.J. Levene et al., 2003] and Oxford Nanopore [J.J. Kasianowicz et al., 1996], of the four signals that are exploited for detecting SVs with short reads – assembly, split read, PEM, and read depth –, only assembly and split reads will continue to be important. When a complete genome can be sequenced from a single library, then only split read signal may be used. Due to the long-sequence matches, differences among variant calling algorithms will hopefully be significantly reduced, making the use of integrative or ensemble methods less compelling. Note that it is not possible to compare Apollo with conventional SV callers since they are designed to deal with short read alignments.

Apollo predicts a SV event based on the Kairos alignment, thus naturally uses population information. Conventional SV calling algorithms often have not exploited population information [K. Chen et al., 2009, K. Ye et al., 2009, I. Hajirasouliha et al., 2010, A. Abyzov, and M. Gerstein, 2011a]. Reads from a single individual may be sensitively characterized by definition of each SV, but the accuracy of calling is low when an SV is predicted based on low-coverage reads. Moreover, there is no simple way to improve precision if only a single individual is mapped against a reference genome. When a population-scale SV analysis is performed, the sample size for each allele is increased, leading to more accurate predictions [R.E. Handsaker et al, 2011]. In addition, the sequences of other individuals in a population help to accurately locate the breakpoints because the aligned sequences can be compared base-by-base. With a population-aware method, rare SV events can be identified. In addition, thanks to Kairos' inverse mapping strategy, Apollo does not require time-consuming masking, alignment sorting, and merging procedures.

Finally, I note that large SV events disrupt collinearity much more than multiple small mutations. The correct identification of a specific SV event requires a consecutive checking of partial patterns in terms of the occurrence in each individual. Hence, to improve the detection accuracy, more efforts should be made before building a population index such that it is needed to prepare more complete and longer sequences. The contiguity and correctness of genomic sequences can be improved by the advances in sequencing technology and bioinformatics algorithms in terms of genome assembly and sequencing error corrections.

4.5 Conclusion

Structural variants can have drastic phenotypic effects. The detection of SV events based on short read alignments is well known to be highly variable, based on the specific algorithm employed [M. Mohiyuddin et al., 2015]. I have introduced a SV calling algorithm, Apollo, based on long sequence alignments obtained from another algorithm I have developed, Kairos. Given complete genomes and precise genetic information stored in a population index, the combination of Kairos and Apollo could detect different types of SVs without applying a complex computational pipeline that involves tools

not optimized for each other in conjunction with arbitrary heuristics. Apollo captures a wide variety of SVs among individuals in a population of related individuals. Since many biological studies are based on signals of mutations occurring at different allele frequencies, the combination of Kairos and Apollo algorithms should fill an important lacuna in the spectrum of tools currently available. Providing accurate information about SVs at nucleotide resolution will lead to better insights into genome evolution and ultimately changes in phenotypic traits.

4.6 References

A. Abyzov, and M. Gerstein (2011a) AGE: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision, Bioinformatics, 27(5): 595-603.

A. Abyzov, A.E. Urban, M. Snyder, and M. Gerstein (2011b) CNVnator: An approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing, Genome Res., 21(6):974-84.

P.J. Campbell et al. (2008) Identification of somatically acquired rearrangements in cancer using genome-wide massively parallel paired-end sequencing, Nat Genet., 40(6):722-9.

K. Chen et al. (2009) BreakDancer: an algorithm for high-resolution mapping of genomic structural variation, Nat Methods, 6(9):677-81.

D.Y. Chiang, G. Getz, D.B. Jaffe, M.J. O'Kelly, X. Zhao, S.L. Carter, C. Russ, C. Nusbaum, M. Meyerson, and E.S. Lander (2009) High-resolution mapping of copy-number alterations with massively parallel sequencing, Nat Methods, 6(1):99-103.

E.E. Eichler (2001) Recent duplication, domain accretion and the dynamic mutation of the human genome, Trends Genet., 17(11):661-9.

L. Feuk, A.R. Carson, and S.W. Scherer (2006) Structural variation in the human genome, Nature Reviews Genetics 7:85-97.

J.L. Freeman et al. (2006) Copy number variation: new insights in genome diversity, Genome Res., 16(8):949-61.

I. Hajirasouliha, F. Hormozdiari, C. Alkan, J.M. Kidd, I. Birol, E.E. Eichler, and S.C. Sahinalp (2010) Detection and characterization of novel sequence insertions using paired-end next-generation sequencing, Bioinformatics, 26(10):1277-1283.

R.E. Handsaker, J.M. Korn, J. Nemesh, and S.A. McCarroll (2011) Discovery and genotyping of genome structural polymorphism by sequencing on a population scale, Nat Genet., 43(3):269-76

F. Hormozdiari, C. Alkan, E.E. Eichler, and S.C. Sahinalp (2009) Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes, Genome Res., 19(7):1270-8.

A.J. Iafrate, L. Feuk, M.N. Rivera, M.L. Listewnik, P.K. Donahoe, Y. Qi, S.W. Scherer, and C. Lee (2004) Detection of large-scale variation in the human genome. Nat Genet., 36(9):949-51.

International HapMap Consortium (2003) The International HapMap Project, Nature, 426(6968):789-96.

International HapMap Consortium (2005) A haplotype map of the human genome, Nature, 437:1299-1320.

International SNP Map Working Group (2001) A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms, Nature, 409(6822):928-33.

A. Kallioniemi, O.P. Kallioniemi, D. Sudar, D. Rutovitz, J.W. Gray, F. Waldman, and D. Pinkel, (1992) Comparative genomic hybridization for molecular cytogenetic analysis of solid tumors, Science, 258(5083):818-21.

J.J. Kasianowicz, E. Brandin, D. Branton, and D.W. Deamer (1996) Characterization of individual polynucleotide molecules using a membrane channel, Proc Natl Acad Sci USA, 93(24): 13770–13773.

J.O. Korbel et al. (2007) Paired-end mapping reveals extensive structural variation in the human genome, Science. 318(5849):420-6.

G. Landan, and D. Graur (2007) Heads or tails: a simple reliability check for multiple sequence alignments, Mol Biol Evol., 24(6):1380-3.
B. Langmead, and S.L. Salzberg (2012) Fast gapped-read alignment with Bowtie 2. Nature Methods, 9:357–359.
R.M. Layer, C. Chiang, A.R. Quinlan, and I.M. Hall (2014) LUMPY: a probabilistic framework for structural variant discovery, Genome Biol., 15(6):R84.
S. Lee, E. Chera, and M. Brudno (2008) A robust framework for detecting structural variations in a genome, Bioinformatics. 24(13):i59–i67.
S. Lee, F. Hormozdiari, C. Alkan, and M. Brudno (2009) MoDIL: detecting small indels from clone-end sequencing with mixtures of distributions, Nat Methods., 6(7):473-4.
M.J. Levene, J. Korlach, S.W. Turner, M. Foquet, H.G. Craighead, and W.W. Webb (2003) Zero-mode waveguides for single-molecule analysis at high concentrations, Science, 299(5607):682-6.
H. Li, and R. Durbin (2009) Fast and accurate short read alignment with Burrows-Wheeler trans-form, Bioinformatics, 25(14):1754-60.
K. Lin, S. Smit, G. Bonnema, G. Sanchez-Perez, and D. de Ridder (2015) Making the difference: integrating structural variation detection tools, Brief Bioinform. 16(5):852-64.
M. Margulies et al. (2005) Genome sequencing in microfabricated high-density picolitre reactors, Nature, 437(7057):376-80.
R.E. Mills, C.T. Luttig, C.E. Larkins, A. Beauchamp, C. Tsui, W.S. Pittard, S.E. Devine (2006) An initial map of insertion and deletion (INDEL) variation in the human genome, Genome Res., 16(9):1182-90.
M. Mohiyuddin, J.C. Mu, J. Li, N.B. Asadi, M.B. Gerstein, A. Abyzov, W.H. Wong, and H.Y.K. Lam (2015) MetaSV: an accurate and integrative structural-variant caller for next generation sequencing, Bioinformatics, 31(16):2741-4
D. Pinkel et al. (1998) High resolution analysis of DNA copy number variation using comparative genomic hybridization to microarrays, Nat Genet., 20(2):207-11.
A.R. Quinlan, R.A. Clark, S. Sokolova, M.L. Leibowitz, Y. Zhang, M.E. Hurles, J.C. Mell, and I.M. Hall (2010) Genome-wide mapping and assembly of structural variant breakpoints in the mouse genome, Genome Res., 20(5):623-35.
T. Rausch, T. Zichner, A. Schlattl, A.M. Stütz, V. Benes, and J.O. Korbe (2012) DELLY: structural variant discovery by integrated paired-end and split-read analysis, Bioinformatics, 28(18):i333-i339.
R. Redon, et al. (2006) Global variation in copy number in the human genome, Nature. 444(7118): 444-54.
J. Sebat et al. (2004) Large-scale copy number polymorphism in the human genome, Science, 305(5683):525-8.
A.J. Sharp et al. (2005) Segmental duplications and copy-number variation in the human genome. Am J Hum Genet., 77(1):78-88.
S. Sindi, E. Helman, A. Bashir, and B.J. Raphael (2009) A geometric approach for classification and comparison of structural variants, Bioinformatics, 25(12):i222-30.
S. Solinas-Toldo, S. Lampel, S. Stilgenbauer, J. Nickolenko, A. Benner, H, Döhner, T. Cremer, and P. Lichter (1997) Matrix-based comparative genomic hybridization: biochips to screen for genomic imbalances, Genes Chromosomes Cancer, 20(4):399-407.
The 1000 Genomes Project Consortium (2012) An integrated map of genetic variation from 1,092 human genomes, Nature, 491(7422):56-65.
E. Tuzun et al. (2005) Fine-scale structural variation of the human genome, Nat Genet., 37(7):727-32.
K. Ye, M.H. Schulz, Q. Long, R. Apweiler, and Z. Ning (2009) Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads, Bioinformatics, 25(21):2865–2871.

Chapter 5

5. Metagenomic taxonomy classifier

In this chapter, I describe a project partially presented at the Cold Spring Harbor Laboratory meeting on Genome Informatics [E.C. Lim, 2015]. I reproduce some materials courtesy of Cold Spring Harbor Laboratory Archives. New York.

5.1 Introduction

The largest group of organisms on the earth are the microbes: their total number is estimated to be on the order of 10^{30} [J. Kallmeyer, 2012]. They can obtain energy from a wide range of sources, including inorganic molecules, and even directly use electricity, and can be found in harsh environments such as the icefields of Antarctica, hydrothermal vents at the bottom of the ocean, and extremely acidic lakes. They also dwell in the human body and their roles in health and disease are being studied in the Human Microbiome Project. The imbalance between beneficial and harmful microbes may cause a variety of disease. For example, human gut bacteria may be critical for colorectal cancer [J. Ahn et al., 2013, S. DeWeerdt, 2015]. Other diseases are caused by virus infections. SARS [X.Y. Ge et al., 2013], MERS coronavirus [R.J. de Groot et al., 2013] and Ebola virus have caused major public health concerns due to their high fatality rates [S. Baize et al., 2014]. An accurate and prompt determination of viral sequences would thus be very useful as well [S. Baize, 2014, Y.G. Tong et al., 2015].

The collective microbial community in a sample is called the microbiome; its collected genome is called a metagenome. Advances in next generation sequencing (NGS) technology have greatly advanced metagenomics, which is rapidly moving from the quantitative estimation of species in a sample to questions of biological activities and functions supported by the microbiome in a sample. Initially, microbiome studies relied exclusively on sequencing of 16S rDNA amplicons; more recently, these approaches are surpassed by whole-genome shotgun (WGS) sequencing approaches [N. Shah et al, 2011]. The accuracy of the abundance estimation of species is highly dependent on the accurate classification of sequence reads. Thousands of microbes have been cultured and their genomes have been completely sequenced, but the majority of species remain uncultured and unsequenced [M. Albertsen et al., 2013, D.R. Garza, and B.E. Dutilh, 2015].

A taxonomy classification is a hierarchical multi-label classification problem, meaning that assignments at a specific taxon can have multiple labels, with the selected taxon including lower rank statistics. An unknown species may be classified to a certain genus by identifying homologous sequences. This problem will become less difficult with long read technology, since complete genome sequences will be far easier to obtain than before. This trend will definitely improve the accuracy of taxonomy classifications, and abundance estimates, which are useful to correlate the functional characteristics with the environments given.

An important aspect in the taxonomic classification problem is binning, a process to assign discrete values in a certain interval derived from a discretization of continuous data. In metagenomics, binning is a process of classification for each read to a bin of reads or the clustering of reads. Sequence similarity is a major criterion to classify reads; this can be achieved with a supervised learning method that defines a transfer function expressing a model of complete reference genomes. If a read comes from an unknown organism, the sequence similarity of the read could not exceed a threshold value of either probability or similarity, leading to a random and incorrect species assignment [S.S. Mande et al., 2012]. However, for a higher rank, i.e., genus, rather than species, the assignment may be more

successful since the chance of finding similar sequences at a higher taxonomic level is always higher. The Lowest Common Ancestor (LCA) method, which reports the most viable predictions given database or statistical model, is a popular method to deal with unknown species. The high number of genomes in the database is a positive factor to improve the classification accuracy but it comes with a burden for large space requirements [W. Gerlach, and J. Stoye, 2011, K. Ueno et al., 2014].

Methods for classification of sequence reads can be divided into two categories: composition- and similarity-based. The first applies machine learning or statistics while the second classifies based on the sequence alignments. Recently, *k*-mer based classification, which is a hybrid method, has become more popular due to its time-efficiency. Kraken builds a database mapping *k*-mers to the LCA before classification [D.E. Wood, and S.L. Salzberg, 2014]. The database is retrieved to count the number of *k*-mers in a taxonomy subtree where each *k*-mer frequency represents the weight. The leaf node with the maximum weight path is the class to which a read is assigned. CLARK utilizes a similar *k*-mer counting as Kraken to build a database [R. Ounit, 2015], but instead of using all the *k*-mers, only discriminative *k*-mers, which contain unique *k*-mers in a rank, are considered, reducing space complexity.

Despite the high efficiency of *k*-mer based algorithms, a major drawback is their uneven sensitivity, with small *k*, e.g, below 17, increasing sensitivity at the expense of precision. Optimal *k* is determined before building a database and cannot be changed afterwards. One could consider constructing multiple databases with different *k*, but such databases would be difficult to maintain due to spatio-temporal inefficiency. Notably, because the optimal value of *k* depends on the input sequences, different taxa in a metagenomic sample will be classified with different accuracy. This is even worse in practice, where empirical values are used. Moreover, sequencing errors can significantly influence accuracy of classification. Finally, classification at species resolution is difficult because *k*-mer methods perform classification on top of exact matches, and a single substitution due to sequencing error increases the probability of mis-classification to a different species.

The aim of the taxonomy classifier that I present here is to obtain higher sensitivity and precision, better time-efficiency, but with less space requirements in assigning taxa than with existing classifiers. These computational benefits are naturally obtained by the use of the population index explained in Chapter 1 over fixed *k*-mer data structures. The memory space for each *k*-mer is reduced due to the compressibility of the FM-index. The length of patterns can be determined in running time rather than be assigned as a fixed parameter, leading to a better sensitivity compared with fixed *k*-mer based methods. The highly sensitive and memory-efficient metagenomics taxonomy classifier I introduce here is Poseidon.

5.2 Method

A population index provides basic string operations on the FM-index (see section 1.4.2). Further information for each text are accessible from a population index through an inverse sentinel array depending on the application. For the taxonomy classification, any sequence patterns should turn into taxonomy identifiers encapsulating ranks and relevant scientific names, i.e., species - *Arabidopsis thaliana*, and genus - *Arabidopsis*. A distinctive feature of Poseidon is the use of the population index, thereby overcoming some of the limitations of conventional methods due to the use of a fixed parameter *k*. The length of *k*-mers can be changed depending on the context to maximize sensitivity. These dynamic changes in parameter *k* are achieved by backtracking. A read is segmented into partial patterns by checking the frequency of each pattern, implementing split-read alignments. The length of *k*-mers is summed up denoting a cumulative match for each species. The longest *k* with the relevant tax-

onomy identifiers is also kept. The taxonomy identifiers of the longest cumulative matches may highly likely represent the true organisms. If the longest *k* is longer than half the read length, the relevant taxonomy identifier would replace the predictions made by the normal cumulative matches (Algorithm ShortReadClassification).

Definition

k is a variable representing the length of a *k*-mer that found in the population index at least once.

Occ() is the function that returns the occurrence of a pattern.

P is the current pattern.

m is the minimum *k* that user specifies.

H is a map that contains suffix ids as keys, and the lengths of cumulative matches as values.

M is a pair that contains the longest *k* and a set of suffix ids.

Algorithm ShortReadClassification

Input: a read R
Output: suffix ids representing taxonomy identifiers
FOR all positions IN R:
IF m ≤ k:
    Backtrack a sequence from the last base until Occ(P) ≤ 1
Adds k to the values of H[suffix ids]
Replaces M with the current k and suffix ids if k is longer than the one in previous M
Decreases the current position by k
Update the last base to the current position
  ELSE
Decreases the current position by 1
IF k in M is longer than half of the read:
  RETURN the set of suffix ids of length k
ELSE
  RETURN the set of suffix ids having the longest cumulative matches

5.3 Evaluation

5.3.1 Preparation

Let *T* be a set of ground truth assignments that an evaluation method recognizes. Assume that a database contains *Escherichia coli*, but not *Proteus vulgaris*, and 10,000 reads for each species are simulated from the reference genomes. When one evaluates the accuracy of classifiers, *T* is, for example, 10,000 and not 20,000, since *Proteus vulgaris* cannot be recognized by the evaluation scheme. For Kraken, *T* is defined as the number of total assignments at a certain rank. This definition can cause confusion when interpreting the results. For instance, when the sensitivity of different algorithms is comparable altogether, one may consider the absolute number of total assignments. However, if some assignments are not recognized by the evaluation scripts, the results reflect biased metrics. Thus, I posit that my definition of *T* yields more reliable estimates of the true accuracy.

A homologous sequence is a partially identical sequence occurring in different species, e.g., from closely related species, or from an LGT event, but is longer than the minimum pattern length. Let *P* be the number of all the prediction attempts, which is divided into true positives (TPs) and false positives (FPs). A TP is defined as a case where the prediction equals the ground truth label; an FP is the opposite. In the evaluation of taxonomy classifiers, FPs at species resolution can be unreliable due to confounding homologous sequences, and therefore metrics dependent on FPs are not calculated. The overall or average sensitivity is TP/*T*, and sensitivity for each class can be defined as TP_{class}/T_{class}. The overall precision is TP/*P*. *P* deviates from *T* depending on the types of classifiers. *P* is smaller or equal to *T* for a multi-class classifier (*P*=*T*-unclassified) while it can be either smaller or greater than *T* for a multi-label classifier.

The presence of homologous sequences influence the precision of a multi-label classifier at species level or strain level. For instance, if one simulates reads from conserved regions, the classification procedure of a multi-label classifier would yield multiple assignments for a single read but it will include the ground truth label with high probability. Nonetheless, the classification of a multi-class classifier would generate a single label for a read. This outlines that the precision of a multi-class classifier at species level is inherently higher than the one of a multi-label classifier. When there is a homologous sequence, Poseidon yields multiple labels for the read.

To evaluate multi-label classifiers, ground truth homologous sequences, which are partially identical sequences occurring in different species, should be marked with all the available labels, the taxonomy identifiers, at a given rank. For example, when a homologous sequence A is common in species B and C, the ground truth assignments for the read are two taxonomy identifiers of B and C. In practice, such assignments are infeasible since the number of simulated reads is too large. In this study, complete genomes of all bacteria, viruses, fungi, and draft assemblies of bacteria obtained from NCBI on 10, Jul, 2015 were used to simulate 1,000 reads for each species (Table 17, 18). One-to-one assignments were assumed so that multi-class algorithms such as Kaken, and CLARK could be included in the evaluation.

Simulations were carried with Mason [M. Holtgrewe, 2010], and ART [W. Huang et al., 2012] with different error rates on a dataset of 8,294 species. The parameters used here for Mason are the same as used for Mason in the Kraken paper [D.E. Wood, and S.L. Salzberg, 2014]. Mapping read identifiers to taxonomy identifiers is only necessary at species resolution since higher ranks are easily accessible in the taxonomy tree. Performance was evaluated at species and genus level except for some virus reads that lack genus identifiers. The direct parent ranks of undefined cases are included for fair comparisons.

The HiSeq_accuracy dataset contains 10 genera and species. A clade-exclusion experiment is performed on the MiSeq dataset such that *Proteus vulgaris* is excluded while *P. mirabilis* and *P. penneri* are included in the population index. The "recognizable (by the evaluation scheme)" labels in the Miseq dataset are 10 genera, and 9 species because of the exclusion of *P. vulgaris*. The simBA5 dataset consists of randomly sampled genomes with varying coverage depth. Though 615 genera and 1202 species are labeled, many species merely have a single read, which eventually will be assigned to a ground truth species (Table 16). Some draft assemblies of bacteria fail to produce simulated reads because contig lengths are too short, under 100 bp. For instance, I excluded Cont154.1-3, Cont154.5, and Cont154.9 in the Streptococcus_anginosus_F0211_uid61277 assembly. After simulating MiSeq reads, the number of reads was sometimes insufficient, which is expected to be 1,000, then the simulation was rerun with the read length parameter value of 100 bp instead of 250 bp. Even with this change, still some reads did not reach at the expected number, thus they are compensated by oversampling. ART is capable of simulating more reads than Mason, but it has the drawback that the number of reads is sometimes unpredictable, generating few millions of reads despite given 1,000-read limit. Thus, a fixed binary of ART is applied.

Table 16. Datasets. Datasets according to Kraken paper **[D.E. Wood, and S.L. Salzberg, 2014]**.

Dataset	Simulator	# Genus	# Species	# reads
HiSeq	Mason	10	10	10,000
MiSeq	Mason	10	10	10,000
simBA5	Mason	615	1202	10,000

Table 17. The number of labels in datasets simulated with Mason. The source data are from NCBI on 10, Jul, 2015.

Name*	Genera	Species	# reads
Bacteria	716	1518	2,785,000
Bacteria DRAFT (HS20)	732	2432	6,883,000
Bacteria DRAFT (MS)	707	2240	6,030,000
Viruses (HS20)	519	4303	4,400,000
Viruses (MS)	519	4301	4,397,000
Fungi	21	26	26,000

*HS20 denotes HiSeq2000 reads, MS for MiSeq.

Table 18. The number of labels in datasets simulated with ART. The source data are from NCBI on 10, Jul, 2015.

Name*	# Genus	# Species	# reads
Bacteria	716	1518	2,785,000
Bacteria DRAFT (HS20)	735	2447	6,929,444
Bacteria DRAFT (MS)	735	2447	7,172,051
Viruses (HS20)	519	4303	4,400,000
Viruses (MS)	519	4301	4,404,000
Fungi	21	26	26,000

*HS20 denotes HiSeq2000 reads, MS for MiSeq.

5.3.2 Results

The evaluation was performed for the simulated datasets with an AMD-Opteron 64-core processor machine and 512 Gb physical memory in a Linux environment. The performance of Poseidon was compared to Kraken [D.E. Wood, and S.L. Salzberg, 2014] and CLARK [R. Ounit, 2015] with respect to sensitivity, precision, memory consumption, and runtime. The memory consumption of Kraken and CLARK grew as the length of *k*-mer increased, while it more uniform for Poseidon (Fig. 36). For the database including all bacteria draft assemblies, Poseidon was 8.2 times more efficient than Kraken and 3.2 times more than CLARK at k=31. For the database comprising only complete bacteria genomes, Poseidon was 9.4 times more effective than Kraken and 6.4 times more than CLARK at k=31. The actual memory consumption of Poseidon for the population index was nearly constant at 20 Gb independently of k.

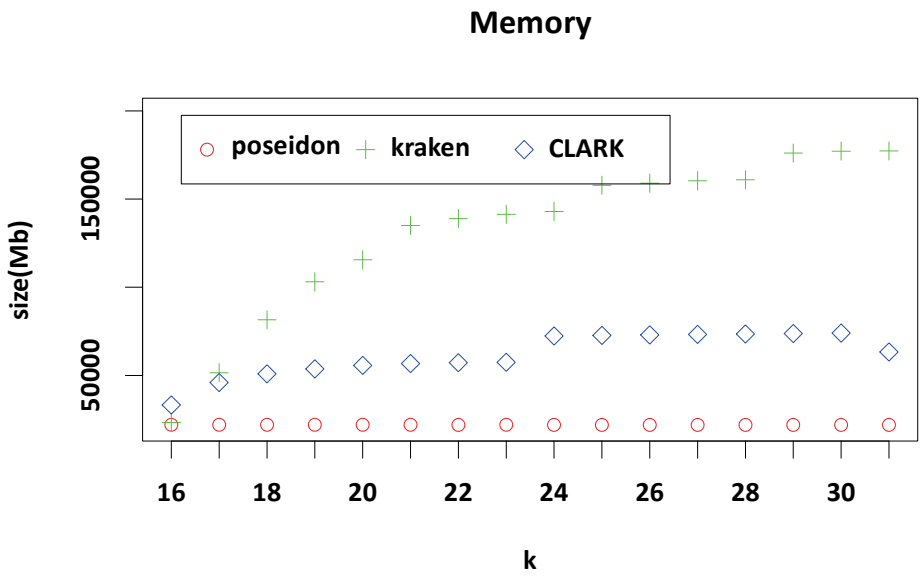

Figure 36. Memory consumption. Bacteria draft genomes are used to build the indices.

The assignment speed of Poseidon and Kraken, both of which were a third slower than CLARK (Fig. 37), but the overall runtime of CLARK was much slower because of inefficiency in database loading (Fig. 38). The results shown were obtained on draft assemblies of bacteria (HS20) data.

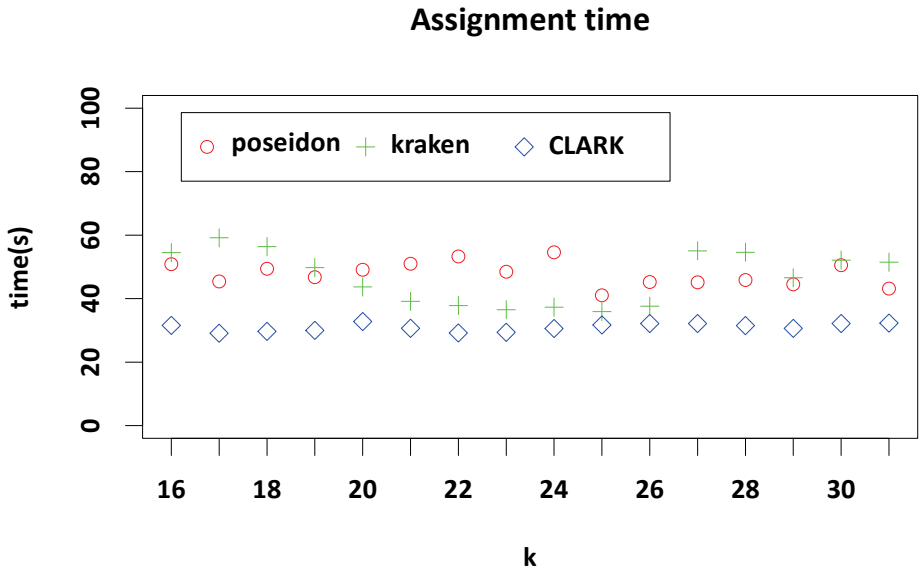

Figure 37 Assignment speed. This indicates the total amount of time for assignments.

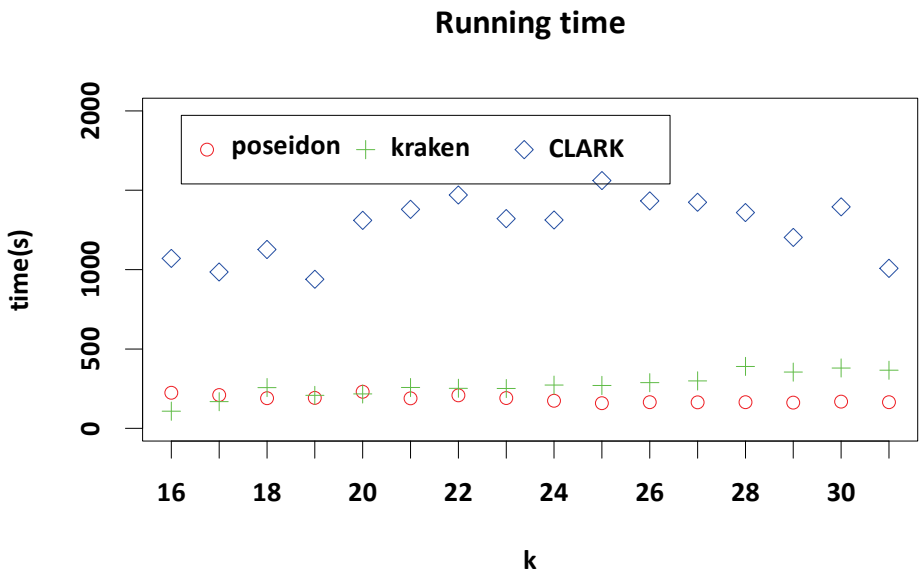

Figure 38 Running time. Running time is in practice more important than assignment time.

At both the species and genus level, Poseidon outperformed Kraken and CLARK in terms of sensitivity, most notably at the species level (Fig. 39, 40). Precision of Poseidon was generally lower at the species level (Fig. 41), but comparable to the other two at the genus level (Fig. 42). Notably, sensitivity of Poseidon was constantly high, from k=16 to k=31 (Fig. 39, 40). Thus, Poseidon provides an excellent compromise between sensitivity and precision.

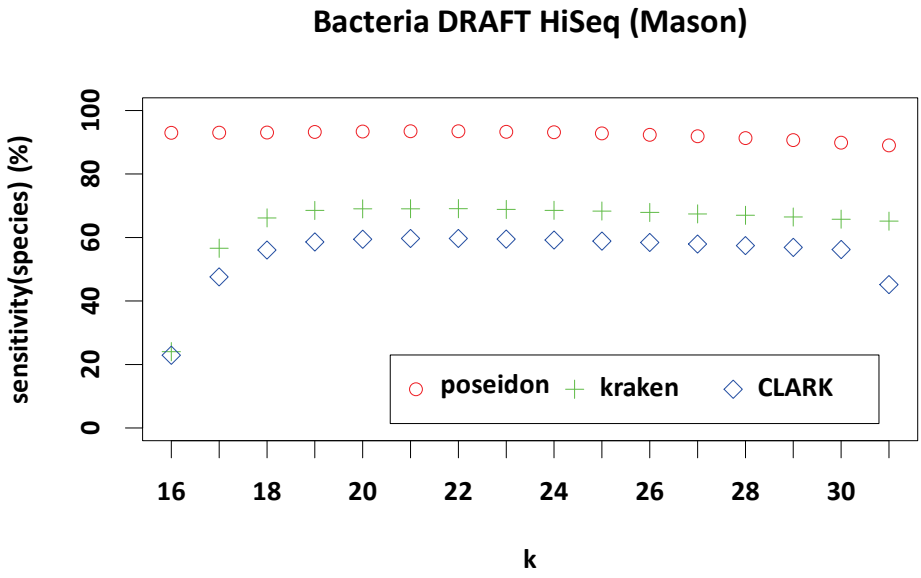

Figure 39 Sensitivity of bacteria draft assembly classification simulated by Mason with the HiSeq profile at species level. Poseidon achieved the best species level sensitivity. The experiment on the dataset simulated by ART did render almost identical results with the one simulated by Mason, thus it is not shown.

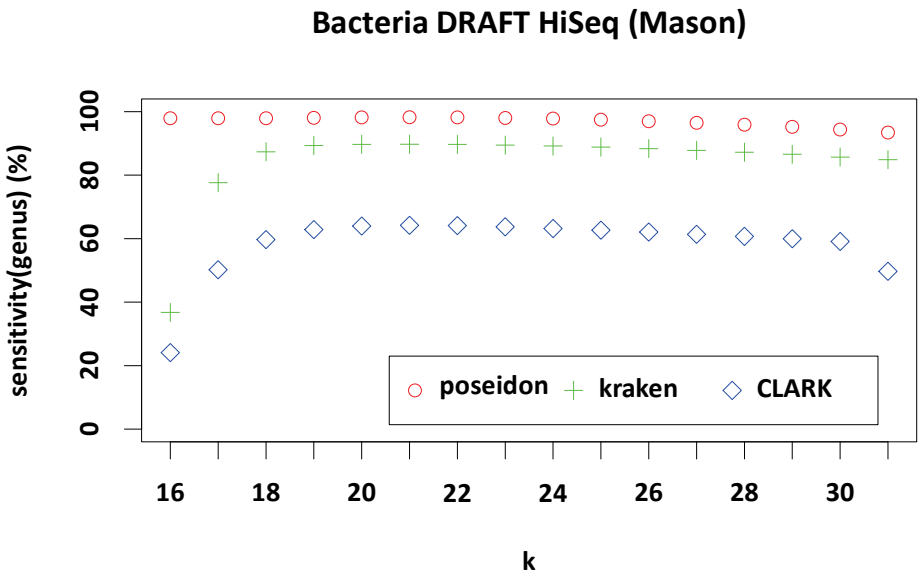

Figure 40 Sensitivity of bacteria draft assembly classification simulated by Mason with the HiSeq profile at genus level. Poseidon achieved the best genus level sensitivity. The experiment on the dataset simulated by ART did render almost identical results with the one simulated by Mason, thus it is not shown.

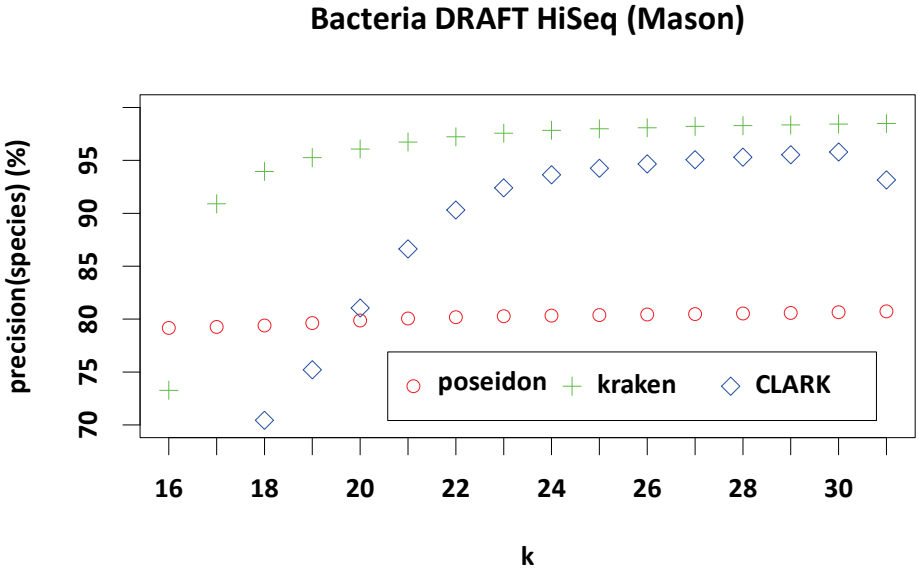

Figure 41 Precision of bacteria draft assembly classification simulated by Mason with the HiSeq profile at species level. Poseidon obtained at lower precision than the other two algorithms. The experiment on the dataset simulated by ART did render almost identical results with the one simulated by Mason, thus it is not shown.

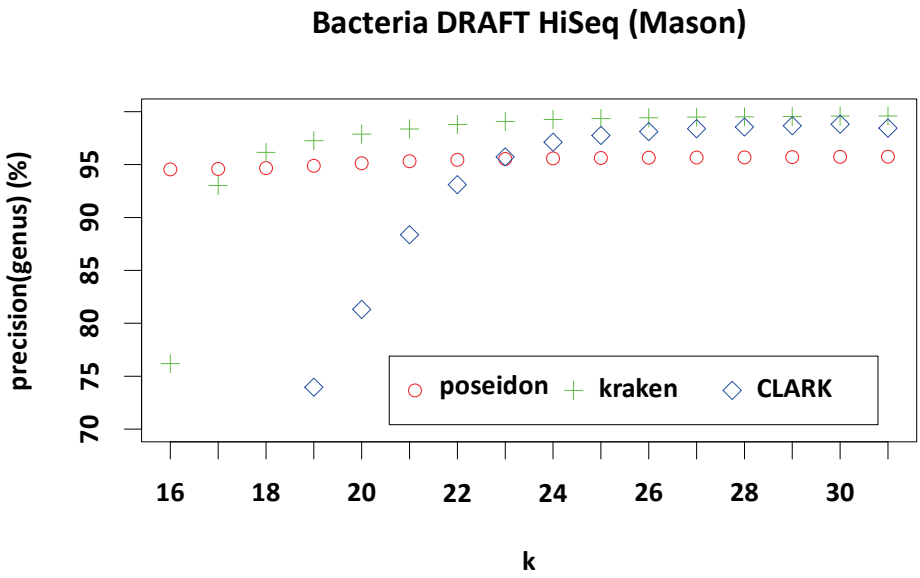

Figure 42 Precision of bacteria draft assembly classification simulated by Mason with the HiSeq profile at genus level. Poseidon obtained at lower precision than the other two algorithms. The experiment on the dataset simulated by ART did render almost identical results with the one simulated by Mason, thus it is not shown.

Poseidon predicted the *Proteus* genus with sensitivity of 41.1% and precision of 99.5157% in the clade exclusion experiment. Though Kraken and CLARK achieved precision of 100%, sensitivity of both were much lower in this experiment, 24.5% for Kraken and 25.1% for CLARK (no supporting figures). The average genus sensitivity of Poseidon was above 90% (Fig. 40); this large sensitivity gap between a regular and clade exclusion experiment is indicative of how the inclusion of additional species in the index improves sensitivity.

Sensitivity and precision were similar for datasets simulated with either Mason or ART, except for fungi, where Kraken produced 11% more mis-classifications in the Mason dataset (Table 19, 20), while CLARK failed in classifying fungi in all experiments (Table 19-22). CLARK had the poorest performance in the virus classification tasks at the species level (Table 19, 21). CLARK slightly outperformed Kraken for bacteria complete genome datasets (Table 19). The gaps in sensitivity at species level between Poseidon and CLARK were 7% to 16% (Table 19). For bacteria draft assemblies, Poseidon achieved much higher sensitivity at species level than Kraken and CLARK, up to 24% and 34% (Table 19). Kraken had higher sensitivity at the genus level than CLARK, but Poseidon achieved always higher genus level sensitivity, up to 11%, than Kraken (Table 20). CLARK's overall sensitivity was much poorer than with Poseidon and Kraken. In particular, sensitivity for fungi and virus datasets was near zero (Table 19, 20).

Kraken had the highest precision in most species level experiments (Table 21), although the differences were not as pronounced as they were at the genus level (Table 22). In the best cases (Viruses (HS20) - Mason, Bacteria DRAFT (HS20) - ART), Kraken's precision at species level was about 20% higher than the one of Poseidon (Table 21), while at genus level it was only 3-5% (Table 22). This low precision at the species level was caused by the fact that Poseidon, a multi-label classifier, predicted many more labels than a multi-class classifier such as Kraken. While Kraken predicted a single label for a homologous sequence originated from a common ancestor, Poseidon produced additional labels

if these have the highest and equal probability. Interestingly, Poseidon obtained the best sensitivity and precision for fungi in all the experiments, which may indicate that Poseidon can classify more complex genomes better than the other algorithms.

For instance, assume that a sequence of *Zaire Ebolavirus* is simulated, thus the ground truth label is *Zaire Ebolavirus* at species level and *Ebolavirus* at genus level. Coincidently, this sequence is shared by the other species in the same genus, *Sudan*, *Reston*, *Taï Forest*, and *Bundibugyo Ebolavirus*. Thus, Kraken predicts *Zaire ebolavirus* or simply *Ebolavirus* genus. For Kraken, the precision at the species level is not that much affected. However, Poseidon will yield all the possible taxonomy identifiers for a single homologous sequence such as *Sudan*, *Reston*, *Taï Forest*, and *Bundibugyo Ebolavirus*. This multi-labeling behavior greatly degrades the apparent prediction at species level since the ground truth label of the sequence is *Zaire Ebolavirus*. The other species in the same genus only increase the number of predictions, leading to lower precision. However, when one changes the taxonomy rank to genus, both algorithms report with same precision in this example.

Table 19. Sensitivity at species level with *k* giving the best species level sensitivity.

Name	Simulator	Poseidon	Kraken	CLARK
Bacteria (HS20)	ART	**95.3049 (k=22)**	81.7673 (k=22)	84.7517 (k=31)
Bacteria (HS20)	Mason	**98.7638 (k=22)**	80.613 (k=21)	82.2265 (k=22)
Bacteria (MS)	ART	**97.0446 (k=24)**	86.9922 (k=28)	90.2371 (k=31)
Bacteria (MS)	Mason	**96.5408 (k=25)**	86.3921 (k=22)	89.2171 (k=31)
Bacteria DRAFT (HS20)	ART	**93.228 (k=22)**	69.1699 (k=22)	58.8982 (k=22)
Bacteria DRAFT (HS20)	Mason	**93.4815 (k=21)**	69.0768 (k=22)	59.6889 (k=21)
Bacteria DRAFT (MS)	ART	**95.5458 (k=24)**	78.0754 (k=28)	65.9049 (k=25)
Bacteria DRAFT (MS)	Mason	**95.4087 (k=24)**	78.4149 (k=22)	66.855 (k=23)
Viruses (HS20)	ART	**94.5471 (k=22)**	84.4962 (k=22)	0 (k=20)
Viruses (HS20)	Mason	**94.0826 (k=22)**	83.3491 (k=21)	0 (k=19)
Viruses (MS)	ART	**96.3292 (k=23)**	88.9728 (k=31)	0 (k=31)
Viruses (MS)	Mason	**95.6554 (k=25)**	88.0053 (k=22)	0 (k=20)
Fungi (HS20)	ART	**99.8423 (k=21)**	99.6077 (k=22)	0
Fungi (HS20)	Mason	**99.2846 (k=21)**	87.9346 (k=21)	0
Fungi (MS)	ART	**99.9885 (k=18~31)**	99.7077 (k=23)	0
Fungi (MS)	Mason	**99.7808 (k=22)**	88.2769 (k=23)	0

Table 20. Sensitivity at genus level with *k* giving the best species level sensitivity.

Name	Simulator	Poseidon	Kraken	CLARK
Bacteria (HS20)	ART	**99.1211 (k=22)**	95.3735 (k=22)	85.0653 (k=31)
Bacteria (HS20)	Mason	**98.7485 (k=22)**	95.4446 (k=21)	82.7914 (k=22)
Bacteria (MS)	ART	**99.4843 (k=24)**	96.6003 (k=28)	90.6662 (k=31)
Bacteria (MS)	Mason	**99.466 (k=25)**	97.066 (k=22)	89.6548 (k=31)
Bacteria DRAFT (HS20)	ART	**98.0099 (k=22)**	88.128 (k=22)	63.5678 (k=22)
Bacteria DRAFT (HS20)	Mason	**98.2412 (k=21)**	89.679 (k=22)	64.1879 (k=21)
Bacteria DRAFT (MS)	ART	**98.6716 (k=24)**	91.5959 (k=28)	72.0541 (k=25)
Bacteria DRAFT (MS)	Mason	**98.8836 (k=24)**	93.457 (k=22)	73.0266 (k=23)
Viruses (HS20)	ART	**96.4616 (k=22)**	88.4214 (k=22)	0.0897 (k=20)
Viruses (HS20)	Mason	**96.2271 (k=22)**	87.7676 (k=21)	0.0844 (k=19)
Viruses (MS)	ART	**97.3803 (k=23)**	91.2069 (k=31)	0.1039 (k=31)
Viruses (MS)	Mason	**97.1198 (k=25)**	90.7067 (k=22)	0.0986 (k=20)
Fungi (HS20)	ART	**99.8731 (k=21)**	99.6462 (k=22)	0
Fungi (HS20)	Mason	**99.3269 (k=21)**	88.0269 (k=21)	0
Fungi (MS)	ART	**100 (k=18~31)**	99.7192 (k=23)	0
Fungi (MS)	Mason	**99.8077 (k=22)**	88.3154 (k=23)	0

Table 21. Precision at species level with *k* giving the best species level sensitivity.

Name	Simulator	Poseidon	Kraken	CLARK
Bacteria (HS20)	ART	88.8172 (k=22)	99.146 (k=22)	**99.5161 (k=31)**
Bacteria (HS20)	Mason	88.131 (k=22)	**98.8059 (k=21)**	98.1797 (k=22)
Bacteria (MS)	ART	91.6248 (k=24)	**99.4125 (k=28)**	99.3715 (k=31)
Bacteria (MS)	Mason	90.3728 (k=25)	98.7208 (k=22)	**99.2787 (k=31)**
Bacteria DRAFT (HS20)	ART	78.8388 (k=22)	**96.8722 (k=22)**	89.1626 (k=22)
Bacteria DRAFT (HS20)	Mason	80.0638 (k=21)	**97.225 (k=22)**	86.6328 (k=21)
Bacteria DRAFT (MS)	ART	84.0007 (k=24)	**97.7152 (k=28)**	90.9033 (k=25)
Bacteria DRAFT (MS)	Mason	84.3788 (k=24)	**96.8711 (k=22)**	90.1003 (k=23)
Viruses (HS20)	ART	81.296 (k=22)	**99.6802 (k=22)**	0 (k=20)
Viruses (HS20)	Mason	79.7638 (k=22)	**99.5345 (k=21)**	0 (k=19)
Viruses (MS)	ART	86.6957 (k=23)	**99.8599 (k=31)**	0 (k=31)
Viruses (MS)	Mason	83.5409 (k=25)	**99.5409 (k=22)**	0 (k=20)
Fungi (HS20)	ART	99.8769 (k=21)	**99.9923 (k=22)**	0
Fungi (HS20)	Mason	**99.7835 (k=21)**	99.5255 (k=21)	0
Fungi (MS)	ART	**99.9885 (k=18~23)**	99.9884 (k=23)	0
Fungi (MS)	Mason	**99.9345 (k=22)**	99.3593 (k=23)	0

Table 22. Precision at genus level with *k* giving the best species level sensitivity.

Name	Simulator	Poseidon	Kraken	CLARK
Bacteria (HS20)	ART	97.4074 (k=22)	99.3689 (k=22)	**99.8535 (k=31)**
Bacteria (HS20)	Mason	97.6838 (k=22)	**99.0089 (k=21)**	98.0395 (k=22)
Bacteria (MS)	ART	98.1723 (k=24)	99.6801 (k=28)	**99.7954 (k=31)**
Bacteria (MS)	Mason	98.5384 (k=25)	99.1425 (k=22)	**99.7119 (k=31)**
Bacteria DRAFT (HS20)	ART	93.0493 (k=22)	**98.4157 (k=22)**	92.3287 (k=22)
Bacteria DRAFT (HS20)	Mason	95.3199 (k=21)	**98.7819 (k=22)**	88.3653 (k=21)
Bacteria DRAFT (MS)	ART	94.7233 (k=24)	**99.1202 (k=28)**	95.7214 (k=25)
Bacteria DRAFT (MS)	Mason	96.543 (k=24)	**98.5328 (k=22)**	93.6509 (k=23)
Viruses (HS20)	ART	95.866 (k=22)	**99.9513 (k=22)**	90.4424 (k=20)
Viruses (HS20)	Mason	95.6593 (k=22)	**99.9262 (k=21)**	89.8379 (k=19)
Viruses (MS)	ART	96.7855 (k=23)	**99.9627 (k=31)**	93.0635 (k=31)
Viruses (MS)	Mason	96.2351 (k=25)	**99.8996 (k=22)**	87.1707 (k=20)
Fungi (HS20)	ART	99.9269 (k=21)	**99.9961 (k=22)**	0
Fungi (HS20)	Mason	**99.8685 (k=21)**	99.4914 (k=21)	0
Fungi (MS)	ART	**100 (k=18~31)**	99.9884 (k=23)	0
Fungi (MS)	Mason	**99.9615 (k=22)**	99.3381 (k=23)	0

5.4 Discussion

I have introduced a highly sensitive multi-label taxonomy classifier of sequencing reads on top of the population index. This algorithm can provide preliminary information for metagenomics studies such as accurate classification and quantifications of species in the sample. I have not yet addressed the importance of functional classifications of microorganisms in this study. Studying functional roles of the microbiome community, rather than focusing only on which species or strains are found in the sample is likely to be of great important to decipher correlations with particular diseases [The Human Microbiome Project Consortium, 2012, T. Yatsunenko et al., 2012, S. Greenblum et al., 2012]. Though different species can produce the same metabolites, a precise stratification or abundance estimates of microorganisms at species or strain level in the sample would be one of the prerequisite types of information. Since I have demonstrated the algorithmic capability of accurate predictions at species level for metagenomics datasets, the functional quantification can be realized by replacing the abstraction layer of the population index.

5.5 Conclusion

Metagenomics has gained more and more importance due to not only highly abundant microbial community but also active interactions with hosts. Their roles in various disorders and the health of ecosystems may be immensely critical though projects aiming at exhaustive understanding about them has recently embarked. The identification and quantification of taxonomic rank in a dataset would be elementary materials for downstream analyses. Furthermore, the algorithmic capability of accurate classifications for given genomic sequences indicates that it can be furnished to infer more precise functional roles.

I introduce a taxonomy classification algorithm, Poseidon, which achieves a high sensitivity at species and genus level on top of the population index. The hypothesis that the variable length *k*-mer may yield a higher sensitivity than fixed *k*-mer based algorithms is proved by comparing with recent algorithms, Kraken and CLARK. I have clarified why the precision at the species level is much lower than the one at genus level such that Poseidon is a multi-label classifier while the others are not. In addition, Poseidon consumes reasonable space and achieves comparable runtime.

5.6 References

J. Ahn, R. Sinha, Z. Pei, C. Dominianni, J. Wu, J. Shi, J.J. Goedert, R.B. Hayes, and L. Yang (2013) Human Gut Microbiome and Risk for Colorectal Cancer. J Natl Cancer Inst., 105(24):1907-11.
M. Albertsen, P. Hugenholtz, A. Skarshewski, K.L. Nielsen, G.W. Tyson, and P.H. Nielsen (2013) Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes, Nat Biotechnol., 31(6):533-8.
S. Baize (2014) Emergence of Zaire Ebola virus disease in Guinea, N. Engl. J. Med., 371:1418-25.
M.J. Bauer, A.J. Cox, and G. Rosone (2011) Lightweight BWT construction for very large string collections. In Proceedings of the 22nd Annual Symposium on Combinatorial Pattern Matching (CPM 2011), Springer, LNCS 6661.
R.J. de Groot et al. (2013) Middle East Respiratory Syndrome Coronavirus (MERS-CoV): Announcement of the Coronavirus Study Group, J Virol., 87(14):7790-2.
S. DeWeerdt (2015) Microbiome: Microbial mystery, Nature, 521:S10–S11.
P. Ferragina, and G. Manzini (2000) Opportunistic Data Structures with Applications, FOCS 2000, 390.
D.R. Garza, and B.E. Dutilh (2015) From cultured to uncultured genome sequences: metagenomics and modeling microbial ecosystems, Cell Mol Life Sci., 72:4287–308.
X.Y. Ge et al. (2013) Isolation and characterization of a bat SARS-like coronavirus that uses the ACE2 receptor, Nature, 503:535-8.
W. Gerlach, and J. Stoye (2011) Taxonomic classification of metagenomic shotgun sequences with CARMA3, Nucleic Acids Res. 39(14):e91.
S. Greenblum, P.J. Turnbaugh, and E. Borenstein (2012) Metagenomic systems biology of the human gut microbiome reveals topological shifts associated with obesity and inflammatory bowel disease, Proc Natl Acad Sci USA, 109(2):594-9.
M. Holtgrewe (2010) Mason - a read simulator for second generation sequencing data. Technical Report TR-B-10-06, Institut für Mathematik und Informatik, Freie Universität Berlin.
W. Huang, L. Li, J.R. Myers, and G.T. Marth (2012) ART: a next-generation sequencing read simulator, Bioinformatics, 28 (4):593-594.
J. Kallmeyer, R. Pockalny, R. R. Adhikari, D.C. Smith, and S. D'Hondt (2012) Global distribution of microbial abundance and biomass in subseafloor sediment, Proc Natl Acad Sci USA, 109(40):16213-6.
H. Li (2014) Fast construction of FM-index for long sequence reads, Bioinformatics, 30(22):3274-5.

E.C. Lim (2015) Poseidon—A highly sensitive and efficient taxonomy classifier, presented at Cold Spring Harbor Laboratory Meeting (Genome Informatics), Poster 145.
S.S. Mande, M.H. Mohammed, and T.S. Ghosh (2012) Classification of metagenomic sequences: methods and challenges, Brief Bioinform., 13(6):669-81.
R. Ounit, S. Wanamaker, T.J. Close, and S. Lonardi (2015) CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers, BMC Genomics, 16:236.
N. Shah, H. Tang, T.G. Doak, and Y. Ye (2011) Comparing bacterial communities inferred from 16S rRNA gene sequencing and shotgun metagenomics, Pac Symp Biocomput., 165-76.
The Human Microbiome Project Consortium (2012) Structure, function and diversity of the healthy human microbiome, Nature, 486, 207–214.
The NIH HMP Working Group (2009) The NIH Human Microbiome Project, Genome Res. 19:2317-2323.
Y.G. Tong et al. (2015) Genetic diversity and evolutionary dynamics of Ebola virus in Sierra Leone, Nature, 524(7563):93-6.
K. Ueno, A. Ishii, and K Ito (2014) ELM: enhanced lowest common ancestor based method for detecting a pathogenic virus from a large sequence dataset, BMC Bioinformatics, 15:254.
M. Velasquez-Manoff (2015) Gut Microbiome: The Peacekeepers, Nature, 518(7540):S3-11.
D.E. Wood, and S.L. Salzberg (2014) Kraken: ultrafast metagenomic sequence classification using exact alignments, Genome Biology, 15(3):R46.
T. Yatsunenko et al. (2012) Human gut microbiome viewed across age and geography, Nature, 486(7402):222-7.

Chapter 6

6. Epilogue

Why do biologists study living organisms – or life – in the universe?

There might be several different answers. I study living beings to answer my own philosophical hypothesis: *is it possible to overcome death, which is opposite to life, by means of science?* According to my previous observations, unfortunately, I conclude, at this moment, that death is not a target to be deciphered but to be admitted. In a non-scientific area, the 'God's world', many believed in the eternity of life, though no one has observed such a super natural being in the real world. For that reason, a more practical strategy would be to delay death by lowering the probability of getting diseases, or developing medicines. NGS technology has allowed to speed up our understanding of diseases at a molecular level. For instance, the following topics have been actively studied: genetic disorders [P.F. Sullivan, 2015, Deciphering Developmental Disorders Study, 2015, F.K. Wiseman et al., 2015, A. Vivante, and F. Hildebrandt, 2016], infectious diseases [T.N. Petersen et al., 2015, D.J. Park et al., 2015, J. Quick et al., 2016], or cancers derived from somatic mutations [C. Greenman et al., 2007, I.R. Watson et al., 2013, I. Martincorena, and P.J. Campbell, 2015, S.A. Shukla et al., 2015].

Fisher, Wright and Haldane established the foundation of population genetics by applying adequate statistical analyses (refer to section 0.1). A closer look at modern biology reveals that understanding life has led to data-driven comparative analyses. Due to the high volume of NGS datasets, however, the naked human eye is not sufficient to find molecular patterns of dis- or similarities. To observe large-scale molecular dynamics, computers have been widely applied. Bioinformatics and computational biology, coined by Paulien Hogeweg, and Ben Hesper in early 1970s [P. Hogeweg, 2011], have helped to gather similar signals into groups, and find dis- or similarities among those groups in data from biological subjects. The advances in bioinformatics have enabled us to collectively and statistically analyze large NGS datasets in reasonable time.

This study aimed at providing a large scale computational framework to observe dynamics and diversity in populations of individuals of a single species as well as in metagenomics datasets. The first problem encountered was the representation of those genomics datasets. NGS data can be represented in a probabilistic data structure such as a bloom filter based graph (refer to section 0.2.2). However, graphs require sophisticated mechanisms to deal with cyclic representation and to preserve positional information. Unlike graphs, the FM-index provides both compressibility and efficient string operations as described in Chapter 1. The FM-index itself does not directly solve the repeat problem. However, intrinsically, the FM-index does not break up long reads to build an internal representation. Thus, with sufficiently long sequencing reads, the repeat problem would be naturally solved. I explicitly addressed the inverse sentinel array, a mapping from the sentinel letters to the entries in an abstract layer, to define the population index. The abstract layer can represent taxonomy identifiers, genes, and so on, depending on the application.

The second challenge were sequencing errors. They introduce false vertices and edges in a graph notation. False signals propagate along downstream analyses, thus they should be corrected at the first step. I introduced a *k*-mer based sequencing error correction module, Trowel, in Chapter 2. Trowel builds two high-quality *k*-mer indices to correct sequencing errors. Due to the asymmetric *k*-mer structure, Trowel is capable of improving read level accuracy, which is a very important characteristic for

quantitative analyses. I also addressed a serious flaw in the error correction evaluation toolkit, ECET, which has been widely used to evaluate sequencing error correction modules, yet the previous implementation does not comply with the definitions for the metrics (refer to section 2.5.1).

The first application of the population index was a taxonomy classifier, Poseidon, explained in Chapter 5. NGS short reads are classified into a taxonomy identifier based on sequence similarity and cumulative length statistics. Unlike in fixed *k*-mer based methods, no significant fluctuations were observed in a series of evaluations. An overall higher sensitivity for species classification and the highest sensitivity for genus level assignments prove the discriminating power of Poseidon for environmental samples. Since all the species in the datasets consisted of uniformly sampled 1,000 reads, the evaluations in this study may be regarded as less biased than those presented in the Kraken paper (refer to table 16). This successful application suggests that the population index would be a suitable representation for a pan genome.

The third concern was a computationally efficient way to find homologous sequences among individuals in a population. The homology search process is known as a sequence alignment problem. In Chapter 3, I extensively summarized prior research on multiple sequence alignment (MSA) and whole genome alignment (WGA). The problems of previous methods were addressed such that scalability, interpretation, and evaluation issues remained unsolved, causing a halt of progress in the field. On top of the population index, I presented a new multiple whole genome aligner, Kairos, which combines the strong points of both MSA and WGA algorithms. A reproducible evaluation scheme is suggested using simulated positions of variants to evaluate the alignments rather than ambiguously comparing alignments themselves. The positional information and the number of alignments vary greatly among tools, thus introducing variants of known positions presents a reasonable alternative to the direct comparison of alignments, which is in fact could not easily established as the 'true' aligments remain unknown. It should be noted that Kairos is fundamentally different from short read aligners such as BWA [H. Li, and R. Durbin, 2009] or bowtie 2 [B. Langmead, and S.L. Salzberg, 2012], both of which align sequencing reads against a single reference genome. The population index in this study represents a pan genome rather than a single reference genome. Moreover, Kairos is intended to be used with long reads such as for those from SMRT sequencing technology or complete genome sequences. As emphasized, sequencing errors in long reads should be corrected before starting any analysis pipelines.

The last challenge was tracking the genetic dissimilarity among individuals in a population. A population could be represented by a set of genomes from a single species, or a cell population from a single organism in which each cell contains somatic mutations. Since homologous sequences can be identified by the Kairos algorithm, additional analyses regarding the neighboring context of alignment blocks would uncover the heterogeneous elements in particular individuals. I presented a structural variant (SV) calling algorithm, Apollo, in Chapter 4. Though its precision should still be improved, a high sensitivity, around 90% except for translocation events, was achieved. The detection of translocation events are typically more difficult with respect to high sensitivity and precision because of dual error sources: insertions and deletions. To the best of my knowledge, there is no previous algorithm which confers multiple whole genome alignments alongside variant information from point mutations to structural variants, and can align thousands of *Arabidopsis thaliana* genomes in reasonable time and space.

Comparative genomics may reveal hidden patterns with respect to diagnosis of genetic disorders, detection of somatic mutations, and rapid identification of pathogens. The accumulation of such knowledge may contribute to developing clinical medications or stop the spread of infectious viruses. In conclusion, the major contributions of this study are 1) alleviating tedious and time consuming efforts to merge heterogeneous alignments, 2) detecting similarities and differences among samples in a population (Kairos, and Apollo), 3) suggesting an evaluation scheme for SV calling, 4) conferring an

overall high sensitivity, 5) introducing a highly sensitive taxonomy classifier (Poseidon), and 6) a de-noising scheme (Trowel), and 7) providing reference uses of the population index for future researches, which is a well-suited representation for a pan genome.

6.1 References

Deciphering Developmental Disorders Study (2015) Large-scale discovery of novel genetic causes of developmental disorders, Nature, 519(7542):223-8.
C. Greenman et al. (2007) Patterns of somatic mutation in human cancer genomes, Nature, 446(7132): 153–158.
P. Hogeweg (2011) The Roots of Bioinformatics in Theoretical Biology, PLoS Comput Biol., 7(3): e1002021.
B. Langmead, and S.L. Salzberg (2012) Fast gapped-read alignment with Bowtie 2. Nature Methods, 9:357–359.
H. Li, and R. Durbin (2009) Fast and accurate short read alignment with Burrows-Wheeler trans-form, Bioinformatics, 25(14):1754-60.
I. Martincorena, and P.J. Campbell (2015) Somatic mutation in cancer and normal cells, Science, 349(6255):1483-9.
D.J. Park et al. (2015) Ebola Virus Epidemiology, Transmission, and Evolution during Seven Months in Sierra Leone, Cell, 161(7):1516-26.
T.N. Petersen et al. (2015) Meta-genomic analysis of toilet waste from long distance flights; a step towards global surveillance of infectious diseases and antimicrobial resistance, Sci Rep., 5:11444.
J. Quick et al. (2016) Real-time, portable genome sequencing for Ebola surveillance, Nature, 530(7589):228-32.
S.A. Shukla et al. (2015) Comprehensive analysis of cancer-associated somatic mutations in class I HLA genes, Nat Biotechnol., 33(11):1152-8.
P.F. Sullivan (2015) Genetics of disease: Associations with depression, Nature, 523(7562):539-40.
A. Vivante, and F. Hildebrandt (2016) Exploring the genetic basis of early-onset chronic kidney disease, Nat Rev Nephrol. doi: 10.1038/nrneph.2015.205.
I.R. Watson, K. Takahashi, P.A. Futreal, and L. Chin (2013) Emerging patterns of somatic mutations in cancer, Nat Rev Genet., 14(10):703–718.
F.K. Wiseman, T. Al-Janabi, J. Hardy, A. Karmiloff-Smith, D. Nizetic, V.L. Tybulewicz, E.M. Fisher, and A. Strydom (2015) A genetic cause of Alzheimer disease: mechanistic insights from Down syndrome, Nat Rev Neurosci., 16(9):564-74.