Error Correction of Second-Generation Sequencing Reads

by

Eric Marinier

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2014

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The introduction of second-generation DNA sequencers has enabled researchers to explore biological information in ways never before possible. These sequencers provide increased throughput over first-generation sequencers at decreasing costs. However, the information produced by these sequencing technologies contains errors which may complicate downstream analyses. The error correction problem involves locating sequencing errors and making edits that correct or remove errors. We introduce Pollux, a platform-independent error corrector which identifies and fixes errors produced by second-generation sequencing technologies. We evaluate Pollux on several diploid bacterial data sets. Using standardized test data, Pollux corrects 85% of Roche 454 GS Junior, 86% of Ion Torrent PGM, and 94% of Illumina MiSeq errors. We compare Pollux to several current error correctors. Pollux performs comparably with the most effective correctors when correcting Illumina data and makes significant improvements when correcting Roche 454 and Ion Torrent PGM data. Furthermore, we provide evidence that Pollux can correct errors in the presence of varying coverage and improves the quality of sequence assemblies.

Acknowledgements

I would like to thank my supervisors, Daniel G. Brown and Brendan J. McConkey, for their guidance and support during my studies. I would also like to thank my committee members, Bin Ma and Andrew Doxey, for their review of this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The introduction of high-throughput sequencing reads has allowed numerous sequencing applications, including *de novo* genome assembly [1], genetic disease detection [18], and cancer mutation discovery [46], to be performed in significantly less time at decreasing costs [18]. In pharmacogenomics, DNA sequence data is used to find genetic variations which have an effect on drug efficacy and toxicity [7, 18, 29]. These applications require high quality data to perform analyses. However, sequencing technologies produce a non-trivial number of errors which complicate downstream analyses. This work introduces platform-independent error correction software named Pollux. Pollux is capable of correcting a variety of sequencing errors produced by different sequencing technologies and is applicable for numerous applications.

1.1 Motivation

Second-generation sequencing technologies have revolutionized genome sequencing [18]. They provide massive throughput at a relatively low cost and enable research that would have not been practical otherwise. However, the increased throughput and reduced cost comes at the expense of read length and quality [18], relative to previous Sanger sequencing technology. There are currently three predominant technologies used for DNA sequencing: Roche 454, Ion Torrent, and Illumina. However, the overwhelming majority of error correctors [19, 20, 22, 26, 55] primarily target reads produced by Illumina sequencers. This is in part because of Illumina's popularity and the relative simplicity in correcting substitution errors in Illumina data when compared to correcting other error types present in Roche 454 and Ion Torrent data. Modern error correctors [19, 20] have expanded their capabilities to remove insertion and deletion errors in Illumina reads. However, they still remain largely ineffective at correcting Roche 454 and Ion Torrent data because of the presence of homopolymer repeats. We show this in detail in Section 5.3.

1.2 Background

All known living organisms encode their genetic instructions required for development and functioning in deoxyribonucleic acid (DNA) molecules. These DNA molecules typically exist as a double-stranded structure, with each strand complementary to the other. DNA is organized within the cell into chromosomes, and may additionally be present within plasmids for bacteria, or within mitochondria or chloroplasts within eukaryotes. The en-

tire collection an organism's genetic information is its *genome*. The process by which DNA is copied is called *DNA replication*. The complementary nature of DNA enables double-stranded DNA to be constructed from one strand through a process called *DNA polymerization*.

The process of obtaining DNA or RNA sequence information from an individual or group is called *sequencing* and machines that produce this information are termed *sequencers*. The first generation of sequencers use a technique called *Sanger Sequencing*. The second, or next, generation of sequencers moved away from this approach and adopted various high-throughput techniques. Typically, sequencing information takes the form of *reads*, text strings describing the DNA or RNA composition of a fragment within the sequencing target. Characters in reads correspond to nucleotide *bases* in the DNA. These bases are adenine (A), cytosine (C), guanine (G), and thymine (T) in DNA, with thymine replaced by uracil (U) in RNA. A region consisting of many repeats of the same base, such as AAAAAA, is a *homopolymer*. *Coverage* refers the amount of oversampling in a sequencing run or specific location within a sequencing target. The mean coverage a sequencing project is an approximation of how many times each position in the genome will be observed in the set of reads. However, there may still exist regions that are not sequenced as a consequence of sampling and sequencing methodology. Sequencing applied to one species is *genome sequencing* and an assemblage of multiple species is *metagenome sequencing*. Metagenomics studies sequence information from species in an environmental sample. A challenge of metagenomics is characterizing function in the presence of similar individuals as it is not immediately obvious which sequence belongs to which individual.

The complete process by which fragmented reads are reassembled to create a full picture

of a sequencing target is called *sequence assembly*. An initial step in sequence assembly is joining fragmented reads into contiguous sequences called *contigs*. Contigs can be assembled into *scaffolds* if information is known about the approximate number of base pairs between contigs. These scaffolds connect and order multiple contigs into larger structures which may correspond to chromosomes or plasmids. Scaffold construction is accomplished with *paired-end* reads (Figure 1.1), which may be used to bridge missing regions between sequenced contigs. These reads are similar to non-paired-end reads, but specify the approximate number of bases between them.

The concept of k-mers is used extensively throughout this work. A k-mer represents an ungapped sequence of length k. A k-mer profile consists of the $r - k + 1$ k-mers which comprise a sequencing read, where r is the length of the read and k is the length of the k-mers. Where k-mers are repeated across multiple reads, they may be counted and we refer to the dictionary of (kmer, number of counts) pairs as the set of *k-mer counts*. The sequencing quantities of kilobase (kb), megabase (mb), and gigabase (gb) are used throughout this work and are equal to 1000, 1,000,000 and 1,000,000,000 bases respectively.

1.3 Error Correction Problem

The error correction problem involves identifying and correcting read errors introduced during nucleotide sequencing. These errors are not introduced at uniform random locations [12, 40], but can appear more frequently in certain sites that are more prone to errors as an artifact of sequencing technology. Common sources of sequencing errors are imper-

Figure 1.1: An example of paired-end sequencing. The pairs of reads are sequenced from the same fragment with Read 1 sequenced in the forward direction and Read 2 sequenced in the reverse direction.

fect biochemical processes and inaccurate base calling [23, 40]. Additionally, errors may be introduced during procedures which prepare DNA for sequencing, thereby modifying the DNA content of the target before sequencing even occurs, and resulting in accurate sequencing of erroneous bases [28, 53]. The number and type of errors depends primarily on the sequencing technology employed and the number of sequenced bases, but also on the true frequency of error-prone regions such as homopolymers and certain sequence motifs [27, 45].

The error types common to all sequencing methods are substitution, insertion, and deletion. These errors represent inaccurate or missing sequence information within reads. A more specific sequencing error is a homopolymer region being miscalled in its length, resulting in spurious insertions or deletions of the repeated nucleotide. Substitution, or mismatch, errors are single base errors where one base is replaced by another and are corrected by replacing the substituted base. Insertion errors are erroneous bases inserted into the sequence and are corrected by deleting the erroneous bases. Conversely, deletion errors are bases removed from a sequence and are corrected by inserting the removed bases

back into the sequence. The error types and rates of Roche 454, Ion Torrent, and Illumina are varied as a consequence of the differences in their sequencing methodology. This is discussed in greater detail in Chapter 2.

Correction requires locating erroneous bases and modifying the read to be accurate with respect to the original sequencing target. This process is complicated by a number of factors. The first complication involves biases in sequencing technologies. Sequencing errors are not uniformly random and instead may occur more frequently in the error-prone regions of specific sequencing technologies, such as homopolymer regions sequenced by Roche 454 and Ion Torrent technologies. These error-prone regions result in coinciding errors and are more difficult to correct because with increasing error frequency they become increasingly difficult to distinguish from non-erroneous bases. An additional complication involves distinguishing between a sequencing error and a biological variation, such as a difference in sequence within a repeated region of the genome. Such variations, when found in low-coverage repeats, may appear as errors that can be corrected to a high-coverage alternative. However, these reads are correct and describe real biological content. Similarly, diploid genomes with pairs of homologous chromosomes will have many sites which differentiate their chromosomes. In infrequent cases, this genetic variation may appear with the same frequency as sequencing errors and may be incorrectly identified as such.

1.4 Results

We develop software named Pollux which corrects read errors produced by Roche 454, Ion Torrent, and Illumina sequencing technologies. The errors introduced by these sequencers is discussed in Chapter 2. Pollux corrects many substitution, insertion, and deletion errors by removing discontinuities between adjacent k-mer counts in reads. These discontinuities often correspond to sequencing errors. However, as we discuss in Section 5.1.3, they also correspond to biological mutations. Sequencing errors may be corrected by modifying the bases which appear responsible for the discontinuity. We evaluate the fitness of a correction by whether or not it removes these discontinuities. This is described in detail in Chapter 4.

We perform a number of experiments to evaluate how successful Pollux is at correcting sequencing errors. We align uncorrected and corrected *E. coli* reads to a high quality reference genome and use changes in alignment errors to evaluate our corrections (Section 5.1.2). Pollux corrects the majority of errors in these data sets. The alignment evaluation procedure is used by many subsequent experiments. We similarly find that Pollux performs well on a simulated metagenome data set of diverse bacteria (Section 5.2). We compare Pollux to several error correctors and use several data sets (Section 5.3). Pollux performs comparably to other error correctors when correcting Illumina data and makes significant improvements when correcting Roche 454 and Ion Torrent data. Finally, we show Pollux improves the quality of some genome assemblies when using corrected reads (Section 5.4).

Chapter 2

Sequencing Technologies

2.1 First-Generation Sequencing

The draft Human Genome Project [6] was primarily composed of many bacterial artificial chromosomes (BACs) produced with Sanger sequencing. These BACs contained human DNA fragments of approximately length 100 kb and were amplified using the bacteria's own replication pathways. BACs were amplified in bacterial culture, sheared into 2-3 kb fragments, subcloned onto plasmid vectors, and selectively isolated before sequencing. This process was costly and labourious. Meanwhile, sequencing approaches were moving away from BAC-based sequencing and towards whole-genome shotgun (WGS) methods [31]. Sanger WGS methods involve directly shearing the genome and placing fragments into plasmid subclones. The subclones are oversampled and paired-end information is generated to allow assembly of whole genomes. The DNA preparation involved in WGS was

a major improvement over BAC preparation because it allowed genomes to be sequenced more rapidly and readily [31]. However, both methods used the same sequencing procedure described below [31].

The Sanger sequencing procedure is first described in Sanger *et al.* 1975 [52] and automated in Smith *et al.* 1986 [56]. More modern Sanger sequencing is accomplished by introducing a small proportion of dye-terminator nucleotides into the DNA replication procedure. These nucleotides are ligated with fluorescent markers which identify their connected base. When incorporated, these nucleotides terminate polymerization. This procedure results in DNA fragments of various sizes, each with a fluorescent signal identifying their terminating base. Fragments are run through a polyacarylimide gel using electrostatic forces. The gel separates the fragments according to their molecular weight and the spatial configuration of fluorescent markers reveals the DNA composition of the sequencing target.

2.2 Second-Generation Sequencing

The introduction of second-generation sequencers resulted in significantly more information being produced in less time [31] at decreasing costs [18]. An overview of second-generation sequencing times and costs is provided in Table 2.1. This technology has made it practical for more researchers to sequence and assemble complicated mammalian genomes [8, 25]. Second-generation sequencers no longer require the preparation of BACs and instead use the WGS assembly techniques developed towards the end of Sanger sequencing. However, the reads produced by second-generation sequencers are typically shorter and often more

prone to errors. The three second-generation sequencing technologies that generated the data used in this work are Illumina MiSeq, Ion Torrent PGM, and Roche 454 GS Junior. These technologies use different methodologies to produce genetic sequence information which are imperfect and introduce errors. A detailed description of these technologies follows.

2.2.1 Roche 454 Sequencing

Roche 454 sequencing was made commercially available in 2004 as the first high-throughput, whole-genome shotgun sequencing technology [31]. Roche 454 uses an approach called pyrosequencing, which produces light during DNA polymerization that can be translated into DNA bases. In pyrosequencing, the process of adding a DNA base releases a pyrophosphate molecule, which in turn initiates a chemical reaction in the firefly enzyme, luciferase, producing a flash of light [31]. The intensity of this light depends on the number of consecutive bases polymerized.

The Roche 454 sequencing procedure is described in Mardis 2008 [31] and what follows is a paraphrase of their description. The DNA preparation step involves random shearing of the genome into small fragments, ligating short adaptor sequences to the ends of the DNA fragments, and mixing fragments with agarose beads. The agarose beads are equipped with short nucleotide sequences complementary to the specific adapter sequences in the fragment library. The beads and fragments associate in a solution with many more beads than fragments such that on average there is not more than one fragment per bead. Next, the bead-fragment complexes are isolated and fragments are amplified using polymerase

Platform	Read Length	Throughput	Time	Machine Cost
454 GS FLX Titanium XL+	700	700 mb	23 h	$500,000
Ion Torrent PGM (316)	200	100 mb	2 h	$50,000
Illumina MiSeq	150	1 gb	27 h	$125,000
Roche 454 GS Junior	400	35 mb	12 h	$108,000

Platform	Reagent Cost	Primary Error	Base Error Rates
454 GS FLX Titanium XL+	$6200	Indel	0.5%
Ion Torrent PGM (316)	$750	Indel	1.2%
Illumina MiSeq	$750	Substitution	1-2%
Roche 454 GS Junior	$1100	Indel	1%

Table 2.1: Metrics for second-generation sequencing technologies as of 2012. Costs are presented in US dollars. Read lengths and throughputs are the maximum over available sequencing protocols. Time is with respect to a maximum throughput run. Data from p. 903 of Henson *et al.* 2012 [18].

chain reaction so that the surface of the bead is covered with many copies of the same fragment. Sequencing is accomplished by placing DNA-rich beads into individual wells on a picotiter plate. Wells contain enzymes which facilitate DNA polymerization and others which catalyze further downstream reactions required for luciferase to produce light. A solution containing only one type of nucleotide is washed over the plate and into the wells. If the next nucleotide base needed to be polymerized on a bead in a given well is the base introduced, then the polymerization reaction occurs and a flash of light is produced. As all fragments on a bead are identical, they should all be in the same stage of polymerization and together contribute to the intensity of light produced. If a fragment contains a repeated base, or homopolymer run, all repeats are polymerized during the same step and the intensity of light produced is proportional to the number of repeated bases. This process is repeated by sequentially choosing different bases to be used in the nucleotide solution washed over the plate. The flashes of light originating from individual

wells after specific bases are introduced are translated into an ordered series of bases which corresponds to the DNA fragment or read present in the well. This process is completed when polymerization of all fragments is complete.

The intensity of light produced is an analog signal proportional to the number of repeated bases. This signal must be translated to a digital one that determines the number of reported nucleotide bases appearing consecutively. The analog-to-digital conversion is the primary source of errors in the Roche 454 sequencing technology [28]. Many sequencing errors are therefore a consequence of miscalled homopolymer repeat lengths [28, 31]. Roche 454 GS Junior has reported total error rates of 0.5% [28] and indel rates of 0.38% [27]. Luo *et al.* report a homopolymer error rate bias with Roche 454 FLX Titanium reads within AT-rich homopolymers. Furthermore, they find errors are more frequent in homopolymers of greater length.

2.2.2 Illumina Sequencing

Similar to the Roche 454 sequencing method, Illumina sequencing relies on creating homogeneous fragment clusters and identifying polymerized bases by detecting light [31]. However, Illumina creates clusters as spots on a plate whereas Roche 454 uses a water-in-oil emulsion. The primary difference between Roche 454 and Illumina technologies is in their nucleotide polymerization procedure. Roche 454 technologies can add multiple bases of the same type in one cycle whereas Illumina incorporates only one base at time. This makes homopolymer repeat errors much more rare within Illumina data.

The Illumina sequencing procedure is described in Mardis 2008 [31] and we paraphrase

their description. The genome is randomly sheared into small fragments and adapters are ligated to the ends of these single-stranded fragments. Fragments are then randomly attached to a flow cell and amplified into homogeneous fragment clusters using a bridge amplification technique. This amplification process requires the next base in the sequence to be added simultaneously to all reads. Polymerization of single-stranded fragments creates double-stranded fragments. These double-stranded fragments are denatured into two separate single-stranded fragments which remain attached to the flow cell. Amplification is repeated, using each single-stranded fragment as a starting point, until fragment clusters are sufficiently large. Sequencing is performed by introducing DNA polymerase reagents and fluorescently labelled variants of all four nucleotides simultaneously. These labelled nucleotides are chemically blocked to prevent further polymerization, as with Sanger sequencing, after they are incorporated. This means that, unlike Roche 454, bases are added individually. An optical instrument images the fragment clusters and the fluorescent labels contribute to a signal which is translated into a base in a read. A subsequent chemical process then removes fluorescent labels and the polymerization terminating components, thereby preparing fragment clusters for the next round of polymerization and imaging.

While Illumina sequencing involves a conversion of an analog signal to a digital signal, this does not involve conversion of a measured intensity of light into some number of nucleotide bases. The Illumina sequencing base caller distinguishes between four wavelengths of light and makes a single base call using that information, with miscalls introducing substitution errors [23]. As a consequence of Illumina's single base polymerization procedure, is it unlikely to miscall homopolymer lengths [28]. Sequencing errors appear to originate from amplification steps during preparation and low signal quality resulting from sequence-

specific regions and a degrading sequencing environment over time [40]. This may be a consequence of fading intensity, decreasing quality of cluster strands, and an accumulation of fluorescent dyes between sequencing cycles [23].

Illumina MiSeq has estimated substitution error rates of 0.1% and indel rates of under 0.001% [27]. However, additional studies place total error rates at 0.80% [45]. The error rate will depend on the sequencing target and evaluation method. MiSeq errors are not uniformly distributed across the genome [40], but appear to be more frequent around homopolymer runs [41, 45], GGC triplets [40], or towards the 3' ends of reads [54]. There appears to be a higher frequency of mismatches within 10 bases downstream of both a GGC triplet in the forward direction and its reverse compliment (GCC) in the reverse direction. However, there seems to be no correlation between the GGC triplet and a higher mismatch rate if the following triplet is AT-rich [45]. Furthermore, these errors seem to represent as little as 0.0015% of bases [28]. Interestingly, Luo *et al.* [28] report homopolymer errors with Illumina Genome Analyzer II in 1% of genes reported from assembly. Likewise, Quail *et al.* [45] report errors after homopolymer tracts of length 20 and greater. This suggests that homopolymers may indirectly introduce sequencing errors into Illumina reads.

2.2.3 Ion Torrent Sequencing

Ion Torrent sequencing differs from Roche 454 and Illumina MiSeq technologies in that it does not use an optical sensor. The Ion Torrent technology was introduced to overcome the need for electromagnetic sensors and specialized reagents [48]. Sequencing is performed on integrated circuits containing sensors which detect the release of hydrogen ions during

DNA polymerization [48]. This is conceptually similar to the approach used in Roche 454 technologies, which uses a series of chain reactions to observe light produced by DNA polymerase. The Ion Torrent approach to sequencing allows machines to be manufactured at a lower cost while still producing significant throughput (Table 2.1) [18].

We paraphrase the Ion Torrent sequencing procedure described in Rothberg *et al.* 2011 [48]. The DNA preparation step closely resembles the Roche 454 sequencing preparation described in Section 2.2.1. The genome is fragmented and fragments are ligated to adapters. The fragments are then amplified onto beads and placed into wells with other sequencing reagents. The wells contain several copies of the same DNA fragment in an environment which allows DNA polymerization. Sequencing involves washing all four nucleotides in a stepwise manner. When an added base is complementary to the base awaiting polymerization, then the bases are incorporated to the DNA fragments. This reaction occurs for every base within a homopolymer. The reactions release protons which shift the pH environment in the well proportional to the number of bases incorporated. The shift in pH is detected by a sensor below the well, converted to a voltage, and finally converted to some number of nucleotide bases. Immediately following a flow of nucleotides, a wash is used to remove any remaining nucleotides. This process is repeated until all fragments have been polymerized.

It is unsurprising that the sequencing errors produced Ion Torrent sequencing are very similar to those produced by Roche 454 technologies. The technologies both incorporate all homopolymer bases simultaneously and must convert analog signals to a number of bases corresponding to the length of the homopolymer: light intensity for Roche 454, voltage for Ion Torrent. This process is inaccurate and is a major source of errors in Ion Torrent sequencing [45]. Ion Torrent PGM has reported total error rates at 1.71% [45] and indel

rates of 1.5% [27]. The accuracy of PGM reads appears to steadily decrease towards the end of the read [27]. Furthermore, Ion Torrent PGM has a higher observed error rate for calling homopolymers of any length than Roche 454 GS Junior [45].

2.3 Summary

The first generation of sequencing was accomplished using Sanger sequencing methods. The initial Sanger approach used many bacterial artificial chromosomes (BAC). This approach was costly and time consuming. The BAC sequencing approach was eventually replaced by whole-genome shotgun sequencing. Sanger WGS methods improved over BAC methods by enabling more rapid sequencing. Second-generation sequencing improved on WGS methods by further increasing throughput and reducing costs. However, second-generation reads are much shorter than first-generation reads and contain a non-trivial number of errors. The three second-generation sequencing technologies considered in this work are Roche 454, Ion Torrent, and Illumina. Roche 454 sequencing is accomplished using an approach called pyrosequencing, which observes light flashes produced as a consequence of DNA polymerization. The pyrosequencing approach has difficulty resolving the number of repeated bases in homopolymer runs and, as a consequence, the primary source of errors in Roche 454 sequencing is homopolymer repeats. Similarly, Ion Torrent sequencing is accomplished by observing changes in pH after DNA polymerization occurs. Ion Torrent data is characterized by an abundance of homopolyer repeat errors. Finally, Illumina sequencing performs sequencing by polymerizing nucleotides, including homopolymer repeats, individually. Substitution errors are therefore dominant in Illumina

sequencing and homopolymer repeat errors are very rare. The expected error rates for second-generation sequencers is in the range of 0.1% to 2%.

Chapter 3

Related Work

Sequencing errors create problems for numerous application which use sequencing reads. We describe sequence assembly approaches, which are necessarily required to overcome errors to produce meaningful assemblies, and highlight the error correction procedures used by these approaches. However, assembly is not necessary for all sequencing applications and other analyses not requiring assemblers can also benefit from stand-alone error correction. We describe below the assembly procedure and existing stand-alone error correction algorithms similar to our work.

3.1 Sequence Assembly

The sequence assembly problem is closely related to the error correction problem. Sequencing reads used in assembly contain errors which complicate the assembly process. These errors obscure true sequence overlaps and introduce erroneous DNA subsequences into the

assembly. Sequencing errors must therefore be removed or corrected to create an accurate assembly. The earliest efforts of first-generation sequence assembly involved manually aligning print-outs of reads by hand [18]. The following decades improved the throughput of Sanger sequencing methods and the increasing data demanded computational assembly methods. These computational methods performed assemblies by automating the read alignment process [2]. However, the landscape of sequence assembly changed with the introduction of second-generation sequencing technologies. Second-generation reads were significantly shorter and had tremendously greater throughput [18]. The technologies developed for first-generation sequencing, which expected long reads and low coverage, were no longer appropriate and new assembly methods needed to be developed.

Sequence assembly is achieved by observing overlaps in reads and producing long sequences which are a product of overlapping sequences. In this respect, it is similar to the longest common substring (LCS) problem. However, this is complicated by repetitions in the true sequences, sequencing errors, and by the computational complexity of the problem. The assembly process typically involves the creation of *contigs*, or contiguous sequences, and the formation of *scaffolds*, which attempt to connect contigs using additional information [39]. There are three main approaches to sequence assembly [39]. Overlap-Layout-Consensus (OLC) assemblers use an overlap graph and use computationally expensive sequence alignment methods [37]. The de Bruijn graph (DBG) assemblers use a k-mer graph [37]. These methods use much less memory and computational time than OLC methods. However, they have difficulty resolving sequencing errors in reads and repeated regions within genomes [39]. Greedy assemblers tend to be based on either OLC or DBG methods. However, greedy methods are not commonly used because they cannot

easily incorporate global information into their assembly process [39].

There are a number of challenges which complicate sequence assembly. Sequences are repeated within genomes more often than would be expected at random [18]. The consequence of this repetition is that it becomes very challenging to differentiate and fully assemble repeated regions longer than the read length without additional information. This is further complicated when assembling a diploid genome with pairs of nearly identical chromosomes, which will contain repeated regions throughout and between them. Furthermore, it is especially challenging to differentiate between sequencing errors and true mutations within nearly identical repeat regions [37]. Another complication involves staying within the bounds of practical computability [18]. This often requires the implementation of heuristics to guide the assembly process. We focus on sequence assembly without a reference genome to highlight their approach to the necessary task of handling sequencing errors.

3.1.1 Greedy Assemblers

The early implementations of first-generation assemblers for viral genomes used a greedy approach [18]. This was possible because of the relatively simple complexity of some viral genomes [13]. Likewise, the first implementations of second-generation assembly packages used greedy algorithms [37]. These include SSAKE [58] and SHARCGS [11]. The greedy assembly process involves selecting a read or contig and extending it with another which produces the next highest scoring overlap [37]. The contigs then grow by greedy extension and this process is repeated until there remain no possible extension. As is characteristic of

greedy algorithms, this approach can fall into a local maximum by performing an extension that would have helped produced contigs of greater size. Greedy assemblers may use a graph implementation in which nodes represent reads or contigs and edges represent the overlaps between them. These algorithms produce a single path through the graph by considering only the highest scoring edge and then merging the connected nodes.

Greedy assemblers are prone to incorporating false-positive overlaps into contigs [37]. They will propagate errors as the assembler continues to build on false overlaps and connect unrelated sequences. Sequencing errors are implicitly avoided by selecting perfect overlaps before imperfect ones. However, this ignores the possibility of having coinciding errors in multiple reads as a consequence of error prone regions. These reads will produce perfect overlaps and be incorporated in the same manner as non-erroneous reads. Similar approaches attempt to ignore imperfect overlaps entirely. SHARCGS [11] filters errors by removing reads which do not contain a minimum number of perfect overlaps with other reads. Additionally, it optionally requires the combined quality of overlaps, as determined by the sequencer, to meet a minimum threshold. However, these heuristics produce an incomplete assembly as a consequence of regions which were sequenced infrequently within the quasi-random genome sampling.

3.1.2 Overlap-Layout-Consensus

Overlap-Layout-Consensus (OLC) assemblers became popular as a means of assembling larger and more complicated genomes sequenced by Sanger technologies [37]. These approaches were necessary to address the size of the genomes and the complicated repeat

structures within. Some examples of OLC assemblers include the Celera assembler [38], its revised pipeline CABOG [36], and Newbler [32].

Overlaps are precomputed from many computationally expensive pairwise alignments [37]. However, this procedure is aided by a seed and extend heuristic which looks for exact matches of length k and performs alignments which originate from these locations. This heuristic is sensitive to both the length of k and errors within overlap seeds. The overlap graph is constructed from these overlaps and an approximate genome layout is determined. The nodes of overlap graphs represent reads and edges represent overlaps between these reads. The OLC algorithm must determine paths through the graph which represent possible contigs. This is achieved by performing many multiple sequence alignments to determine the exact layout and consensus sequence. However, these multiple sequence alignments are very expensive to compute and heuristics are therefore used to calculate the consensus sequence progressively.

Sequencing error correction is performed in the consensus stage of assembly. This stage is more robust to sequencing errors than the overlap detection stage and less sensitive to imperfect overlaps. The Newbler assembler [32] performs correction in this stage by exploiting sequence coverage. However, this is done within the instrument-specific "flow space" of Roche 454 sequencing technologies. As described in Section 2.2.1, homopolymer lengths are observed as signal intensities and are converted into a fixed number of nucleotide bases. Newbler maintains the signal intensities of homopolymer repeats and rounds into a "base space" after the consensus signal intensity is calculated from many overlapping reads. This "flow space" correction is much more sensitive to the underlying technology which produces these errors than any correction which operates in a "base space". However, this

procedure is specific to the Roche 454 sequencing technology.

3.1.3 De Bruijn Graph

The de Bruijn graph model of assembly became increasingly popular with the introduction of high throughput second-generation sequencing technologies. The genome assemblies of the Sanger era required 7x to 10x coverage [18], which may be interpreted as expected over-sampling. However, second-generation sequencing technologies produce runs with upwards of 50x coverage [18]. DBG assemblers were introduced to address the problem of short reads and high coverage which are slow to assemble using the OLC assembly approach. DBGs require significantly less memory to maintain as they typically do not maintain entire reads throughout the assembly process. They are therefore more suitable for assemblies which have limited memory resources. Euler [43] was the first assembler to use a DBG model and was later improved on by Velvet [60] and ALLPATHS [15].

De Bruijn graphs were developed independently of sequence assembly and assembly implementations are sometimes referred to as k-mer graphs [37]. These graphs do not explicitly maintain reads and overlaps. Instead, each k-mer observed in the read set forms a node of fixed length k and edges are placed between all pairs of k-mers that are the prefix and suffix of $k+1$-mers of reads. The result is a graph of short sequences connected by edges indicating overlap. If reads were error free, the genome would correspond to some path through this graph. The advantage of the de Bruijn graph approach is that it scales well with high sequence coverage. This is because every k-mer is only added once to the graph. However, connections are added to existing nodes as they are observed in

reads. Any unbranching path through the graph represents an unambiguous contig which is typically compressed into a single node with a size larger than k. Conversely, repeated regions within the genome correspond to branches in the graph. Repeats longer than length k are impossible to resolve without additional sequencing or assembly techniques. The scaffolding process uses paired-end information, which provides an approximate inter-read distance, to resolve repeat regions and close gaps. Additionally, some assemblers resolve short repeats and simple errors by threading reads through the graph during the graph reduction process [4].

The consequence of the improved scalability is sensitivity to sequencing errors [18]. As noted in Sections 3.2.1 and 4.2, a single base error changes k k-mers into ones unlikely to be observed elsewhere in the read set. However, if no information regarding sequence coverage is maintained, it can be difficult to resolve these errors. The majority of error correction within de Bruijn graph assemblers involves observing erroneous graph topology and attempting to resolve the graph into simpler paths. Errors near the end of reads typically create short "tips" in which a path is connected to the graph only at one end. Errors within the read create short "bubbles" in which graph paths branch and reconnect at nearby locations. Much of the error correction involves removing these "tips" and "bubbles" [37]. However, these bubbles may correspond to real mutations, and assemblers must either construct two contigs, increasing the risk of a misassembly error, or merge branches into a single contig which may not be accurate [18]. Some assemblers additionally address errors by preprocessing reads to remove errors [37]. Other attempts incorporate some information of coverage into the graph and remove paths or errors with less support [42]. However, this risks abandoning areas of the genome which were, by random nature,

sequenced at a lower coverage.

3.2 Stand-Alone Error Correction

There has been a substantial effort in the development of stand-alone error correction software [19, 22, 26, 55]. These correctors are designed to correct errors within second-generation sequencing reads. They must avoid introducing new errors and optionally remove reads which contribute no information or which might complicate downstream processes. However, the majority of these correctors [22, 26, 55] are only designed to correct errors produced by Illumina sequencing technologies. These correctors primarily target simple substitution errors within these technologies. While there has been an effort [19, 20] to additionally correct insertion and deletion errors, we will see in Section 5.3 that these correctors perform inadequately on Roche 454 and Ion Torrent sequencing data.

Second-generation error correction methods typically involve either a k-spectrum approach, such as Quake [22] and Reptile [59], or a multiple sequence alignment (MSA) approach, such as Coral [50], ECHO [21], and SGA [55]. The k-spectrum correctors try to correct reads such that all k-mers that comprise a read have counts above a certain threshold. MSA correctors use an approach that resembles the error correction procedure used in OLC assembly (Section 3.1.2). We describe in detail Quake, a k-spectrum based error corrector, which is similar to the work in this thesis, and additionally describe SGA [55], RACER [20], Musket [26], and BLESS [19]. We compare the performance of these error correctors against Pollux in Section 5.3.

3.2.1 Quake

Quake [22] was published in 2010 and corrects substitution errors in high-throughput, second-generation sequencing projects. Specifically, Quake currently targets reads produced by Illumina sequencing technologies. Similar to our work, Quake employs a k-mer coverage approach to error correction. Quake is designed for sequencing projects with greater than 15x coverage [22], as it uses k-mer redundancy to locate and correct errors. However, unlike our correction software, Quake incorporates quality scores into its k-mer correction algorithm. Quality scores are a measure of confidence about the accuracy of a base and are assigned during the base calling process [12, 22]. These scores allow Quake to make corrections which are motivated by an existing measure of trust. Correction is achieved by categorizing a read's k-mers into either trusted or untrusted and making changes to a read until all its k-mers are trusted. When a read contains many untrusted k-mers which are not corrected, the read is filtered from the set of corrected reads.

Similar to our approach in Section 4.2, Quake makes the observation that single base errors alter the k-mers that overlap an erroneous base and uses this information to inform corrections. A frequent consequence is that these erroneous k-mers appear only once or twice within the entire set of k-mers. The authors therefore assume that k-mers with low coverage are uncommon in a high-throughput sequencing project and a consequence of a sequencing errors. Quake first counts all of the k-mers within sequencing reads and categorizes k-mers as either low-coverage untrusted or high-coverage trusted based on the number of times they occur in the entire data set. However, rather than incrementing a k-mer count by 1 when it is observed, Quake instead increments the count by the product

of the probabilities the base calls are accurate, as defined by the quality scores, for all bases in the k-mer. They refer to this process as *q-mer counting*. The q-mer counting procedure can be understood as weighted k-mer counting that approximates k-mer counts over the error distribution defined by quality scores. The authors observe that counts of error k-mers and true k-mers exist as two overlapping distributions. The q-mer count threshold between untrusted and trusted is chosen by selecting a cutoff between these two distributions such that the estimated ratio of error k-mers to true k-mers is sufficiently high.

Quake locates potentially erroneous bases by first exploring the intersection of a read's untrusted k-mers. These intersections are the base positions shared by all untrusted k-mers in the read. In the case of a single substitution error, the error will affect k k-mers and the intersection will be the erroneous base (Figure 3.1). However, if there exists multiple errors within a read, the intersection of untrusted k-mers may be empty. When this is the case, Quake expands its search space by exploring the union of untrusted k-mers. However, we will show in Section 4.2 that this requires searching a larger search space than necessary. Quake recognizes this and employs heuristics to avoid searching a larger space. When the intersection of trusted k-mers is empty, Quake trusts all bases which overlap the rightmost and leftmost trusted k-mers bordering the untrusted regions. Additionally, Quake creates correction clusters in longer reads containing multiple errors, within which localized correction may be performed.

There is some doubt about the effectiveness of quality scores provided by sequencers. There is evidence to suggest that high quality scores overestimate the true quality of bases while low quality scores instead underestimate the true quality of these bases [12]. However,

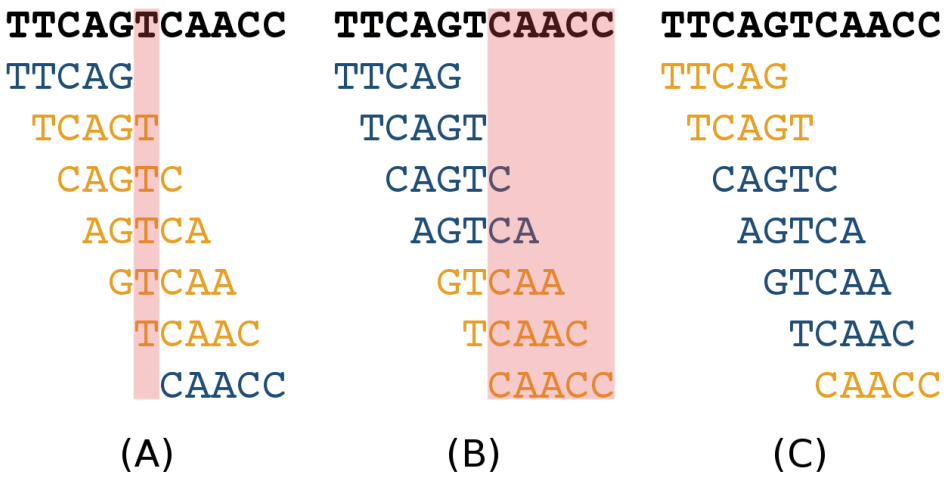

Figure 3.1: Localizing errors in Quake. The untrusted (orange) and trusted (blue) k-mers of reads (black) are shown horizontally. The intersection of untrusted k-mers is shown as a vertical bar and is used to locate positional errors (A). However, this intersection may contain multiple positions (B) or may be empty as a consequence of multiple errors in a read (C) [22].

the authors argue that edit-distance based correction methods should not ignore quality scores entirely as they can be useful in directing the correction search space. Furthermore, quality scores can be informed by known sequencing biases, such as A and C being mistaken for one another in Illumina technology because they share a detection laser [12]. Quake abandons regions containing more than 12 positions with poor quality scores. The authors define a position to have poor quality if the probability of having an error of at least 1%. This is done because the authors found the software was not effective at correcting these regions. The decision to quickly abandon these regions is motivated in part by computational requirements. However, as we will see in Section 5.5, these regions may contain a considerable number of correctable errors.

A consequence of using a trusted k-mers approach to correction is that it is compli-

cated when the same sequencing error appears in multiple reads. Since Quake makes the assumption that sequencing errors produce k-mers that appear infrequently, it may be less effective at correcting data sets containing errors that do not occur at uniformly distributed random positions. Furthermore, as explained in Sections 2.2.1 and 2.2.3, Roche 454 and Ion Torrent sequencing introduce insertion and deletion errors as a consequence of homopolymer repeats. However, Quake does not directly correct these errors.

Since Quake requires a single threshold for separating untrusted and trusted k-mers, it may be unsuitable for correction of projects with low or mixed coverage. However, the authors note that low-coverage regions may be present in projects with sufficient overall coverage due to the random nature of high-throughput sequencing technologies [22]. Furthermore, Quake will have difficulty correcting metagenomic projects in which multiple targets are sequenced at various levels of coverage [24]. These projects would require a more flexible correction methodology which can accommodate these levels of coverage. Additionally, a single threshold approach is complicated by repeated sequences which will have a higher than expected coverage. This is problematic for Quake when attempting choose a threshold for untrusted and trusted k-mers. However, the authors avoid this by determining true k-mer coverage by sampling from multiple distributions.

3.2.2 SGA

The SGA error corrector is a standalone component of the SGA assembly pipeline [55] and uses multiple correction strategies. The first strategy involves classifying k-mers into either untrusted or trusted based on their multiplicity. This approach is similar to Quake [22]

and other correctors [19, 20, 26]. The SGA error corrector assumes that base-calling errors occur independently at random [55]. SGA identifies read positions which are not present in any k-mer with a frequency higher than a specified multiplicity threshold. These positions are substituted with the three other nucleotide bases and accepted if they produce a k-mer with a frequency which exceeds the threshold. However, as we will see in Section 5.3, this strategy does not perform well in the presence of insertion and deletion errors found in Roche 454 and Ion Torrent reads.

The second, and default [55], correction strategy in SGA involves finding inexact overlaps in reads. This strategy locates the set of reads which overlap the read in question using a seed and extend algorithm. A multiple sequence alignment is constructed from the set of reads and sequencing errors are removed using a simple consensus-based correction procedure. The SGA algorithm uses overlap and consensus techniques in a manner similar to the OLC assembly process described in Section 3.1.2. SGA employs heuristics to avoid miscorrecting true sequence variation in diploid genomes. These heuristics remove reads containing multiple conflicted positions from the multiple sequence alignment. These conflicts must be supported by multiple reads. However, performing many multiple sequence alignments can be a costly operation.

3.2.3 RACER

The authors of RACER provide a short description of their method in Ilie and Molnar 2013 [20]. Similar to other error correctors [19, 22, 55], RACER identifies a k-mer multiplicity threshold and makes corrections such that a read's k-mers exceed this threshold. RACER

encodes k-mers in a 2-bit nucleotide alphabet and maintains observed k-mers in a hash table. The error correction procedure involves exploring the eight nucleotide possibilities on either side of a k-mer. A correction is made when a nucleotide substitution improves the multiplicity of a k-mer over the threshold. However, as we will see in Section 5.3, this approach introduces many errors in the presence of insertion and deletion errors.

3.2.4 Musket

Musket is introduced in Liu *et al.* 2013 [26] and, similar to other correctors [20, 22, 19, 55], uses a k-spectrum approach to correction which classifies k-mers as either untrusted or trusted. The first stage of Musket involves construction of the k-spectrum by counting k-mers observed in reads. However, Musket uses a Bloom filter to reduce the number of k-mers it maintains in its hash table. Similar to the k-mer counting procedure in Pollux, Musket removes all unique k-mers from the hash table after construction. What remains is a library of k-mers that have been observed multiple times in the set of reads.

The error correction procedure involves multiple strategies. The first strategy locates potentially erroneous positions within a read which do not overlap any trusted k-mers. Musket explores substitution corrections at these positions and accepts corrections that make all k-mers covering the position trusted. However, no correction will improve all k-mers when there exists substitution errors in close proximity. Musket uses an aggressive correction strategy to remove these errors. Musket recognizes that the transition between a trusted k-mer and an adjacent untrusted k-mer reveals the position of a potential error. Substitution at this position are explored and correction requires, by default, that at least

2 k-mers become trusted. However, this requirement suggests that Musket may have difficulty correcting adjacent errors.

3.2.5 BLESS

The BLESS algorithm is described in Heo *et al.* 2014 [19] and, similar to Musket [26], uses a Bloom filter to reduce memory requirements. Similar to other error correctors [20, 22, 26, 55], BLESS is a k-spectrum error corrector which categorizes k-mers using a k-mer multiplicity threshold. The k-mers which exceed this threshold are considered solid and those that do not are considered weak. BLESS counts the multiplicity of k-mers by distributing k-mers into several files, counting the contents of a file using a hash table, and then programming solid k-mers into a Bloom filter. This strategy greatly reduces the memory requirements of error correction because the number of solid k-mers will be significantly less than the number of overall k-mers.

Correction is accomplished by converting weak k-mers into solid k-mers. This is similar to Quake's strategy of converting untrusted k-mers into trusted k-mers. BLESS locates sequencing errors using an observation that errors should not overlap solid k-mers. Therefore, read positions which do not overlap solid k-mers may be erroneous. These positions are modified to produce solid k-mers. BLESS additionally extends reads in a manner similar to sequence assemblers to enable correction of errors located towards the ends of reads.

3.3 Summary

Sequence assembly shares many of the challenges faced by error correction. The assemblers must resolve and connect imperfect sequence overlaps created by sequencing errors. The three major approaches to second-generation sequence assembly include: Greedy, Overlap-Layout-Consensus, and de Bruijn graph. The greedy assembly approach involves extending one contig with the contig or read that produces the next highest scoring overlap. One strategy for greedy error correction is filtering reads which do not produce enough perfect or high-quality overlaps. However, this heuristic will produce incomplete assemblies as a consequence of low sequence coverage. Overlap-Layout-Consensus approaches construct assemblies by performing computationally expensive pairwise alignments. Sequencing errors are resolved by taking the consensus sequence of aligned sequences. This approach is robust to errors. However, it is computationally expensive and many heuristics are used to approximate this procedure. De Bruijn graph assemblers reduce sequencing reads into a graph with nodes of length k and edges determined by the observed overlaps. Error correction involves identifying erroneous graph topology and resolving the graph into simpler paths. DBG strategies have greater difficulty resolving repeat regions in the genome.

Stand-alone error correctors employ similar strategies as sequence assemblers. The multiple sequence alignment approach to error correction uses the same strategies as Overlap-Layout-Consensus assemblers. Likewise, k-spectrum error correctors use similar approaches as de Bruijn graph assemblers to avoid computationally expensive sequence alignment. Additionally, the greedy assembler heuristic of filtering reads is used by some stand-alone error correctors to remove uncorrectable reads. We find that the overwhelming majority

of second-generation error correctors use a k-spectrum approach that categorizes reads as either untrusted and trusted. The correction operation in these correctors performs edits that convert untrusted k-mers into trusted k-mers. However, these error correctors use a single threshold to separate untrusted and trusted k-mers and may be unsuitable for applications with variable coverage. Furthermore, many of these correctors only perform substitution corrections and, as we will see in Section 5.3, may therefore be unsuitable for Roche 454 and Ion Torrent correction.

Chapter 4

Error Correction

We introduce Pollux, a platform-independent error corrector for second-generation sequencing technologies. Similar to other methods [19, 22, 26, 47], we approach the problem of error correction using k-mers, consecutive k-letter sequences identified in reads. However, our approach does not identify individual k-mers as erroneous [19, 22], but rather compares the counts of adjacent k-mers within reads and identifies discontinuities between these counts. These k-mer count discontinuities within reads are used to find likely error locations and evaluate correctness. We decompose a read into its k-mers and their associated k-mer counts, which are the number of times each k-mer has appeared in the entire set of reads. When we observe an unexpected change in counts between consecutive k-mer counts in a read, we locate the nucleotide position associated with the discrepancy and identify the position as a potential error source. We explore possible corrections simultaneously and accept corrections which remove the k-mer count discrepancies. A pseudocode of our error correction algorithm can be found in Figure 4.1. The detailed explanation of the algorithm

follows.

We use default k-mer lengths of 31 throughout this work. This length is slightly larger than typically used in assembly [3, 60]. However, length 31 k-mers are considered by other error correctors, such as BLESS [19]. We use longer k-mers because this lets us avoid common short repeats which might otherwise confound our correction procedure. Specifically, $k = 31$ is used because it is the longest odd k that can be represented in a 64-bit word.

4.1 Setup

Pollux begins by scanning across all reads in the data set. A basic preprocessing step removes all leading and trailing wildcard N characters in the sequence. Internal wildcard characters are replaced with either A, C, G, or T in a sequential manner. This allows Pollux to operate within a compressed, four-character alphabet and treat internal, mis-replaced wildcards as substitution errors. Furthermore, experiments are repeatable because the wildcard replacement is deterministic. Pollux identifies all the k-mers of length k in each read (Figure 4.2) and increments their respective counts in a k-mer hash table as they are observed. We maintain a record of k-mers and their reverse complements. The reverse complement is included because of DNA's double-stranded nature. These complements correspond to the same information and should be observed with comparable frequency throughout the data set. Pollux only records observed k-mers and therefore maintains an extremely small subset of all 4^k possible k-mers of length k.

```
1: table ← hash table
2: for all reads read do
3:    trim_N(read) {remove leading and trailing Ns}
4:    replace_N(read) {replace internal Ns with A, C, G, T}
5:    for all k-mers kmer in read do
6:       hash(kmer)
7:    end for
8: end for
9: for all k-mers kmer in table do
10:   if table[kmer] = 1 then
11:      remove(kmer, table)
12:   end if
13: end for
14: for all reads read do
15:   errors ← find_errors(read) {locate all errors in read}
16:   pos ← next(errors)
17:   while pos ≥ 0 do
18:      {explore possible corrections at position}
19:      subs ← substitutions(read, pos)
20:      ins ← insertions(read, pos)
21:      dels ← deletions(read, pos)
22:      {select the best correction at position}
23:      correction ← best(subs, ins, dels)
24:      if valid(correction) then
25:         apply(correction)
26:         errors ← find_errors(read)
27:      else
28:         homops ← homopolymers(read, pos)
29:         correction ← best(homops)
30:         if valid(correction) then
31:            apply(correction)
32:            errors ← find_errors(read)
33:         end if
34:      end if
35:      pos ← next(errors)
36:   end while
37:   if low_information(read) then
38:      filter(read)
39:   end if
40: end for
```

Figure 4.1: The Pollux error correction algorithm.

```
AGGCCTATTCGATTCGAAATCGAGGATATCGATCGTACTGGATGCTATCGATCGATCACTGCAGT

AGGCCTATTCGATTCGAAATCGAGGATATCG                                  (5)
 GGCCTATTCGATTCGAAATCGAGGATATCGA                                 (6)
  GCCTATTCGATTCGAAATCGAGGATATCGAT                                (5)
   CCTATTCGATTCGAAATCGAGGATATCGATC                               (6)

                                (...)

(6)                              ATCGTACTGGATGCTATCGATCGATCACTGC
(7)                               TCGTACTGGATGCTATCGATCGATCACTGCA
(8)                                CGTACTGGATGCTATCGATCGATCACTGCAG
(8)                                 GTACTGGATGCTATCGATCGATCACTGCAGT
```

Figure 4.2: An example of counting the k-mers of length 31 that comprise a single read. The read is shown above and a subset of the corresponding k-mers are found below. Possible k-mer counts, with respect to the entire data set, are provided in parenthesis.

After aggregating k-mer information from all reads, Pollux frees a significant amount of memory by removing k-mers observed once from the hash table and implementing a policy of reporting a count of 1 whenever a k-mer is not found in the hash table. This is similar to the approach used by Musket [26]. In development, we found that removing these k-mers significantly reduces memory requirements and improves execution times during error correction. This is particularly true for larger, error-prone data sets, such as Ion Torrent PGM. In one *E. coli* data set [27], which we evaluate in Section 5.1.2, 46% of Roche 454 GS Junior (1), 85% of Ion Torrent PGM (1), and 41% of Illumina MiSeq k-mers are unique. Keeping these k-mers in the table contributes no additional information and can be safely removed using this strategy. When multiple files are corrected simultaneously, unique k-mers may be removed from the hash table after processing each file or after all preprocessing is complete. This strategy can significantly improve execution times when memory limits

are being approached. However, the memory bottleneck remains during k-mer counting and requires additional strategies [9, 35, 49] to improve. The number of unique k-mers will depend on sequencing depth, the error rate of the sequencing technology, and the number of error prone regions [35, 49].

4.2 Locating Errors

Pollux locates errors by observing k-mer count discontinuities inside reads and using that information to pinpoint potentially problematic bases. Pollux constructs a k-mer counts array for each read and fills each entry with the count of the k-mer that is left-anchored at that index position in the sequence (Figure 4.3). There are $r-k+1$ such counts associated with a read, where r is the length of the read and k is the length of the k-mers. Reads of length shorter than k are ignored and left uncorrected.

A read that is not erroneous is assumed to have a k-mer count profile that is reflective of random sampling across the genome. This is motivated by the WGS approach second-generation technologies use to produce sequencing information. Second-generation sequencers fragment the sequencing target [31] and sample enough fragments such that, on average, a position in the genome will be sequenced numerous times. The sequencing process can be thought of as selecting many starting locations within a genome. Sequencing reads necessarily overlap each other and oversample the genome. The k-mer count profile assumption holds that k-mers which comprise reads reveal the amount of oversampling, or coverage, a particular region of the genome has experienced and that this coverage can be characterized by a Poisson process [4]. It is unexpected that coverage will deviate

Sequence	T	T	C	T	T	A	G	G	G	G
***k*-mer Count**	8	8	6	6	6	6	7	.	.	.
Index	0	1	2	3	4	5	6	7	8	9

Figure 4.3: A simple k-mer count profile associated with a non-erroneous read. The k-mers are of length 4. The k-mers are left-anchored and counts represent the number of times the k-mer has been observed in the entire data set.

significantly between adjacent k-mers within genomes containing little repetition, such as bacterial genomes. A significant deviation in k-mer counts would require an unexpected number of reads to either begin or end at exactly the same location within the genome, or a repeat in the true genome sequence.

A read that contains a characteristic sequencing error will have adjacent k-mer counts that deviate unexpectedly from a random sampling process (Figure 4.4). This is because many sequencing errors occur infrequently [27, 28]. We assume that the erroneous sequence will appear less frequently in reads than the true sequence. These errors produce deviations in coverage and appear infrequently relative to the coverage of their corresponding region of the genome. A key assumption of our method is that most k-mers which overlap an erroneous location will have low k-mer counts. These erroneous k-mer counts comprise a region of the k-mer count profile which will deviate significantly from the rest of the profile. For example, a substitution error, located at least k bases away from the ends of the read, will affect k k-mer counts. If this error is unique within the data set, these k-mer counts will drop to 1. A similarly defined insertion error will affect $k+n$ k-mer counts and a deletion error will affect $k-n$ counts, where n is the length of the indel. Thus, unexpected drops in k-mer counts and corresponding low coverage regions are often a marker of errors. However, we do not immediately make this assumption.

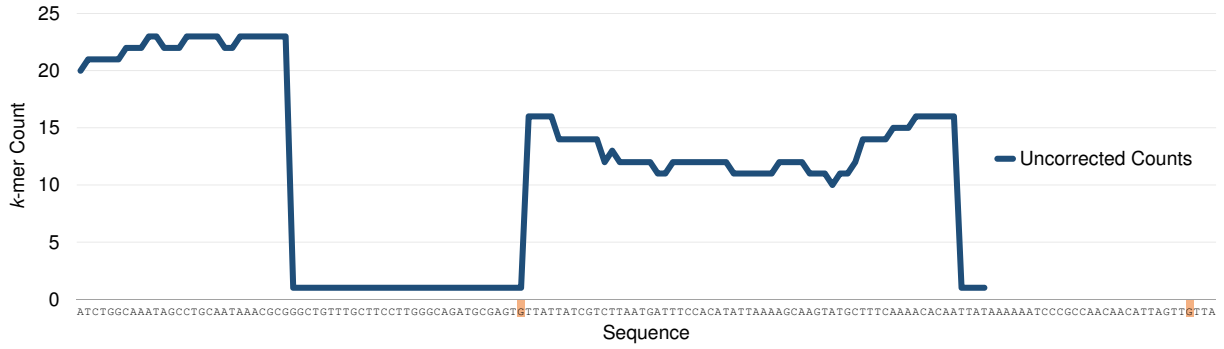

Figure 4.4: An example of the k-mer counts associated with an Illumina MiSeq read containing two highlighted substitution errors. The k-mers are of length 31. A data point corresponds to the number of times a left-anchored k-mer, starting at that given position within the sequence, has been observed in the entire read set. The first error is located near the middle of the read and affects the counts associated with 31 k-mers. The second error is located only four positions in from the 3' end of the read and affects only 4 k-mer counts.

We do not identify reads containing low k-mer counts to be erroneous if such counts appear to follow a random sampling process with no discontinuities. We define consecutive k-mer counts to be potentially erroneous if their difference is larger than a specified threshold. This threshold requires consecutive k-mer counts to have a difference of greater than 3 and greater than 20% the larger count to be flagged as a possible error. This threshold is quite sensitive, and will identify discontinuities corresponding to homopolymer repeats as well as substitution and indel errors. It additionally works quite well for both high-coverage and medium-coverage sequencing projects. The fixed-number component of the threshold operates when correcting low and medium coverage regions while the percent-based component operates during high coverage correction. The thresholds are designed to be conservative in high coverage and operate well in moderate coverage. At low coverage, our approach becomes unable to recognize many errors as there is insufficient k-mer count

information available. However, we cannot expect to recognize all low-coverage errors, as the amount of signal required to reliable identify an error is not available to any approach that only considers k-mer multiplicity.

We determine the erroneous nucleotide position N to be $N = d$ if we observe a low-to-high k-mer count discontinuity and $N = d + k$ if we observe a high-to-low discontinuity, where d is the left index of the discontinuity (Figure 4.5). This is applicable for substitution, insertion, deletion, and homopolymer errors. In the case of deletion errors, a low-to-high k-mer count discontinuity will point to the base immediately before the deletion and a high-to-low discontinuity will point to the base immediately following. With respect to homopolymers, the leftmost base in a homopolymer run is used as an anchor during correction and is found by scanning left from $N+1$ in the case of a low-to-high discontinuity and $N - 1$ with a high-to-low discontinuity.

4.3 Correction and Evaluation

The correction and evaluation procedures are very closely related and are performed together. The correction step involves making a change in the nucleotide sequence at the erroneous base location and the evaluation step creates a measure of fitness by which other corrections are compared. We choose to explore all substitution, insertion, and deletion corrections within a read in a single step. We explore homopolymer corrections independently after all other correction attempts have been exhausted. This is because homopolymer errors frequently coincide in an often predictable pattern which complicates correction. A corrected homopolymer may still appear erroneous when evaluated and un-

Low-to-High

Sequence	A	C	C	C	G	A	C	T	G	C	...
***k*-mer counts**	1	1	1	1	18	17	17	16	16	17	...
Index	00	01	02	03	04	05	06	07	08	09	...

High-to-Low

Sequence	A	C	C	A	G	A	C	G	G	C	...
***k*-mer counts**	17	17	17	18	1	1	1	1	17	16	...
Index	00	01	02	03	04	05	06	07	08	09	...

Figure 4.5: Locating erroneous bases using k-mer counts. The erroneous base N is located at position $N = d$ in the case of a low-to-high discontinuity (top) and $N = d + k$ in the case of a high-to-low discontinuity (bottom), where d is the left index of the discontinuity and $k = 4$. The erroneous bases are underlined.

corrected homopolymers often have less severe discontinuities in their k-mer count profiles. This makes comparing substitution, insertion, and deletion evaluations incompatible with homopolymer evaluations.

The correction and evaluation processes are repeated multiple times within a read until either there remains no detected errors which are correctable or we determine we have made too many corrections. In the later case, we revert all changes and do not correct the read. The maximum number of allowable corrections is the greater of 30 and 20% the length of the read before corrections. The purpose of having a maximum number of corrections is primarily to avoid over-correcting a highly erroneous read which would be better removed. It additionally prevents the software from becoming caught in a cycle of antagonistic corrections being performed repeatedly.

4.3.1 Substitutions, Insertions, and Deletions

Pollux evaluates substitution, insertion, and deletion corrections by making the appropriate sequence change, updating the read's k-mer counts array, evaluating the new k-mer counts, and reverting changes before explore the next possible correction (Figure 4.6). This process is repeated for all substitution, insertion, and deletion corrections. Substitution corrections are attempted by replacing the erroneous base with each other nucleotide base and assessing the success. Pollux performs insertion corrections by deleting the erroneous base and deletion corrections by inserting the four nucleotide bases to the left and right of the erroneous base. This results in three substitution, one insertion, and eight deletion corrections explored and evaluated simultaneously.

The evaluation procedure relates directly to the error location problem. However, instead of locating a k-mer count discrepancy, Pollux quantifies the fitness of a correction by the number of k-mer count transitions that have been improved. This is the number of additional k-mer count transitions for which there are no discontinuities. Therefore, when we evaluate the corrections, we first build the k-mer counts array to reflect the updated sequence. However, we note that the recorded k-mer counts in the hash table are never updated as a consequence of corrections. We choose to evaluate k-mer counts such that the evaluation k-mers overlap largely with the region of the read which appears to contain no errors. As most discontinuities can be evaluated from one of two possible directions, approaching discontinuities from the the non-erroneous direction provides the greatest context for evaluation. Since Pollux corrects errors by removing k-mer count discontinuities, it would otherwise be possible to remove discontinuities by reducing all

Erroneous Read

Sequence	A	T	C	G	A	A	A	A	T	C	G	A	...
***k*-mer counts**	21	20	20	19	01	01	01	01	01	01	01	01	...

Substitution Correction - Substitute "C"

Sequence	A	T	C	G	A	A	A	**C**	T	C	G	A	...
***k*-mer counts**	21	20	20	19	01	01	01	01	01	01	01	01	...

Substitution Correction - Substitute "G"

Sequence	A	T	C	G	A	A	A	**G**	T	C	G	A	...
***k*-mer counts**	21	20	20	19	*19*	01	01	01	01	01	01	01	...

Insertion Correction - Delete

Sequence	A	T	C	G	A	A	A	T	C	G	A	...
***k*-mer counts**	21	20	20	19	01	01	01	01	01	01	01	...

Deletion Correction - Insert "G" Left

Sequence	A	T	C	G	A	A	A	**G**	A	T	C	G	A	...
***k*-mer counts**	21	20	20	19	*19*	*19*	*19*	*19*	*19*	*19*	*19*	*19*	*19*	...

Deletion Correction - Insert "T" Right

Sequence	A	T	C	G	A	A	A	A	**T**	T	C	G	A	...
***k*-mer counts**	21	20	20	19	01	01	01	01	01	01	01	01	01	...

Figure 4.6: A non-homopolymer correction example. The same read is explored for possible corrections. The length of the k-mers is 4. The erroneous base is underlined, the correction bolded, and the improved k-mer counts italicized where applicable. The substitute "G" operation improves the number of continuous k-mer counts by one. However, the insert "G" left operation improved many more k-mer counts and would be selected as the appropriate correction.

k-mer counts within a read to 1. The contextual information provided by the evaluation k-mers prevents this. Furthermore, this directional approach allows us to perform multiple corrections within close proximity.

The very first k-mer we choose to evaluate is the k-mer that entirely overlaps the trusted region of the read and borders the erroneous region. We define a region to be trusted if it contains no k-mer count discontinuities. Additional evaluation k-mers extend into the erroneous region. In order for a correction to be accepted, it must improve the counts of multiple evaluation k-mers up to at least the k-mer which contains the first base following the erroneous base. This requires at least two k-mer counts to be improved when correcting substitution and insertion corrections, which replace and delete a base respectively, and at least three counts to improved when correcting deletion errors, because a deletion correction will always insert a base that improves at least one k-mer count. We require information about the region after the erroneous base to prevent propagating an error further down the read by performing misleading substitutions or insertions. However, a consequence of this evaluation method is that we do not correct adjacent errors outside of homopolymer errors, which are evaluated separately. Pollux considers all possible substitution, insertion, and deletion corrections simultaneously and selects the correction which extends the size trusted region maximally.

4.3.2 Homopolymers

Homopolyer corrections are attempted only after all other possible kinds of corrections at a location have been exhausted. If a substitution, insertion, or deletion correction was

found meeting an acceptable criteria, then a homopolymer correction is not attempted. Homopolymer errors differ from other errors in that their correction may require inserting or deleting multiple adjacent bases. Furthermore, the same homopolymer repeat will often be miscalled in multiple reads. When homopolymer errors coincide in this way, they significantly reduce the number of reads containing accurate-length homopolymers and increase the number of reads containing erroneous-length homopolymers. This complicates the error correction procedure used in Pollux. These coinciding homopolymer errors will cause the k-mer counts associated with a read containing an accurate homopolymer length to appear discontinuous and k-mer count discontinuities associated with erroneous homopolymer lengths to appear less severe than their counterparts. This phenomena is particularly common for very long homopolymers. Roche 454 and Ion Torrent technologies have an increased error rate when converting intensity signals produced by homopolymer runs to the correct number of nucleotide bases. We observe that correct homopolymer lengths are more frequently observed than incorrect lengths in reads containing the same homopolymer region (Figure 4.7). Such reads will therefore have the largest k-mer counts associated with them and deviations from this length will produce smaller k-mer counts.

When Pollux detects a possible homopolymer error, it explores a range of homopolymer lengths and the evaluation process selects the length which maximizes the average of two evaluation k-mers (Figure 4.8). When the selected length is different than the original homopolymer length, then a correction performed and reported. Pollux orientates itself on the leftmost homopolymer base. This base is important because regardless of the direction of approach to homopolymer correction, the leftmost base does not change its index as the length of the homopolymer is varied. When correcting a high-to-low error, the leftmost

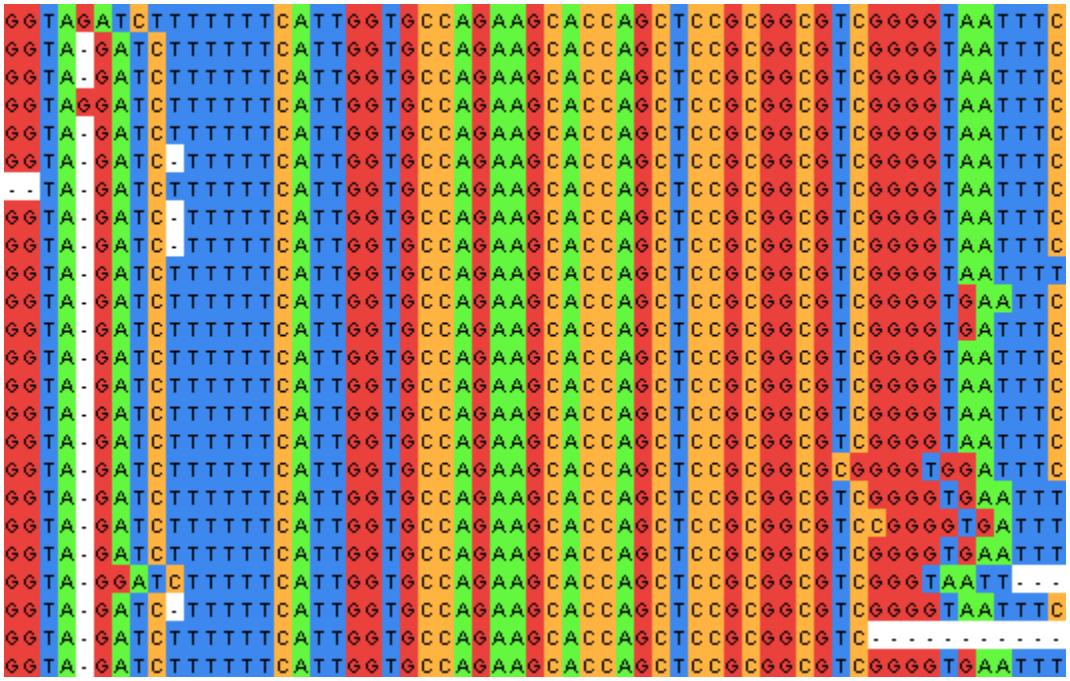

Figure 4.7: An example of homopolymer sizes as would be observed by our error correction procedure. We align all reads which contain the same 31-mer adjacent to the homopolymer region containing several Ts (left). The length of the homopolymer region is 6 in the reference genome. The correct homopolymer length is the one most frequently observed. However, multiple reads deviate from this length by 1 base.

homopolymer base is found by scanning left from the base immediately to the left of the erroneous base until a different base is discovered. When correcting a low-to-high error, the leftmost homopolymer base is found by scanning left from the base immediately to the right of the erroneous base. This base is used as an anchor when adjusting the length of the homopolymer and evaluating the corrections.

Pollux considers possible homopolymer lengths from one half to twice the initial length. We have found homopolymer errors outside this range to be very rare. Additionally, we include the possibility of homopolymers being reduced to single nucleotides. We use two evaluation k-mers in homopolymer correction. The first k-mer originates from the trusted region, overlaps the entire homopolymer, and overlaps one base of the erroneous region of the read opposite the trusted region. The second k-mer is similar to the first, except it is shifted one base further towards the erroneous region. These evaluation k-mers include important contextual information necessary for a confident correction. Pollux is required to evaluate the k-mers containing the entire homopolymer and the two bases immediately following for the same reasons as other corrections: any evaluation k-mers which do not entirely overlap the erroneous region will be misleading. Evaluation k-mers which terminate within a homopolymer region will have higher counts than the corresponding k-mers that overlap the entire region and the following base. This is because k-mers which terminate within a homopolymer region will be common to all homopolymer lengths and confound evaluations. The k-mer count discontinuities present in reads containing accurate length homopolymers would be avoided if evaluation k-mers did not include enough contextual information and instead erroneously short homopolymer lengths would be selected. The contextual information requirement allows us to make fewer false positive corrections at

Homopolymer Length - 2

Sequence	TCACGCAGCTTATCCAGCAGTGGCATCATTTTCCA	...
***k*-mer counts**	34 34 **01 01** 01	...

Homopolymer Length - 1

Sequence	TCACGCAGCTTATCCAGCAGTGGCATCATTTTTCCA	...
***k*-mer counts**	34 34 35 **15 16** 15	...

Original Homopolymer Length

Sequence	TCACGCAGCTTATCCAGCAGTGGCATCATTTTTTCCA	...
***k*-mer counts**	34 34 35 22 **19 19** 18	...

Homopolymer Length + 1

Sequence	TCACGCAGCTTATCCAGCAGTGGCATCATTTTTTTCCA	...
***k*-mer counts**	34 34 35 22 **03 03** 03	...

Homopolymer Length + 2

Sequence	TCACGCAGCTTATCCAGCAGTGGCATCATTTTTTTTCCA	...
***k*-mer counts**	34 34 35 22 03 **01 01** 01	...

Figure 4.8: An example of the same Ion Torrent PGM read undergoing a homopolymer correction and evaluation. The length of the k is 31 and the original homopolymer length is 6. The evaluation k-mers are bolded. We find that the original length is selected as the appropriate homopolymer length. This example illustrates the nature of homopolymer errors: there often exists alternative explanations which are not rare and the homopolymer may appear erroneous during correction.

the expense of missing errors which are located immediately within two bases of the homopolymer region. An erroneous base located in these positions may result in insufficient signal required for correction.

The homopolymer evaluation process is motivated by our observation that the true homopolymer lengths tend to have the maximum k-mer counts associated with them. However, it is a different approach to correction than used when correcting substitution, insertion, and deletion errors, which operate by removing discontinuities in the k-mer count profile rather than maximizing k-mer counts. We implemented a homopolymer correction process which attempted to minimize the difference in k-mer counts associated with the discontinuities on either side of the homopolymer run. However, we discovered this approach was less successful than simply maximizing the average of our evaluation k-mers. We suspect it is important to require as small an evaluation window as possible as this approach is sensitive to errors within k k-mers of a homopolymer error.

4.4 Summary

Pollux approaches the problem of error correction by identifying discontinuities between adjacent k-mer counts and making corrections which remove these discontinuities. Pollux constructs a library of k-mer counts which is not updated during correction. When a discontinuity is observed between adjacent k-mer counts, Pollux attempts all substitution, insertion, and deletion corrections. Pollux selects the correction which best improves the size of the region of the read with no k-mer count discontinuities. Pollux attempts homopolymer error correction only after exhausting all other correction types at a given

position. The evaluation process used by Pollux requires observing the base immediately following the erroneous region. This prevents Pollux from performing misleading corrections. However, as a consequence, Pollux is unable to correct adjacent errors which are not within homopolymer repeats.

Chapter 5

Experiments

We develop evaluation methodology which infers uncorrected, corrected, and introduced errors using sequence alignments. This methodology is used to evaluate the effect Pollux when correcting single and mixed genomes. Similarly, we characterize Pollux's behaviour when correcting reads at low coverage. We compare Pollux with several error correctors on a variety of data sets. We additionally evaluate Pollux's effect on sequence assembly and provide evidence to support the read filtering strategy used by Pollux.

5.1 *E. coli* Reference Alignments

The primary application of Pollux is correcting errors in second-generation sequencing reads generated from one species. We evaluate effect of Pollux by aligning uncorrected and corrected reads to a reference genome and infer uncorrected, corrected, and introduced errors from these alignments. We use well-characterized Roche 454, Ion Torrent, and

Illumina reads, and a high-quality Roche 454 reference generated from the same *E. coli* isolate [27]. Other error correctors [22] have been evaluated by their ability to improve sequence assemblies. However, here we directly evaluate the effect of our error correction software on sequencing reads. We provide information about Pollux's effect on sequence assembly in Section 5.4.

5.1.1 *E. coli* Sequencing Data

Loman *et al.* provide *E. coli* data consisting of read sets from Roche 454 GS Junior, Ion Torrent Personal Genome Machine, and Illumina MiSeq generated from the same O104:H4 isolate, which was the source of a food poisoning outbreak in Germany in 2011 [34]. The sequencing metrics for these runs can be found in Table 5.1. These benchtop machines are designed to accommodate the needs of small laboratories that cannot afford larger high-throughout sequencers, which are typically suited for large-scale applications [27]. However, these sequencing technologies introduce errors at an often higher frequency than their larger counterparts. Nonetheless, they produce a substantial amount of data within a matter of hours or days.

The authors additionally provide a reference genome constructed from reads generated by a Roche 454 GS FLX+ system. These reads were of very high quality, having a modal length of 812 bases and over 99% of sequenced bases as Q64 bases (99.99996% accurate). The reads were used in combination with a paired-end library with 8 kb inserts to produce a reference assembly to which the benchtop sequencers could be compared with confidence. This reference *E. coli* genome consists of multiple scaffolds corresponding to the single

Platform (Run)	Number of Reads (1000s)	Number of Bases (kb)	Mean Read Length
Roche 454 GS Junior (1)	136	71,000	522
Roche 454 GS Junior (2)	138	71,711	521
Ion Torrent PGM (1)	2,484	303,579	122
Ion Torrent PGM (2)	2,155	260,017	120
Illumina MiSeq	1,767	250,357	141

Table 5.1: Sequencing metrics for Roche 454 GS Junior, Ion Torrent PGM, and Illumina MiSeq runs targeting the same *E. coli* isolate. There are two GS Junior runs and two Ion Torrent runs. The single MiSeq run is derived from a multiplexed sequencing run in which multiple targets were sequenced together and separated afterwards to reduce cost.

bacterial chromosome and two large plasmids. There exists 153 gaps within the scaffolds corresponding to repeated regions, as the sequencing strategy is unable to provide sufficient information of these regions for the assembly software to resolve them.

Sequencing all data sets from the same *E. coli* isolate means that the vast majority of differences observed between the reads and reference genome will be a consequence of sequencing errors and not genomic variation. We use this data set in an attempt to avoid confounding errors with the mutations that would otherwise be observed more frequently and possibly outnumber errors in highly divergent bacterial strains. However, while the reference genome is of very high quality, there will still exist sequencing and assembly errors within it. One complication might involve deletion errors, in which a base is removed from the observed sequence and does not have an associated quality score with it, thereby making the high quality base statistic misleading in that respect. However, we remain confident that the number of sequencing and assembly errors present in the reference genome is outweighed by the sequencing errors introduced by the benchtop sequencers.

The authors perform a comparison of the benchtop reads against their Roche 454 GS

FLX+ reference genome [27]. They note that the Shiga toxin-producing phage and two plasmids have significantly higher sequence coverage than elements in bacterial chromosome within the Illumina and Ion Torrent data sets. In particular, there is 25x coverage of the chromosome and 625x coverage of the plasmids in the Illumina data set. This amounts to approximately half the reads mapping to the large plasmids. This effect is much less pronounced in the Ion Torrent data sets. Using the Roche 454 GS FLX+ reference as a ground truth, Loman *et al.* align the benchtop reads to this reference and infer sequencing errors produced by the benchtop sequencers. They show that Illumina MiSeq reads contain the fewest sequencing errors overall, followed by Roche 454 GS Junior, and finally Ion Torrent PGM (Table 5.2). PGM reads aligned the least successfully, with 10% of all reads being discarded because no suitable alignment was found. Furthermore, the authors confirm that the primary error type is substitution within Illumina MiSeq and homopolymer within Roche 454 GS Junior and Ion Torrent PGM [27].

5.1.2 Pollux Evaluation

We use data from the Loman *et al.* benchtop sequencing comparison study [27] to evaluate how well our software performs on bacterial genome data produced by Roche 454 GS Junior, Ion Torrent Personal Genome Machine (PGM), and Illumina MiSeq sequencing technologies. We correct each benchtop sequencing data set independently and additionally correct hybrid data containing GS Junior (1), PGM (1) and MiSeq reads. A summary of our software's reported corrections can be found in Table 5.3. These corrections include true positives, false positives, and compound corrections which may be evaluated differently

Platform	Substitutions	Insertions & Deletions	Reads Aligned
Roche 454 GS Junior	N/A	0.4%	99%
Ion Torrent PGM	N/A	1.5%	90%
Illumina MiSeq	0.1%	0.001%	99%

Table 5.2: The per base alignment errors reported by Loman *et al.* when aligning Roche 454 GS Junior, Ion Torrent PGM, and Illumina MiSeq reads to a Roche 454 GS FLX+ assembled reference genome. The data sets were all sequenced from the same isolate. Sequence alignments are used to infer errors introduced during the sequencing process.

when aligned to the reference. Consistent with the expected error types present, we disable homopolymer corrections when running the MiSeq data. We found these errors were very rare and disabling such corrections avoids introducing a small number of errors. The low abundance of homopolymers errors in the MiSeq data is supported by Loman *et al.*, who suggest there exists less than 0.001% insertion and deletion errors in the data [27]. We allow all corrections in the remaining data sets.

A total of 99% of the GS Junior (1), 89% of the PGM (1), and 99% of the MiSeq reads are retained as high-information reads after correction. While 89% of PGM reads may seem lower than expected, we note that Loman *et al.* were only able to align 90% of PGM data to the *E. coli* reference. This suggests that approximately 10% of the reads are not sufficiently recognizable as *E. coli* and are possibly of very low quality. Furthermore, the low-information reads removed by Pollux are corrected, but are separated from the high-information reads because they appear to contribute little to the downstream assembly processes. These reads may be used if application-appropriate, such as with low-abundance, long-jump sequencing information, which bridges contigs and forms larger scaffolds during assembly.

Pollux requires 3, 5, and 24 minutes for the MiSeq, GS Junior (1), and PGM (2) *E. coli*

Platform (Run)	Corrections				Reads Removed
	Mismatches	Insertions	Deletions	Homopolymers	
GS Junior (1)	24 K	164 K	56 K	17 K	1%
GS Junior (2)	21 K	168 K	54 K	14 K	1%
PGM (1)	611 K	2,609 K	1,864 K	1,107 K	11%
PGM (2)	561 K	2,215 K	1,765 K	969 K	13%
MiSeq	251 K	2 K	3 K	0	1%
Hybrid	947 K	3,008 K	2,599 K	696 K	7%

Table 5.3: The number of corrections reported and low-information reads removed as reported by Pollux. All reads are sequenced from the same O104:H4 *E. coli* isolate. Substitution, insertion, deletion, and homopolymer corrections are performed on all data sets except for MiSeq, in which we do not include homopolymer corrections. The percentage of reads which were removed as a consequence of greater than 50% unique k-mers is provided under "Reads Removed".

data sets, respectively, when executed on a 8-core Linux machine with an Intel Core i7-3820 (3.60 gigahertz) processor and 64 GB of memory. The maximum memory requirements were 9.7 GB when correcting the Ion Torrent PGM (2) data set. However, the Ion Torrent PGM *E. coli* data sets had significantly more errors reported than the similar MiSeq and GS Junior data sets. These errors likely contributed to the longer execution times, as there existed more k-mer count discontinuities which needed to be explored. In comparison, the Illumina MiSeq data set required only a maximum of 1 GB of memory. The upper memory requirements are often temporary and a consequence of using the hash table during k-mer counting, which requires additional space when expanded, and is often reduced by removing unique k-mers from the hash table and freeing unused memory.

The evaluation is performed by aligning reads to the reference before and after correction and observing alignment changes. We use SMALT [44] to align uncorrected and corrected reads to the reference *E. coli* scaffolds. A custom Python script is used to aggre-

gate information from the SMALT alignment about corrected, uncorrected, and introduced errors. We observe whether a read is aligned or unaligned to the reference and whether the reference scaffold it is aligned with has changed. During the assessment procedure, we discard incompatible results such as read pairs which do not align to the same reference scaffold (*e.g.* reads aligning to similar repetitive regions) or have alignment starting positions further than twenty bases apart. This is to prevent confounding mutations within repeated regions with corrections. Similar to Loman *et al.*, we ignore soft-clipped alignment regions and we additionally ignore all aligned bases which are not contained within the mutual alignment interval of the pair of reads with respect to the reference. These removal processes leave us with 99% of the GS Junior (1), 95% of the PGM (1), and 94% of the MiSeq aligned bases for analysis.

We create a list of errors in both reads determined by their error type (mismatch, insertion, or deletion) and position with respect to the reference. The reference genome is used as the frame of reference because it is possible for reads to be reverse-complimented by the alignment software. This process reverses reads and might complicate our evaluation. The index position of errors with respect to the read index may also change. Another complication is with regard to N characters. These characters represent wildcards within DNA sequence and are often a product of knowing the distance between two contiguous sequences, but having insufficient information about the actual sequence content between them. We consider a single nucleotide alignment with an N to be erroneous when the N is located only within the read and not erroneous when located within the reference. This is important because otherwise any read which aligns to a region of the reference containing an N will be reported as having uncorrectable mismatch errors.

When an error is found in an uncorrected read, but not in its corresponding corrected read, we report a corrected error. Conversely, when an error is found in a corrected read, but there is no such error in the corresponding uncorrected read, we report an introduced error. If the same error appears in both the uncorrected and corrected reads, we report it as an uncorrected error. The results of the alignment comparisons using the Loman *et al. E. coli* data are found in Table 5.4 and Table 5.5. We assume that these alignment errors are primarily a consequence of sequencing errors. Therefore, this procedure reveals our software's effect on removing the errors introduced by these sequencing technologies.

We correct the majority of errors within all data sets and corrections are sequencing technology appropriate. We introduce few errors with respect to the number of errors in the uncorrected reads. We correct 85% GS Junior (2), 88% PGM (1), and 94% MiSeq errors in the Loman *et al.* [27] benchtop sequencing data sets. However, we introduce under 4% new errors in all data sets; that is, we correct about 20 errors for every 1 error introduced. The software performs best on Illumina MiSeq data, correcting the vast majority of errors while introducing very few errors. We report 95% of substitution errors, the dominant error type, corrected in our MiSeq (1) data set while introducing only 1% more of such errors. We appear to have some difficulty correcting MiSeq insertion errors, with only 10% of MiSeq insertion errors corrected. However, these insertion errors make up only 0.2% of the total errors, and may also include insertion errors present within the Roche 454 GS FLX+ reference assembly. The next most successful corrections were performed on the Ion Torrent PGM (1) data. We correct 91% of insertion and 86% of deletion errors, which are primarily consequence of Ion Torrents' difficulty with homopolymer regions, in the PGM (1) data set while introducing 2% more insertion, 5% more deletion errors, and

Platform (Run)	Corrected (Abundance (counts/kb))			
	Total	Mismatches	Insertions	Deletions
GS Junior (1)	**81%**	76% (0.33)	82% (2.28)	81% (0.81)
GS Junior (2)	**85%**	79% (0.24)	86% (2.23)	83% (0.76)
PGM (1)	**88%**	82% (1.68)	91% (7.67)	86% (7.10)
PGM (2)	**86%**	80% (1.72)	90% (7.47)	84% (7.70)
MiSeq	**94%**	95% (0.85)	10% (0.0015)	78% (0.007)
Hybrid	**94%**	89% (1.29)	93% (4.25)	96% (4.05)

Table 5.4: The percentage of errors corrected by Pollux in a variety of *E. coli* data sets [27]. The corrected abundance is reported as the number errors corrected per thousand sequenced bases.

3% more errors overall. Finally, we perform least successfully on the Roche 454 GS Junior data, although we correct the majority of errors. We correct 86% of insertion and 83% of deletion errors in the GS Junior (2) data set while only introducing 2% more insertions and 6% more deletion errors. We introduce 15% more substitution errors in GS Junior (2), but the overall number of errors introduced in the data set is 4%. We suspect the introduced substitutions are a consequence of correcting low abundance mutations within repeat regions to their higher abundance alternatives.

Pollux corrects 94% of all errors in hybrid data containing the Roche 454 GS Junior (1), Ion Torrent PGM (1), and Illumina MiSeq reads. This is as successful as corrections performed independently on Illumina MiSeq data and more successful than correcting either Roche 454 GS Junior and Ion Torrent PGM independently. We find substitutions, insertions, and deletions to be corrected in overwhelming majority with deletion corrections being most successful. Pollux introduces 2% more errors into the hybrid data. This is proportionally more than was introduced when correcting Illumina MiSeq. However, the individual rates of error introduction are additionally 2%. These results suggest that

	Introduced (Abundance (counts/kb))			
Platform (Run)	Total	Mismatches	Insertions	Deletions
GS Junior (1)	**4%**	11% (0.048)	2% (0.065)	6% (0.063)
GS Junior (2)	**4%**	15% (0.044)	2% (0.059)	6% (0.050)
PGM (1)	**3%**	2% (0.046)	2% (0.16)	5% (0.44)
PGM (2)	**4%**	2% (0.052)	2% (0.16)	6% (0.52)
MiSeq	**1%**	1% (0.010)	4% (0.0007)	8% (0.0007)
Hyrbid	**2%**	2% (0.032)	2% (0.082)	2% (0.077)

Table 5.5: The percentage of errors introduced by Pollux in a variety of *E. coli* data sets [27]. Introduced errors are a consequence of alignment errors found in corrected reads but not in uncorrected reads. The percentage of introduced errors is calculated dividing the number of introduced errors of one category by the number of corrected and uncorrected errors of the same category. The introduced abundance is reported as the number errors introduced per thousand sequenced bases.

combining reads from different sequencing technologies, in addition to increasing coverage, may enable one technology to cover the weaknesses of another. For example, the addition of Illumina data to Roche 454 or Ion Torrent data will simplify homopolymer corrections.

There are a number of issues which should be considered when interpreting the results. The reference genome is sequenced using Roche 454 GS FLX+ and it will contain some errors. This will give the MiSeq data the appearance of having higher than expected amounts of uncorrected indel errors. This is because any indel errors which are incorporated into the Roche 454 reference genome will appear as uncorrected errors when evaluating the MiSeq reads. Although we expect very few of these to be present in the reference, they may appear with greater frequency in the reference than in the MiSeq reads and our summary of indel corrections. Additionally, a small number of corresponding uncorrected and corrected reads may produce equal-scoring alignments which differ only slightly. When these alignments contain an error that is resolved differently in each alignment, the error

may appear to be corrected and reintroduced. However, we expect alignment noise to be minimal since we both the reads and the reference are sequenced from the same *E. coli* isolate and the reference is of high quality. Furthermore, since our software attempts to find corrections which improve k-mer counts maximally, it is possible to report compound errors as an alternative error type. For example, a homopolymer with a single insertion adjacent to a homopolymer with a single deletion may appear as a mismatch and be corrected as such. This does not have an adverse affect on the correction itself, but may result in reporting more substitution corrections than expected.

5.1.3 Introduced Errors

We locate introduced errors by reporting alignment problems that appear to be introduced as a consequence of error correction. The locations of these problems are determined by the alignment evaluation procedure described in Section 5.1.2. We manually explore some of these errors and suggest explanations for the introductions.

A common source of introduced errors appears to be miscorrecting biological alternatives within repeat regions. Within the *E. coli* genome, there exist large regions that are repeated (Figure 5.1). However, the repeated regions are not always identical and sometimes contain differences which distinguish these regions from their counterparts. This makes error correction challenging when, as a consequence of repeat region frequency within the genome, low abundant alternatives appear alongside highly abundant alternatives. This can give repeated regions containing rare alternatives the appearance of being erroneous. The k-mer count profiles associated with such regions will sometimes have unexpected

discontinuities. These discontinuities may be removed by removing miscorrecting the rare alternative with the more common alternative.

We include one such example (Figure 5.2) of our error correction software replacing one variant with its more abundant alternative. The error was introduced into read M10_0139:1:2:20739:2553#ATCACG-1 at position 2,059,940 on reference scaffold 1 (the chromosome). We use BLAST [61] to compare the O104:H4 *E. coli* genome against itself and observe four regions within the chromosome that overlap with this location. The regions are of length 2.4M, 1759, 1753, and 664, with the longest region corresponds to a large contiguous region of the *E. coli* chromosome which aligns unambiguously with itself. The three smaller alignments correspond to very similar regions with 98.1%, 97.5%, and 94.3% identity respectively, which are found elsewhere in the chromosome. The introduced error corresponds with a variant at position 2,059,940 which is observed in three out of four of the repeat regions. The k-mer counts associated with the uncorrected read suggest that the position is correctable by replacing the base with its more abundant alternative and thereby smoothing many more k-mer counts.

5.1.4 Low Coverage Correction

The coverage of the sequencing projects has a significant effect on the ability of Pollux to remove errors. However, this is common to all error correctors which depend on multiple observations of genome positions. This is because Pollux requires multiple observations of the same location in the genome to observe errors and evaluate corrections within sequencing reads. We evaluate Pollux at various levels of coverage. Pollux corrects proportionally

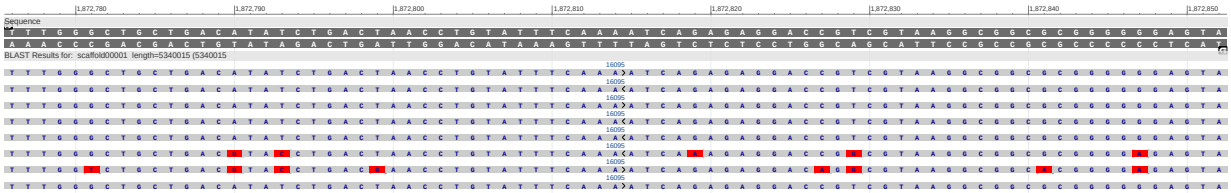

Figure 5.1: An example of repetition within the chromosome of *E. coli* O104:H4. The two dark grey tracks at the top of the alignment represent the forward and reverse strand of the large circular chromosome. The remaining light grey tracks represent regions of the same chromosome which align to the given region. Multiple alignment tracks are observed when the region is repeated within the chromosome. The red highlights mismatch errors, which are presumably a consequence of mutations, within the aligned regions.

more errors as coverage increases. The results are summarized in Figure 5.3.

We evaluate the effect of coverage on correction with Roche 454 GS Junior (2), Illumina MiSeq, and Ion Torrent PGM (1) reads [27]. The evaluation procedure is as described in Section 5.1.2. Lower coverage read sets are created by randomly selecting a fraction of the reads. This is accomplished by including each read with probability p and varying p to create read sets with differing coverages. The coverage is calculated by dividing the total number bases within a read set by the number of bases in all scaffolds of the reference genome. This represents the expected number of times each base in the genome has been sequenced. The percent of errors corrected corresponds to the proportional amount of errors within included reads.

Pollux corrects proportionally more errors as coverage increases. With respect to Roche 454 GS Junior (2) and Ion Torrent PGM (1), the effect increases significantly until approximately 10x coverage. However, Pollux requires greater coverage of Ion Torrent data to correct proportionally as many errors as within Roche 454 and Illumina data. The Illumina MiSeq data suggests that Pollux is sensitive to correction at very low levels of coverage.

Figure 5.2: An example of an error introduced by Pollux into a read sequenced from the *E. coli* O104:H4 genome. Bottom: The region of the chromosome with which the uncorrected read and repeat regions align. The uncorrected read corresponds to the uppermost light grey track and has its edges marked vertically on the left and right. Mismatches with the reference marked with red. The location of the introduced error has been marked with an arrow. Top: The k-mer counts of the read immediately before and after correction. The introduced error location has been highlighted.

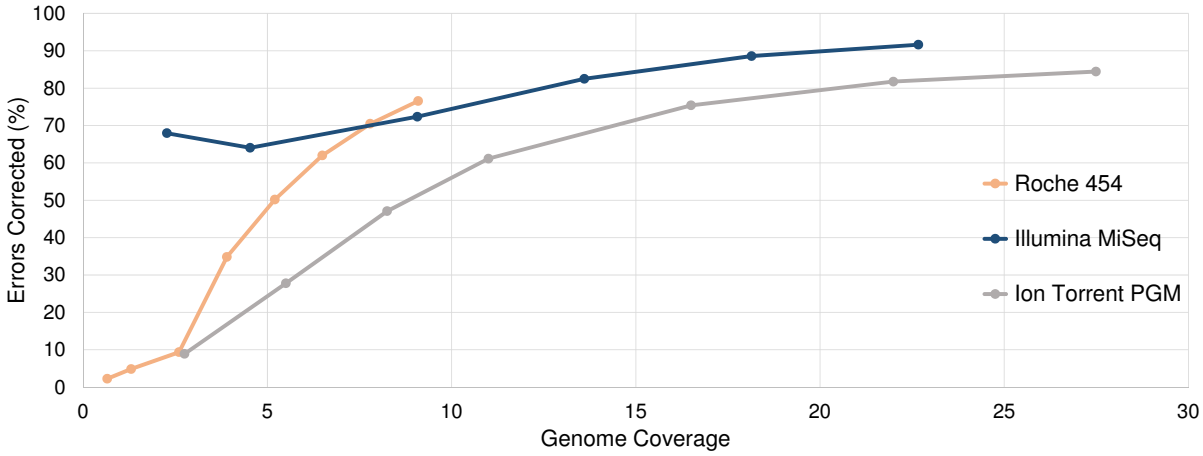

Figure 5.3: The percentage of errors corrected by Pollux at various levels of sequencing coverage. Lower coverage data sets are produced by randomly selecting a subset of reads.

However, as noted in Section 5.1.1, approximately half of Illumina MiSeq reads correspond to small plasmids with 625x coverage while the chromosome has 25x coverage. This would account for a significant amount of the errors corrected at the lower levels of whole-genome coverage. It appears our software requires approximately 5x-10x coverage to make the majority of corrections within a read set.

5.2 Mixed-Genome Reference Alignments

We believe that our algorithm will be suitable for some metagenomic applications that target diverse communities, such as soil metagenomics [57]. Metagenomics involves sequencing an environmental community rather than an individual or isolated species [24]. There are a number of complications which make metagenomic error correction and evaluation more challenging [10]. As the sequencing target is a community, it may contain closely

related individuals with differences in genomic content [17]. This is similar to the problem of variants existing within repeated regions of one individual. However, these regions may be unique within an individual's genome, but exist as similar alternatives throughout a community. Another challenge is the varying sequencing coverage of individuals within the community [24]. Some individuals may be sequenced very rarely while others are sequenced in extreme abundance. Finally, it is difficult to create a data set similar to the *E. coli* data set used in Section 5.1.2. With current technology, a data set of this quality would require isolating and sequencing many individuals within a community. This process would be extremely costly and time consuming. Nonetheless, we attempt to characterize the effect of Pollux on a simulated metagenomic data set despite these challenges.

5.2.1 Mixed-Genome Sequencing Data

We generate a mixed genome data set to approximate real metagenomics data. This is accomplished by combining reads from multiple sequencing projects using similar technologies. We combine two data sets from GAGE [51] with *E. coli* reads [27] to create mixed genome data. This data set is comprised of uncorrected Illumina data from *E. coli* [27], *S. aureus* [51], and *R. sphaeroides* [51] (Table 5.6). The *E. coli* reads are the same as previously evaluated and were sequenced using an Illumina MiSeq sequencer whereas the *S. aureus* and *R. sphaeroides* reads were sequenced using an Illumina Genome Analyzer II sequencer. The data used from the GAGE project is uncorrected and paired fragment files are concatenated into a single file. The *E. coli* reference was assembled from Roche GS FLX+ reads, while the *S. aureus* and *R. sphaeroides* references were assembled with reads

Organism	Sequencing Technology	Number of Reads (1000s)	Number of Bases (kb)	Mean Read Length	Total Coverage
E. coli	MiSeq	1,767	250,357	141	45
S. aureus	Genome Analyzer II	1,294	130,705	101	45
R. sphaeroides	Genome Analyzer II	2,050	207,138	101	45
Mixed	Illumina	5,111	588,200	115	45

Table 5.6: The sequencing metrics for a mixed genome data set comprised of 35% *E. coli*, 25% *S. aureus*, and 40% *R. sphaeroides* reads. All the reads are sequenced using Illumina technology. However, the *E. coli* data set is produced from a MiSeq while the *S. aureus* and *R. sphaeroides* data sets are produced from a Genome Analyzer II.

generated from Sanger sequencing [51]. As our error correction and evaluation procedures ignore the order of reads, we concatenated all read sets into a single file and similarly concatenated all references into a single reference file. This mixed data set was comprised of 35% *E. coli*, 25% *S. aureus*, and 40% *R. sphaeroides* reads. This data set represents a rough approximation of metagenomics data set and is motivated by the desire to use real sequencing data over artificial data. However, we note that a better approximation would only include reads produced by the same sequencing instrument.

5.2.2 Pollux Evaluation

We correct these mixed reads and evaluate the effect using the same alignment procedure described in Section 5.1.2. We again choose to disable homopolymer corrections when correcting Illumina data because it avoids introducing a small number of avoidable errors. A total of 19% of reads were removed using our k-mer removal criteria and not considered in further downstream analyses. Additionally, we evaluated 94% of aligned bases after discarding incompatible alignment locations and soft-clipped bases. While removing 19%

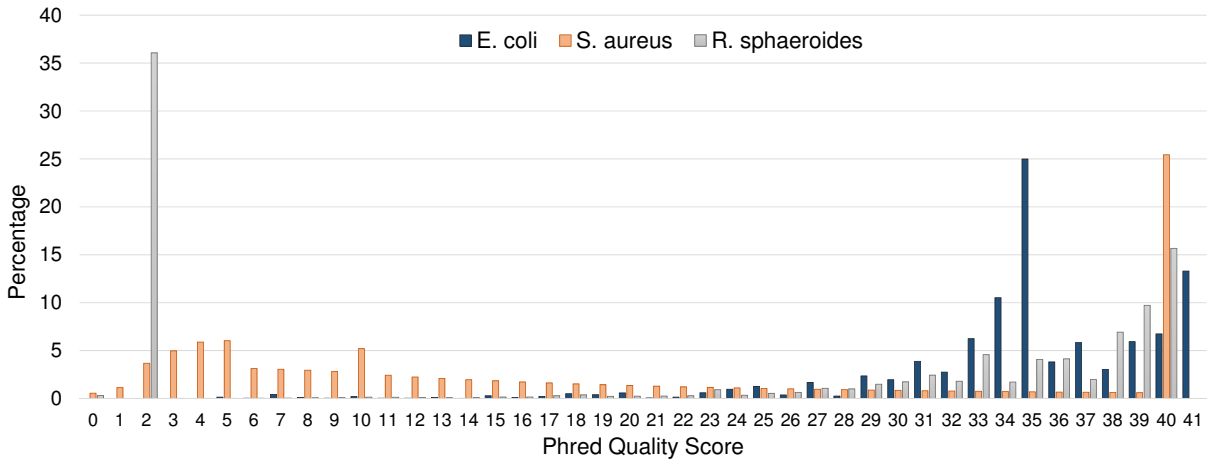

Figure 5.4: The Phred quality scores of the data sets comprising our mixed genome. The percent abundance per base of each quality score within the individual data sets is shown. Quality scores reflect confidence in an observed base with higher values being exponentially more accurate than smaller values.

of reads seems substantial, we note that the Quake [22] and ALLPATHS-LG [3] error correction procedures removed 37% and 36% of the *S. aureus* reads, respectively, and similarly removed 26% and 31% of the *R. sphaeroides* reads in the GAGE genome study [51] when correcting reads independently. This aggressive filtering is further supported by the observation that the *S. aureus* and *R. sphaeroides* data sets contain a significant number of poor quality score bases (Figure 5.4). The *R. sphaeroides* data has very polarized quality scores, with 36% of base scores having Phred [14] quality 2, which corresponds to an accuracy of 37%. This is polarization of quality scores is observed in the *S. aureus* data, although to a lesser extent and spread over many more qualities scores. The median base quality in *S. aureus* is 15, with a base call accuracy of 96.84%, 33 in *R. sphaeroides*, with an accuracy of 99.95%, and 35 in *E. coli*, with an accuracy of 99.97%.

	Corrected (Abundance (counts/kb))			
Organism	Total	Mismatches	Insertions	Deletions
Mixed	**82%**	82% (5.9)	70% (0.087)	73% (0.058)

	Introduced (Abundance (counts/kb))			
Organism	Total	Mismatches	Insertions	Deletions
Mixed	**0.61%**	0.31% (0.022)	11% (0.014)	11% (0.0091)

Table 5.7: The results of Pollux's error correction on a mixed genome data set comprised of *E. coli*, *S. aureus*, and *R. sphaeroides* reads. The abundances are shown as counts/kb and reflect the frequency of the errors relative to the number of bases.

A summary of our results can be found in Table 5.7. Pollux corrects a total of 82% of errors in total with only 0.6% more errors introduced. Specifically, Pollux corrects 82% of substitution errors, 70% of insertion errors, and 73% of deletion errors, with 98% of all of corrections being substitutions and the remaining 2% distributed between insertions and deletions. We introduce as few as 0.6% errors, the most abundant of which being substitutions. We appear to introduce proportionally more insertion and deletion errors than we did with our *E. coli* Illumina MiSeq correction. However, within non-filtered reads, we appear to introduce proportionally fewer errors overall.

5.3 Comparison

We compare Pollux other error correctors. These include Quake [22], SGA [55], BLESS [19], Musket [26], and RACER [20]. Quake and SGA are popular second-generation error correctors whereas Musket, RACER, and BLESS represent recent efforts in this area. We show that Pollux performs comparably with these error correctors and especially improves on Roche 454 and Ion Torrent error correction.

5.3.1 Sequencing Data

We use the Roche 454 GS Junior (1), Ion Torrent PGM (1), and Illumina Miseq *E. coli* sequencing data sets available in the Loman *et al.* comparison [27]. We additionally include *S. aureus* data (SRX007714, SRX016063) available in GAGE [51], *M. tuberculosis* (ERR400373), and *L. pneumophila* (SRR801797). A summary of the sequencing metrics of all data is provided in Table 5.8. The coverage varies from 13x with Roche 454 GS Junior *E. coli* to 260x with Illumina HiSeq *M. tuberculosis*.

5.3.2 Error Corrector Evaluation

We evaluate the effect of correction in the same manner as described in Section 5.1.2. The results of this comparison are located in Table 5.9. We use $k = 31$ for all software except Quake, which uses $k = 19$ because of hardware memory limitations when using a 64 GB memory machine. This is because of problems with JELLYFISH [30], Quake's memory efficient k-mer counting utility. We therefore disable JELLYFISH when running Quake at the expense of memory efficiency. However, Quake specifies using k-mers of approximately this size for its error correction [22]. We note that Quake, SGA, and Musket are intended to only correct Illumina sequencing data. However, we include the effect of their corrections on Roche 454 and Ion Torrent data for completeness. Pollux, Quake, and SGA perform read filtering whereas BLESS, Musket, and Racer do not filter reads and may appear to correct fewer errors as a consequence.

We find that Pollux corrects the greatest percentage of errors in all but the GS Junior *E. coli* data and HiSeq *L. pneumophila*; Pollux is a close second with these sets. Pollux

Organism	Sequencing Technology	Number of Reads (K)	Number of Bases (kb)	Mean Read Length	Total Coverage
E. coli	GS Junior	136	71,000	522	13
E. coli	PGM	4,484	303,579	122	55
E. coli	MiSeq	1,767	250,357	141	45
S. aureus	Genome Analyzer II	1,294	130,705	101	45
L. pneumophila	HiSeq	8,850	885,022	100	260
M. tuberculosis	HiSeq	2,093	316,035	151	72

Table 5.8: The sequencing metrics for a data used in the error correction comparison. We include one Roche 454 technology GS Junior and one Ion Torrent PGM data set. We additionally include multiple Illumina data sets, including MiSeq, HiSeq, and Genome Analyzer II.

filters reads with similar aggressiveness as Quake and SGA. The effect of read filtering is significant within *S. aureus* Illumina data. Pollux is able to obtain a high percentage of errors corrected in this data because of its ability to remove reads which do not contribute information. This is supported by the observation that an assembly generated from this data improves significantly following correction (Section 5.4.2). Pollux introduces errors slightly more than other software. However, it is comparable to RACER when correcting Illumina technologies and comparable to all other software when correcting Roche 454 and Ion Torrent technologies.

RACER corrects the most errors within the Roche 454 GS Junior *E. coli* data, corrects the majority of errors in the PGM data, and performs extremely well on both Illumina data sets while filtering no reads. The amount of errors introduced by RACER is comparable to Pollux within the Illumina data. However, RACER introduces a significant number of errors within GS Junior at 24% and PGM at 16%. This suggests that while RACER is capable of correcting insertion and deletion errors, it performs best with Illumina corrections.

	Illumina MiSeq - *E. coli*		
Software	Errors Corrected (%)	Errors Introduced (%)	Reads Removed (%)
Pollux	**93.81**	1.28	0.89
Quake	58.78	0.05	0.92
SGA	78.65	0.09	1.11
BLESS	83.21	0.10	0.00
Musket	81.75	0.15	0.00
RACER	86.59	1.60	0.00

	Illumina Genome Analyzer II - *S. aureus*		
Software	Errors Corrected (%)	Errors Introduced (%)	Reads Removed (%)
Pollux	**87.04**	0.38	31.73
Quake	75.30	0.10	29.81
SGA	47.45	0.02	10.71
BLESS	55.32	0.06	0.00
Musket	45.04	0.14	0.00
RACER	75.76	0.28	0.00

	Illumina HiSeq - *L. pneumophila*		
Software	Errors Corrected (%)	Errors Introduced (%)	Reads Removed (%)
Pollux	96.16	0.13	6.01
Quake	**99.66**	0.00	4.25
SGA	84.61	0.03	3.87
BLESS	87.61	0.03	0.00
Musket	83.33	0.10	0.00
RACER	94.09	0.16	0.00

	Illumina HiSeq - *M. tuberculosis*		
Software	Errors Corrected (%)	Errors Introduced (%)	Reads Removed (%)
Pollux	**69.98**	0.83	6.51
Quake	68.06	0.12	3.06
SGA	31.99	0.16	0.46
BLESS	60.01	0.13	0.00
Musket	44.68	0.80	0.00
RACER	65.14	1.28	0.00

	Roche 454 GS Junior - *E. coli*		
Software	Errors Corrected (%)	Errors Introduced (%)	Reads Removed (%)
Pollux	81.02	4.14	0.82
Quake	0.21	0.00	0.07
SGA	8.94	3.63	1.64
BLESS	34.68	1.27	0.00
Musket	0.00	0.00	0.00
RACER	**82.03**	24.27	0.00

	Ion Torrent PGM - *E. coli*		
Software	Errors Corrected (%)	Errors Introduced (%)	Reads Removed (%)
Pollux	**87.83**	3.43	10.90
Quake	12.35	2.03	37.60
SGA	5.43	1.12	0.16
BLESS	22.82	0.52	0.00
Musket	9.40	4.88	0.00
RACER	67.86	15.95	0.00

Table 5.9: Comparison of error correction software on multiple data. The evaluation is performed by aligning corresponding uncorrected reads and corrected reads, which were not removed, against a reference genome using SMALT. Corrected errors are an aggregate of all alignment errors which are found in uncorrected reads but not in corrected reads. Similarly, introduced errors are an aggregate of all alignment errors found in corrected reads but not in uncorrected reads and are relative to the sum of corrected and uncorrected errors. The percentage of reads removed by each software is noted. We note that Quake, SGA, and Musket were designed to only correct Illumina sequencing data.

We find that Quake corrects the most errors within Illumina HiSeq *L. pneumophila* data. Quake corrects 99.66% of corrections while effectively introducing no errors. The *L. pneumophila* data has 260x coverage and sequencing errors should be corrected effectively using Quake's untrusted vs. trusted k-mer strategy (Section 3.2.1). Quake performs well when correcting Illumina MiSeq *S. aureus* data. Similar to Pollux, Quake filters a substantial number of reads and corrects many more errors in the remaining reads than most other correctors. SGA, BLESS, and Musket correct and introduce similar amounts of errors within Illumina data with BLESS performing consistently better than SGA and Musket. These error correctors remove a significant number of errors while introducing extremely few errors. However, of these three, only BLESS performs meaningful corrections within GS Junior and PGM data.

All correction software perform poorly on Illumina HiSeq *M. tuberculosis* data. However, the inadequate corrections may be a result of complicated errors in the data or a consequence of having a reference genome which is not similar enough to the sequencing reads. The reference genome was assembled from *M. tuberculosis H37Rv* isolated from a Beijing and Manila patient whereas the reads (ERR400373) were sequenced from an Oxfordshire patient. The reference may therefore contain mutations not found in the sequencing reads which appear as uncorrected errors. Pollux performs the greatest proportion of corrections at 69.98%. However, this is at the expense of 6.51% of reads being filtered. Quake performs comparably with 68.06% of errors corrected and 3.06% of reads filtered. SGA performs most poorly with only 31.99% of reads corrected.

We conclude that the current landscape of error correction software has largely been designed specifically for Illumina sequencing technologies. These error correctors either

perform poorly when correcting insertion, deletion, and homopolymer errors, or, in the case RACER, introduce an overwhelming amount of errors when performing corrections on Roche 454 and Ion Torrent technologies. Pollux competes with these error correctors when correcting Illumina data and performs better than existing correctors when correcting Roche 454 and Ion Torrent data.

5.4 Assembly

We evaluate Pollux as a preprocessing step before *de novo* sequence assembly. We use three data sets and two assemblers for our evaluation. Velvet [60] is used to assemble *E. coli* and *S. aureus* Illumina sequencing reads and MIRA [5] is used to assemble Ion Torrent *E. coli* reads. As shown in the GAGE study [51], error correction software can significantly improve the quality of assemblies.

5.4.1 Sequencing Data

The *E. coli* Illumina MiSeq [27] and *S. aureus* Illumina Genome Analyzer II [51] read sets are used to evaluate the effect of our software on assembly when correcting paired-end reads with both short and long insert lengths. These reads represent a longer fragment of DNA which has been sequenced partially from opposite ends. The approximate insert length of the fragments corresponds to the length of the fragments. This paired-end approach assists the assembly process by bridging contigs over areas of repetition. The *E. coli* reads used in our experiment are paired and have an average read length of 142. The *S. aureus* data

set consists of paired fragment reads of length 101 with average insert lengths of 180 and long-range paired-end reads of length 37 with average insert lengths of 3500.

The *E. coli* Illumina reads are corrected using our software's pair-end correction. This is unlike the similar correction performed in Section 5.1.2, which corrected the reads as though they were all individual reads. The reason for this is because the paired nature of the read is not relevant when evaluating corrections in a context that does not need to connect reads. However, paired information is important when performing assembly, because the connection between the reads assists in joining contigs and scaffolds. A special paired-end filtering procedure is required when correcting paired-end reads to avoid having a miss-pairing of reads in the corrected read set. This can be done by filtering pairs together or by grouping all reads which become unpaired as a result of filtering into a separate file. We choose to filter paired-end reads in complete pairs only when both reads have k-mer counts of 1 comprising more than half of k-mers. This is because the benefit of bridging these alignment gaps is often worth the expense of including an otherwise low information read within a pair. The *S. aureus* paired fragment reads are corrected together using our software's paired-end correction and are likewise corrected for the short-jump reads. However, we choose not to remove any short-jump reads as nearly half of short-jump reads were flagged as having more than 50% unique k-mers. As with paired-end reads with short inserts, removing these short-jump reads renders much of the valuable connection information unusable.

The *E. coli* Ion Torrent reads [27] reads are unpaired and are corrected using our software's normal error correction procedure. The consequence of using unpaired reads is that the assembly software will be unable to produce scaffolds, as there is no information

available which connects distant reads. As noted in Section 5.1.2, the Ion Torrent PGM reads required the longest time to correct out of all *E. coli* reads in the Loman *et. al* benchtop comparison.

5.4.2 Velvet Evaluation

Velvet [60] is used to assemble the *E. coli* and *S. aureus* Illumina data sets. Velvet is designed for *de novo* assembly of short reads using de Bruijn graphs. We used default settings for the paired *E. coli* reads and assembly settings as described by GAGE [51] for the *S. aureus* reads. The results of the assemblies can be found in Table 5.10. We compare the common assembly metrics number of scaffolds and N50 of assemblies using uncorrected and corrected reads. The N50 statistic represents the contig length such that equal or greater length contigs account for at least 50% the length of the assembly. We additionally include the NGA50 statistic, as calculated by QUAST [16], which requires a reference genome. The NGA50 statistic represents the contig length such that aligned contigs of equal or greater length account for 50% the length of the reference after breaking contig misassemblies. This provides a more accurate metric of genome connectedness.

There are fewer scaffolds and larger N50 values in the error corrected assemblies than there are in there uncorrected counterparts. The uncorrected *E. coli* assembly produces 2120 scaffolds with an N50 of 31 kb and a maximum scaffold of size 220 kb. This improves to 1840 scaffolds with an N50 of 37 kb and a maximum contig of size 290 kb. However, we find the NGA50 improves only slightly from 84 kb to 85 kb. The *S. aureus* assembly improves more significantly. The uncorrected assembly produces 737 scaffolds with an N50

	Uncorrected			Corrected		
Assembly	Scaffolds	N50 (kb)	NGA50 (kb)	Scaffolds	N50 (kb)	NGA50 (kb)
E. coli	2,120	31	84	1,840	37	85
S. aureus	737	192	145	603	1,771	202

Table 5.10: Comparison of *de novo* assemblies using uncorrected and corrected reads using Velvet. *E. coli* reads are paired and assembled using default parameters. *S. aureus* reads are paired with average inserts of length 180 and short jump reads with average inserts of length 3500. These reads are assembled using parameterization as described in GAGE.

of 192 kb and a maximum scaffold size of 435 kb. The corrected assembly reduces the number of scaffold to 603 and has a N50 of 1771 kb, which is the maximum scaffold size, and is longer than half the genome length. Furthermore, the NGA50 improves substantially after correction, increasing from 145 kb to 202 kb.

5.4.3 MIRA Evaluation

MIRA [5] is used to assemble Ion Torrent sequenced *E. coli* reads. MIRA is designed to assemble a variety of second-generation sequencing reads using an overlap-layout-consensus approach. We used default MIRA settings for the assembly. There is no paired-end information provided to MIRA. Therefore, the assembly is unable to connect distant contigs and produce scaffolds. The results of the assemblies can be found in Table 5.11. We consider the full assemblies of uncorrected and corrected reads and compare the number of contigs and the size of N50 values. We again include the NGA50 statistic, as calculated by QUAST [16].

We find that our corrections do not produce as significant effect on the assembly as they did with our assembly of Illumina data using Velvet. There are almost half as many contigs

	Uncorrected			Corrected		
Assembly	Contigs	N50 (kb)	NGA50 (kb)	Contigs	N50 (kb)	NGA50 (kb)
E. coli	1,779	77	66	1,017	37	36

Table 5.11: Comparison of *de novo* assemblies using uncorrected and corrected Ion Torrent *E. coli* reads using the MIRA assembler. The reads are unpaired and assembled using default parameters.

produced when using corrected data, suggesting that the assembly is more connected. However, the N50 value is only half as large when using corrected reads and the largest contig shrinks from 205 kb to 107 kb. This suggests that the contigs are more medium sized when using corrected data and more varied in size when using uncorrected data. Furthermore, the estimated size of the *E. coli* genome is 5.521 mb, as provided by the reference genome. MIRA predicts this to be 5.672 mb when using uncorrected reads and 5.454 mb when using corrected reads. The NGA50 value decreases from 77 kb when using uncorrected reads to 36 kb when using corrected reads. This is despite QUAST reporting the uncorrected assembly containing 67 misassemblies and only 31 misassemblies in corrected data.

5.5 Read Filtering

We now show that our approach to optionally filtering reads based on unique k-mers is an improvement over a naive quality score approach. Furthermore, we show that there are many errors which are correctable within reads which contain many poor quality score bases. However, it is not surprising that many errors exist within regions reported as having poor accuracy. The purpose of read filtering is to remove reads which are of poor

quality and uncorrectable, and unlike other error correction schemes [22], we choose not to filter reads which appear to be of poor quality before attempting corrections.

In order to verify this we compared the effect of removing reads using Pollux's unique k-mer approach with a simple quality score approach. In Pollux, reads that contain more than 50% unique k-mers after attempting correction are removed. The quality score approach is accomplished using a custom Python script. The script removes reads which contain more than 10% low quality bases, as provided by the sequencing technology. We define low quality bases to be a Phred [14] quality score of Q10 or less. We then consider the reads which are removed by the quality score approach but not by our k-mer approach. These are reads that are designated as having poor quality scores, but which Pollux considers valuable. We find that Pollux is capable of correcting many of the errors in these reads despite having an abundance of low quality scores. The most notable difference is with respect to the Ion Torrent PGM (1) *E. coli* data set (Table 5.12). We find that Pollux corrects 88% of the 3.4M errors found in reads that would be discarded exclusively through a simple quality score approach. Pollux's k-mer based removal approach can evaluate the usefulness of a read after attempting corrections, retaining more information than filtering using a simple quality based approach before correction.

5.6 Summary

We evaluate the effect of error correction using a methodology which infers uncorrected, corrected, and introduced errors from changes in sequence alignments. We show that Pollux corrects the majority of sequencing errors in Roche 454 GS Junior, Ion Torrent PGM, and

Filtering Strategy	Reads Removed	Errors (Abundance (counts/kb))		
		Corrected	Uncorrected	Introduced
Naive	869 K	3,409 K (17.20)	465 K (2.35)	125 K (0.62)
Pollux	71 K	3 K (0.77)	34 K (8.20)	2K (0.42)

Table 5.12: The effect of Pollux error correction on reads filtered exclusively by Pollux and a naive quality score filtering strategy. Pollux corrects many of the errors within reads removed using the naive strategy. However, unsurprisingly, Pollux performs very few corrections on reads that it filters after attempting corrections. The counts per kb statistic is reported with respect to aligned bases in the set of filtered reads.

Illumina MiSeq *E. coli* reads while introducing few errors. Pollux performs similarly well on a simulated metagenomic data set comprised of *E. coli*, *S. aureus*, and *R. sphaeroides* reads. However, we show these corrections require approximately 5x-10x sequencing coverage. When compared to other error correctors, Pollux performs comparably on Illumina data and exceptionally better on Roche 454 and Ion Torrent data. We find Pollux's error correction improves the quality of an example Illumina assembly when paired with Velvet [60]. However, it appears Pollux does not improve the quality of an example Ion Torrent assembly when paired with the MIRA [5] assembler. Finally, we show Pollux's read filtering strategy allows for successful correction of many reads which might otherwise be removed.

Chapter 6

Conclusion

This work introduces and evaluates an approach to error correction of sequencing reads. We introduce platform independent, second-generation error correction software named Pollux. The correction algorithm works by observing unexpected changes in k-mer counts within sequencing reads and attempting edits which remove these unexpected observations. The k-mer count approach used by our software is highly effective at correcting errors across different sequencing platforms, including Roche 454, Ion Torrent, and Illumina.

Pollux improves over other error correctors by targeting multiple sequencing technologies and error types. While other error correctors target only Illumina sequencing, we have designed Pollux to correct errors common to all sequencing technologies. Pollux is additionally appropriate for correcting hybrid data. We show our implementation is sensitive to errors at a low sequencing coverage and can correct errors in the presence of highly variable coverage. Additionally, we find our software corrects the majority of errors in a

mixed genome environment, suggesting it will be useful on more complicated metagenomic sequencing projects.

We show that our corrections are sequencing technology appropriate and agree with published findings. The number and type of errors reported and corrected by our software agrees with alignment evaluations performed by ourselves and Loman *et al.* [27]. We evaluate our software's effect as a preprocessing step before sequence assembly and show that we are able to improve the quality of assemblies on some data sets. We believe our software is a versatile tool that may be used to improve a variety of sequencing applications.

6.1 Future Work

While the current version of the software is effective, there are some areas that may be improved by future refinements. The number of corrected errors could be improved by targeting adjacent errors and other coincident errors that are not homopolymers. Within the homopolymer correction algorithm, we do not correct errors other than homopolymer repeats and thereby ignore all other multinucleotide errors. Correcting these errors would require exploring a larger space of correction possibilities. We recommend disabling homopolyer corrections when correcting exclusively Illumina data and leaving single insertion and deletion corrections enabled. The number of potential homopolymer errors that would not be otherwise removed with single indel corrections is extremely minimal and it is likely to be close to the number of errors that would be introduced. While this strategy is effective, the homopolymer correction might be improved further to reduce the number of miscorrections. There is also room for improvement when correcting the rarer error types

in sequencing technologies. For example, we appear to be slightly overcorrecting substitution errors in GS Junior reads and potentially under-correcting indel errors in MiSeq reads. Additionally, we have a somewhat lower success rate correcting deletion errors than insertion errors across all technologies. This may be a consequence of deletions resulting in fewer k-mer count evaluations and therefore simpler to correct.

We believe that our success correcting a mixed data set lends evidence to our correction software's ability to correct more complicated mixed data sets such as metagenomic data. This is supported by our successful correction of *E. coli* MiSeq data which contains 25x chromosome coverage and 625x plasmid coverage, suggesting our software is able to correct the majority of errors in the presence of highly variable coverage. However, Pollux has some difficulty correcting rare alternatives in the presence of highly abundant alternatives. This might be resolved by placing more emphasis on quality scores and disallowing corrections which change a high-quality base. The memory limitations of Pollux might be resolved by incorporating dedicated k-mer counting software [9, 35, 49]. This inclusion would allow Pollux to operate on larger data sets. However, as these counters are probabilistic, this would be at the expense of correction accuracy.

References

[1] Can Alkan, Saba Sajjadian, and Evan E Eichler. Limitations of Next-Generation Genome Sequence Assembly. *Nature Methods*, 8(1):61–65, 2011.

[2] Serafim Batzoglou, David B Jaffe, Ken Stanley, Jonathan Butler, Sante Gnerre, Evan Mauceli, Bonnie Berger, Jill P Mesirov, and Eric S Lander. ARACHNE: A Whole-Genome Shotgun Assembler. *Genome Research*, 12(1):177–189, 2002.

[3] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya A Shlyakhter, Matthew K Belmonte, Eric S Lander, Chad Nusbaum, and David B Jaffe. ALLPATHS: De Novo Assembly of Whole-Genome Shotgun Microreads. *Genome Research*, 18(5):810–820, 2008.

[4] Mark J Chaisson, Dumitru Brinza, and Pavel A Pevzner. De Novo Fragment Assembly with Short Mate-Paired Reads: Does the Read Length Matter? *Genome Research*, 19(2):336–346, 2009.

[5] Bastien Chevreux. *MIRA: An Automated Genome and EST Assembler*. PhD thesis, German Cancer Research Center Heidelberg, 2005.

[6] International Human Genome Sequencing Consortium. Initial Sequencing and Analysis of the Human Genome. *Nature*, 409(6822):860–921, 2001.

[7] Kristine R Crews, J Kevin Hicks, Ching-Hon Pui, Mary V Relling, and William E Evans. Pharmacogenomics and Individualized Medicine: Translating Science into Practice. *Clinical Pharmacology & Therapeutics*, 92(4):467–475, 2012.

[8] Rami A Dalloul, Julie A Long, Aleksey V Zimin, Luqman Aslam, Kathryn Beal, Le Ann Blomberg, Pascal Bouffard, David W Burt, Oswald Crasta, Richard PMA Crooijmans, et al. Multi-Platform Next-Generation Sequencing of the Domestic Turkey (Meleagris Gallopavo): Genome Assembly and Analysis. *PLoS Biology*, 8(9):e1000475, 2010.

[9] Sebastian Deorowicz, Agnieszka Debudaj-Grabysz, and Szymon Grabowski. Disk-Based kk-mer Counting on a PC. *BMC Bioinformatics*, 14(1):160, 2013.

[10] Narayan Desai, Dion Antonopoulos, Jack A Gilbert, Elizabeth M Glass, and Folker Meyer. From Genomics to Metagenomics. *Current Opinion in Biotechnology*, 23(1):72–76, 2012.

[11] Juliane C Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. SHARCGS, A Fast and Highly Accurate Short-Read Assembly Algorithm for De Novo Genomic Sequencing. *Genome Research*, 17(11):1697–1706, 2007.

[12] Juliane C Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. Substantial Biases in Ultra-Short Read Data Sets from High-Throughput DNA Sequencing. *Nucleic Acids Research*, 36(16):e105–e105, 2008.

[13] Esteban Domingo and John Holland. RNA Virus Mutations and Fitness for Survival. *Annual Reviews in Microbiology*, 51(1):151–178, 1997.

[14] Brent Ewing, LaDeana Hillier, Michael C Wendl, and Phil Green. Base-Calling of Automated Sequencer Traces Using Phred I Accuracy Assessment. *Genome Research*, 8(3):175–185, 1998.

[15] Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J Ribeiro, Joshua N Burton, Bruce J Walker, Ted Sharpe, Giles Hall, Terrance P Shea, Sean Sykes, et al. High-Quality Draft Assemblies of Mammalian Genomes from Massively Parallel Sequence Data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[16] Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. QUAST: Quality Assessment Tool for Genome Assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.

[17] Jo Handelsman. Metagenomics: Application of Genomics to Uncultured Microorganisms. *Microbiology and Molecular Biology Reviews*, 68(4):669–685, 2004.

[18] Joseph Henson, German Tischler, and Zemin Ning. Next-Generation Sequencing and Large Genome Assemblies. *Pharmacogenomics*, 13(8):901–915, 2012.

[19] Yun Heo, Xiao-Long Wu, Deming Chen, Jian Ma, and Wen-Mei Hwu. BLESS: Bloom Filter-Based Error Correction Solution for High-Throughput Sequencing Reads. *Bioinformatics*, page btu030, 2014.

[20] Lucian Ilie and Michael Molnar. RACER: Rapid and Accurate Correction of Errors in Reads. *Bioinformatics*, page btt407, 2013.

[21] Wei-Chun Kao, Andrew H Chan, and Yun S Song. ECHO: A Reference-Free Short-Read Error Correction Algorithm. *Genome Research*, 21(7):1181–1192, 2011.

[22] David R Kelley, Michael C Schatz, Steven L Salzberg, et al. Quake: Quality-Aware Detection and Correction of Sequencing Errors. *Genome Biology*, 11(11):R116, 2010.

[23] Martin Kircher, Udo Stenzel, Janet Kelso, et al. Improved Base Calling for the Illumina Genome Analyzer using Machine Learning Strategies. *Genome Biology*, 10(8):R83, 2009.

[24] Victor Kunin, Alex Copeland, Alla Lapidus, Konstantinos Mavromatis, and Philip Hugenholtz. A Bioinformatician's Guide to Metagenomics. *Microbiology and Molecular Biology Reviews*, 72(4):557–578, 2008.

[25] Ruiqiang Li, Wei Fan, Geng Tian, Hongmei Zhu, Lin He, Jing Cai, Quanfei Huang, Qingle Cai, Bo Li, Yinqi Bai, et al. The Sequence and *de novo* Assembly of the Giant Panda Genome. *Nature*, 463(7279):311–317, 2009.

[26] Yongchao Liu, Jan Schröder, and Bertil Schmidt. Musket: A Multistage k-mer Spectrum-Based Error Corrector for Illumina Sequence Data. *Bioinformatics*, 29(3):308–315, 2013.

[27] Nicholas J Loman, Raju V Misra, Timothy J Dallman, Chrystala Constantinidou, Saheer E Gharbia, John Wain, and Mark J Pallen. Performance Comparison of Benchtop High-Throughput Sequencing Platforms. *Nature Biotechnology*, 30(5):434–439, 2012.

[28] Chengwei Luo, Despina Tsementzi, Nikos Kyrpides, Timothy Read, and Konstantinos T Konstantinidis. Direct Comparisons of Illumina vs. Roche 454 Sequencing Tech-

nologies on the Same Microbial Community DNA Sample. *PLOS ONE*, 7(2):e30087, 2012.

[29] Ashraf G Madian, Heather E Wheeler, Richard Baker Jones, and M Eileen Dolan. Relating Human Genetic Variation to Variation in Drug Responses. *Trends in Genetics*, 28(10):487–495, 2012.

[30] Guillaume Marçais and Carl Kingsford. A Fast, Lock-Free Approach for Efficient Parallel Counting of Occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011.

[31] Elaine R Mardis. Next-Generation DNA Sequencing Methods. *Annual Review of Genomics and Human Genetics*, 9:387–402, 2008.

[32] Marcel Margulies, Michael Egholm, William E Altman, Said Attiya, Joel S Bader, Lisa A Bemben, Jan Berka, Michael S Braverman, Yi-Ju Chen, Zhoutao Chen, et al. Genome Sequencing in Microfabricated High-Density Picolitre Reactors. *Nature*, 437(7057):376–380, 2005.

[33] Jeffrey A Martin and Zhong Wang. Next-Generation Transcriptome Assembly. *Nature Reviews Genetics*, 12(10):671–682, 2011.

[34] Alexander Mellmann, Dag Harmsen, Craig A Cummings, Emily B Zentz, Shana R Leopold, Alain Rico, Karola Prior, Rafael Szczepanowski, Yongmei Ji, Wenlan Zhang, et al. Prospective genomic characterization of the German enterohemorrhagic Escherichia coli O104: H4 outbreak by rapid next generation sequencing technology. *PloS One*, 6(7):e22751, 2011.

[35] Páll Melsted and Jonathan K Pritchard. Efficient Counting of k-mers in DNA Sequences Using a Bloom Filter. *BMC Bioinformatics*, 12(1):333, 2011.

[36] Jason R Miller, Arthur L Delcher, Sergey Koren, Eli Venter, Brian P Walenz, Anushka Brownley, Justin Johnson, Kelvin Li, Clark Mobarry, and Granger Sutton. Aggressive Assembly of Pyrosequencing Reads with Mates. *Bioinformatics*, 24(24):2818–2824, 2008.

[37] Jason R Miller, Sergey Koren, and Granger Sutton. Assembly Algorithms for Next-Generation Sequencing Data. *Genomics*, 95(6):315–327, 2010.

[38] Eugene W Myers, Granger G Sutton, Art L Delcher, Ian M Dew, Dan P Fasulo, Michael J Flanigan, Saul A Kravitz, Clark M Mobarry, Knut HJ Reinert, Karin A Remington, et al. A Whole-Genome Assembly of Drosophila. *Science*, 287(5461):2196–2204, 2000.

[39] Niranjan Nagarajan and Mihai Pop. Sequence Assembly Demystified. *Nature Reviews Genetics*, 14(3):157–167, 2013.

[40] Kensuke Nakamura, Taku Oshima, Takuya Morimoto, Shun Ikeda, Hirofumi Yoshikawa, Yuh Shiwa, Shu Ishikawa, Margaret C Linak, Aki Hirai, Hiroki Takahashi, et al. Sequence-Specific Error Profile of Illumina Sequencers. *Nucleic Acids Research*, page gkr344, 2011.

[41] Thomas D Otto, Mandy Sanders, Matthew Berriman, and Chris Newbold. Iterative Correction of Reference Nucleotides (iCORN) Using Second Generation Sequencing Technology. *Bioinformatics*, 26(14):1704–1707, 2010.

[42] Yu Peng, Henry CM Leung, Siu-Ming Yiu, and Francis YL Chin. IDBA-UD: A De Novo Assembler for Single-Cell and Metagenomic Sequencing Data with Highly Uneven Depth. *Bioinformatics*, 28(11):1420–1428, 2012.

[43] Pavel A Pevzner, Haixu Tang, and Michael S Waterman. An Eulerian Path Approach to DNA Fragment Assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[44] Hannes Ponstingl and Zemin Ning. SMALT - A New Mapper for DNA Sequencing Reads. 2010. Poster presented at Intelligent Systems for Molecular Biology (ISMB), July 11–13, John B. Hynes Convention Center, Boston, USA. Software available at https://www.sanger.ac.uk/resources/software/smalt/.

[45] Michael A Quail, Miriam Smith, Paul Coupland, Thomas D Otto, Simon R Harris, Thomas R Connor, Anna Bertoni, Harold P Swerdlow, and Yong Gu. A Tale of Three Next Generation Sequencing Platforms: Comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq Sequencers. *BMC Genomics*, 13(1):341, 2012.

[46] Jorge S Reis-Filho et al. Next-Generation Sequencing. *Breast Cancer Research*, 11(Suppl 3):S12, 2009.

[47] Roy Ronen, Christina Boucher, Hamidreza Chitsaz, and Pavel Pevzner. SEQuel: Improving the Accuracy of Genome Assemblies. *Bioinformatics*, 28(12):i188–i196, 2012.

[48] Jonathan M Rothberg, Wolfgang Hinz, Todd M Rearick, Jonathan Schultz, William Mileski, Mel Davey, John H Leamon, Kim Johnson, Mark J Milgrew, Matthew Ed-

wards, et al. An Integrated Semiconductor Device Enabling Non-Optical Genome Sequencing. *Nature*, 475(7356):348–352, 2011.

[49] Rajat Shuvro Roy, Debashish Bhattacharya, and Alexander Schliep. Turtle: Identifying Frequent k-mers with Cache-Efficient Algorithms. *Bioinformatics*, page btu132, 2014.

[50] Leena Salmela and Jan Schröder. Correcting Errors in Short Reads by Multiple Alignments. *Bioinformatics*, 27(11):1455–1461, 2011.

[51] Steven L Salzberg, Adam M Phillippy, Aleksey Zimin, Daniela Puiu, Tanja Magoc, Sergey Koren, Todd J Treangen, Michael C Schatz, Arthur L Delcher, Michael Roberts, et al. GAGE: A Critical Evaluation of Genome Assemblies and Assembly Algorithms. *Genome Research*, 22(3):557–567, 2012.

[52] Fred Sanger and Alan R Coulson. A Rapid Method for Determining Sequences in DNA by Primed Synthesis with DNA Polymerase. *Journal of Molecular Biology*, 94(3):441–448, 1975.

[53] Patrick D Schloss, Dirk Gevers, and Sarah L Westcott. Reducing the Effects of PCR Amplification and Sequencing Artifacts on 16S rRNA-Based Studies. *PloS One*, 6(12):e27310, 2011.

[54] Jan Schröder, James Bailey, Thomas Conway, and Justin Zobel. Reference-Free Validation of Short Read Data. *PLOS ONE*, 5(9):e12681, 2010.

[55] Jared T Simpson and Richard Durbin. Efficient De Novo Assembly of Large Genomes Using Compressed Data Structures. *Genome Research*, 22(3):549–556, 2012.

[56] Lloyd M Smith, Jane Z Sanders, Robert J Kaiser, Peter Hughes, Chris Dodd, Charles R Connell, Cheryl Heiner, Stephen BH Kent, and Leroy E Hood. Fluorescence Detection in Automated DNA Sequence Analysis. *Nature*, 321(6071):674–679, 1986.

[57] Susannah Green Tringe, Christian Von Mering, Arthur Kobayashi, Asaf A Salamov, Kevin Chen, Hwai W Chang, Mircea Podar, Jay M Short, Eric J Mathur, John C Detter, et al. Comparative Metagenomics of Microbial Communities. *Science*, 308(5721):554–557, 2005.

[58] René L Warren, Granger G Sutton, Steven JM Jones, and Robert A Holt. Assembling Millions of Short DNA Sequences Using SSAKE. *Bioinformatics*, 23(4):500–501, 2007.

[59] Xiao Yang, Karin S Dorman, and Srinivas Aluru. Reptile: Representative Tiling for Short Read Error Correction. *Bioinformatics*, 26(20):2526–2533, 2010.

[60] Daniel R Zerbino and Ewan Birney. Velvet: Algorithms for De Novo Short Read Assembly Using de Bruijn Graphs. *Genome Research*, 18(5):821–829, 2008.

[61] Zheng Zhang, Scott Schwartz, Lukas Wagner, and Webb Miller. A Greedy Algorithm for Aligning DNA Sequences. *Journal of Computational Biology*, 7(1-2):203–214, 2000.