ASSESSMENT AND CORRECTION OF ERRORS IN DNA SEQUENCING TECHNOLOGIES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Can Fırtına
December 2017

Assessment and correction of errors in DNA sequencing technologies
By Can Fırtına
December 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Can Alkan(Advisor)

A. Ercüment Çiçek

Tolga Can

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

ASSESSMENT AND CORRECTION OF ERRORS IN DNA SEQUENCING TECHNOLOGIES

Can Fırtına
M.S. in Computer Engineering
Advisor: Can Alkan
December 2017

Next Generation Sequencing technologies differ by several parameters where the choice to use whether short or long read sequencing platforms often leads to trade-offs between accuracy and read length. In this thesis, I first demonstrate the problems in reproducibility in analyses using short reads. Our comprehensive analysis on the reproducibility of computational characterization of genomic variants using high throughput sequencing data shows that repeats might be prone to ambiguous mapping. Short reads are more vulnerable to repeats and, thus, may cause reproducibility problems. Next, I introduce a novel algorithm *Hercules*, the first machine learning-based long read error correction algorithm. Several studies require long and accurate reads including *de novo* assembly, fusion and structural variation detection. In such cases researchers often combine both technologies and the more erroneous long reads are corrected using the short reads. Current approaches rely on various graph based alignment techniques and do not take the error profile of the underlying technology into account. Memory- and time- efficient machine learning algorithms that address these shortcomings have the potential to achieve better and more accurate integration of these two technologies. Our algorithm models every long read as a profile Hidden Markov Model with respect to the underlying platform's error profile. The algorithm learns a posterior transition/emission probability distribution for each long read and uses this to correct errors in these reads. Using datasets from two DNA-seq BAC clones (CH17-157L1 and CH17-227A2), and human brain cerebellum polyA RNA-seq, we show that Hercules-corrected reads have the highest mapping rate among all competing algorithms and highest accuracy when most of the basepairs of a long read are covered with short reads.

Keywords: DNA, sequencing, repeats, error correction, long reads, machine learning.

ÖZET

DNA DİZİLİM TEKNOLOJİLERİNDEKİ HATALAR ÜZERİNE DEĞERLENDİRME VE HATALARIN DÜZELTİLMESİ

Can Fırtına
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Can Alkan
Aralık 2017

Yeni Nesil Dizileme teknolojileri birçok değişkende birbirleri arasında farklılık göstermektedir. Kısa parçalı dizileme ya da uzun parçalı dizileme teknolojileri arasında yapılacak bir seçim ise parçaların doğruluk oranı ya da uzunlukları arasında bir tercih gerektirir. Bu tezde ilk olarak, kısa parçaların kullamıyla yapılan analizlerin yeniden üretilebilirliği konusundaki problemleri belirtiyorum. Yeni nesil dizileme teknolojileri kullanılarak genomik farklılıkların karakteristikleri üzerinde yaptığımız geniş çalışma göstermektedir ki tekrarlayan dizileme parçaları, dizilerin muğlak bir şekilde haritalanmasına sebep olabilir. Kısa parçalar tekrarlamaya daha yatkın olduklarından dolayı bu parçaların kullanıldığı deneylerin yeniden üretilebilmesinde problemler yaşanması mümkündür. Bu tezde ikinci olarak, özgün bir algoritma olan *Hercules*'i sunuyorum. *Hercules*, uzun parçalardaki hataların düzeltilmesi için makine öğrenimi tekniğini kullanan ilk algoritmadır. *De novo* yöntemiyle haritalama, yapısal farklılıkların araştırılması gibi birçok araştırma uzun ve hatasız parçaların kullanımını gerektirmektedir. Bu durumlarda, araştırmacılar genellikle uzun parçalardaki hataların düzeltilmesini kısa parçalar ile yapmaktadırlar. Çizge yapısı ve hizalama temelli güncel düzeltme yöntemleri, dizileme teknolojisinin hata profilini göz ardı etmektedirler. Hata profilini el alan, hafıza ve zaman konusunda elverişli makine öğrenimi teknikleri, hataları daha iyi düzeltme ve daha her iki teknolojiyi daha iyi birleştirme konusunda potansiyele sahiptirler. Sunduğumuz algoritma, her bir uzun parçayı, kullanılan teknolojinin hata profiline uygun bir profile Hidden Markov Model'i şeklinde tasarlamaktadır. Algoritmamız, geçiş ve emisyon olasılıklarını bütün uzun parçalar için öğrenip, değiştirerek uzun parçalardaki hataların düzeltilmesini sağlamaktadır. DNA diziliminden iki adet veri dizisi (CH17-157L1 ve CH17-227A2) ve RNA diziliminden bir adet veri dizisi (human brain cerebellum polyA) kullanarak, *Hercules* tarafından hataların giderildiği parçaların, diğer algoritmalar kullanılarak hataların düzeltilmesine kıyasla en yüksek haritalama oranına ve uzun parçaların büyük bölümü kısa parçalarla kaplandığı durumlarda en yüksek hatasızlık oranına sahip olduğunu gösteriyoruz.

Anahtar sözcükler: DNA, dizileme, tekrarlar, hata düzeltme, büyük parça, makine öğrenimi.

Acknowledgement

I would like to express my sincerest gratitude to my supervisor, Asst. Prof. Can Alkan, who has guided and supported me continuously for three years during both my undergraduate and graduate study at Bilkent University. Without his wisdom, patience, and his support, the studies presented in this thesis would have never been completed. I owe him much more than a gratitude for simply being more than just a supervisor: also a friend, and a role-model.

I also would like to thank Asst. Prof. A. Ercument Cicek for suggesting a novel approach for correcting the errors in long reads with the use of profile Hidden Markov Models. His ideas and encouragement led us to introduce a novel method that will hopefully have a good impact on the computational biology field. I also would like to thank him for sharing joyful and insightful conversations with me.

I am also grateful to Asst. Prof. Oznur Tastan for her long-standing support, for being a great supervisor of our senior project, for providing me with the opportunity to be a co-author with her, for her invaluable comments during the preparation of my first published article. I will always consider myself lucky to have my first research experience under her supervision.

I am thankful to Prof. Levent Onural for allocating the generous computational resources for the research we have conducted together. It was a real privilege to conduct a research under his supervision. His wisdom and his guidance have already shaped the path that I took for my research career.

I thank Prof. Tolga Can for giving his precious time to read this thesis.

I acknowledge TÜBİTAK-1001-215E172 grant and a Marie Curie Career Integration Grant (303772).

The last but not the least, I would like to thank my mother and father, Emine and Turan Fırtına, for their endless love, dedication, and their support. I will always be grateful to them for believing in me. I would like to dedicate this thesis to my mother and my father so I can carry a piece from them even when we live far away.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The advancements in Next Generation Sequencing (NGS) technologies have increased the demand on producing genome sequence data for many research questions, and prompted pilot projects to test its power in clinical settings [1]. Any "medical test" to be reliably used in the clinic has to be proven to be both accurate and reproducible. However, the fast-evolving nature of NGS technologies make it difficult to achieve full reproducibility to generate essentially same data using the same DNA resources. NGS technologies suffer from two fundamental limitations. First and foremost, no platform is yet able generate a chromosome-long read. Average read length ranges from 100 bp to 10 Kbp depending on the platform. Second, reads are not error-free. The most ubiquitous platform, Illumina, produces the most accurate (<0.1% error rate), yet the shortest (100-150 bp) reads. Short read lengths present challenges in accurate and reproducible analyses [2, 3], as well as in building reliable assemblies [4, 5, 6]. On the other hand, Pacific Biosciences Single Molecule, Real-Time (SMRT) sequencing technology is capable of producing reads approximately 10 kbp-long on average with a substantially higher (∼15%) error rate [7]. Similarly, Oxford Nanopore Technologies (ONT) platform can generate longer reads (up to ∼900 Kbp) with the price of an even higher error rate (>20%) [8].

In Chapter 2, we first showed that resequencing the same DNA library with the same model NGS instrument twice and analyzing the data with the same algorithms may lead to different variation call sets [9]. There may be multiple reasons for this effect, such as

degradation of DNA between two sequencing experiments, signal processing and base calling errors during sequencing, or different GC biases introduced while making sequencing libraries from the same DNA [9].

Aside from the potential problems in the "wet lab" side, there may be additional complications in the "dry lab" analysis due to alignment errors and ambiguities due to genomic repeats. The repetitive nature of the human genome causes ambiguity in read mapping when the read length is short [2]. A 100 bp read generated by the Illumina platform may align to hundreds of genome locations with similar edit distance. The BWA-MEM [10] mapper's approach to handle such ambiguity is randomly selecting one location, and assigning the mapping quality to zero to inform the variant calling algorithms that the alignment may not be accurate.

Although many algorithms exist for NGS data analysis, only a handful of computational pipelines for read mapping and variant calling may be considered as "standard" such as those that are commonly used in large scale genome projects (i.e. the 1000 Genomes Project [11]). For example, to discover and genotype variants using Illumina data, first the reads are mapped to reference genome assembly using BWA-MEM [10], Bowtie [12], or a similar tool [13, 14] then the alignment files are processed using SAMtools [15] and Picard[1], and finally the single nucleotide variants (SNV) and indels are predicted and filtered using GATK [16], Platypus [17], or Freebayes [18]. Structural variation (SV) discovery is even more challenging as exemplified by the 1000 Genomes Project [11, 19], where more than 20 algorithms were used to characterize SVs.

Recently, the Genome in a Bottle Project [20] was started to set standards for accurate NGS data analysis for both research and clinical uses by addressing the differences in detection performances of different algorithms and different sequencing platforms.

We investigated whether some of the commonly used variant discovery algorithms make use of the mapping quality information, and how they react to genomic repeats. Briefly, we aligned two whole genome shotgun (WGS) data sets, one low and one high coverage genome sequenced as part of the 1000 Genomes Project [11] to the human reference genome (GRCh37) **twice** using the same parameters. In the second mapping, we shuffled the order

[1]http://broadinstitute.github.io/picard/

of reads to make sure that the same random numbers are not used for the same reads. We then generated two SNV and indel call sets each from each genome.

We observed substantial differences in the call sets generated by all of the variant discovery tools we tested except VariationHunter/ CommonLAW. However, VariationHunter explicitly requires a deterministic read mapper, therefore we removed it from further comparisons. GATK's HaplotypeCaller showed discordancies of 1.06% to 1.7% in SNV/indel call sets, where Freebayes showed the most concordancy (up to 99.2%). Genome STRiP showed the greatest discrepancy in structural variation calls (up to 25%). Our results raise questions about reproducibility of callsets generated with several commonly used genomic variation discovery tools.

Our discovery on reproducibility issues of DNA sequencing data suggests that it becomes harder to achieve reproducibility when repeats start to occur frequently. Short reads are vulnerable to repeats as the probability of mapping to a same location with the same edit distance is always higher than that of long reads. This ambiguity (1) raises questions about analyses and results that are produced after short read mapping and (2) increases time complexity of a short read mapping where the long read mapping of the same DNA sample is less time consuming [21]. However, the challenge of using long reads is much more obvious that long sequencing platforms introduce at least 15% error within long reads that they produce [7, 8]. Complementary strengths and weaknesses of these platforms make it attractive to use them in harmony, i.e. merging the advantage of the longer reads generated by PacBio and ONT platforms with the accuracy of Illumina. Arguably, one can achieve high basepair accuracy using only PacBio or ONT reads, if the sequence coverage is sufficiently high [22]. However, the relatively higher costs associated with long read platforms make this approach prohibitive. Therefore, it is still more feasible to correct the errors in long reads using more accurate short reads in a hybrid manner.

There are two major hybrid approaches for long read error correction. First approach starts with aligning short reads onto long reads generated from the same sample, implemented by several tools such as PacBioToCA [23], LSC [24], proovread [25], and Colormap [26]. Leveraging the relatively higher coverage and accuracy of short reads, these algorithms fix the errors in long reads by calculating a consensus of the short reads over the same segment of the nucleic acid sequence. The second approach aligns long reads over a de Bruijn graph

constructed using short reads, and the k-mers on the de Bruijn graph that are connected with a long read are then merged into a new, corrected form of the long read. Examples of this approach are LoRDEC [27], Jabba [28], HALC [29], and LoRMA [30].

The alignment based approach is highly dependent on the performance of the aligner. Therefore, accuracy, running time, and memory usage of an aligner will directly affect the performance of the downstream correction tool. The use of de Bruijn graphs eliminates the dependence on external aligners, and implicitly moves the consensus calculation step into the graph construction. However, even with the very low error rate in short reads and the use of shorter k-mers when building the graph, the resulting de Bruijn graph may contain bulges and tips, that are typically treated as errors and removed [31]. Accurate removal of such graph elements depend on the availability of high coverage data to be able to confidently discriminate real edges from the ones caused by erroneous k-mers [32, 33]. Therefore, performance of de Bruijn graph based methods depends on high sequence coverage.

Here, in Chapter 3, we introduce a new alignment-based hybrid error correction algorithm, *Hercules* [34], to improve basepair accuracy of long reads. Hercules is the first machine learning-based long read error correction algorithm. The main advantage of Hercules over other alignment-based tools is the novel use of profile hidden Markov models (dubbed profile HMM or pHMM). Hercules models each long and erroneous read as a template profile HMM. It uses short reads as observations to train the model via Forward-Backward algorithm [35], and learns posterior transition and emission probabilities. Finally, Hercules decodes the most probable sequence for each profile HMM using Viterbi algorithm [36]. Profile HMMs are first popularized by the HMMER alignment tool for protein families [37] and are used to calculate the likelihood of a protein being a member of a family. To the best of our knowledge, this is the first use of a profile HMM as template with the goal of learning the posterior probability distributions.

All alignment-based methods in the literature, (i) are dependent on the aligner's full CIGAR string for each basepair to correct, (ii) have to perform majority voting to resolve discrepancies among short reads, and (iii) assume all error types (i.e., deletion) are equally likely to happen. Whereas, Hercules only takes the starting positions of the aligned short reads from the aligner, which minimizes the dependency on the aligner. Then, starting from

that position, it sequentially and probabilistically accounts for the evidence provided per short read, instead of just independently using majority voting per base-pair. Finally, prior probabilities for error types can be configured based on the error profile of the platform to be processed. As prior probabilities are not uniform, the algorithm is better positioned to predict the posterior outcome. Thus, it can also distinguish long read technologies.

We tested Hercules on the following datasets: (i) two BAC clones of complex regions of human chromosome 17, namely, CH17-157L1 and CH17-227A2 [7], and (ii) human brain cerebellum polyA RNA-seq data that LSC provides [24]. As the ground truth, (i) for BAC clones, we used finished assemblies generated from Sanger sequencing data from the same samples and (ii) for human brain data we used GRCh38 assembly. We show that Hercules-corrected reads has the highest mapping rate among all competing algorithms. We report $\sim$20% improvement over the closest de Bruijn based method, $\sim$2% improvement over the closest alignment-based method, and $\sim$30% improvement over the original version of long reads. We also show that, should long reads have high short read breadth of coverage provided by the aligner (i.e., 90%), Hercules outputs the largest set of most accurate reads (i.e., $\geq$95% accuracy).

Chapter 2

Effects of read length and genomic repeats on the reproducibility of genomic analyses

2.1 Methods

2.1.1 Data acquisition

We downloaded both whole genome and whole exome sequencing data sets from the 1000 Genomes Project [11] FTP server.

2.1.2 Read mapping, shuffling, and BAM file processing

We used Bowtie [12], RazerS3 [14], and BWA-MEM [10], and mrFAST [13] (Table S2) to align the reads generated by the Illumina platform to human reference genome (GRCh37) using default options. For testing the effects of read order, we randomly shuffled the reads in the FASTQ file using an in-house program, while keeping the relative order of read pairs

intact. The reason for reshuffling the reads is the following. In our small scale test, we noticed that BWA-MEM uses the same pseudorandom number generator seed in all mapping experiments. This causes the same ambiguously mapping read to be randomly assigned to the same position when the read order is intact. However, when we shuffle the reads, the random number that corresponds to the read changes, causing it to be placed to another random location. Note that, the DNA molecules are hybridized randomly to the oligos on the flow cell, thus, our read randomization simulates the randomness in cluster generation. Next, we used SAMtools [15] to merge, sort, and index BAM files, and Picard[1] to remove PCR duplicates (MarkDuplicates). We then followed the GATK's "best practices" guide [38] to realign around indels (RealignerTargetCreator and IndelRealigner) and recalibrate base quality values (BaseRecalibrator). We used the resulting BAM files for SNV, indel, and SV calling. The names and version numbers of the tools we used are listed in Table S2.

2.1.3 SNVs and indels

We used GATK HaplotypeCaller [16], SAMtools [15], Freebayes [18], and Platypus [17] to characterize SNV and indels. We followed the developers' recommendations and default parameters for all variant calling tools, including potential false positive filters. Specifically, we used both Variant Quality Score Recalibrator and SnpCluster methods to filter out false positives in GATK call sets, and for other tools we required a variant quality of at least 30. For GATK, we used the GATK Resource Bundle version 2.8 as the reference genome and its annotations, and variant score recalibration training material.

2.1.4 Structural variation

For structural variation discovery using the BWA-generated BAM files, we tested the reproducibility of the calls produced by DELLY [39], LUMPY [40], Genome STRiP [41], and VariationHunter/ CommonLAW [42, 43]. We note that VariationHunter explicitly remaps reads to the reference genome using mrFAST, which is a deterministic mapper, therefore we removed it from further comparisons. We used default parameters for each tool and followed recommendations in relative documentations.

2.1.5 Variant annotation and comparison

We downloaded the coordinates for segmental duplications, genes, coding exons, and common repeats from the UCSC Genome Browser [44]. We then used the BEDtools suite [45] and standard UNIX tools to calculate the discrepancies among the call sets and their underlying sequence annotations.

2.2 Results

2.2.1 Data and tools

We downloaded two WGS data sets, one at low coverage (∼5X, HG00096) and one at high coverage (∼ 44X, HG02107), and 12 WES data sets with coverage ranging from 120X to 656X from the 1000 Genomes Project [11] (Table S1). We tested the behaviors of three different read mappers, four SNV/indel callers, and three SV characterization algorithms (Table S2, Methods).

2.2.2 Small scale test for ambiguous mapping

We first sub sampled 1 million reads from HG00096, and mapped it to the human reference genome (GRCh37) using Bowtie, RazerS3, mrFAST, and BWA-MEM [10]. Next, we randomly shuffled the reads in the FASTQ file (Methods), and remapped the reordered reads to GRCh37 using the same tools. The read randomization simulates the random nature of DNA hybridization on the flow cell. We confirmed that mrFAST and Bowtie generated the same alignments, as described in their respective documentations, where BWA-MEM mapped several reads to different locations due to placing such reads to random locations (Table S3).

2.2.3 Read mapping in parallel

Due to the large number of reads generated by NGS platforms, it is a common practice to use scatter/gather operations (or, its implementation using the MapReduce framework) to distribute the work load to large number of CPUs in a cluster. This approach leverages the embarrassingly parallel nature of read mapping, where the FASTQ files that typically contain >50 million reads are divided into "chunks" with just 1-2 million reads per file, the reads in each chunk are mapped separately, and the resulting BAM files are combined. Reasoning from our observation of different random placements of ambiguous reads when the reads are shuffled, we employed the scatter/gather method to map 1 million reads twice, using different chunk sizes. In this experiment we divided the reads into chunks of 50,000 and 100,000 read pairs, mapped them using BWA-MEM, and observed mapping discordance ratios similar to that of random shuffling (2.1%, Table S4). We also observed less pronounced differences in read mapping when different number of threads are used for the same FASTQ file (0.05%, Table S5).

2.2.4 WGS analysis

We then repeated the same mapping strategy to the full versions of all data sets we downloaded, but we mapped using only BWA-MEM, since we observed the other mappers to be deterministic based on the small scale test. We also investigated BWA-MEM's behavior of random placements using the HG00096 genome, and interestingly, although BWA-MEM reported zero mapping qualities for most of the discrepant read mappings (∼97%), it also assigned high MAPQ values (≥30) for a fraction of them (∼0.75%; Table S6).

2.2.5 Single nucleotide variants and indels

We used GATK's HaplotypeCaller, Freebayes, Platypus, and SAMtools to characterize SNVs and indels within the HG00096 and HG02107 genomes using recommended parameters for each tool (Methods). We did not evaluate GATK UnifiedGenotyper since it is deprecated by its developers. We then compared each call set generated by the same tools using the

reads in original vs. shuffled order using BEDtools [45], and found up to 1.70% of variants to be called in one alignment of the same data but not in the other (Table 2.1). Next, we investigated the underlying sequence context of the SNVs and indels differently detected using the same tools with two different alignments (i.e. original vs shuffled order). As expected, 72- to 80% of the discrepant calls were found within common repeats and segmental duplications (Tables S7,S9,S10,S11). In most genomic analysis studies duplications and repeats are removed from analyses; however, in this study we observed discrepancies in functionally important regions (i.e. coding exons). For example, 253-1,249 SNVs that were called from one alignment but not another map to coding exons (Table S7). Furthermore, 1,543 of the 1,884 (81.9%) discordant exonic SNVs predicted by GATK HaplotypeCaller (either original or shuffled order, non-reduntant total) **did not** intersect with any common repeats or segmental duplications.

Freebayes, Platypus, and SAMtools predictions were more reproducible, as >98.5% of the calls were identical, and the number of exonic discrepant SNV calls were substantially lower than that of GATK's (Tables S9,S10,S11).

2.2.6 Structural variation

Next, we analyzed the deletion calls predicted using DELLY, LUMPY, and Genome STRiP. All three SV detection tools we tested showed 1.55%- to 25.01% difference in call sets using the original vs. shuffled order read data sets (Table 2.1). Similarly, the discrepancies were mostly found within repeats and duplications, however, only a couple of deletion calls intersected with coding exons (Tables S12, S16, S17).

Using DELLY we predicted ∼3% of deletion, ∼4% of tandem duplication, ∼6% of inversion, and ∼3.6% of translocation calls to be specific to a single alignment, and >91% of these differences intersected with common repeats. Owing to the difficulties in predicting these types of SVs, more discrepant calls intersected with functionally important regions (i.e. genes and coding exons; Tables S13,S14,S15).

Table 2.1: Summary of SNV, small indel, and deletion calls.

Tool	**HG00096**				
	Original		Shuffled		*Diff (%)*
	All	*Private*	*All*	*Private*	
HaplotypeCaller	2,279,678	10,898	2,294,808	26,028	1.06
Freebayes	2,400,545	992	2,400,595	1,042	0.08
SAMtools	2,277,691	2,683	2,277,674	2,666	0.24
Platypus	2,022,412	2,342	2,022,294	2,224	0.23
DELLY	1,325	37	1,323	35	5.29
LUMPY	1,366	12	1,363	9	1.55
Genome STRiP	1,218	25	1,212	25	4.04
	HG02107				
	Original		Shuffled		*Diff (%)*
	All	*Private*	*All*	*Private*	
HaplotypeCaller	4,654,338	54,051	4,625,648	25,361	1.70
Freebayes	5,174,644	4,715	5,189,285	19,356	0.46
SAMtools	5,355,604	9,838	5,355,053	9,287	0.36
Platypus	4,642,336	6,200	4,642,300	6,164	0.27
DELLY	13,517	831	13,505	819	11.51
LUMPY	9,786	182	9,853	249	4.49
Genome STRiP	3,452	482	3,477	508	25.01

We list the number of SNV, small indel, and deletion calls in the genomes of HG00096 and HG02107 characterized by different tools using the reads in the original (i.e. as released by 1000 Genomes Project), and shuffled order. Calls that are specific to one order of reads are listed as *Private*. The difference percentage is calculated as the total number of *Private* calls divided by the number of calls in the union set (i.e. $\frac{|(O \setminus S) \cup (S \setminus O)|}{|O \cup S|}$, O: original, S: shuffled).

2.2.7 Reusing the same alignments

More interestingly, when we ran GATK's HaplotypeCaller on the same BAM file twice we observed discrepant calls similar to using two different BAM files generated from original vs shuffled read order (Table S18). Other tools produced no discrepancies (Tables S20-S28). Detailed analysis of these discordancies revealed that 21,497 of the 21,510 (>99.9%) "second-run specific" HaplotypeCaller calls were initially found in the first run, however filtered in the Variant Quality Score Recalibration (VQSR) step. Similarly, 10,631 of the 10,646 "first-run specific" HaplotypeCaller calls were eliminated by VQSR in the second run. We

then performed a line-by-line analysis in such calls, and found that the *VQSLOD* score was calculated differently, although the training data was the same in both runs. We speculate that this is due to the random sampling of the training data to reduce computational burden[1]. We then confirmed our observation by rerunning the VQSR filter on one of the VCF files five times. Each iteration of the VQSR filtering generated a different set of VQSLOD values, causing different variants to be filtered.

2.2.8 Exome analysis

Finally, we tested the effect of discordant call sets generated by GATK even with the same alignment files using 12 whole exome shotgun sequence (WES) data sets from the 1000 Genomes Project (Table S1). We followed the same alignment, post-processing for the WES data sets. We then generated two call sets each using HaplotypeCaller on the same BAM files, followed with VQSR filtering. In this experiment, we used the multi-sample calling options. HaplotypeCaller produced discordant calls at 1-to 3% rate (Tables S18,S29).

2.3 Discussion

The introduction of NGS platforms quickly revolutionized the way we do biological research[46, 47]. Furthermore, NGS technology already started to make impact on human health in the form of personalized medicine through clinical sequencing [1], breast cancer subtype detection [48], and small molecule drug target site identification [49]. However, both sequencing technologies themselves [9], and the computational tools are still far from being mature in terms of accuracy [50, 51]. The complexity and repetitive nature of human genomes introduce further challenges for reliable characterization of genomic variants [2].

In this chapter, we documented the effects of different approaches to handle ambiguities in read mapping due to genomic repeats. We focused on more widely used computational tools for read mapping and variant calling, and observed that random placement of ambiguously

[1]This random subsampling can be seen in the GATK code VariantDataManager.java at `https://github.com/broadgsa/gatk-protected/` (commit ID: `8ea4dcab8d78e7a7d573fcdc519bd0947a875c06`, line 255)

mapping reads have an effect on called variants. Although discordancies within repeats are less of a concern due to their relatively negligible effects to phenotype, we also discovered hundreds to thousands variants differently detected within coding exons. HaplotypeCaller showed the most discrepancies, where the discordant calls were less pronounced in Freebayes and Platypus results. Using the same alignments twice, we found that the callers themselves are deterministic, however, they return different call sets when the same data is remapped. Interestingly, we observed differences in call sets generated using HaplotypeCaller even when the same alignments and variant filtration training data sets were provided. Although we could not fully characterize the reasons of this observation with GATK, since HaplotypeCaller algorithm is yet unpublished, we observed that the differences were mainly due to differences in calculation of the *VQSLOD* score by the VQSR filter (Results). Therefore, a second source of randomness we observed is within the training step of the VQSR filter, which is specific to GATK.

2.3.1 Recommendations

We, point out that randomized algorithms may achieve better accuracy in practice, albeit without 100% reproducibility. Full reproducibility could only be achieved through using deterministic methods. Therefore, for full reproducibility, we recommend to opt for a deterministic read mapper such as Bowtie, mrFAST, etc., and a deterministic variant caller, such as Platypus or Freebayes for SNV and indels. We note that all SV calling algorithms we surveyed in our study are deterministic algorithms, therefore the SV call sets can be fully reproducible when they are used together with a deterministic mapper. Another approach may be more strict filtering of variants that map to repeats and duplications, however this may result in lower detection power in functionally important duplicated genes such as the MHC and KIR loci. It may be possible to work around the GATK's VQSLOD calculation problem outlined above by setting the `maxNumTrainingData` parameter and other downsampling parameters to high values, however, we recommend disabling these randomizations by default to be a better practice for uninformed users. In our tests, changing only the `maxNumTrainingData` parameter did not fully resolve the variant filtration problem, which points that there may be other downsampling and/or randomization step within the VQSR filter.

2.3.2 Conclusion

Mapping short reads to repetitive regions accurately still remains an open problem [2]. Bowtie and mrFAST use edit distance and paired-end span distance to deterministically assign a single "best" map location to ambiguously mapping reads, where BWA-MEM selects a random map location all mapping properties are calculated the same. BWA-MEM assigns a zero mapping quality to such randomly selected alignments. This approach is still valid since it informs the downstream analysis tools for problematic alignments, however, as we have documented in this manuscript, several variant discovery tools do not fully utilize this information. Complete analysis of the reasons for these discrepancies may warrant code inspection and full disclosure of every algorithmic detail.

The differences in call sets we observed in this study have similar accuracy when compared to 1000 Genomes data (Supplementary Table S30, S31). In addition a recent study did not find any significant difference between deterministic and non-deterministic mappers in terms of accuracy [52]. It is still expected to have differences between different algorithms and/or parameters, but obtaining different results should not be due to the order of *independently generated* reads in the input file. We may simply count these discordancies as false positives and negatives, and such discordancies may not have any adverse effects in practice, however, we argue that computational predictions should not be affected by luck, and inaccuracies in computational results should be deterministic so they can be better understood and characterized. We are in exciting times in biological research thanks to the development of NGS technologies. However, under the shining lights of the discoveries we make in this "big biology" revolution, it can be easy to overlook that the methods matter. No genomic variant characterization algorithm achieves 100% accuracy yet, even with simulation data, but it is only possible to analyze and understand the shortcomings of deterministic algorithms, and impossible to fully understand how an algorithm performs if it makes random choices.

Chapter 3

Error correction for long reads

3.1 Methods

3.1.1 Overview of Hercules

Hercules [34] corrects errors (insertions, deletions, and substitutions) present in long read sequencing platforms such as PacBio SMRT [53] and Oxford Nanopore Technologies [8], using reads from a more accurate orthogonal sequencing platform, such as Illumina [54]. We refer to reads from the former are referred as "long reads" and the latter as "short reads" in the remainder of this chapter. The algorithm starts with preprocessing the data and obtains the short-to-long read alignment. Then, for each long read, Hercules constructs a pHMM template using the error profile of the underlying platform as priors. It then uses the Forward-Backward algorithm to learn the posterior transition/emission probabilities, and finally, uses the Viterbi algorithm to decode the pHMM to output the corrected long read (Figure 3.1).

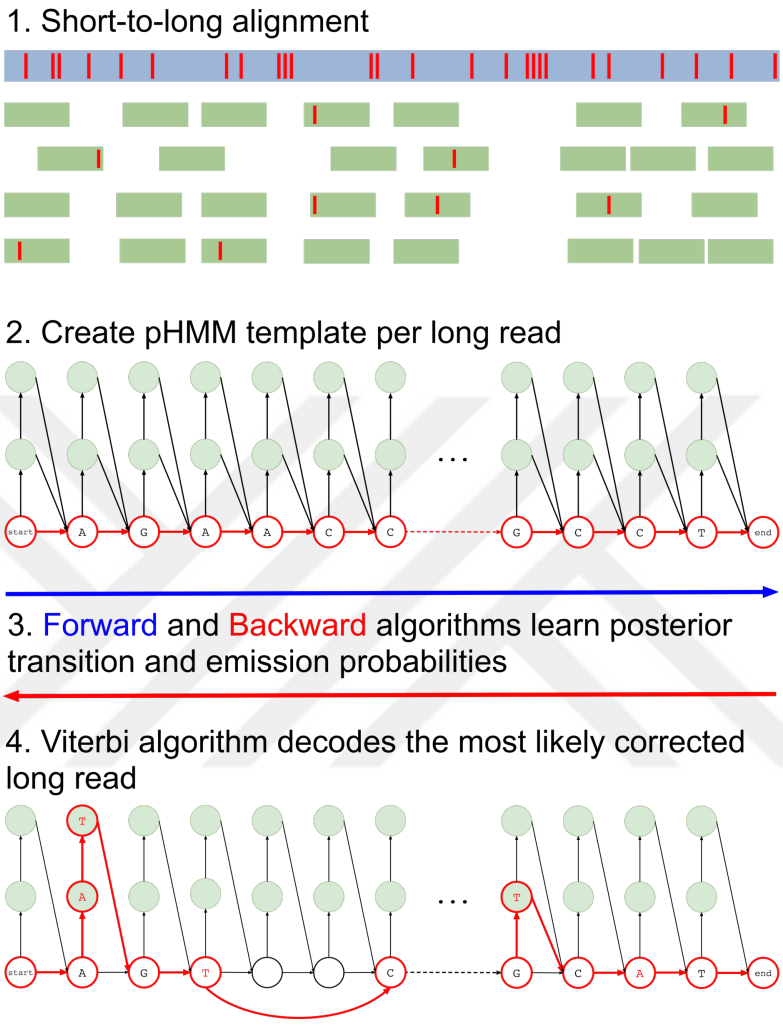

Figure 3.1: Overview of the Hercules algorithm. At the first step (1), short reads are aligned to long reads using an external tool. Here, red bars on the reads correspond to erroneous locations. Then, for each long read Hercules creates a pHMM with priors set according to the underlying technology as shown in (2). Using the starting positions of the aligned short reads, Forward-Backward algorithm learns posterior transition and emission probabilities. Finally, Viterbi algorithm finds the most likely path of transitions and emissions as highlighted with red colors in (3). The prefix and the suffix of the input long read in this example is "AGAACC...GCCT". After correction, substring "AT" inserted right after the first "A". Third "A" is changed to "T" and following two base-pairs are deleted. Note that deletion transitions are omitted other than this arrow, and only two insertion states are shown for clarity of the figure. On the suffix, a "T" is inserted and second to last base-pair is changed from "C" to "A".

3.1.2 Preprocessing

3.1.2.1 Compression

Similar to the approach of LSC [24], Hercules starts with compressing both short and long reads using a run-length encoding scheme. That is, it compresses repeating base-pairs to a single base-pair (e.g., AAACTGGGAC $\rightarrow$ ACTGAC). This is because PacBio and ONT platforms are known to produce erroneous homopolymer runs, i.e. consecutive bases may be erroneously repeated [53, 8], which affects the short read alignment performance drastically. Hercules, then, recalculates the original positions based on the original long reads. Even though this option is performed by default, it is still possible to skip the compression phase. Hercules obtains 3-fold increase in the most accurate reads (i.e., >95% accuracy) when compression phase is enabled. Doing so, the number of aligned reads also increases by ~30%.

3.1.2.2 Short Read Filtering

After the step described above, the nominal length of some compressed reads may be substantially shortened. This would in turn cause such reads to be ambiguously aligned to the long reads, which is already a common problem in short read sequencing due to repeats [2, 3]. To reduce this burden, Hercules removes any compressed short read, if its number of non-N characters is less than a specified threshold (set to 40 by default). In addition to PCR and optical duplicates inherent in short read data, it is also highly likely that compression step will generate new duplicate sequences. These duplicate reads will not carry any extra information for correcting the long reads. Therefore, after compression, Hercules may remove these duplicates using a Bloom filter with 0.0001 false positive rate, keeping only a single read as the representative of the group. Duplicate removal is an optional phase, and we do not remove duplicates by default.

3.1.2.3 Alignment and Decompression

Hercules is an alignment-based error correction tool. It outsources the alignment process to a third party aligner. It is compatible with any aligner that provides output in SAM format [15]. However, we suggest mappers that can report multiple possible alignments such as Bowtie2 [55] as we would prefer to cover most of the long read with short reads. Even though this might cause incorrect mappings, Hercules is able to probabilistically downplay the importance of such reads during the learning stage given sufficient short read depth. Hercules also assumes that the resulting alignment file is sorted. Thus, the file in SAM/BAM format must be coordinate sorted in order to ensure a proper correction with Hercules. Alignments are calculated using either compressed or original long and short reads (those that pass filtering step for the latter) as described in previous sections. After receiving the alignment positions from the aligner, Hercules decompresses both short and long reads and recalculates alignment start locations.

All other alignment-based error correction methods in the literature use the CIGAR string provided by the aligner. CIGAR string specifies where each basepair of the short read is mapped on the long read and where insertions and deletions have occurred. Then, per base-pair on the long read, they perform a majority voting among covering short reads. Hence, *learning* of correct basepairs is actually performed by the aligner, which makes the performance of such tools fully dependent on the aligner choice. Even though, Hercules is also an alignment-based method, the only information it receives from the aligner is the starting position of the mapping. Forward algorithm learns posterior probabilities starting from that position and it can go beyond the covered region. Thus, despite using the starting point information from the aligner, the algorithm can decide the short read is aligned after that point and also that the alignment is essentially different than what is claimed by the aligner in the CIGAR string. This minimizes the dependency of the algorithm on the aligner and makes Hercules the first alignment-based approach to reclaim the consensus learning procedure from the aligner.

3.1.3 The Profile HMM Structure

Hercules models each long read as a pHMM. In a traditional profile HMM [37], there are three types of states: deletion, insertion and match (mismatch) states. The aim is to represent a family of proteins (amino acid sequences) and then to use it to decide if an unknown protein is likely to be generated from this model (family). Our goal in representing a long read as a *template* pHMM is entirely different. For the first time, we use short reads that we know are generated from the source (e.g., same section of RNA/DNA), to update the model (not the topology but the transition and emission probabilities). So, the goal in the standard application is to calculate the likelihood of a *given* sequence, whereas in our application there is no given sequence. We would like to calculate the most likely sequence the model would produce, among all possible sequences that can be produced using the letters in the alphabet Σ. This requires us to impose some restrictions on the standard pHMM concept. First, we remove the self loop over insertion states. Instead, our model has multiple insertions states per position (basepair) on the long read. The number of insertion states is an input the algorithm. Second, we substitute deletion states with *deletion transitions*. The number of possible consecutive basepairs that can be deleted is an input parameter to the algorithm as well.

After we construct the model, we use the error profile of the underlying technology to initialize the prior transition and emission probabilities. Hercules first learns posterior transition and emission probabilities of each "long read profile HMM" using short reads that align to the corresponding long read. Then, it decodes the pHMM using the Viterbi algorithm [36] to generate the most probable (i.e. corrected, or consensus) version of the long read.

It should be noted that the meanings of insertion and deletion is reversed in the context of the pHMM. That is, an insertion state would insert a basepair into the erroneous long read. However, this means that the original long read did not have that nucleotide and had a deletion in that position. Same principle also applies to deletions.

Next, we first define the structure of the model (states and transitions) and explain how we handle different types of errors (i.e. substitutions, deletions, and insertions). We, then, describe the training process. Finally, we explain how Hercules decodes the model and

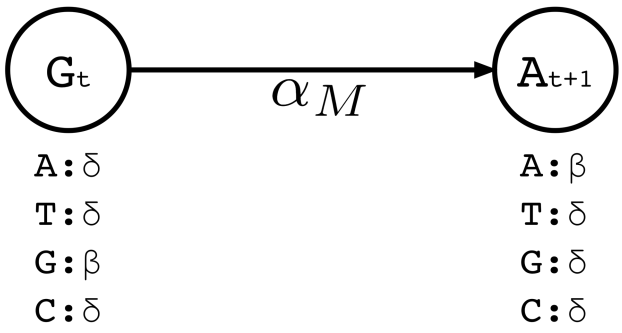

Figure 3.2: A small portion of the profile HMM built by Hercules. Here, two match states are shown, where the corresponding long read includes G at location t, followed by nucleotide A at location $t+1$. At state t, the emission probability for G is set to β, and emission probabilities for A, C, T are each set to the substitution error rate δ. Match transition probability between states t and $t+1$ is initialized to α_M.

generates the consensus sequence.

3.1.3.1 Match States

Similar to traditional pHMM, we represent each basepair in the long read by a *match* state. There are four emission possibilities ($\Sigma = \{A, C, G, T\}$). The basepair that is observed in the uncorrected long read for that position t is initialized with the highest emission probability (β), while the probabilities for emitting the other three basepairs are set to the expected substitution error rate for the long read sequencing technology (δ) (Note that $\beta \gg \delta$ and $\beta + 3\delta = 1$).

We then set transition probabilities between consecutive match states t and $t+1$ as "match transitions" with probability α_M. Figure 3.2 exemplifies a small portion of the profile HMM that shows only the match states and match transitions. From a match state at position t, there are also transitions to (i) the first insertion state at position t (I_t^1), and (ii) to all match states at positions $t+1+x$ where $1 \leq x \leq k$ and k determines the number of possible deletions.

3.1.3.2 Insertion States

Insertion states have self-loop transitions in standard profile HMMs to allow for multiple insertions at the same site, which creates ambiguity for error correction for two reasons. First, self-loops do not explicitly specify the maximum number of insertions. Thus, it is not possible predetermine how many times that a particular self-loop will be followed while decoding. Second, each iteration over the loop has to emit the same basepair. Since decoding phase prefers most probable basepair at each state, it is not possible for a standard profile HMM to choose different nucleotides from the same insertion state.

Instead of a single insertion state with a self loop per basepair in the long read, we construct l multiple insertion states for every match state at position t (e.g., $I_t^1 \dots I_t^l$). We replace self-loops with transitions between consecutive insertion states ($I_t^j \dots I_t^{j+1}$) for position t (see Figure 3.3). The probability for each of such transitions is α_I and the number of insertion states per position l determines the maximum number of allowed insertions between two match states, which are set through user-specified parameters. All insertion states at position t have transitions to the match state at positions $t+1$ (end states is also considered as a match state). For $I_t^1 \dots I_t^{l-1}$ the probability of those transitions are α_M, and for I_t^l, it is $\alpha_M + \alpha_I$. All those states also have transitions to all match states at positions $t+1+x$ where $1 \leq x \leq k$ and k determines the number of possible deletions.

We also set emission probabilities for the insertion states and assume they are equally likely. However, for the insertion states at position t, we set the probability of emitting the nucleotide $X \in \Sigma$ to zero, if X is the most likely nucleotide at the match state of position $t+1$. This makes the insertion more likely to happen at the last basepair of the homopolymer run. Otherwise, the likelihood of inserting a basepair is shared among all insertions states of the run, and it is less likely for any them to be selected during the decoding phase. All other basepairs have their emission probabilities set to 0.33 (i.e. $\frac{1}{|\Sigma|-1}$)- see Figure 3.3.

3.1.3.3 Deletion Transitions

In standard pHMM, a deletion state needs to be visited to skip (delete) a basepair, which does do not emit any characters. However, as described in the following sections, calculation

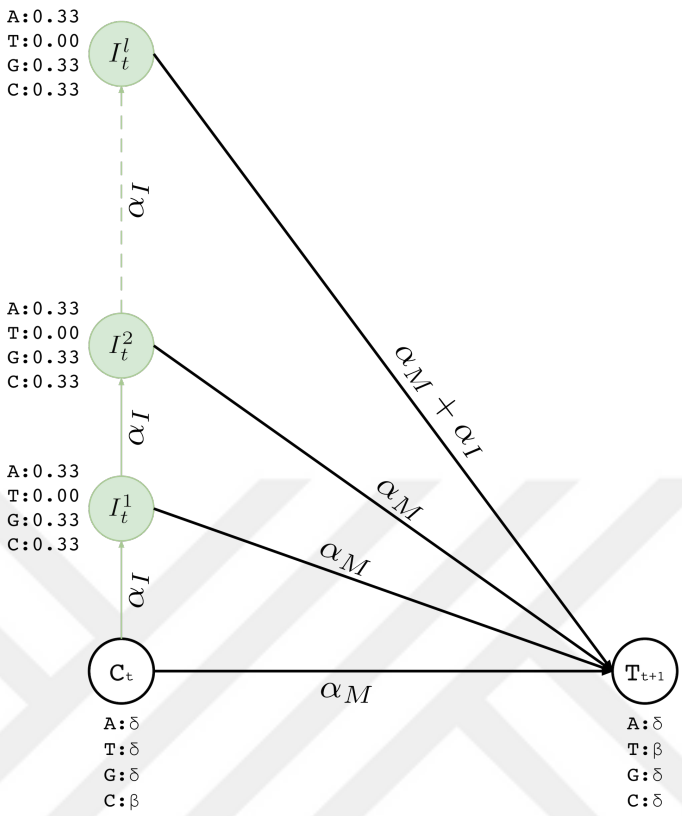

Figure 3.3: Insertion states for position t. Here we show two match states (C_t and T_{t+1}) and l insertion states ($I_t^1 \dots I_t^l$). The number of insertion states limit the insertion length to at most l after basepair t of the corresponding long read. We also incorporate equal emission probabilities for all basepairs, except for the basepair represented by the corresponding match state $t+1$ (T in this example).

of the forward and backward probabilities for each state is based on the basepair of the short read considered at that time. This means at each step, the Forward-Backward algorithm *consumes* a basepair of the short read to account for the evidence the short read provides for that state. As deletion states do not consume any basepairs of the short read, this in turn results in an inflation on the number of possibilities to consider and substantially increases the computation time. In an extreme case, the short read implying its mapped region on the long read may be deleted entirely. Since we are using an external aligner, such extreme cases are unlikely. Thus, we model deletions as transitions, instead of having deletion states. In our model, a transitions to $(1+x)$-step away match states is established to delete x basepairs. As shown in Figure 3.4, match and insertion states at t^{th} position have a transition to all match states at positions $t+1+x$, where $1 \leq x \leq k$ and k is an input parameter determining maximum number of deletions per transition. We calculate the probability of a deletion of

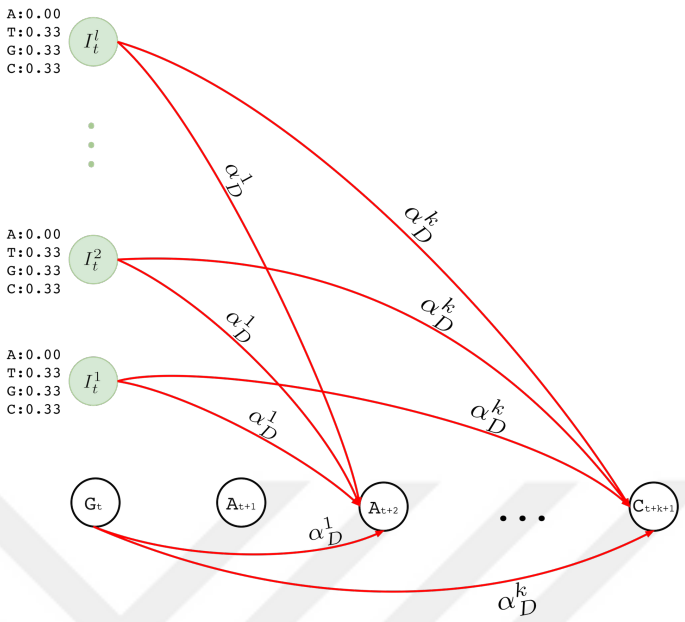

Figure 3.4: Deletion transitions (red) in Hercules pHMM. Insertion states of position t have the same deletion transitions with the match state of the same position, t. Any transition from position t to $t+1+x$ removes x characters, skipping the match states between t and $t+1+x$, with a_D^x probability, where $1 \leq x \leq k$

x basepairs, α_D^x, as shown in Equation 1.

$$\alpha_D^x = \frac{f^{k-x}\alpha_{del}}{\sum_{j=0}^{k-1} f^j} \quad 1 \leq x \leq k \tag{1}$$

Equation 1 is a normalized version of a polynomial distribution where α_{del} is the overall deletion probability (i.e. $\alpha_{del} = 1 - \alpha_M - \alpha_I$), and $f \in [1, \infty)$. As f value increases, probabilities of further deletions decrease accordingly.

3.1.3.4 Training

An overall illustration of a complete pHMM for a long read of length n is shown in Figure 3.5 where (i) match states are labeled with M_t where $M \in \Sigma$, (ii) insertion states are labeled with I_t^1 and I_t^2, and (iii) deletion transitions for $k = 1$ are shown. Note that t^{th} basepair of a long read has one match state and two insertion states where $1 \leq t \leq n$. There are also

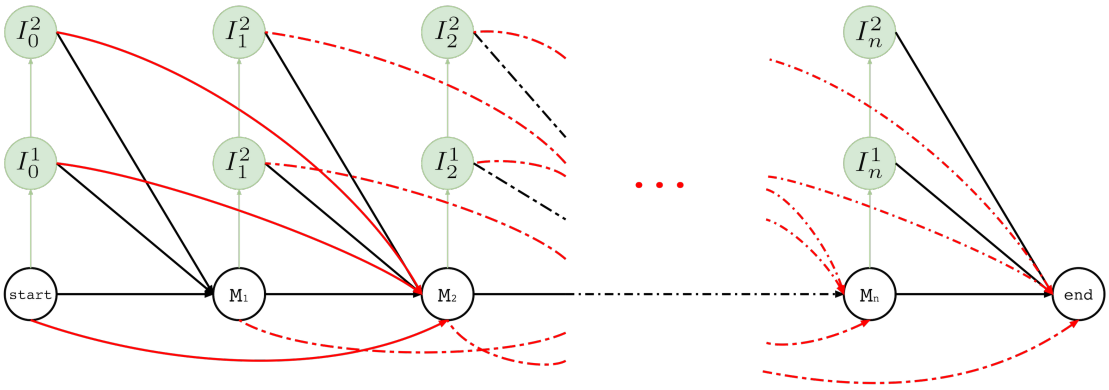

Figure 3.5: Hercules profile HMM in full. Here we show the overall look at the complete graph that might be produced for a long read M where its t^{th} character is M_t and $M_t \in \{A, T, G, C\}$ $1 \leq t \leq n$ and n is the length of the long read M (i.e. $|M| = n$). In this example, there are n many match states and two insertion states for each match state. Only one character deletion is allowed at one transition because transitions may only skip one match state.

deletion transitions from every state at t^{th} position to $(t+2)^{th}$ match state where $(t+2) \leq n$. This example structure allows only one character deletion at one transition since it is only capable of skipping one match state.

Let the complete pHMM for a long read be the graph $G(V, E)$. Per each mapped short read s, we extract a sub-graph $G_s(V_s, E_s)$. Here G_s corresponds to the covered region of the long read with that short read. We train each G_s using the Forward-Backward algorithm [35]. Hercules may not consider some of the short reads that align to same position in a long read to reduce computational burden. We provide *max coverage*, *mc*, option to only consider *mc* many number of short reads for a position during a correction, where $mc = 20$ by default. Short reads are selected based on their edit distance.

States that will be included in V_s are determined by several factors. First, assume that s is aligned to start from the q^{th} character of the long read. First, all match and insertion states between positions $[q - 1,\ q + m)$ are included, where m is the length of the short read. If there are insertion errors, deletion transitions might be followed which may require training phase to consider r more positions where r is a parameter to the algorithm, which is fixed to $\lceil m/3 \rceil$. Thus all match and insertion states between $[q + m,\ q + m + r)$ are also added. Finally, the match state at position $q + m + r$ is also included (end state), and M_{q-1} acts as the start state. Every transition $E_{ij} \in E$ connecting state i and state j are included in E_s if $i \in V_s$ and $j \in V_s$. Each $E_{ij} \in E_s$ is associated with a transition probability α_{ij}

as described in previous subsections. For every pair of states, $i \in V_s$ and $j \in V_s$, $\alpha_{ij} = 0$ if $E_{ij} \notin E_s$.

Let $e_i(s[t])$ be the emission probability of the t^{th} basepair of s from state i. We recursively calculate forward probabilities $F_t(i)$, $\forall i \in V_s$, where $1 \leq t \leq m$ and $F_1(i)$ is the base case.

$$F_1(j) = \alpha_{start-j} e_j(s[1]) \quad s.t. \quad j \in V_s, \quad E_{start-j} \in E_s \tag{2.1}$$

$$F_t(j) = \sum_{i \in V_s; E_{ij} \in E_s} F_{t-1}(i) \alpha_{ij} e_j(s[t]) \quad j \in V_s, \quad 1 < t \leq m \tag{2.2}$$

The forward probability $F_t(j)$ denotes the probability of being at state j having observed t basepairs of s. All transitions that lead to state j contributes to the probability with (i) the forward probability of the origin state i calculated with the $t-1^{th}$ basepair in s, multiplied by (ii) the probability of the transition from i to j, which is then multiplied by (iii) the probability of emitting $s[t]$ from state j.

Backward probability $B_t(j)$ is also calculated recursively as follows, for each state $j \in V_s$ where $B_m(j)$ is the base case.

$$B_m(j) = \alpha_{j-end} \quad j \in V_s, \quad E_{j-end} \in E_s \tag{3.1}$$

$$B_t(j) = \sum_{i \in V_s; E_{ij} \in E_s} \alpha_{ji} e_i(s[t+1]) B_{t+1}(i) \quad j \in V_s, \ 1 \leq t < m \tag{3.2}$$

The backward probability $B_t(i)$ denotes the probability of being at state j and then observing the basepairs $t+1$ to m from s. All outgoing transitions from state j to a destination state i contribute to this probability with (i) the backward probability of the state i with the $t+1^{th}$ basepair of s, multiplied by (ii) the transition probability from the state j to state i, which is then multiplied by (iii) the probability of i emitting the $t+1^{th}$ basepair of s.

During calculations of both forward and backward probabilities, at each time step t, we only consider mf number of highest forward probabilities of the previous time step, $t-1$. This significantly reduces that time required to calculate forward and backward probabilities, while ensuring not to miss any important information if mf value is chosen as we suggest in the next section.

After calculation of the forward and backward probabilities (expectation step), posterior transmission and emission probabilities are calculated (maximization step), as shown in equations 4 and 5, respectively.

$$\hat{e}_i(X) = \frac{\sum_{t=1}^{m} F_t(i)B_t(i)(s[t] == X)}{\sum_{t=1}^{m} F_t(i)B_t(i)} \quad \forall X \in \Sigma, \forall i \in V_s \tag{4}$$

$$\hat{\alpha}_{ij} = \frac{\sum_{t=1}^{m-1} \alpha_{ij} e_j(s[t+1]) F_t(i) B_{t+1}(j)}{\sum_{t=1}^{m-1} \sum_{x \in V_s} \alpha_{ix} e_x(s[t+1]) F_t(i) B_{t+1}(x)} \quad \forall E_{ij} \in E_s \tag{5}$$

As we explain above, updates of emission and transition probabilities are exclusive for each sub-graph G_s. Thus, there can be overlapping states and transitions that are updated within multiple sub-graphs, independently using the prior probabilities. For such cases, updated values are averaged with respect to posterior probabilities from each subgraph. Uncovered regions keep prior transition and emission probabilities, and G is updated with the posterior emission and transmission probabilities.

3.1.3.5 Decoding

Decoding is the final process for the correction of a long read. It reads the updated complete graph, G, to decode the corrected version of a long read. For this purpose, we use the Viterbi algorithm [36] to decode the consensus sequence. We define the consensus sequence as the most likely sequence the pHMM produces after learning new parameters. The algorithm takes G for each long read and finds a path from the start state to the end state, which

yields the most likely transitions and emissions using a dynamic programming approach. For each state j, the algorithm calculates $v_t(j)$, which is the maximum marginal forward probability j obtains from its predecessors when emitting the t^{th} basepair of the corrected long read. It also keeps a back pointer, $b_t(j)$, which keeps track of the predecessor state i that yields the $v_t(j)$ value. Similar to calculations of forward and backward probabilities, we also consider mf many most likely Viterbi values from the previous time step, $t-1$, to calculate $v_t(j)$.

Let $e_j \in \Sigma$ be the basepair that is most likely to be emitted from state j and T be the length of the decoded sequence which is initially unknown. The algorithm recursively calculates v values for each position t of a decoded sequence as described in equations 6.1 and 6.3. The algorithm stops at iteration T^* such that for the last *iter* iterations, the maximum value we have observed for $v(end)$ cannot be improved - *iter* is a parameter and set to 100 by default. T is then set to t^* such that $v_{t^*}(end)$ is the maximum among all iterations $1 \leq t \leq T^*$.

1. Initialization

$$v_1(j) = a_{start-j}e_j \quad \forall j \in V \tag{6.1}$$

$$b_1(j) = start \quad \forall j \in V \tag{6.2}$$

2. Recursion

$$v_t(j) = \max_{i \in V} v_{t-1}(i)\alpha_{ij}e_j \quad \forall j \in V, 1 < t \leq T \tag{6.3}$$

$$b_t(j) = \operatorname*{argmax}_{i \in V} v_{t-1}(i)\alpha_{ij}e_j \quad \forall j \in V, 1 < t \leq T \tag{6.4}$$

3. Termination

$$v_T(end) = \max_{i \in V} v_T(i)\alpha_{i-end} \tag{6.5}$$

$$b_T(end) = \operatorname*{argmax}_{i \in V} v_T(i)\alpha_{i-end} \tag{6.6}$$

3.2 Results

3.2.1 Experimental setup

We implemented Hercules in C++ using the SeqAn library [56]. The source code is available under the BSD 3-clause license at https://github.com/BilkentCompGen/Hercules. We compare our method against Colormap, HALC, LoRDEC, LSC, and proovread. We ran all tools on a server with 4 CPUs with a total of 56 cores (Intel Xeon E7-4850 2.20GHz), and 1 TB of main memory. We assigned 60 threads to all programs including Hercules.

We used Bowtie2 with the multi-mapping option enabled to align the reads, and then we sorted resulting alignment files using SAMtools [15]. For Hercules, we set our parameters as follows: max insertion length ($l = 3$), max deletion length ($k = 10$), match transition probability ($a_M = 0.75$), insertion transition probability ($a_I = 0.20$), deletion transition probability ($a_{del} = 0.05$), deletion transition probability distribution factor ($f = 2.5$), match emission probability ($\beta = 0.97$), max coverage ($mc = 1$), max filter size ($mf = 100$). We ran all other programs with their default settings.

In our benchmarks we did not include the error correction tools that use another correction tool internally such as LoRMA, which explicitly uses LoRDEC. Similarly, HALC also uses LoRDEC to refine its corrected reads. However, it is possible to turn off this option, therefore we benchmarked HALC without LoRDEC. We also exclude Jabba in our comparisons because it only provides corrected fragments of long reads and clips the uncorrected prefix and suffixes, which results in shorter reads. All other methods consider the entire long read. Additionally, several error correction tools such as LSC and proovread do not report such reads that they could not correct. Others, however, such as Hercules, LoRDEC, and Colormap report both corrected and uncorrected reads, preserving the original number of reads. In order to make the results of all correction tools comparable, we re-insert the original versions of the discarded reads to LSC and proovread output. We ran Colormap with its additional option (OEA) that incorporates more refinements on a corrected read. We observed that there was no difference between running OEA option or not in terms of the accuracy of its resulting reads, although Colormap with OEA option required substantially higher run time.

Table 3.1: Summary of error correction

Tool	Number of aligned reads											
	CH17-157L1 BAC clone				CH17-227A2 BAC clone				Human brain transcriptome			
	Mapped	80-90%	90-95%	>95%	Mapped	80-90%	90-95%	>95%	Mapped	80-90%	90-95%	>95%
Uncorrected	33,842	17,582	1,974	461	45,625	25,356	7,385	219	150,000	107,493	17,884	921
Colormap	36,391	13,586	7,904	6,235	46,209	18,398	12,364	4,715	150,663	49,803	35,412	50,518
HALC	35,220	17,867	2,974	2,249	46,680	23,969	9,662	2,356	150,970	93,500	29,822	6,682
LoRDEC	36,812	10,320	7,397	12,167	47,304	8,875	7,010	**25,844**	149,923	40,914	38,927	56,818
LSC	43,431	13,534	7,511	12,277	55,853	13,619	10,205	23,509	150,771	33,910	36,706	**66,726**
Hercules	**44,229**	13,609	7,569	**12,583**	**56,140**	13,433	9,809	24,269	**151,903**	34,360	38,352	65,880
proovread	38,962	14,918	6,714	7,956	47,344	17,460	9,233	11,140	150,235	82,344	21,891	25,944

We report error-corrected PacBio reads generated from two BAC clones (CH17-157L1 and CH17-227A2) using Illumina reads from the same resource, and long reads from human brain transcriptome using Illumina reads from the Human Body Map 2.0 project. We report the accuracy as the alignment identity as calculated by the BLASR [57] aligner. *Mapped* refers to the number of any reads aligned to the Sanger-assembled reference for these clones with any identity, where the other columns show the number of alignments within respective identity brackets.

3.2.2 Data sets

To compare Hercules's performance with other tools we used two DNA-seq and 1 RNA-seq data sets that were generated from two bacterial artificial chromosome (BAC) clones, previously sequenced to resolve complex regions in human chromosome 17 [7]. These clones are sequenced with two different HTS technologies: CH17-157L1 (231 Kbp) data set includes 93,785 PacBio long reads (avg 2,576bp; 1046X coverage) and 372,272 Illumina paired-end reads (76 bp each, 245X coverage); and CH17-227A2 (200 Kbp) data set includes 100,729 PacBio long reads (avg 2,663bp; 338X coverage) and 342,528 Illumina paired-end reads (76 bp each, 260X coverage). Additionally, sequences and finished assemblies generated with the Sanger platform are also available for the same BAC clones, which we use as the gold standard to test the correction accuracy. The RNA-seq data is generated from a human brain sample, which is also used in the original publication of the LSC algorithm [24]. This transcriptome data set includes 155,142 long reads (avg 879bp). The corresponding 64,313,204 short reads are obtained from the Human Body Map 2.0 project (SRA GSE30611). We calculated error correction accuracy using the human reference genome (GRCh38) for the human brain transcriptome data as the gold standard.

Table 3.2: Correction accuracy given high breadth of coverage (> 90%) in CH17-227A2.

Tool	No. of Aligned Reads			
	Mapped	80-90%	90-95%	>95%
Uncorrected	4,429	2,135	2,093	26
Colormap	4,432	668	2,345	1,271
HALC	4,433	1,789	2,436	58
LoRDEC	4,425	124	224	4,006
LSC	4,473	45	149	4,264
Hercules	**4,476**	43	142	**4,273**
proovread	4,434	1,722	1,665	880

The input included 9,449 PacBio reads which at least 90% of their length were covered by short reads. We report the accuracy as the alignment identity as calculated by the BLASR [57] aligner. *Mapped* column refers to the number of any reads aligned to the Sanger-assembled reference for this clone, where the other columns show the number of alignments within respective identity brackets.

3.2.3 Correction accuracy

After correction, we align the corrected reads to the corresponding ground truth using BLASR [57] with the *noSplitSubreads* option, which forces entire read to align, if possible. Then, we simply calculate the accuracy of a read as reported by BLASR as alignment identity. We report the number of reads that align to the gold standard, and the alignment accuracy in four different accuracy brackets. We observe that the number of confident alignments were the highest in both BAC clones and the human brain transcriptome for the reads corrected by Hercules (Table 3.1). Furthermore, for CH17-157L1, Hercules returned the largest number of corrected reads with the highest (>95%) accuracy. In contrast, the LoRDEC had slightly more reads with the highest (>95%) accuracy for CH17-227A2 compared to Hercules. We further investigated the reasons for lower Hercules performance for this clone, and we found that only ∼10% of the reads had > 90% short reads breadth coverage. For such reads with better coverage, Hercules's performance surpassed that of LoRDEC and all other correction tools (Table 3.2). See Section 3.3 for a discussion on the challenges of comparing accuracy of error correction tools.

Table 3.3: Requirements of computational resources for all methods we compared.

	CH17-1571		CH17-221A2		Human Brain	
Tool	Run time	Memory (GB)	Run time	Memory (GB)	Run time	Memory (GB)
Hercules	1h 07m 49s	3.20	2h 10m 09s	3.16	2h 13m 50s	8.83
Colormap	1h 58m 57s	9.89	3h 10m 51s	9.23	5h 30m 49s*	20.68*
HALC	16m 25s	26.29	**25m 09s**	23.09	44m 43s	180.32
LoRDEC	**12m 43s**	**1.20**	26m 19s	**1.33**	**34m 28s**	**3.770**
LSC	1h 04m 55s	11.08	1h 40m 51s	9.00	26h 29m 34s	73.59
proovread	1h 34m 24s	7.91	2h 39m 57s	4.03	4h 26m 04s	11.45

We ran all tools on the same server using 60 threads. We report wall clock run times, and peak memory usage (GB) for each tool. * We ran Colormap without its OEA refinement option for the human brain transcriptome data as it is substantially slower than default options.

3.2.4 Run time and memory requirements

Finally, we benchmarked computational requirements for each tool (Table 3.3). We observed that LoRDEC was the fastest algorithm, and the run times of alignment-based tools were similar. Running time of Hercules is still on the same scale even though the learning procedure is more time consuming because of per-read calls to Forward Backward and Viterbi algorithms. Since error correction is an offline and one-time task we argue that given the gain in accuracy this run time is acceptable.

We also investigated the memory requirements of the tools we benchmarked (Table 3.3). We show that Hercules uses only a modest amount of memory, second to only LoRDEC. In our experiment data sets, the maximum memory Hercules required was 8 GB, which makes it usable in commodity desktop computers. We note that the memory requirements of Hercules scale linearly with the lengths of the long reads to be corrected simultaneously in multiple threads, and the LoRDEC memory usage will depend on the size of the de Bruijn graph. Note that LoRDEC is alignment-free, and it builds a de Bruijn graph for the entire short read data set. Thus, its memory usage will be determined by the size of the input DNA (whole genome, whole transcriptome, or clones).

3.3 Discussion

Long reads such as PacBio are attractive alternatives to short Illumina reads due to the ability to span across common repeats, which present analytical and computational challenges to accurately characterize and assemble genomes. However, their error rate is also substantially higher than that of short reads, inferring additional challenges in basepair accuracy. Therefore, it is of utmost importance to correct short indel and substitution errors either using very high coverage long read data, which substantially increases sequencing cost, or orthogonal and less expensive short read sequencing data. Here we introduced a new algorithm, Hercules, as the first hidden Markov model based method to correct erroneous long reads using short but accurate Illumina data.

Profile HMMs were previously proven to be powerful for the problems in sequence analysis such as multiple sequence alignment and protein classification [58], however they were not exploited for error correction before. In this thesis, we modified the standard pHMM to leverage their probabilistic basepair consensus representation to correct long reads given set of aligned short reads. Our proposed pHMM-like structure offers a flexible approach since its initial parameters can be redefined based on the error profile of any other error-prone sequencing technology, such as Oxford Nanopore. Moreover, although Hercules uses the short to long read information, it is largely independent of the underlying aligner, as it only uses the map start location, and uses the entire short read to train the pHMM.

Benchmarking aligner-based error correction tools is a complicated job. Like any other long-short read alignment based correction tool, Hercules is also dependent on the performance of an aligner. It can only attempt to correct a region if there is at least one short read covering the region. Therefore, choice of an alignment tool is also a crucial for producing more reliable reads. First problem is that it is not always possible to have a fair comparison among those error correction tools as we cannot plug the same aligner into their pipeline. For instance, proovread offers a modified version of BWA-MEM aligner within its bundle where it is not possible to use standard BWA-MEM. This may also cause proovread to be outdated when BWA-MEM gets updates. LSC offers use of Bowtie2 and Hisat [59] aligner tools within its current version (2.0). Second problem is that the set of output reads are not always the same for each method. LSC and proovread only returns the processed long

reads and do not report the reads that could not be corrected. Some tools, however, such as LoRDEC or Hercules, report every long read, including the reads that cannot be corrected. Therefore, any fair comparison between any correction tool should make sure that number of the reads that are compared are the same. Furthermore, Jabba only outputs the corrected regions of a long read, not even the whole read. HALC uses BLASR internally and BLASR does not support an input that has a size higher than 4GB. Thus, size of the reads was also another criteria to be able to include HALC in our benchmark. It is even more complicated to compare alignment based tools with de Bruijn graph based tools. The latter are not dependent on performance of an any external aligners whereas the former only correct the short-read-covered regions provided by an aligner. Therefore, uncovered regions, and thus, uncorrected regions of a long read with an aligner based method, maybe corrected by de Bruijn graph based methods. While comparing these two approaches, a fair comparison would be among the corrected reads that also have high short read breadth of coverage (i.e. with at least 90% alignment coverage) to take away the coverage bias.

Hercules is slower and more memory demanding compared to LoRDEC. However, as we have explained in the Results section, the memory requirement of LoRDEC will scale with the length of the input genome to be sequenced. In terms of run time, the main bottleneck for Hercules lies in the pHMM training for each long read. There are several algorithms proposed to improve running time and memory requirements of the standard Viterbi algorithm and Forward/Backward likelihood calculations that may be used to improve running time [60, 61, 62]. Additionally Hercules may further be accelerated through SIMD vectorization of pHMM training and decoding [63, 64, 65] that we leave as future work.

Even though running time results may seem advantageous for de Bruijn graph based tools, they lack the confidence the aligner provides regarding the match of the short read. Hence, it is possible for those methods to corrupt regions rather than correcting it As an indication of this trend, we see that LoRDEC-corrected-reads fail map to the ground truth as much as Hercules or LSC, even though significant percentage of their corrected reads have accuracy $\geq 95\%$. Total number of long reads that can be aligned to a ground truth is higher for LSC and Hercules compared to LoRDEC, which is an indication that the aligner confidence provides extra information compared to the graph based approaches.

We implement original versions of forward, backward, and Viterbi calculations but it is

still possible to adapt possible more efficient calculations as a future work. This would require adaptability of a formula that is applicable with our proposed pHMM graph. Moreover, the error profile of each sequencing machine is different than other machines. Thus, even though we suggest a set of default parameters, those parameters may still not be optimized for different machines. All of the correction tools in the literature are not aware of type of sequence machine and the error profile, instead they work deterministically within the boundaries of alignment or de Bruijn graph. However, Hercules provides more flexible structure where it is possible to approach a sequencing data given the error profile of the sequencing machine. It is, in principle, easy to extend Hercules to correct errors in data generated with future technologies, or future iterations of current PacBio and ONT platforms, by simply readjusting the profile HMM priors according to the error model.

As with its current version, Hercules introduces a novel approach for correcting long reads by using a machine learning technique. Its structure allows to implement new mathematical models to calculate the probabilities, likelihoods, and the decoding formula whenever possible. Therefore, a new focus as future work for an attempt to correct long reads may mainly be dependent on proposing new mathematical models based on our proposed structure in order to improve accuracy, complexity, and memory requirements of our algorithm.

Bibliography

[1] L. G. Biesecker, J. C. Mullikin, F. M. Facio, C. Turner, P. F. Cherukuri, R. W. Blakesley, G. G. Bouffard, P. S. Chines, P. Cruz, N. F. Hansen, J. K. Teer, B. Maskeri, A. C. Young, N. I. S. C. C. S. P. , T. A. Manolio, A. F. Wilson, T. Finkel, P. Hwang, A. Arai, A. T. Remaley, V. Sachdev, R. Shamburek, R. O. Cannon, and E. D. Green, "The ClinSeq project: piloting large-scale genome sequencing for research in genomic medicine" *Genome Res*, vol. 19, pp. 1665–1674, Sep 2009.

[2] T. J. Treangen and S. L. Salzberg, "Repetitive DNA and next-generation sequencing: computational challenges and solutions" *Nat Rev Genet*, vol. 13, pp. 36–46, Jan 2012.

[3] C. Firtina and C. Alkan, "On genomic repeats and reproducibility" *Bioinformatics*, vol. 32, pp. 2243–2247, Aug 2016.

[4] C. Alkan, S. Sajjadian, and E. E. Eichler, "Limitations of next-generation genome sequence assembly" *Nat Methods*, vol. 8, pp. 61–65, Jan 2011.

[5] M. J. P. Chaisson, R. K. Wilson, and E. E. Eichler, "Genetic variation and the de novo assembly of human genomes" *Nat Rev Genet*, Oct 2015.

[6] K. M. Steinberg, V. A. Schneider, C. Alkan, M. J. Montague, W. C. Warren, D. M. Church, and R. K. Wilson, "Building and improving reference genome assemblies" *Proceedings of the IEEE*, vol. 105, pp. 422–435, Mar 2017.

[7] J. Huddleston, S. Ranade, M. Malig, F. Antonacci, M. Chaisson, L. Hon, P. H. Sudmant, T. A. Graves, C. Alkan, M. Y. Dennis, R. K. Wilson, S. W. Turner, J. Korlach, and E. E. Eichler, "Reconstructing complex regions of genomes using long-read sequencing technology" *Genome Res*, vol. 24, pp. 688–696, Apr 2014.

[8] M. Jain, S. Koren, J. Quick, A. C. Rand, T. A. Sasani, J. R. Tyson, A. D. Beggs, A. T. Dilthey, I. T. Fiddes, S. Malla, H. Marriott, K. H. Miga, T. Nieto, J. O'Grady, H. E. Olsen, B. S. Pedersen, A. Rhie, H. Richardson, A. Quinlan, T. P. Snutch, L. Tee, B. Paten, A. M. Phillippy, J. T. Simpson, N. J. Loman, and M. Loose, "Nanopore sequencing and assembly of a human genome with ultra-long reads" *bioRxiv*, 2017.

[9] P. Kavak, B. Yuksel, S. Aksu, M. O. Kulekci, T. Gungor, F. Hach, S. C. Sahinalp, T. H. G. Project, C. Alkan, and M. S. Sagiroglu, "Robustness of massively parallel sequencing platforms" *PLoS One*, vol. 10, no. 9, p. e0138259, 2015.

[10] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM" *arXiv preprint arXiv:1303.3997*, 2013.

[11] The 1000 Genomes Project Consortium, "A global reference for human genetic variation" *Nature*, vol. 526, pp. 68–74, Sep 2015.

[12] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome" *Genome Biol*, vol. 10, p. R25, Mar 2009.

[13] C. Alkan, J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, C. Baker, M. Malig, O. Mutlu, S. C. Sahinalp, R. A. Gibbs, and E. E. Eichler, "Personalized copy number and segmental duplication maps using next-generation sequencing" *Nat Genet*, vol. 41, pp. 1061–1067, Oct 2009.

[14] D. Weese, M. Holtgrewe, and K. Reinert, "RazerS 3: faster, fully sensitive read mapping" *Bioinformatics*, vol. 28, pp. 2592–2599, Oct 2012.

[15] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and . G. P. D. P. Subgroup, "The sequence alignment/map format and SAMtools" *Bioinformatics*, vol. 25, pp. 2078–2079, Aug 2009.

[16] M. A. DePristo, E. Banks, R. Poplin, K. V. Garimella, J. R. Maguire, C. Hartl, A. A. Philippakis, G. del Angel, M. A. Rivas, M. Hanna, A. McKenna, T. J. Fennell, A. M. Kernytsky, A. Y. Sivachenko, K. Cibulskis, S. B. Gabriel, D. Altshuler, and M. J. Daly, "A framework for variation discovery and genotyping using next-generation DNA sequencing data" *Nat Genet*, vol. 43, pp. 491–498, May 2011.

[17] A. Rimmer, H. Phan, I. Mathieson, Z. Iqbal, S. R. F. Twigg, W. G. S. C. , A. O. M. Wilkie, G. McVean, and G. Lunter, "Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications" *Nat Genet*, vol. 46, pp. 912–918, Aug 2014.

[18] E. Garrison and G. Marth, "Haplotype-based variant detection from short-read sequencing" *arXiv preprint arXiv:1207.3907*, Jul 2012.

[19] R. E. Mills, K. Walter, et al., "Mapping copy number variation by population-scale genome sequencing" *Nature*, vol. 470, pp. 59–65, Feb 2011.

[20] J. M. Zook, B. Chapman, J. Wang, D. Mittelman, O. Hofmann, W. Hide, and M. Salit, "Integrating human sequence data sets provides a resource of benchmark snp and indel genotype calls" *Nat Biotechnol*, vol. 32, pp. 246–251, Mar 2014.

[21] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing" *Briefings in Bioinformatics*, vol. 11, no. 5, pp. 473–483, 2010.

[22] K. Berlin, S. Koren, C.-S. Chin, J. P. Drake, J. M. Landolin, and A. M. Phillippy, "Assembling large genomes with single-molecule sequencing and locality-sensitive hashing" *Nat Biotechnol*, vol. 33, pp. 623–630, Jun 2015.

[23] S. Koren, M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis, and A. M. Phillippy, "Hybrid error correction and de novo assembly of single-molecule sequencing reads" *Nat Biotechnol*, vol. 30, pp. 693–700, Jul 2012.

[24] K. F. Au, J. G. Underwood, L. Lee, and W. H. Wong, "Improving PacBio long read accuracy by short read alignment" *PloS One*, vol. 7, p. e46679, 2012.

[25] T. Hackl, R. Hedrich, J. Schultz, and F. Förster, "proovread: large-scale high-accuracy pacbio correction through iterative short read consensus" *Bioinformatics*, vol. 30, pp. 3004–3011, Nov. 2014.

[26] E. Haghshenas, F. Hach, S. C. Sahinalp, and C. Chauve, "Colormap: Correcting long reads by mapping short reads" *Bioinformatics*, vol. 32, no. 17, pp. i545–i551, 2016.

[27] L. Salmela and E. Rivals, "LoRDEC: accurate and efficient long read error correction" *Bioinformatics*, vol. 30, pp. 3506–3514, Dec 2014.

[28] G. Miclotte, M. Heydari, P. Demeester, S. Rombauts, Y. Van de Peer, P. Audenaert, and J. Fostier, "Jabba: hybrid error correction for long sequencing reads" *Algorithms for Molecular Biology*, vol. 11, p. 10, 2016.

[29] E. Bao and L. Lan, "Halc: High throughput algorithm for long read error correction" *BMC Bioinformatics*, vol. 18, no. 1, p. 204, 2017.

[30] L. Salmela, R. Walve, E. Rivals, and E. Ukkonen, "Accurate self-correction of errors in long reads using de bruijn graphs" *Bioinformatics*, vol. 33, no. 6, pp. 799–806, 2016.

[31] M. Chaisson, P. Pevzner, and H. Tang, "Fragment assembly with short reads" *Bioinformatics*, vol. 20, pp. 2067–2074, Sep 2004.

[32] D. R. Zerbino and E. Birney, "Velvet: algorithms for de novo short read assembly using de Bruijn graphs" *Genome Res*, vol. 18, pp. 821–829, May 2008.

[33] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol, "ABySS: a parallel assembler for short read sequence data" *Genome Res*, vol. 19, pp. 1117–1123, Jun 2009.

[34] C. Firtina, Z. Bar-Joseph, C. Alkan, and A. E. Cicek, "Hercules: a profile HMM-based hybrid error correction algorithm for long reads" *bioRxiv*, 2017.

[35] L. E. Baum, "An inequality and associated maximization thechnique in statistical estimation for probabilistic functions of markov process" *Inequalities*, vol. 3, pp. 1–8, 1972.

[36] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm" *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[37] S. R. Eddy, "Profile hidden Markov models" *Bioinformatics*, vol. 14, pp. 755–763, 1998.

[38] G. A. Van der Auwera, M. O. Carneiro, C. Hartl, R. Poplin, G. Del Angel, A. Levy-Moonshine, T. Jordan, K. Shakir, D. Roazen, J. Thibault, E. Banks, K. V. Garimella, D. Altshuler, S. Gabriel, and M. A. DePristo, "From fastq data to high confidence variant calls: the genome analysis toolkit best practices pipeline" *Curr Protoc Bioinformatics*, vol. 11, pp. 11.10.1–11.10.33, Oct 2013.

[39] T. Rausch, T. Zichner, A. Schlattl, A. M. Stütz, V. Benes, and J. O. Korbel, "DELLY: structural variant discovery by integrated paired-end and split-read analysis" *Bioinformatics*, vol. 28, pp. i333–i339, Sep 2012.

[40] R. M. Layer, C. Chiang, A. R. Quinlan, and I. M. Hall, "LUMPY: a probabilistic framework for structural variant discovery" *Genome Biol*, vol. 15, no. 6, p. R84, 2014.

[41] R. E. Handsaker, V. Van Doren, J. R. Berman, G. Genovese, S. Kashin, L. M. Boettger, and S. A. McCarroll, "Large multiallelic copy number variations in humans" *Nat Genet*, vol. 47, pp. 296–303, Mar 2015.

[42] F. Hormozdiari, C. Alkan, E. E. Eichler, and S. C. Sahinalp, "Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes" *Genome Res*, vol. 19, pp. 1270–1278, Jul 2009.

[43] F. Hormozdiari, I. Hajirasouliha, A. McPherson, E. E. Eichler, and S. C. Sahinalp, "Simultaneous structural variation discovery among multiple paired-end sequenced genomes" *Genome Res*, vol. 21, pp. 2203–2212, Dec 2011.

[44] W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, and D. Haussler, "The human genome browser at UCSC.bed" *Genome Res*, vol. 12, pp. 996–1006, Jun 2002.

[45] A. R. Quinlan and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features" *Bioinformatics*, vol. 26, pp. 841–842, Mar 2010.

[46] E. R. Mardis, "The impact of next-generation sequencing technology on genetics" *Trends Genet*, vol. 24, pp. 133–141, Mar 2008.

[47] M. L. Metzker, "Sequencing technologies - the next generation" *Nat Rev Genet*, vol. 11, pp. 31–46, Jan 2010.

[48] I. E. Bosdet, T. R. Docking, Y. S. Butterfield, A. J. Mungall, T. Zeng, R. J. Coope, E. Yorida, K. Chow, M. Bala, S. S. Young, M. Hirst, I. Birol, R. A. Moore, S. J. Jones, M. A. Marra, R. Holt, and A. Karsan, "A clinically validated diagnostic second-generation sequencing assay for detection of hereditary BRCA1 and BRCA2 mutations" *J Mol Diagn*, vol. 15, pp. 796–809, Nov 2013.

[49] R. Rodriguez and K. M. Miller, "Unravelling the genomic targets of small molecules using high-throughput sequencing" *Nat Rev Genet*, vol. 15, pp. 783–796, Dec 2014.

[50] C. Alkan, B. P. Coe, and E. E. Eichler, "Genome structural variation discovery and genotyping" *Nat Rev Genet*, vol. 12, pp. 363–376, May 2011.

[51] R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song, "Genotype and SNP calling from next-generation sequencing data" *Nat Rev Genet*, vol. 12, pp. 443–451, Jun 2011.

[52] A. Cornish and C. Guda, "A comparison of variant calling pipelines using Genome in a Bottle as a reference" *Biomed Res Int*, vol. 2015, p. 456479, 2015.

[53] J. Eid, A. Fehr, et al., "Real-time DNA sequencing from single polymerase molecules" *Science*, vol. 323, pp. 133–138, Jan 2009.

[54] D. R. Bentley, S. Balasubramanian, et al., "Accurate whole human genome sequencing using reversible terminator chemistry" *Nature*, vol. 456, pp. 53–59, Nov 2008.

[55] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with Bowtie 2" *Nature Methods*, vol. 9, pp. 357–359, Mar. 2012.

[56] A. Döring, D. Weese, T. Rausch, and K. Reinert, "SeqAn an efficient, generic C++ library for sequence analysis" *BMC Bioinformatics*, vol. 9, p. 11, Jan. 2008.

[57] M. J. Chaisson and G. Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory" *BMC Bioinformatics*, vol. 13, p. 238, 2012.

[58] B.-J. Yoon, "Hidden markov models and their applications in biological sequence analysis" *Current Genomics*, vol. 10, no. 6, pp. 402–415, 2009.

[59] D. Kim, B. Langmead, and S. L. Salzberg, "Hisat: a fast spliced aligner with low memory requirements" *Nature Methods*, vol. 12, no. 4, pp. 357–360, 2015.

[60] M. S. Ryan and G. R. Nudd, "The viterbi algorithm" *Department of Computer Science research report,* University of Warwick, CS-RR-238, 1993.

[61] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications" in *Proc. IEEE GLOBECOM '89*, pp. 1680–1686, IEEE, 1989.

[62] H.-L. Lou, "Implementing the viterbi algorithm" *IEEE Signal processing magazine*, vol. 12, no. 5, pp. 42–52, 1995.

[63] S. R. Eddy, "Accelerated profile HMM searches" *PLoS Computational Biology*, vol. 7, no. 10, p. e1002195, 2011.

[64] M. Ferreira, N. Roma, and L. M. Russo, "Cache-oblivious parallel SIMD Viterbi decoding for sequence search in HMMER" *BMC Bioinformatics*, vol. 15, no. 1, p. 165, 2014.

[65] J. Ou, J. Cai, and Q. Lin, "Using SIMD technology to speed up likelihood computation in HMM-based speech recognition systems" in *Proc. Language and Image Processing 2008 Int. Conf. Audio*, pp. 123–127, July 2008.

Appendix A

Data

Supplementary Table S1: List of data sets used in this study.

Sample	Type	No. reads (mapped)	Read length (bp)	Coverage*
HG00096	WGS	146,347,388	100	4.88X
HG02107	WGS	1,319,393,745	101	44.42X
HG00250	WES	219,687,521	76	333.93X
HG00251	WES	115,030,162	90	207.05X
HG00330	WES	108,059,848	90	194.51X
HG00731	WES	327,901,475	76	498.41X
HG00732	WES	141,038,355	90	253.87X
HG01075	WES	431,625,989	76	656.07X
HG01083	WES	59,336,723	101	119.86X
HG01133	WES	61,187,487	101	123.60X
HG01136	WES	185,754,635	76	282.35X
HG01140	WES	97,205,053	90	174.97X
HG01167	WES	156,335,569	76	237.63X
HG01176	WES	63,122,492	101	127.51X

We downloaded 2 whole genome (WGS) and 12 whole exome (WES) data sets from the 1000 Genomes Project [11]. We used the WES data sets to test the SNV calls generated by GATK only. *Coverage is calculated assuming the genome size as 3 Gbp for WGS data, and the exome size as 50 Mbp for WES data.

Supplementary Table S2: Tools and their version numbers we used in this study.

Tool	**Version**	**Purpose**
Bowtie* [12]	2.2.5	Read Mapping
RazerS3* [14]	3.4	Read Mapping
mrFAST* [13]	2.6.1.0	Read Mapping
BWA-MEM [10]	0.7.12-r1039	Read Mapping
Picard[1]	1.105	BAM processing
SAMtools [15]	1.2	SNVs and indels
GATK [16]	3.4-0-g7e26428	SNVs and indels
Freebayes [18]	v0.9.21-19-gc003c1e	SNVs and indels
Platypus [17]	1.2.1	SNVs and indels
DELLY [39]	0.6.7	Structural Variation
LUMPY [40]	0.2.11	Structural Variation
Genome STRiP [41]	2.00.1602	Structural Variation

*In small-scale test only. ‡VariationHunter and CommonLAW explicitly remap reads using mrFAST, which we show to be a deterministic mapper.

Supplementary Table S3: Summary of alignments of 1 million reads sampled from HG00096 in original vs. shuffled order.

Tool	**Original**			**Shuffled**			**Run time**
	Total	*Private*	*Private (%)*	*Total*	*Private*	*Private (%)*	
mrFAST	904,952	0	-	904,952	0	-	106m 41s
Bowtie	929,393	0	-	929,393	0	-	8m 56sec
RazerS3	650,086	5,949	0.9	650,088	5,951	0.9	20m 38s
BWA-MEM	897,548	18,862	2.1	897,534	18,848	2.1	11m 21s

Command lines:
mrfast –search human_g1k_v37.fasta –seq1 file_1.fq –seq2 file_2.fq -o mrfast.sam –min 0 –max 1000 –pe
bowtie2 -x human_g1k_v37 -1 file_1.fq -2 file_2.fq -S bowtie.sam
razers3 -i 96 -tc 12 -m 1 -o razers3.sam human_g1k_v37.fasta file_1.fq file_2.fq
bwa mem -t 16 human_g1k_v37.fasta file_1.fq file_2.fq > bwa.sam
We further filtered mappings and removed both hard and soft-clipped reads, and mappings with edit distance >4 in this small-scale experiment only to easily characterize private mappings. We used the full set of alignments in the variant caller tests.

Supplementary Table S4: Summary of scatter/gather based parallel read mapping of 1 million reads using BWA-MEM.

Chunk Size	*Total*	*Private*	*Private (%)*	*Private (MAPQ=0)*	*Private (MAPQ≥30)*
50,000	997,192	20920	2.1	13,907	4,346
100,000	997,202	20930	2.1	13,916	4,305

We mapped 1 million reads in a scatter/gather fashion using two different chunk sizes (50,000 and 100,000 read pairs per chunk). We first divided 1 million reads (500,000 read pairs) to different chunks, mapped them separately using BWA-MEM, and then merged the resulting BAM files. We also removed the secondary and supplementary mappings from the alignments (i.e. -F 256+2048) for non-redundant comparison. Map locations that are specific to one chunk size are listed as *Private*. We also list the number of private mappings with 0 and ≥ 30 mapping quality.

Supplementary Table S5: Summary of multithreaded read mapping of ∼49 million reads using BWA-MEM.

Threads			Compared to					
		Total	1	2	4	8	16	32
Private to	1	48,166,707	-	25,532	28,397	29,081	29,151	29,682
	2	48,166,737	25,562	-	14,433	15,978	16,140	16,833
	4	48,166,777	28,467	14,473	-	4,836	5,265	5,806
	8	48,166,777	29,151	16,018	4,836	-	2,244	3,259
	16	48,166,777	29,221	16,180	5,265	2,244	-	2,509
	32	48,166,777	29,752	16,873	5,806	3,259	2,509	-

We mapped ∼49 million reads (SRR062634) using different number of threads using BWA-MEM. We also removed the secondary and supplementary mappings from the alignments (i.e. -F 256+2048) for non-redundant comparison.

Supplementary Table S6: Summary of read mapping of the HG00096 genome in both original and shuffled order using BWA-MEM.

Order	*Total*	*Private*	*Private (%)*	*Private (MAPQ=0)*	*Private (MAPQ≥30)*
Original	146,347,388	3,725,916	2.54	3,614,369	27,927
Shuffled	146,347,372	3,725,916	2.54	3,614,438	27,873

Map locations that are specific to one order of reads (including the alignment flags) are listed as *Private*. We also list the number of private mappings with 0 and ≥ 30 mapping quality.

Supplementary Table S7: Summary of HaplotypeCaller call sets.

HaplotypeCaller	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	2,279,678	1,160,574	83,944	2,098,633	53,726
Shuffled total	2,294,808	1,172,610	91,573	2,107,371	53,954
Original only	10,898	8,577	1,917	9,205	253
Shuffled only	26,028	20,613	9,546	18,381	572
Original only (in dbSNP 138)	10,802	8,495	1,892	4,224	132
Shuffled only (in dbSNP 138)	24,536	19,580	9,033	8,234	344
Original only (homozygous)	10,648	8,401	1,842	9,075	252
Shuffled only (homozygous)	25,404	20,236	9,220	18,042	561
Original only (SNV)	10,775	8,474	1,877	9,123	252
Shuffled only (SNV)	25,914	20,516	9,519	18,328	572
HaplotypeCaller	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	4,654,338	2,423,277	149,867	4,442,738	158,971
Shuffled total	4,625,648	2,399,863	148,480	4,416,242	158,379
Original only	54,051	43,084	6,020	48,034	1,249
Shuffled only	25,361	19,670	4,633	21,538	657
Original only (in dbSNP 138)	51,002	40,733	5,490	20,943	628
Shuffled only (in dbSNP 138)	23,224	18,002	4,277	9,208	344
Original only (homozygous)	8,228	6,287	1,546	7,122	275
Shuffled only (homozygous)	3,318	2,725	468	2,831	47
Original only (SNV)	53,166	42,338	5,734	47,515	1,241
Shuffled only (SNV)	24,535	18,964	4,385	21,049	643

We list the total number of (SNV)s and indels characterized by HaplotypeCaller using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S8: Summary of UnifiedGenotyper call sets.

UnifiedGenotyper	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	2,470,690	1,335,457	201,057	2,163,192	56,994
Shuffled total	2,174,342	1,096,859	84,130	2,000,691	53,044
Original only	315,174	247,777	117,368	189,169	6,103
Shuffled only	18,826	9,179	441	18,173	567
Original only (in dbSNP 138)	265,445	210,294	105,250	83,544	3,503
Shuffled only (in dbSNP 138)	18,774	9,125	420	8,164	288
Original only (homozygous)	111,373	94,551	25,313	85,263	1,945
Shuffled only (homozygous)	18,664	9,083	422	18,042	559
Original only (SNV)	314,767	247,454	117,161	189,091	6,099
Shuffled only (SNV)	18,782	9,138	426	18,160	565
UnifiedGenotyper	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	4,489,253	2,283,267	132,449	4,294,650	157,013
Shuffled total	4,723,109	2,480,235	173,500	4,494,733	161,795
Original only	38,635	30,414	9,669	29,858	1,237
Shuffled only	272,491	227,382	50,720	229,941	6,019
Original only (in dbSNP 138)	38,091	29,935	9,523	13,742	690
Shuffled only (in dbSNP 138)	262,671	219,986	49,207	102,557	3,327
Original only (homozygous)	35,484	28,211	8,682	27,160	1,080
Shuffled only (homozygous)	14,214	11,311	2,257	11,989	317
Original only (SNV)	37,995	29,865	9,513	29,412	1,231
Shuffled only (SNV)	271,821	226,836	50,569	229,411	6,017

We list the total number of (SNV)s and indels characterized by UnifiedGenotyper using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S9: Summary of FreeBayes call sets.

Freebayes	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	2,400,545	1,336,125	172,214	2,125,570	54,445
Shuffled total	2,400,595	1,336,148	172,214	2,125,675	54,447
Original only	992	789	400	341	11
Shuffled only	1,042	812	400	446	13
Original only (in dbSNP 138)	549	394	270	113	8
Shuffled only (in dbSNP 138)	566	399	250	138	10
Original only (homozygous)	265	224	93	125	0
Shuffled only (homozygous)	263	215	81	163	6
Original only (SNV)	848	655	361	281	9
Shuffled only (SNV)	899	687	352	350	11
Freebayes	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	5,174,644	2,906,071	380,096	4,735,347	170,214
Shuffled total	5,189,285	2,917,114	382,898	4,748,004	170,638
Original only	4,715	3,878	1,394	2,711	61
Shuffled only	19,356	14,921	4,196	15,368	485
Original only (in dbSNP 138)	1,570	1,154	707	379	20
Shuffled only (in dbSNP 138)	9,149	6,610	2,522	3,371	166
Original only (homozygous)	2,128	1,802	404	1,697	33
Shuffled only (homozygous)	902	716	288	533	6
Original only (SNV)	3,415	2,712	1,231	1,618	45
Shuffled only (SNV)	12,633	9,292	3,632	9,190	316

We list the total number of (SNV)s and indels characterized by Freebayes using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S10: Summary of SAMtools call sets.

SAMtools	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	2,277,691	1,269,164	176,433	2,055,105	69,037
Shuffled total	2,277,674	1,269,121	176,401	2,055,168	69,042
Original only	2,683	2,092	1,361	878	56
Shuffled only	2,666	2,049	1,329	941	61
Original only (in dbSNP 138)	1,822	1,364	1,113	378	26
Shuffled only (in dbSNP 138)	1,845	1,367	1,088	384	35
Original only (homozygous)	1,012	824	453	462	32
Shuffled only (homozygous)	989	765	450	480	24
Original only (SNV)	2,492	1,919	1,286	830	55
Shuffled only (SNV)	2,488	1,886	1,255	865	60
SAMtools	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	5,355,604	3,055,463	417,874	4,862,553	175,803
Shuffled total	5,355,053	3,055,015	417,735	4,862,421	175,790
Original only	9,838	7,688	4,494	4,274	198
Shuffled only	9,287	7,240	4,355	4,142	185
Original only (in dbSNP 138)	5,690	4,356	3,219	1,193	89
Shuffled only (in dbSNP 138)	5,553	4,178	3,159	1,220	82
Original only (homozygous)	2,179	1,813	938	1,056	34
Shuffled only (homozygous)	2,102	1,716	908	1,050	44
Original only (SNV)	8,597	6,572	4,148	3,458	182
Shuffled only (SNV)	8,074	6,161	3,993	3,233	148

We list the total number of (SNV)s and indels characterized by SAMtools using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S11: Summary of Platypus call sets.

Platypus	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	2,022,412	1,073,056	115,590	1,843,781	47,144
Shuffled total	2,022,294	1,072,940	115,538	1,843,952	47,166
Original only	2,342	1,778	1,097	1,119	27
Shuffled only	2,224	1,662	1,045	1,290	49
Original only (in dbSNP 138)	1,776	1,351	864	507	18
Shuffled only (in dbSNP 138)	1,723	1,295	811	528	26
Original only (homozygous)	1,107	935	335	661	9
Shuffled only (homozygous)	1,071	875	309	757	27
Original only (SNV)	2,079	1,571	973	977	27
Shuffled only (SNV)	1,977	1,466	937	1,134	44
Platypus	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	4,642,336	2,470,826	235,690	4,355,365	156,396
Shuffled total	4,642,300	2,470,844	235,640	4,355,392	156,401
Original only	6,200	4,741	2,255	4,004	123
Shuffled only	6,164	4,759	2,205	4,031	128
Original only (in dbSNP 138)	4,620	3,580	1,732	1,508	62
Shuffled only (in dbSNP 138)	4,675	3,590	1,745	1,550	68
Original only (homozygous)	1,653	1,350	463	1,249	28
Shuffled only (homozygous)	1,698	1,391	488	1,368	35
Original only (SNV)	5,113	3,895	1,932	3,212	113
Shuffled only (SNV)	5,164	3,967	1,940	3,342	115

We list the total number of (SNV)s and indels characterized by Platypus using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S12: Summary of DELLY call sets (deletions).

DELLY	**HG00096**				
Deletions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	1,325	1,052	495	460	17
Shuffled total	1,323	1,054	491	453	15
Original only	37	32	21	7	2
Shuffled only	35	34	17	0	0
Original only (homozygous)	3	2	1	2	0
Shuffled only (homozygous)	4	3	2	0	0
DELLY	**HG02107**				
Deletions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	13,517	9,237	3,372	7,999	489
Shuffled total	13,505	9,229	3,365	7,997	490
Original only	831	698	273	158	1
Shuffled only	819	690	266	156	2
Original only (homozygous)	28	22	12	15	0
Shuffled only (homozygous)	25	19	6	12	0

We list the total number of deletions characterized by DELLY using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S13: Summary of DELLY call sets (tandem duplications).

DELLY	**HG00096**				
Tandem dups.	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	767	677	390	242	2
Shuffled total	773	681	392	246	2
Original only	25	24	15	2	0
Shuffled only	31	28	17	6	0
Original only (homozygous)	3	3	1	0	0
Shuffled only (homozygous)	0	0	0	0	0
DELLY	**HG02107**				
Tandem dups.	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	6,389	5,157	2,423	2,036	114
Shuffled total	6,457	5,229	2,446	2,047	113
Original only	384	334	152	61	1
Shuffled only	452	406	175	72	0
Original only (homozygous)	13	10	3	3	0
Shuffled only (homozygous)	9	9	4	1	0

We list the total number of duplications characterized by DELLY using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S14: Summary of DELLY call sets (inversions).

DELLY	**HG00096**				
Inversions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	610	545	202	63	2
Shuffled total	621	557	209	64	2
Original only	30	29	14	1	0
Shuffled only	41	41	21	2	0
Original only (homozygous)	5	5	3	0	0
Shuffled only (homozygous)	1	1	1	0	0
DELLY	**HG02107**				
Inversions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	7,242	5,915	2,757	1,968	185
Shuffled total	7,317	5,977	2,804	1,990	183
Original only	465	414	148	25	2
Shuffled only	540	476	195	47	0
Original only (homozygous)	14	13	5	0	0
Shuffled only (homozygous)	30	29	8	0	0

We list the total number of inversions characterized by DELLY using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S15: Summary of DELLY call sets (translocations).

DELLY	**HG00096**				
Translocations	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	4,732	4,449	1,533	300	13
Shuffled total	4,722	4,443	1,519	303	13
Original only	165	153	62	1	0
Shuffled only	155	147	48	4	0
Original only (homozygous)	9	9	2	0	0
Shuffled only (homozygous)	17	17	4	0	0
DELLY	**HG02107**				
Translocations	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	59,362	45,595	18,445	12,175	582
Shuffled total	60,337	46,358	18,684	12,223	577
Original only	3,126	2,407	1,021	229	13
Shuffled only	4,101	3,170	1,260	277	8
Original only (homozygous)	105	75	41	6	0
Shuffled only (homozygous)	107	74	27	8	0

We list the total number of translocations characterized by DELLY using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S16: Summary of Genome STRiP call sets (deletions).

Genome STRiP	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	1,218	1,139	320	398	48
Shuffled total	1,212	1,133	320	395	46
Original only	25	24	6	6	2
Shuffled only	25	18	6	4	0
Original only (homozygous)	1	1	0	0	0
Shuffled only (homozygous)	1	1	0	0	0
Genome STRiP	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	3,452	3,316	1,749	864	488
Shuffled total	3,477	3,340	1,769	861	486
Original only	483	468	298	80	55
Shuffled only	508	493	319	77	53
Original only (homozygous)	4	4	0	3	1
Shuffled only (homozygous)	5	5	0	3	1

We list the total number of (SV)s characterized by Genome STRiP using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S17: Summary of LUMPY call sets (deletions, chromosome 3 only).

LUMPY	**HG02107 Chromosome 3**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original total	307	255	36	137	27
Shuffled total	300	248	36	135	27
Original only	9	9	0	2	0
Shuffled only	2	2	0	0	0
Original only (homozygous)	1	1	0	0	0
Shuffled only (homozygous)	1	1	0	0	0

We list the total number of (SV)s characterized by LUMPY using Original vs Shuffled order reads, and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S18: Summary of HaplotypeCaller call sets using the same BAM file twice.

HaplotypeCaller	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	2,279,678	1,160,574	83,944	2,098,633	53,726
Original_2 total	2,290,542	1,169,155	90,115	2,104,565	53,945
Original_1 only	10,646	8,396	1,689	8,681	151
Original_2 only	21,510	16,977	7,860	14,613	370
Original_1 only (in dbSNP 138)	10,599	8,354	1,680	4,112	136
Original_2 only (in dbSNP 138)	20,374	16,213	7,439	6,912	316
Original_1 only (homozygous)	10,564	8,343	1,664	8,631	151
Original_2 only (homozygous)	21,233	16,805	7,691	14,523	367
Original_1 only (SNV)	10,630	8,382	1,682	8,677	151
Original_2 only (SNV)	21,499	16,968	7,854	14,612	370
HaplotypeCaller	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	4,654,338	2,423,277	149,867	4,317,854	118,842
Original_2 total	4,717,918	2,480,878	163,546	4,368,535	119,862
Original_1 only	44,293	34,901	4,629	39,235	715
Original_2 only	107,873	92,502	18,308	89,916	1,735
Original_1 only (in dbSNP 138)	41,782	33,034	4,231	17,475	499
Original_2 only (in dbSNP 138)	102,255	87,936	17,222	40,172	1,282
Original_1 only (homozygous)	10,442	8,455	1,293	9,351	154
Original_2 only (homozygous)	33,704	29,413	6,519	26,681	547
Original_1 only (SNV)	44,148	34,770	4,556	39,189	715
Original_2 only (SNV)	107,697	92,350	18,224	89,880	1,733
HaplotypeCaller	**12 WES Data Sets**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	2,904,954	1,344,022	123,450	2,958,114	152,407
Original_2 total	2,866,885	1,302,256	114,435	2,939,566	156,298
Original_1 only	55,539	45,888	13,570	45,310	1,648
Original_2 only	17,470	4,122	4,555	26,762	5,539
Original_1 only (homozygous)	46,794	39,579	10,589	38,178	901
Original_2 only (homozygous)	9,312	2,726	2,432	13,345	1,972
Original_1 only (SNV)	55,491	45,861	13,547	45,288	1,647
Original_2 only (SNV)	17,429	4,114	4,542	26,700	5,526

We list the total number of (SNV)s and indels characterized by HaplotypeCaller using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S19: Summary of UnifiedGenotyper call sets using the same BAM file twice.

UnifiedGenotyper	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	2,470,690	1,335,457	201,057	2,163,192	56,994
Original_2 total	2,470,091	1,334,992	200,980	2,162,754	56,982
Original_1 only	20,107	10,412	958	17,979	447
Original_2 only	19,508	9,947	881	17,541	435
Original_1 only (in dbSNP 138)	19,618	9,943	782	8,186	318
Original_2 only (in dbSNP 138)	19,085	9,552	747	8,073	282
Original_1 only (homozygous)	19,260	9,599	527	17,965	447
Original_2 only (homozygous)	18,771	9,236	508	17,534	434
Original_1 only (SNV)	20,077	10,388	941	17,972	447
Original_2 only (SNV)	19,481	9,923	868	17,541	435
UnifiedGenotyper	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	4,489,253	2,283,267	132,449	4,174,497	117,338
Original_2 total	4,729,796	2,485,831	179,785	4,369,974	121,086
Original_1 only	39,112	30,338	9,325	29,770	902
Original_2 only	279,655	232,902	56,661	225,247	4,650
Original_1 only (in dbSNP 138)	38,378	29,804	9,209	13,995	689
Original_2 only (in dbSNP 138)	266,185	222,200	54,265	102,812	3,440
Original_1 only (homozygous)	31,498	24,884	8,070	22,873	718
Original_2 only (homozygous)	15,586	12,444	2,537	12,903	279
Original_1 only (SNV)	38,991	30,241	9,275	29,731	899
Original_2 only (SNV)	279,518	232,800	56,605	225,180	4,650
UnifiedGenotyper	**12 WES Data Sets**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	3,342,830	1,577,426	136,341	3,342,756	159,280
Original_2 total	3,395,316	1,622,844	147,204	3,386,814	159,323
Original_1 only	12,526	3,512	4,513	15,880	3,558
Original_2 only	65,012	48,930	15,376	59,938	3,601
Original_1 only (homozygous)	7,704	2,508	2,550	9,329	1,628
Original_2 only (homozygous)	54,711	43,837	12,640	47,215	1,691
Original_1 only (SNV)	12,458	3,497	4,480	15,789	3,539
Original_2 only (SNV)	64,941	48,891	15,347	59,854	3,588

We list the total number of (SNV)s and indels characterized by UnifiedGenotyper using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S20: Summary of Freebayes call sets using the same BAM file twice.

Freebayes	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	2,400,545	1,336,125	172,214	2,188,094	73,623
$Original_2$ total	2,400,545	1,336,125	172,214	2,188,094	73,623
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (in dbSNP 138)	0	0	0	0	0
$Original_2$ only (in dbSNP 138)	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0
$Original_1$ only (SNV)	0	0	0	0	0
$Original_2$ only (SNV)	0	0	0	0	0
Freebayes	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	5,174,644	2,906,071	380,096	4,735,347	170,214
$Original_2$ total	5,174,644	2,906,071	380,096	4,735,347	170,214
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (in dbSNP 138)	0	0	0	0	0
$Original_2$ only (in dbSNP 138)	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0
$Original_1$ only (SNV)	0	0	0	0	0
$Original_2$ only (SNV)	0	0	0	0	0

We list the total number of (SNV)s and indels characterized by Freebayes using same BAM file twice ($Original_1$ vs $Original_2$), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S21: Summary of SAMtools call sets using the same BAM file twice.

SAMtools	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	2,277,691	1,269,164	176,433	2,055,105	69,037
$Original_2$ total	2,277,691	1,269,164	176,433	2,055,105	69,037
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (in dbSNP 138)	0	0	0	0	0
$Original_2$ only (in dbSNP 138)	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0
$Original_1$ only (SNV)	0	0	0	0	0
$Original_2$ only (SNV)	0	0	0	0	0
SAMtools	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	5,355,604	3,055,463	417,874	4,862,553	175,803
$Original_2$ total	5,355,604	3,055,463	417,874	4,862,553	175,803
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (in dbSNP 138)	0	0	0	0	0
$Original_2$ only (in dbSNP 138)	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0
$Original_1$ only (SNV)	0	0	0	0	0
$Original_2$ only (SNV)	0	0	0	0	0

We list the total number of (SNV)s and indels characterized by SAMtools using same BAM file twice ($Original_1$ vs $Original_2$), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S22: Summary of Platypus call sets using the same BAM file twice.

Platypus	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	2,022,412	1,073,056	115,590	1,843,781	47,144
Original_2 total	2,022,412	1,073,056	115,590	1,843,781	47,144
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (in dbSNP 138)	0	0	0	0	0
Original_2 only (in dbSNP 138)	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0
Original_1 only (SNV)	0	0	0	0	0
Original_2 only (SNV)	0	0	0	0	0
Platypus	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	4,642,336	2,470,826	235,690	4,230,891	116,859
Original_2 total	4,642,336	2,470,826	235,690	4,230,891	116,859
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (in dbSNP 138)	0	0	0	0	0
Original_2 only (in dbSNP 138)	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0
Original_1 only (SNV)	0	0	0	0	0
Original_2 only (SNV)	0	0	0	0	0

We list the total number of (SNV)s and indels characterized by Platypus using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S23: Summary of DELLY call sets using the same BAM file twice (deletions).

DELLY	**HG00096**				
Deletions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	1,325	1,052	495	485	23
$Original_2$ total	1,325	1,052	495	485	23
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0
DELLY	**HG02107**				
Deletions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	13,517	9,237	3,372	7,999	489
$Original_2$ total	13,517	9,237	3,372	7,999	489
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0

We list the total number of deletions characterized by DELLY using same BAM file twice ($Original_1$ vs $Original_2$), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S24: Summary of DELLY call sets using the same BAM file twice (tandem duplications).

DELLY	**HG00096**				
Tandem dups.	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	767	677	390	249	3
$Original_2$ total	767	677	390	249	3
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0
DELLY	**HG02107**				
Tandem dups.	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
$Original_1$ total	6,389	5,157	2,423	2,036	114
$Original_2$ total	6,389	5,157	2,423	2,036	114
$Original_1$ only	0	0	0	0	0
$Original_2$ only	0	0	0	0	0
$Original_1$ only (homozygous)	0	0	0	0	0
$Original_2$ only (homozygous)	0	0	0	0	0

We list the total number of duplications characterized by DELLY using same BAM file twice ($Original_1$ vs $Original_2$), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S25: Summary of DELLY call sets using the same BAM file twice (inversions).

DELLY	**HG00096**				
Inversions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	610	545	202	63	2
Original_2 total	610	545	202	63	2
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0
DELLY	**HG02107**				
Inversions	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	7,242	5,915	2,757	1,968	185
Original_2 total	7,242	5,915	2,757	1,968	185
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0

We list the total number of inversions characterized by DELLY using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S26: Summary of DELLY call sets using the same BAM file twice (translocations).

DELLY	**HG00096**				
Translocations	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	4,732	4,449	1,533	348	20
Original_2 total	4,732	4,449	1,533	348	20
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0
DELLY	**HG02107**				
Translocations	*All*	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	59,362	45,595	18,445	12,175	582
Original_2 total	59,362	45,595	18,445	12,175	582
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0

We list the total number of translocations characterized by DELLY using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S27: Summary of Genome STRiP call sets using the same BAM file twice (deletions).

Genome STRiP	**HG00096**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	1,218	1,139	320	398	48
Original_2 total	1,218	1,139	320	398	48
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0
Genome STRiP	**HG02107**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	3,452	3,316	1,749	864	488
Original_2 total	3,452	3,316	1,749	864	488
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0

We list the total number of (SV)s characterized by Genome STRiP using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S28: Summary of LUMPY call sets using the same BAM file twice (deletions, chromosome 3 only).

LUMPY	**HG02107 Chromosome 3**				
	All	*Repeat*	*Duplication*	*Gene*	*Exon*
Original_1 total	307	255	36	137	27
Original_2 total	307	255	36	137	27
Original_1 only	0	0	0	0	0
Original_2 only	0	0	0	0	0
Original_1 only (homozygous)	0	0	0	0	0
Original_2 only (homozygous)	0	0	0	0	0

We list the total number of (SV)s characterized by LUMPY using same BAM file twice (Original_1 vs Original_2), and further classify total and private calls that map to common repeats, segmental duplications, gene models, and coding exons.

Supplementary Table S29: Summary of HaplotypeCaller call sets from 12 WES data sets using the same BAM file twice.

	Original$_1$		Original$_2$	
HaplotypeCaller	*All*	*Private*	*All*	*Private*
HG00250	57,102	801	58,757	2,456
HG00251	58,216	810	59,888	2,482
HG00330	57,774	847	59,389	2,462
HG00731	52,623	719	54,605	2,701
HG00732	56,625	799	58,547	2,721
HG01075	81,556	1,219	83,102	2,765
HG01083	62,035	861	63,676	2,502
HG01133	55,079	670	56,736	2,327
HG01136	55,760	666	57,491	2,397
HG01140	59,226	732	60,772	2,278
HG01167	53,422	805	55,434	2,817
HG01176	84,434	1,252	86,091	2,909

We list the total number of (SNV)s and indels characterized by HaplotypeCaller simultaneously in 12 WES data sets using same BAM files twice (Original$_1$ vs Original$_2$). Private: variants found in one execution of HaplotypeCaller and not the other.

Supplementary Table S30: Performance of callers compared to 1000 Genomes OMNI SNP chip using the same samples.

OMNI	**HG00096**							
	True Positive	*False Positive*	*True Negative*	*False Negative*	*Genotype Error*	*Precision*	*Recall*	*Accuracy*
HaplotypeCaller 1	410,031	5,424	1,678,715	288,997	32,032	0.91629	0.58657	0.86000
HaplotypeCaller 2	409,904	5,424	1,678,715	289,153	32,003	0.91633	0.58636	0.86000
HaplotypeCaller Shuffled	409,943	5,422	1,678,717	289,122	31,995	0.91636	0.59000	0.86000
FreeBayes	368,576	726	1,683,413	328,626	33,858	0.91421	0.52865	0.84961
FreeBayes Shuffled	368,581	723	1,683,416	328,623	33,856	0.91423	0.52865	0.84961
SAMtools	357,952	1,587	1,682,552	344,158	28,952	0.92139	0.50982	0.84485
SAMtools Shuffled	357,961	1,590	1,682,549	344,147	28,954	0.92138	0.50983	0.84486
Platypus	299,609	5,338	1,678,771	375,215	25,237	0.90740	0.44398	0.82979
Platypus Shuffled	299,611	5,340	1,678,769	375,206	25,242	0.90738	0.44398	0.82979
OMNI	**HG02107**							
	True Positive	*False Positive*	*True Negative*	*False Negative*	*Genotype Error*	*Precision*	*Recall*	*Accuracy*
HaplotypeCaller 1	768,092	10,582	1,614,337	28,838	1,260	0.98481	0.96381	0.98321
HaplotypeCaller 2	768,302	10,615	1,614,304	28,569	1,319	0.98470	0.96414	0.98328
HaplotypeCaller Shuffled	767,494	10,629	1,614,290	29,443	1,253	0.98475	0.96305	0.98294
FreeBayes	754,824	1,772	1,623,147	42,128	1,238	0.99602	0.94713	0.98137
FreeBayes Shuffled	755,205	1,799	1,623,120	41,744	1,241	0.99599	0.94762	0.98151
SAMtools	768,616	3,414	1,621,505	28,301	1,297	0.99390	0.96448	0.98637
SAMtools Shuffled	768,615	3,408	1,621,511	28,303	1,296	0.99391	0.96448	0.98637
Platypus	618,355	7,828	1,617,091	58,376	1,111	0.98574	0.91373	0.97076
Platypus Shuffled	618,353	7,826	1,617,093	58,374	1,114	0.98574	0.91374	0.97076

We compare the call sets generated with different callers (original and shuffled) with the genotypes characterized by the 1000 Genomes Project using high density OMNI SNP chips.

Supplementary Table S31: Performance of callers compared to 1000 Genomes call set using the same samples.

1000G	**HG00096**							
	True Positive	*False Positive*	*True Negative*	*False Negative*	*Genotype Error*	*Precision*	*Recall*	*Accuracy*
HaplotypeCaller 1	1,755,897	7,720	74,394,633	7,720	180,912	0.90299	0.99562	0.99742
HaplotypeCaller 2	2,062,619	15,209	74,387,144	15,209	209,042	0.90193	0.99268	0.99687
HaplotypeCaller Shuffled	1,763,761	7,790	74,394,563	7,790	182,710	0.90252	0.99560	0.99740
FreeBayes	1,702,886	6,977	74,395,377	6,977	210,308	0.88684	0.99591	0.99706
FreeBayes Shuffled	1,702,929	6,971	74,395,383	6,971	210,310	0.88684	0.99592	0.99706
SAMtools	1,672,206	9,676	74,392,677	9,676	178,863	0.89867	0.99424	0.99740
SAMtools Shuffled	1,672,232	9,666	74,392,687	9,666	178,873	0.89867	0.99425	0.99740
Platypus	1,458,048	7,297	74,395,056	7,297	147,975	0.90375	0.99502	0.997861
Platypus Shuffled	1,457,995	7,312	74,395,041	7,312	147,990	0.90373	0.99500	0.99786
1000G	**HG02107**							
	True Positive	*False Positive*	*True Negative*	*False Negative*	*Genotype Error*	*Precision*	*Recall*	*Accuracy*
HaplotypeCaller 1	3,705,845	12,862	73,569,468	12,862	102,624	0.96977	0.99654	0.99834
HaplotypeCaller 2	4,199,867	27,802	73,554,528	27,802	128,157	0.96419	0.99342	0.99764
HaplotypeCaller Shuffled	3,676,596	12,920	73,569,410	12,920	102,379	0.96959	0.99649	0.99834
FreeBayes	3,997,561	9,266	73,573,069	9,266	115,627	0.96970	0.99768	0.99827
FreeBayes Shuffled	4,002,945	9,377	73,572,959	9,377	115,666	0.96970	0.99766	0.99827
SAMtools	4,170,590	15,272	73,567,061	15,272	122,851	0.96794	0.99635	0.99803
SAMtools Shuffled	4,170,654	15,235	73,567,098	15,235	122,805	0.96796	0.99636	0.99803
Platypus	3,723,505	11,094	73,571,236	11,094	101,341	0.97068	0.99702	0.99840
Platypus Shuffled	3,723,629	11,079	73,571,251	11,079	101,332	0.97069	0.99703	0.99840

We compare the call sets generated with different callers (original and shuffled) with the genotypes characterized by the 1000 Genomes Project using the same data sets. Since the 1000 Genomes final release call set is extensively curated, we assume that it represents the gold standard for the genomes we analyzed.

Supplementary Table S32: Correction accuracy given high breadth of coverage (> 90%) in CH17-157L1.

Tool	No. of Aligned Reads			
	Mapped	80-90%	90-95%	>95%
Uncorrected	1,575	9,22	599	1
Colormap	1,578	162	513	873
HALC	1,579	780	746	11
LoRDEC	1,574	71	258	1,223
LSC	1,585	24	128	1,424
Hercules	**1,585**	18	112	**1,449**
proovread	1,575	726	467	334

The input included 2,515 PacBio reads, which at least 90% of their length were covered by short reads. We report the accuracy as the alignment identity as calculated by the BLASR [57] aligner. *Mapped* column refers to the number of any reads aligned to the Sanger-assembled reference for this clone, where the other columns show the number of alignments within respective identity brackets.

Supplementary Table S33: Effects of sequence coverage on correction accuracy for the CH17-157L1 BAC clone.

Depth	Number of aligned reads											
	Hercules				LoRDEC				LSC			
	Mapped	80-90%	90-95%	>95%	Mapped	80-90%	90-95%	>95%	Mapped	80-90%	90-95%	>95%
20x	40,359	14,251	7,136	7,835	37,672	11,315	7,249	11,441	38,340	14,011	6,569	6,848
10x	38,333	14,513	6,478	5,851	36,575	13,962	6,917	5,937	37,623	14,489	6,222	5,613
5x	36,401	15,326	6,108	2,957	36,028	16,579	4,566	2,849	36,329	15,284	5,797	3,288
1x	35,007	16,917	3,975	901	34,843	17,760	2,054	1,335	34,738	17,008	3,679	8,82

The depth of coverage of Illumina reads for the original data set is 245X. Here, we down-sampled the short reads to obtain coverages of 20x, 10x, 5x, and 1x. We provided Hercules LoRDEC, and LSC with down-sampled short reads to evaluate the effects of sequence coverage on correction accuracy. *Mapped* column refers to the number of any reads that are aligned to the Sanger-assembled reference for this clone with any identity, where the other columns show the number of alignments within respective identity brackets.

Appendix B

Code

We release our FASTQ read shuffling tool, shell scripts to map reads and call variants, and the VCF files generated for this thesis at the Zenodo data archival site. The DOI for this submission is 10.5281/zenodo.32611.

The source code for *Hercules* is available under the BSD 3-clause license at https://github.com/BilkentCompGen/Hercules.