UNIVERSITY OF CALIFORNIA
SANTA CRUZ

METHODOLOGICAL ADVANCEMENTS FOR GENOME RECONSTRUCTION BY HAPLOTYPING LONG READ SEQUENCE DATA

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

BIOMOLECULAR ENGINEERING AND BIOINFORMATICS

by

Trevor W. Pesout

March 2022

The Dissertation of Trevor W. Pesout
is approved:

____
Professor Benedict Paten, Chair

____
Professor David Haussler

____
Professor Tobias Marschall

____
Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Methodological advancements for genome reconstruction by haplotyping long read sequence data

by

Trevor W. Pesout

Second-generation sequencing technology and accompanying analyses resulted in a deluge of information about variation in human populations, enabling large-scale association studies and precision medicine. However, there are genomic contexts which cannot be analyzed using these technologies. With the advent of long-read sequencing, previously unmappable regions of the genome have become accessible, paving the way for more comprehensive analyses of the human genome. However, new methods are required to leverage the increased length of these data as well as mitigate the poor sequence accuracy. In this work, I present an accurate and efficient application "Margin", which uses a Hidden Markov Model to separate read and variant data into haplotypes. I describe work to validate the method and show applicability in variant calling, I demonstrate ways to overcome systematic errors in nanopore sequence data and correct assembled sequence, and I document the tool's use in a state-of-the-art variant caller for Oxford Nanopore and PacBio HiFi data used to generate reference materials and make medical diagnoses.

To my parents,

and to

Rule Number One.

Acknowledgments

To start, I would like to thank my advisor, Benedict Paten. Benedict was welcoming when I approached him looking for a lab and was supportive even though my education in biology consisted of a single class taken in high school. A grad student needs a leader with sense of direction, and Benedict has a keen eye for the appropriate high-level goals to move the field forward, knowing the kind of analyses which will be useful years down the line. A grad student also needs someone who can help with the details of a task, and Benedict has a powerful grasp of low-level minutiae for an array of methods and theory, and he is able (and willing) to explain them. Beyond this, Ben has been supportive of who I am as a person, including the priorities I have between my personal and professional life, the way that I communicate, and the moral choices I've had to make during my time here. Lastly, and I mention this because I think that its significance is often not recognized, he is fun to be around, has a sense of humor, and doesn't take himself too seriously. Being well-rounded is important, I value that in him, and I think it makes him a better PI.

I would also like to thank the other members of my committee. David Haussler is a legend in the field, and I am incredibly grateful to have had his advice and direction during the course of my PhD. He has played a large part in the success and renown of the department at UC Santa Cruz and has pushed the field forward in countless ways; I am grateful to him for that as well. Tobias Marschall has also played an integral part in my development. There are many fields which are cutthroat and competitive, including Computer Science where I started my doctoral studies. My first real project as a bioinformatics student involved an idea which Benedict and Tobias had both come to, and instead of trying to race the other to publication, we all worked together to evaluate it and demonstrate its use. This was a welcome surprise for me and it set the stage for the rest of my work, where sharing and collaboration have been at the forefront of our priorities.

To that point, I have found the bioinformatics field to be generally collaborative, emphasizing the sharing of data and theory. While it feels silly to thank an entire field, I would like to at least note that I think this is exceptional and am proud that the broader intellectual community I'm part of works towards making things better for everyone instead of just themselves. This includes the BME department here at UC Santa Cruz which fosters these morals in the student body.

I would like to thank my entire lab, including my fellow grad students, the engineers I've worked with, and the administrative and support staff. In particular, working closely with my friend and teammate Kishwar Shafin has been an honor, he is brilliant and motivated and has been integral to the success of the many projects we worked on together. Additionally, he is a profoundly moral person and has a scathing wit, both of which I appreciate. I would also like to thank my cohort in the lab: Marina Haukness, Ryan Lorig-Roach, and Colleen Bosworth. I'm so happy to have grown with you all and appreciate having had you there to celebrate and commiserate with. I would like to thank Arjun Rao, John Vivian, and Jordan Eizinga as role models in the lab who helped me get started and keep going. Jordan in particular has been instrumental in my growth as a student and a laborer and a human. I would also like to thank Miten Jain, who is a pleasure to work with personally and intellectually and has provided a foundation of work on which I depended.

Thank you also to the grad student community here at Santa Cruz. It has been incredibly rewarding and exhilarating to get to know everyone and to hear about the work that is being done as well as the people who are doing it. Far more importantly, it's been great to drink beer with you all and sing and dance and cry and run around and play inter-tube water polo and watch Cats after midnight and play frisbee and basketball and drink in the park when we couldn't go into a brewery and catch Pokemon and dress up and have bonfires and jump in the ocean and fight the power and eat pizza and talk shit and swim down the river and ride bikes and play cards and all the other things that we did.

I also need to express my deep gratitude to all the students, faculty, and campus community members who were involved in labor movements towards a more just and equitable work environment. Santa Cruz housing costs have become progressively more expensive, and in late 2019 enough students had reached a breaking point that action needed to be taken. Students used an array of peaceful methods to show the necessity of a cost-of-living adjustment, and as a result had to deal with stonewalling, government-grade surveillance, racial profiling, and violence. I am grateful to the COLA wildcats for the risks they took and the sacrifices they made on behalf of graduate students. This labor action in part lead to the formation of the Student Researchers United, a union of graduate student researchers. I am grateful to all those involved in this process, and am proud to have contributed to the effort. There are many who were involved in this process, but I would especially like to acknowledge Jesse Suzuki who was willing to represent students at legislative and judicial hearings, and to Jordan Eizinga who motivated me as well as many in the department to participate in this process. I would lastly like to thank many of the BME faculty for their support and assistance during the unionization campaign, in particular Angela Brooks for organizing a letter of support from the faculty.

Santa Cruz has been my home for most of my adult life and I have many different communities here which I want to acknowledge. First and foremost is the Neighborhood Night Crew with whom I've spent the majority of my Tuesday nights, including Scott, Sylvie, Hope, Maria, Adina, Eric, Ben, Alix, Alix, Dave, Emma, Gerrald, Steph, Tatty, and so many more. You all have been such a joy and I love all of you. I play Ultimate Frisbee on the beach and Goaltimate on the grass (shoutout to Coop for consistently organizing) with a huge group of people I'm very grateful for. I'd like to thank the folks at ProductOps for their love and support: Bob, Dean, Paul, Catherine, Chloe, and many more.

Lastly I'd like to thank my family. My partner Adina is a true joy in my life, thank you for your support in this process and all the fun we have together. I'd like to thank my cats, although I'm sure they don't care: Calvin, Dorito (RIP), Rainey and Chloe. My sister Sarah is my oldest companion and I'm very proud of her and grateful that she is in my life. Finally, my parents Ann and Jim who have supported me in so many ways during my doctoral work and beyond. I can't express enough how much the person I am and all that I have accomplished is a product of the loving and nurturing environment I grew up in, thank you so much and I love you forever.

Part I

Introduction and Background

Chapter 1

Introduction

Over two decades ago the first expansive construction of the human genome completed, costing three hundred million dollars, taking fifteen months to finish, and requiring work from over two hundred and fifty scientists from across the globe. Today, reconstruction of a human genome can go from 2ml of blood to genetic medical diagnoses in less than a standard workday. Innovations in computational approaches and physical methodologies have moved the field forward, as well as layers of understanding built upon new conceptual frameworks. This has involved thousands of students and scientists and engineers making incremental progress in a variety of domains. As with many doctoral studies, my work represents a small improvement on a specific analysis towards a general goal using a particular type of data.

At a high level, the goal we're moving towards is accurate reconstruction of genetic sequence. As with all improving fields, how we do this has addressed increasingly difficult areas of this problem. Twenty years ago, this meant producing a general image of what the simplest parts of the human genome looked like. Reference-dependent variant calling methods were then developed to assay the variety in human populations, using sequencing technologies that are inexpensive and produce accurate (but small) pieces of data. As we got good at this type of genetic inspection, it became apparent that there are areas of the genome which will never be successfully analyzed using existing methods on this type of data. These hard-to-reach areas, specifically where genetic sequences are highly repetitive or exist in multiple places in the genome, need a different data modality to be accessible.

The development of the nanopore sequencer (the particular type of data my dissertation focuses on) produces sequencing data long enough that it can be used successfully in these contexts. However, this new datatype is not a strict improvement on older sequencing technologies; they produced accurate sequence with random errors, and nanopore sequencers produce inaccurate sequence with systematic errors. We needed new methods to leverage the increased length as well as strategies to ameliorate the poor sequence accuracy.

One of these analyses (the analysis on which I have spent my dissertation working) is the determination of haplotypes in a collection of long reads. As humans have two copies of each chromosome, we can theoretically separate sequencing data into two partitions. This partitioning enables downstream methods to perform haplotype-specific analyses, which more accurately reflects the underlying genetic structure, which in turn leads to improved sequence reconstruction. If we can figure out how to use long reads to effectively reconstruct genetic sequence, we can use it to build understanding of previously inaccessible regions of the genome, and this knowledge will enable the next generation of scientists and specialists to focus on harder problems and design better methodologies.

This brings us to the specific work I've done for my dissertation. The bulk of this work has revolved around a single code base called *Margin*, descended from a generalized Hidden Markov Model implementation that my advisor Benedict Paten wrote when he was a grad student. After joining the lab in 2016, I was given a "theoretically functional" project which could accurately phase simulated reads. All we needed to do was figure out how to make it work on real data.

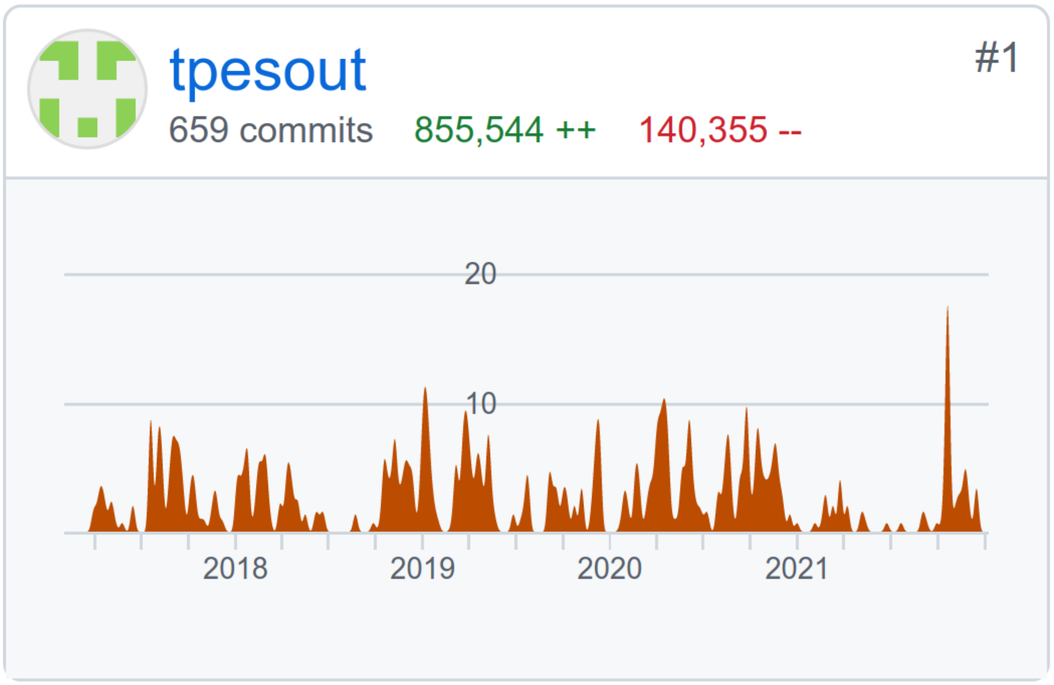


Figure 1.1: "Just make it work on real data"

Five years and eight hundred thousand lines of code later, "make it work" ended up being marginally more complicated than I expected. From a bird's eye view, the task was simple and straightforward. The view from the ground was full of weeds and dirt and an incredible variety of bugs. The Sequence Alignment/Map codebase is an inscrutable labyrinth, comprehensible only by Heng Li and `gcc`. Error profiles for nanopore data are different for species, strand, sequencer, basecaller versions, and probably the radio's volume during DNA extraction. Homopolymers in the DNA result in systematic errors in the data, but compressing them helps (plus it improves alignment!), but that unfortunately makes alignment worse, so you definitely shouldn't do it. Apparently some troglodyte decided that switching the parameter ordering of `qsort_r` between BSD and GNU was a good idea, and my sanity and self respect withered to dust in between the `#ifdef`'s needed to contain the problem.

All that being said, it works! I've built a tool that can effienctly and accurately turn long reads on a reference and a set of heterozygous variants into haplotyped sequence data. At first it worked slowly and expensively and poorly, but after five years of development a whole genome run went from days and thousands of dollars, to hours and tens of dollars, to 36 minutes and $1.35. Not everyone in the field cares about the time and cost a tool takes to run, but as someone who has spent more time thinking of himself as a computer scientist than a bioinformatician, I can tell you that efficiency is critical in any application you want to use at scale. If you want it to be useful everywhere, it needs to be fast and cheap and easy.

The bioinformatics landscape is full of tools which are not these things. It is rife with half-baked graduate student "code" designed to be run manually in a very specific environment using data in non-standardized formats which has since been deleted. Programs which can only really be run by their author and were only even built to show proof of concept in a manuscript, after which the tool is discarded or (at best) forgotten. Finding out some other poor rube is trying to use their code sends shivers down the weary and hunched spines of these students.

I jest, but there is truth to the idea that much in the world of bioinformatics is not production ready. When reading this manuscript, it is my hope that the reader finds the tool I've developed does not fall into this category. I show consistent improvements in efficiency, usability, and accuracy. In Chapter 3, I validate the core phasing algorithm and demonstrate that it improves variant calling performance. In Chapter 4, I improve the tool to better handle nanopore data, and to identify and correct variant sites in genome assemblies. In Chapter 5, I show that the tool is refined and performant, and I detail its inclusion in a state-of-the-art variant calling pipeline. In Chapter 6, I briefly describe how it has been applied in a project generating ultra-fast clinical diagnoses.

Chapter 2

Background

To lay out all the knowledge and methodologies upon which my dissertation builds would take far longer than it took to complete my dissertation. I include in this background section topics which are directly applicable or relevant to my work, and focus specifically on human analyses.

2.1 DNA Sequencing Technologies

The first widespread method of sequencing DNA was developed in 1977 by Frederick Sanger [142], after whom the technology was named. Sanger sequencing involves use of dideoxythymidine triphosphate during DNA replication, stopping replication and leaving a residue for which the length can be measured using electrophoresis on acrylamide gels. This process can be replicated for each of the four nucleotides, enabling an accurate determination of the nucleotide sequence. An improved version of this method is still used today, as it can accurately (99.999% accuracy) generate sequences up to 1000bp in length [150].

This methodology was largely eclipsed in use by technologies colloquially referred to as second-generation sequencing, next-generation sequencing, or shotgun sequencing. There are many implementations of this methodology, but all follow a similar high-level workflows involving DNA fragmentation, adapter ligation, amplification, and florescence-based imaging [150]. Illumina sequencing [13] is the most common of these methods, where a single-stranded DNA molecule is repeatedly annealed to an array, extended, and then denatured to create a cluster of clonal copies. These clusters are finally observed in process where a removable fluorophore label is attached at each round of synthesis, a laser is used to excite the label, and the nucleotide can be determined optically. Furthermore, this allows for an estimation of accuracy for each base, which is used in downstream processing. The process is highly parallelizable resulting in a cheaper method for production of larger amounts of sequence data, albeit less accurate (99-99.9% accuracy) and with smaller sizes (50-300bp), than Sanger sequencing. Illumina sequencing often is performed in paired-end mode, where a size-selected piece of DNA has both ends sequenced to produce a pair of reads which are known to originate from the same molecule.

Third-generation sequencing, sometimes referred to as long read sequencing, is dominated by two different technologies and implementations both which directly measure DNA molecules instead of a product of replication. Pacific Biosciences uses fluorophores incorporated during DNA synthesis to determine the nucleotide sequence [42]. Recent improvements to the technology involve circularizing the DNA and generating a consensus sequence by reading the molecule multiple times, producing high-fidelity or HiFi data with 10-30kb read lengths and accuracies up to 99-99.9% [160, 171].

The second long read sequencing technology was conceived of by researchers at UCSC [33] and turned into a consumer product by Oxford Nanopore Technologies (ONT). Nanopore sequencing attaches a motor protein to a DNA molecule which is used to feed a single strand through a pore in a membrane using an applied electric field. Current is measured as nucleotides translocate through the pore, and the underlying sequence can be inferred from these data [69]. Originally this inference was performed using Hidden Markov Models, but later improvements to basecalling software have turned to deep neural nets for sequence determination. Nanopore read data are generally longer than any other methodology, with observed lengths ranging from tens of thousands to millions of bases. The impressive length of these data is offset by their relatively lower accuracy (90%+ accurate), with the majority of errors being found in homopolymers [130, 148].

2.2 Methods for Genome Inference

Determining underlying genomic sequence for a person is the necessary first step for all analyses, from large-scale studies of pathology and phenotypic association to precision medicine. Efforts to infer genomic sequence are confounded by an assortment of difficulties; first I list these difficulties and then later detail how various methods used during analysis handle (poorly and successfully) these issues.

2.2.1 Difficulties in Genome Inference

Beyond errors related to sequencing technology, the main confounding issue in genome inference is that the human genome is largely composed of repetitive and duplicated sequence.

Human centromeres and the regions around them are primarily composed of multiple different repetitive structures [108]. These are largely described as satellite DNA [174]: sequences of short (a single base pair) to long (a thousand base pairs) repeated in tandem, with this repetitive structure itself repeated. They include repeats of AT-rich 171bp sequences called alpha-satellites, which form larger repetitive units known as high-order repeats (HORs), which then organize into multi-megabase satellite arrays of which one or more can be found on human centromeres. Human telomeres are also highly repetitive, characterized by a 6bp sequence repeated to a total length of 5kb-15kb [113].

Beyond repetitive sequence, there are a classes of mobile DNA elements called transposable elements (TEs, or transposons) which can insert themselves into our DNA. They fall into functional groupings (Class I TE or retrotransposons, and Class II TE or DNA transposons), followed by further groupings based on sequence structure (Alus, LINEs, SINEs, VNTRs, SVAs), and further tracked by lineages (AluYa5, AluSx, L1Hs, L1PA2, etc). Transposable elements are thought to represent up to 44% of the human genome, with the majority of these insertion events having occurred far back in our genetic history and have since been neutered by mutation or methylation [44, 112, 36].

Beyond repetitive sequence and transposons, there are biological processes which can cause large-scale structural changes in our genome. These can include Copy Number Polymorphisms (CNPs), where hundreds of kilobases of genetic sequence is duplicated or deleted. They also include inversions, where a segment of DNA is inverted relative to a reference [145, 22]. The community has adopted the convention that any genetic event larger than 50bp (including these large-scale events as well as transposon driven changes) is deemed a Structural Variant (SV).

Beyond repetitive sequence, transposons, and structural variation, there are regions of the genome which were duplicated at some point in our ancestry. As an example, the NOTCH2NL gene family includes three paralogous copies of a protein-coding gene sequence related to brain function [47]. This class of duplication is reflected in our reference material, whereas SVs are relative to the reference (albeit an arbitrary relativity).

The final significant difficulty to most genome inference efforts is that the human genome is diploid. We inherit one set of chromosomes from each parent, so variation can exist twice at any genomic locus. When we use one of the various methods of genome inference, we have to understand the relationship between haplotypes to properly analyze or record it.

2.2.2 Human Genome Reference

The genome inference problem became much easier with the first assembly of the human genome in 2001 [84]. This process involved a massive amount of work from an international group of collaborators where they created Bacterial Artificial Chromosomes (BACs) by inserting 100-200kb of human sequence into a bacterial genome, grew a colony of this bacteria, used shotgun sequencing to generate read data, assembled the sequence inserted into the bacterial genome, and then stitched the assembled sequences together. The result was approximately 2.69 gigabases (Gb) of sequence.

The human reference sequence has been corrected and updated many times since its initial publication, with primarily two sequences being used by recent analyses. These are managed by the Genome Reference Consortium, and are referred to as GRCh37 (or hg19) and GRCh38 (or hg38) with 2.91Gb and 2.98Gb of primary non-N sequence respectively. GRCh38 includes modeled centromeres, a significant improvement over previous iterations although not truly representative of underlying sequence [143].

Recently, another human reference sequence has been generated by the Telomere-to-Telomere Consortium (T2T) using extensive sequencing, assembly, and correction of an effectively-haploid cell line CHM13 [117]. This flagship reference is the first complete representation of the human genome.

Lastly, work by the Human Pangenome Reference Consortium has been done to generate a pangenome reference: a graph of sequence data theoretically describing almost all variation in the human population. This graph-based reference is structurally much different from any of the previous linear references and will require different methodologies for use.

2.2.3 Reference-Based Methods

The reason genome inference became easier with the first human genome assembly (and all the references since then) is because the reference sequence is very similar to the underlying sequence for any given human. An average human differs from the reference at 4.1 to 5 million sites, including 23,000 to 28,000 structural variants, from a reference with over 3 billion bases [158, 36]. This similarity between the reference and a sample allows read data to be localized much more easily using a method called alignment, described in the next section.

Alignments would not be manageable without technological infrastructure to track, manage, and manipulate data. The foundation of alignment standardization is the Sequence Align/Map (SAM) format and accompanying toolkit `samtools`, written by Heng Li [95].

2.2.3.1 Alignment

Alignment is a process where a piece of sequencing data is localized on a reference and each nucleotide is matched to a position (or anchored to a position for inserts and deletes) on the reference. Generally this process involves two steps: finding the general area on the reference where the sequencing read originated from, and then finding the fine-grained alignment between the entire read and a portion of the reference. This paradigm is often referred to as *seed and extend*.

To find seeds, most aligners will either use suffix trees or some form of hashed k-mers to find exact matches. A suffix tree is a data structure used to efficiently to find exact matches for a string, and is used by the most common short-read aligner `bwa` [94]. A K-mer based methodology is used by the most common long-read aligner `minimap2` [92], which uses a hashed representation of a subset of k-mers called *minimizers* [135] for improved efficiency. Often a read will have exact matches to multiple locations in the genome; to find the most likely region of origin from seeds, aligners will cluster or chain seeds together.

Once a region has been selected, aligners will use long-standing dynamic programming alignment algorithms such as Needleman-Wunsch [116] or Smith-Waterman [154] to find the most likely exact matching of nucleotides. Of particular mention is a class of algorithms (used by `minimap2`, among others) that uses affine gap costs where the penalty for a gap does not scale linearly with size, a reflection of underlying biological likelihoods where the insertion or deletion of a sequence of length n is not n times less likely than a sequence of length one [6, 57, 58].

Because of the amount of duplicated or repetitive sequence in the reference, aligners can not be confident which duplication or repeat the read sequence most likely originated from. To address this, aligners will include a measure of confidence in the alignment (Map Quality, or MQ score) that can be used by downstream analyses. Often, low MQ scores will be accompanied by secondary alignments: alignments which are also optimal or slightly less optimal but which may still merit consideration.

Third generation long reads are particularly advantageous for inference in hard-to-map regions, as they are more likely to have an anchor in a region of unique sequence. Thus anchored, the portion of the read extending into non-unique sequence can be analyzed with confidence.

2.2.3.2 Variant Calling

After a set of reads are aligned to the reference, examining sites where the read data reflects a different nucleotide is a straightforward method of inferring how the sample differs from the reference. The visualization (for a human or a computer) of many reads overlapping a reference region is generally refered to as a *pileup*. As alignment required the SAM format, tracking variation data also requires a specification; during the first large-scale variation study, the 1000 Genomes Project developed the Variant Call Format (VCF) and accompanying toolkit `vcftools` [32] to describe differences with respect to a reference.

One of the first variant callers was `bcftools` (developed by the authors of `samtools`) which generated statistical summaries of pileups taken directly from the alignment files [95, 90]. As NGS proliferated and an array of sequence data from a variety of sources became accessible, a set of tools and workflow of normalization and analysis steps was developed. This Genome Analysis Toolkit (GATK) and their best-practices workflow became an industry standard for use with the incredible amount of highly accurate short read data that was available [106]. Recently, tools such as DeepVariant which use deep neural nets have been adopted as they outperform many of the older statistical methods [126].

These tools excel at finding small variants (called point variation) which are classified as Single Nucleotide Polymorphisms (SNPs) or Insertions/Deletions (INDELs).

Structural variation generally cannot be detected using the same paradigms. Novel sequence can't be identified by short read alignment, duplicated sequence will exist at other places in the reference resulting in untrustworthy mappings, and large scale inversions align normally to the reference except at their boundaries. Methods do exist [72, 85, 131, 4], but are less adept at identifying structural variation.

Third generation data with drastically longer lengths is much more suited to SV calling. The tool Sniffles has been adopted by much of the industry to do reference-based SV calling [146]. While long reads are better able to detect large variation, the higher error rate makes calling of point variation more difficult. In particular, nanopore sequencing can result in errors falling into consistent patterns (such as undercounting homopolymers, or biases in basecalling depending on which DNA strand is read), patterns which are consistent enough that simple pileup-based inferences will produce false positives at untenable rates.

2.2.4 *De Novo* Assembly

De Novo Assembly is the other main method of genome inference. Assembly is a reference-agnostic process, where only raw sequencing data is used to infer the underlying genomic sequence. Methods for doing this largely fall into three categories [23], all of which are graph based.

The first method is referred to as the overlap-layout-consensus (OLC) framework. In this methodology, overlaps between reads are detected, overlapping reads are merged together, and finally a consensus sequence is generated from the overlapping reads. In this framework, reads are represented as nodes, and overlaps as edges. This can be applied to both short and long reads and was used in some of the first assembly algorithms [114, 66, 11].

The second method is called de Bruijn graphs. Reads are first decomposed into overlapping kmers (GATTACA $\rightarrow$ GAT,ATT,TTA,TAC,ACA). Overlapping kmers are used to build a graph, and contigs are determined by finding paths in the graph supported by read data. In this method, kmers (sequence) are nodes, edges are overlap between nodes, and reads are represented as paths through the graph. This can be more computationally efficient than OLC methods, as pairwise relationships between reads don't need to be calculated. However, this depends on having highly accurate reads, making these methods not as effective with nanopore data. Various short read assemblers have been developed using this framework [55, 102, 151].

The third method uses string graphs [115], which involve constructing a graph describing read overlaps using a matching function, then removing inferrable edges from this graph using a process called transitive reduction. The result is a simplified graph where all paths in the graph have read-based support. FALCON, one of the first long-read assemblers for PacBio data uses this framework [27].

The first *de novo* assembly of the human genome was in 2001 as described above [84]. The process to generate this was incredibly complicated and expensive. With the advent of long reads, the process has become orders of magnitude less expensive and easier. In 2018, researchers at UCSC generated a human assembly using nanopore data using the Canu assembler [77, 70]. This assembly took 40 thousand CPU hours and was the first demonstration of a successful human assembly using nanopore data. Since then, many other nanopore-based assembly techniques have been developed which drastically reduce the time and cost of assembly [74, 141, 148].

2.2.4.1 Polishing

Most assemblers are focused on colocating sequencing data and generating a consensus sequence, and can often fail to produce sequence which is as consistently accurate as the reference sequence. Part of this is because the methodologies for grouping reads and generating sequence are different, and partly because the long read data most useful for grouping reads is not as accurate as second generation sequencing. The process of refining an assembled sequence is referred to as *polishing*. Generally, polishers involve a similar process to variant calling: reads are aligned to the assembly and pileup- or graph-based methodologies are used to identify differences and correct the assembled sequence. There are a wide range of tools which do this [137, 156, 148], but the most used is a short-read polishing toolkit called Pilon [167].

2.3 Phasing methodologies

Most of the work presented in my dissertation involves accurate phasing of sequence or variant data. When calling a variant against a linear reference first the genotype needs to be determined, where we decide if the variant is heterozygous (existing on one haplotype) or homozygous (existing on both haplotypes). Then we can perform phasing, where we assign the alleles of a heterozygous variant to each of the (generally) two haplotypes. Sometimes it is possible to divide data into two disparate sets, and sometimes it is possible to assign them specifically to the maternal or the paternal haplotype. In the first case, the regions where the two sets are consistently separated are called *phase sets* or *phase blocks*. The haplotypes, generally enumerated 1 and 2, are not necessarily consistent across phase sets. Understanding the phasing of a set of variants has applications in medicine, where two variants with a functional effect can produce different phenotypes in a sample if they are *in cis* (on the same haplotype) or *in trans* (on different haplotypes) [157]. This process can also be performed on sequence data, where each read can be assigned to a haplotype. This can improve the accuracy of downstream analyses.

Variant phasing is generally done in one of three ways. The first is using direct observation: a single sequencing read which describes two heterozygous loci is evidence that those loci are on the same haplotype. This could be from a paired-end Illumina read, from any of the long read sequencing technologies, or from other methods such as Hi-C [82, 51] or Strand-Seq [127].

There have been various methodological approaches to tackling this algorithmically while accounting for the potential of sequencing error. Many early tools using paired-end short read data would construct a fragment matrix relating reads and allele support, and then using minimum error correction (MEC) criterion to update a minimal number of sites to produce a perfect bipartition, including HapCUT [10] and MaxSAT [63]. With the advent of long read sequencing, this problem has become easier to solve as reads span more heterozygous sites. Tools including WhatsHap [124, 105] and HapCUT2 [39] continued to use MEC to solve this problem on long-read data.

Methods which do not rely on direct molecular observations generally use trios (child, mother, father) or pedigrees (trio including siblings) to determine haplotypes. These rely on principles of Mendelian segregation, where (absent the presence of *de novo* mutation) an allele found only in one of the parents must have originated on that haplotype. Various methods exist which perform this type of analysis [133, 134, 173].

Finally, there are also methods to haplotype variants using population inference, where the presence of certain variants in specific haplogroups can be used to identify co-inherited variation in a sample. These methods tend to be purely statistical and are not deterministic, and furthermore produce smaller phase sets than other methods [54].

Assigning haplotypes to sequencing data is a similar process to variant phasing, and has resulted in improvements in diploid genome assembly. The most common method of this involves trio-binning, where haplotype-informative k-mers are determined using short reads from a sample's parents and the presence of these k-mers in long reads is used to assign them to a maternal or paternal haplotype [76]. Recently, the diploid assembly method hifiasm has been used to produce remarkably accurate diploid assemblies using PacBio HiFi reads binned into haplotypes using short-read trio data [24] or Strand-Seq data [25].

2.3.1 Polyploid Phasing

Phasing in polyploid species (having more than two copies of each autosome) has significance in understanding plant evolution and agriculture improvement, as many commercially-farmed plant species (such as wheat, sugarcane, and potato) are polyploid [132]. When phasing diploid samples, evidence of linkage between alleles at heterozygous variant sites for one haplotype implies linkage between the remaining alleles for the other haplotype. This is not true for polyploid genomes where multiple haplotypes can exhibit the same allele at heterozygous sites. This adds computational complexity to any analysis and makes diploid-specific optimizations (which are necessary to solve NP-hard problems such as MEC) inapplicable.

Many of the algorithmic frameworks used in diploid phasing are difficult to apply to polyploid genomes. For example, when using MEC in regions where multiple haplotypes are similar, there is no penalty for merging reads from these similar haplotypes into a single partition. This can result in one large false haplotype (where a correct partition would have multiple haplotypes with similar sequence), and another false haplotype composed primarily of erroneous reads (where the reads actually originate from multiple different haplotypes) [144].

One of the first tools for solving polyploid phasing was HapCompass, which uses a graph-based framework to describe similarity between reads based on their agreement on alleles at variant sites, and removes weighted edges (reflective of read similarity) to produce spanning trees which correspond to haplotypes, an criterion called minimum weighted edge removal (MWER) [2, 3]. Another tool called HapTree takes an inductive approach by finding the most likely set of haplotypes on a small set of SNPs using a maximum-likelihood estimation framework, and then iteratively extending this set until all variant sites have been considered [14]. H-PoP is another successful tool which uses a Polyploid Balanced Optimal Partition (PBOP) method, where reads are partitioned into a set number of groups with minimal differences between reads in the same group and maximal differences between reads in different groups [175].

Recently, the ability to phase polyploid genomes has been introduced into the existing tool WhatsHap. WhatsHap Polyphase takes a two-step approach. First, reads are clustered together using a graph-based method with reads as nodes and weighted edges reflecting similarity between reads. The number of expected haplotypes does not contribute to the clustering step; similar regions across multiple haplotypes are generally merged together. Second, reads are threaded through these clusters to produce sequences for the expected number of haplotypes, using coverage statistics to identify regions where multiple haplotypes are locally identical. The tool will divide the results into phase blocks at regions of uncertainty using parameterized thresholds, enabling users to fine-tune the tradeoff between phasing accuracy and phase block length. WhatsHap Polyphase shows large improvements over the previous state-of-the-art method in switch and Hamming error rates, even when controlling for phase block size [144].

2.4 Sample Reference Materials

As much of the work in this document is method development, understanding the landscape of reference materials is important context. The bulk of this data has been generated by the Genome In A Bottle Consortium (GIAB). This consortium has endeavored to produce publicly-available sequence data and platinum-grade variant calls for benchmarking and analysis [178, 177, 166]. These efforts include the generation of multiple types of sequencing data, variant calling with multiple technologies, collation of results across data and methods, and determination of high-quality variants accompanied by regions of certainty and uncertainty. Without this data it would be impossible to accurately compare methods for human genome inference.

The samples documented by these efforts come from three child-father-mother trios, all which have given permission for their cell lines to be immortalized and preserved for future sequencing. They are referred to by an ancestral designation, NIST ID, and Coriell lines and are: the CEPH Mother/Daughter, HG001, or NA12878 (and her parents, spouse, spouse's parents, and children); the Ashkenazim Son, HG002, or GM24385 (and his parents HG003 and HG004); the Chinese Son, HG005, or GM24631 (and his parents HG006 and HG007). All of work presented here has been evaluated on at least one of these cell lines.

Part II

Demonstrating the Utility of Read Haplotyping

Chapter 3

Demonstration of the applicability of HMMs to phasing reads, and the utility of phased reads in variant calling

3.1 Preamble

What follows is the full text from my first manuscript "Haplotype-aware genotyping from noisy long reads" [37] published in Genome Biology in which I share first-authorship with Jana Ebler and Marina Haukness. In this paper we present two implementations of the same high-level algorithm and demonstrate that using phasing information improves variant calling for long read sequencing.

For this publication, I worked with Marina to develop, test, and evaluate MarginPhase. I helped write the text of the paper, produce figures, and performed the coverage analysis for Figure 3.6.

While WhatsHap (the implementation our collaborators developed) has become an industry standard for phasing since publication, our implementation of MarginPhase was largely an experimental proof of concept at this point. Neither tool has ever been used as a variant caller, although both have now been successfully used in variant calling pipelines. Much of what what I learned during this chapter's work about handling nanopore data was instrumental in the development of MarginPolish (presented in Chapter 4), and the foundation laid here enabled all the work presented in Chapter 5.

3.2 Abstract

Motivation: Current genotyping approaches for single nucleotide variations (SNVs) rely on short, relatively accurate reads from second generation sequencing devices. Presently, third generation sequencing platforms able to generate much longer reads are becoming more widespread. These platforms come with the significant drawback of higher sequencing error rates, which makes them ill-suited to current genotyping algorithms. However, the longer reads make more of the genome unambiguously mappable and typically provide linkage information between neighboring variants.

Results: In this paper we introduce a novel approach for haplotype-aware genotyping from noisy long reads. We do this by considering bipartitions of the sequencing reads, corresponding to the two haplotypes. We formalize the computational problem in terms of a Hidden Markov Model and compute posterior genotype probabilities using

the forward-backward algorithm. Genotype predictions can then be made by picking the most likely genotype at each site. Our experiments indicate that longer reads allow significantly more of the genome to potentially be accurately genotyped. Further, we are able to use both Oxford Nanopore and Pacific Biosciences sequencing data to independently validate millions of variants previously identified by short-read technologies in the reference NA12878 sample, including hundreds of thousands of variants that were not previously included in the high-confidence reference set.

3.3 Introduction

Reference-based genetic variant identification comprises two related processes: genotyping and phasing. Genotyping is the process of determining which genetic variants are present in an individual's genome. A genotype at a given site describes whether both chromosomal copies carry a variant allele, only one of them carries it, or whether the variant allele is not present at all. Phasing refers to determining an individual's haplotypes, which consist of variants that lie near each other on the same chromosome and are inherited together. To completely describe all of the genetic variation in an organism, both genotyping and phasing are needed. Together, the two processes are called *diplotyping*.

Many existing variant analysis pipelines are designed for short DNA sequencing reads [162, 1]. Though short reads are very accurate at a per-base level, they can suffer from being difficult to unambiguously align to the genome, especially in repetitive or duplicated regions [96]. The result is that millions of bases of the reference human genome are not currently reliably genotyped by short reads, primarily in multi-megabase gaps near the centromeres and short arms of chromosomes [5]. While short reads are unable to uniquely map to these regions, long reads can potentially span into or even across them. This makes it so long reads are advantageous over short reads for tasks such as haplotyping, large structural variant detection, and *de novo* assembly [128, 21, 70, 147]. Here, we attempt to demonstrate the utility of long reads for more comprehensive genotyping.

Long read DNA sequencing technologies are rapidly falling in price and increasing in general availability. Such technologies include Single Molecule Real Time (SMRT) Sequencing by Pacific Biosciences (PacBio) and nanopore sequencing by Oxford Nanopore Technologies (ONT). However, due to the historically greater relative cost and higher sequencing error rates of these technologies, little attention has been given thus far to the problem of genotyping single nucleotide variants (SNVs) with long reads. Recently, [60] have taken first steps in this direction, but their approach does not scale to process whole human genomes in reasonable time.

For an illustration of the benefit of using long reads to diplotype, consider Figure 3.1. Shown are three SNV positions covered by long reads. The gray sequences represent the true haplotype sequences and reads are colored in blue and red. The colors correspond to the haplotype which the respective read stems from: the red ones from the upper sequence, and the blue ones from the lower one. Since sequencing errors

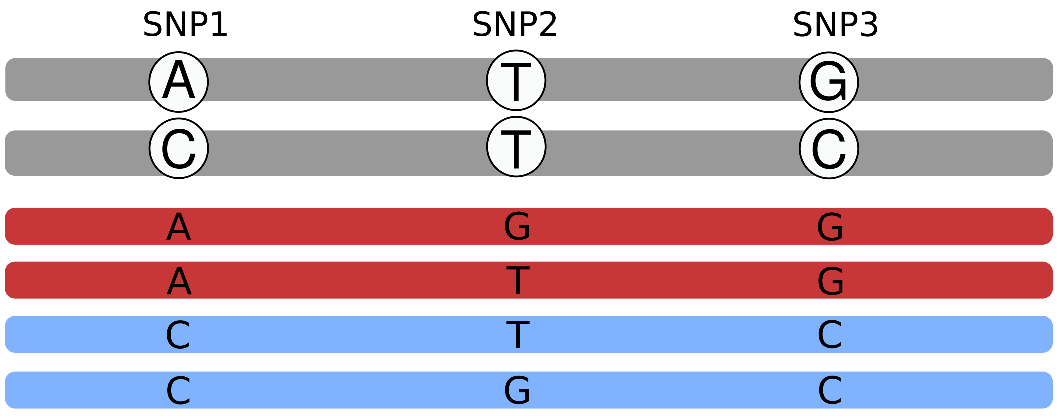


Figure 3.1: **Motivation.** Gray sequences illustrate the haplotypes; the reads are shown in red and blue. The red reads originate from the upper haplotype, the blue ones from the lower. Genotyping each SNV individually would lead to the conclusion that all of them are heterozygous. Using the haplotype context reveals uncertainty about the genotype of the second SNV.

can occur, the alleles supported by the reads are not always equal to the true ones in the haplotypes shown in gray. Considering the SNVs individually, we would probably genotype the first one as A/C, the second one as T/G and the third one as G/C, since the number of reads supporting each allele are the same. This leads to a wrong genotype prediction for the second SNV. However, if we knew which haplotype each read stems from, that is, if we knew their colors, then we would be unsure about the genotype of the second SNV. It could also be G/G or T/T, since the reads stemming from the same haplotypes must support the same alleles. Therefore, using haplotype information during genotyping makes it possible to compute more reliable genotype predictions and to detect uncertainties.

Contributions. In this paper, we show that for contemporary long read technologies, read-based phase inference can be simultaneously combined with the genotyping process for SNVs to produce accurate diplotypes and to detect variants in regions not mappable by short reads. We show that key to this inference is the detection of linkage relationships between heterozygous sites within the reads. To do this, we describe a novel algorithm to accurately predict diplotypes from noisy long reads that scales to deeply sequenced human genomes. We achieve this by considering bipartitions of all given sequencing reads, corresponding to the two haplotypes of an individual. The problem is formalized using a Hidden Markov Model (HMM) from which we compute genotype likelihoods using the forward-backward algorithm and make genotype predictions by determining the likeliest genotype at each position.

We then apply this algorithm to diplotype one individual from the 1000 Genomes Project, NA12878, using long reads from both PacBio and ONT. NA12878 has been extensively sequenced and studied, and the Genome in a Bottle consortium has published sets of highly confident variant calls [176]. We demonstrate that our method is accurate, that it can be used to confirm variants in regions of uncertainty, and that it allows for the discovery of variants in regions which are unmappable using short DNA read sequencing technologies.

3.4 Methods

We describe a probabilistic model for diplotype and genotype inference, and in this paper use it to find maximum posterior probability genotypes. The approach builds upon the WhatsHap approach [125], but incorporates a full probabilistic allele inference model into the problem. It has similarities to that proposed by [83], but we here frame the problem using Hidden Markov Models (HMMs).

3.4.1 Alignment Matrix

Let $\mathbf{M}$ be an alignment matrix whose rows represent sequencing *reads* and whose columns represent genetic *sites*. Let m be the number of rows, let n be the number of columns, and let $\mathbf{M}_{i,j}$ be the jth element in the ith row. In each column let $\Sigma_j \subset \Sigma$ represent the set of possible *alleles* such that $\mathbf{M}_{i,j} \in \Sigma_j \cup \{-\}$, the "$-$" gap symbol representing a site at which the read provides no information. We assume no row or column is composed only of gap symbols, an uninteresting edge case. An example alignment matrix is shown in Figure 3.2. Throughout the following we will be informal and refer to a row i or column j, being clear from the context whether we are referring to the row or column itself or the coordinate.

$$\begin{array}{cccccc}
 & 1 & 2 & 3 & 4 & 5 \\
1 & A & G & T & - & - \\
2 & A & G & T & - & - \\
3 & - & C & - & G & - \\
4 & - & C & T & G & - \\
5 & - & - & T & C & T \\
6 & - & - & T & C & T
\end{array}$$

Figure 3.2: **Alignment Matrix.** Here, the alphabet of possible alleles is the set of DNA nucleotides, i.e. $\Sigma = \{A, C, G, T\}$

3.4.2 Genotype Inference Problem Overview

A diplotype $H = (H^1, H^2)$ is a pair of haplotype (segments); a *haplotype (segment)* $H^k = H^k_1, H^k_2, \ldots, H^k_n$ is a sequence of length n whose elements represents alleles such that $H^k_j \in \Sigma_j$. Let $B = (B^1, B^2)$ be a bipartition of the rows of $\mathbf{M}$ into two parts (sets): B^1, the first part, and B^2, the second part. We use bipartitions to represent which haplotypes the reads came from, of the two in a genome. By convention we assume that the first part of B are the reads arising from H^1 and the second part of B are the reads arising from H^2. The problem we analyze is based upon a probabilistic model that essentially represents the (Weighted) Minimum Error Correction (MEC) problem [29, 59], while modeling the evolutionary relationship between the two haplotypes and so imposing a cost on bipartitions that create differences between the inferred haplotypes.

For a bipartition B, and making an i.i.d. assumption between sites in the reads:

$$P(H|B,\mathbf{M}) = \prod_{j=1}^{n} \sum_{Z_j \in \Sigma_j} P(H_j^1|B^1,Z_j)P(H_j^2|B^2,Z_j)P(Z_j)$$

Here $P(Z_j)$ is the prior probability of the ancestral allele Z_j of the two haplotypes at column j, by default we can use a simple flat distribution over ancestral alleles (but see below). The posterior probability $P(H_j^k|B^k,Z_j) =$

$$\frac{P(H_j^k|Z_j)\prod_{\{i \in B^k : \mathbf{M}_{i,j} \neq -\}} P(\mathbf{M}_{i,j}|H_j^k)}{\sum_{Y_j \in \Sigma_j} P(Y_j|Z_j)\prod_{\{i \in B^k : \mathbf{M}_{i,j} \neq -\}} P(\mathbf{M}_{i,j}|Y_j)}$$

for $k \in \{1,2\}$, where the probability $P(H_j^k|Z_j)$ is the probability of the haplotype allele H_j^k given the ancestral allele Z_j. For this we can use a continuous time Markov model for allele substitutions, such as Jukes-Cantor [20], or some more sophisticated model that factors the similarities between alleles (see below). Similarly, $P(\mathbf{M}_{i,j}|H_j^k)$ is the probability of observing allele $\mathbf{M}_{i,j}$ in a read given the haplotype allele H_j^k.

The genotype inference problem we consider is finding for each site:

$$\arg\max_{(H_j^1,H_j^2)} P(H_j^1,H_j^2|\mathbf{M}) = \arg\max_{(H_j^1,H_j^2)} \sum_{B} P(H_j^1,H_j^2|B,\mathbf{M})$$

i.e. finding the genotype (H_j^1,H_j^2) with maximum posterior probability for a generative model of the reads embedded in $\mathbf{M}$.

3.4.3 A Graphical Representation Of Read Partitions

For a column j in $\mathbf{M}$, a row i is *active* if the first non-gap symbol in row i occurs at or before column j and the last non-gap symbol in row i occurs at or after column j. Let A_j be the set of active rows of column j. For a column j a row i is *terminal* if its last non-gap symbol occurs at column j or $j = n$. Let A'_j be the set of active, non-terminal rows of column j.

Let $B_j = (B_j^1, B_j^2)$ be a bipartition of A_j into a first part B_j^1 and a second part B_j^2. Let $\mathbf{B_j}$ be the set of all possible such bipartitions of the active rows of j. Similarly, let $C_j = (C_j^1, C_j^2)$ be a bipartition of A'_j, and $\mathbf{C_j}$ be the set of all possible such bipartitions of the active, non-terminal rows of j.

For two bipartitions $B = (B^1, B^2)$ and $C = (C^1, C^2)$, B is *compatible* with C if the subset of B^1 in $C^1 \cup C^2$ is a subset of C^1, and, similarly, the subset of B^2 in $C^1 \cup C^2$ is a subset of C^2. Note this definition is symmetric and reflexive, although not transitive.

Let $G = (V_G, E_G)$ be a directed graph. The vertices V_G are the set of bipartitions of both the active rows and the active, non-terminal rows for all columns of $\mathbf{M}$ and a special *start* and *end* vertex, i.e. $V_G = \{start, end\} \cup (\bigcup_j \mathbf{B_j} \cup \mathbf{C_j})$. The edges E_G are a subset of compatibility relationships, such that (1) for all j there is an edge $(B_j \in \mathbf{B_j}, C_i \in \mathbf{C_j})$ if B_j is compatible with C_j, (2) for all $0 < j < n$ there is an edge $(C_j \in \mathbf{C_j}, B_{j+1} \in \mathbf{B_{j+1}})$ if C_j is compatible with B_{j+1}, (3) there is an edge from the start vertex to each member of $\mathbf{B_1}$, and (4) there is an edge from each member of $\mathbf{B_n}$ to the end vertex (Note that $\mathbf{C_n}$ is empty and so contributes no vertices to G). Figure 3.3 shows an example graph.

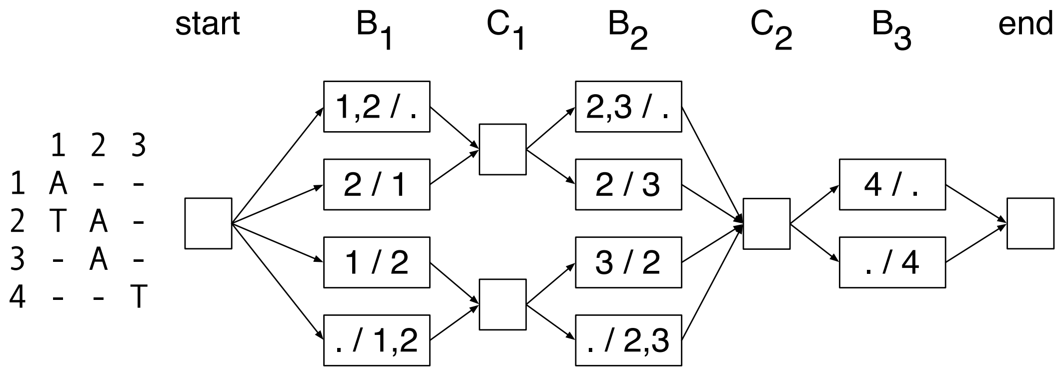


Figure 3.3: **Example Graph.** Left: An alignment matrix. Right: The corresponding directed graph representing the bipartitions of active rows and active non-terminal rows, where the labels of the nodes indicate the partitions, e.g. '1,2 / .' is shorthand for $A = (\{1,2\},\{\})$.

The graph G has a large degree of symmetry and the following properties are easily verified:

- For all j and all $B_j \in \mathbf{B_j}$, the indegree and outdegree of B_j is 1.
- For all j the indegree of all members of $\mathbf{C_j}$ is equal.
- Similarly, for all j the outdegree of all members of $\mathbf{C_j}$ is equal.

Let the *maximum coverage*, denoted *maxCov*, be the maximum cardinality of a set A_j over all j. By definition, $maxCov \leq m$. Using the above properties it is easily verified that: (1) the cardinality of G (number of vertices) is bounded by this maximum coverage, being less than or equal to $2+(2n-1)2^{maxCov}$, and (2) the size of G (number of edges) is at most $2n2^{maxCov}$.

Let a directed path from the start vertex to the end vertex be called a *diploid path*, $D = (D_1 = start, D_2, \ldots, D_{2n+1} = end)$. The graph is naturally organized by the columns of $\mathbf{M}$, so that $D_{2j} = (B_j^1, B_j^2) \in \mathbf{B_j}$ and $D_{2j+1} = (C_{j+1}^1, C_{j+1}^2) \in \mathbf{C_j}$ for all $0 < j \leq n$. Let $B_D = (B_D^1, B_D^2)$ denote a pair of sets, where B_D^1 is the union of the first parts of the vertices of $D_2, \ldots, D_{2n+1}$ and, similarly, B_D^2 is the union of second parts of the vertices of $D_2, \ldots, D_{2n+1}$.

B_D^1 and B_D^2 are disjoint because otherwise there must exist a pair of vertices within D that are incompatible, which is easily verified to be impossible. Further, because D visits a vertex for every column of $\mathbf{M}$, it follows that the sum of the cardinalities of these two sets is m. B_D is therefore a bipartition of the rows of $\mathbf{M}$ which we call a *diploid path bipartition.*

Lemma 3.4.1. *The set of diploid path bipartitions is the set of bipartitions of the rows of* $\mathbf{M}$ *and each diploid path defines a unique diploid path bipartition.*

Proof. We first prove that each diploid path defines a unique bipartition of the rows of $\mathbf{M}$. For each column j of $\mathbf{M}$, each vertex $B_j \in \mathbf{B_j}$ is a different bipartition of the same set of active rows. B_j is by definition compatible with a diploid path bipartition of a diploid path that contains it, and incompatible with every other member of $\mathbf{B_j}$. It follows that for each column j two diploid paths with the same diploid path bipartition must visit the same node in $\mathbf{B_j}$, and, by identical logic, the same node in $\mathbf{C_j}$, but then two such diploid paths are therefore equal.

There are 2^m partitions of the rows of $\mathbf{M}$. It remains to prove that there are 2^m diploid paths. By the structure of the graph, the set of diploid paths can be enumerated backwards by traversing right-to-left from the end vertex by depth-first search and exploring each incoming edge for all encountered nodes. As stated previously, the only vertices with indegree greater than one are for all j the members of $\mathbf{C_j}$, and each member of $\mathbf{C_j}$ has the same indegree. For all j the indegree of C_j is clearly $2^{|C_j|-|B_j|}$: two to the power of the number of number of active, terminal rows at column j. The number of possible paths must therefore be $\prod_{j=1}^{n} 2^{|C_j|-|B_j|}$. As each row is active and terminal in exactly one column, we obtain $m = \sum_j |C_j| - |B_j|$ and therefore:

$$2^m = \prod_{j=1}^{n} 2^{|C_j|-|B_j|}$$

. $\square$

3.4.4 A Hidden Markov Model For Genotype and Diplotype Inference

In order to infer diplotypes, we define a Hidden Markov Model which is based on G, but additionally represents all possible genotypes at each genomic site (i.e. in each B column). To this end, we define the set of states $\mathbf{B_j} \times \Sigma_j \times \Sigma_j$, which contains a state for each bipartition of the active rows at position j and all possible assignments of alleles in Σ_j to the two partitions. Additionally, the HMM contains a hidden state for each bipartition in $\mathbf{C_j}$, exactly as defined for G above. Transitions between states are defined by the compatibility relationships of the corresponding bipartitions as before. This HMM construction is illustrated in Figure 3.4.

For all j and all $C_j \in \mathbf{C_j}$ each outgoing edge has transition probability $P(a_1, a_2) =$

$\sum_{Z_j} P(a_1|Z_j)P(a_2|Z_j)P(Z_j)$, where $(B_j, a_1, a_2) \in \mathbf{B_j} \times \Sigma_j \times \Sigma_j$ is the state being transitioned to. Similarly, each outgoing edge of the start node has transition probability $P(a_1, a_2)$. The outdegree of all remaining nodes is 1, so these edges have transition probability 1.

The start node, the end node, and members of $\mathbf{C_j}$ for all j are silent states, and hence do not emit symbols. For all j, members of $\mathbf{B_j} \times \Sigma_j \times \Sigma_j$ output the entries in the j-th column of $\mathbf{M}$ that are different from "–". We assume every matrix entry to be associated with an error probability, which we can compute from $P(\mathbf{M}_{ij}|H_j^k)$ defined previously. Based on this, the probability of observing a specific output column of $\mathbf{M}$ can be easily calculated.

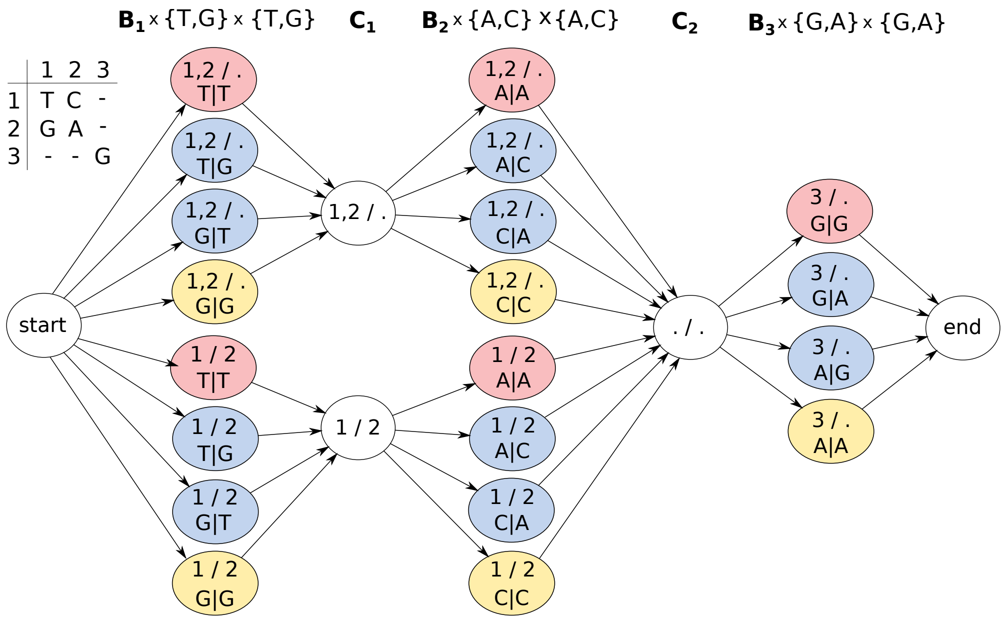


Figure 3.4: **Genotyping HMM.** Colored states correspond to bipartitions of reads and allele assignments at that position. States in C_1 and C_2 correspond to bipartitions of reads covering positions 1 and 2 or 2 and 3, respectively. In order to compute genotype likelihoods after running the forward-backward algorithm, states of the same color have to be summed up in each column.

3.4.4.1 Computing Genotype Likelihoods

The goal is to compute genotype likelihoods for the possible genotypes for each variant position using the HMM defined above. Performing the forward-backward algorithm returns forward and backward probabilities of all hidden states. Using those, the posterior distribution of a state $(B, a_1, a_2) \in \mathbf{B_j} \times \Sigma_j \times \Sigma_j$, corresponding to bipartition

B and assigned alleles a_1 and a_2, can be computed as

$$P((B,a_1,a_2)|\mathbf{M}) = \frac{\alpha_j(B,a_1,a_2)\cdot\beta_j(B,a_1,a_2)}{\sum\limits_{B'\in\mathcal{B}(A_j)}\sum\limits_{a'_1,a'_2\in\Sigma_j}\alpha_j(B',a'_1,a'_2)\cdot\beta_j(B',a'_1,a'_2)} \tag{3.1}$$

where $\alpha_j(B,a_1,a_2)$ and $\beta_j(B,a_1,a_2)$ denote forward and backward probabilities of the state (B,a_1,a_2) and $\mathcal{B}(A_j)$, the set of all bipartitions of A_j. The above term represents the probability for a bipartition $B = (B^1,B^2)$ of the reads in A_j and alleles a_1 and a_2 assigned to these partitions. In order to finally compute the likelihood for a certain genotype, one can marginalize over all bipartitions of a column, and all allele assignments corresponding to that genotype.

Example 3.4.1. *In order to compute genotype likelihoods for each column of the alignment matrix, posterior state probabilities corresponding to states of the same color in Figure 3.4 need to be summed up. For the first column, adding up the red probabilities gives the genotype likelihood of genotype T/T, blue of genotype G/T and yellow of G/G.*

3.4.5 Implementations

We created two independent software implementations of this model, one based upon WhatsHap and one from scratch, which we call MarginPhase. Each uses different optimizations and heuristics that we briefly describe.

3.4.5.1 WhatsHap Implementation

We extended the implementation of WhatsHap ([125], `bitbucket.org/whatshap/whatshap`) to enable haplotype aware genotyping of bi-allelic variants based on the above model. WhatsHap focuses on re-genotyping variants, i.e. it assumes SNV positions to be given. In order to detect variants, a simple SNV calling pipeline was developed. It is based on `samtools mpileup` [95] which provides information about the bases supported by each read covering a genomic position. A set of SNV candidates is generated by selecting genomic positions at which the frequency of a non-reference allele is above a fixed threshold (0.25 for PacBio data, 0.4 for Nanopore data) and the absolute number of reads supporting the non-reference allele is at least 3.

Allele Detection. In order to construct the alignment matrix, a crucial step is to determine whether each read supports the reference or the alternative allele at each of n given genomic positions. In WhatsHap, this is done based on re-aligning sections of the reads [105]. Given an existing read alignment from the provided BAM file, its sequence in a window around the variant is extracted. It is aligned to the corresponding region of the reference sequence and additionally, to the alternative sequence, which is artificially produced by inserting the alternative allele into the reference. The alignment cost is computed by using affine gap costs. Phred scores representing the probabilities for opening and extending a gap and for a mismatch in the alignment can be estimated from the given BAM file. The allele leading to a lower alignment cost is assumed to be supported by the read and is reported in the alignment matrix. If both alleles lead to the same cost, the corresponding matrix entry is "–". The absolute difference of both alignment scores is assigned as a weight to the corresponding entry in the alignment matrix. It can be interpreted as a phred scaled probability for the allele being wrong and is utilized for the computation of output probabilities.

Read Selection. Our algorithm enumerates all bipartitions of reads covering a variant position and thus has a runtime exponential in the maximum coverage of the data. To ensure that this quantity is bounded, the same read selection step implemented previously in the WhatsHap software is run before constructing the HMM and computing genotype likelihoods. Briefly, a heuristic approach described in [48] is applied, which selects phase informative reads iteratively taking into account the number of heterozygous variants covered by the read and its quality.

Transitions. Defining separate states for each allele assignment in $\mathbf{B_j}$ enables easy incorporation of prior genotype likelihoods by weighting transitions between states in $\mathbf{C_{j-1}}$ and $\mathbf{B_j} \times \Sigma_j \times \Sigma_j$. Since there are two states corresponding to a heterozygous genotype in the bi-allelic case (0|1 and 1|0), the prior probability for the heterozygous genotype is equally spread between these states.

In order to compute such genotype priors, the same likelihood function underlying the approaches described in [64] and [38] was utilized. For each SNV position, the model computes a likelihood for each SNV to be absent, heterozygous, or homozygous based on all reads that cover a particular site. Each read contributes a probability term to the likelihood function, which is computed based on whether it supports the reference or the alternative allele [64]. Furthermore, the approach accounts for statistical uncertainties arising from read mapping and has a runtime linear in the number of variants to be genotyped [38]. Prior genotype likelihoods are computed before read selection. In this way, information of all input reads covering a position can be incorporated.

3.4.5.2 MarginPhase Implementation

MarginPhase (`github.com/benedictpaten/marginPhase`) is an experimental, open source implementation of the described HMM written in C. It differs from the WhatsHap implementation in the method it uses to explore bipartitions and the method to generate allele support probabilities from the reads.

Read Bipartitions. The described HMM scales exponentially in terms of increasing read coverage. For typical 20-60x sequencing coverage (i.e. average number of active rows per column) it is impractical to store all possible bipartitions of the rows of the matrix. MarginPhase implements a simple, greedy pruning and merging heuristic outlined in recursive pseudocode as follows:

The procedure computePrunedHMM takes an alignment matrix and returns a connected subgraph of the HMM for **M** that can be used for inference, choosing to divide the input alignment matrix into two if the number of rows exceeds a threshold t, recursively.

The sub-procedure mergeHMMs takes two pruned HMMs for two disjoint alignment matrices with the same number of columns and joins them together in the natural

```
procedure computePrunedHMM($\mathbf{M}$)
    if maxCov $\geq t$ then
        Divide $\mathbf{M}$ in half to create two matrices, $\mathbf{M_1}$ and $\mathbf{M_2}$, such
            that $\mathbf{M_1}$ is the first $\frac{n}{2}$ rows of $\mathbf{M}$ and $\mathbf{M_2}$ is the remaining
            rows of $\mathbf{M}$.
        $\mathbf{HMM_1} \leftarrow$ computePrunedHMM($\mathbf{M_1}$)
        $\mathbf{HMM_2} \leftarrow$ computePrunedHMM($\mathbf{M_2}$)
        $\mathbf{HMM} \leftarrow$ mergeHMMs($\mathbf{HMM_1}$,$\mathbf{HMM_2}$)
    else
        Let $\mathbf{HMM}$ be the read partitioning HMM for $\mathbf{M}$.
    return subgraph of $\mathbf{HMM}$ including visited states and transitions
        each with posterior probability of being visited $\geq v$, and which
        are on a path from the start to end nodes.
```

way such that if at each site i there are $|\mathbf{B_i^1}|$ states in $\mathbf{HMM_1}$ and $|\mathbf{B_i^2}|$ in $\mathbf{HMM_2}$ then the resulting HMM will have $|\mathbf{B_i^1}| \times |\mathbf{B_i^2}|$ states. This is illustrated in Figure 3.5. In the experiments used here $t = 8$ and $v = 0.01$.

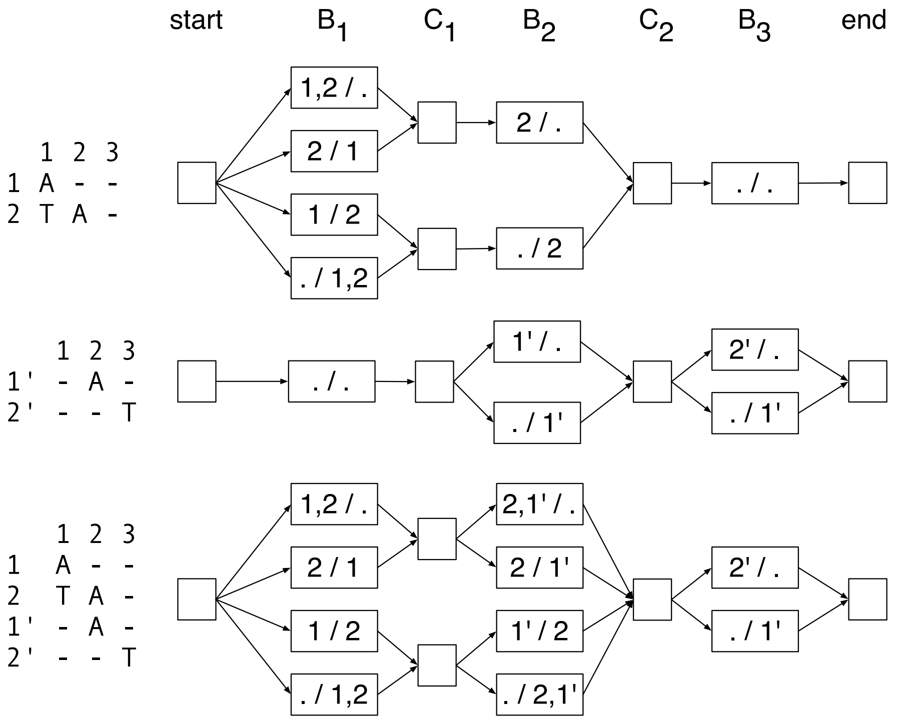


Figure 3.5: The merger of two read partitioning HMMs with the same number of columns. Top and middle: Two HMMs to be merged; bottom: the merged HMM. Transition and emission probabilities not shown.

Allele Supports. In MarginPhase, the alignment matrix has a site for each base in the reference genome. To generate the allele support from the reads, for each read we calculate the posterior probability of each allele using the implementation of the banded forward-backward pairwise alignment described in [69]. The result is that for each reference base, for each read that overlaps (according to an initial guide alignment extracted from the SAM/BAM file) the reference base we calculate the probability of each possible nucleotide (i.e. { ‘A’, ‘C’, ‘G’, ‘T’ }). Gaps are ignored and treated as missing data. This approach allows summation over all alignments within the band.

3.5 Results

3.5.1 Data Preparation and Evaluation

To test our methods, we used sequencing data for NA12878 from two different long read sequencing technologies. NA12878 is a participant from the 1000 Genomes Project [1] who has been extensively sequenced and analyzed. We used Oxford Nanopore reads from [70] and PacBio reads from [177]. Both sets of reads were aligned to GRCh38 with `minimap2`, a mapper designed to align error-prone long reads [91].

To ensure that any variants we found were not artifacts of misalignment, we filtered out reads flagged as secondary or supplementary, as well as reads with a mapping quality score less than 30. Genome-wide, this left approximately 12 million Nanopore reads and 34 million PacBio reads. The Nanopore reads had a median depth of 37× and length of 5950, including a set of ultra-long reads with lengths up to 900 kilobases. The PacBio reads had a median depth of 46× and length of 2650.

To validate the performance of our methods, we used callsets from Genome

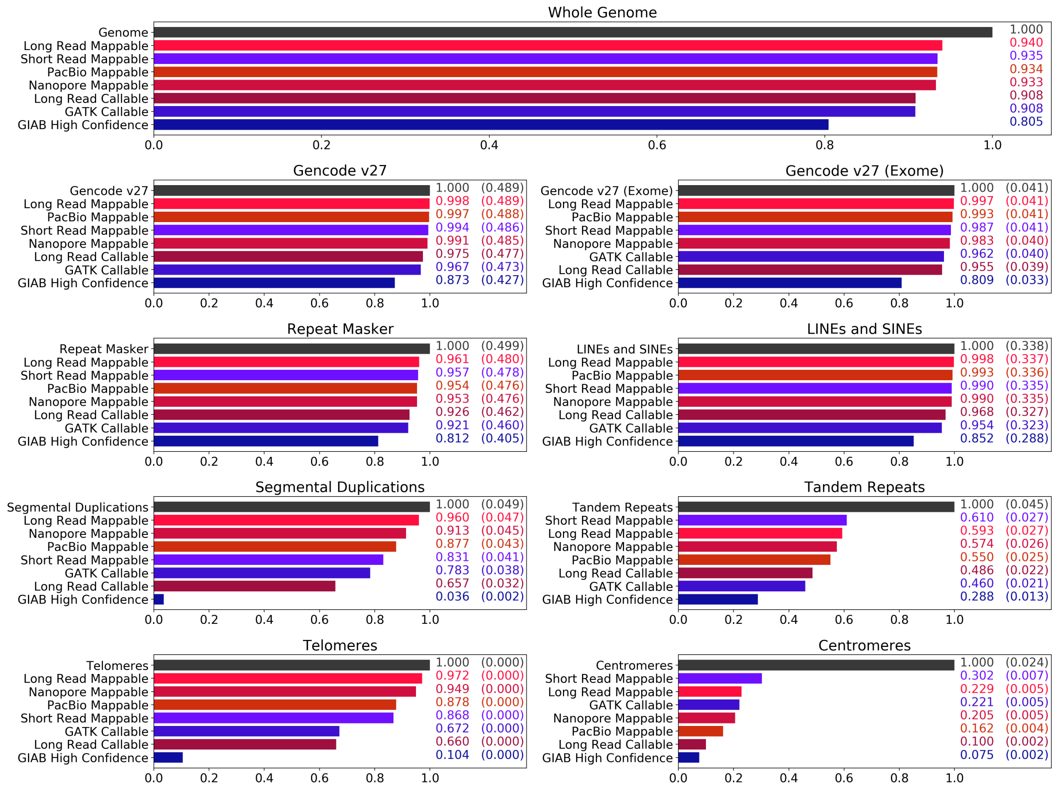


Figure 3.6: **Reach of short read and long read technologies.** The callable and mappable regions for NA12878 spanning various repetitive or duplicated sequences on GRCh38 are shown. Feature locations are determined based on BED tracks downloaded from the UCSC Genome Browser [73]. Other than the Gencode regions [62, 138], all features are subsets of the Repeat Masker [153] track. Four coverage statistics for long reads (shades of red) and three for short reads (shades of blue) are shown. The labels 'PacBio Mappable' and 'Nanopore Mappable' describe areas where at least one primary read with GQ ≥ 30 has mapped, and 'Long Read Mappable' describes where this is true for at least one of the long read technologies. 'Long Read Callable' describes areas where both read technologies have coverage of at least 10 and less than twice the median coverage. 'GIAB High Confidence', 'GATK Callable' and 'Short Read Mappable' are the regions associated with the evaluation callsets. For the feature-specific plots, the numbers on the right detail coverage over the feature and coverage over the whole genome (parenthesized).

in a Bottle's (GIAB) benchmark small variant calls v3.3.2 [176]. First, we compared against GIAB's set of high confidence calls, generated by a consensus algorithm spanning multiple sequencing technologies and variant calling programs. The high confidence regions associated with this callset exclude structural variants, centromeres, and heterochromatin. We used this to show our method's accuracy in well-understood and easy-to-map regions of the genome.

We also analyzed our results compared to two callsets which were used in the construction of GIAB's high confidence variants, one made by GATK HaplotypeCaller v3.5 (GATK/HC, [162]) and the other by Freebayes 0.9.20 [53], both generated from a 300× PCR-free Illumina sequencing run [176].

All of our evaluation statistics were generated with the tool `vcfeval` from Real Time Genomics [30]. We restrict the analysis to SNVs due to the error distribution of both PacBio and Nanopore long reads which leads to insertions and deletions being the most common type of sequencing error by far [79, 119].

Short read variant callers. We explored the suitability of current state-of-the-art callers for short reads to process long read data (using default settings), but were unsuccessful. The absence of base qualities in the PacBio data prevented any calling; for Nanopore data, FreeBayes was prohibitively slow and neither Platypus nor GATK/HC produced calls.

3.5.2 Long Read Coverage

We determined the regions where long and short reads can be mapped to the human genome. In Figure 3.6, various coverage metrics for short and long reads are plotted against different genomic features, which were mostly selected for being repetitive or duplicated.

The callsets on the Illumina data made by GATK/HC and FreeBayes come with two BED files describing where calls were made with some confidence. The first, described in Figure 3.6 as *Short Read Mappable*, was generated using GATK CallableLoci v3.5 and includes regions where there is a) at least a read depth of 20, and b) at most a depth of twice the median depth, only including reads with mapping quality of at least 20. This definition of callable only considers read mappings. The second, described as *GATK Callable*, was generated from the GVCF output from GATK/HC by excluding areas with genotype quality less than 60. This is a more sophisticated definition of callable as it reflects the effects of homopolymers and tandem repeats. We use these two BED files in our analysis of how short and long reads map differently in various areas of the genome.

For long reads, we show four coverage statistics. The records marked as "Mappable" describe areas where there is at least one high quality long read mapping (PacBio, Nanopore, and *Long Read Mappable* for areas where at least one of the technologies mapped). The *Long Read Callable* entries cover a conservative region which has a sufficient read depth to illustrate the efficacy of our method; it covers regions where both sequencing technologies had a minimum depth of ten and maximum of 2× the median depth (similar to the CallableLoci metric).

Figure 3.6 shows that in almost all cases, long reads map to more area than is mappable by short reads. For example, nearly half a percent of the genome is mappable by long reads but not short reads. Long reads also map to one percent more of the exome, and thirteen percent more of segmental duplications. Centromeres and Tandem Repeats are outliers to this generalization, where neither PacBio nor Nanopore cover appreciably more than Illumina.

3.5.3 Comparison Against High Confidence Truthset

To validate our method, we first analyzed the SNV detection and genotyping performance of our algorithm using the GIAB high confidence callset as a benchmark. All variants reported in these statistics fall within the GIAB high confidence regions.

Figure 3.7 (top) shows precision and recall of our algorithms on both the PacBio and Oxford Nanopore data sets. MarginPhase and WhatsHap perform similarly overall. MarginPhase achieved higher precision and recall on Nanopore reads, with precision of 0.7686 and recall of 0.8089, compared to WhatsHap's precision of 0.7131 and recall of 0.7248 on the same set of Nanopore reads. WhatsHap obtained better results on PacBio data, with a precision of 0.9738 and recall of 0.9593, compared to MarginPhase's precision of 0.9497 and recall of 0.9147.

In addition to considering the two methods individually, we examine a combined set of variants which occur in both the calls made by WhatsHap on the PacBio reads and MarginPhase on the Nanopore data and where both tools report the same genotype. This improves the precision to 0.9969 at a recall of 0.7859. In further analysis, we refer to this combined variant set as *Long Read Variants*. It reflects a high precision subset of long read variants, validated independently by both sequencing technologies.

In order to further analyze the quality of the genotype predictions of our methods, we computed the genotype concordance of our callsets with respect to the GIAB ground truth inside of the high confidence regions. This was done by considering all variant positions correctly identified by MarginPhase and WhatsHap, and finding what fraction of these were also correctly genotyped (homozygous or heterozygous) with respect to the truth set. Figure 3.7 (bottom) shows the results. On the PacBio data, WhatsHap genotypes 99.78% of the variants contained in the truth set correctly, and MarginPhase genotypes 96.59% correctly. On the Nanopore data, MarginPhase performs slightly better by genotyping 98.02% of the SNVs contained in the GIAB callset correctly, while WhatsHap computed correct genotypes for 97.42% of the variants overlapping the GIAB truth set. Considering the intersection of the WhatsHap calls on PacBio, and MarginPhase calls on Nanopore data (i.e. our *Long Read Variants* set), we obtain a genotype concordance of 99.98%.

3.5.4 Cutting and Downsampling Reads

Our genotyping model incorporates haplotype information into the genotyping process by using the property that long sequencing reads can cover multiple variant positions. Therefore, one would expect the genotyping results to improve as the length

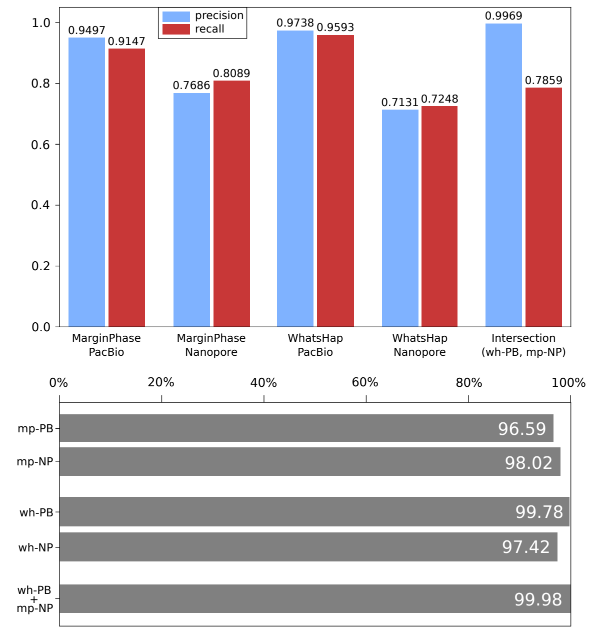


Figure 3.7: **Precision and Recall (Top)** of MarginPhase and WhatsHap on PacBio and Nanopore data sets in GIAB high confidence regions. **Genotype Concordance (Bottom)** (wrt. GIAB high confidence calls) of MarginPhase (mp, top) and WhatsHap (wh, middle) callsets on PacBio (PB) and Nanopore (NP) data. Furthermore, genotype concordance for the intersection of the calls made by WhatsHap on the PacBio and MarginPhase on the Nanopore reads is shown (bottom).

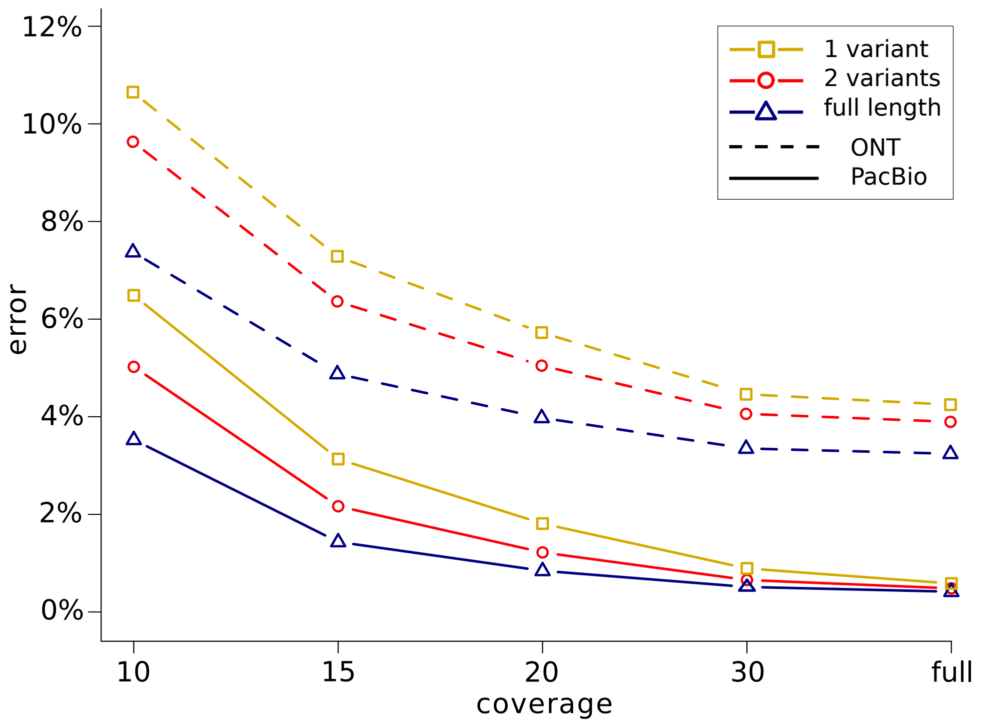


Figure 3.8: **Genotyping Errors** (wrt. to GIAB calls) as a function of coverage. The full length reads were used for genotyping (blue) and additionally, reads were cut such as to cover at most two variants (red) and one variant (yellow). Solid lines correspond to PacBio, dashed lines to Nanopore data.

of the provided sequencing reads increases. Furthermore, the coverage of the data would also affect the genotyping results.

In order to examine how the genotyping performance depends on the length of the sequencing reads and the coverage of the data, the following experiment was performed using the WhatsHap implementation. Both data sets (PacBio, Nanopore) were downsampled to average coverages $10\times, 20\times, 25\times$ and $30\times$. All SNVs inside of the high confidence regions in the GIAB truth set were re-genotyped from each of the resulting downsampled read sets, as well as from the full coverage data sets. Two versions of the genotyping algorithm were considered. First, the full length reads as given in the BAM files were provided to WhatsHap. Second, in an additional step prior to genotyping, the aligned sequencing reads were cut into shorter pieces such that each resulting fragment covered at most two variants. Additionally, we cut reads into fragments covering only one variant position. The genotyping performances of these genotyping procedures were finally compared by determining the amount of incorrectly genotyped variants.

Figure 3.8 shows the results of this experiment. On both data sets, the genotyping error increases as the length of reads decreases. Especially at lower coverages, the genotyping algorithm benefits from using the full length reads, which leads to much lower genotyping errors compared to using the shorter reads. In general, the experiment demonstrates that incorporating haplotype information gained from long reads does indeed improve the genotyping performance. Computing genotypes based on bipartitions of reads that represent possible haplotypes of the individual helps to reduce the number

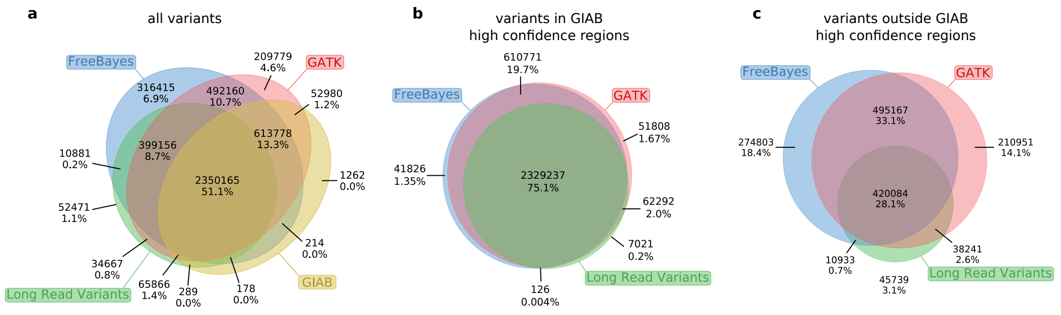


Figure 3.9: **Confirming Short Read Variants.** We examine all distinct variants found by our method, GIAB High Confidence, GATK/HC, and FreeBayes. Raw variant counts appear on top of each section, and the percentage of total variants is shown on bottom.

of genotyping errors, because it makes it easier to detect sequencing errors in the given reads.

3.5.5 Callset Consensus Analysis

In Figure 3.9, we further dissect the relation of our intersection call set (*Long Read Variants*, which refers to variants called by both WhatsHap on PacBio reads and MarginPhase on nanopore reads) to the GIAB truth set, as well as to the callsets from GATK/HC and FreeBayes, which both contributed to the GIAB truth set.

Figure 3.9a reveals that 399 156 variants in our *Long Read Variants* callset were called by both the GATK Haplotype Caller and FreeBayes, but are not in the GIAB truth set. To gather additional support for the quality of these calls, we consider two established quality metrics: the transition/transversion ratio (Ti/Tv), and the heterozygous/non-ref homozygous ratio (het/hom) [168]. The Ti/Tv ratio of these variants is 2.10 and the het/hom ratio is 1.29. These ratios are comparable to those of the GIAB truth set, which are 2.10 and 1.55, respectively. An examination of the Platinum Genomes benchmark set [35], an alternative to GIAB, reveals 71371 such long-read validated variants outside of their existing truth set.

We hypothesized that a callset based on long reads is particularly valuable in regions that were previously difficult to characterize. To investigate this, we separately examined the intersections of our *Long Read Variants* callset with the two short-read callsets both inside the GIAB high confidence regions and outside of them, see Figure 3.9b and Figure 3.9c, respectively. These Venn diagrams clearly indicate that the concordance of GATK and FreeBayes was indeed substantially higher in high confidence regions than outside. An elevated false positive rate of the short-read callers outside the high confidence regions is a plausible explanation for this observation. Interestingly, the fraction of calls concordant between FreeBayes and GATK for which we gather additional support is considerably lower outside the high confidence regions. This is again compatible with an increased number of false positives in the short read callsets, but we emphasize that these statistics should be interpreted with care in the absence of a reliable truth set for these regions.

3.5.6 Candidate Novel Variants

To demonstrate that our method allows for variant calling on more regions of the genome than short read variant calling pipelines, we have identified 15 498 variants which lie outside of the *Short Read Mappable* area, but inside the *Long Read Callable* regions, i.e. regions in which there is sequencing depth of at least 10 and not more than 2× the median depth for both sequencing technologies. We determined that 4.43 megabases of the genome (0.146%) is only mappable by long reads in this way.

Table 3.1 provides the counts of all variants found in each of the regions from Figure 3.6, as well as the counts for candidate variants, among the different types of genomic features described in Section 3.2. Over two thirds of the candidate variants occurred in the repetitive or duplicated regions described in the UCSC Genome Browser's repeatMasker track. The transition/transversion ratio of NA12878's 15 498 candidate variants is 1.64, and the heterozygous/homozygous ratio of these variants is 0.31. Given that we observe one candidate variant in every 325 haplotype bases, compared to one variant in every 1151 haplotype bases in the GIAB truth set, these candidate variants exhibit a 3.6× increase in the haplotype variation rate.

3.5.7 Runtimes

Whole genome variant detection using WhatsHap took 147 CPU hours on PacBio reads and 79.5 hours on Nanopore, of which genotyping took 42.2 and 32.8 hours respectively. The MarginPhase implementation took 583 CPU hours on PacBio and 330 on Nanopore, with an additional 1730 and 1220 hours for realignment.

3.6 Discussion

We present a method that uses a Hidden Markov Model to partition long reads into haplotypes, which we found to improve the quality of variant calling. This is evidenced by our experiment in cutting and downsampling reads, where reducing the number of variants spanned by any given read leads to decreased performance at all levels of read coverage.

Our analysis of the method against a high confidence truth set in high confidence regions shows false discovery rates (corresponding to one minus precision) between

Table 3.1: Distribution of candidate novel variants across different regions of interest. All variants refers to the variants in the Long Read Variants set, and Novel Variant Candidates are those described in Section 3.6.

	All Variants	**Novel Variant Candidates**
Total	2,913,942	15,498
Gencode v27 (ALL)	1,363,064	5,594
Gencode v27 exome	86,357	538
Repeat Masker	1,583,684	10,677
LINEs	690,859	5,161
SINEs	421,340	1,432
Segmental Duplications	157,341	5,683
Tandem Repeats	96,871	5,437
Centromeres	18,644	2,031
Telomeres	295	14

3 and 6 percent for PacBio, and between 24 and 29 percent for Nanopore. However, when considering a conservative set of variants confirmed by both long read technologies, the false discovery rate drops to around 0.3%, comparable with contemporary short read methods in these regions.

In analyzing the area of the genome with high quality long read mappings, we found roughly a half a percent of the genome (approximately fifteen megabases) that is mappable by long reads but not by short reads. This includes one percent of the human exome, as well as over ten percent of segmental duplications. Even though some of these areas have low read counts in our experimental data, the fact that they have high quality mappings means that they should be accessible with sufficient sequencing. We note that this is not the case for centromeric regions, where Illumina reads were able to map over twice as much as we found in our PacBio data. This may be a result of the low quality in long reads preventing them from uniquely mapping to these areas with an appreciable level of certainty.

Over our entire set of called variants, the Ti/Tv and het/hom ratios were similar to those reported by the truth set. The Ti/Tv ratio of 2.18 is slightly above the 2.10 reported in the GIAB callset, and the Het/Hom ratio of 1.36 is lower than the 1.55 found in the GIAB variants. In the 15 498 novel variant candidates produced by our method in regions unmappable by short reads, the Ti/Tv ratio of 1.64 is slightly lower than that of the truth set. This is not unexpected as gene-poor regions such as these tend to have more transversions away from C:G pairs [7]. We also observe that the Het/Hom ratio dropped to 0.31, which could be due to systematic biases in our callset or in the reference genome. The rate of variation in these regions was also notably different than in the high confidence regions, where we find three variants per thousand haplotype bases (3.6× the rate in high confidence regions). A previous study analyzing NA12878 [170] also found an elevated variation rate in regions where it is challenging to call variants, such as low complexity regions and segmental duplications. The study furthermore found clusters of variants in these regions, which we also observe.

The high precision of our intersected Nanopore/PacBio long read variants set makes it useful as strong evidence for confirming existing variant calls. As shown in the read coverage analysis, in both the GIAB and Platinum Genomes efforts many regions cannot be called with high confidence. In the excluded regions of GIAB we found just under 400 thousand variants using both Nanopore and PacBio reads with our methods, which were additionally confirmed with Illumina reads by two other variant callers, FreeBayes and GATK/HC. Given the extensive support of these variants from multiple sequencing technologies and variant callers, these variants are good candidates for addition to the GIAB truth set. Expansion of benchmark sets to harder-to-genotype regions of the human genome is generally important for the development of more comprehensive genotyping methods, and we plan to work with these efforts to use our results.

Further, our method is likely to prove useful for future combined diplotyping algorithms when both genotype and phasing is required, for example as may be used when constructing phased diploid *de novo* assemblies [27] or in future hybrid long/short read diplotyping approaches.

3.7 Acknowledgements

We thank the GIAB project for providing the data sets used. In particular, we thank Justin Zook for helpful discussions on how to use GIAB data, and Miten Jain for help with the nanopore data.

This work was supported, in part, by the National Human Genome Research Institute of the National Institutes of Health under Award Number 5U54HG007990 and grants from the W.M. Keck foundation and the Simons Foundation.

Part III

Allele Detection and Selection for Assembly Polishing With Nanopore Data

Chapter 4

Groundwork for a diploid-aware assembly polisher

4.1 Preamble

What follows is selected text from the Nature Biotechnology publication "Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes" [148], in which I share first-authorship with Kishwar Shafin, Ryan Lorig-Roach, Marina Haukness, and Hugh E. Olsen. This paper comprises four separate projects: the generation of ONT sequencing data for eleven samples, assembly of these samples using a novel method and three existing methods with comparisons across methodologies, demonstration of a polishing method using the Margin framework, and an improved polishing method involving a recurrent neural network ran on statistical summaries generated by Margin. For this chapter, I have excluded sections from the

Online Methods which do not pertain to work that I performed.

For this publication, I generated and evaluated a share of the assemblies alongside Kishwar Shafin, Ryan Lorig-Roach, and Marina Haukness; I implemented, refined, and evaluated the polishing method with Benedict Paten; and I worked closely with Kishwar Shafin to develop and fine-tune the RNN method. I helped write the main text of the paper and generate figures, including the MarginPolish and HELEN sections of the Online Methods.

I intended the work presented in this chapter to lay the groundwork for a diploid-aware polisher, but focused exclusively on haploid polishing for this publication. After submission, as progress was made towards a diploid implementation we found that different methodologies were better suited for allele detection and genotyping. The identification and correction of erroneous sequence that MarginPolish performs was largely transitioned to other tools in the eventual toolchain, but many of the improvements and additions to the Margin framework described in this chapter were critical to its eventual success.

4.2 Abstract

Present workflows for producing human genome assemblies from long-read technologies have cost and production time bottlenecks that prohibit efficient scaling to large cohorts. We demonstrate an optimized PromethION nanopore sequencing method for eleven human genomes. The sequencing, performed on one machine in nine days, achieved an average 63x coverage, 42 Kb read N50, 90% median read identity and 6.5x coverage in 100 Kb+ reads using just three flow cells per sample. To assemble these data we introduce new computational tools: Shasta - a *de novo* long read assembler, and MarginPolish & HELEN - a suite of nanopore assembly polishing algorithms. On a single commercial compute node Shasta can produce a complete human genome assembly in under six hours, and MarginPolish & HELEN can polish the result in just over a day, achieving greater than 99.9% identity (QV30) for haploid samples from nanopore reads alone. We evaluate assembly performance for diploid, haploid and trio-binned human samples in terms of accuracy, cost, and time and demonstrate improvements relative to current state-of-the-art methods in all areas. We further show that addition of proximity ligation (Hi-C) sequencing yields near chromosome-level scaffolds for all eleven genomes.

4.3 Introduction

Short-read sequencing reference-assembly mapping methods only assay about 90% of the current reference human genome assembly [37], and closer to 80% at high-confidence [181]. The latest incarnations of these methods are highly accurate with respect to single nucleotide variants (SNVs) and short insertions and deletions (indels) within this mappable portion of the reference genome [126]. However, short reads are much less able to *de novo* assemble a new genome [16], to discover structural variations (SVs) [4, 80] (including large indels and base-level resolved copy number variations), and are generally unable to resolve phasing relationships without exploiting transmission information or haplotype panels [22].

Third generation sequencing technologies, including linked-reads [12, 45, 169] and long-read technologies [69, 42], get around the fundamental limitations of short-read sequencing for genome inference by providing more information per sequencing observation. In addition to increasingly being used within reference guided methods [37, 67, 146, 124], long-read technologies can generate highly contiguous *de novo* genome assemblies [27].

Nanopore sequencing, as commercialized by Oxford Nanopore Technologies (ONT), is particularly applicable to *de novo* genome assembly because it can produce high yields of very long 100+ kilobase (Kb) reads [70]. Very long reads hold the promise of facilitating contiguous, unbroken assembly of the most challenging regions of the human genome, including centromeric satellites, acrocentric short arms, rDNA arrays, and recent segmental duplications [41, 47, 71]. We contributed to the recent consortium-wide effort to perform the *de novo* assembly of a nanopore sequencing based human genome [70]. This earlier effort required considerable resources, including 53 ONT MinION flow cells and an assembly process that required over 150,000 CPU hours and weeks of wall-clock time, quantities that are unfeasible for production scale replication.

Making nanopore long-read *de novo* assembly easy, cheap and fast will enable new research. It will permit both more comprehensive and unbiased assessment of human variation, and creation of highly contiguous assemblies for a wide variety of plant and animal genomes. Here we report the *de novo* assembly of eleven diverse human genomes at near chromosome scale using a combination of nanopore and proximity-ligation (HiC) sequencing [12]. We demonstrate a substantial improvement in yields and read lengths for human genome sequencing at reduced time, labor, and cost relative to earlier efforts. Coupled to this, we introduce a toolkit for nanopore data assembly and polishing that is orders of magnitude faster than state-of-the-art methods.

4.4 Results

4.4.1 Nanopore sequencing eleven human genomes in nine days

We selected for sequencing eleven, low-passage (six passages), human cell lines of the offspring of parent-child trios from the 1000 Genomes Project (1KGP) [31] and Genome-in-a-Bottle (GIAB) [177] sample collections. Samples were selected to maximize captured allelic diversity (see Online Methods).

We performed PromethION nanopore sequencing and HiC Illumina sequencing for the eleven genomes. Briefly, we isolated HMW DNA from flash-frozen 50 million cell pellets using the QIAGEN Puregene kit, with some modifications to the standard protocol to ensure DNA integrity (see Online Methods). For nanopore sequencing, we performed a size selection to remove fragments less than 10 kilobases (Kb) using the Circulomics SRE kit, followed by library preparation using the ONT ligation kit (SQK-LSK109). We used three flow cells per genome, with each flow cell receiving a nuclease flush every 20-24 hours. This flush removed long DNA fragments that could cause the pores to become blocked over time. Each flow cell received a fresh library of the same sample after the nuclease flush. A total of two nuclease flushes were performed per flow cell, and each flow cell received a total of three sequencing libraries. We used Guppy version 2.3.5 with the high accuracy flipflop model for basecalling (see Online Methods).

The nanopore sequencing for these eleven genomes was performed in nine days, producing 2.3 terabases of sequence. This was made possible by running up to 15 flow cells in parallel during these sequencing runs. Results are shown in Fig. 4.1 and Supplementary Tables A.1, A.2, and A.3. Nanopore sequencing yielded an average of 69 gigabases (Gb) per flow cell, with the total throughput per individual genome ranging between 48x (158 Gb) and 85x (280 Gb) coverage per genome (Fig. 4.1a). The read N50s for the sequencing runs ranged between 28 Kb and 51 Kb (Fig. 4.1b). We aligned nanopore reads to the human reference genome (GRCh38) and calculated their alignment identity to assess sequence quality (see Online Methods). We observed that the median and modal alignment identity was 90% and 93% respectively (Fig. 4.1c). The sequencing data per individual genome included an average of 55x coverage arising from 10 Kb+ reads, and 6.5x coverage from 100 Kb+ reads (Fig. 4.1d). This was in large part due to size-selection which yielded an enrichment of reads longer than 10 Kb. To test the generality of our sequencing methodology for other samples, we sequenced high-molecular weight DNA isolated from a human saliva sample using identical sample preparation. The library was run on a MinION (approximately one sixth the throughput of a ProMethION flow cell) and yielded 11 Gb of data at a read N50 of 28 Kb (Supplementary Table A.4), extrapolating both are within the lower range achieved with cell line derived DNA.

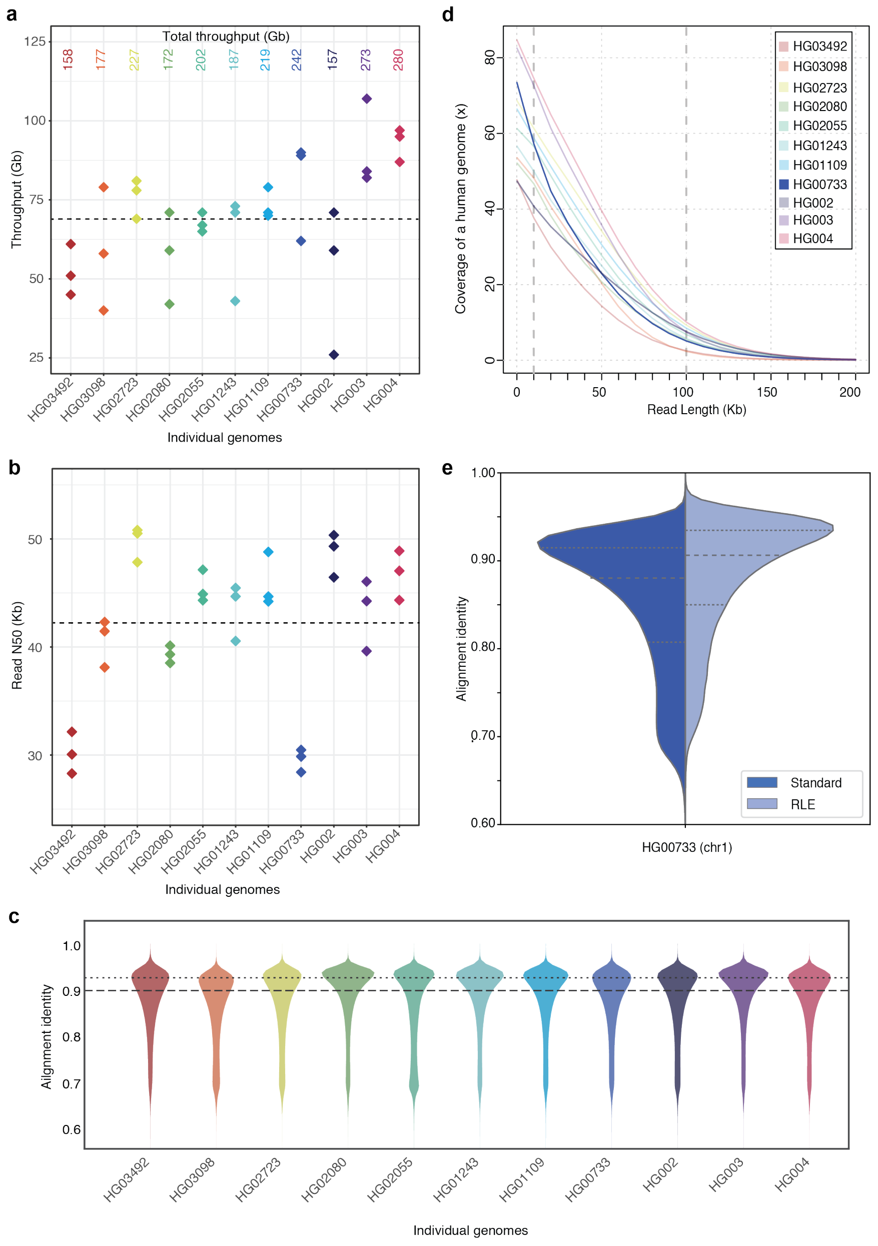


Figure 4.1: **Nanopore sequencing results. (a)** Throughput in gigabases from each of three flowcells for eleven samples, with total throughput at top. **(b)** Read N50s for each flowcell. **(c)** Alignment identities against GRCh38. Medians in a, b and c shown by dashed lines, dotted line in c is mode. **(d)** Genome coverage as a function of read length. Dashed lines indicate coverage at 10 and 100 Kb. HG00733 is bolded as an example. **(e)** Alignment identity for standard and run-length encoded (RLE) reads. Data for HG00733 chromosome 1 are shown. Dashed lines denote quartiles.

4.4.2 Shasta: assembling a human genome from nanopore reads in under 6 hours

To assemble the genomes, we developed a new *de novo* assembly algorithm, Shasta. Shasta was designed to be orders of magnitude faster and cheaper at assembling a human-scale genome from nanopore reads than the Canu assembler used in our earlier work [70]. A detailed description of algorithms and computational techniques used is provided in the Online Methods section. Here we summarize key points:

- During most Shasta assembly phases, reads are stored in a homopolymer-compressed (HPC) form using *Run-Length Encoding* (RLE) [92, 139, 111]. In this form, identical consecutive bases are collapsed, and the base and repeat count are stored. For example, `GATTTACCA` would be represented as (`GATACA`, `113121`). This representation is insensitive to errors in the length of homopolymer runs, thereby addressing the dominant error mode for Oxford Nanopore reads [69]. As a result, assembly noise due to read errors is decreased, and significantly higher identity alignments are facilitated (Fig. 4.1e).
- A *marker representation* of reads is also used, in which each read is represented as the sequence of occurrences of a predetermined, fixed subset of short k-mers (*marker representation*) in its run-length representation.
- A modified *MinHash* [18, 15] scheme is used to find candidate pairs of overlapping reads, using as *MinHash* features consecutive occurrences of m markers (default $m = 4$).

- Optimal alignments in marker representation are computed for all candidate pairs. The computation of alignments in marker representation is very efficient, particularly as various banded heuristics are used.

- A *Marker Graph* is created in which each vertex represents a marker found to be aligned in a set of several reads. The marker graph is used to assemble sequence after undergoing a series of simplification steps.

- The assembler runs on a single machine with a large amount of memory (typically 1-2 TB for a human assembly). All data structures are kept in memory, and no disk I/O takes place except for initial loading of the reads and final output of assembly results.

To validate Shasta, we compared it against three contemporary assemblers: Wtdbg2 [140], Flye [75] and Canu [78]. We ran all four assemblers on available read data from two diploid human samples, HG00733 and HG002, and one haploid human sample, CHM13. HG00733 and HG002 were part of our collection of eleven samples, and data for CHM13 came from the T2T consortium [159].

Canu consistently produced the most contiguous assemblies, with contig NG50s of 40.6, 32.3, and 79.5 Mb, for samples HG00733, HG002, and CHM13, respectively (Fig. 4.2a). Flye was the second most contiguous, with contig NG50s of 25.2, 25.9, and 35.3 Mb, for the same samples. Shasta was next with contig NG50s of 21.1, 20.2, and 41.1 Mb. Wtdbg2 produced the least contiguous assemblies, with contig NG50s of 15.3, 13.7, and 14.0 Mb.

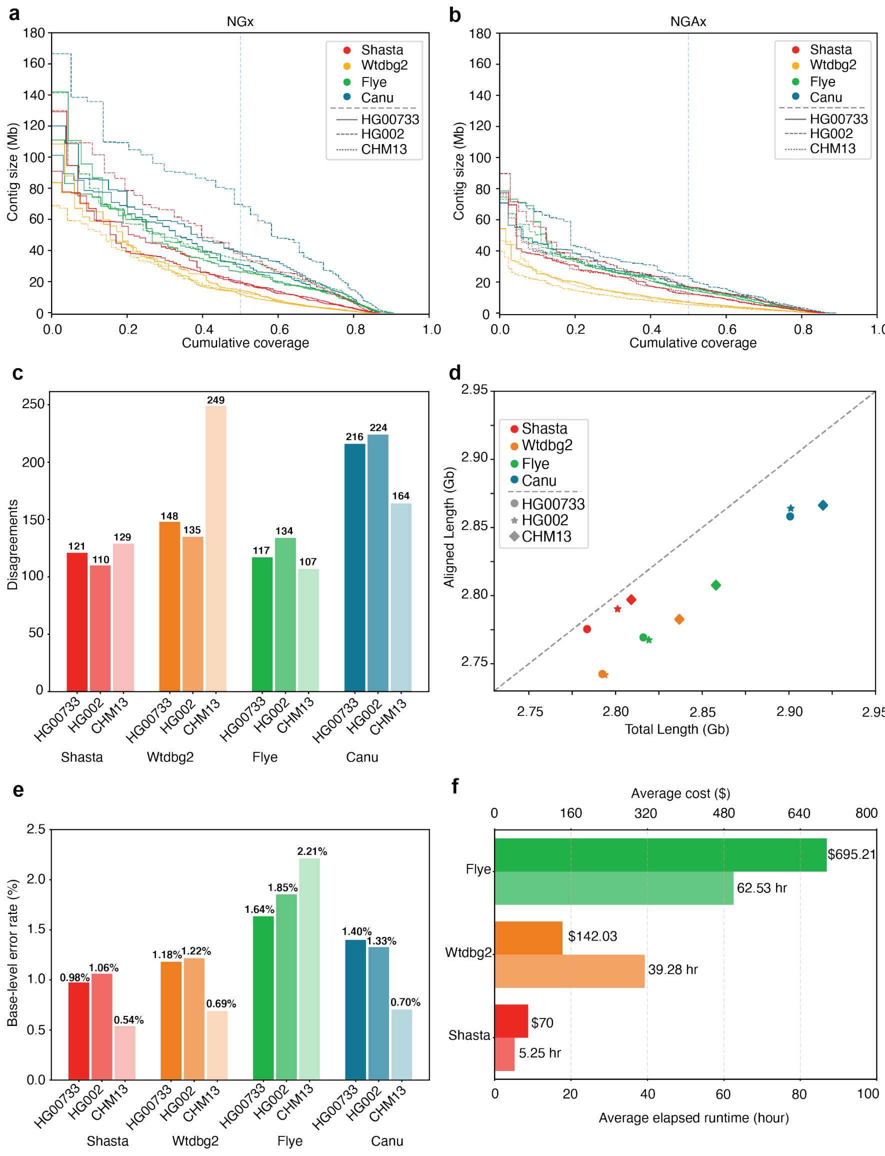


Figure 4.2: **Assembly results for four assemblers and three human samples, before polishing. (a)** NGx plot showing contig length distribution. The intersection of each line with the dashed line is the NG50 for that assembly. **(b)** NGAx plot showing the distribution of *aligned* contig lengths. Each horizontal line represents an aligned segment of the assembly unbroken by a disagreement or unmappable sequence with respect to GRCh38. The intersection of each line with the dashed line is the aligned NGA50 for that assembly. **(c)** Assembly disagreement counts for regions outside of centromeres, segmental duplications and, for HG002, known SVs. **(d)** Total generated sequence length vs. total aligned sequence length (against GRCh38). **(e)** Balanced base-level error rates for assembled sequences. **(f)** Average runtime and cost for assemblers (Canu not shown).

Conversely, aligning the samples to GRCh38 and evaluating with QUAST [110], Shasta had between 4.2 to 6.5x fewer disagreements (locations where the assembly contains a breakpoint with respect to the reference assembly) per assembly than the other assemblers (Supplementary Table A.5). Breaking the assemblies at these disagreements and unaligned regions with respect to GRCh38, we observe much smaller absolute variation in contiguity (Fig. 4.2b, Supplementary Table A.5). However, a substantial fraction of the disagreements identified likely reflect true SVs with respect to GRCh38. To address this we discounted disagreements within chromosome Y, centromeres, acrocentric chromosome arms, QH-regions, and known recent segmental duplications (all of which are enriched in SVs[8, 155]); in the case of HG002, we further excluded a set of known SVs [180]. We still observe between 1.2x to 2x fewer disagreements in Shasta relative to Canu and Wtdbg2, and comparable results against Flye (Fig. 4.2c, Supplementary Table A.6). To account for differences in the fraction of the genomes assembled, we analysed disagreements contained within the intersection of all the assemblies (i.e. in regions where all assemblers produced a unique assembled sequence). This produced results highly consistent with the prior analysis, and suggests Shasta and Flye have the lowest and comparable rates of misassembly (Online Methods, Supplementary Table. A.7). Finally, we used QUAST to calculate disagreements between the T2T Consortium's chromosome X assembly, a highly curated, validated assembly [159] and the subset of each CHM13 assembly mapping to it; Shasta has 2x to 17x fewer disagreements than the other assemblers while assembling almost the same fraction of the assembly (Supplementary Table A.8).

Canu consistently assembled the largest genomes (avg. 2.91 Gb), followed by Flye (avg. 2.83 Gb), Wtdbg2 (avg. 2.81 Gb) and Shasta (avg. 2.80 Gb). We would expect the vast majority of this assembled sequence to map to another human genome. Discounting unmapped sequence, the differences are smaller: Canu produced an avg. 2.86 Gb of mapped sequence per assembly, followed by Shasta (avg. 2.79 Gb), Flye (avg. 2.78 Gb) and Wtdbg2 (avg. 2.76 Gb) (Fig. 4.2d; see Online Methods). This analysis supports the notion that Shasta is currently relatively conservative vs. its peers, producing the highest proportion of directly mapped assembly per sample.

For HG00733 and CHM13 we examined a library of bacterial artificial chromosome (BAC) assemblies (see Online Methods). The BACs were largely targeted at known segmental duplications (473 of 520 BACs lie within 10 Kb of a known duplication). Examining the subset of BACs for CHM13 and HG00733 that map to unique regions of GRCh38 (see Online Methods), we find Shasta contiguously assembles all 47 BACs, with Flye performing similarly (Supplementary Table A.9). In the full set we observe that Canu (411) and Flye (282) contiguously assemble a larger subset of the BACs than Shasta (132) and Wtdbg2 (108), confirming the notion that Shasta is relatively conservative in these duplicated regions (Supplementary Table A.10). Examining the fraction of contiguously assembled BACs of all BACs represented in each assembly we can measure an aspect of assembly correctness. In this regard Shasta (97%) produces a much higher percentage of correct BACs in duplicated regions vs. its peers (Canu: 92%, Flye 87%, Wtdbg2 88%). In the intersected set of BACs attempted by all assemblers (Supplementary Table A.11) Shasta: 100%, Flye: 100%, Canu: 98.50% and Wtdbg2:

90.80% all produce comparable results.

Shasta produced the most base-level accurate assemblies (avg. balanced error rate 0.98% on diploid and 0.54% on haploid), followed by Wtbdg2 (1.18% on diploid and 0.69% on haploid), Canu (1.40% on diploid and 0.71% on haploid) and Flye (1.64% on diploid and 2.21% on haploid) (Fig. 4.2e); see Online Methods, Supplementary Table A.12. We also calculated the base level accuracy in regions covered by all the assemblies and observe results consistent with the whole genome assessment (Supplementary Table A.13).

Shasta, Wtdbg2 and Flye were run on a commercial cloud, allowing us to reasonably compare their cost and run time (Fig. 4.2e; see Online Methods). Shasta took an average of 5.25 hours to complete each assembly at an average cost of $70 per sample. In contrast, Wtdbg2 took 7.5x longer and cost 3.7x as much, and Flye took 11.9x longer and cost 9.9x as much. The Canu assemblies were run on a large compute cluster, consuming up to $19,000 (estimated) of compute and took around 4-5 days per assembly (see Online Methods, Supplementary Tables A.14, A.15).

To assess the utility of using Shasta for SV characterization we created a workflow to extract putative heterozygous SVs from Shasta assembly graphs (Online Methods). Extracting SVs from an assembly graph for HG002, the length distribution of indels shows the characteristic spikes for known retrotransposon lengths (Supplementary Fig. A.1). Comparing these SVs to the high-confidence GIAB SV set we find good concordance, with a combined F1 score of 0.68 (Supplementary Table A.16).

4.4.3 Contiguously assembling MHC haplotypes

The Major Histocompatibility Complex (MHC) region is difficult to resolve using short reads due to its repetitive and highly polymorphic nature [17], but recent efforts to apply long read sequencing to this problem have shown promise [70, 161]. We analyzed the assemblies of CHM13 and HG00733 to see if they spanned the region. For the haploid assembly of CHM13 we find MHC is entirely spanned by a single contig in all 4 assemblers' output, and most closely resembles the GL000251.2 haplogroup among those provided in GRCh38 (Fig. 4.3a; Supplementary Fig. A.2 and Supplementary Table A.17). In the diploid assembly of HG00733 two contigs span the large majority of the MHC for Shasta and Flye, while Canu and Wtdbg2 span the region with one contig (Fig. 4.3b; Supplementary Fig. A.3). However, we note that the chimeric diploid assembly leads to sequences that do not closely resemble any haplogroup (see Online Methods).

To attempt to resolve haplotypes of HG00733 we performed trio-binning [76], where we partitioned all the reads for HG00733 into two sets based on likely maternal or paternal lineage and assembled the haplotypes (see Online Methods). For all haplotype assemblies the global contiguity worsened significantly (as the available read data coverage was approximately halved, and further, not all reads could be partitioned), but the resulting disagreement count decreased (Supplementary Table A.18). When using haploid trio-binned assemblies, the MHC was spanned by a single contig for the maternal haplotype (Fig. 4.3c, Supplementary Fig. A.4, Supplementary Table A.19),

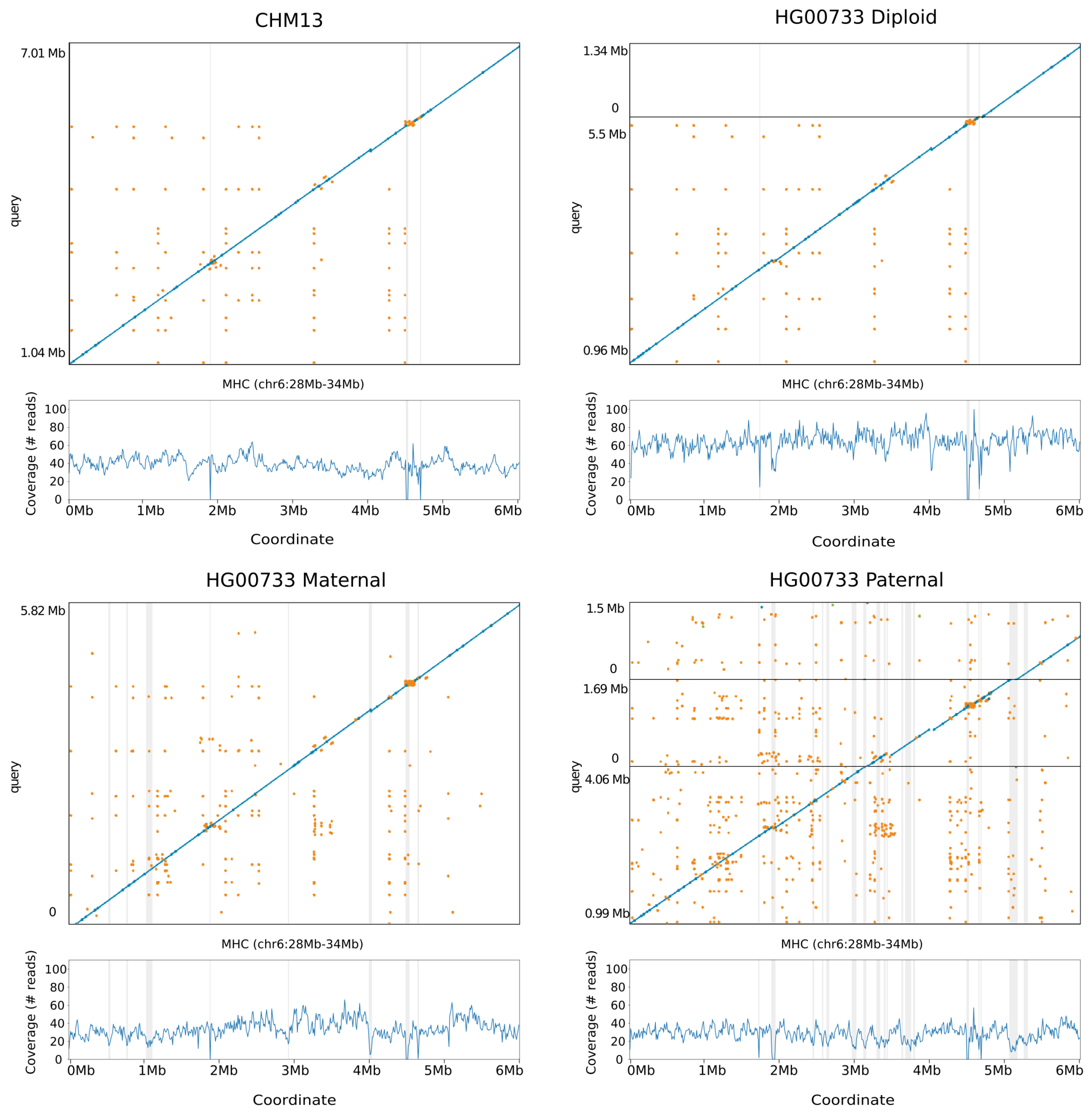


Figure 4.3: **Shasta MHC assemblies vs GRCh38.** Unpolished Shasta assembly for CHM13 and HG00733, including HG00733 trio-binned maternal and paternal assemblies. Shaded gray areas are regions in which coverage (as aligned to GRCh38) drops below 20. Horizontal black lines indicate contig breaks. Blue and green describe unique alignments (aligning forward and reverse, respectively) and orange describes multiple alignments.

with high identity to GRCh38 and having the greatest contiguity and identity with the GL000255.1 haplotype. For the paternal haplotype, low coverage led to discontinuities (Fig. 4.3d) breaking the region into three contigs.

4.4.4 Deep neural network based polishing achieves greater than QV30 long-read only haploid polishing accuracy

Accompanying Shasta, we developed a deep neural network based consensus sequence polishing pipeline designed to improve the base-level quality of the initial assembly. The pipeline consists of two modules: MarginPolish and HELEN. MarginPolish uses a banded form of the forward-backward algorithm on a pairwise hidden Markov model (pair-HMM) to generate pairwise alignment statistics from the RLE alignment of each read to the assembly [34]. From these statistics MarginPolish generates a weighted RLE Partial Order Alignment (POA) graph [86] that represents potential alternative local assemblies. MarginPolish iteratively refines the assembly using this RLE POA, and then outputs the final summary graph for consumption by HELEN. HELEN employs a multi-task recurrent neural network (RNN) [107] that takes the weights of the MarginPolish RLE POA graph to predict a nucleotide base and run-length for each genomic position. The RNN takes advantage of contextual genomic features and associative coupling of the POA weights to the correct base and run-length to produce a consensus sequence with higher accuracy.

To demonstrate the effectiveness of MarginPolish and HELEN, we compared them with the state-of-the-art nanopore assembly polishing workflow: four iterations of Racon polishing [163] followed by Medaka [100]. Here MarginPolish is analogous in function to Racon, both using pair-HMM based methods for alignment and POA graphs for initial refinement. Similarly, HELEN is analogous to Medaka, in that both use a deep neural network and both work from summary statistics of reads aligned to the assembly.

Figure 4.4a and Supplementary Tables A.20, A.21 and A.22 detail error rates for the four methods performed on the HG00733 and CHM13 Shasta assemblies (see Online Methods) using Pomoxis [101]. For the diploid HG00733 sample MarginPolish and HELEN achieve a balanced error rate of 0.388% (QV 24.12), compared to 0.455% (QV 23.42) by Racon and Medaka. For both polishing pipelines, a significant fraction of these errors are likely due to true heterozygous variations. For the haploid CHM13 we restrict comparison to the highly curated X chromosome sequence provided by the T2T consortium [159]. We achieve a balanced error rate of 0.064% (QV 31.92), compared to Racon and Medaka's 0.110% (QV 29.59).

For all assemblies, errors were dominated by indel errors, e.g. substitution errors are 3.16x and 2.9x fewer than indels in the polished HG000733 and CHM13 assemblies, respectively. Many of these errors relate to homopolymer length confusion; Fig. 4.4b analyzes the homopolymer error rates for various steps of the polishing workflow for HG00733. Each panel shows a heatmap with the true length of the homopolymer run on the y-axis and the predicted run length on the x-axis, with the color describing the likelihood of predicting each run length given the true length. Note that the dispersion of the diagonal steadily decreases. The vertical streaks at high run lengths in the

MarginPolish and HELEN confusion-matrix are the result of infrequent numerical and encoding artifacts (see Online Methods, Supplementary Fig. A.5).

Figure 4.4c and Supplementary Table A.23 show the overall error rate after running MarginPolish and HELEN on HG00733 assemblies generated by different assembly tools, demonstrating that they can be usefully employed to polish assemblies generated by other tools.

To investigate the benefit of using short reads for further polishing, we polished chromosome X of the CHM13 Shasta assembly after MarginPolish and HELEN using 10X Chromium reads with the Pilon polisher [167]. This led to a ˜2x reduction in base errors, increase the Q score from ˜QV32 (after polishing with MarginPolish and HELEN) to ˜QV36 (Supplementary Table A.24). Notably, attempting to use Pilon polishing on the raw Shasta assembly resulted in much poorer results (QV24).

Figure 4.4d and Supplementary Table A.25 describe average runtimes and costs for the methods (see Online Methods). MarginPolish and HELEN cost a combined $107 and took 29 hours of wall-clock time on average, per sample. In comparison Racon and Medaka cost $621 and took 142 wall-clock hours on average, per sample. To assess single-region performance we additionally ran the two polishing workflows on a single contig (roughly 1% of the assembly size), MarginPolish/HELEN was 3.0x faster than Racon (1x)/Medaka (Supplementary Table A.26).

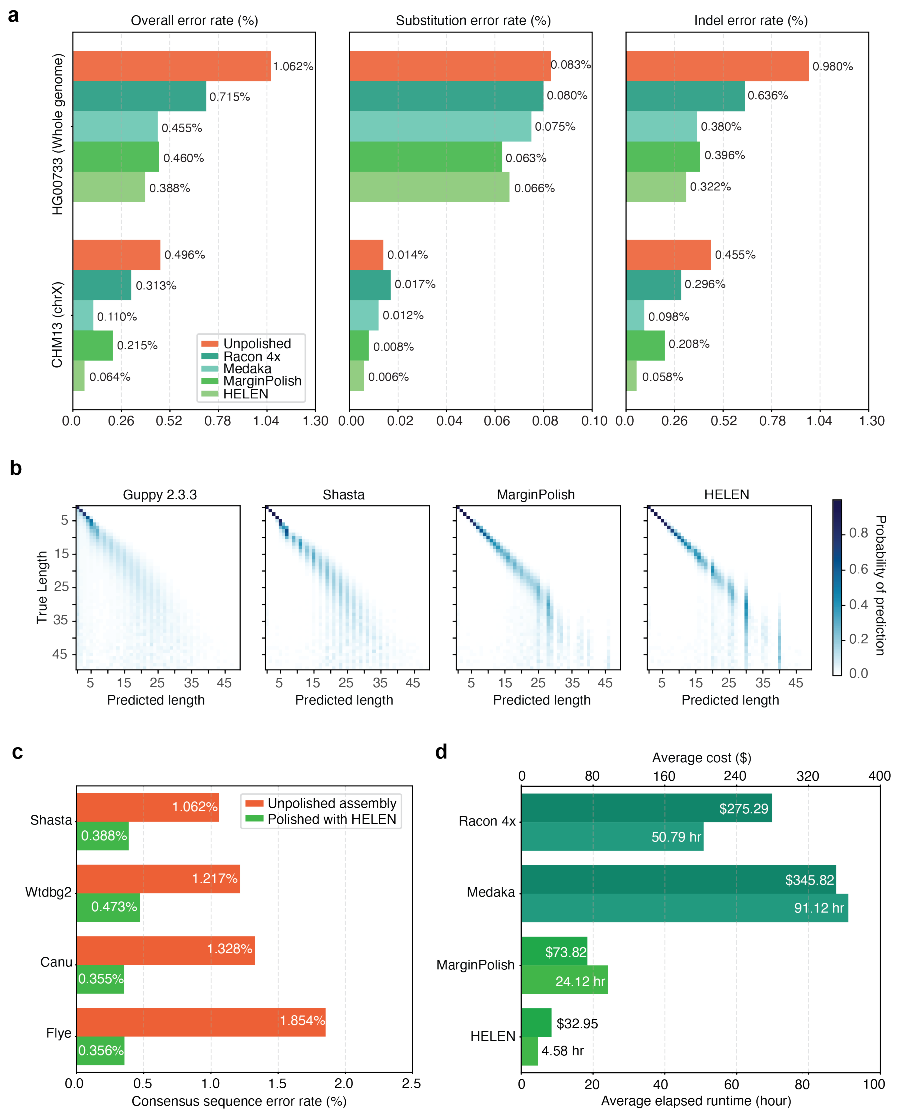


Figure 4.4: **Polishing Results.** **(a)** Balanced error rates for the four methods on HG00733 and CHM13. **(b)** Row-normalized heatmaps describing the predicted run-lengths (x-axis) given true run lengths (y-axis) for four steps of the pipeline on HG00733. **(c)** Error rates for MarginPolish and HELEN on four assemblies. **(d)** Average runtime and cost.

Sample	Assembler	Polisher	Genes Found %	Missing Genes	Complete Genes %
HG00733	Canu	HELEN	99.741	51	67.038
	Flye	HELEN	99.405	117	71.768
	Wtdbg2	HELEN	97.429	506	66.143
	Shasta	HELEN	99.228	152	68.069
	Shasta	Medaka	99.141	169	66.27
CHM13	Shasta	HELEN	99.111	175	74.202
	Shasta	Medaka	99.035	190	73.836

Table 4.1: CAT transcriptome analysis of human protein coding genes for HG00733 and CHM13.

4.4.5 Long-read assemblies contain nearly all human coding genes

To evaluate the accuracy and completeness of an assembled transcriptome we ran the Comparative Annotation Toolkit [46], which can annotate a genome assembly using the human GENCODE [49] reference human gene set (Table 4.1, Online Methods, Supplementary Tables A.27, A.28, A.29, and A.30.).

For the HG00733 and CHM13 samples we found that Shasta assemblies polished with MarginPolish and HELEN were close to representing nearly all human protein coding genes, having, respectively, an identified ortholog for 99.23% (152 missing) and 99.11% (175 missing) of these genes. Using the restrictive definition that a coding gene is complete in the assembly only if it is assembled across its full length, contains no frameshifts, and retains the original intron/exon structure, we found that 68.07% and 74.20% of genes, respectively, were complete in the HG00733 and CHM13 assemblies. Polishing the Shasta assemblies alternatively with the Racon-Medaka pipeline achieved similar but uniformly less complete results.

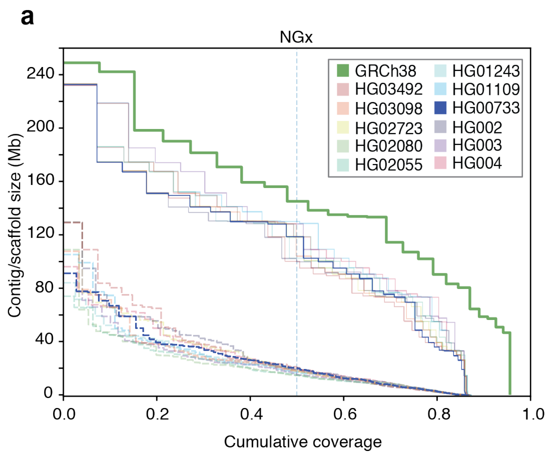
Comparing the MarginPolish and HELEN polished assemblies for HG00733 generated with Flye, Canu and Wtdbg2 to the similarly polished Shasta assembly we found that Canu had the fewest missing genes (just 51), but that Flye, followed by Shasta, had the most complete genes. Wtdbg2 was clearly an outlier, with notably larger numbers of missing genes (506). For comparison we additionally ran BUSCO [152] using the eukaryote set of orthologs on each assembly, a smaller set of 303 expected single-copy genes (Supplementary Tables A.31 and A.32). We find comparable performance between the assemblies, with small differences largely recapitulating the pattern observed by the larger CAT analysis.

4.4.6 Comparing to a PacBio HiFi Assembly

We compared the CHM13 Shasta assembly polished using MarginPolish and HELEN with the recently released Canu assembly of CHM13 using PacBio HiFi reads [165]; HiFi reads being based upon circular consensus sequencing technology that delivers significantly lower error rates. The HiFi assembly has lower NG50 (29.0 Mb vs. 41.0 Mb) than the Shasta assembly (Supplementary Fig. A.6). Consistent with our other comparisons to Canu, the Shasta assembly also contains a much lower disagreement count relative to GRCh38 (1073) than the Canu based HiFi assembly (8469), a difference which remains after looking only at disagreements within the intersection of the assemblies (380 vs. 594). The assemblies have an almost equal NGAx (˜20.0Mb), but the Shasta assembly covers a smaller fraction of GRCh38 (95.28% vs. 97.03%) (Supplementary Fig. A.7, Supplementary Table A.33). Predictably, the HiFi assembly has a higher QV value than the polished Shasta assembly (QV41 vs. QV32).

4.4.7 Assembling, polishing and scaffolding 11 human genomes at near chromosome scale

To achieve chromosome length sequences we scaffolded all of the polished Shasta assemblies with HiC proximity-ligation data using HiRise [129] (see Online Methods, Fig. 4.5a). On average, 891 joins were made per assembly. This increased the



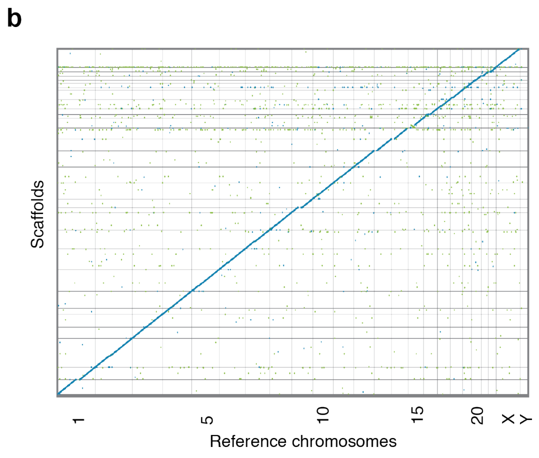


Figure 4.5: **HiRise scaffolding for 11 genomes.** **(a)** NGx plots for each of the 11 genomes, before (dashed) and after (solid) scaffolding with HiC sequencing reads, GRCh38 minus alternate sequences is shown for comparison. **(b)** Dot plot showing alignments between the scaffolded HG00733 Shasta assembly and GRCh38 chromosome scaffolds. Blue indicates forward aligning segments, green indicates reverse, with both indicating unique alignments.

scaffold NG50s to near chromosome scale, with a median of 129.96 Mb, as shown in Fig. 4.5a, with additional assembly metrics in Supplementary Table A.36. Proximity-ligation data can also be used to detect misjoins in assemblies. In all 11 Shasta assemblies, no breaks to existing contigs were made while running HiRise to detect potential misjoins. Aligning HG00733 to GRCh38, we find no major rearrangements and all chromosomes are spanned by one or a few contigs (Fig. 4.5b), with the exception of chrY which is absent because HG00733 is female. Similar results were observed for HG002 (Supplementary Fig. A.8).

4.5 Discussion

In this paper we demonstrate the sequencing and assembly of eleven diverse human genomes in a time and cost efficient manner using a combination of nanopore and proximity ligation sequencing.

The PromethION realizes dramatic improvements in yield per flow cell, allowing the sequencing of each genome with just three flow cells at an average coverage of 63x. This represents a large reduction in associated manual effort and a dramatic practical improvement in parallelism; a single PromethION allows up to 48 flow cells to be run concurrently. Here we completed all 2.3 terabases of nanopore data collection in nine days on one PromethION, running up to 15 flow cells simultaneously (it is now possible to run 48 concurrently). In terms of contemporary long-read sequencing platforms, this throughput is unmatched.

Due to the length distribution of human transposable elements, we found it better to discard reads shorter than 10 Kb to prevent multi-mapping. The Circulomics SRE kit reduced the fraction of reads ¡10 Kb to around 13%, making the majority usable for assembly. Conversely, the right tail of the read length distribution is long, yielding an average of 6.5x coverage per genome in 100 Kb+ reads. This represents an enrichment of around 7 fold relative to our earlier MinION effort [70]. In terms of assembly, the result was an average NG50 of 18.5 Mb for the 11 genomes, ˜3x higher than in that initial effort, and comparable with the best achieved by alternative technologies [42, 172]. We found the addition of HiC sequencing for scaffolding necessary to achieve chromosome scale, making 891 joins on average per assembly. However, our results are consistent with previous modelling based on the size and distribution of large repeats in the human genome, which predicts that an assembly based on 30x coverage of such 100 Kb+ reads would approach the continuity of complete human chromosomes [70, 159].

Relative to alternate long-read and linked-read sequencing, the read identity of nanopore reads has proven lower [69, 70]. However, original reports of 66% identity [69] for the original MinION are now historical footnotes: we observe modal read identity of 92.5%, resulting in better than QV30 base quality for haploid polished assembly from nanopore reads alone. The accurate resolution of highly repetitive and recently duplicated sequence will depend on long-read polishing, because short-reads are generally not uniquely mappable. Further polishing using complementary data types, including PacBio HiFi reads [172] and 10x Chromium [104], will likely prove useful in achieving QV40+ assemblies.

The advent of third generation technologies has dramatically lowered the cost of high-contiguity long-read *de novo* assembly relative to earlier methods [87]. This cost reduction is still clearly underway. The first MinION human assembly cost ~$40,000 in flow cells and reagents [70]. After a little over a year, the equivalent cost per sample here was ~$6,000. At bulk with current list-pricing, this cost would be reduced to ~$3,500 per genome. It is not unreasonable to expect further yield growth and resulting cost reduction of nanopore and competing platforms such that we foresee $1,000 total sequencing cost high-contiguity *de novo* plant and animal genome assembly being achieved - a milestone that will likely make many ambitious comparative genomic efforts economic [120, 89].

With sequencing efficiency for long-reads improving, computational considerations are paramount in figuring overall time, cost and quality. Simply put, large genome *de novo* assembly will not become ubiquitous if the requirements are weeks of assembly time on large computational clusters. We present three novel methods that provide a pipeline for the rapid assembly of long nanopore reads. Shasta can produce a draft human assembly in around six hours and $70 using widely available commercial cloud nodes. This cost and turnaround time is much more amenable to rapid prototyping and parameter exploration than even the fastest competing method (Wtdbg2), which was on average 7.5x slower and 3.7x more expensive. Connected together, the three tools presented allow a polished assembly to be produced in ~24 hours and for ~$180, against the fastest comparable combination of Wtdbg2, Racon, and Medaka which costs 5.3x more and is 4.3x slower while producing measurably worse results in terms of disagreements, contiguity and base-level accuracy. Substantial further parallelism of polishing, the dominant time component in our current pipeline, is easily possible. We are now working toward the goal of having a half-day turn around of our complete computational pipeline. With real-time base calling, a DNA-to-*de novo* assembly could be achieved in less than 96 hours with little difficulty. Such speed could make these techniques practical for screening human genomes for abnormalities in difficult-to-sequence regions.

All three presented computational methods employ run-length encoding of reads. By operating on homopolymer-compressed nucleotide sequences, we mitigate effects of the dominant source of error in nanopore reads [130] and enable the use of different models for addressing alignment and run-length estimation orthogonally.

Shasta produces a notably more conservative assembly than competing tools, trading greater correctness for contiguity and total produced sequence. For example, the ratio of total length to aligned length is relatively constant for all other assemblers, where approximately 1.6% of sequence produced does not align across the three evaluated samples. In contrast, on average just 0.38% of Shasta's sequence does not align to GRCh38, representing a more than 4x reduction in unaligned sequence. Additionally, we note substantially lower disagreement counts, resulting in much smaller differences between the raw NGx and corrected NGAx values. Shasta also produces substantially more base-level accurate assemblies than the other competing tools. MarginPolish and HELEN provide a consistent improvement of base quality over all tested assemblers, with more accurate results than the current state-of-the-art long read polishing workflow.

We have assembled and compared haploid, trio-binned and diploid samples.

Trio binned samples show great promise for haplotype assembly, for example contiguously assembling an MHC haplogroup, but the halving of effective coverage resulted in ultimately less contiguous human assemblies with higher base-error rates than the related, chimeric diploid assembly. This can potentially be rectified by merging the haplotype assemblies to produce a pseudo-haplotype or increasing sequencing coverage. Indeed the improvements in contiguity and base accuracy in CHM13 over the diploid samples illustrate what can be achieved with higher coverage of a haploid sample. We believe that one of the most promising directions for the assembly of diploid samples is the integration of phasing into the assembly algorithm itself, as pioneered by others [27, 52, 88]. We anticipate that the novel tools we've described here are suited for this next step: the Shasta framework is well placed for producing phased assemblies over structural variants, MarginPolish is built off of infrastructure designed to phase long reads [37], and the HELEN model could be improved to include haplotagged features for the identification of heterozygous sites.

4.6 Acknowledgements

The authors are grateful for support from the following individuals. Dan Turner, David Stoddart, Androo Markham, and Jonathan Pugh (ONT) provided advice on method development and basecalling. Chris Wright (ONT) provided advice for Medaka. Daniel Garalde, and Rosemary Dokos (ONT) provided advice on the PromethION for parallelized DNA sequencing and basecalling.

The authors are grateful to Amazon Web Services (AWS) for hosting the data via their AWS Public Dataset Program.

AP and SK were supported by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health. This work utilized the computational resources of the NIH HPC Biowulf cluster (`https://hpc.nih.gov`).

Sidney Bell and Charlotte Weaver from Chan Zuckerberg Initiative (CZI) provided support on development and documentation. CZI further supported this effort by funding the usage of Amazon Web Services (AWS) for the project.

Certain commercial equipment, instruments, or materials are identified to specify adequately experimental conditions or reported results. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment, instruments, or materials identified are necessarily the best available for the purpose.

This work was supported, in part, by the National Institutes of Health (award numbers: 5U54HG007990, 5T32HG008345-04, 1U01HL137183, R01HG010053, U01HL137183, and U54HG007990 to BP and DH; R01HG010329 to SRS and DH), by Oxford Nanopore Research Grant SC20130149 (MA), the Howard Hughes Medical Institute (DH), the Gencode (NIH/NHGRI/EMBL) Grant 2U41HG007234, TopMEd (NIH/NHLBI) Grant 1U01HL137183, W.M. Keck Foundation Grant DT06172015, BD2K Grant 1U54HG007990, the technological pilot using ONT nanopore sequencing (NIH/NHGRI) Grant 3U24HG009084-03S1, and High coverage long read nanopore sequencing of reference human genomes

(NIST) Grant 70NANB18H224.

4.7 Online Methods

4.7.1 Analysis methods

4.7.1.1 Read alignment identities

To generate the identity violin plots (Fig. 4.1c/e) we aligned all the reads for each sample and flowcell to GRCh38 using `minimap2` [92] with the `map-ont` preset. Using a custom script `get_summary_stats.py` in the repository `https://github.com/rlorigro/nanopore_assembly_and_polishing_assessment`, we parsed the alignment for each read and enumerated the number of matched ($N_=$), mismatched (N_X), inserted (N_I), and deleted (N_D) bases. From this, we calculated *alignment identity* as $N_=/(N_= + N_X + N_I + N_D)$. These identities were aggregated over samples and plotted using the `seaborn` library with the script `plot_summary_stats.py` in the same repository. This method was used to generate both Figure 4.1c and Figure 4.1e. For Figure 4.1e, we selected reads from HG00733 flowcell1 aligned to GRCh38 chr1. The "Standard" identities are used from the original reads/alignments. To generate identity data for the "RLE" portion, we extracted the reads above, run-length encoded the reads and chr1 reference, and followed the alignment and identity calculation process described above. Sequences were run-length encoded using a simple script (`github.com/rlorigro/runlength_analysis/blob/master/runlength_encode_fasta.py`) and aligned with minimap2 using the `map-ont` preset and `--k 19`.

4.7.1.2 Runtime and Cost Analysis

Our runtime analysis was generated with multiple methods detailing the amount of time the processes took to complete. These methods include the unix command `time` and a home-grown resource tracking script which can be found in the `https://github.com/rlorigro/TaskManager` repository. We note that the assembly and polishing methods have different resource requirements, and do not all fully utilize available CPUs, GPUs, and memory over the program's execution. As such, we report runtimes using wall clock time and the number of CPUs the application was configured to use, but do not convert to CPU hours. Costs reported in the figures are the product of the runtime and AWS instance price. Because portions of some applications do not fully utilize CPUs, cost could potentially be reduced by running on a smaller instance which would be fully utilized, and runtime could be reduced by running on a larger instance which can be fully utilized for some portion of execution. We particularly note the long runtime of Medaka and found that for most of the total runtime, only a single CPU was used. Lastly, we note that data transfer times are not reported in runtimes. Some of the data required or generated exceeds hundreds of gigabytes, which could be potentially significant in relation to the runtime of the process. Notably, the images generated by MarginPolish and consumed by HELEN were often greater than 500 GB in total.

All recorded runtimes are reported in the supplement. For Shasta, times were recorded to the tenth of the hour. All other runtimes were recorded to the minute. All runtimes reported in figures were run on the Amazon Web Services cloud platform

(AWS).

Shasta runtime reported in Fig. 4.2f was determined by averaging across all 12 samples. Wtdbg2 runtime was determined by summing runtimes for wtdbg2 and wtpoa-cns and averaging across the HG00733, HG002, and CHM13 runs. Flye runtime was determined by averaging across the HG00733, HG002, and CHM13 runs, which were performed on multiple instance types (x1.16xlarge and x1.32xlarge). We calculated the total cost and runtime for each run and averaged these amounts; no attempt to convert these to a single instance type was performed. Precise Canu runtimes are not reported, as they were run on the NIH Biowulf cluster. Each run was restricted to nodes with 28 cores (56 hyperthreads) (2x2680v4 or 2x2695v3 Intel CPUs) and 248GB of RAM or 16 cores (32 hyperthreads) (2x2650v2 Intel CPUs) and 121GB of RAM. Full details of the cluster are available at `https://hpc.nih.gov`. The runs took between 219 and 223 thousand CPU hours (4-5 wall-clock days). No single job used more than 80GB of RAM/12 CPUs. We find the r5.4xlarge ($1.008 per hour) to be the cheapest AWS instance type possible considering this resource usage, which puts estimated cost between $18,000 and $19,000 per genome.

For MarginPolish, we recorded all runtimes, but used various thread counts that did not always fully utilize the instance's CPUs. The runtime reported in the figure was generated by averaging across 8 of the 12 samples, selecting runs that used 70 CPUs (of the 72 available on the instance). The samples this was true for were GM24385, HG03492, HG01109, HG02055, HG02080, HG01243, HG03098, and CHM13. Runtimes for read alignments used by MarginPolish were not recorded. Because MarginPolish requires an aligned BAM, we found it unfair to not report this time in the figure as it is a required step in the workflows for MarginPolish, Racon, and Medaka. As a proxy for the unrecorded read alignment time used to generate BAMs for MarginPolish, we added the average alignment time recorded while aligning reads in preparation for Medaka runs. We note that the alignment for MarginPolish was done by piping output from `minimap2` directly into `samtools sort`, and piping this into `samtools view` to filter for primary and supplementary reads. Alignment for Medaka was done using `mini_align`, which is a wrapper for `minimap2` bundled in Medaka that simultaneously sorts output.

Reported HELEN runs were performed on GCP except for HG03098, but on instances that match the AWS instance type `p2.8xlarge` in both CPU count and GPU (NVIDIA Tesla P100). As such, the differences in runtime between the platforms should be negligible, and we have calculated cost based on the AWS instance price for consistency. The reported runtime is the sum of time taken by `call_consensus.py` and `stitch.py`. Unannotated runs were performed on UCSC hardware.

Racon runtimes reflect the sum of four series of read alignment and polishing. The time reported in the figure is the average of the runtime of this process run on the Shasta assembly for HG00733, HG002, and CHM13.

Medaka runtime was determined by averaging across the HG00733, HG002, and CHM13 runs after running Racon 4$\times$ on the Shasta assembly. We again note that this application in particular did not fully utilize the CPUs for most of the execution, and in the case of HG00733 appeared to hang and was restarted. The plot includes the average runtime from read alignment using `minialign`; this is separated in the tables in the supplementary results. We ran Medaka on an `x1.16xlarge` instance, which had more memory than was necessary. When determining cost, we chose to price the run based on the cheapest AWS instance type that we could have used accounting for configured CPU count and peak memory usage (`c5n.18xlarge`). This instance could have supported 8 more concurrent threads, but as the application did not fully utilize the CPUs we find this to be a fair representation.

4.7.1.3 BAC Analysis

At a high level, the BAC analysis was performed by aligning BACs to each assembly, quantifying their resolution, and calculating identity statistics on those that were fully resolved.

We obtained 341 BACs for CHM13 [81, 164] and 179 for HG00733 [22] (complete BAC clones of `VMRC62`), which had been selected primarily by targeting complex or highly duplicated regions. We performed the following analysis on the full set of of BACs (for CHM13 and HG00733), and a subset selected to fall within unique regions of the genome. To determine this subset, we selected all BACs which are greater than 10 Kb away from any segmental duplication, resulting in 16 of HG00733 and 31 of CHM13. This subset represents simple regions of the genome which we would expect all assemblers to resolve.

For the analysis, BACs were aligned to each assembly with the command `minimap2 --secondary=no -t 16 -ax asm20 assembly.fasta bac.fasta`

`> assembly.sam` and converted to a PAF-like format which describes aligned regions of the BACs and assemblies. Using this, we calculated two metrics describing how resolved each BAC was: *closed* is defined as having 99.5% of the BAC aligned to a single locus in the assembly; *attempted* is defined as having a set of alignments covering >= 95% of the BAC to a single assembly contig where all alignments are at least 1kb away from the contig end. If such a set exists, it counts as attempted. We furthermore calculate median and mean identities (using alignment identity metric described above) of the closed BACs. These definitions were created such that a contig that is counted as attempted but not closed likely reflects a disagreement. The code for this can be found at `https://github.com/skoren/bacValidation`.

4.7.2 MarginPolish

Throughout we used MarginPolish (`https://github.com/ucsc-nanopore-cgl/MarginPolish`) version 1.0.0.

MarginPolish is an assembly refinement tool designed to sum over (marginalize) read to assembly alignment uncertainty. It takes as input a genome assembly and set of aligned reads in BAM format.

It outputs a refined version of the input genome assembly after attempting to correct base-level errors in terms of substitutions and indels (insertions and deletions). It can also output a summary representation of the assembly and read alignments as a weighted partial order alignment graph (POA), which is used by the HELEN neural network based polisher described below.

It was designed and is optimized to work with noisy long ONT reads, although parameterization for other, similar read types is easily possible. It does not yet consider signal-level information from ONT reads. It is also currently a haploid polisher, in that it does not attempt to recognize or represent heterozygous polymorphisms or phasing relationships. For haploid genome assemblies of a diploid genome it will therefore fail to capture half of all heterozygous polymorphisms.

Algorithm Overview MarginPolish works in overview as follows:

1. Reads and the input assembly are converted to their run-length encoding (RLE) (see Shasta description above for description and rationale).

2. A restricted, weighted Partial Order Alignment [86] (POA) graph is constructed representing the RLE input assembly and potential edits to it in terms of substitutions and indels.

3. Within identified regions of the POA containing likely assembly errors:

   - A set of alternative sequences representing combinations of edits are enumerated by locally traversing the POA within the region.
   - The likelihood of the existing and each alternative sequence is evaluated given the aligned reads.
   - If an alternative sequence with higher likelihood than the current reference exists then the assembly at the location is updated with this higher likelihood sequence.

4. Optionally, the program loops back to step 2 to repeat the refinement process (by default it loops back once).

5. The modified RLE assembly is expanded by estimating the repeat count of each base given the reads using a simple Bayesian model. The resulting final, polished assembly is output. In addition, a representation of the weighted POA can be output.

Innovations Compared to existing tools MarginPolish is most similar to Racon [163], in that they are comparable in speed, both principally use small-parameter HMM like models and both do not currently use signal information. Compared to Racon MarginPolish has some key innovations that we have found to improve polishing accuracy:

- MarginPolish, as with our earlier tool in the Margin series [37], uses the forward-backward and forward algorithms for pair hidden Markov models (HMMs) to sum over all possible pairwise alignments between pairs of sequences instead of the single most probable alignment (Viterbi). Considering all alignments allows more information to be extracted per read.

- The POA graph is constructed from a set of weights computed from the posterior alignment probabilities of each read to the initial assembled reference sequence (see below), the result is that MarginPolish POA construction does not have a read-order dependence. This is somewhat similar to that described by HGAP3

[26]. Most earlier algorithms for constructing POA graphs have a well known explicit read order dependence that can result in undesirable topologies [86].

- MarginPolish works in run-length encoded space, which results in considerably less alignment uncertainty and correspondingly improved performance.
- MarginPolish, similarly to Nanopolish [99], evaluates the likelihood of each alternative sequence introduced into the assembly. This improves performance relative to a faster but less accurate algorithm that traces back a consensus sequence through the POA graph.
- MarginPolish employs a simple chunking scheme to break up the polishing of the assembly into overlapping pieces. This results in low memory usage per core and simple parallelism.

Below steps 2, 3 and 5 of the MarginPolish algorithm are described in detail. In addition, the parallelization scheme is described.

Partial Order Alignment Graph Construction To create the POA we start with the existing assembled sequence $s = s_1, s_2, \ldots s_n$ and for each read $r = r_1, r_2, \ldots, r_m$ in the set of reads R use the Forward-Backward algorithm with a standard 3-state, affine-gap pair-HMM to derive posterior alignment probabilities using the implementation described in [122]. The parameters for this model are specified in the `polish.hmm` subtree of the JSON formatted parameters file, including `polish.hmm.transitions`, and `polish.hmm.emissions`. Current defaults were tuned via expectation maximization [69] of R9.4 ONT reads aligned to a bacterial reference; we have observed the parameters for this HMM seem robust to small changes in base-caller versions. The result of running the Forward-backward algorithm is three sets of posterior probabilities:

- Firstly *match probabilities*: the set of posterior match probabilities, each the probability $P(r_i \diamond s_j)$ that a read base r_i is aligned to a base s_j in s.

- Secondly *insertion probabilities*: the set of posterior insertion probabilities, each the probability $P(r_i \diamond -j)$ that a read base r_i is inserted between two bases s_j and s_{j+1} in s, or, if $j = 0$, inserted before the start of s, or, if $j = n$, after the end of s.

- Thirdly *deletion probabilities*, the set of posterior deletion probabilities, each the probability $P(-i \diamond s_j)$ that a base s_j in s is deleted between two read bases r_i and r_{i+1}. (Note, because a read is generally an incomplete observation of s we consider the probability that a base in s is deleted before the first position or after the last position of a read as 0).

As most probabilities in these three sets are very small and yet to store and compute all the probabilities would require evaluating comparatively large forward and backward alignment matrices we restrict the set of probabilities heuristically as follows:

- We use a banded forward-backward algorithm, as originally described here [123]. To do this we use the original alignment of the read to s as in the input BAM file. Given that s is generally much longer than each read this allows computation of each forward-backward invocation in time linearly proportional to the length of each read, at the cost of restricting the probability computation to a sub-portion of the overall matrix, albeit one that contains the vast majority of the probability mass.

- We only store posterior probabilities above a threshold (`polish.pairwiseAlignmentParameters.threshold`, by default 0.01), treating smaller probabilities as equivalent as zero.

The result is that these three sets of probabilities are a very sparse subset of the complete sets.

To estimate the posterior probability of a multi-base insertion of a read substring $r_i, r_{i+1}, \ldots r_k$ at a given location j in s involves repeated summation over terms in the forward and backward matrices. Instead to approximate this probability we heuristically use:

$$P(r_i, r_{i+1}, \ldots r_k \diamond -j) = \underset{l \in [i,k]}{\arg\min} P(r_l \diamond -j)$$

the minimum probability of any base in the multi-base insertion being individually inserted at the location in s as a proxy, a probability that is an upper-bound on the actual probability.

Similarly we estimate the posterior probability of a deletion involving more than one contiguous base s at a given location in a read using analogous logic. As we store a sparse subset of the single-base insertion and deletion probabilities and given these probability approximations it is easy to calculate all the multi-base indel probabilities with value greater than t by linear traversal of the single-based insertion and deletion probabilities after sorting them, respectively, by their read and s coordinates. The result of such calculation is expanded sets of insertion and deletion probabilities that include multi-base probabilities.

To build the POA we start from s, which we call the *backbone*. The backbone is a graph where each base s_j in s corresponds to a node, there are special source and sink nodes (which do not have a base label), and the directed edges connect the nodes for successive bases s_j, s_{j+1} in s, from the source node to the node for s_1, and, similarly, from the node for s_n to the sink node.

Each non-source/sink node in the backbone has a separate weight for each possible base $x \in \{A,C,G,T\}$. This weight:

$$w(j,x) = \sum_{r \in R} \sum_{i} \nvDash_x(r_i) P(r_i \diamond s_j)$$

where $\nvDash_x(r_i)$ is an indicator function that is 1 if $r_i = x$ and otherwise 0, corresponds to the sum of match probabilities of read elements of base x being aligned to s_j. This weight has a probabilistic interpretation: it is the total number of expected observations of the base x in the reads aligned to s_j, summing over all possible pairwise alignments of the reads to s. It can be fractional because of the inherent uncertainty of these alignments, e.g. we may predict only a 50% probability of observing such a base in a read.

We add *deletion edges*, which connect nodes in the backbone. Indexing the nodes in the backbone from 0 (the source) to the source $n+1$ (the sink), a deletion edge between positions j and k in the backbone corresponds to the deletion of bases

$j, j+1, \dots k-1$ in s. Each deletion edge has a weight equal to the sum of deletion probabilities for deletion events that delete the corresponding base(s) in s, summing over all possible deletion locations in all reads. Deletions with no weight are not included. Again, this weight has a probabilistic interpretation: it is the expected number of times we see the deletion in the reads, and again it may be fractional.

We represent insertions as nodes labelled with an insertion sequence. Each insertion node has a single incoming edge from a backbone node, and a single outgoing edge to the next backbone node in the backbone sequence. Each insertion is labeled with a weight equal to the sum of probabilities of events that insert the given insertion sequence between the corresponding bases in s. The resulting POA is a restricted form of a weighted, directed acyclic graph (Fig. 4.6(A) shows an example).

Frequently either an insertion or deletion can be made between different successive bases in s resulting in the same edited sequence. To ensure that such equivalent events are not represented multiple times in the POA, and to ensure we sum their weights correctly, we 'left shift' indels to their maximum extent. When shifting an indel results in multiple equivalent deletion edges or insertions we remove the duplicate elements, updating the weight of the residual element to include the sum of the weights of the removed elements. For example, the insertion of 'AT' in Fig. 4.6 is shifted left to its maximal extent, and could include the merger of an equivalent 'AT' insertion starting two backbone nodes to the right.

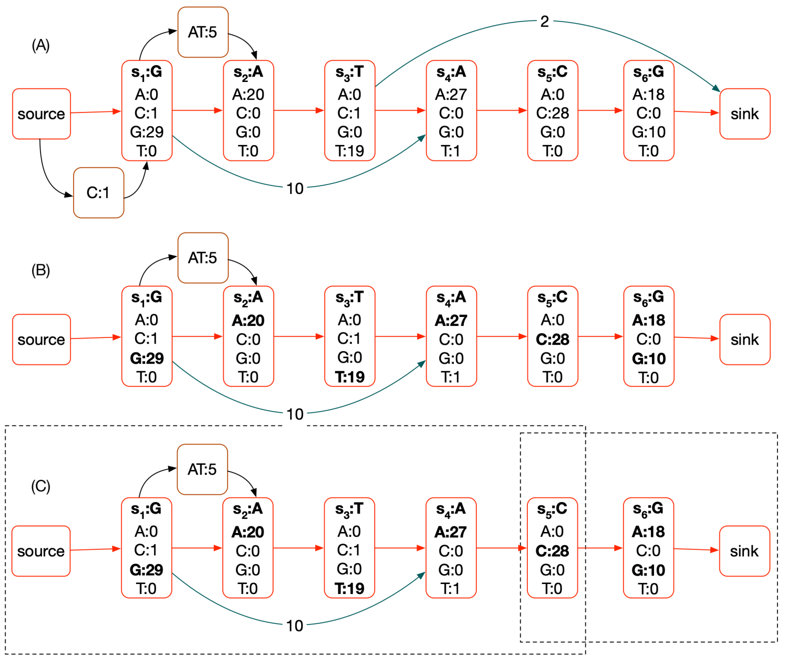


Figure 4.6: A) An example POA, assuming approximately 30x read coverage. The backbone is shown in red. Each non-source/sink node has a vector of weights, one for each possible base. Deletion edges are shown in teal, they also each have a weight. Finally insertion nodes are shown in brown, each also has a weight. (B) A pruned POA, removing deletions and insertions that have less than a threshold weight and highlighting plausible bases in bold. There are six plausible nucleotide sequences represented by paths through the POA and selections of plausible base labels: G;AT;A;T;A;C:A, G;AT;A;T;A;C:G, G;A;T;A;C:A, G;A;T;A;C:G, G;A;C:A, G;A;C:G. To avoid the combinatorial explosion of such enumeration we identify subgraphs (C) and locally enumerate the possible subsequences in these regions independently (dotted rectangles identify subgraphs selected). In each subgraph there is a source and sink node that does not overlap any proposed edit.

Local Haplotype Proposal After constructing the POA we use it to sample alternative assemblies. We first prune the POA to mark indels and base substitutions with weight below a threshold, which are generally the result of sequencing errors (Fig. 4.6(B)). Currently this threshold (`polish.candidateVariantWeight`=0.18, established empirically) is normalized as a fraction of the estimated coverage at the site, which is calculated in a running window around each node in the backbone of 100 bases. Consequently if fewer than 18% of the reads are expected to include the change then the edit is pruned from consideration.

To further avoid a combinatorial explosion we sample alternative assemblies locally. We identify subgraphs of s containing indels and substitutions to s then in each subgraph, defined by a start and end backbone vertex, we enumerate all possible paths between the start and end vertex and all plausible base substitutions from the backbone sequence. The rationale for heuristically doing this locally is that two subgraphs separated by one or more *anchor* backbone sites with no plausible edits are conditionally independent of each other given the corresponding interstitial anchoring substring of s and the substrings of the reads aligning to it. Currently, any backbone site more than `polish.columnAnchorTrim`=5 nodes (equivalent to bases) in the backbone from a node overlapping a plausible edit (either substitution or indel) is considered an anchor. This heuristic allows for some exploration of alignment uncertainty around a potential edit. Given the set of anchors computation proceeds by identifying successive pairs of anchors separated by subgraphs containing the potential edits, with the two anchors considered the source and sink vertex.

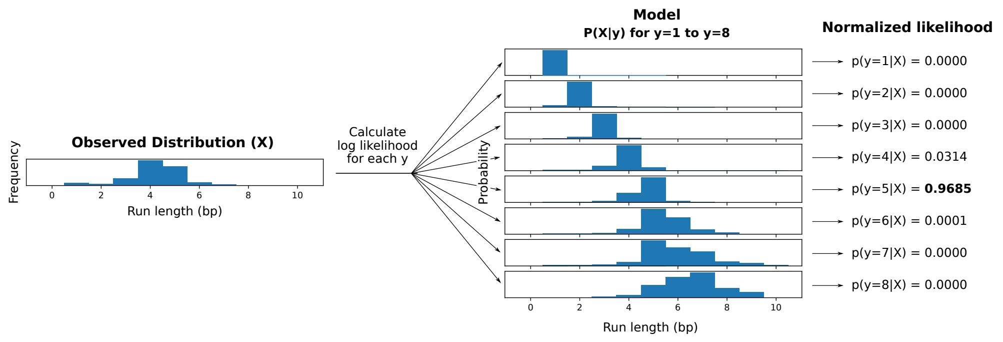


Figure 4.7: **Visual representation of run length inference.** This diagram shows how a consensus run length is inferred for a set of aligned lengths (X) that pertain to a single position. The lengths are factored and then iterated over, and log likelihood is calculated for every possible true length up to a predefined limit. Note that in this example, the most frequent observation (4bp) is not the most likely true length (5bp) given the model.

A Simple Bayesian Model for Run-length Decoding Run-length encoding allows for separate modelling of length and nucleotide error profiles. In particular, length predictions are notoriously error prone in nanopore basecalling. Since homopolymers produce continuous signals, and DNA translocates at a variable rate through the pore, the basecaller often fails to infer the true number of bases given a single sample. For this reason, a Bayesian model is used for error correction in the length domain, given a distribution of repeated samples at a locus.

To model the error profile, a suitable reference sequence is selected as the truth set. Reads and reference are run-length encoded and aligned by their nucleotides. The alignment is used to generate a mapping of observed lengths to their true length (y,x) where $y = true$ and $x = observed$ for each position in the alignment. Observations from alignment are tracked using a matrix of predefined size $(y_{max} = 50, x_{max} = 50)$ in which each coordinate contains the corresponding count for (y,x). Finally the matrix is normalized along one axis to generate a probability distribution of $P(X|y_j)$ for j in $[1, y_{max}]$. This process is performed for each of the 4 bases.

With enough observations, the model can be used to find the most probable true run length given a vector of observed lengths X. This is done using a simple log likelihood calculation over the observations x_i for all possible true lengths y_j in Y, assuming the length observations to be independent and identically distributed. The length y_j corresponding to the greatest likelihood $P(X|y_j, Base)$ is chosen as the consensus length for each alignment position (Fig. 4.7).

4.7.2.1 Training

To generate a model, we ran MarginPolish with reads from a specific basecaller version aligned to a reference (GRCh38) and specified the `--outputRepeatCounts` flag. This option produces a TSV for each chunk describing all the observed repeat counts aligned to each backbone node in the POA. These files are consumed by a script in the `https://github.com/rlorigro/runlength_analysis` repository, which generates a RLE consensus sequence, aligns to the reference, and performs the described process to produce the model.

The `allParams.np.human.guppy-ff-235.json` model used for most of the

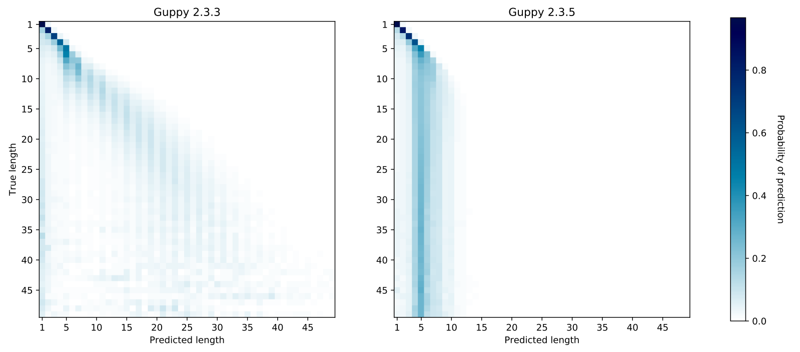


Figure 4.8: Run-length confusion in different versions of Guppy base caller

analysis was generated from HG00733 reads basecalled with Guppy Flipflop v2.3.5 aligned to GRCh38, with chromosomes 1, 2, 3, 4, 5, 6, and 12 selected. The model `allParams.np.human.guppy-ff-233.json` was generated from Guppy Flipflop v2.3.3 data and chromosomes 1-10 were used. This model was also used for the CHM13 analysis, as the run-length error profile is very similar between v2.3.3 and v2.3.1 (v2.3.5 has a drastically different error profile, as is shown below in Fig. 4.8).

Parallelization and Computational Considerations To parallelize MarginPolish we break the assembly up into chunks of size `polish.chunkSize`=1000 bases, with an overlap of `polish.chunkBoundary`=50 bases. We then run the MarginPolish algorithm on each chunk independently and in parallel, stitching together the resulting chunks after finding an optimal pairwise alignment (using the default hmm described earlier) of the overlaps that we use to remove the duplication. We can further parallelize the algorithm across machines or processes using a provided Toil script.

Memory usage scales with thread count, read depth, and chunk size. For this reason, we downsample reads in a chunk to `polish.maxDepth`=50× coverage by counting total nucleotides in the chunk N_c and discarding reads with likelihood $1 - (\texttt{chunkSize} + 2 * \texttt{chunkBoundary}) * \texttt{maxDepth}/N_c$. With these parameters, we find that 2GB of memory per thread is sufficient to run MarginPolish on genome-scale assemblies. Across 13 whole-genome runs, we averaged roughly 350 CPU hours per gigabase of assembled sequence.

4.7.3 HELEN: Homopolymer Encoded Long-read Error-corrector for Nanopore

HELEN is a deep neural network based haploid consensus sequence polisher. HELEN employs a multi-task recurrent neural network (RNN) [107] that takes the weights of the partial order alignment (POA) graph of MarginPolish to predict a base and a run-length for each genomic position. MarginPolish constructs the POA graph by performing multiple possible alignments of a single read that makes the weights associative to the correct underlying base and run-length. The RNN employed in HELEN takes advantage of the transitive relationship of the genomic sequence and associative coupling of the POA weights to the correct base and run-length to produce a consensus sequence with higher accuracy.

The error-correction with HELEN is done in three steps. First, we generate tensor-like images of genomic segments with MarginPolish that encodes POA graph weights for each genomic position. Then we use a trained RNN model to produce predicted bases and run-lengths for each of the generated images. Finally, we stitch the chunked sequences to get a contiguous polished sequence.

4.7.3.1 Image Generation

MarginPolish produces an image-like summary of the final POA state for use by HELEN. At a high level, the image summarizes the weighted alignment likelihoods of all reads divided into nucleotide, orientation, and run-length.

The positions of the POA nodes are recorded using three coordinates: the position in the backbone sequence of the POA, the position in the insert sequences between backbone nodes, and the index of the run-length block. All backbone positions have an insert coordinate of 0. Each backbone and insert coordinate includes one or more run-length coordinate.

When encoding a run-length, we divide all read observations into blocks from 0 to 10 inclusive (this length is configurable). For cases where no observations exceed the maximum run-length, a single run-length image can describe the POA node. When an observed run-length exceeds the length of the block, the run-length is encoded as that block's maximum (10), and the remaining run-length is encoded in successive blocks. For a run-length that terminates in a block, its weight is contributed to the run-length 0 column in all successive blocks. This means that the records for all run-length blocks of a given backbone and insert position have the same total weight. As an example, consider three read positions aligned to a node with run-lengths of 8, 10, and 12. These require two run-length blocks to describe: the first block includes one 8 and two 10s, and the second includes two 0s and one 2.

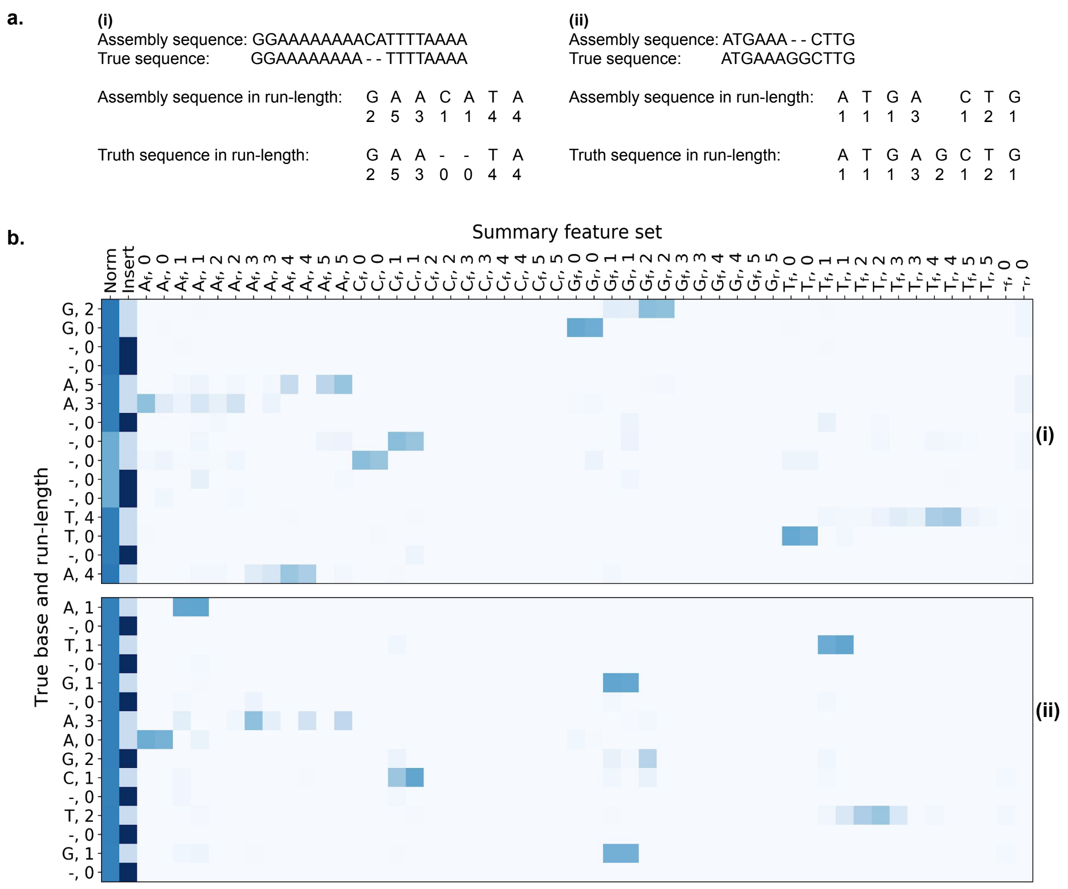


Figure 4.9: **MarginPolish Images** A graphical representation of images from two labeled regions selected to demonstrate: the encoding of a single POA node into two run-length blocks (i), a true deletion (i), and a true insert (ii). The y-axis shows truth labels for nucleotides and run-lengths, the x-axis describes features in the images, and colors show associated weights.

The information described at each position (backbone, insert, and run-length) is encoded in 92 features: each nucleotide {A, C, T, G} and run-length {0, 1, .., 10}, plus a gap weight (for deletions in read alignments). The weights for each of these 45 observations are separated into forward and reverse strand for a total of 90 features. The weights for each of these features are normalized over the total weight for the record and accompanied by an additional data point describing the total weight of the record. This normalization column for the record is an approximation of the read depth aligned to that node. Insert nodes are annotated with a binary feature (for a final total of 92); weights for an insert node's alignments are normalized over total weight at the backbone node it is rooted at (not the weight of the insert node itself) and gap alignment weights are not applied to them.

Labeling nodes for training requires a truth sequence aligned to the assembly reference. This provides a genome-scale location for the true sequence and allows the its length to help in the resolution of segmental duplications or repetitive regions. When a region of the assembly is analyzed with MarginPolish, the truth sequences aligned to that region are extracted. If there is not a single truth sequence which approximately matches the length of the consensus for this region, we treat it as an uncertain region and no training images are produced. Having identified a suitable truth sequence, it is aligned to the final consensus sequence in non-run-length space with Smith-Waterman. Both sequences and the alignment are then run-length encoded, and true labels are matched with locations in the images. All data between the first and last matched nodes are used in the final training images (leading and trailing inserts or deletes are discarded). For our training, we aligned the truth sequences with `minimap2` using the `asm20` preset and filtered the alignments to include only primary and supplementary alignments (no secondary alignments).

Fig. 4.9 shows a graphical representation of the images. On the y-axis we display true nucleotide labels (with the dash representing no alignment / gap) and true run-length. On the x-axis the features used as input to HELEN are displayed: first the normalization column (the total weight at the backbone position), second the insert column (the binary feature encoding whether the image is for a backbone or insert node), forty-eight columns describing the weights associated with read observations (stratified by nucleotide, run-length, strand), and two columns describing weights for gaps in read alignments (stratified by strand). In this example, we have reduced the maximum run-length per block from 10 to 5 for demonstrative purposes.

We selected these two images to highlight three features of the model: the way multiple run-length blocks are used to encode observations for a single node, and the relevant features around a true gap and a true insert that enable HELEN to correct these errors.

To illustrate multiple run length blocks, we highlight two locations on on image (i). The first are the nodes labeled (A,5) and (A,3). This is the labeling for a true (A,8) sequence separated into two blocks. See that the bulk of the weight is on the (A,5) features on the first block, with most of that distributed across the (A,1-3) features on the second. Second, observe the nodes on (i) labeled (T,4) and (T,0). Here we show the true labeling of a (T,4) sequence where there are some read observations extending into a second run-length block.

To show a features of a true gap, note on (i) the non-insert nodes labeled (-,0). We know that MarginPolish predicted a single cytosine nucleotide (as it is a backbone node and the (C,1) nodes have the bulk of the weight. Here, HELEN is able to use the low overall weight (the lighter region in the normalization column) at this location as evidence of fewer supporting read alignments and can correct the call.

The position labeled (G,2) on (ii) details a true insertion. It is not detected by MarginPolish (as all insert nodes are not included in the final consensus sequence). Read support is present for the insert, less than the backbone nodes in this image but more than the other insert nodes. HELEN can identify this sequence and correct it.

Finally, we note that the length of the run length blocks results in streaks at multiples of this length (10) for long homopolymers. The root of this effect lies in the basecaller producing similar prediction distributions for these cases (ie, the run length predictions made by the basecaller for a true run length of 25 are similar to the run length predictions made for a true run length of 35, see Fig. 4.4b Guppy 2.3.3). This gives the model little information to differentiate upon, and the issue is exacerbated by the low occurrence of long run lengths in the training data. Because the model divides run length observations into chunks of size 10, it tends to call the first chunks correctly (having length 10) but has very low signal for the last chunk and most often predicts 0.

Part IV

A State-of-the-Art Variant Caller

Chapter 5

Read and variant phasing in a state-of-the-art variant calling toolchain

5.1 Preamble

What follows is the full text from the Nature Methods publication "Haplotype-aware variant calling with PEPPER-Margin-DeepVariant enables high accuracy in nanopore long-reads" [149], in which I share first-authorship with Kishwar Shafin, and Pi-Chuan Chang. This paper describes and evaluates a variant calling pipeline which depends on margin to efficiently phase reads.

For this publication, I improved and streamlined margin including functional improvements to the codebase as well as considerable fine-tuning of parameters for different sequence data and desired output. Kishwar and I wrote the majority of the manuscript together.

This chapter is the culmination of years of work building a tool which can partition reads into maternal and paternal haplotypes. Accurate phasing is a critical part of this pipeline, providing the context needed to solve difficult edge cases as well as identifying co-inheritied variants.

I am incredibly proud to have contributed to the work presented in this chapter.

5.2 Abstract

Long-read sequencing has the potential to transform variant detection by reaching currently difficult-to-map regions and routinely linking together adjacent variations to enable read based phasing. Third-generation nanopore sequence data has demonstrated a long read length, but current interpretation methods for its novel pore-based signal have unique error profiles, making accurate analysis challenging. Here, we introduce a haplotype-aware variant calling pipeline PEPPER-Margin-DeepVariant that produces state-of-the-art variant calling results with nanopore data. We show that our nanopore-based method outperforms the short-read-based single nucleotide variant identification method at the whole genome-scale and produces high-quality single nucleotide variants in segmental duplications and low-mappability regions where short-read based genotyping fails. We show that our pipeline can provide highly-contiguous phase blocks across the genome with nanopore reads, contiguously spanning between 85% to 92% of annotated genes across six samples. We also extend PEPPER-Margin-DeepVariant to PacBio HiFi data, providing an efficient solution with superior performance than the

current WhatsHap-DeepVariant standard. Finally, we demonstrate *de novo* assembly polishing methods that use nanopore and PacBio HiFi reads to produce diploid assemblies with high accuracy (Q35+ nanopore-polished and Q40+ PacBio-HiFi-polished).

5.3 Introduction

Most existing reference-based small variant genotyping methods are tuned to work with short-reads [31, 106]. Short-reads have high base-level accuracy but frequently fail to align unambiguously in repetitive regions [97]. Short-reads are also generally unable to provide substantial read-based phasing information, and therefore require using haplotype panels for phasing [22] that provide limited phasing information for rarer variants.

Third-generation sequencing technologies, like linked-reads [12, 45, 169] and long-reads [69, 42], produce sequences that can map more confidently in the repetitive regions of the genome [68], overcoming the fundamental limitations of short-reads. Long-reads can generate highly contiguous *de novo* assemblies [109, 98, 148, 24, 118, 74, 141], and they are increasingly being used by reference-based analysis methods [100, 103, 40, 171, 37, 67, 146, 124]. The Genome-In-A-Bottle Consortium (GIAB) [177] used the additional power of long-reads and linked-reads to expand the small variant benchmarking set to cover more of the genome [166]. This was essential to the PrecisionFDA challenge V2, which quantified the limitations of short read-based methods to accurately identify small variants in repetitive regions[121].

Oxford Nanopore Technologies (ONT) is a commercial nanopore-based high-throughput [148] long-read sequencing platform that can generate 100kb+ long reads [148, 70]. Nanopore long-reads can confidently map to repetitive regions of the genome [68] including centromeric satellites, acrocentric short arms, and segmental duplications [109, 71, 47, 41]. Nanopore sequencing platform promises same-day sequencing and analysis [43], but the base-level error characteristics of the nanopore-reads, being both generally higher and systematic, make small variant identification challenging [130].

Pacific Biosciences (PacBio) provides a single-molecule real-time (SMRT) sequencing platform that employs circular consensus sequencing (CCS) to generate highly-accurate (99.8%) high-fidelity (PacBio HiFi) reads that are between 15kb-20kb long [171]. The overall accuracy of PacBio-HiFi-based variant identification is competitive with short-read based methods [171]. These highly accurate long-reads enabled the small variant benchmarking of major histocompatibility complex (MHC) region [28] and difficult-to-map regions [166].

In our previous work, we introduced DeepVariant, a universal small variant calling method based on a deep convolutional neural network (CNN) [126]. We showed that by retraining the neural network of DeepVariant, we can generate highly accurate variant calls for various sequencing platforms [126]. To limit the computational time, DeepVariant only uses the neural network on candidate sites identified with simple heuristics. However, the higher error-rate of nanopore-reads [148, 130] causes too many candidate variants to be picked up by the heuristic-based candidate finder of DeepVariant, limiting the extension of DeepVariant to nanopore sequencing platform.

Phasing long reads has been shown to enable or improve methods for small variant calling, structural variant calling, and genome assembly [22, 171, 67, 124, 39, 136, 27, 76, 127]. Previously, we trained DeepVariant on PacBio HiFi long-read data and showed highly competitive performance against short-read based methods for small variant identification [171]. However, the run-time of the haplotype-aware mode of DeepVariant with PacBio HiFi reads remain a bottleneck for production-level scaling.

Sufficiently accurate nanopore long-read based accurate small variant identification would enable new research. It could allow same-day sequencing and variant calling by using highly multiplexed sequencing with the PromethION device. It could allow researchers to study genomic variants in the most difficult regions of the genome. Similarly, making PacBio HiFi haplotype-aware genotyping efficient would allow researchers to adopt to production scale haplotype-aware genotyping.

Here we present a haplotype-aware genotyping pipeline PEPPER-Margin-DeepVariant that produces state-of-the art small variant identification results with nanopore and PacBio HiFi long-reads. PEPPER-Margin-DeepVariant outperforms other existing nanopore-based variant callers like Medaka[100], Clair [103], and longshot [40]. For the first time we report that nanopore-based single nucleotide polymorphism (SNP) identification with PEPPER-Margin-DeepVariant outperforms short-read based SNP identification with DeepVariant at whole genome scale. For PacBio HiFi reads, we report PEPPER-Margin-DeepVariant is more accurate, 3× faster, and 1.4× cheaper than the current haplotype-aware pipeline DeepVariant-WhatsHap-DeepVariant. We analyzed our pipeline in the context of GENCODE[61] genes and report phasing errors in less than 1.5% of genes, with over 88% of all genes being contiguously phased across six samples. Finally, we extended PEPPER-Margin-DeepVariant to polish nanopore-based *de novo* assemblies with nanopore and PacBio HiFi reads in a diploid manner. We report Q35+ nanopore-based and Q40+ PacBio-HiFi-polished assemblies with lower switch error rate compared to the unpolished assemblies.

5.4 Results

5.4.1 Haplotype-aware variant calling

PEPPER-Margin-DeepVariant is a haplotype-aware pipeline for identifying small variants against a reference genome with long-reads. The pipeline employs several methods to generate highly-accurate variant calls (Figure 5.1a). Details of these methods are in the online methods section. An overview is presented here:

1. **PEPPER-SNP**: PEPPER-SNP finds single nucleotide polymorphisms (SNPs) from the read alignments to the reference using a recurrent neural network (RNN). The method works in three steps:

    - *Image generation*: We take the reads aligned to a reference genome and generate base-level summary statistics in a matrix-like format for each location of the genome. We do not encode insertions observed in reads at this stage.
    - *Inference*: We use a gated recurrent unit (GRU)-based RNN that takes the base-level statistics generated in the previous step and the provides likelihood of the two most likely bases present at each genomic location.

- *Find candidates*: We take all of the base-mismatches observed in the reads-to-reference alignment. We calculate a likelihood using the predictions from the inference step and if a base-mismatch has high likelihood of being a potential variant, we pick the mismatch to be a potential SNP. The likelihood of the bases at any location also helps to assign a genotype.

2. **Margin**: Margin is a phasing and haplotyping method that takes the SNPs reported by PEPPER-SNP and generates a haplotagged alignment file using a hidden Markov Model (HMM).

    - *Read-allele alignment*: We first extract read substrings around allelic sites and generate alignment likelihoods between reads and alleles. These are used as emission probabilities in the phasing HMM.

    - *Phasing Variants*: We construct an HMM describing genotypes and read bipartitions at each variant site which enforces consistent partitioning between sites. After running the forward-backward algorithm, we marginalize over the posterior probability distribution at each site to calculate the most likely phased genotype (aka diplotype).

    - *Haplotagging reads*: After determining haplotypes using the maximum probability haplotype decoding, we decide from which haplotype each read originated from by calculating the probability of the read arising from each of the two haplotypes and picking the haplotype with maximum likelihood. If a read spans no variants or has equal likelihood for each haplotype, it is assigned the a “not haplotagged” tag.

- *Chunking*: The genome is broken up into 120kb chunks with 20kb of overlap between chunks. Variant and read phasing occurs separately on each chunk, enabling a high degree of parallelism. Chunks are stitched together using the haplotype assignment of the reads shared between adjacent chunks.

3. **PEPPER-HP**: PEPPER-HP takes the haplotagged alignment file and finds potential SNP, insertion, and deletion (INDEL) candidate variants using a recurrent neural network (RNN). In this step PEPPER-HP ranks all variants arising from the read-to-reference alignment and picks variants with high-likelihood derived from the RNN output. Filtering candidates enables DeepVariant to efficiently genotype the candidates and produce a highly accurate variant set as it removes errors. PEPPER-HP is used only during Oxford Nanopore-based variant calling and has proved unnecessary while using PacBio HiFi reads.

   - *Image generation*: We generate base-level summary statistics for each haplotype independently. Summary statistics for each haplotype use both reads that were haplotagged to that haplotype and which were not haplotagged. In this scheme, we encode insertions observed in the reads.

   - *Inference*: We use a GRU-based RNN that takes the haplotype-specific summary statistics and predicts two bases at each location of the genome, one for each haplotype. This haplotype-aware inference scheme allows us to determine most likely alleles in a haplotype-specific manner.

- *Find candidates*: In the find candidates step we find all SNP and INDEL candidates arising from the read alignment to the reference. We use the haplotype-specific predictions from the inference step to generate the likelihood of each candidate variant belonging to haplotype-1 or haplotype-2. Using the likelihood values we propose candidates with high likelihood for genotyping with DeepVariant.

4. **DeepVariant**: DeepVariant identifies variants in a three step process:

   - *Make examples*: Prior to this work [126], the *make examples* stage of DeepVariant used simple heuristics to identify possible variant positions for classification. The different error profile of Oxford Nanopore required the more sophisticated logic from PEPPER to generate a tractable number of candidates for classification. DeepVariant was modified to take the candidate variant set from PEPPER-HP and the haplotagged alignment from Margin, and to generate the tensor input set with read features as channels (base, base quality, mapping quality, strand, whether a read supports the variant, and the bases that mismatch the reference). Reads are sorted by their haplotype tag.
   - *Call variants*: This stage applies a model trained specifically for Oxford Nanopore data with inputs provided by PEPPER-Margin. Apart from training on new data, and the sorting of reads by haplotype, other software components of this step are unchanged.

- *Postprocess variants*: Converts the output probability into a VCF call and resolves multi-allelic cases. No other changes were made from previously published descriptions.

DeepVariant has more total parameters than PEPPER, and models with more parameters can generally train to higher accuracy at the cost of increased runtime. By combining PEPPER with DeepVariant in this way, we allow the faster neural network of PEPPER to efficiently scan much more of the genome, and to leverage the larger neural network of DeepVariant to achieve high accuracy on a tractable number of candidates.

5. **Margin**: Margin takes the output of DeepVariant and the alignment file to generate a phased VCF file using the same Hidden Markov Model as described before. In this mode, it annotates the VCF with high-confidence phasesets using heuristics over the reads assigned to each variant's haplotype. It creates a new phaseset if there is no linkage between adjacent sites, if there is an unlikely binomial p-value for the bipartition of reads at a site, or if there is high discordancy between read assignments over adjacent variants.

It is challenging to identify accurate variants with Oxford nanopore reads due to the error rate. Heuristic-based approaches show robust solutions for highly-accurate sequencing platforms [126, 171] but fail when introduced with erroneous reads. For example, in 90x HG003 ONT data, at 10% allele frequency, we find 20$\times$ more erroneous variants than true variants (Supplementary Figure B.2).

Existing variant callers, like Clair[103], that use allele frequency to find a set of candidates often need to set the threshold too high, excluding many true variants from being detected. Our pipeline demonstrates an efficient solution using an RNN to find candidates with PEPPER and genotype the candidates accurately with DeepVariant.

The use of haplotype information to get better genotyping results with erroneous reads has been demonstrated before[37, 171]. The schema of the PEPPER-Margin-DeepVariant pipeline follows a similar design of PacBio HiFi-based DeepVariant[171] and Medaka[100] that use haplotyping to provide better genotyping results. However, Medaka[100] is a consensus caller that presents the predicted sequence per position that does not match with reference sequence as variants. In contrast, PEPPER applies the predictions of the RNN to the candidates to find likely candidate variants for DeepVariant to accurately genotype. While maintaining similar candidate sensitivity of the heuristic-based approach, PEPPER reduces the number of erroneous homozygous candidate variants significantly (Supplementary figure B.2).

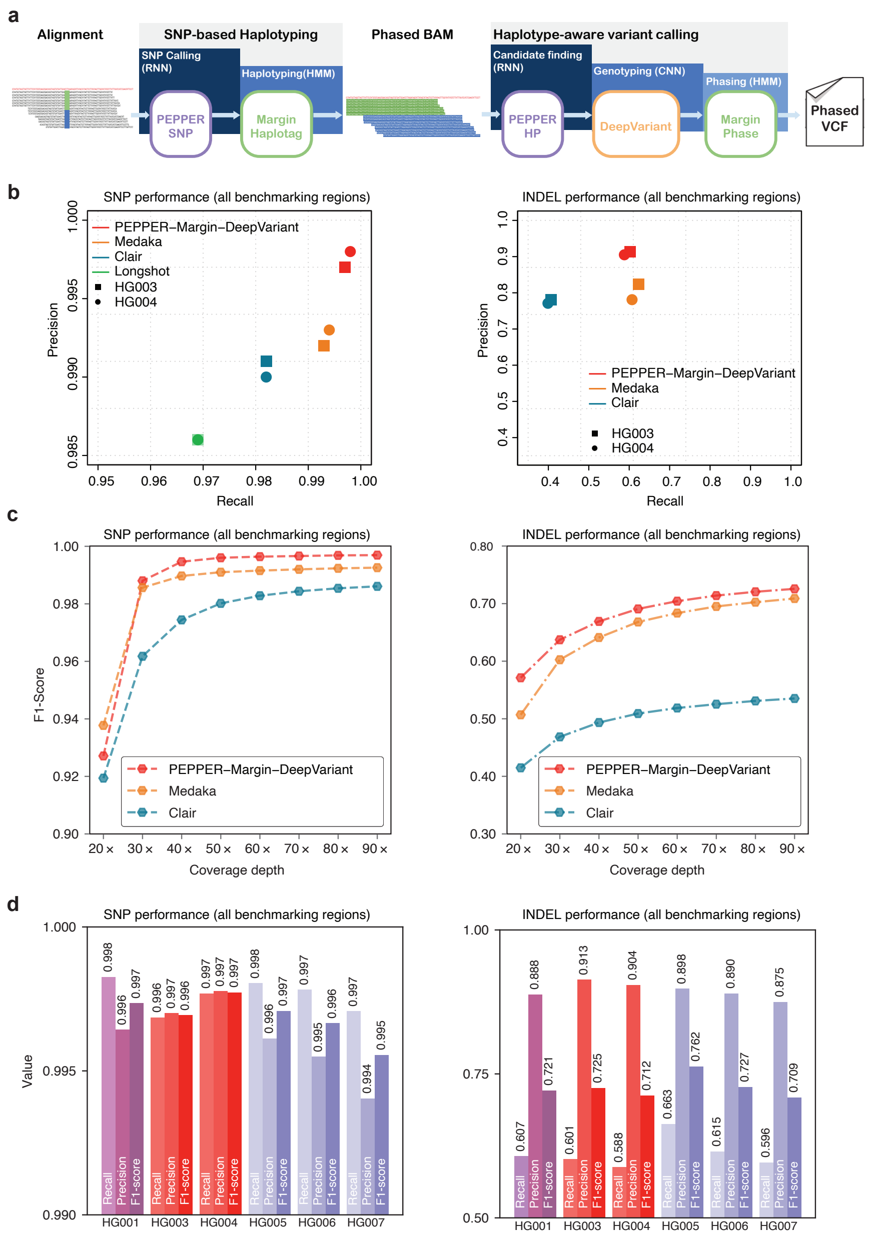


Figure 5.1: **Nanopore variant calling results.** **(a)** Illustration of haplotype-aware variant calling using PEPPER-Margin-DeepVariant. **(b)** Nanopore variant calling performance comparison between different nanopore-based variant callers. **(c)** Evaluating variant calling performance at different coverage of HG003. **(d)** Variant calling performance of PEPPER-Margin-DeepVariant on six GIAB samples.

We trained PEPPER-Margin-DeepVariant on HG002 sample using Genome-In-A-Bottle (GIAB) v4.2.1 benchmarking set [166]. We trained PEPPER and DeepVariant on `chr1-chr19` and tested on `chr20` and used `chr21-chr22` as holdout sets (See Online Methods).

5.4.2 Nanopore variant calling performance

We compared the nanopore variant calling performance of PEPPER-Margin-DeepVariant against Medaka[100], Clair[103], and Longshot[40]. We called variants on two samples HG003 and HG004, with 90× coverage. We also compared the performance against Medaka and Clair for the HG003 sample at various coverages ranging from 20× to 90×. Finally, we benchmarked the variant calling performance of PEPPER-Margin-DeepVariant on six Genome-In-A-Bottle (GIAB) samples.

PEPPER-Margin-DeepVariant produces more accurate nanopore-based SNP calls (F1-scores of 0.9969 and 0.9977) for HG003 and HG004 respectively than Medaka (0.9926, 0.9933), Clair (0.9861, 0.9860), and Longshot (0.9775, 0.9776). We also observe higher INDEL performance with PEPPER-Margin-DeepVariant (F1-scores of 0.7257 and 0.7128 for HG003 and HG004) compared to Medaka (0.7089, 0.7128) and Clair (0.5352, 0.5260)(Figure 5.1b, Supplementary table B.1).

To assess the robustness of our method, we evaluated the variant calling performance with HG005 sample on GRCh38 and GRCh37 against two GIAB truth versions (v3.3.2 [181] and v4.2.1 [166]). In this comparison, we see PEPPER-Margin-DeepVariant perform similarly between GRCh37 GIABv3.3.2 (SNP F1-Score: 0.9971, INDEL F1-Score: 0.7629) and GRCh38 v4.2.1 (SNP F1-Score: 0.9974, INDEL F1-Score: 0.7678) and has higher accuracy compared to Medaka (GRCH37: SNP 0.9938, INDEL 0.7629, GRCh38: SNP 0.9927, INDEL 0.7406), Clair (GRCH37: SNP 0.9789, INDEL 0.5666, GRCh38: SNP 0.9787, INDEL 0.5675) and longshot (GRCh37: SNP 0.9803, GRCh38: SNP 0.9767) (Supplementary table B.2). Overall, we see PEPPER-Margin-DeepVariant has consistent performance between different samples, reference sequence and truth sets.

We performed a Mendelian concordance analysis of our method with the HG005/HG006/HG007 trio on GRCh38 inside and outside of the HG005 v4.2.1 high-confidence regions (Supplementary table B.3). In the 2.5 Gb high confidence region we observed a paternal and maternal concordance of 99.90%, with overall concordance of 99.75%. In the 315Mb region outside of high confidence excluding centromeres we observed a paternal concordance of 98.20%, maternal concordance of 97.80%, with overall concordance of 95.52%.

To understand performance over realistic coverage ranges, we downsampled the HG003 nanopore sample at coverages varying between 20× and 90× and compared PEPPER Margin-DeepVariant against Medaka and Clair. The INDEL performance of PEPPER-Margin-DeepVariant achieves the highest F1-score at any coverage compared to other tools (Figure 5.1c, Supplementary table B.4). At coverage above 30×, PEPPER-Margin-DeepVariant achieves a higher F1-score than Medaka and Clair (Supplementary table B.5). Overall, we observe that PEPPER-Margin-DeepVariant can yield high-quality variant calls at above 40× coverage on Oxford Nanopore data.

We investigated the nanopore variant calling performance of PEPPER-Margin-DeepVariant on six GIAB samples (HG001, HG003-HG007), each sample with various coverage (Supplementary Table B.6) and against GRCh37 and GRCh38 reference genomes (Supplementary Table B.7). PEPPER-Margin-DeepVariant achieves SNP F1-score 0.995 or higher and INDEL F1-score of 0.709 or higher for each sample, demonstrating the ability to generalize the variant calling across samples and reference genomes (Figure 5.1d, Supplementary table B.8).

We also assessed the ability to use PEPPER-HP as a variant caller if we tune the method for a balanced precision and recall. We find that PEPPER-HP outperforms Medaka in SNP accuracy while having a comparable INDEL accuracy. However, PEPPER-HP in itself is not able to achieve the genotyping accuracy DeepVariant provides. As PEPPER-HP uses a compressed representation of nucleotide bases, it fails to achieve the high genotyping accuracy compared to DeepVariant's CNN (Supplementary table B.9).

Similar to the nanopore-based haplotype-aware pipeline, the PacBio HiFi-based PEPPER-Margin-DeepVariant pipeline produces highly accurate variant calls. In the PacBio HiFi pipeline, we do not use PEPPER-HP to find candidate variants; the highly accurate (99.8%) PacBio HiFi reads are suitable for the heuristic-based candidate generation approach of DeepVariant [171, 126]. We analyzed the PacBio HiFi PEPPER-Margin-DeepVariant variant calling performance on the 35x HG003 and HG004 from precisionFDA [121] against DeepVariant-WhatsHap-DeepVariant (current state-of-the-art method [121, 171]) and DeepVariant-Margin-DeepVariant. In this comparison we see that DeepVariant-Margin-DeepVariant produces the best performance (HG003 SNP-F1: 0.9991 INDEL-F1: 0.9945 , HG004 SNP-F1: 0.9992, INDEL-F1: 0.9942) compared to DeepVariant-WhatsHap-DeepVariant (HG003 SNP-F1:0.9990 INDEL-F1:0.9942 , HG004 SNP-F1: 0.9992 , INDEL-F1: 0.9940) and PEPPER-Margin-DeepVariant (HG003 SNP-F1:0.9990, INDEL-F1:0.9944, HG004 SNP-F1: 0.9992 , INDEL-F1: 0.9941) (Supplementary table B.10).

We compared the run-time and cost of Oxford nanopore-based variant calling pipelines on 50× and 75× HG001 data (Supplementary table B.11) using the GCP platform with instance sizes best matching CPU and memory requirements. Clair (HG001-50×: 2.5h/$11.40, HG001-75×: 3.1h/$14.13) is the fastest and cheapest, but fails to generate high-quality variant calls (Figure 5.1, Supplementary Table B.1). Longshot (HG001-50×: 51h/$49, HG001-75×: 74h/$139) and Medaka (CPU-HG001-50×: 95h/$90, CPU-HG001-75×: 117h/$175, GPU-HG001-50×: 40h/$97, GPU-HG001-75×: 46h/$22) fail to use all available CPU resources, resulting in long runtimes. PEPPER-Margin-DeepVariant is designed for CPU and GPU platforms. On a CPU-platform, PEPPER-Margin-DeepVariant (HG001-50×: 13h/$60, HG001-75×: 15h/$68) is 8× faster than Medaka and 4× faster than Longshot while providing the best variant calling performance. On GPU-platforms we see further runtime improvement with PEPPER-Margin-DeepVariant (HG001-50×: 7h/$70, HG001-75×: 9h/$94). On PacBio HiFi data the PEPPER-Margin-DeepVariant pipeline outperforms DeepVariant-WhatsHap-DeepVariant and is 3× faster and 1.4× cheaper, establishing a faster and more accurate solution to haplotype-aware variant calling with PacBio HiFi data (Supplementary table B.12, Supplementary table B.13). Overall, PEPPER-Margin-DeepVariant provides a scalable solution to haplotype-aware variant calling with nanopore-based long reads, as it is designed to efficiently use all available resources.

5.4.3 Nanopore, Illumina and PacBio HiFi variant calling performance comparison

We compared the variant calling performance of Oxford Nanopore and PacBio HiFi long-read based PEPPER-Margin-DeepVariant against Illumina short-read based DeepVariant method [9]. We used 35x Illumina NovaSeq, 35x PacBio HiFi, and 90x Oxford Nanopore reads basecalled with Guppy v4.2.2 for HG003 and HG004 samples available from PrecisionFDA [121]. We used GIAB v4.2.1 benchmarking data for HG003 and HG004, which is notable for including difficult-to-map regions. Finally we used GIAB v2.0 stratificiations to compare variant calling performance in difficult-to-map regions and low-complexity regions of the genome.

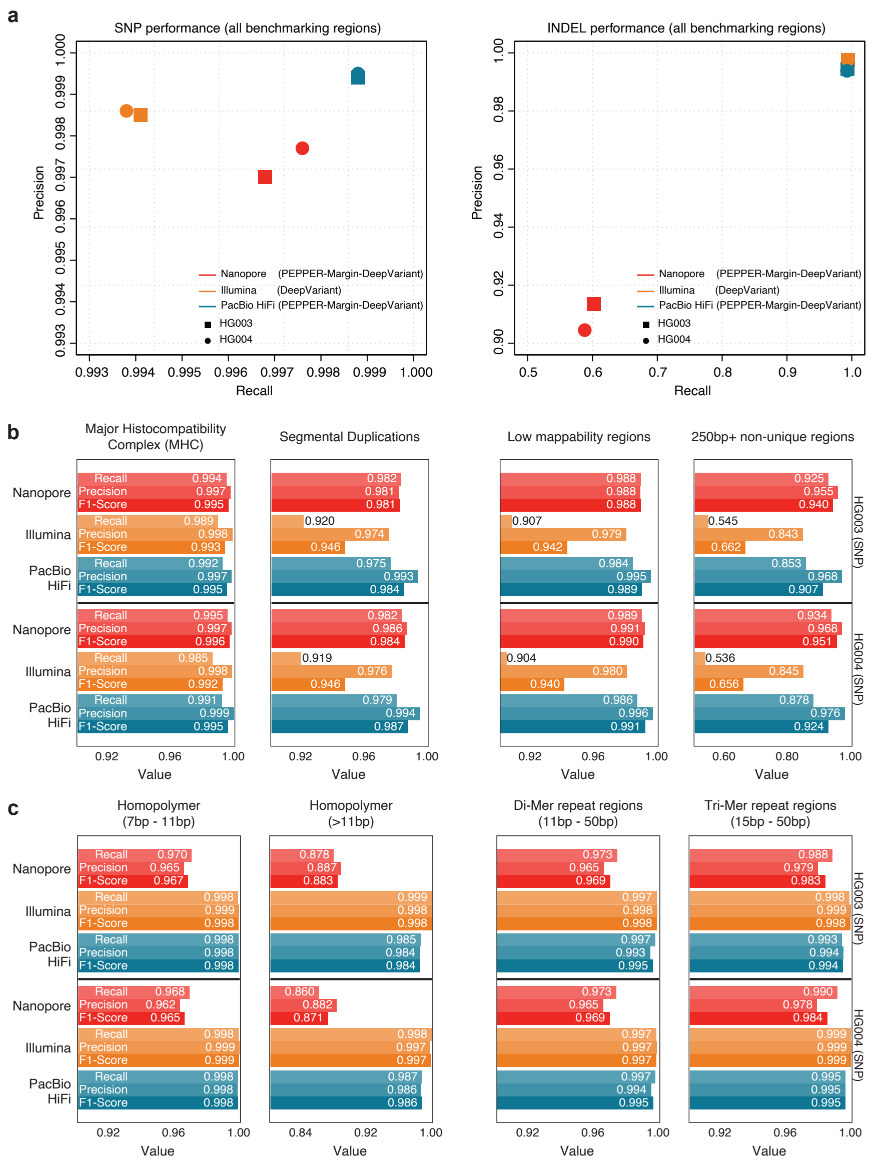


Figure 5.2: **Comparison between Nanopore, Illumina and PacBio HiFi variant calling performance.** **(a)** SNP and INDEL performance comparison of Nanopore, Illumina and PacBio HiFi in all benchmarking regions. **(b)** SNP performance comparison in difficult-to-map regions of the genome. **(c)** SNP performance comparison in low-complexity regions of the genome.

The SNP F1-score of PacBio HiFi (HG003 SNP-F1: 0.9990, HG004 SNP-F1: 0.9992) is higher than Oxford Nanopore (HG003 SNP-F1: 0.9969, HG004 SNP-F1: 0.9977) and Illumina (HG003 SNP-F1: 0.9963, HG004 SNP-F1: 0.9962) in all benchmarking regions. Notably, both long-read sequencing platforms outperform the short-read based method in accurate SNP identification performance. The INDEL F1-score of Oxford Nanopore (HG003 INDEL-F1: 0.7257, HG004 INDEL-F1: 0.7128) is well below the performance with PacBio HiFi (HG003 INDEL-F1: 0.9945, HG004 INDEL-F1: 0.9941) and Illumina (HG003 INDEL-F1: 0.9959, HG004 INDEL-F1: 0.9958) suggesting further improvement required for nanopore-based methods. On HG003 PacBio-CLR data, we observed a SNP-F1 score of 0.9892 with our method and 0.9755 with Longshot (Supplemental Table B.14). Overall, we find that haplotype-aware long-read-based variant calling produces high-quality SNP variant calls comparable to those produced by short-read-based variant identification methods (Figure 5.2a, Supplementary table B.15). This is the first demonstration we are aware of in which SNP variant calls with Oxford Nanopore data achieved similar accuracy to Illumina SNP variant calls.

In segmental duplications, 250bp+ non-unique regions, and low-mappability regions where short-reads have difficulty in mapping, we observe the average SNP F1-scores of Illumina (Seg. Dup. F1-score: 0.94, 250bp+ non-unique:0.66, Low-mappability: 0.94) drop sharply for both HG003 and HG004 samples. Long-read based PacBio HiFi (Seg. Dup. F1-score: 0.99, 250bp+ non-unique:0.90, Low-mappability: 0.99) and Oxford Nanopore (Seg. Dup. F1-score: 0.98, 250bp+ non-unique:0.94, Low-mappability: 0.98) produce more accurate SNP variants. In the major histocompatibility complex

(MHC) region, we see Oxford Nanopore (HG003 SNP F1-score: 0.9958, HG004 SNP F1-score: 0.9966) achieve best performance followed by PacBio HiFi (HG003 SNP F1-score: 0.9951, HG004 SNP F1-score: 0.9955) and Illumina HG003 SNP F1-score: 0.9939, HG004 SNP F1-score: 0.9921). In general, the long-read-based haplotype-aware methods outperform short-reads in more repetitive regions of the genome (Figure 5.2b, Supplementary table B.16).

In low-complexity regions like homopolymer, di-mer and tri-mer repeat regions of the genome, the average variant calling performance of Nanopore drops (7bp-11bp homopolymer SNP F1-score: 0.96, 11bp+ homopolymer SNP F1-Score: 0.88) for both HG003 and HG004 samples compared to Illumina (7bp-11bp homopolymer SNP F1-score: 0.998, 11bp+ homopolymer SNP F1-Score: 0.998) and PacBio HiFi (7bp-11bp homopolymer SNP F1-score: 0.998, 11bp+ homopolymer SNP F1-Score: 0.984). In 11bp-50bp di-mer and 15bp-50bp tri-mer repeat regions of the genome, we see the average performance of Oxford Nanopore (di-mer SNP F1-score: 0.969, tri-mer SNP F1-score: 0.984) is lower than PacBio HiFi (di-mer SNP F1-score: 0.995, tri-mer SNP F1-score: 0.995) and Illumina (di-mer SNP F1-score: 0.998, tri-mer SNP F1-score: 0.998). Overall, the Illumina short-read based variant calling method achieves higher accuracy in low-complexity regions of the genome (Figure 5.2c, Supplementary table B.17).

We further compare the variant calling performance of Illumina, PacBio HiFi and ONT in “easy regions” (the inverse of all difficult regions: excluding all tandem repeats, homopolymers, imperfect homopolymers, difficult to map regions, segmental duplications, and GC <25% or >65%) which cover 76% of the genome [166]. In this comparison, we see ONT variant calling performance (SNP: 0.9988 INDEL: 0.9719) is comparable to Illumina (SNP: 0.9997, INDEL: 0.9996) and HiFi (SNP: 0.9999, INDEL: 0.9997) showing that in easy regions, all technologies can generate high-quality variants (Supplementary table B.18). We further look into regions with no tandem repeats (covering 86% of the genome) and see that ONT performance (SNP: 0.9981, INDEL: 0.97) is comparable to Illumina (SNP: 0.996, INDEL: 0.996). However in tandem repeat and homopolymer regions, the ONT SNP calling performance drops from 0.998 to 0.9748, and the INDEL calling performance drops from 0.97 to 0.54 (Supplementary Table B.19) suggesting that ONT variant calling can generate competitive variant calling in the 86% of the genome outside tandem repeat and homopolymer regions, and it suffers only in the 4% of the genome which is highly repetitive.

5.4.4 Phaseset and Haplotagging Accuracy

We compared phaseset accuracy for Margin and WhatsHap on HG001 against GIAB's phased v3.3.2 variants with 25× nanopore, 50× nanopore, 75× nanopore, and 35× PacBio HiFi data. We generated genotyped variants with PEPPER-Margin-DeepVariant, and used both Margin and WhatsHap to phase the final variant set. The phasesets produced by both tools were analyzed using `whatshap stats` and `whatshap compare` against the trio-confirmed truth variants in high-confidence regions.

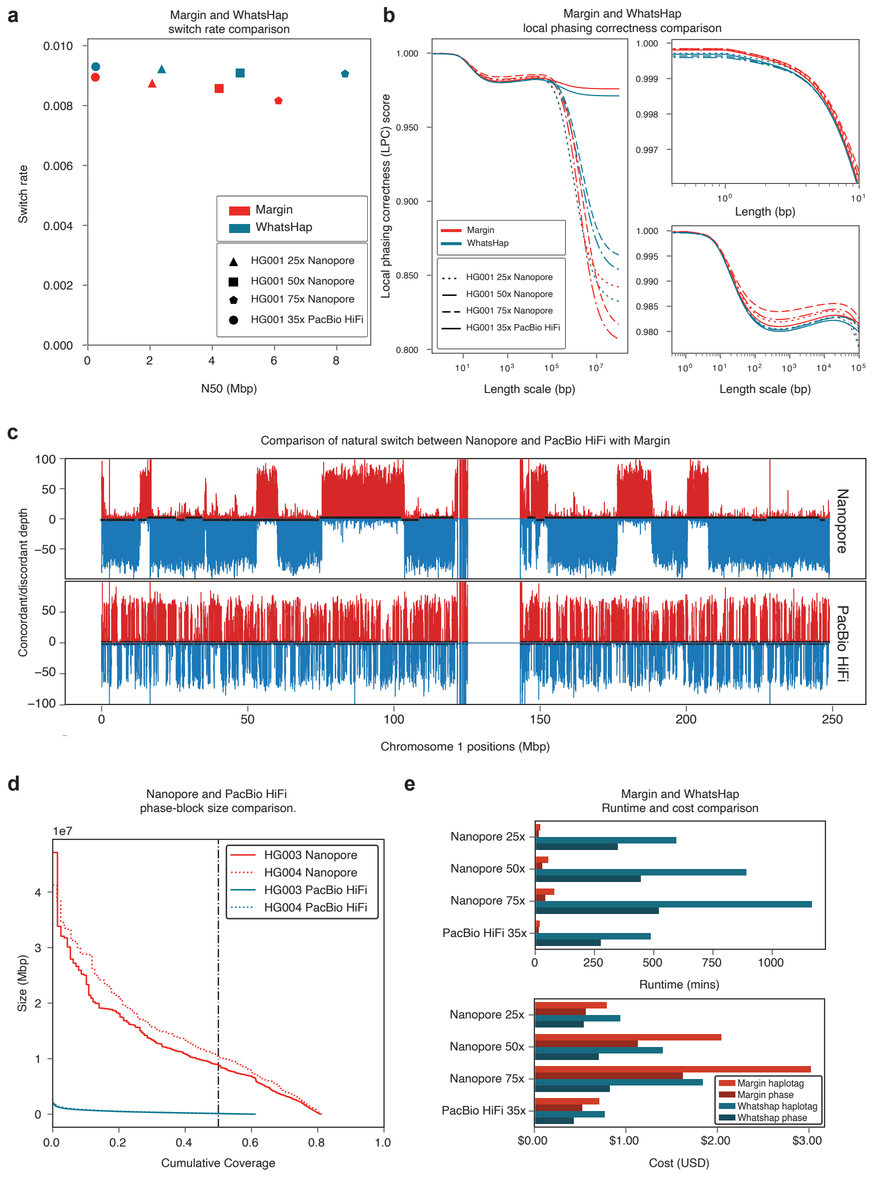


Figure 5.3: **Margin and WhatsHap phasing results. (a)** Phaseset switch rate to N50. **(b)** Novel metric “Local Phasing Correctness” analyzing phaseset accuracy across different length scales. **(c)** “Natural Switch Rate” describing haplotagging accuracy for reads. **(d)** Phaseset N50 for Nanopore and PacBio HiFi data on HG003 and HG004. **(e)** Cost and runtime comparison between Margin and Whatshap.

For all datasets, Margin had a lower switch error rate (0.00875, 0.00857, 0.00816, 0.00895) than WhatsHap (0.00923, 0.00909, 0.00906, 0.00930), but lower phaseset N50 (2.07, 4.21, 6.13, 0.24 Mb) than WhatsHap (2.37, 4.90, 8.27, 0.25 Mb) (Figure 5.3a, Supplementary Tables B.20 and B.21).

We also compared phaseset accuracy for Margin and WhatsHap on the same data using a novel metric we call "Local Phasing Correctness" (LPC). In brief, the LPC is a value between 0 and 1 that summarizes whether every pair of heterozygous variants is correctly phased relative to each other. The contribution of each pair of variants is weighted based on the distance between them, with the weights varying according to a tunable parameter, the "length scale". The length scale can be understood roughly as the scale of distances that influence the metric (see online methods). The LPC is a generalization of the standard metrics of switch error rate and Hamming rate, which have a close relationship with the LPC at length scales 0 and infinity respectively. We plot the LPC across various length scale values (Figure 5.3b). Margin produced more accurate phasing for all length scales for 25× nanopore and 35× CCS. Margin also produced more accurate phasing for 50× nanopore for length scales up to 128kb and for 75× nanopore for length scales up to 242kb, after which WhatsHap outperforms Margin. Both tools exhibit local maxima for length scales from 20-30 kilobases.

To analyze haplotagging accuracy, we artificially constructed an admixture sample by trio-binning reads from HG005 and HG02723 and combining an equal amount of maternal reads from each sample, resulting in a 55× nanopore alignment and a 35× PacBio HiFi alignment. We ran PEPPER-SNP, haplotagged each alignment with Margin using these variants, and compared the number of direct-matched reads R_d (truth H1 to tagged H1 or truth H2 to tagged H2) and cross-matched reads R_c (truth H1 to tagged H2 or truth H2 to tagged H1) of the output. In Figure 5.3c, for each 10kb bucket in chr1 we plot the number of reads that were direct-matched (top, red) and cross-matched (bottom, blue) for both data types, with phasesets plotted in black alternating between top and bottom. With this "Natural Switch" plot, it is possible to identify consistent phasing as regions where the majority of reads are either direct- or cross-matched, and switch errors in the haplotagging as regions where the majority of reads transition between the two. As the plot shows, nanopore reads allow us to haplotag consistently with phase sets in the range of tens of megabases, whereas PacBio HiFi reads cannot be used for long-range haplotagging. For each bucket we can calculate a local haplotagging accuracy using the ratio: $\max(R_c, R_d)/(R_c + R_d)$. On average the haplotagging accuracy is 0.9626 for ONT data and 0.9800 for HiFi data using Margin (Supplementary Table B.22). Full plots including local haplotagging accuracy visualization are shown in Supplementary Figures B.3, B.4, B.5, B.6. As Margin has higher haplotagging accuracy compared to WhatsHap, we see that the variant calling with Margin exhibits higher accuracy compared to WhatsHap for both Oxford Nanopore and PacBio HiFi data (Supplementary Table B.23, Supplementary Table B.12).

Lastly we compare the runtime and cost for the haplotag and phase actions on the four HG001 datasets using Margin and WhatsHap. When configured to use 64 threads, Margin at peak used 35GB of memory on a GCP instance `n1-highcpu-64` costing \$2.27/hr. This results in a total cost (for haplotagging and phasing) of \$1.35

(36m) for 25x ONT, \$3.17 (84m) for 50x ONT, \$4.64 (123m) for 75x ONT, \$1.23 (33m) for 35x PacBio HiFi. Given WhatsHap's concurrent use of two threads and three GB of memory we determined it could be run most cheaply on the GCP `n1-standard-2` instance type for \$0.095/hr, resulting in a total cost of \$1.48 (941m), \$2.10 (1336m), \$2.66 (1688m), and \$1.20 (764m) respectively (Supplementary table B.24)

5.4.5 Gene Analysis

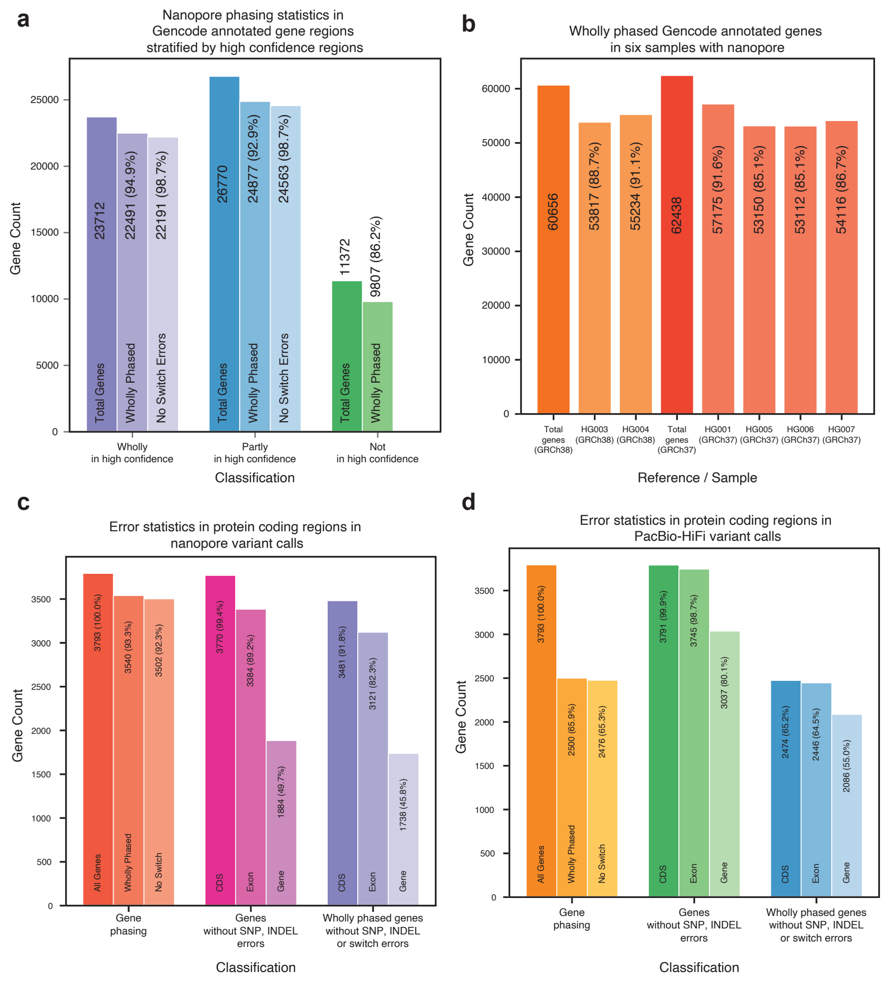


Figure 5.4: **Gene analysis.** **(a)** Phasing analysis for HG001 over GENCODE annotated gene regions, stratified by GIAB high confidence coverage. Percentages are relative to their predecessor. **(b)** Wholly phased GENCODE annotated gene regions. Percentages are relative to the total genes annotated on the reference. **(c)** Error statistics including wholly phased genes, genes without SNP or INDEL errors, and wholly phased genes without SNP, INDEL, or switch errors over a subset of the protein coding genes. **(d)** The same statistics on HG001 with 35x PacBio HiFi data.

We performed an analysis of Margin's phasing over genic regions to understand its utility for functional studies. With 75× nanopore data from HG001 on GRCh37, we classified each of the GENCODE v35 genes[50] (coding and non-coding) as wholly, partially, or not spanned for the GIAB v3.3.2 high confidence regions, the phasesets proposed by Margin, and the switch errors determined by `whatshap compare` between the two. In Figure 5.4a, we first plot the number of gene bodies as spanned by high confidence regions (23712 wholly, 26770 partly, 11372 not), then further compare how many of each division were wholly spanned by Margin's phasesets (22491, 24877, 9807), and finally how many of these had no detected phasing errors (22191, 24563, unknown). In Figure 5.4b, we plot the number of genes wholly phased by Margin on GRCh38 (60656) for HG003 and HG004 (53817, 55234) and on GRCh37 (62438) for HG001, HG005, HG006, and HG007 (57175, 53150, 53112, 54116).

We analyzed accuracy statistics for PEPPER-Margin-DeepVariant with the same HG001 data used above stratified by GENCODE annotations. SNP and INDEL accuracies are largely similar between stratifications of all regions, all genes, and all protein coding genes, with improved performance for protein coding sequence (including CDS, start codon, and stop codon annotations for protein coding genes) (Supplementary Tables B.25, B.26).

We combined the accuracy and phasing analysis by selecting the 3793 protein coding genes which had at least 80% of their coding sequence covered by the high confidence regions and analyzed the presence of phasing and SNP/INDEL errors on HG001 with 75x nanopore (Figure 5.4c) and 35x PacBio HiFi (Figure 5.4d) reads (Supplementary Tables B.27, B.28). Nanopore had better read phasing for these genes, with 3540 wholly spanned by Margin's phasesets and only 38 exhibiting a switch error (1.07%), as compared to PacBio HiFi with 2500 wholly spanned genes and 24 switch errors (0.96%). We then counted the number of genes which had no SNP or INDEL errors in the high confidence region for the entire gene, all annotated exons in the gene, and all coding sequences in the gene; PacBio HiFi performs best for this metric with 3037, 3745, and 3791 respectively as compared to nanopore with 1884, 3384, and 3770 perfectly called regions. Lastly, we identified how many of these gene regions were perfectly captured (wholly phased with no switch errors and having no SNP or INDEL miscalls) for the entire gene (1738 for nanopore, 2086 for PacBio HiFi), for all annotated exons (3121, 2446), and for all coding sequences (3481, 2471). For nanopore data, we find that for 91.8% of genes the CDS is fully phased and genotyped without error, and for 82.3% of genes all exons are fully phased and genotyped without error.

5.4.6 Diploid polishing of *de novo* assemblies

Oxford Nanopore-based assemblers like Flye [74] and Shasta [148] generate haploid assemblies of diploid genomes. By calling and phasing variants against the haploid contigs they produce, it is possible to polish the haploid assembly into a diploid assembly. We implemented such a diploid de novo assembly polishing method with PEPPERMargin-DeepVariant (Figure 5.5a). It can polish haploid Oxford Nanopore-based assemblies with either Nanopore or PacBio HiFi reads.

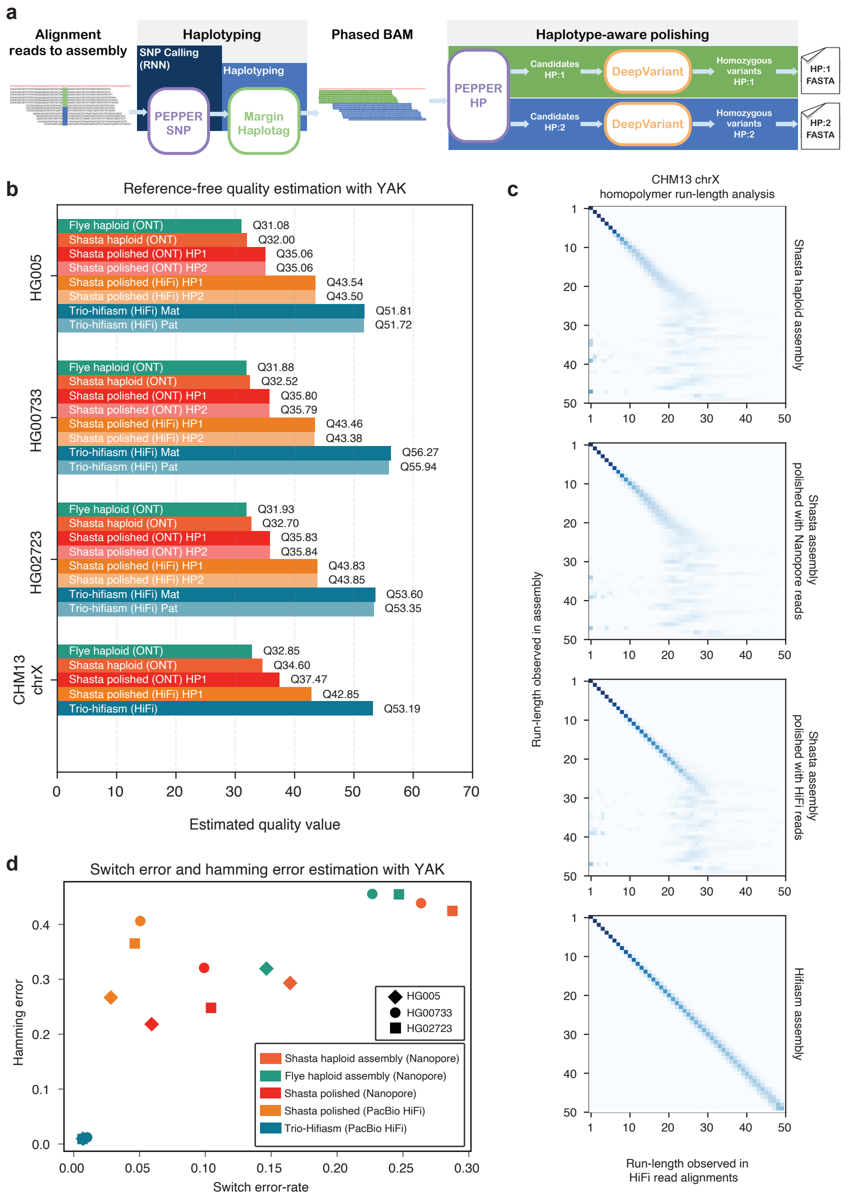


Figure 5.5: **Diploid assembly polishing results.** **(a)** Illustration of the diploid assembly polishing pipeline. **(b)** Estimated quality values of assemblies using YAK. **(c)** CHM13-chrX run-length confusion matrix between different assemblies and PacBio HiFi reads aligned to the corresponding assembly. **(d)** Switch error and hamming error comparison between assemblies.

The assembly polishing pipeline employs the modules similarly to the variant calling pipeline. The difference between variant calling and assembly polishing is after we phase the alignment file using the initial set of SNPs, we take candidates from each haplotype independently and classify the candidate as error or not-error using DeepVariant. This entails converting the genotyping classification used in variant calling to a binary classification to predict if a candidate is true error or not. A detailed description of this method is presented in the online methods.

5.4.7 Diploid *de novo* assembly polishing performance

We generated haploid assemblies using Shasta [148] and Flye [74] for diploid samples HG005, HG00733, HG02723, and haploid sample CHM13 (chrX) using nanopore reads, and we polished the Shasta assemblies using ONT and PacBio HiFi reads. To evaluate the base-level accuracy of the assemblies we use the kmer-based tool YAK[24], which uses Illumina trio data to estimate sequence quality, switch error rates, and hamming error rates. We compare the haploid assemblies, polished diploid assemblies, and trio-aware diploid assemblies generated with hifiasm[24]. Hifiasm uses parental short-read data to generate maternal and paternal assemblies.

The estimated quality values (QV) of nanopore-based assemblies with Shasta (HG005: QV32, HG00733: QV32.7, HG02723: QV32.52) assembler are higher than the nanopore-based Flye assemblies (HG005: QV31.08, HG00733: QV31.93, HG02723: QV31.88). Furthermore, the NG50s of the Shasta assemblies (HG005: 39.83Mbp, HG00733: 42.49Mbp, HG02723: 49.18Mbp) are higher compared to the Flye assemblies (HG005: 37.25Mbp, HG00733: 36.60Mbp, HG02723: 39.65Mbp) (Supplementary table B.29).

As Shasta generated higher quality assemblies compared to Flye, we polished the Shasta assemblies with the PEPPER-Margin-DeepVariant diploid polisher. The nanopore-polished assemblies achieve Q35+ estimated quality (HG005: Q35.06, HG00733: QV35.83, HG02723: QV35.8) and PacBio-HiFi-polished assemblies achieve Q40+ estimated quality (HG005: Q43.5, HG00733: QV43.8, HG02723: QV43.8) for all three diploid samples. Finally, we show that the unpolished CHM13-chrX Shasta assembly (QV34.6) can be improved to QV36.9 with nanopore-based and QV42.7 PacBio-HiFi-based assembly polishing with PEPPER-Margin-DeepVariant. Compared to the nanopore-based Shasta assemblies, the trio-aware PacBio HiFi assembler hifiasm achieves higher quality assemblies with respect to base-level accuracy (HG005: QV51.81, HG00733: 53.6, HG02723: 55.94, CHM13-chrX: QV53.03) but the NG50 of the hifiasm assemblies are lower for HG00733 and HG02723 samples (HG005: 51.32Mbp, HG00733: 32.47Mbp, HG02723: 22.21Mbp). In summary, PEPPER-Margin-DeepVariant achieves Q35+ ONT-based assembly polishing and Q40+ PacBio-HiFi-based assembly polishing of ONT assemblies (Figure 5.5b, Supplementary table B.29, Supplementary table B.30).

The dominant error modality for ONT data are homopolymers[148]. In Figure 5.5c we show the run-length confusion matrix of PacBio HiFi read alignments to four chrX assemblies of CHM13-chrX. The Shasta assembly starts to lose resolution at run-lengths greater than 7 (RL-7) and loses all resolution around RL-25. The nanopore-polished assembly improves homopolymer resolution up to RL-10, but also fails to resolve run-lengths greater than RL-25. The PacBio HiFi polished assembly has fair resolution up to RL-25. The trio-hifiasm assembly shows accurate homopolymer resolution up to and beyond RL-50.

Figure 5.5d shows the switch error-rate of the assemblies. The switch error-rate of haploid Shasta assemblies (HG005: 0.16, HG00733: 0.26, HG02723: 0.28) reduce after polishing with PEPPER-Margin-DeepVariant (HG005: 0.05, HG00733: 0.09, HG02723: 0.10) with ONT data. Similarly, the hamming error rate of the Shasta assemblies (HG005: 0.29, HG00733: 0.43, HG02723: 0.42) reduce after polishing the assemblies with ONT-data (HG005: 0.20, HG00733: 0.31, HG02723: 0.24). Compared to the ONT-polished assemblies the PacBio-HiFi-polished assemblies have higher hamming error-rate (HG005: 0.26, HG00733: 0.40, HG02723: 0.36) but lower switch error-rate (HG005: 0.02, HG00733: 0.04, HG02723: 0.04). The trio-hifiasm that use maternal and paternal short-reads to resolve haplotypes have much lower switch error-rate and hamming error-rate (Figure 5.5d, Supplementary table B.30).

The trio-hifiasm method is able to phase large structural variants in the assemblies. Therefore, trio-hifiasm is expected to produce globally higher quality assemblies. PEPPER-Margin-DeepVariant can not achieve similar global accuracy by polishing haploid assemblies in a diploid manner with small variants. In table 5.1, we compare HG005 assemblies at the small variant level. The analysis show that the F1-score of unpolished Shasta assembly (INDEL: 0.1203, SNP: 0.4928) improves significantly after polishing with nanopore reads using PEPPER-Marin-DeepVariant (INDEL: 0.3611, SNP: 0.9825). The PacBio-HiFi-polished Shasta assembly achieves similar F1-score (INDEL: 0.9565, SNP: 0.9976) compared to the trio-hifiasm assembly (INDEL: 0.9733, SNP: 0.9988). This analysis provide evidence that PEPPER-Margin-DeepVariant can effectively improve the assembly quality at small variant level.

The current version of the PEPPER-Margin-DeepVariant pipeline does not attempt to polish structural variants (SVs, >50bp in size). The resulting haplotypes preserve all SVs initially contained in the input assembly. Since the input assemblies are haploid, only one (randomly assembled) allele for each heterozygous SV is retained within the pair of output haplotypes. To benchmark SV recall and precision, we first called SVs from the assemblies using svim-asm [65] and then validated the reconstructed SV sets using the previously described approach [179]. Our benchmarks using HG002, HG005, HG0073 and HG02733 genomes show that input Shasta assemblies on average contained signatures of 94.6% and 48.3% of homozygous and heterozygous SVs, respectively. After polishing using PEPPER-Margin-DeepVariant, the average reconstruction rate slightly increased to 95.7% and 50.9% for homozygous and heterozygous SVs, respectively. The average SV precision was 81.6% before and 83.2% after polishing (Supplementary Table B.31).

Method	Polished with	Type	True positives	False negatives	False positives	Recall	Precision	F1 score
Shasta	-	INDEL	128989	253403	1629782	0.3373	0.0732	0.1203
		SNP	1279988	1696540	939228	0.4300	0.5769	0.4928
Shasta	Nanopore	INDEL	279793	102605	906353	0.7317	0.2397	0.3611
		SNP	2940462	36058	68863	0.9879	0.9771	0.9825
Shasta	PacBio HiFi	INDEL	367819	14575	19341	0.9619	0.9512	0.9565
		SNP	2971733	4787	9689	0.9984	0.9968	0.9976
Hifiasm	-	INDEL	374002	8390	12451	0.9781	0.9686	0.9733
		SNP	2973193	3320	3730	0.9989	0.9987	0.9988

Table 5.1: Small variant accuracy evaluation of HG005 assemblies against GIAB HG005 v3.3.2 benchmarking set. We derive a small variant set against GRCh37 from the assemblies using dipcall[93] and compare the variant calls against HG005 GIAB benchmark. We restrict our analysis in regions that are assembled by both Shasta and trio-hifiasm and falls in the high-confidence region defined by GIAB.

5.5 Discussion

Long-read sequencing technology is allowing gapless human genome assembly [109] and enabling investigations in the most repetitive regions of the genome[121].

In this work, we present PEPPER-Margin-DeepVariant, a state-of-the-art long-read variant calling pipeline for Oxford nanopore data. For the first time, we show that nanopore-based SNP identification outperforms a state-of-the-art short-read based method at whole genome scale. Particularly in segmental duplication and difficult-to-map regions, the nanopore-based method outshines the short-read based method. It seems likely, therefore, that the anticipated widespread application of long-read variant calling will for the first time accurately illuminate variation in these previously inaccessible regions of the genome.

The genomic contexts where nanopore SNP accuracy suffers for our pipeline are identifiable, meaning that variant calls in these regions can be treated with skepticism while calls outside these contexts can be handled with confidence. The one obvious area that Nanopore variant calling lags is in INDEL accuracy. While the results achieved here are to our knowledge the best shown so far, we believe it is likely that further technological innovations at the platform level will be required to make nanopore INDEL accuracy on par with other technologies in all genomic contexts. However, we find that in the 86% of the genome without tandem repeats or homopolymers, INDEL calls from our method are already of high quality.

PEPPER-Margin-DeepVariant is designed for whole-genome sequencing analysis. Although targeted sequencing with the Oxford Nanopore platform is reasonably popular, several issues may limit the application. For example, read length, read quality, coverage, and heterozygosity of the target region are expected to be fairly different than whole-genome sequencing. Further investigation and benchmarking are required to extend support for variant calling on amplicon sequencing data.

Oxford Nanopore provides a highly-multiplexed sequencing solution with its PromethION device [148]. With this device and the PEPPER-Margin-DeepVariant pipeline described here it should be comfortably possible to go from biosample collection to complete genome inferences in under half a day. This fast turnaround should enable its use in a medical context, where diagnosis for acute disease situations requires speed.

We have demonstrated our nanopore-based phasing is able to wholly phase 85% of all genes with only 1.3% exhibiting a switch error. This phasing ability could play a useful role in population genetics studies [157, 19] and clinical genomics [54]. For clinical applications the accurate identification of compound-heterozygotes should be particularly valuable.

We have extended PEPPER-Margin-DeepVariant to PacBio HiFi reads and demonstrated a more accurate and cheaper solution to the existing WhatsHap-DeepVariant variant calling methods, making cohort-wide variant calling and phasing with PacBio-HiFi more accessible. Currently, we find PacBio-HiFi sequencing analyzed with our method has the best performance, but we expect that improvements to nanopore pore technology and basecalling may close this gap.

We have demonstrated diploid polishing of nanopore-based haploid assemblies with PEPPER-Margin-DeepVariant. We achieve Q35+ nanopore polished assemblies and Q40+ PacBio-HiFi-polished assemblies. We observe that our polishing method can resolve homopolymer errors up to 20bp with PacBio HiFi data. However, our polishing method fails to resolve 25bp+ long homopolymers indicating that they need to be resolved during the consensus generation of the *de novo* assembly methods. As nanopore assembly methods like Shasta move toward generating fully resolved diploid genome assemblies like trio-hifiasm, our polishing method can enable nanopore-only Q40+ polished diploid assemblies.

5.6 Acknowledgements

Research reported in this publication was supported by the National Human Genome Research Institute of the National Institutes of Health under Award Numbers

U41HG010972, 1R01HG008742, U01HG010961, R01HG009737 and 2U41HG007234. Research reported in this publication was supported by the National Heart, Lung, And Blood Institute of the National Institutes of Health under Award Number U01HL137183. We thank Circulomics Inc. for sharing HG001 Nanopore data. We thank Justin Zook and Justin Wagner for providing a draft version of the HG005 v4.2.1 benchmark. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Chapter 6

Successful Application of the PEPPER-Margin-DeepVariant Pipeline

The PEPPER-Margin-DeepVariant pipeline is the first demonstration of successful ONT-based variant calling. As Illumina sequencing has obvious shortcomings in specific genomic contexts and PacBio HiFi data can be prohibitively slow and expensive for some applications, this represents the cheapest and fastest method for accurate genomic inference available today. This has resulted in a successful application of the toolkit in a medical context.

A pipeline was developed where the PromethION sequencer was used to concurrently sequence from 48 flow cells until a sufficient amount of sequence data was generated, with basecalling and alignment performed in near real time. Variant calling was performed using a hardware-accelerated version of the PEPPER-Margin-DeepVariant pipeline, and a method for fast variant filtration was used to select candidate variants for manual review. At its fastest, this entire process from sample collection to initial diagnosis took seven hours eighteen minutes.

This work resulted in two primary publications, “Ultrarapid Nanopore Genome Sequencing in a Critical Care Setting” published in the New England Journal of Medicine [56] and “Accelerated identification of disease-causing variants with ultra-rapid nanopore genome sequencing” to be published in Nature Biotechnology, with an additional case study “Ultra-rapid nanopore whole genome genetic diagnosis of dilated cardiomyopathy in an adolescent with cardiogenic shock” to be published in Circulation: Genomic and Precision Medicine.

I was involved in this project and contributed to the manuscript, but I don’t include the full text of these papers as the bulk of the work to transform this application into a record-breaking medical pipeline was largely performed by John E. Gorzynski, Sneha D. Goenka, and Kishwar Shafin.

Appendix A

Supplementary information for efficient de novo assembly of eleven human genomes in nine days

Supplementary Results

Nanopore sequencing eleven human genomes in nine days

Supplementary Table A.1: Read N50s stratified by sample and flowcell (three for each sample) for 11 samples.

Sample	Flowcell No.	Flowcell N50	Sample N50
GM24143	1	48891	46757
	2	47044	
	3	44335	
GM24149	1	46054	43306
	2	44245	
	3	39618	
GM24385	1	50349	48705
	2	49319	
	3	46448	
HG00733	1	29862	29584
	2	30473	
	3	28417	
HG01109	1	48795	45894
	2	44218	
	3	44670	
HG01243	1	45467	43567
	2	44681	
	3	40554	
HG02055	1	44320	45457
	2	47148	
	3	44902	
HG02080	1	38519	39319
	2	40123	
	3	39315	
HG02723	1	50509	49723
	2	47842	
	3	50817	
HG03098	1	41463	40629
	2	42308	
	3	38115	
HG03492	1	32149	30168
	2	30063	
	3	28292	
Average	**-**	**41889**	**42101**

Supplementary Table A.2: Throughput stratified by sample and flowcell (three for each sample) in gigabases (Gb) for 11 samples.

Sample	Flowcell No.	Flowcell (Gb)	Sample (Gb)	Coverage
GM24143	1	87	280	84.72
	2	97		
	3	95		
GM24149	1	82	273	82.6
	2	107		
	3	84		
GM24385	1	26	157	47.43
	2	71		
	3	59		
HG00733	1	62	242	73.45
	2	90		
	3	89		
HG01109	1	71	219	66.48
	2	79		
	3	70		
HG01243	1	71	187	56.68
	2	73		
	3	43		
HG02055	1	71	202	61.33
	2	67		
	3	65		
HG02080	1	71	172	52.21
	2	42		
	3	59		
HG02723	1	81	227	68.7
	2	69		
	3	78		
HG03098	1	79	177	53.63
	2	40		
	3	58		
HG03492	1	61	158	47.74
	2	45		
	3	51		
Average	**-**	**69**	**208**	**63.18**

Supplementary Table A.3: Mean, median, and modal values for read alignment identities of 11 samples, aligned to GRCh38. Metrics were generated per read. Total gigabases of read data for each sample are detailed in Supplementary Table A.2

Sample	Mean	Median	Mode
GM24143	0.87188	0.89651	0.920
GM24149	0.87665	0.90511	0.930
GM24385	0.88276	0.91143	0.935
HG00733	0.87165	0.89682	0.925
HG01109	0.87033	0.89845	0.930
HG01243	0.88525	0.91435	0.935
HG02055	0.87215	0.90572	0.930
HG02080	0.88188	0.91259	0.935
HG02723	0.84914	0.87565	0.920
HG03098	0.85522	0.88156	0.915
All samples:	**0.87251**	**0.90068**	**0.930**

Supplementary Table A.4: Summary read statistics derived from human saliva sequencing.

Reads	Bases	Mean Length	Median Length	Read N50
594,753	10,961,203,887	18,430	15,580	27,778

A.0.1 Shasta: assembling a human genome from nanopore reads in under 6 hours

Supplementary Table A.5: QUAST assembly metrics of three samples on four assemblers before polishing, compared against GRCh38 with no alternate contigs.

Sample	Metric	Shasta	Wtdbg2	Flye	Canu
HG00733	# contigs	2,150	5,086	1,852	778
	Total length	2,783,599,890	2,792,376,827	2,816,034,584	2,900,719,051
	N50	24,429,871	18,763,119	28,763,002	44,759,083
	NG50	21,088,309	15,338,021	25,227,330	40,627,903
	# disagreements	814	3,985	6,555	4,570
	Genome fraction (%)	94.982	92.938	95.763	96.404
	Duplication ratio	0.995	1.005	0.986	1.014
	# mismatches per 100 kbp	156.21	248.78	506.12	231.24
	# indels per 100 kbp	453.97	664.90	1,480.91	677.26
	Total aligned length	2,775,307,347	2,742,343,142	2,769,440,009	2,858,769,830
	NA50	16,052,981	9,106,500	18,577,806	21,157,324
	NGA50	12,765,264	7,787,949	16,267,214	19,945,150
HG002	# contigs	1,847	5,310	1,627	767
	Total length	2,801,200,983	2,793,889,694	2,819,241,152	2,901,099,163
	N50	23,346,484	15,380,722	31,253,170	33,064,788
	NG50	20,205,529	13,750,884	25,917,293	32,340,595
	# disagreements	901	3,572	5,881	3,882
	Genome fraction (%)	95.622	93.136	96.228	96.959
	Duplication ratio	0.995	1.004	0.981	1.009
	# mismatches per 100 kbp	167.75	261.72	549.10	231.39
	# indels per 100 kbp	520.33	796.71	1,650.63	792.45
	Total aligned length	2,792,458,737	2,743,401,414	2,768,347,339	2,863,787,213
	NA50	16,068,951	8,564,600	18,803,788	21,330,391
	NGA50	14,189,972	7,361,363	16,079,132	18,175,258
CHM13	# contigs	1,236	6,428	1,269	558
	Total length	2,809,087,051	2,836,802,421	2,857,931,691	2,919,690,848
	N50	46,037,322	15,522,332	36,829,446	80,507,947
	NG50	41,091,906	14,039,241	35,319,460	79,504,166
	# disagreements	1,051	4,202	5,452	4,768
	Genome fraction (%)	95.307	93.124	96.022	96.553
	Duplication ratio	1.000	1.017	0.997	1.014
	# mismatches per 100 kbp	155.15	256.17	443.85	226.04
	# indels per 100 kbp	358.45	535.46	1,023.79	484.46
	Total aligned length	2,798,043,587	2,780,449,715	2,807,157,420	2,864,418,837
	NA50	23,475,255	6,786,237	18,991,999	25,611,947
	NGA50	18,990,051	5,892,796	17,032,972	23,819,455

A.0.2 Contiguously assembling MHC haplotypes

Supplementary Table A.6: QUAST disagreement count for four assemblers on different regions of the genome for four samples. We report disagreements that happen in all chromosomes of GRCh38, then incrementally exclude centromeric regions, segmental duplication regions (Seg Dups), and all other regions enriched for SVs (chrY, acrocentric chromosome arms, and QH-regions)

Sample	Assembler	Disagreements in GRCh38 autosomes and chrX, chrY	Disagreements outside centromeres	Disagreements outside centromeres and seg dups	Disagreements outside centromeres, seg dups, chrY, acrocentric chr arms, and QH-regions
HG002	Shasta	901	755	284	121
	Flye	5881	1226	513	117
	Canu	3882	2347	689	216
	Wtdbg2	3572	1213	484	148
HG00733	Shasta	814	662	256	110
	Flye	6555	1261	604	134
	Canu	4570	2791	755	224
	Wtdbg2	3985	1166	474	135
CHM13	Shasta	1051	795	333	129
	Flye	5452	1228	448	107
	Canu	4768	2764	864	164
	Wtdbg2	4202	1519	592	249

Supplementary Table A.7: Disagreement count in the intersection of the assemblies for each sample (see Online Methods). Total Disagreements describes all disagreements found in 100bp windows before taking the intersection; note that these counts are very close to those reported by QUAST. Consensus Disagreements describes disagreements in the intersection of the four assemblies. Genome fraction describes total coverage over GRCh38 for the consensus sequence.

Sample	Assembler	Total Disagreements	Consensus Disagreements	Genome Fraction
HG002	Shasta	863	179	87.16%
	Flye	5823	178	87.16%
	Canu	3779	328	87.16%
	Wtdbg2	3509	215	87.16%
HG00733	Shasta	792	161	87.43%
	Flye	6546	178	87.43%
	Canu	4524	383	87.43%
	Wtdbg2	3975	205	87.43%
CHM13	Shasta	1033	242	87.53%
	Flye	5446	217	87.53%
	Canu	4682	712	87.53%
	Wtdbg2	4190	404	87.53%

Supplementary Table A.8: Disagreement count and fraction of genome covered on chromosome X for four assemblers on CHM13 assemblies with no polishing, compared to the chromosome X assembly from the Telomere-to-Telomere Consortium. These numbers were obtained via running QUAST.

Assembler	Disagreements	Genome Fraction
Shasta	5	97.73%
Wtdbg2	87	94.17%
Flye	18	98.41%
Canu	9	98.16%

Supplementary Table A.9: BAC analysis on selected dataset. BACs were selected (31 of CHM13 and 16 of HG00733) for falling within unique regions of the genome, specifically >10 Kb away from the closest segmental duplication. *Closed* refers to the number of BACs for which 99.5% of their length aligns to a single locus in the assembly. *Attempted* refers to the number of BACs which have an alignment for >5 Kb of sequence with >90% identity to only one contig (BACs which have such alignments to multiple contigs are excluded). Identity metrics are for *closed* BACs.

Sample	Assembler	BAC counts				Median Quality		Mean Quality	
		Total	Attempted	Closed	Closed of attempted %	Identity %	QV	Identity %	QV
CHM13	Canu	31	31	30	96.77	99.40	22.18	99.34	21.84
	Flye	31	31	31	100.00	97.58	16.17	97.65	16.28
	Shasta	31	31	31	100.00	99.55	23.51	99.51	23.07
	Wtdbg2	31	29	28	96.55	99.46	22.71	99.39	22.15
HG00733	Canu	16	16	15	93.75	98.74	18.98	98.61	18.56
	Flye	16	16	16	100	97.99	16.97	98.01	17.02
	Shasta	16	16	16	100	98.84	19.38	98.79	19.20
	Wtdbg2	16	16	16	100	98.81	19.26	98.79	19.20

Supplementary Table A.10: BAC analysis on full dataset, 341 on CHM13 and 179 on HG00733. *Closed* refers to the number of BACs for which 99.5% of their length aligns to a single locus. *Attempted* refers to the number of BACs which have an alignment for ¿5Kb of sequence with ¿90% identity to only one contig (BACs which have such alignments to multiple contigs are excluded). Identity metrics are for *closed* BACs.

Sample	Assembler Polisher	BAC counts				Median Quality		Mean Quality	
		Total	Attempted	Closed	Closed of attempted %	Identity %	QV	Identity %	QV
CHM13	Canu	341	309	287	92.88	99.22	21.07	98.93	19.7
	Flye	341	227	202	88.98	97.54	16.09	97.51	16.03
	Shasta	341	94	92	97.87	99.47	22.74	99.37	21.99
	Wtdbg2	341	70	62	88.57	99.36	21.96	99.28	21.43
HG00733	Canu	179	137	124	90.51	98.73	18.95	98.43	18.05
	Flye	179	98	80	81.63	98.09	17.18	97.76	16.49
	Shasta	179	42	40	95.23	98.76	19.08	98.13	17.30
	Wtdbg2	179	52	46	88.46	98.70	18.87	98.02	17.04

Supplementary Table A.11: BAC analysis intersection of attemted BACs by all four assemblers, 65 on CHM13 and 27 on HG00733. *Closed* refers to the number of BACs for which 99.5% of their length aligns to a single locus. *Attempted* refers to the number of BACs which have an alignment for ¿5Kb of sequence with ¿90% identity to only one contig (BACs which have such alignments to multiple contigs are excluded). Identity metrics are for *closed* BACs.

Sample	Assembler Polisher	BAC counts				Median Quality		Mean Quality	
		Total	Attempted	Closed	Closed of attempted %	Identity %	QV	Identity %	QV
CHM13	Canu	65	65	64	98.50	99.29	21.53	99.21	21.01
	Flye	65	65	65	100.00	97.57	16.16	97.61	16.22
	Shasta	65	65	65	100.00	99.50	23.03	99.41	22.33
	Wtdbg2	65	65	59	90.80	99.39	22.17	99.29	21.49
HG00733	Canu	27	27	26	96.30	98.66	18.76	98.54	18.37
	Flye	27	27	27	100.00	98.07	17.14	98.08	17.16
	Shasta	27	27	27	100.00	98.80	19.23	98.30	17.71
	Wtdbg2	27	27	26	96.30	98.75	19.01	98.53	18.32

Supplementary Table A.12: Base-level accuracies on four different assemblers for three samples. Analysis is performed with whole-genome truth sequences.

Sample	Assembler	Percentage Errors			
		Balanced	Identity	Deletion	Insertion
HG002 Guppy 2.3.5	Shasta	0.975%	0.061%	0.849%	0.065%
	Wtdbg2	1.181%	0.080%	1.073%	0.029%
	Canu	1.400%	0.065%	1.316%	0.020%
	Flye	1.636%	0.068%	0.450%	1.118%
HG00733 Guppy 2.3.5	Shasta	1.062%	0.083%	0.887%	0.093%
	Wtdbg2	1.217%	0.108%	1.059%	0.051%
	Canu	1.328%	0.074%	1.224%	0.031%
	Flye	1.854%	0.089%	0.445%	1.320%
CHM13 Guppy 2.3.1	Shasta	0.540%	0.039%	0.430%	0.072%
	Wtdbg2	0.689%	0.068%	0.583%	0.038%
	Canu	0.705%	0.038%	0.643%	0.024%
	Flye	2.213%	0.051%	0.448%	1.715%

Supplementary Table A.13: Base-level accuracies on four different assemblers for three samples in the regions of intersection of the assemblies. Analysis is performed only on regions where all assemblers have an assembled sequence.

Sample	Assembler	Percentage Errors			
		Balanced	Identity	Deletion	Insertion
HG002 Guppy 2.3.5	Shasta	0.943%	0.056%	0.823%	0.064%
	Wtdbg2	1.145%	0.077%	1.041%	0.028%
	Canu	1.319%	0.050%	1.253%	0.016%
	Flye	1.554%	0.063%	0.432%	1.059%
HG00733 Guppy 2.3.5	Shasta	1.021%	0.064%	0.875%	0.083%
	Wtdbg2	1.162%	0.088%	1.034%	0.041%
	Canu	1.307%	0.065%	1.213%	0.030%
	Flye	1.847%	0.068%	0.431%	1.348%
CHM13 Guppy 2.3.1	Shasta	0.513%	0.016%	0.406%	0.048%
	Wtdbg2	0.660%	0.054%	0.575%	0.030%
	Canu	0.692%	0.027%	0.645%	0.021%
	Flye	2.198%	0.036%	0.460%	1.702%

Supplementary Table A.14: Runtime and cost of three assembly workflows on Amazon Web Services (AWS) platform.

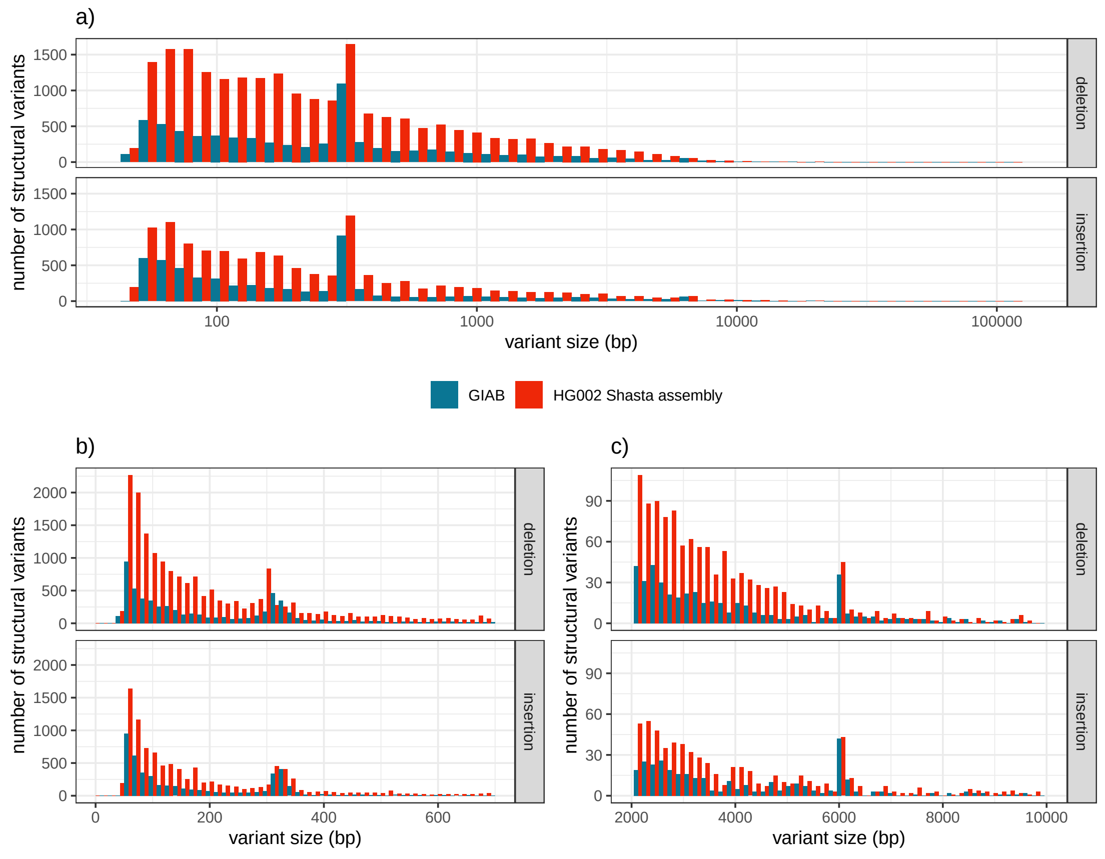
Method	Sample	Minutes	Threads Used	Peak Memory	AWS Instance Type	AWS Instance Cost
WTDBG2	HG00733	2971	63	365	r5a.16xlarge	$3.62
	GM24385	1752	63	293	r5a.16xlarge	$3.62
	CHM13	1655	63	312	r5a.16xlarge	$3.62
WTDBG2 (wtpoa-cns)	HG00733	248	31	12	r5a.16xlarge	$3.62
	GM24385	274	24	12	r5a.16xlarge	$3.62
	CHM13	257	31	12	r5a.16xlarge	$3.62
Flye	HG00733	3421	123	1013	x1.32xlarge	$13.34
	GM24385	3749	64	727	x1.16xlarge	$6.67
	CHM13	4084	126	911	x1.32xlarge	$13.34
Shasta	HG00733	298	128	966	x1.32xlarge	$13.34
	HG01109	355	128	-	x1.32xlarge	$13.34
	HG01243	296	128	-	x1.32xlarge	$13.34
	HG02055	309	128	-	x1.32xlarge	$13.34
	HG02080	276	128	-	x1.32xlarge	$13.34
	HG02723	373	128	-	x1.32xlarge	$13.34
	HG03098	238	128	-	x1.32xlarge	$13.34
	HG03492	200	128	-	x1.32xlarge	$13.34
	GM24385	240	128	692	x1.32xlarge	$13.34
	GM24149	427	128	-	x1.32xlarge	$13.34
	GM24143	451	128	-	x1.32xlarge	$13.34
	CHM13	317	128	-	x1.32xlarge	$13.34

Supplementary Table A.15: Runtime breakdown for each step of the Shasta assembler.

Sample	Input	MinHash	Align-ments	Marker graph creation	Trans-itive reduc-tion	Assemble	Output	Other	Total
HG00733	30	9	93	73	17	15	2	55	298
HG01109	29	10	136	89	16	17	2	53	355
HG01243	23	7	104	73	16	15	2	51	296
HG02055	25	9	113	73	15	15	2	53	309
HG02080	22	7	95	67	15	14	2	49	276
HG02723	29	9	146	89	19	16	2	59	373
HG03098	23	8	73	53	14	14	2	47	238
HG03492	19	7	57	44	11	14	2	40	200
GM24385	20	7	92	49	12	13	2	41	240
GM24149	34	11	149	124	21	18	2	64	427
GM24143	35	11	168	120	24	18	2	69	451
CHM13	21	6	173	67	12	13	2	46	345
Average	26	8	117	77	16	15	2	52	317
Percent of total	8%	3%	37%	24%	5%	5%	1%	17%	100%

Supplementary Table A.16: Structural variants extracted from HG002 assembly graph compared to GIAB SV set in high-confidence regions.

Metric	HG002					
	TP	FP	FN	Precision	Recall	F_1
Total	2961	1580	1202	0.6521	0.7117	0.6806
Inserts	2152	1203	810	0.6414	0.7117	0.7289
Deletes	809	377	392	0.6821	0.6681	0.6750



Supplementary Figure A.1: Size distribution of structural variants (¿50 bp) extracted from the Shasta assembly graph for HG002 and the structural variants in the Genome In A Bottle (GIAB) catalog for the same sample. a) Full size distribution for deletions (top) and insertion (bottom), in log-scale. b) and c) zoom in the two peaks caused by Alu (300 bp) and L1 (6 Kbp) insertion polymorphisms.

Supplementary Table A.17: CHM13 MHC unpolished Shasta assembly as compared to the nearest matching haplotype in hg38 (GL000251.2)

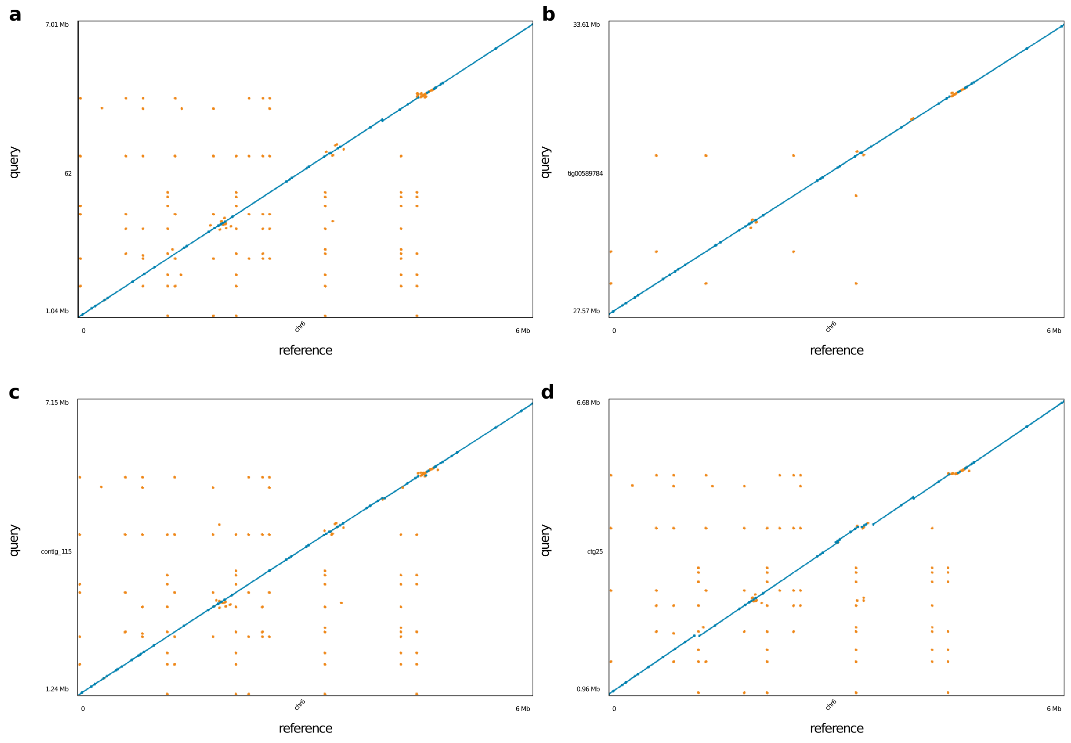
Assembler	Best Contig	Disagreements	Largest Aligned	Mismatch Rate	Indel Rate
Shasta	62	6	2,788,362	0.00296	0.00399
Canu	tig00589784	5	2,792,139	0.00331	0.00607
Flye	contig_115	6	2,787,570	0.00543	0.01106
wtdbg2	ctg25	32	1,819,753	0.00553	0.00576

Supplementary Table A.18: QUAST results for the HG00733 trio-binned maternal reads, using all four assemblers.

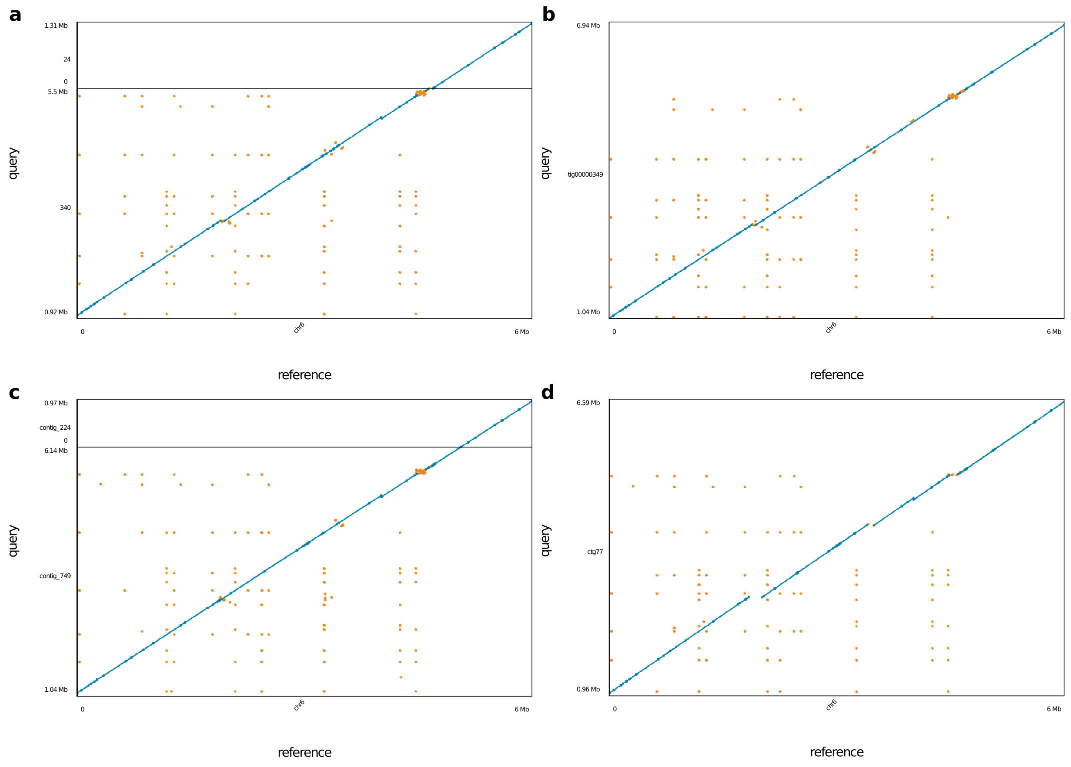
Metric	HG00733-Mother			
	Shasta	Wtdbg2	Flye (initial)	Canu
# contigs	1,934	4,028	1,634	877
Total length	2,754,225,214	2,690,619,717	2,791,893,188	2,829,920,708
N50	9,071,623	14,125,235	25,658,831	19,451,828
NG50	7,702,138	10,217,387	23,775,989	16,507,795
# disagreements	705	3,661	6,082	2,161
Genome fraction (%)	90.824	87.373	92.121	92.298
Duplication ratio	0.993	0.996	0.982	0.999
# mismatches per 100 kbp	194.15	287.89	549.61	232.72
# indels per 100 kbp	576.55	859.83	1585.30	724.67
Total aligned length	2,748,135,723	2,650,821,801	2,751,532,754	2,798,797,021
NA50	7,805,090	7,615,651	15,615,208	11,947,316
NGA50	6,339,949	5,584,544	12,833,996	10,085,023

Supplementary Table A.19: HG00733 Maternal trio binned MHC unpolished Shasta assembly as compared to the nearest matching haplotype in hg38 (GL000255.1)

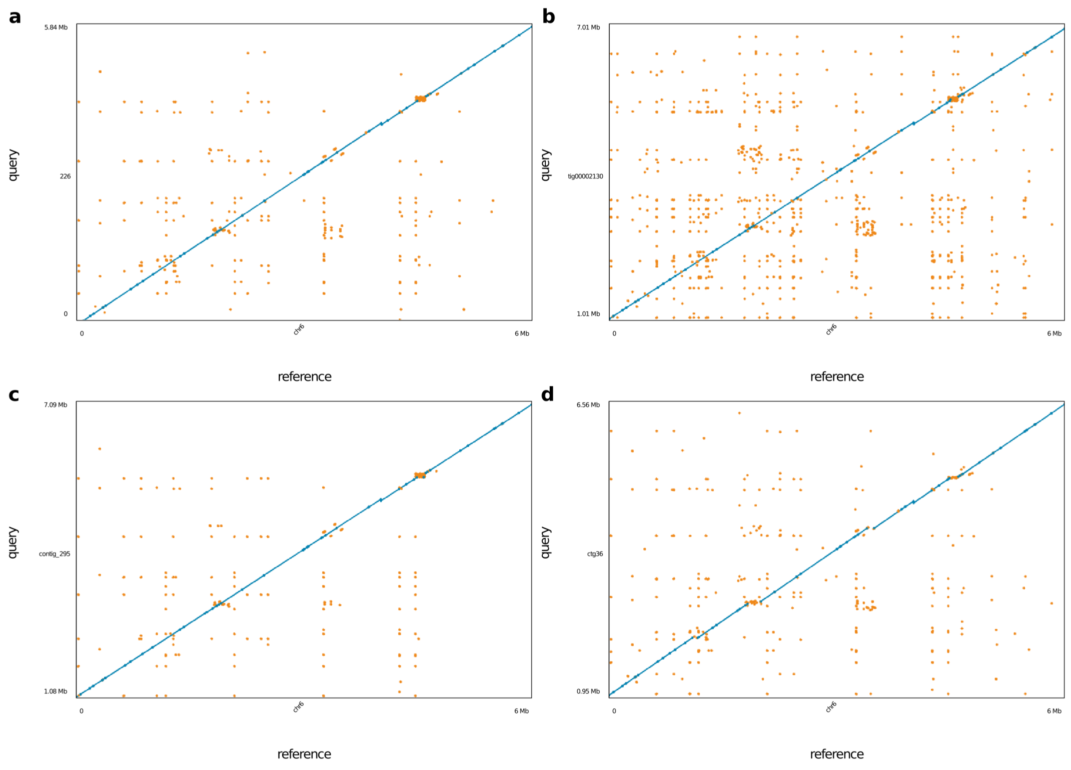
Assembler	Best Contig	Disagreements	Largest Aligned	Mismatch Rate	Indel Rate
Shasta	226	0	4,289,729	0.00206	0.00538
Canu	tig00002130	0	4,289,729	0.00182	0.00676
Flye	contig_295	0	4,289,729	0.00579	0.01759
wtdbg2	ctg36	23	1,418,939	0.00592	0.00905



Supplementary Figure A.2: Dotplot of unpolished CHM13 MHC assembly vs hg38 chr6:28000000-34000000 for the each of the 4 assemblers tested. **(a)** Shasta **(b)** Canu **(c)** Flye (no native polish) **(d)** wtdbg2. Blue dots represent unique alignments and orange dots represent repetitive alignments.



Supplementary Figure A.3: Dotplot of unpolished HG00733 diploid MHC assembly vs hg38 chr6:28000000-34000000 for the each of the 4 assemblers tested. **(a)** Shasta **(b)** Canu **(c)** Flye (no native polish) **(d)** wtdbg2. Blue dots represent unique alignments and orange dots represent repetitive alignments.



Supplementary Figure A.4: Dotplot of unpolished HG00733 maternal haploid MHC assembly vs hg38 chr6:28000000-34000000 for the each of the 4 assemblers tested. **(a)** Shasta **(b)** Canu **(c)** Flye (no native polish) **(d)** wtdbg2. Blue dots represent unique alignments and orange dots represent repetitive alignments.

A.0.3 Deep neural network based polishing achieves QV30 long-read only polishing accuracy

Supplementary Table A.20: Base-level accuracies comparing Racon & Medaka and MarginPolish & HELEN pipelines on Shasta assemblies for three samples. Analysis is performed with whole-genome truth sequences.

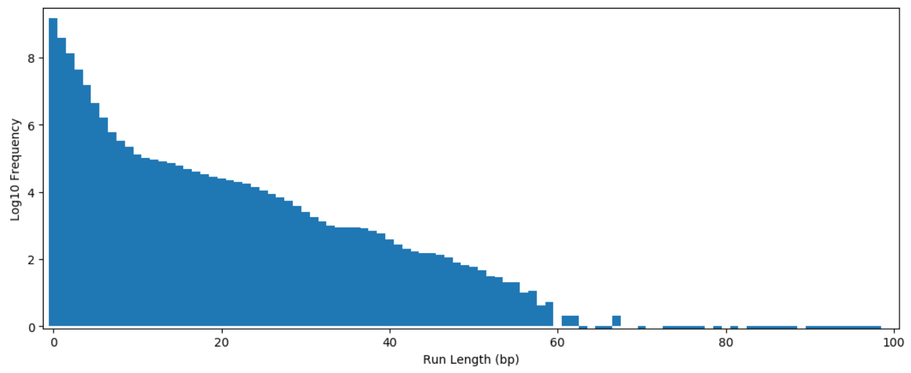
Sample	Polisher		Percentage Errors			
	Method	Model	Balanced	Identity	Deletion	Insertion
HG002 Guppy 2.3.5	Shasta	Unpolished	0.975%	0.061%	0.849%	0.065%
	Racon	4x	0.665%	0.054%	0.579%	0.032%
	Medaka	r941_flip235	0.393%	0.051%	0.303%	0.039%
	MarginPolish	guppy_ff235	0.372%	0.043%	0.248%	0.081%
	HELEN	rl941_flip235	0.279%	0.038%	0.171%	0.070%
HG00733 Guppy 2.3.5	Shasta	Unpolished	1.062%	0.083%	0.887%	0.093%
	Racon	4x	0.715%	0.080%	0.570%	0.066%
	Medaka	r941_flip235	0.455%	0.075%	0.311%	0.069%
	MarginPolish	guppy_ff235	0.460%	0.063%	0.278%	0.118%
	HELEN	rl941_flip235	0.388%	0.066%	0.202%	0.120%
CHM13 Guppy 2.3.1	Shasta	Unpolished	0.540%	0.039%	0.430%	0.072%
	Racon	4x	0.367%	0.037%	0.199%	0.131%
	Medaka	r941_flip213	0.329%	0.033%	0.037%	0.259%
	MarginPolish	guppy_ff233	0.281%	0.027%	0.071%	0.184%
	HELEN	rl941_flip233	0.206%	0.027%	0.062%	0.117%

Supplementary Table A.21: QUAST results for the Shasta assemblies for all samples, post polishing with MarginPolish-HELEN.

Sample	# contigs	Total length	N50	NG50	# mis-assemblies	Genome fraction (%)	# mismatches per 100 kbp	# indels per 100 kbp	Total aligned length	NA50	NGA50
GM24143	2,042	2,802,437,249	23,531,777	19,936,924	970	95.025	128.63	142.77	2,794,379,803	16,323,510	13,840,294
GM24149	2,368	2,816,566,939	20,798,256	17,752,973	990	95.416	130.54	134.60	2,806,847,428	13,174,778	12,128,076
GM24385	1,685	2,819,474,365	23,520,830	20,346,145	960	95.609	127.44	152.17	2,810,951,083	16,200,287	14,315,298
HG00733	1,962	2,800,357,697	24,600,414	21,701,762	877	94.976	126.23	137.92	2,792,792,711	16,156,822	12,971,070
HG01109	2,111	2,820,988,852	21,532,001	18,279,481	1,033	95.564	136.51	140.59	2,811,696,923	13,162,850	12,012,786
HG01243	1,936	2,819,065,027	22,753,128	20,884,160	920	95.521	137.50	143.02	2,810,262,570	16,040,951	14,115,348
HG02055	1,903	2,819,836,390	17,485,643	16,302,857	971	95.592	142.23	162.43	2,810,300,557	13,840,319	12,123,357
HG02080	1,814	2,803,471,776	18,701,305	15,584,440	920	95.045	128.16	134.35	2,794,749,368	12,401,739	11,561,569
HG02723	1,813	2,805,268,038	25,163,327	20,265,678	1,110	95.062	143.30	147.09	2,796,332,696	15,390,923	13,175,818
HG03098	1,790	2,811,295,217	22,571,315	19,620,076	986	95.395	144.36	170.40	2,802,844,336	14,045,283	12,089,849
HG03492	1,811	2,811,690,127	24,629,163	22,891,947	854	95.364	126.61	147.22	2,804,103,412	16,317,390	12,930,516
CHM13	1,186	2,819,245,173	46,206,794	41,255,275	1,107	95.281	136.58	140.38	2,808,536,514	23,540,225	19,532,176

Supplementary Table A.22: Base-level accuracies comparing Racon & Medaka and MarginPolish & HELEN pipelines against CHM13 Chromosome-X. The truth Chromosome-X sequence used reflects the most accurate haploid truth sequence available.

Sample	Polisher		Percentage Errors			
	Method	Model	Balanced	Identity	Deletion	Insertion
CHM-13 Chromosome-X	Shasta	Unpolished	0.469%	0.014%	0.404%	0.051%
	Racon	4x	0.313%	0.017%	0.192%	0.104%
	Medaka	r941_flip213	0.110%	0.012%	0.035%	0.063%
	MarginPolish	guppy_ff233	0.215%	0.008%	0.055%	0.153%
	HELEN	rl941_flip233	0.143%	0.007%	0.041%	0.095%
		rl941_flip231	0.064%	0.006%	0.036%	0.022%



Supplementary Figure A.5: Log frequency of each run length as found in the GRCh38 reference for all bases A,C,G,T up to 100bp. Run lengths greater than 15 account for approximately 0.012% of all homopolymer runs in GRCh38.

Supplementary Table A.23: Base-level accuracies improvements with MarginPolish and HELEN pipeline on four different assemblers for two samples. Analysis is performed with whole-genome truth sequences.

Sample	Polisher		Percentage Errors			
	Method	Model	Balanced	Identity	Deletion	Insertion
HG00733 Guppy 2.3.5	Shasta	Unpolished	1.062%	0.083%	0.887%	0.093%
	MarginPolish	guppy_ff235	0.460%	0.063%	0.278%	0.118%
	HELEN	rl941_flip235	0.388%	0.066%	0.202%	0.120%
	Wtdbg2	Unpolished	1.217%	0.108%	1.059%	0.051%
	MarginPolish	guppy_ff235	0.538%	0.083%	0.333%	0.122%
	HELEN	rl941_flip235	0.473%	0.089%	0.257%	0.127%
	Canu	Unpolished	1.328%	0.074%	1.224%	0.031%
	MarginPolish	guppy_ff235	0.438%	0.050%	0.290%	0.098%
	HELEN	rl941_flip235	0.355%	0.050%	0.206%	0.099%
	Flye	Unpolished	1.854%	0.089%	0.445%	1.320%
	MarginPolish	guppy_ff235	0.425%	0.062%	0.257%	0.106%
	HELEN	rl941_flip235	0.356%	0.064%	0.183%	0.109%
CHM13 Guppy 2.3.1	Shasta	Unpolished	0.540%	0.039%	0.430%	0.072%
	MarginPolish	guppy_ff233	0.281%	0.027%	0.071%	0.184%
	HELEN	rl941_flip233	0.206%	0.027%	0.062%	0.117%
	Wtdbg2	Unpolished	0.689%	0.068%	0.583%	0.038%
	MarginPolish	guppy_ff233	0.361%	0.049%	0.112%	0.201%
	HELEN	rl941_flip233	0.296%	0.053%	0.115%	0.129%
	Canu	Unpolished	0.705%	0.038%	0.643%	0.024%
	MarginPolish	guppy_ff233	0.255%	0.013%	0.075%	0.168%
	HELEN	rl941_flip233	0.173%	0.012%	0.058%	0.103%
	Flye	Unpolished	2.213%	0.051%	0.448%	1.715%
	MarginPolish	guppy_ff233	0.256%	0.022%	0.058%	0.176%
	HELEN	rl941_flip233	0.185%	0.024%	0.052%	0.109%

Supplementary Table A.24: Single-chromosome error rates after polishing with short reads. 10X Chromium reads for sample CHM13 were used to polish via Pilon polishing software. The top half of the table shows the results of three rounds of Pilon, starting from the CHM13 Shasta chrX assembly that had been polished with MarginPolish and HELEN. The bottom half shows the results of three rounds of Pilon, starting from the raw Shasta assembly.

Sample	Assembly	Percentage Errors				Q Scores			
		Balanced	Identity	Deletion	Insertion	Balanced	Identity	Deletion	Insertion
CHM13 ChrX	Shasta (polished)	0.064%	0.006%	0.036%	0.022%	31.92	42.40	34.42	36.51
	Pilon 1x	0.025%	0.004%	0.012%	0.008%	36.06	43.75	39.16	40.75
	Pilon 2x	0.023%	0.004%	0.012%	0.007%	36.29	43.51	39.32	41.34
CHM13 ChrX	Shasta (raw)	0.468%	0.014%	0.404%	0.051%	23.29	38.57	23.94	32.95
	Pilon 1x	0.449%	0.011%	0.395%	0.043%	23.48	39.78	24.03	33.68
	Pilon 2x	0.425%	0.011%	0.373%	0.041%	23.71	39.49	24.29	33.84

Supplementary Table A.25: Runtime and cost of two polishing workflows on Amazon Web Services (AWS) platform.

Method	Sample	Minutes	Threads Used	Peak Memory	Instance Type	Instance Cost
Racon (4x)	HG00733	3099	62	574	r5a.24xlarge	$5.42
	GM24385	2342	62	501	r5a.24xlarge	$5.42
	CHM13	3700	62	281	r5a.24xlarge	$5.42
Medaka mini_align	HG00733	611	62	101	c5.18xlarge	$3.06
	GM24385	489	62	115	c5.18xlarge	$3.06
	CHM13	810	60	143	c5.18xlarge	$3.06
Medaka call_consensus	HG00733	8611	62	164	c5n.18xlarge	$3.89
	GM24385	3355	62	150	c5n.18xlarge	$3.89
	CHM13	2532	62	149	c5n.18xlarge	$3.89
MarginPolish	HG00733	680	90	66	m5.metal	$4.61
	HG01109	912	70	57	c5.18xlarge	$3.06
	HG01243	835	70	65	c5.18xlarge	$3.06
	HG02055	733	70	77	c5.18xlarge	$3.06
	HG02080	793	70	64	c5.18xlarge	$3.06
	HG02723	1000	64	60	c5.18xlarge	$3.06
	HG03098	852	70	78	c5.18xlarge	$3.06
	HG03492	777	70	80	c5.18xlarge	$3.06
	GM24385	842	70	66	c5.18xlarge	$3.06
	GM24149	1037	64	103	c5.18xlarge	$3.06
	GM24143	1051	64	84	c5.18xlarge	$3.06
	CHM13	739	70	65	c5.18xlarge	$3.06
HELEN consensus	HG00733	216	8 GPUs	-	p2.8xlarge	$7.20
	HG01109	204	8 GPUs	-	p2.8xlarge	$7.20
	HG01243	233	8 GPUs	-	p2.8xlarge	$7.20
	HG02080	212	8 GPUs	-	p2.8xlarge	$7.20
	HG03098	216	8 GPUs	-	p2.8xlarge	$7.20
	GM24385	208	8 GPUs	-	p2.8xlarge	$7.20
	GM24143	226	8 GPUs	-	p2.8xlarge	$7.20
HELEN stitch	HG00733	59	32	-	p2.8xlarge	$7.20
	HG01109	50	32	-	p2.8xlarge	$7.20
	HG01243	49	32	-	p2.8xlarge	$7.20
	HG02080	54	32	-	p2.8xlarge	$7.20
	HG03098	65	32	-	p2.8xlarge	$7.20
	GM24385	68	32	-	p2.8xlarge	$7.20
	GM24143	62	32	-	p2.8xlarge	$7.20

Supplementary Table A.26: Runtime and cost of two polishing workflows run on a 29 Mb contig from the HG00733 Shasta assembly. MarginPolish uses an improved stitch method not used in original runs and Racon was run once instead of four times as was done in the full runs. All runs were configured to use 32 CPUs, except for the GPU runs which were performed with 16 CPUs and 1 GPU (Tesla P100).

Application	**Runtimes**	**Avg Runtime**
MarginPolish	16.6	16.46
	16.47	
	16.31	
HELEN consensus (CPU)	97.46	95.86
	95.55	
	94.56	
HELEN consensus (GPU)	1.63	1.67
	1.72	
	1.65	
HELEN stitch	0.76	0.78
	0.78	
	0.80	
Racon 1x	52.00	52.04
	52.15	
	51.98	
mini_align	3.01	3.00
	3.00	
	2.98	
Medaka (CPU)	17.26	17.01
	16.78	
	16.98	
Medaka consensus (GPU)	10.55	10.62
	10.73	
	10.57	
Medaka stitch (GPU)	0.68	0.68
	0.68	
	0.68	

A.0.4 Long-read assemblies contain nearly all human coding genes

Supplementary Table A.27: Transcript-level analysis with Comparative Annotation Toolkit (CAT) of MarginPolish & HELEN and Racon & Medaka on three samples from Shasta assemblies.

Metric		HG002		HG00733		CHM13	
		HELEN	MEDAKA	HELEN	MEDAKA	HELEN	MEDAKA
Transcripts Found	Total	83093	83105	83002	82928	82833	82807
	Percent	99.536	99.551	99.427	99.339	99.225	99.194
Full mRNA Coverage	Total	25721	20367	28612	26573	40132	38081
	Percent	30.811	24.397	34.274	31.832	48.074	45.617
Full CDS Coverage	Total	41396	36248	45104	43956	53089	52297
	Percent	49.588	43.421	54.030	52.655	63.595	62.646
Transcripts With Frameshift	Total	35339	40783	31333	32647	23261	24441
	Percent	42.332	48.854	37.534	39.108	27.864	29.278
Transcripts With Original Introns	Total	76880	76883	76618	76463	76807	76803
	Percent	92.094	92.098	91.780	91.594	92.006	92.002
Transcripts With Full CDS Coverage	Total	41396	36248	45104	43956	53089	52297
	Percent	49.588	43.421	54.030	52.655	63.595	62.646
Transcripts With Full CDS Coverage And No Frameshifts	Total	41245	36158	44982	43860	52966	52160
	Percent	49.407	43.313	53.884	52.540	63.448	62.482
Transcripts With Full CDS Coverage And No Frameshifts And Original Introns	Total	41021	35952	44692	43546	52616	51807
	Percent	49.139	43.067	53.536	52.163	63.028	62.059

Supplementary Table A.28: Gene-level analysis with Comparative Annotation Toolkit (CAT) of MarginPolish & HELEN and Racon & Medaka on three samples from Shasta assemblies.

Metric		HG002		HG00733		CHM13	
		HELEN	MEDAKA	HELEN	MEDAKA	HELEN	MEDAKA
Genes Found	Total	19536	19531	19537	19511	19505	19490
	Percent	99.268	99.243	99.273	99.141	99.111	99.035
Genes With Frameshift	Total	10933	12165	9941	10081	7300	7564
	Percent	55.554	61.814	50.513	51.225	37.093	38.435
Genes With Original Introns	Total	18212	18198	18151	18113	18217	18202
	Percent	92.541	92.47	92.231	92.038	92.566	92.49
Genes With Full CDS Coverage	Total	11070	10066	11812	11756	13648	13534
	Percent	56.25	51.148	60.02	59.736	69.35	68.77
Genes With Full CDS Coverage And No Frameshifts	Total	12454	11570	13127	13081	14625	14562
	Percent	63.283	58.791	66.702	66.468	74.314	73.994
Genes With Full CDS Coverage And No Frameshifts And Original Introns	Total	12422	11539	13098	13042	14603	14531
	Percent	63.12	58.633	66.555	66.27	74.202	73.836
Missing Genes	Total	144	149	143	169	175	190
	Percent	0.732	0.757	0.727	0.859	0.889	0.965

Supplementary Table A.29: Transcript-level analysis with Comparative Annotation Toolkit (CAT) of four HG00733 assemblies polished with MarginPolish and HELEN.

Metric		HG00733			
		Flye HELEN	Canu HELEN	Wtdbg2 HELEN	Shasta HELEN
Transcripts Found	Total	83267	83334	81484	82974
	Percent	99.745	99.825	97.609	99.394
Full mRNA Coverage	Total	33078	28488	28889	30378
	Percent	39.624	34.126	34.606	36.390
Full CDS Coverage	Total	41396	44877	45321	46965
	Percent	59.754	53.758	54.290	56.259
Transcripts With Frameshift	Total	27293	32230	29525	29657
	Percent	32.694	38.608	35.368	35.526
Transcripts With Original Introns	Total	77412	77583	74683	76613
	Percent	92.731	92.936	89.462	91.774
Transcripts with Full CDS Coverage	Total	49883	44877	45321	46965
	Percent	59.754	53.758	54.290	56.259
Transcripts with Full CDS Coverage And No Frameshifts	Total	49766	44737	45217	46802
	Percent	59.614	53.590	54.165	56.064
Transcripts with Full CDS Coverage And No Frameshifts And Original Introns	Total	49459	44412	44924	46505
	Percent	59.247	53.201	53.814	55.708

Supplementary Table A.30: Gene-level analysis with Comparative Annotation Toolkit (CAT) of four HG00733 assemblies polished with MarginPolish and HELEN

Metric		HG00733			
		Flye HELEN	Canu HELEN	Wtdbg2 HELEN	Shasta HELEN
Genes Found	Total	19563	19629	19174	19528
	Percent	99.405	99.741	97.429	99.228
Genes With Frameshift	Total	8698	10160	9323	9464
	Percent	44.197	51.626	47.373	48.089
Genes With Original Introns	Total	18345	18460	17709	18154
	Percent	93.216	93.801	89.985	92.246
Genes With Full CDS Coverage	Total	12966	11889	11817	12207
	Percent	65.884	60.412	60.046	62.027
Genes With Full CDS Coverage And No Frameshifts	Total	14145	13221	13047	13419
	Percent	71.875	67.18	66.296	68.186
Genes With Full CDS Coverage And No Frameshifts And Original Introns	Total	14124	13193	13017	13396
	Percent	71.768	67.038	66.143	68.069
Missing Genes	Total	117	51	506	152
	Percent	0.595	0.259	2.571	0.772

Supplementary Table A.31: BUSCO results of three samples using two polishing workflows on Shasta assemblies.

Sample	Metric	Shasta MarginPolish HELEN	Shasta Racon (4x) Medaka
HG00733	Complete BUSCOs (C)	87.20%	87.10%
	Complete and single-copy BUSCOs (S)	84.20%	83.80%
	Complete and duplicated BUSCOs (D)	3.00%	3.30%
	Fragmented BUSCOs (F)	4.60%	5.30%
	Missing BUSCOs (M)	8.20%	7.60%
HG002	Complete BUSCOs (C)	89.40%	88.80%
	Complete and single-copy BUSCOs (S)	84.80%	85.80%
	Complete and duplicated BUSCOs (D)	4.60%	3.00%
	Fragmented BUSCOs (F)	3.60%	4.30%
	Missing BUSCOs (M)	7.00%	6.90%
CHM13	Complete BUSCOs (C)	86.50%	86.80%
	Complete and single-copy BUSCOs (S)	82.50%	82.80%
	Complete and duplicated BUSCOs (D)	4.00%	4.00%
	Fragmented BUSCOs (F)	5.90%	5.30%
	Missing BUSCOs (M)	7.60%	7.90%

Supplementary Table A.32: BUSCO results for four assemblers on HG00733, post polishing with MarginPolish and HELEN.

Metric	HG00733			
	Flye	Canu	Wtdbg2	Shasta
Complete BUSCOs (C)	87.50%	89.80%	85.80%	87.20%
Complete and single-copy BUSCOs (S)	84.50%	86.80%	82.20%	84.20%
Complete and duplicated BUSCOs (D)	3.00%	3.00%	3.60%	3.00%
Fragmented BUSCOs (F)	5.30%	3.00%	6.30%	4.60%
Missing BUSCOs (M)	7.20%	7.20%	7.90%	8.20%

A.0.5 Comparing to a PacBio HiFi Assembly

Supplementary Table A.33: CHM13 QUAST results for Shasta, MarginPolish, HELEN and PacBio HiFi assembly. Stratified disagreement counts were added after manual determination.

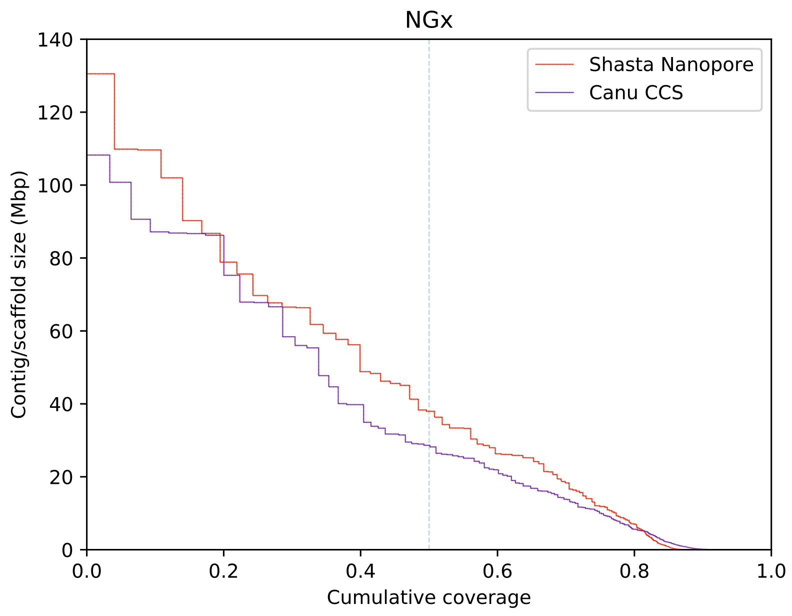
Metric	CHM13	
	Nanopore Shasta MarginPolish, HELEN	PacBio-HiFi Canu Racon
# contigs	1622	5206
Total length	2819245173	3031026325
N50	46206794	29522819
NG50	41255275	29092230
# disagreements	1107	8666
# disagreements outside Centromeres	801	2999
# disagreements outside centromeres and Seg Dups	314	893
Genome fraction (%)	95.281	97.030
# mismatches per 100 kbp	136.58	274.84
# indels per 100 kbp	140.38	32.99
Total aligned length	2808536514	2954558720
NA50	23540225	20440378
NGA50	19532176	20029136

Supplementary Table A.34: Disagreement count in the intersection of the assemblies between the PacBio-HiFi and the Shasta assembly of CHM13. Total Disagreements is all disagreements found in 100bp before windows before taking the intersection, note it is very close to that reported by QUAST. Consensus disagreements: Disagreements in the intersection of the four assemblies.

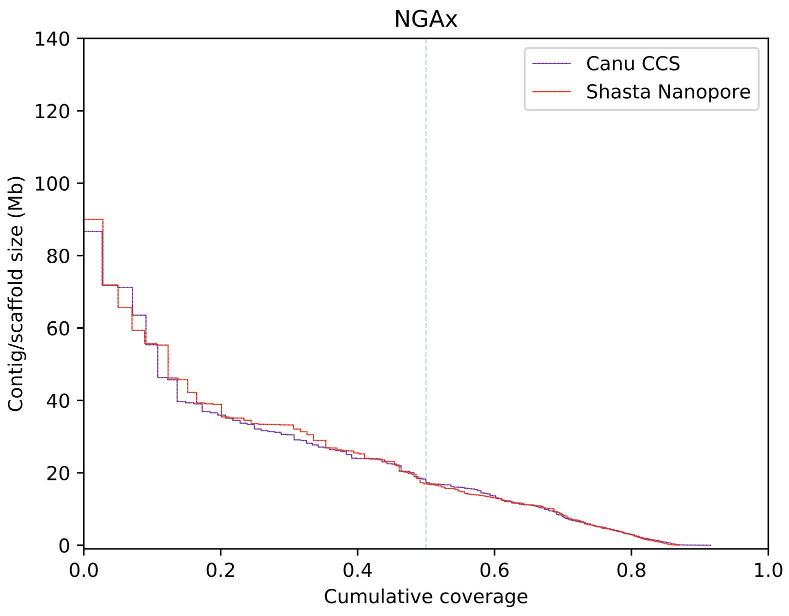
Sample	Assembler	Total disagreements	Consensus disagreements
CHM13	PacBio-HiFi	8469	594
	Shasta	1073	380

Supplementary Table A.35: CHM13 Chromosome-X error rate analysis with Pomoxis for Shasta, MarginPolish, HELEN, and PacBio HiFi assembly.

Sample	Sequencing Platform	Method		Percentage errors			
		Assembler	Polisher	Balanced	Identity	Deletion	Insertion
CHM13 Chr-X	PacBio HiFi	Canu	Racon	0.008%	0.001%	0.004%	0.003%
	Nanopore	Shasta	MarginPolish & HELEN	0.064%	0.006%	0.036%	0.022%

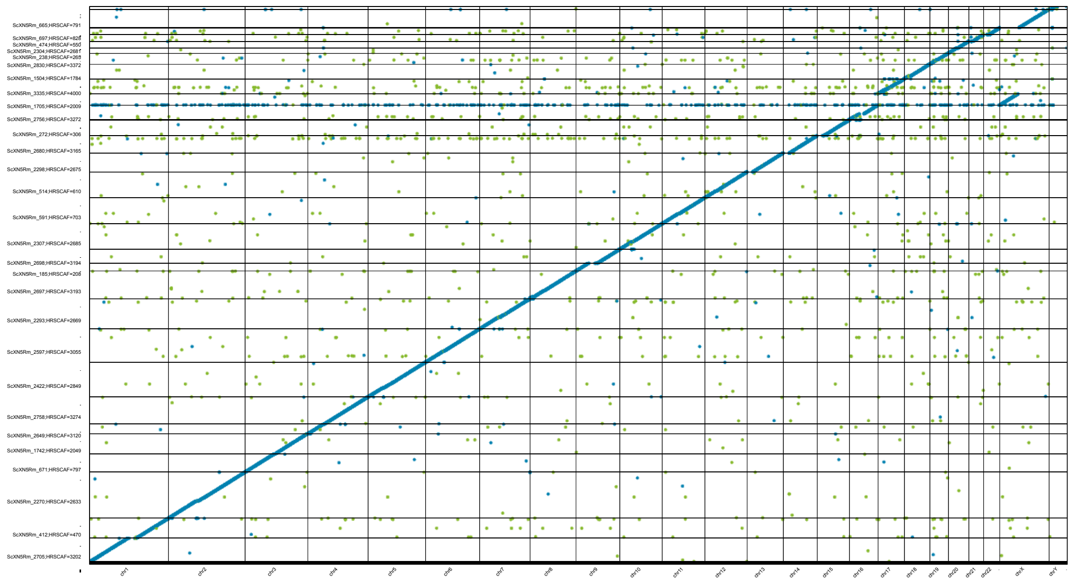


Supplementary Figure A.6: Contig NGx for CHM13 Shasta-HELEN nanopore assembly vs Canu CCS (HiFi) assembly



Supplementary Figure A.7: Contig NGAx for CHM13 Shasta-HELEN nanopore assembly vs Canu CCS (HiFi) assembly

A.0.6 Assembling, polishing and scaffolding 11 human genomes at near chromosome scale



Supplementary Figure A.8: Dotplot for the scaffolded HG002 assembly, aligned with GRCh38. Blue dots represent unique alignments and orange dots represent repetitive alignments.

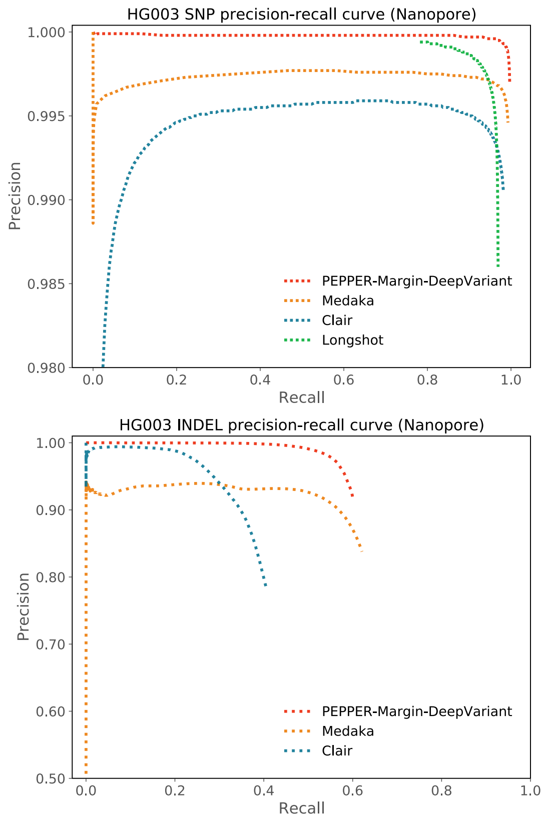
Supplementary Table A.36: QUAST results for all 11 Shasta assemblies scaffolded with HiRise, post polishing with MarginPolish-HELEN

Sample	# contigs	Total length	N50	NG50	# mis-assemblies	# scaffold gap extensive mis-assembies	Genome fraction (%)	# mismatches per 100 kbp	# indels per 100 kbp	Total aligned length	NA50	NGA50
GM24143	1,184	2,802,523,049	129,960,437	128,216,303	1,466	4	95.027	128.28	142.79	2,792,775,664	20,657,530	16,966,477
GM24149	1,323	2,816,683,224	129,643,816	128,275,807	1,530	11	95.417	130.24	134.58	2,804,735,382	18,446,390	15,435,923
GM24385	1,019	2,819,527,260	118,169,209	102,591,941	1,335	6	95.606	127.19	152.25	2,809,570,528	22,369,161	16,601,924
HG00733	1,056	2,800,455,909	129,857,865	118,785,172	1,337	8	94.974	126.16	138.09	2,791,610,554	22,141,375	17,570,210
HG01109	1,156	2,821,098,626	130,282,751	130,166,418	1,529	5	95.559	136.73	140.63	2,809,413,640	19,932,703	17,228,023
HG01243	1,006	2,819,162,443	128,571,344	118,762,399	1,381	7	95.517	137.47	143.03	2,808,041,766	22,146,722	17,559,055
HG02055	977	2,819,933,140	130,184,428	128,180,737	1,387	8	95.587	141.91	162.46	2,809,195,864	21,057,279	18,446,049
HG02080	934	2,803,570,658	129,931,575	128,451,196	1,470	9	95.041	127.98	134.36	2,793,854,132	20,418,609	16,379,851
HG02723	982	2,805,356,030	130,365,062	128,975,828	1,499	9	95.06	143.45	147.13	2,794,747,200	20,232,566	17,865,825
HG03098	926	2,811,385,538	130,040,472	128,535,908	1,439	4	95.391	144.36	170.40	2,801,774,564	22,165,948	17,439,948
HG03492	901	2,811,782,250	130,277,907	100,251,163	1,381	7	95.362	126.54	147.23	2,803,106,787	20,001,587	16,836,756

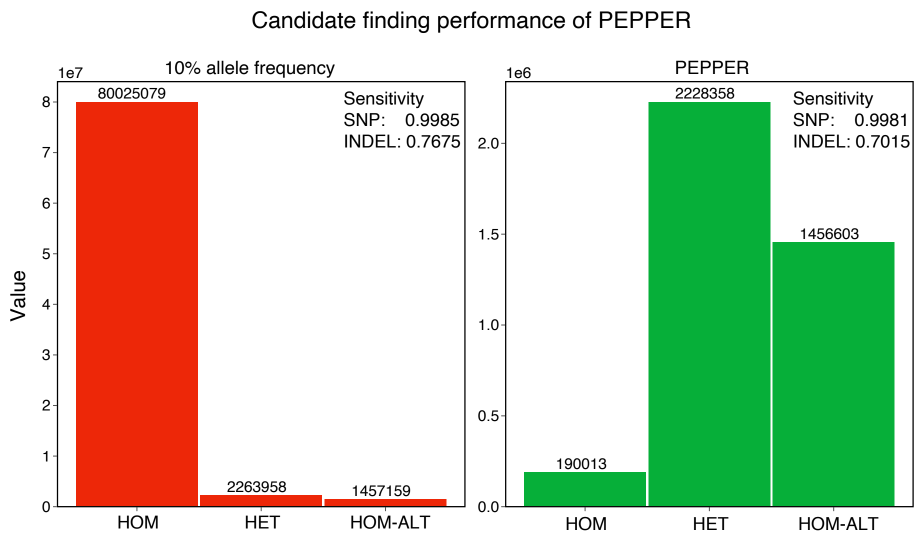
Appendix B

Supplementary information for Haplotype-aware variant calling with PEPPER-Margin-DeepVariant

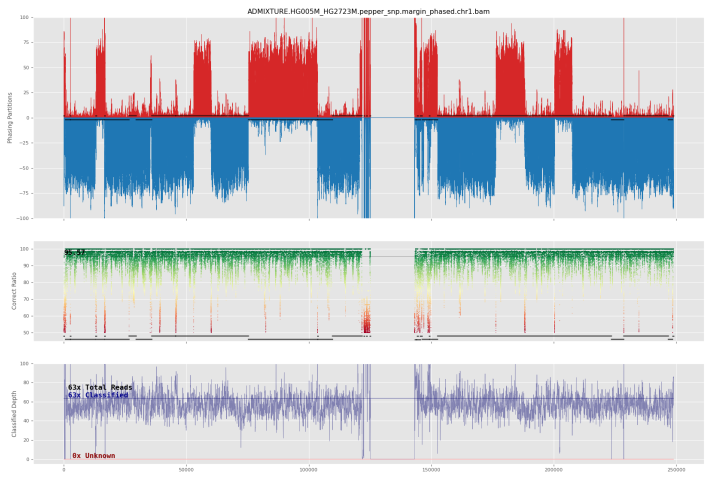
Supplementary Figures



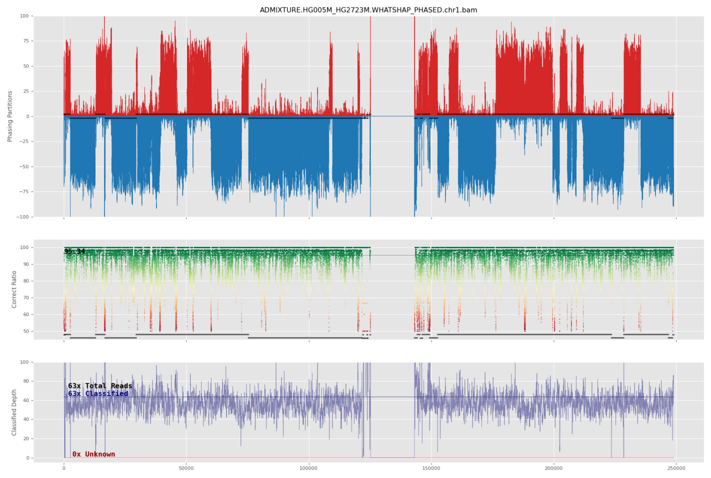
Supplementary Figure B.1: Precision-Recall plot of HG003 for nanopore-based variant callers.



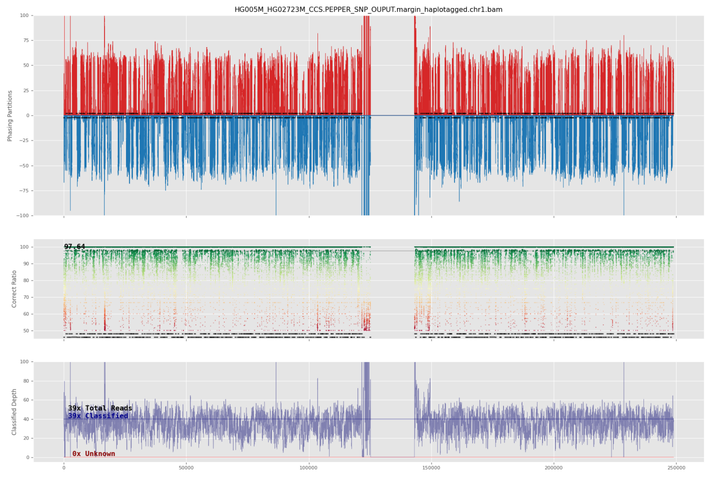
Supplementary Figure B.2: HG003 ONT 90x candidate finding performance comparison between 10% heuristic based approach and PEPPER.



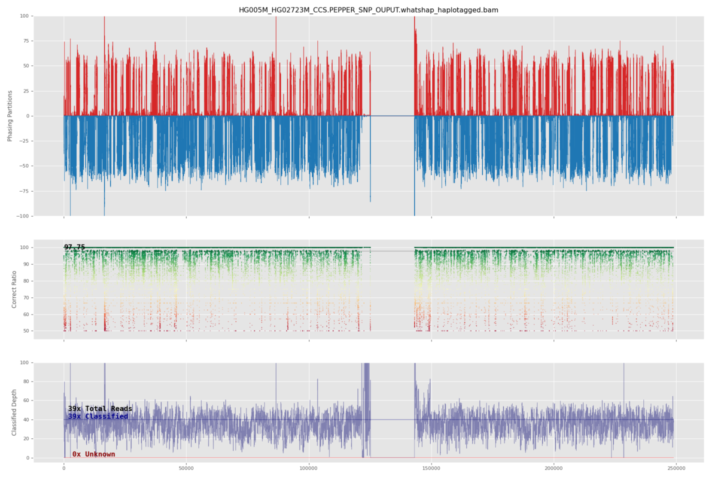
Supplementary Figure B.3: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from nanopore data phased by Margin



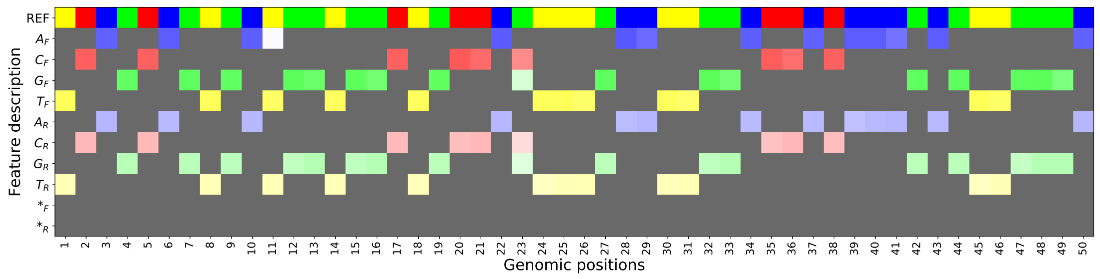
Supplementary Figure B.4: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from nanopore data phased by WhatsHap



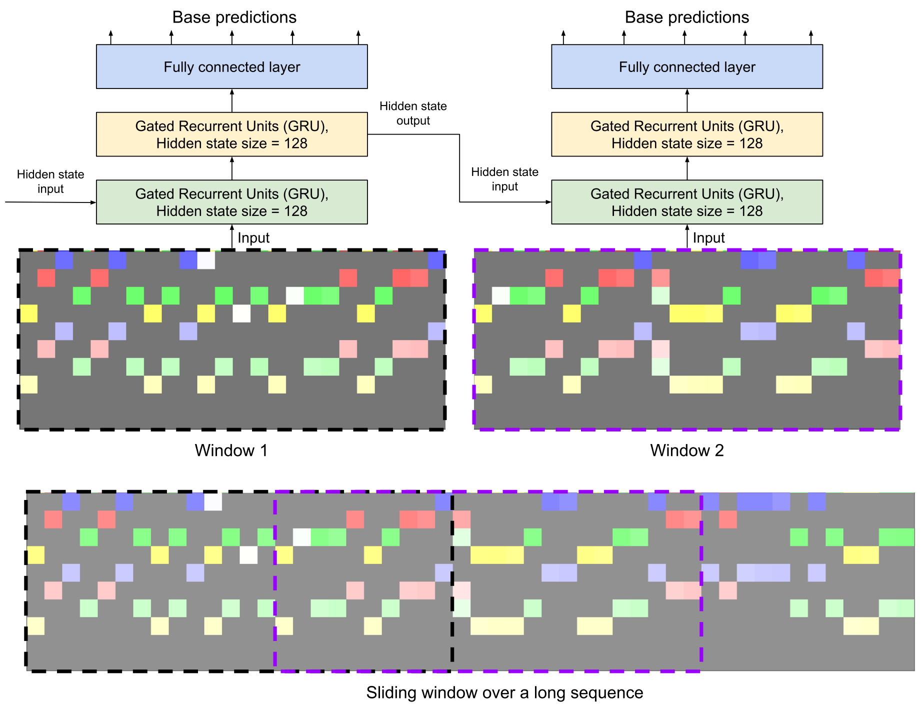
Supplementary Figure B.5: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from PacBio HiFi data phased by Margin



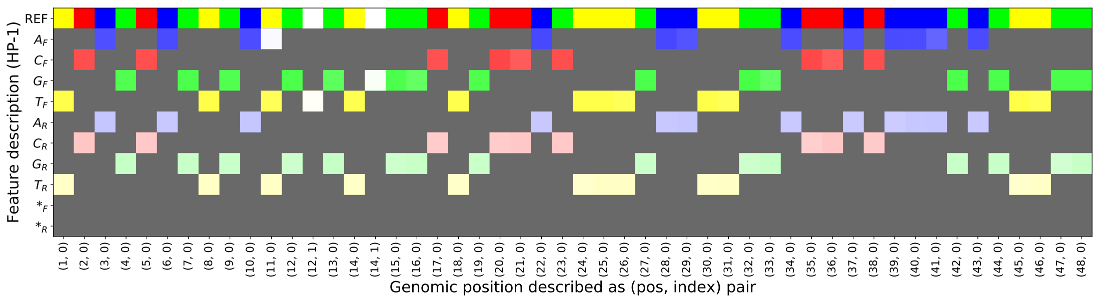
Supplementary Figure B.6: Full Natural Switch plot for chr1 of an admixture of HG005 and HG02723's maternal haplotypes from PacBio HiFi data phased by WhatsHap

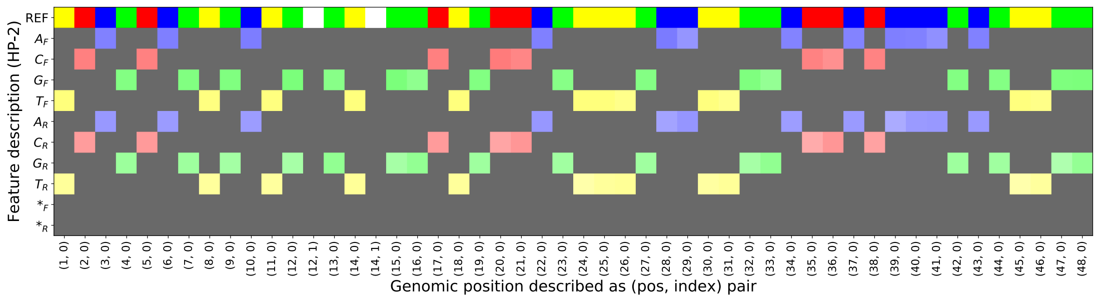


Supplementary Figure B.7: PEPPER-SNP image generation scheme.

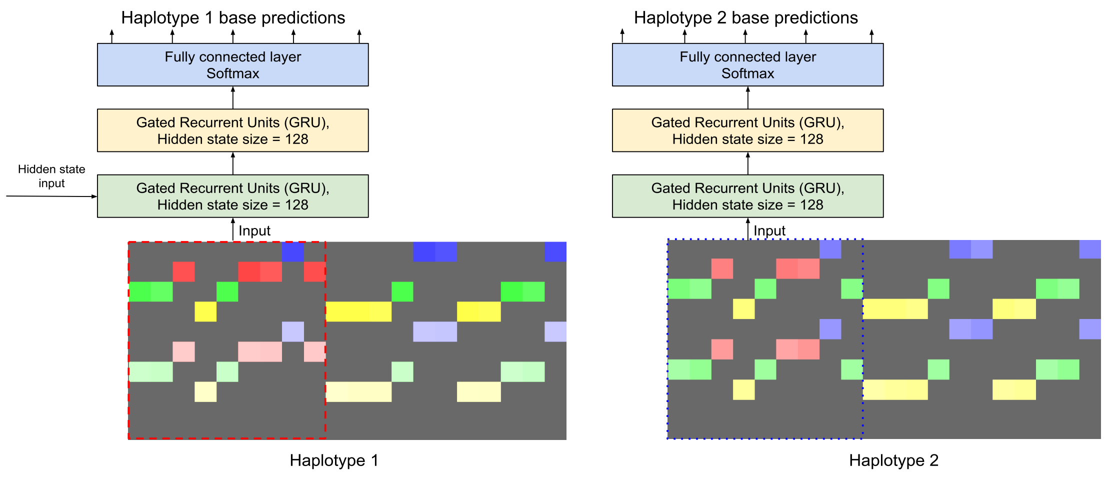

Base predictions
Fully connected layer
Gated Recurrent Units (GRU), Hidden state size = 128
Hidden state output
Hidden state input
Gated Recurrent Units (GRU), Hidden state size = 128
Input
Window 1
Base predictions
Fully connected layer
Gated Recurrent Units (GRU), Hidden state size = 128
Hidden state input
Gated Recurrent Units (GRU), Hidden state size = 128
Input
Window 2
Sliding window over a long sequence

Supplementary Figure B.8: PEPPER-SNP inference scheme.





Supplementary Figure B.9: PEPPER-HP haplotype specific image generation scheme. Each row describes an encoded feature and each column describes a reference position. The top summary is derived from reads with haplotag 1 (HP-1) and the bottom is derived from reads with haplotag 2 (HP-2).



Supplementary Figure B.10: PEPPER-HP haplotype-specific inference scheme.

Supplementary Results

B.0.1 Variant calling results

Sample Name	Type	Method	True positives	False negatives	False positives	Recall	Precision	F1-score
HG003	SNP	P-M-DV	3317032	10463	9958	0.9969	0.9970	0.9969
		Medaka	3293174	22716	26549	0.9931	0.9920	0.9926
		Clair	3266489	61006	31220	0.9817	0.9905	0.9861
		Longshot	3224643	102852	45780	0.9691	0.9860	0.9775
	INDEL	P-M-DV	303643	200858	29400	0.6019	0.9136	0.7257
		Medaka	313033	189639	69434	0.6227	0.8226	0.7089
		Clair	205364	299137	58367	0.4071	0.7812	0.5352
HG004	SNP	P-M-DV	3338882	7728	7474	0.9977	0.9978	0.9977
		Medaka	3286457	20743	23309	0.9937	0.9930	0.9933
		Clair	3285625	60985	32021	0.9818	0.9903	0.9860
		Longshot	3243183	103427	45387	0.9691	0.9862	0.9776
	INDEL	P-M-DV	300258	210261	32429	0.5881	0.9046	0.7128
		Medaka	306050	198390	88346	0.6067	0.7807	0.6828
		Clair	203825	306694	61454	0.3993	0.7708	0.5260

Supplementary Table B.1: Oxford nanopore variant calling performance comparison between Medaka, Clair, Longshot and PEPPER-Margin-DeepVariant (P-M-DV) on HG003 and HG004 with 90× coverage.

Sample	Ref. GRC	GIAB version	Type	Method	True pos.	False neg.	False pos.	Recall	Precision	F1-score
HG 005	h37	v3.3.2	SNP	P-M-DV	3036676	5947	11807	**0.9980**	**0.9961**	**0.9971**
				Medaka	3026174	16361	21414	0.9946	0.9930	0.9938
				Clair	2982163	60460	68238	0.9801	0.9776	0.9789
				Longshot	2980529	62094	57698	0.9796	0.9810	0.9803
			IN-DEL	P-M-DV	258720	131438	30008	**0.6631**	**0.8981**	**0.7629**
				Medaka	264300	125852	66019	0.6774	0.8043	0.7354
				Clair	175540	214617	54654	0.4499	0.7650	0.5666
	h38	v4.2.1 (draft)	SNP	P-M-DV	3269767	7563	9624	**0.9977**	**0.9971**	**0.9974**
				Medaka	3256035	21207	26871	0.9935	0.9918	0.9927
				Clair	3210942	66388	73550	0.9797	0.9776	0.9787
				Longshot	3189831	87499	64687	0.9733	0.9801	0.9767
			IN-DEL	P-M-DV	278726	138235	30957	**0.6685**	**0.9019**	**0.7678**
				Medaka	284407	132546	68363	0.6821	0.8100	0.7406
				Clair	187276	229685	56443	0.4491	0.7706	0.5675

Supplementary Table B.2: Oxford nanopore variant calling performance comparison between Medaka, Clair, Longshot and PEPPER-Margin-DeepVariant (P-M-DV) on HG005 sample between two reference (GRCh37 and GRCh38) and GIAB truth set (v3.3.2 and v4.2.1).

Confidence	Total Region Size	Paternal Concordance	Maternal Concordance	Concordance	Checked Records	Indeterminate Consistency	Mendelian Violations
GIAB High	2.504 Gb	$\frac{2899287}{2902160}$ (99.9%)	$\frac{2906561}{2909420}$ (99.9%)	$\frac{2257125}{2262783}$ (99.75%)	$\frac{3588024}{4791528}$ (74.88%)	$\frac{1323587}{3588024}$ (36.89%)	$\frac{7312}{3588024}$ (0.20%)
GIAB Low	0.315 Gb	$\frac{544760}{554736}$ (98.20%)	$\frac{536611}{548668}$ (97.80%)	$\frac{394059}{412533}$ (95.52%)	$\frac{775185}{1052572}$ (73.65%)	$\frac{356121}{775185}$ (45.94%)	$\frac{25005}{775185}$ (3.23%)

Supplementary Table B.3: Mendelian consistency for HG005, HG006, HG007 trio in and out of GIAB high confidence regions v4.2.1 using `rtg mendelian` on GRCh38. “Checked records” denotes output from the tool with description “Records were variant in at least 1 family member and checked for Mendelian constraints”, “Indeterminate Consistency” denotes output from the tool with description “Records had indeterminate consistency status due to incomplete calls”, and “Mendelian Violoations” denotes output from the tool with description “Records contained a violation of Mendelian constraints”. GIAB Low was generated by excluding GIAB’s high confidence BED from GRCh38 as well as centromeric regions.

HG003 coverage	**Method**	**True positives**	**False negatives**	**False positives**	**Recall**	**Precision**	**F1-score (INDEL)**
10x	P-M-DV	169072	335430	248485	0.3351	0.4074	0.3677
	Medaka	147181	351183	2222020	0.2953	0.0631	0.1039
	Clair	78015	426486	31587	0.1546	0.7133	0.2542
20x	P-M-DV	239619	264882	96644	0.4750	0.7164	0.5712
	Medaka	229787	268549	182403	0.4611	0.5628	0.5069
	Clair	142647	361854	40955	0.2827	0.7785	0.4148
30x	P-M-DV	265318	239183	64214	0.5259	0.8083	0.6372
	Medaka	264877	233132	119190	0.5319	0.6949	0.6025
	Clair	167998	336503	44982	0.3330	0.7905	0.4686
40x	P-M-DV	278902	225599	51381	0.5528	0.8472	0.6691
	Medaka	284431	217108	103944	0.5671	0.7373	0.6411
	Clair	180964	323537	48517	0.3587	0.7905	0.4935
50x	P-M-DV	288480	216021	43169	0.5718	0.8723	0.6908
	Medaka	297390	206752	91135	0.5899	0.7702	0.6681
	Clair	189538	314963	51278	0.3757	0.7891	0.5090
60x	P-M-DV	294414	210087	38056	0.5836	0.8878	0.7042
	Medaka	301161	198584	82479	0.6026	0.7896	0.6835
	Clair	195079	309422	53275	0.3867	0.7877	0.5187
70x	P-M-DV	298553	205948	34079	0.5918	0.8997	0.7139
	Medaka	306842	194792	76519	0.6117	0.8048	0.6951
	Clair	199070	305431	55055	0.3946	0.7856	0.5253
80x	P-M-DV	301312	203189	31269	0.5972	0.9079	0.7205
	Medaka	309376	191507	72591	0.6177	0.8142	0.7024
	Clair	202551	301950	56606	0.4015	0.7840	0.5310
90x	P-M-DV	303643	200858	29400	0.6019	0.9136	0.7257
	Medaka	313033	189639	69434	0.6227	0.8226	0.7089
	Clair	205364	299137	58367	0.4071	0.7812	0.5352

Supplementary Table B.4: Comparison on INDEL performance between Medaka, Clair and PEPPER-Margin-DeepVariant (P-M-DV) variant callers at different coverages of HG003 sample.

HG003 coverage	**Method**	**True positives**	**False negatives**	**False positives**	**Recall**	**Precision**	**F1-score (SNP)**
10x	P-M-DV	3015493	312002	2510007	0.9062	0.5458	0.6813
	Medaka	2998322	288605	8528164	0.9122	0.2602	0.4049
	Clair	2067633	1259862	585625	0.6214	0.7793	0.6914
20x	P-M-DV	3286124	41371	475385	0.9876	0.8736	0.9271
	Medaka	3228058	59716	369037	0.9818	0.8974	0.9377
	Clair	3026716	300779	229952	0.9096	0.9294	0.9194
30x	P-M-DV	3308068	19427	60871	0.9942	0.9819	0.9880
	Medaka	3248842	34884	59816	0.9894	0.9819	0.9856
	Clair	3194577	132918	121085	0.9601	0.9635	0.9618
40x	P-M-DV	3312504	14991	20633	0.9955	0.9938	0.9947
	Medaka	3279473	29155	39141	0.9912	0.9882	0.9897
	Clair	3237789	89706	80265	0.9730	0.9758	0.9744
50x	P-M-DV	3314808	12687	13806	0.9962	0.9959	0.9960
	Medaka	3298374	26404	33504	0.9921	0.9899	0.9910
	Clair	3254017	73478	58229	0.9779	0.9824	0.9802
60x	P-M-DV	3315655	11840	12062	0.9964	0.9964	0.9964
	Medaka	3271294	24807	30926	0.9925	0.9906	0.9916
	Clair	3260364	67131	46813	0.9798	0.9858	0.9828
70x	P-M-DV	3316257	11238	11217	0.9966	0.9966	0.9966
	Medaka	3283443	24010	28991	0.9927	0.9913	0.9920
	Clair	3263513	63982	39636	0.9808	0.9880	0.9844
80x	P-M-DV	3316750	10745	10219	0.9968	0.9969	0.9969
	Medaka	3280595	23263	27321	0.9930	0.9917	0.9924
	Clair	3265361	62134	34616	0.9813	0.9895	0.9854
90x	P-M-DV	3317032	10463	9958	0.9969	0.9970	0.9969
	Medaka	3293174	22716	26549	0.9931	0.9920	0.9926
	Clair	3266489	61006	31220	0.9817	0.9905	0.9861

Supplementary Table B.5: Comparison on SNP performance between Medaka, Clair and PEPPER-Margin-DeepVariant (P-M-DV) variant callers at different coverages of HG003 sample.

File	Read N50	Gb	Coverage
HG001	21443	309.88	93.91
HG002	50317	160.39	48.6
HG003	44550	277.38	84.05
HG004	47996	284.32	86.16
HG005	49297	182.53	55.31
HG006	50019	163.87	49.66
HG007	50423	132.75	40.23

Supplementary Table B.6: Sample-wise nanopore read coverage for seven Genome-In-A-Bottle (GIAB) samples.

Sample Name	Reference	Version	Regions covered by benchmark (bp)
HG001	GRCh37	v3.3.2	2437907771
HG002	GRCh38	v4.2.1	2542242843
HG003	GRCh38	v4.2.1	2528531102
HG004	GRCh38	v4.2.1	2524487531
HG005	GRCh37	v3.3.2	2376855757
HG006	GRCh37	v3.3.2	2393652163
HG007	GRCh37	v3.3.2	2394471248

Supplementary Table B.7: Details of Genome-In-A-Bottle truth set used for each genome.

Sample	**Type**	**Total truth**	**True positives**	**False negatives**	**False positives**	**Recall**	**Precision**	**F1-score**
HG001	SNP	3209309	3203740	5569	11466	0.9983	0.9964	0.9973
	INDEL	481841	292565	189276	37718	0.6072	0.8883	0.7213
HG003	SNP	3327495	3317032	10463	9958	0.9969	0.9970	0.9969
	INDEL	504501	303643	200858	29400	0.6019	0.9136	0.7257
HG004	SNP	3346610	3338882	7728	7474	0.9977	0.9978	0.9977
	INDEL	510519	300258	210261	32429	0.5881	0.9046	0.7128
HG005	SNP	3042623	3036676	5947	11807	0.9980	0.9961	0.9971
	INDEL	390158	258720	131438	30008	0.6631	0.8981	0.7629
HG006	SNP	3053660	3047013	6647	13802	0.9978	0.9955	0.9967
	INDEL	394727	242909	151818	30518	0.6154	0.8901	0.7277
HG007	SNP	3069407	3060423	8984	18351	0.9971	0.9940	0.9956
	INDEL	397103	236816	160287	34295	0.5964	0.8753	0.7094

Supplementary Table B.8: PEPPER-Margin-DeepVariant performance on six GIAB samples with Oxford nanopore data.

Sample	Variant Type	Method	True positives	False negatives	False positives	Recall	Precision	F1-score
HG003	SNP	P-M-DV	3317032	10463	9958	**0.9969**	**0.9970**	**0.9969**
		Medaka	3293174	22716	26549	0.9931	0.9920	0.9926
		PEPPER-HP	3311863	15632	27711	0.9953	0.9917	0.9935
	INDEL	P-M-DV	303643	200858	29400	**0.6019**	**0.9136**	**0.7257**
		Medaka	313033	189639	69434	0.6227	0.8226	0.7089
		PEPPER-HP	310722	193779	151334	0.6159	0.6784	0.6456
HG004	SNP	P-M-DV	3338882	7728	7474	**0.9977**	**0.9978**	**0.9977**
		Medaka	3286457	20743	23309	0.9937	0.9930	0.9933
		PEPPER-HP	3333967	12643	25961	0.9962	0.9923	0.9942
	INDEL	P-M-DV	300258	210261	32429	**0.5881**	**0.9046**	**0.7128**
		Medaka	306050	198390	88346	0.6067	0.7807	0.6828
		PEPPER-HP	307935	202584	190089	0.6032	0.6247	0.6137

Supplementary Table B.9: Oxford nanopore variant calling performance comparison between PEPPER-HP (tuned for balanced precision and recall), Medaka and PEPPER-Margin-DeepVariant on HG003 and HG004 with 90× coverage.

Sample	Haplotype-aware pipeline	Runtime hh:mm:ss	Cost USD($)	INDEL F1-Score	SNP F1-Score
HG003	DeepVariant-Whatshap-DeepVariant	15:11:52	$36.24	0.9942	0.9990
	DeepVariant-Margin-DeepVariant	08:03:43	$36.71	0.9945	0.9991
	PEPPER-Margin-DeepVariant	05:55:28	$26.99	0.9944	0.9990
HG004	DeepVariant-Whatshap-DeepVariant	15:47:29	$37.35	0.9940	0.9992
	DeepVariant-Margin-DeepVariant	08:26:36	$38.45	0.9942	0.9992
	PEPPER-Margin-DeepVariant	05:58:57	$27.26	0.9941	0.9992

Supplementary Table B.10: PacBio HiFi variant calling performance and runtime comparison between three haplotype-aware pipelines on 35× coverage HG003 and HG004 samples. For PEPPER, Margin and DeepVariant we used $4.56/h n1-standard-96 and for WhatsHap we used $0.09/h n1-standard-2 instance types on google cloud platform. The F1-scores are derived by comparing the variant calls against GIAB v4.2.1 benchmark variants for HG003 and HG004.

Sample	Method	CPUs	Memory	GPUs	Instance cost/h	Total runtime	Total cost
HG001 50x ONT	Longshot	16vCPUs	104 GB	-	$0.95	51:25:31	$48.84
	Clair	96vCPUs	360 GB	-	$4.56	02:30:05	$11.40
	Medaka	16vCPUs	104 GB	1x NVIDIA Tesla P100	$2.41	40:21:11	$97.24
		16vCPUs	104 GB	-	$0.95	95:14:01	$90.47
	PEPPER Margin DeepVariant	96vCPUs	360 GB	-	$4.56	12:59:19	$59.28
		96vCPUs	360 GB	4x NVIDIA Tesla P100	$10.4	6:41:56	$70
HG001 75x ONT	Longshot	32vCPUs	208 GB	-	$1.89	73:56:43	$139.73
	Clair	96vCPUs	360 GB	-	$4.56	03:05:46	$14.13
	Medaka	32vCPUs	206GB	2x NVIDIA Tesla P100	$4.81	46:58:11	$225.87
		32vCPUs	206GB	-	$1.50	116:41:04	$175.025
	PEPPER Margin DeepVariant	96vCPUs	360 GB	-	$4.56	14:44:40	$68.4
		96vCPUs	360 GB	4x NVIDIA Tesla P100	$10.4	9:05:01	$94.4

Supplementary Table B.11: Run-time and cost analysis of Oxford nanopore-based variant calling pipelines on 50x and 75x HG001 data. We used various `n1-series` instance types available on Google Cloud Platform (GCP). The we calculated the cost using the GCP cost calculator. Logs of all the runs are publicly available (See supplementary Notes).

Sample name	Pipeline	Type	True positives	False negatives	False positives	Recall	Precision	F1-score
HG003 35x	DeepVariant	INDEL	499277	5224	4901	0.9896	0.9907	0.9902
		SNP	3323609	3886	2883	0.9988	0.9991	0.9990
	DV-WH-DV	INDEL	501509	2992	2935	0.9941	0.9944	0.9942
		SNP	3323655	3840	2734	0.9988	0.9992	0.9990
	DV-M-DV	INDEL	501567	2934	2746	**0.9942**	**0.9948**	**0.9945**
		SNP	3323586	3909	1841	**0.9988**	**0.9994**	**0.9991**
	P-M-DV	INDEL	501539	2962	2816	0.9941	0.9946	0.9944
		SNP	3323607	3888	2501	0.9988	0.9992	0.9990
HG004 35x	DeepVariant	INDEL	504939	5580	5217	0.9891	0.9902	0.9896
		SNP	3343142	3468	2274	0.9990	0.9993	0.9991
	DV-WH-DV	INDEL	507288	3231	2966	0.9937	0.9944	0.9940
		SNP	3343074	3536	1771	0.9989	0.9995	0.9992
	DV-M-DV	INDEL	507351	3168	2846	**0.9938**	**0.9946**	**0.9942**
		SNP	3342966	3644	1491	**0.9989**	**0.9996**	**0.9992**
	P-M-DV	INDEL	507313	3206	2903	0.9937	0.9945	0.9941
		SNP	3342928	3682	1721	0.9989	0.9995	0.9992

Supplementary Table B.12: PacBio HiFi variant calling perfomance comparison between PEPPER-Margin-DeepVariant (P-M-DV), DeepVariant-WhatsHap-DeepVariant (DV-WH-DV), DeepVariant-Margin-DV (DV-M-DV), DeepVariant only.

Sample	Pipeline	SNP calling runtime	Phasing runtime	Variant calling runtime	Total runtime	Cost
HG003 35x PacBio HiFi	DeepVariant Whatshap DeepVariant	03:48:45 (n1-std-96)	07:32:35 (n1-std-2)	03:50:32 (n1-std-96)	15:11:52	$36.24
	DeepVariant Margin DeepVariant	03:48:45 (n1-std-96)	00:24:51 (n1-std-96)	03:50:07 (n1-std-96)	08:03:43	$36.71
	PEPPER Margin DeepVariant	1:28:49 (n1-std-96)	00:23:25 (n1-std-96)	4:03:14 (n1-std-96)	05:55:28	$26.99
HG004 35x PacBio HiFi	DeepVariant Whatshap DeepVariant	04:03:58 (n1-std-96)	07:44:56 (n1-std-2)	03:58:35 (n1-std-96)	15:47:29	$37.35
	DeepVariant Margin DeepVariant	04:03:58 (n1-std-96)	00:26:09 (n1-std-96)	03:56:29 (n1-std-96)	08:26:36	$38.45
	PEPPER Margin DeepVariant	1:25:03 (n1-std-96)	0:23:41 (n1-std-96)	4:10:13 (n1-std-96)	05:58:57	$27.26

Supplementary Table B.13: PacBio HiFi variant calling run-time comparison between three haplotype-aware pipelines on 35× coverage HG003 and HG004 samples. We used \$4.56/h `n1-standard-96 (n1-std-96)` and \$0.09/h `n1-standard-2 (n1-std-2)` instance types on google cloud platform for this analysis.

Sample	Coverage	Method	True positives	False negatives	False positives	Recall	Precision	F1-score
HG003	26.5x	P-M-DV	3278007	20943	50763	0.9937	0.9847	0.9892
		Longshot	3220875	53788	107929	0.9836	0.9676	0.9755
HG004	10.9x	P-M-DV	2862862	208230	484946	0.9322	0.8551	0.8920

Supplementary Table B.14: SNP Variant accuracy statistics for HG003 and HG004 against GIAB v4.2.1 on GRCh38 using PacBio CLR data

Sample	Variant type	Sequencing technology	True positives	False negatives	False positives	Recall	Precision	F1-score
HG003	SNP	Nanopore	3317032	10463	9958	0.9969	0.9970	0.9969
		Illumina	3307988	19508	4808	0.9941	0.9985	0.9963
		PacBio HiFi	3323607	3888	2501	0.9988	0.9992	0.9990
	INDEL	Nanopore	303643	200858	29400	0.6019	0.9136	0.7257
		Illumina	501546	2955	1276	0.9941	0.9976	0.9959
		PacBio HiFi	501539	2962	2816	0.9941	0.9946	0.9944
HG004	SNP	Nanopore	3338882	7728	7474	0.9977	0.9978	0.9977
		Illumina	3326040	20570	4476	0.9939	0.9987	0.9962
		PacBio HiFi	3342928	3682	1721	0.9989	0.9995	0.9992
	INDEL	Nanopore	300258	210261	32429	0.5881	0.9046	0.7128
		Illumina	507418	3101	1284	0.9939	0.9976	0.9958
		PacBio HiFi	507313	3206	2903	0.9937	0.9945	0.9941

Supplementary Table B.15: Variant calling performance comparison in all benchmark regions between Oxford Nanopore Technology (ONT), Illumina NovaSeq (Illumina) and PacBio HiFi sequencing technology. Illumina variant calls are generated with DeepVariant v1.1 and ONT and PacBio HiFi variant calls are generated with PEPPER-Margin-DeepVariant.

Region	Sample	Platform	Total truth	True positives	False negatives	False positives	Recall	Precision	F1-score
MHC (SNP)	HG003	ONT	19543	19437	106	56	0.9946	0.9971	0.9958
		Illumina	19544	19336	208	31	0.9894	0.9984	0.9939
		PacBio	19543	19391	152	39	0.9922	0.9980	0.9951
	HG004	ONT	19271	19181	90	42	0.9953	0.9978	0.9966
		Illumina	19271	18998	273	31	0.9858	0.9984	0.9921
		PacBio	19271	19112	159	12	0.9917	0.9994	0.9955
Seg. Dup. (SNP)	HG003	ONT	121960	119838	2122	2288	0.9826	0.9813	0.9819
		Illumina	121960	112293	9667	2905	0.9207	0.9748	0.9470
		PacBio	121960	119003	2957	818	0.9758	0.9932	0.9844
	HG004	ONT	122191	120107	2084	1693	0.9829	0.9861	0.9845
		Illumina	122191	112296	9895	2710	0.9190	0.9764	0.9469
		PacBio	122191	119713	2478	672	0.9797	0.9944	0.9870
Low map. (SNP)	HG003	ONT	192520	190380	2140	2152	0.9889	0.9888	0.9889
		Illumina	192520	174763	17757	3627	0.9078	0.9797	0.9423
		PacBio	192520	189453	3067	888	0.9841	0.9953	0.9897
	HG004	ONT	192653	190671	1982	1634	0.9897	0.9915	0.9906
		Illumina	192653	174196	18457	3510	0.9042	0.9803	0.9407
		PacBio	192653	190118	2535	653	0.9868	0.9966	0.9917
250bp+ non-unique (SNP)	HG003	ONT	13608	12594	1014	592	0.9255	0.9552	0.9401
		Illumina	13608	7420	6188	1377	0.5453	0.8436	0.6624
		PacBio	13608	11613	1995	380	0.8534	0.9684	0.9072
	HG004	ONT	13492	12615	877	413	0.9350	0.9683	0.9514
		Illumina	13492	7235	6257	1323	0.5362	0.8455	0.6563
		PacBio	13492	11847	1645	284	0.8781	0.9766	0.9247

Supplementary Table B.16: SNP performance in difficult-to-map regions with Illumina, PacBio HiFi and Oxford nanopore data.

Region	Sample	Platform	Total truth	True positives	False negatives	False positives	Recall	Precision	F1-score
H.poly. (7bp-11bp)	HG003	ONT	70736	68620	2116	2464	0.9701	0.9654	0.9677
		Illumina	70737	70632	105	60	0.9985	0.9992	0.9988
		PacBio	70736	70645	91	93	0.9987	0.9987	0.9987
	HG004	ONT	71141	68912	2229	2665	0.9687	0.9628	0.9657
		Illumina	71141	71045	96	39	0.9987	0.9995	0.9991
		PacBio	71142	71032	110	127	0.9985	0.9982	0.9983
H.Poly. 11bp+	HG003	ONT	12187	10708	1479	1359	0.8786	0.8879	0.8832
		Illumina	12188	12176	12	16	0.9990	0.9987	0.9989
		PacBio	12187	12005	182	203	0.9851	0.9841	0.9846
	HG004	ONT	12494	10751	1743	1441	0.8605	0.8823	0.8713
		Illumina	12494	12478	16	38	0.9987	0.9971	0.9979
		PacBio	12494	12332	162	182	0.9870	0.9861	0.9866
Di-Mer repeat (11bp-50bp)	HG003	ONT	18817	18322	495	668	0.9737	0.9654	0.9695
		Illumina	18817	18778	39	34	0.9979	0.9982	0.9981
		PacBio	18817	18763	54	120	0.9971	0.9939	0.9955
	HG004	ONT	18925	18417	508	676	0.9732	0.9652	0.9692
		Illumina	18925	18880	45	41	0.9976	0.9979	0.9978
		PacBio	18925	18873	52	109	0.9973	0.9945	0.9959
Tri-Mer repeat (15bp-50bp)	HG003	ONT	4179	4129	50	89	0.9880	0.9791	0.9835
		Illumina	4179	4172	7	3	0.9983	0.9993	0.9988
		PacBio	4179	4153	26	22	0.9938	0.9948	0.9943
	HG004	ONT	4213	4175	38	92	0.9910	0.9785	0.9847
		Illumina	4213	4210	3	2	0.9993	0.9995	0.9994
		PacBio	4213	4196	17	18	0.9960	0.9958	0.9959

Supplementary Table B.17: SNP performance in low-complexity regions with Illumina, PacBio HiFi and Oxford nanopore data.

Region	Sample	Type	Total truth	Platform	F1-score	Recall	Precision	True positives	False neg.	False pos.
Not in all difficult regions (76% genome fraction)	HG-003	SNP	2717833	Ilmn.	0.9997	0.9997	0.9997	2717119	713	796
				ONT	0.9988	0.9986	0.9989	2714093	3740	2857
				CCS	0.9999	0.9999	0.9998	2717567	265	416
		IN-DEL	155063	Ilmn.	0.9996	0.9995	0.9997	154988	74	54
				ONT	0.9719	0.9535	0.9910	147852	7211	1343
				CCS	0.9997	0.9997	0.9997	155023	42	39
	HG-004	SNP	2732800	Ilmn.	0.9998	0.9997	0.9998	2732050	750	600
				ONT	0.9996	0.9996	0.9997	2731662	1138	903
				CCS	0.9999	0.9998	0.9999	2732330	470	322
		IN-DEL	156444	Ilmn.	0.9997	0.9996	0.9998	156385	59	31
				ONT	0.9700	0.9503	0.9905	148676	7768	1429
				CCS	0.9997	0.9996	0.9998	156388	59	38

Supplementary Table B.18: Performance in not all difficult regions (easy regions) with Illumina, PacBio HiFi and Oxford nanopore data.

Region	Sample	Type	Total truth	Platform	F1-score	Recall	Precision	True pos.	False neg.	False pos.
Not in all tandem repeats and hom. (86% genome fraction)	HG-003	SNP	3161145	Ilmn.	0.9962	0.9939	0.9985	3141934	19209	4642
				ONT	0.9981	0.9980	0.9982	3154668	6477	5534
				CCS	0.9992	0.9989	0.9996	3157593	3550	1396
		INDEL	184877	Ilmn.	0.9962	0.9937	0.9987	183717	1159	237
				ONT	0.9704	0.9516	0.9900	175933	8944	1784
				CCS	0.9991	0.9989	0.9994	184667	212	105
	HG-004	SNP	3180117	Ilmn.	0.9961	0.9936	0.9986	3159831	20286	4291
				ONT	0.9990	0.9989	0.9991	3176662	3455	2774
				CCS	0.9993	0.9990	0.9997	3176818	3299	1068
		INDEL	186340	Ilmn.	0.9961	0.9935	0.9988	185125	1215	226
				ONT	0.9684	0.9479	0.9898	176633	9707	1825
				CCS	0.9992	0.9988	0.9995	186120	224	92
In all tandem repeats and hom. (4% genome fraction)	HG-003	SNP	166352	Ilmn.	0.9986	0.9982	0.9990	166054	299	166
				ONT	0.9748	0.9760	0.9736	162364	3988	4425
				CCS	0.9976	0.9978	0.9974	165994	358	445
		INDEL	320072	Ilmn.	0.9957	0.9944	0.9969	318271	1800	1039
				ONT	0.5401	0.4002	0.8305	128093	191979	27622
				CCS	0.9918	0.9915	0.9922	317355	2719	2663
	HG-004	SNP	166493	Ilmn.	0.9986	0.9983	0.9989	166209	284	185
				ONT	0.9731	0.9743	0.9720	162220	4273	4700
				CCS	0.9977	0.9979	0.9974	166147	346	440
		INDEL	324622	Ilmn.	0.9956	0.9942	0.9969	322730	1892	1059
				ONT	0.5193	0.3820	0.8108	123993	200629	30601
				CCS	0.9914	0.9908	0.9920	321655	2971	2774

Supplementary Table B.19: Performance comparson of Illumina, PacBio HiFi and Oxford nanopore data in repeat and non-repeat regions.

Data	Tool	Phased Variants	Unphased Variants	Blocks	Median Variants per Block	Average Variants per Block	Median BP per Block	Average BP per Block	Block N50
ONT 25x	Margin	2293009	1008276	2536	347	904	503808	1056597	2067806
	WhatsHap	2452395	849215	2297	395	1068	523694	1177941	2372651
ONT 50x	Margin	2275697	875317	1376	613	1654	853602	1993709	4211518
	WhatsHap	2391670	759715	1172	822	2041	1049089	2355537	4900234
ONT 75x	Margin	2091713	1023259	1167	496	1792	769148	2372510	6126250
	WhatsHap	2393421	722297	812	955	2948	1167915	3430964	8266083
HiFi 35x	Margin	2327420	1035855	14069	15	165	48362	154095	242226
	WhatsHap	2412900	954503	14061	14	172	48362	155745	252972

Supplementary Table B.20: Details of Margin and WhatsHap phasing output on HG001 sample with Oxford Nanopore (ONT) and PacBio HiFi data. Results are generated with `whatshap stats` command.

Data	Tool	Data	Tool	Assessed Pairs	Switches	Switch Rate	Hamming	Hamming Rate
ONT 25x	Margin	ONT 25x	Margin	1901418	16639	0.00875	162341	0.0854
	WhatsHap	ONT 25x	WhatsHap	1917571	17696	0.00923	177660	0.0926
ONT 50x	Margin	ONT 50x	Margin	1895721	16252	0.00857	195897	0.1033
	WhatsHap	ONT 50x	WhatsHap	1926257	17513	0.00909	161079	0.0836
ONT 75x	Margin	ONT 75x	Margin	1759253	14356	0.00816	174052	0.0989
	WhatsHap	ONT 75x	WhatsHap	1927665	17462	0.00906	179655	0.0932
HiFi 35x	Margin	HiFi 35x	Margin	1908770	17077	0.00895	24187	0.0127
	WhatsHap	HiFi 35x	WhatsHap	1914368	17801	0.00930	28973	0.0151

Supplementary Table B.21: Comparison of Margin and WhatsHap phasesets of HG001 sample with Oxford Nanopore (ONT) and PacBio HiFi data. Comparison is performed with `whatshap compare` command.

Data	Tool	Average Accuracy	Average Reads per 1kb	Average Tagged Reads per 1kb
ONT 55x Chr1	Margin	96.26	56.9	56.9
	WhatsHap	95.71	56.9	56.9
CCS 35x Chr1	Margin	98.00	35.5	35.5
	WhatsHap	97.99	35.5	35.5

Supplementary Table B.22: Haplotagging results comparing Margin and WhatsHap on an Admixed sample with an approximately equal amount of reads from the maternal haplotypes of HG005 and HG02723. Accuracy is determined for each kilobase bucket by comparing the number of direct-matched reads R_d (truth H1 to tagged H1 or truth H2 to tagged H2) and cross-matched reads R_c (truth H1 to tagged H2 or truth H2 to tagged H1) and calculating $\max(R_c, R_d)/(R_c + R_d)$, then averaging this value across all buckets in the HG003 high confidence regions.

Sample	Variant Type	Variant caller	True positives	False negatives	False positives	Recall	Precision	F1-score
HG003	SNP	P-M-DV	3317032	10463	9958	0.9969	0.9970	0.9969
		P-WH-DV	3316452	11043	11716	0.9967	0.9965	0.9966
	INDEL	P-M-DV	303643	200858	29400	0.6019	0.9136	0.7257
		P-WH-DV	301732	202769	29507	0.5981	0.9128	0.7227
HG004	SNP	P-M-DV	3338882	7728	7474	0.9977	0.9978	0.9977
		P-WH-DV	3338354	8256	10522	0.9975	0.9969	0.9972
	INDEL	P-M-DV	300258	210261	32429	0.5881	0.9046	0.7128
		P-WH-DV	298389	212130	32383	0.5845	0.9041	0.7100

Supplementary Table B.23: Oxford Nanopore variant calling perfomance comparison between PEPPER-Margin-DeepVariant (P-M-DV) and PEPPER-WhatsHap-DeepVariant (P-WH-DV) only.

Data	Tool	Module	Max Threads	Max Mem.	Runtime (min)	Instance Type	Instance Cost ($/hr)	Cost ($)
ONT 25x	Margin	Haplotag	64	20	21	n1-highcpu-64	2.267	0.79
		Phase VCF	64	15	15	n1-highcpu-64	2.267	0.56
		Total	–	–	36	n1-highcpu-64	2.267	1.36
	WhatsHap	Phase	2	3	347	n1-standard-2	0.095	0.54
		Haplotag	2	3	247	n1-standard-2	0.095	0.39
		Total	–	–	941	n1-standard-2	0.095	1.48
ONT 50x	Margin	Haplotag	64	28	54	n1-highcpu-64	2.267	2.04
		Phase VCF	64	18	30	n1-highcpu-64	2.267	1.13
		Total	–	–	84	n1-highcpu-64	2.267	3.17
	WhatsHap	Phase	2	3	446	n1-standard-2	0.095	0.7
		Haplotag	2	3	444	n1-standard-2	0.095	0.7
		Total	–	–	1336	n1-standard-2	0.095	2.11
ONT 75x	Margin	Haplotag	64	35	80	n1-highcpu-64	2.267	3.02
		Phase VCF	64	22	43	n1-highcpu-64	2.267	1.62
		Total	–	–	123	n1-highcpu-64	2.267	4.64
	WhatsHap	Phase	2	3	522	n1-standard-2	0.095	0.82
		Haplotag	2	3	644	n1-standard-2	0.095	1.01
		Total	–	–	1688	n1-standard-2	0.095	2.67
HiFi 35x	Margin	Haplotag	64	19	19	n1-highcpu-64	2.267	0.71
		Phase VCF	64	18	14	n1-highcpu-64	2.267	0.52
		Total	–	–	33	n1-highcpu-64	2.267	1.24
	WhatsHap	Phase	2	3	277	n1-standard-2	0.095	0.43
		Haplotag	2	3	210	n1-standard-2	0.095	0.33
		Total	–	–	764	n1-standard-2	0.095	1.2

Supplementary Table B.24: Margin/WhatsHap Runtimes. Total runtimes are sum of Haplotag and Phase VCF runtimes for Margin, and sum of 2x Phase and 1x Haplotag for WhatsHap, as `whatshap haplotag` requires a phased VCF.

Type	Data	Gene Type	Subset	Recall	Precision	F1 Score
SNP	Nanopore	all regions	all regions	0.998169	0.996314	0.997241
		all genes	all genes	0.998097	0.996481	0.997289
		protein coding	all cds	0.998641	0.997887	0.998263
			all exons	0.998158	0.997675	0.997916
			all genes	0.99799	0.996751	0.99737
	PacBio HiFi	all regions	all regions	0.999391	0.998062	0.998726
		all genes	all genes	0.999384	0.998197	0.99879
		protein coding	all cds	0.999446	0.998994	0.99922
			all exons	0.999502	0.999117	0.99931
			all genes	0.999382	0.998441	0.998912
INDEL	Nanopore	all regions	all regions	0.60077	0.878512	0.713567
		all genes	all genes	0.595042	0.877943	0.709325
		protein coding	all cds	0.799544	0.926893	0.858522
			all exons	0.632435	0.896594	0.741696
			all genes	0.584914	0.876731	0.701692
	PacBio HiFi	all regions	all regions	0.948736	0.92602	0.937241
		all genes	all genes	0.947847	0.922887	0.935201
		protein coding	all cds	0.984055	0.909278	0.94519
			all exons	0.955149	0.927624	0.941186
			all genes	0.946128	0.918991	0.932362

Supplementary Table B.25: Accuracy stats for ONT and CCS calls made on GRCh37 with HG001 data in high confidence regions against GIAB v3.3.2 stratified by all gene and protein coding gene, further stratified by whole gene, exon, CDS as annotated by GENCODE v35lift37. CDS regions are coding sequences, and include start and stop codons for this analysis.

Gene Region	Subset	Subset Size	Subset High Confidence Size	High Confidence Ratio	High Confidence Whole Genome Ratio
Genome	–	2951332653	2579466415	0.874001	0.874001
All Genes	–	1982798080	1591767788	0.802789	0.539339
Protein Coding	Coding Sequence	114906140	31986772	0.278373	0.010838
Protein Coding	Exon	283314507	92457254	0.326341	0.031327
Protein Coding	Gene Regions	1367165648	1201166019	0.878581	0.406991

Supplementary Table B.26: Size of GENCODE Gene Regions

Phasing Coverage	Switch Errors Present	SNP, INDEL Errors	Gene Region	Count	25 Quartile Gene Size	Median Gene Size	75 Quartile Gene Size
wholly	no error	no error	gene	1738	1438	3332	8005
			exon	3121	2845	10303	34848
			cds	3481	3163	11478	39123
		error	gene	1764	15676	35729	80335
			exon	381	7533	27218	67507
			cds	21	3161	5190	31102
	error	no error	gene	15	1685	4868	15383
			exon	33	8703	19682	86981
			cds	37	8703	20549	72326
		error	gene	23	18072	63546	107182
			exon	5	7680	47627	63546
			cds	1	7680	7680	7680
partially	no error	no error	gene	6	4760	10482	36920
			exon	29	25339	102612	147661
			cds	37	25339	102612	161893
		error	gene	31	44745	110698	223634
			exon	8	54807	99792	318869
			cds	0	–	–	–
	error	all	all	0	–	–	–
not	–	no error	gene	125	1769	3060	8819
			exon	201	2352	8744	29504
			cds	214	2478	9365	29958
		error	gene	91	19316	33429	69811
			exon	15	10690	23836	35712
			cds	2	44794	53073	61351

Supplementary Table B.27: Gencode protein coding genes with coding sequence (CDS, start_codon, and stop_codon) 80% spanned by high confidence stratified by how phased it is by Margin, whether there were switch errors, whether there were SNP or INDEL errors, and gene region for HG001 with 75x Nanopore data on GRCh37. Three gene length quartiles are presented for the groupings.

Phasing Coverage	Switch Errors Present	SNP, INDEL Errors	Gene Region	Count	25 Quartile Gene Size	Median Gene Size	75 Quartile Gene Size
wholly	no error	no error	gene	2086	2184	5907	17294
			exon	2446	2586	8471	27219
			cds	2474	2615	8536	27351
		error	gene	390	23230	51309	102861
			exon	30	7098	15068	65957
			cds	2	60929	108132	155335
	error	no error	gene	18	1690	6798	11879
			exon	23	2321	8703	23001
			cds	24	2567	9247	19731
		error	gene	6	13297	23861	43535
			exon	1	12242	12242	12242
			cds	0	–	–	–
partially	no error	no error	gene	190	13338	34319	67099
			exon	354	28087	68206	146543
			cds	360	28289	68510	145331
		error	gene	170	76695	137702	255101
			exon	6	55573	75267	104898
			cds	0	–	–	–
	error	no error	gene	2	20915	21281	21647
			exon	5	22014	109387	127176
			cds	5	22014	109387	127176
		error	gene	3	118281	127176	138247
			exon	0	–	–	–
			cds	0	–	–	–
not	no error	no error	gene	741	2565	8419	23809
			exon	917	3533	13187	43784
			cds	928	3639	13351	43785
		error	gene	187	30101	75166	156763
			exon	11	10504	29953	68614
			cds	0	–	–	–

Supplementary Table B.28: Gencode protein coding genes with coding sequence (CDS, start_codon, and stop_codon) 80% spanned by high confidence stratified by how phased it is by Margin, whether there were switch errors, whether there were SNP or INDEL errors, and gene region for HG001 with 35x PacBio HiFi data on GRCh37. Three gene length quartiles are presented for the groupings.

Sample	Assembler	Polisher	Assembly haplotype	NG50	Estimated QV YAK (k=31)	Switch error rate	Hamming error
HG005	Flye	-	Haploid	37254637	31.08	0.146333	0.319502
	Shasta	-	Haploid	39831103	32	0.16431	0.293283
		P-M-DV (ONT)	HP-1	39820763	35.06	0.058178	0.207221
			HP-2	39820481	35.06	0.059199	0.218271
		P-M-DV (PacBio HiFi)	HP-1	39808277	43.54	0.028165	0.26687
			HP-2	39809097	43.5	0.028253	0.264903
	Trio-hifiasm	-	mat	51324672	51.81	0.007056	0.009601
			pat	50669010	51.72	0.003106	0.004542
HG00733	Flye	-	Haploid	36602095	31.93	0.226708	0.455478
	Shasta	-	Haploid	42512208	32.7	0.263731	0.4387
		P-M-DV (ONT)	HP-1	42497702	35.83	0.09903	0.319373
			HP-2	42498275	35.84	0.098502	0.320807
		P-M-DV (PacBio HiFi)	HP-1	42475072	43.83	0.050551	0.401146
			HP-2	42476106	43.85	0.049385	0.406129
	Trio-hifiasm	-	mat	32479553	53.6	0.0102	0.012044
			pat	35318917	53.35	0.010144	0.010069
HG02723	Flye	-	Haploid	39652856	31.88	0.24692	0.454764
	Shasta	-	Haploid	49185987	32.52	0.28754	0.424615
		P-M-DV (ONT)	HP-1	49165039	35.8	0.104264	0.248018
			HP-2	49164831	35.79	0.103455	0.238674
		P-M-DV (PacBio HiFi)	HP-1	49146102	43.46	0.046367	0.365246
			HP-2	49143792	43.38	0.046215	0.363784
	Trio-hifiasm	-	mat	19737990	56.27	0.005794	0.007677
			pat	22214675	55.94	0.006683	0.009111

Supplementary Table B.29: Diploid assembly polishing results of PEPPER-Margin-DeepVariant (P-M-DV) pipeline on HG005, HG00733 and HG02723 samples. We report estimated quality value (QV), switch error rate and hamming error using YAK assembly assessment tool.

Sample	Assembler	Polisher	Estimated QV YAK (k=31)
CHM13 chrX	Flye	-	32.85
	Shasta	-	34.601
		P-M-DV (ONT)	36.91
		P-M-DV (PacBio HiFi)	42.765
	Hifiasm	-	53.039

Supplementary Table B.30: Haploid assembly polishing results of PEPPER-Margin-DeepVariant (P-M-DV) pipeline on CHM13-chrX. We report estimated quality value (QV) using YAK assembly assessment tool.

Genome	Test SV set	Truth SV set	Hom. SV recall	Het. SV recall	Precision
HG002	Shasta	GIAB curated	94.4%	46.7%	80.1%
	PEPPER-Margin-DV		94.9%	49.0%	81.1%
	hifiasm		97.8%	97.0%	93.0%
	Shasta	hifiasm	95.2%	48.7%	83.2%
	PEPPER-Margin-DV		96.0%	50.9%	84.7%
HG005	Shasta	hifiasm	93.7%	49.6%	82.3%
	PEPPER-Margin-DV		95.2%	51.7%	84.0%
HG00733	Shasta	hifiasm	95.1%	47.7%	80.1%
	PEPPER-Margin-DV		95.9%	49.7%	81.7%
HG02733	Shasta	hifiasm	94.6%	48.7%	80.7%
	PEPPER-Margin-DV		95.8%	51.3%	82.4%

Supplementary Table B.31: Evaluation of the accuracy and completeness of SV reconstruction of Shasta and PEPPER-Margin-DeepVariant assemblies. Recall and precision were computed using the SVbenchmark tool inside the Tier1 high-confidence regions defined in the HG002 curated set of SVs. Since the set of curated SVs was only available for the HG002 genome, for the remaining genomes SVs recovered from the hifiasm assemblies were used as reference.

Bibliography

[1] 1000 Genomes Consortium. A global reference for human genetic variation. *Nature*, 526(7571):68–74, sep 2015.

[2] Derek Aguiar and Sorin Istrail. Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *Journal of Computational Biology*, 19(6):577–590, 2012.

[3] Derek Aguiar and Sorin Istrail. Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics*, 29(13):i352–i360, 2013.

[4] Can Alkan, Bradley P Coe, and Evan E Eichler. Genome structural variation discovery and genotyping. *Nature Reviews Genetics*, 12(5):363, 2011.

[5] Nicolas Altemose, Karen H. Miga, Mauro Maggioni, and Huntington F. Willard. Genomic Characterization of Large Heterochromatic Gaps in the Human Genome Assembly. *PLoS Computational Biology*, 10(5):e1003628, May 2014.

[6] Stephen F Altschul and Bruce W Erickson. Optimal sequence alignment using affine gap costs. *Bulletin of mathematical biology*, 48(5-6):603–616, 1986.

[7] Peter F Arndt, Terence Hwa, and Dmitri A Petrov. Substantial regional variation in substitution rates in the human genome: importance of gc content, gene density, and telomere-specific effects. *Journal of molecular evolution*, 60(6):748–763, 2005.

[8] Peter A Audano, Arvis Sulovari, Tina A Graves-Lindsay, Stuart Cantsilieris, Melanie Sorensen, AnneMarie E Welch, Max L Dougherty, Bradley J Nelson, Ankeeta Shah, Susan K Dutcher, et al. Characterizing the major structural variant alleles of the human genome. *Cell*, 176(3):663–675, 2019.

[9] Gunjan Baid, Maria Nattestad, Alexey Kolesnikov, Sidharth Goel, Howard Yang, Pi-Chuan Chang, and Andrew Carroll. An extensive sequence dataset of gold-standard samples for benchmarking and development. *bioRxiv*, 2020.

[10] Vikas Bansal and Vineet Bafna. Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, 24(16):i153–i159, 2008.

[11] Serafim Batzoglou, David B Jaffe, Ken Stanley, Jonathan Butler, Sante Gnerre, Evan Mauceli, Bonnie Berger, Jill P Mesirov, and Eric S Lander. Arachne: a whole-genome shotgun assembler. *Genome research*, 12(1):177–189, 2002.

[12] Jon-Matthew Belton, Rachel Patton McCord, Johan Harmen Gibcus, Natalia Naumova, Ye Zhan, and Job Dekker. Hi–c: a comprehensive technique to capture the conformation of genomes. *Methods*, 58(3):268–276, 2012.

[13] David R Bentley, Shankar Balasubramanian, Harold P Swerdlow, Geoffrey P Smith, John Milton, Clive G Brown, Kevin P Hall, Dirk J Evers, Colin L Barnes, Helen R Bignell, et al. Accurate whole human genome sequencing using reversible terminator chemistry. *nature*, 456(7218):53–59, 2008.

[14] Emily Berger, Deniz Yorukoglu, Jian Peng, and Bonnie Berger. Haptree: a novel bayesian framework for single individual polyplotyping using ngs data. *PLoS computational biology*, 10(3):e1003502, 2014.

[15] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623, 2015.

[16] Keith R Bradnam, Joseph N Fass, Anton Alexandrov, Paul Baranay, Michael Bechner, Inanç Birol, Sébastien Boisvert, Jarrod A Chapman, Guillaume Chapuis, Rayan Chikhi, et al. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *GigaScience*, 2(1):10, 2013.

[17] D. Y. Brandt, V. R. Aguiar, B. D. Bitarello, K. Nunes, J. Goudet, and D. Meyer. Mapping Bias Overestimates Reference Allele Frequencies at the HLA Genes in the 1000 Genomes Project Phase I Data. *G3 (Bethesda)*, 5(5):931–941, Mar 2015.

[18] Andrei Z Broder. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, pages 21–29. IEEE, 1997.

[19] Sharon R Browning and Brian L Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

[20] T H Cantor and C R Cantor. Evolution of protein molecules. *Mammalian protein metabolism*, 1:22–123, 1969.

[21] Mark J P Chaisson, Ashley D Sanders, Xuefang Zhao, et al. Multi-platform discovery of haplotype-resolved structural variation in human genomes. *bioRxiv*, page 193144, September 2017.

[22] Mark JP Chaisson, Ashley D Sanders, Xuefang Zhao, Ankit Malhotra, David Porubsky, Tobias Rausch, Eugene J Gardner, Oscar L Rodriguez, Li Guo, Ryan L Collins, et al. Multi-platform discovery of haplotype-resolved structural variation in human genomes. *Nature communications*, 10, 2019.

[23] Mark JP Chaisson, Richard K Wilson, and Evan E Eichler. Genetic variation and the de novo assembly of human genomes. *Nature Reviews Genetics*, 16(11):627–640, 2015.

[24] Haoyu Cheng, Gregory T Concepcion, Xiaowen Feng, Haowen Zhang, and Heng Li. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nature Methods*, pages 1–6, 2021.

[25] Haoyu Cheng, Erich D Jarvis, Olivier Fedrigo, Klaus-Peter Koepfli, Lara Urban, Neil J Gemmell, and Heng Li. Robust haplotype-resolved assembly of diploid individuals without parental data. *arXiv preprint arXiv:2109.04785*, 2021.

[26] Chen-Shan Chin, David H Alexander, Patrick Marks, Aaron A Klammer, James Drake, Cheryl Heiner, Alicia Clum, Alex Copeland, John Huddleston, Evan E Eichler, et al. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 10(6):563, 2013.

[27] Chen-Shan Chin, Paul Peluso, Fritz J Sedlazeck, et al. Phased diploid genome assembly with single-molecule real-time sequencing. *Nature methods*, 13(12):1050, 2016.

[28] Chen-Shan Chin, Justin Wagner, Qiandong Zeng, Erik Garrison, Shilpa Garg, Arkarachai Fungtammasan, Mikko Rautiainen, Sergey Aganezov, Melanie Kirsche, Samantha Zarate, et al. A diploid assembly-based benchmark for variants in the major histocompatibility complex. *Nature communications*, 11(1):1–9, 2020.

[29] Rudi Cilibrasi, Leo van Iersel, Steven Kelk, and John Tromp. The Complexity of the Single Individual SNP Haplotyping Problem. *Algorithmica*, 49(1):13–36, August 2007.

[30] John G. Cleary, Ross Braithwaite, Kurt Gaastra, et al. Comparing variant call files for performance benchmarking of next-generation sequencing variant calling pipelines. *bioRxiv*, 2015.

[31] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56, 2012.

[32] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, Mark A DePristo, Robert E Handsaker, Gerton Lunter, Gabor T Marth, Stephen T Sherry, et al. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011.

[33] David Deamer, Mark Akeson, and Daniel Branton. Three decades of nanopore sequencing. *Nature biotechnology*, 34(5):518–524, 2016.

[34] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids.* Cambridge university press, 1998.

[35] Michael A Eberle, Epameinondas Fritzilas, Peter Krusche, et al. A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome research*, 27(1):157–164, jan 2017.

[36] Peter Ebert, Peter A Audano, Qihui Zhu, Bernardo Rodriguez-Martin, David Porubsky, Marc Jan Bonder, Arvis Sulovari, Jana Ebler, Weichen Zhou, Rebecca Serra Mari, et al. Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science*, 372(6537), 2021.

[37] Jana Ebler, Marina Haukness, Trevor Pesout, Tobias Marschall, and Benedict Paten. Haplotype-aware diplotyping from noisy long reads. *Genome biology*, 20(1):116, 2019.

[38] Jana Ebler, Alexander Schönhuth, and Tobias Marschall. Genotyping inversions and tandem duplications. *Bioinformatics*, 33(24):4015–4023, 2017.

[39] Peter Edge, Vineet Bafna, and Vikas Bansal. Hapcut2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome research*, 27(5):801–812, 2017.

[40] Peter Edge and Vikas Bansal. Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing. *Nature communications*, 10(1):1–10, 2019.

[41] Evan E Eichler, Royden A Clark, and Xinwei She. An assessment of the sequence gaps: unfinished business in a finished human genome. *Nature Reviews Genetics*, 5(5):345, 2004.

[42] John Eid, Adrian Fehr, Jeremy Gray, Khai Luong, John Lyle, Geoff Otto, Paul Peluso, David Rank, Primo Baybayan, Brad Bettman, et al. Real-time dna sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[43] Philipp Euskirchen, Franck Bielle, Karim Labreche, Wigard P Kloosterman, Shai Rosenberg, Mailys Daniau, Charlotte Schmitt, Julien Masliah-Planchon, Franck Bourdeaut, Caroline Dehais, et al. Same-day genomic and epigenomic diagnosis of brain tumors using real-time nanopore sequencing. *Acta neuropathologica*, 134(5):691–703, 2017.

[44] Adam D Ewing, Nathan Smits, Francisco J Sanchez-Luque, Jamila Faivre, Paul M Brennan, Sandra R Richardson, Seth W Cheetham, and Geoffrey J Faulkner. Nanopore sequencing enables comprehensive transposable element epigenomic profiling. *Molecular Cell*, 80(5):915–928, 2020.

[45] Ester Falconer and Peter M Lansdorp. Strand-seq: a unifying tool for studies of chromosome segregation. In *Seminars in cell & developmental biology*, volume 24, pages 643–652. Elsevier, 2013.

[46] Ian T. Fiddes, Joel Armstrong, Mark Diekhans, Stefanie Nachtweide, Zev N. Kronenberg, Jason G. Underwood, David Gordon, Dent Earl, Thomas Keane, and Evan E. et al. Eichler. Comparative annotation toolkit (cat)âĂŤsimultaneous clade and personal genome annotation. *Genome Research*, 28(7):1029–1038, 2018.

[47] Ian T Fiddes, Gerrald A Lodewijk, Meghan Mooring, Colleen M Bosworth, Adam D Ewing, Gary L Mantalas, Adam M Novak, Anouk van den Bout, Alex Bishara, Jimi L Rosenkrantz, et al. Human-specific notch2nl genes affect notch signaling and cortical neurogenesis. *Cell*, 173(6):1356–1369, 2018.

[48] Sarah O Fischer and Tobias Marschall. Selecting reads for haplotype assembly. *bioRxiv*, page 046771, 2016.

[49] Adam Frankish, Mark Diekhans, Anne-Maud Ferreira, Rory Johnson, Irwin Jungreis, Jane Loveland, Jonathan M Mudge, Cristina Sisu, James Wright, Joel Armstrong, et al. Gencode reference annotation for the human and mouse genomes. *Nucleic acids research*, 47(D1):D766–D773, 2018.

[50] Adam Frankish, Mark Diekhans, Anne-Maud Ferreira, Rory Johnson, Irwin Jungreis, Jane Loveland, Jonathan M Mudge, Cristina Sisu, James Wright, Joel Armstrong, et al. Gencode reference annotation for the human and mouse genomes. *Nucleic acids research*, 47(D1):D766–D773, 2019.

[51] Shilpa Garg, Arkarachai Fungtammasan, Andrew Carroll, Mike Chou, Anthony Schmitt, Xiang Zhou, Stephen Mac, Paul Peluso, Emily Hatas, Jay Ghurye, et al. Chromosome-scale, haplotype-resolved assembly of human genomes. *Nature biotechnology*, 39(3):309–312, 2021.

[52] Shilpa Garg, Mikko Rautiainen, Adam M Novak, Erik Garrison, Richard Durbin, and Tobias Marschall. A graph-based approach to diploid genome assembly. *Bioinformatics*, 34(13):i105–i114, 2018.

[53] Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*, 2012.

[54] Gustavo Glusman, Hannah C Cox, and Jared C Roach. Whole-genome haplotyping approaches and genomic medicine. *Genome medicine*, 6(9):1–16, 2014.

[55] Sante Gnerre, Iain MacCallum, Dariusz Przybylski, Filipe J Ribeiro, Joshua N Burton, Bruce J Walker, Ted Sharpe, Giles Hall, Terrance P Shea, Sean Sykes, et al. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, 108(4):1513–1518, 2011.

[56] John E Gorzynski, Sneha D Goenka, Kishwar Shafin, Tanner D Jensen, Dianna G Fisk, Megan E Grove, Elizabeth Spiteri, Trevor Pesout, Jean Monlong, Gunjan Baid, et al. Ultrarapid nanopore genome sequencing in a critical care setting. *The New England journal of medicine*, 2022.

[57] Osamu Gotoh. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708, 1982.

[58] Osamu Gotoh. Optimal sequence alignment allowing for long gaps. *Bulletin of mathematical biology*, 52(3):359–373, 1990.

[59] Harvey J Greenberg, William E Hart, and Giuseppe Lancia. Opportunities for Combinatorial Optimization in Computational Biology. *INFORMS Journal on Computing*, 16(3):211–231, August 2004.

[60] Fei Guo, Dan Wang, and Lusheng Wang. Progressive approach for SNP calling and haplotype assembly using single molecular sequencing data. *Bioinformatics*, February 2018.

[61] J. Harrow, A. Frankish, J. M. Gonzalez, E. Tapanari, M. Diekhans, F. Kokocinski, B. L. Aken, D. Barrell, A. Zadissa, and S. et al. Searle. Gencode: The reference human genome annotation for the encode project. *Genome Research*, 22(9):1760–1774, 2012.

[62] Jennifer Harrow, Adam Frankish, Jose M Gonzalez, et al. Gencode: the reference human genome annotation for the encode project. *Genome research*, 22(9):1760–1774, 2012.

[63] Dan He, Arthur Choi, Knot Pipatsrisawat, Adnan Darwiche, and Eleazar Eskin. Optimal algorithms for haplotype assembly from whole-genome sequence data. *Bioinformatics*, 26(12):i183–i190, 2010.

[64] Jayne Y. Hehir-Kwa, Tobias Marschall, et al. A high-quality human reference panel reveals the complexity and distribution of genomic structural variants. *Nature communications*, 7:12989, 2016.

[65] David Heller and Martin Vingron. Svim-asm: Structural variant detection from haploid and diploid genome assemblies. *bioRxiv*, 2020.

[66] Xiaoqiu Huang, Jianmin Wang, Srinivas Aluru, Shiaw-Pyng Yang, and LaDeana Hillier. Pcap: a whole-genome assembly program. *Genome research*, 13(9):2164–2170, 2003.

[67] John Huddleston, Mark JP Chaisson, Karyn Meltz Steinberg, Wes Warren, Kendra Hoekzema, David Gordon, Tina A Graves-Lindsay, Katherine M Munson, Zev N Kronenberg, Laura Vives, et al. Discovery and genotyping of structural variation from long-read haploid genome sequence data. *Genome research*, 27(5):677–685, 2017.

[68] Chirag Jain, Arang Rhie, Nancy Hansen, Sergey Koren, and Adam M Phillippy. A long read mapping method for highly repetitive reference sequences. *bioRxiv*, 2020.

[69] Miten Jain, Ian T Fiddes, Karen H Miga, Hugh E Olsen, Benedict Paten, and Mark Akeson. Improved data analysis for the minion nanopore sequencer. *Nature methods*, 12(4):351, 2015.

[70] Miten Jain, Sergey Koren, Karen H Miga, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature biotechnology*, 2018.

[71] Miten Jain, Hugh E Olsen, Daniel J Turner, David Stoddart, Kira V Bulazel, Benedict Paten, David Haussler, Huntington F Willard, Mark Akeson, and Karen H Miga. Linear assembly of a human centromere on the y chromosome. *Nature biotechnology*, 36(4):321, 2018.

[72] Latha Kadalayil, Sajjad Rafiq, Matthew JJ Rose-Zerilli, Reuben J Pengelly, Helen Parker, David Oscier, Jonathan C Strefford, William J Tapper, Jane Gibson, Sarah Ennis, et al. Exome sequence read depth methods for identifying copy number changes. *Briefings in bioinformatics*, 16(3):380–392, 2015.

[73] Donna Karolchik, Angela S Hinrichs, Terrence S Furey, Krishna M Roskin, Charles W Sugnet, David Haussler, and W James Kent. The ucsc table browser data retrieval tool. *Nucleic acids research*, 32(suppl_1):D493–D496, 2004.

[74] Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel A Pevzner. Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology*, 37(5):540–546, 2019.

[75] Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel A Pevzner. Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology*, 37(5):540, 2019.

[76] Sergey Koren, Arang Rhie, Brian P Walenz, Alexander T Dilthey, Derek M Bickhart, Sarah B Kingan, Stefan Hiendleder, John L Williams, Timothy PL Smith, and Adam M Phillippy. De novo assembly of haplotype-resolved genomes with trio binning. *Nature biotechnology*, 36(12):1174, 2018.

[77] Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736, 2017.

[78] Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome research*, 27(5):722–736, 2017.

[79] Jonas Korlach. Perspective - Understanding Accuracy in SMRT Sequencing. *Pacific Biosciences*, 2013.

[80] Shunichi Kosugi, Yukihide Momozawa, Xiaoxi Liu, Chikashi Terao, Michiaki Kubo, and Yoichiro Kamatani. Comprehensive evaluation of structural variation detection algorithms for whole genome sequencing. *Genome biology*, 20(1):117, 2019.

[81] Zev N Kronenberg, Ian T Fiddes, David Gordon, Shwetha Murali, Stuart Cantsilieris, Olivia S Meyerson, Jason G Underwood, Bradley J Nelson, Mark JP Chaisson, Max L Dougherty, et al. High-resolution comparative analysis of great ape genomes. *Science*, 360(6393):eaar6343, 2018.

[82] Zev N Kronenberg, Arang Rhie, Sergey Koren, Gregory T Concepcion, Paul Peluso, Katherine M Munson, David Porubsky, Kristen Kuhn, Kathryn A Mueller, Wai Yee Low, et al. Extended haplotype-phasing of long-read de novo genome assemblies using hi-c. *Nature communications*, 12(1):1–10, 2021.

[83] Volodymyr Kuleshov, Dan Xie, Rui Chen, Dmitry Pushkarev, Zhihai Ma, Tim Blauwkamp, Michael Kertesz, and Michael Snyder. Whole-genome haplotyping using long reads and statistical methods. *Nat Biotechnol*, 32(3):261–266, March 2014.

[84] Eric S Lander, Lauren M Linton, Bruce Birren, Chad Nusbaum, Michael C Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 2001.

[85] Ryan M Layer, Colby Chiang, Aaron R Quinlan, and Ira M Hall. Lumpy: a probabilistic framework for structural variant discovery. *Genome biology*, 15(6):1–19, 2014.

[86] Christopher Lee, Catherine Grasso, and Mark F Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 18(3):452–464, 2002.

[87] Hayan Lee, James Gurtowski, Shinjae Yoo, Maria Nattestad, Shoshana Marcus, Sara Goodwin, W Richard McCombie, and Michael Schatz. Third-generation sequencing and the future of genomics. *BioRxiv*, page 048603, 2016.

[88] Samuel Levy, Granger Sutton, Pauline C Ng, Lars Feuk, Aaron L Halpern, Brian P Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F Kirkness, Gennady Denisov, et al. The diploid genome sequence of an individual human. *PLoS biology*, 5(10):e254, 2007.

[89] Harris A Lewin, Gene E Robinson, W John Kress, William J Baker, Jonathan Coddington, Keith A Crandall, Richard Durbin, Scott V Edwards, Félix Forest, M Thomas P Gilbert, et al. Earth biogenome project: Sequencing life for the

future of life. *Proceedings of the National Academy of Sciences*, 115(17):4325–4333, 2018.

[90] Heng Li. A statistical framework for snp calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21):2987–2993, 2011.

[91] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.

[92] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[93] Heng Li, Jonathan M Bloom, Yossi Farjoun, Mark Fleharty, Laura Gauthier, Benjamin Neale, and Daniel MacArthur. A synthetic-diploid benchmark for accurate variant-calling evaluation. *Nature methods*, 15(8):595–597, 2018.

[94] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *bioinformatics*, 25(14):1754–1760, 2009.

[95] Heng Li, Bob Handsaker, Alec Wysoker, et al. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[96] Wentian Li and Jan Freudenberg. Mappability and read length. *Frontiers in Genetics*, 5:381, 2014.

[97] Wentian Li and Jan Freudenberg. Mappability and read length. *Frontiers in genetics*, 5:381, 2014.

[98] Glennis A Logsdon, Mitchell R Vollger, PingHsun Hsieh, Yafei Mao, Mikhail A Liskovykh, Sergey Koren, Sergey Nurk, Ludovica Mercuri, Philip C Dishuck, Arang Rhie, et al. The structure, function, and evolution of a complete human chromosome 8. *bioRxiv*, 2020.

[99] Nicholas J Loman, Joshua Quick, and Jared T Simpson. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature methods*, 12(8):733, 2015.

[100] Oxford Nanopore Technologies Ltd. Medaka, https://github.com/nanoporetech/medaka.

[101] Oxford Nanopore Technologies Ltd. Pomoxis, https://github.com/nanoporetech/pomoxis.

[102] Ruibang Luo, Binghang Liu, Yinlong Xie, Zhenyu Li, Weihua Huang, Jianying Yuan, Guangzhu He, Yanxiang Chen, Qi Pan, Yunjie Liu, et al. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience*, 1(1):2047–217X, 2012.

[103] Ruibang Luo, Chak-Lim Wong, Yat-Sing Wong, Chi-Ian Tang, Chi-Man Liu, Chi-Ming Leung, and Tak-Wah Lam. Exploring the limit of using a deep neural network on pileup data for germline variant calling. *Nature Machine Intelligence*, 2(4):220–227, 2020.

[104] Zhanshan Sam Ma, Lianwei Li, Chengxi Ye, Minsheng Peng, and Ya-Ping Zhang. Hybrid assembly of ultra-long nanopore reads augmented with 10x-genomics contigs: Demonstrated with a human genome. *Genomics*, 2018.

[105] Marcel Martin, Murray Patterson, Shilpa Garg, Sarah Fischer, Nadia Pisanti, Gunnar W Klau, Alexander Schoenhuth, and Tobias Marschall. Whatshap: fast and accurate read-based phasing. *bioRxiv*, page 085050, 2016.

[106] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, et al. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.

[107] Larry Medsker and Lakhmi C Jain. *Recurrent neural networks: design and applications.* CRC press, 1999.

[108] Karen H Miga. Centromere studies in the era of âĂŸtelomere-to-telomereâĂŹgenomics. *Experimental cell research*, 394(2):112127, 2020.

[109] Karen H Miga, Sergey Koren, Arang Rhie, Mitchell R Vollger, Ariel Gershman, Andrey Bzikadze, Shelise Brooks, Edmund Howe, David Porubsky, Glennis A Logsdon, et al. Telomere-to-telomere assembly of a complete human x chromosome. *Nature*, 585(7823):79–84, 2020.

[110] Alla Mikheenko, Andrey Prjibelski, Vladislav Saveliev, Dmitry Antipov, and Alexey Gurevich. Versatile genome assembly evaluation with quast-lg. *Bioinformatics*, 34(13):i142–i150, 2018.

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

[112] Ryan E Mills, E Andrew Bennett, Rebecca C Iskow, and Scott E Devine. Which transposable elements are active in the human genome? *Trends in genetics*, 23(4):183–191, 2007.

[113] Robert K Moyzis, Judy M Buckingham, L Scott Cram, Maria Dani, Larry L Deaven, Myrna D Jones, Julianne Meyne, Robert L Ratliff, and Jung-Rung Wu. A highly conserved repetitive dna sequence,(ttaggg) n, present at the telomeres of human chromosomes. *Proceedings of the National Academy of Sciences*, 85(18):6622–6626, 1988.

[114] Eugene W Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290, 1995.

[115] Eugene W Myers. The fragment assembly string graph. *Bioinformatics*, 21(suppl_2):ii79–ii85, 2005.

[116] Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[117] Sergey Nurk, Sergey Koren, Arang Rhie, Mikko Rautiainen, Andrey V Bzikadze, Alla Mikheenko, Mitchell R Vollger, Nicolas Altemose, Lev Uralsky, Ariel Gershman, et al. The complete sequence of a human genome. *bioRxiv*, 2021.

[118] Sergey Nurk, Brian P Walenz, Arang Rhie, Mitchell R Vollger, Glennis A Logsdon, Robert Grothe, Karen H Miga, Evan E Eichler, Adam M Phillippy, and Sergey Koren. Hicanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads. *Genome research*, 30(9):1291–1305, 2020.

[119] Christopher R O'Donnell, Hongyun Wang, and William B Dunbar. Error analysis of idealized nanopore sequencing. *Electrophoresis*, 34(15):2137–44, aug 2013.

[120] Genome 10K Community of Scientists. Genome 10k: a proposal to obtain whole-genome sequence for 10 000 vertebrate species. *Journal of Heredity*, 100(6):659–674, 2009.

[121] Nathan D Olson et al. precisionfda truth challenge v2: Calling variants from short-and long-reads in difficult-to-map regions. *bioRxiv*, 2020.

[122] B. Paten, D. Earl, N. Nguyen, M. Diekhans, D. Zerbino, and D. Haussler. Cactus: Algorithms for genome multiple sequence alignment. *Genome Research*, 21(9):1512–1528, 2011.

[123] Benedict Paten, Javier Herrero, Kathryn Beal, and Ewan Birney. Sequence progressive alignment, a framework for practical large-scale probabilistic consistency alignment. *Bioinformatics*, 25(3):295–301, 2008.

[124] Murray Patterson, Tobias Marschall, Nadia Pisanti, Leo Van Iersel, Leen Stougie, Gunnar W Klau, and Alexander Schönhuth. Whatshap: weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology*, 22(6):498–509, 2015.

[125] Murray Patterson, Tobias Marschall, Nadia Pisanti, Leo van Iersel, Leen Stougie, Gunnar W Klau, and Alexander Schönhuth. WhatsHap: Weighted haplotype assembly for Future-Generation sequencing reads. *J. Comput. Biol.*, 22(6):498–509, June 2015.

[126] Ryan Poplin et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987, 2018.

[127] David Porubsky, Peter Ebert, Peter A Audano, Mitchell R Vollger, William T Harvey, Pierre Marijon, Jana Ebler, Katherine M Munson, Melanie Sorensen, Arvis Sulovari, et al. Fully phased human genome assembly without parental data using single-cell strand sequencing and long reads. *Nature biotechnology*, 39(3):302–308, 2021.

[128] David Porubsky, Shilpa Garg, Ashley D Sanders, Jan O Korbel, Victor Guryev, Peter M Lansdorp, and Tobias Marschall. Dense and accurate whole-chromosome haplotyping of individual genomes. *Nat. Commun.*, 8(1):1293, November 2017.

[129] Nicholas H Putnam, Brendan L O'Connell, Jonathan C Stites, Brandon J Rice, Marco Blanchette, Robert Calef, Christopher J Troll, Andrew Fields, Paul D Hartley, Charles W Sugnet, et al. Chromosome-scale shotgun assembly using an in vitro method for long-range linkage. *Genome research*, 26(3):342–350, 2016.

[130] Franka J Rang, Wigard P Kloosterman, and Jeroen de Ridder. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome biology*, 19(1):90, 2018.

[131] Tobias Rausch, Thomas Zichner, Andreas Schlattl, Adrian M Stütz, Vladimir Benes, and Jan O Korbel. Delly: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28(18):i333–i339, 2012.

[132] Simon Renny-Byfield and Jonathan F Wendel. Doubling down on genomes: polyploidy and crop plants. *American journal of botany*, 101(10):1711–1725, 2014.

[133] Jared C Roach, Gustavo Glusman, Robert Hubley, Stephen Z Montsaroff, Alisha K Holloway, Denise E Mauldin, Deepak Srivastava, Vidu Garg, Katherine S Pollard, David J Galas, et al. Chromosomal haplotypes by genetic phasing of human families. *The American Journal of Human Genetics*, 89(3):382–397, 2011.

[134] Jared C Roach, Gustavo Glusman, Arian FA Smit, Chad D Huff, Robert Hubley, Paul T Shannon, Lee Rowen, Krishna P Pant, Nathan Goodman, Michael Bamshad, et al. Analysis of genetic inheritance in a family quartet by whole-genome sequencing. *Science*, 328(5978):636–639, 2010.

[135] Michael Roberts, Wayne Hayes, Brian R Hunt, Stephen M Mount, and James A Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.

[136] Oscar L Rodriguez, William S Gibson, Tom Parks, Matthew Emery, James Powell, Maya Strahl, Gintaras Deikus, Kathryn Auckland, Evan E Eichler, Wayne A Marasco, et al. A novel framework for characterizing genomic haplotype diversity

in the human immunoglobulin heavy chain locus. *Frontiers in immunology*, 11, 2020.

[137] Roy Ronen, Christina Boucher, Hamidreza Chitsaz, and Pavel Pevzner. Sequel: improving the accuracy of genome assemblies. *Bioinformatics*, 28(12):i188–i196, 2012.

[138] Kate R Rosenbloom, Cricket A Sloan, Venkat S Malladi, et al. Encode data in the ucsc genome browser: year 5 update. *Nucleic acids research*, 41(D1):D56–D63, 2012.

[139] Jue Ruan. SmartDenovo, https://github.com/ruanjue/smartdenovo.

[140] Jue Ruan and Heng Li. Fast and accurate long-read assembly with wtdbg2. *BioRxiv*, page 530972, 2019.

[141] Jue Ruan and Heng Li. Fast and accurate long-read assembly with wtdbg2. *Nature methods*, 17(2):155–158, 2020.

[142] Frederick Sanger, Steven Nicklen, and Alan R Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the national academy of sciences*, 74(12):5463–5467, 1977.

[143] Valerie A Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A Kitts, Terence D Murphy, Kim D Pruitt, Françoise Thibaud-Nissen, Derek Albracht, et al. Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5):849–864, 2017.

[144] Sven D Schrinner, Rebecca Serra Mari, Jana Ebler, Mikko Rautiainen, Lancelot Seillier, Julia J Reimer, Björn Usadel, Tobias Marschall, and Gunnar W Klau. Haplotype threading: accurate polyploid phasing from long reads. *Genome biology*, 21(1):1–22, 2020.

[145] Jonathan Sebat, B Lakshmi, Jennifer Troge, Joan Alexander, Janet Young, Pär Lundin, Susanne Månér, Hillary Massa, Megan Walker, Maoyen Chi, et al. Large-scale copy number polymorphism in the human genome. *Science*, 305(5683):525–528, 2004.

[146] Fritz J Sedlazeck et al. Accurate detection of complex structural variations using single-molecule sequencing. *Nature methods*, 15(6):461–468, 2018.

[147] Fritz J Sedlazeck, Hayan Lee, et al. Piercing the dark matter: bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics*, page 1, 2018.

[148] Kishwar Shafin et al. Nanopore sequencing and the shasta toolkit enable efficient de novo assembly of eleven human genomes. *Nature biotechnology*, 38(9):1044–1053, 2020.

[149] Kishwar Shafin, Trevor Pesout, Pi-Chuan Chang, Maria Nattestad, Alexey Kolesnikov, Sidharth Goel, Gunjan Baid, Mikhail Kolmogorov, Jordan M Eizenga, Karen H Miga, et al. Haplotype-aware variant calling with pepper-margin-deepvariant enables high accuracy in nanopore long-reads. *Nature Methods*, pages 1–11, 2021.

[150] Jay Shendure and Hanlee Ji. Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145, 2008.

[151] Jared T Simpson, Kim Wong, Shaun D Jackman, Jacqueline E Schein, Steven JM Jones, and Inanç Birol. Abyss: a parallel assembler for short read sequence data. *Genome research*, 19(6):1117–1123, 2009.

[152] Felipe A. SimÃčo, Robert M. Waterhouse, Panagiotis Ioannidis, Evgenia V. Kriventseva, and Evgeny M. Zdobnov. Busco: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*, 31(19):3210–3212, 2015.

[153] AFA Smit, R Hubley, and P Green. Repeatmasker open-4.0. 2013-2015. *URL http://repeatmasker. org*, 2017.

[154] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[155] Peter H Sudmant, Swapan Mallick, Bradley J Nelson, Fereydoun Hormozdiari, Niklas Krumm, John Huddleston, Bradley P Coe, Carl Baker, Susanne Nordenfelt, Michael Bamshad, et al. Global diversity, population stratification, and selection of human copy-number variation. *Science*, 349(6253):aab3761, 2015.

[156] Martin T Swain, Isheng J Tsai, Samual A Assefa, Chris Newbold, Matthew Berriman, and Thomas D Otto. A post-assembly genome-improvement toolkit (pagit) to obtain annotated genomes from contigs. *Nature protocols*, 7(7):1260–1284, 2012.

[157] Ryan Tewhey, Vikas Bansal, Ali Torkamani, Eric J Topol, and Nicholas J Schork. The importance of phase information for human genomics. *Nature Reviews Genetics*, 12(3):215–223, 2011.

[158] The 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

[159] Telomere to-telomere consortium. Ultra-long reads for chm13 genome assembly, https://github.com/nanopore-wgs-consortium/chm13.

[160] Kevin J Travers, Chen-Shan Chin, David R Rank, John S Eid, and Stephen W Turner. A flexible and efficient template format for circular consensus sequencing and snp detection. *Nucleic acids research*, 38(15):e159–e159, 2010.

[161] T. R. Turner, J. D. Hayhurst, D. R. Hayward, W. P. Bultitude, D. J. Barker, J. Robinson, J. A. Madrigal, N. P. Mayor, and S. G. E. Marsh. Single molecule real-time DNA sequencing of HLA genes at ultra-high resolution from 126 International HLA and Immunogenetics Workshop cell lines. *HLA*, 91(2):88–101, 02 2018.

[162] Geraldine A Van der Auwera, Mauricio O Carneiro, Christopher Hartl, et al. From fastq data to high-confidence variant calls: the genome analysis toolkit best practices pipeline. *Current protocols in bioinformatics*, pages 11–10, 2013.

[163] Robert Vaser, Ivan Sović, Niranjan Nagarajan, and Mile Šikić. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome research*, 27(5):737–746, 2017.

[164] Mitchell R Vollger, Glennis A Logsdon, Peter A Audano, Arvis Sulovari, David Porubsky, Paul Peluso, Gregory T Concepcion, Katherine M Munson, Carl Baker, Ashley D Sanders, et al. Improved assembly and variant detection of a haploid human genome using single-molecule, high-fidelity long reads. *BioRxiv*, page 635037, 2019.

[165] Mitchell R. Vollger, Glennis A. Logsdon, Peter A. Audano, Arvis Sulovari, David Porubsky, Paul Peluso, Gregory T. Concepcion, Katherine M. Munson, Carl Baker, Ashley D. Sanders, Diana C.J. Spierings, Peter M. Lansdorp, Michael W. Hunkapiller, and Evan E. Eichler. Improved assembly and variant detection of a haploid human genome using single-molecule, high-fidelity long reads. *bioRxiv*, 2019.

[166] Justin Wagner et al. Benchmarking challenging small variants with linked and long reads. *BioRxiv*, 2020.

[167] Bruce J. Walker, Thomas Abeel, Terrance Shea, Margaret Priest, Amr Abouelliel, Sharadha Sakthikumar, Christina A. Cuomo, Qiandong Zeng, Jennifer Wortman, and Sarah K. et al. Young. Pilon: An integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS ONE*, 9(11):e112963, 2014.

[168] Jing Wang, Leon Raskin, David C Samuels, Yu Shyr, and Yan Guo. Genome measures used for quality control are dependent on gene function and ancestry. *Bioinformatics (Oxford, England)*, 31(3):318–23, feb 2015.

[169] Neil I Weisenfeld, Vijay Kumar, Preyas Shah, Deanna M Church, and David B Jaffe. Direct determination of diploid genome sequences. *Genome research*, 27(5):757–767, 2017.

[170] Neil I Weisenfeld, Shuangye Yin, Ted Sharpe, Bayo Lau, et al. Comprehensive variation discovery in single human genomes. *Nature genetics*, 46(12):1350, 2014.

[171] Aaron M Wenger, Paul Peluso, William J Rowell, Pi-Chuan Chang, Richard J Hall, Gregory T Concepcion, Jana Ebler, Arkarachai Fungtammasan, Alexey Kolesnikov, Nathan D Olson, et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. *Nature biotechnology*, 37(10):1155–1162, 2019.

[172] Aaron M Wenger, Paul Peluso, William J Rowell, Pi-Chuan Chang, Richard J Hall, Gregory T Concepcion, Jana Ebler, Arkarachai Fungtammasan, Alexey Kolesnikov, Nathan D Olson, et al. Highly-accurate long-read sequencing improves variant detection and assembly of a human genome. *bioRxiv*, page 519025, 2019.

[173] Ellen M Wijsman, Joseph H Rothstein, and Elizabeth A Thompson. Multipoint linkage analysis with many multiallelic or dense diallelic markers: Markov chain–monte carlo provides practical approaches for genome scans on general pedigrees. *The American Journal of Human Genetics*, 79(5):846–858, 2006.

[174] Huntington F Willard and John S Waye. Hierarchical order in chromosome-specific human alpha satellite dna. *Trends in Genetics*, 3:192–198, 1987.

[175] Minzhu Xie, Qiong Wu, Jianxin Wang, and Tao Jiang. H-pop and h-popg: heuristic partitioning algorithms for single individual haplotyping of polyploids. *Bioinformatics*, 32(24):3735–3744, 2016.

[176] Justin Zook, Jennifer McDaniel, Hemang Parikh, et al. Reproducible integration of multiple sequencing datasets to form high-confidence snp, indel, and reference calls for five human genome reference materials. *bioRxiv*, 2018.

[177] Justin M Zook, David Catoe, , et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific data*, 3:160025, 2016.

[178] Justin M Zook, Brad Chapman, Jason Wang, David Mittelman, Oliver Hofmann, Winston Hide, and Marc Salit. Integrating human sequence data sets provides a resource of benchmark snp and indel genotype calls. *Nature biotechnology*, 32(3):246–251, 2014.

[179] Justin M Zook, Nancy F Hansen, Nathan D Olson, Lesley Chapman, James C Mullikin, Chunlin Xiao, Stephen Sherry, Sergey Koren, Adam M Phillippy, Paul C Boutros, et al. A robust benchmark for detection of germline large deletions and insertions. *Nature biotechnology*, pages 1–9, 2020.

[180] Justin M Zook, Nancy F Hansen, Nathan D Olson, Lesley M Chapman, James C Mullikin, Chunlin Xiao, Stephen Sherry, Sergey Koren, Adam M Phillippy, Paul C Boutros, et al. A robust benchmark for germline structural variant detection. *BioRxiv*, page 664623, 2019.

[181] Justin M Zook, Jennifer McDaniel, Nathan D Olson, Justin Wagner, Hemang Parikh, Haynes Heaton, Sean A Irvine, Len Trigg, Rebecca Truty, Cory Y McLean, et al. An open resource for accurately benchmarking small variant and reference calls. *Nature biotechnology*, 37(5):561, 2019.